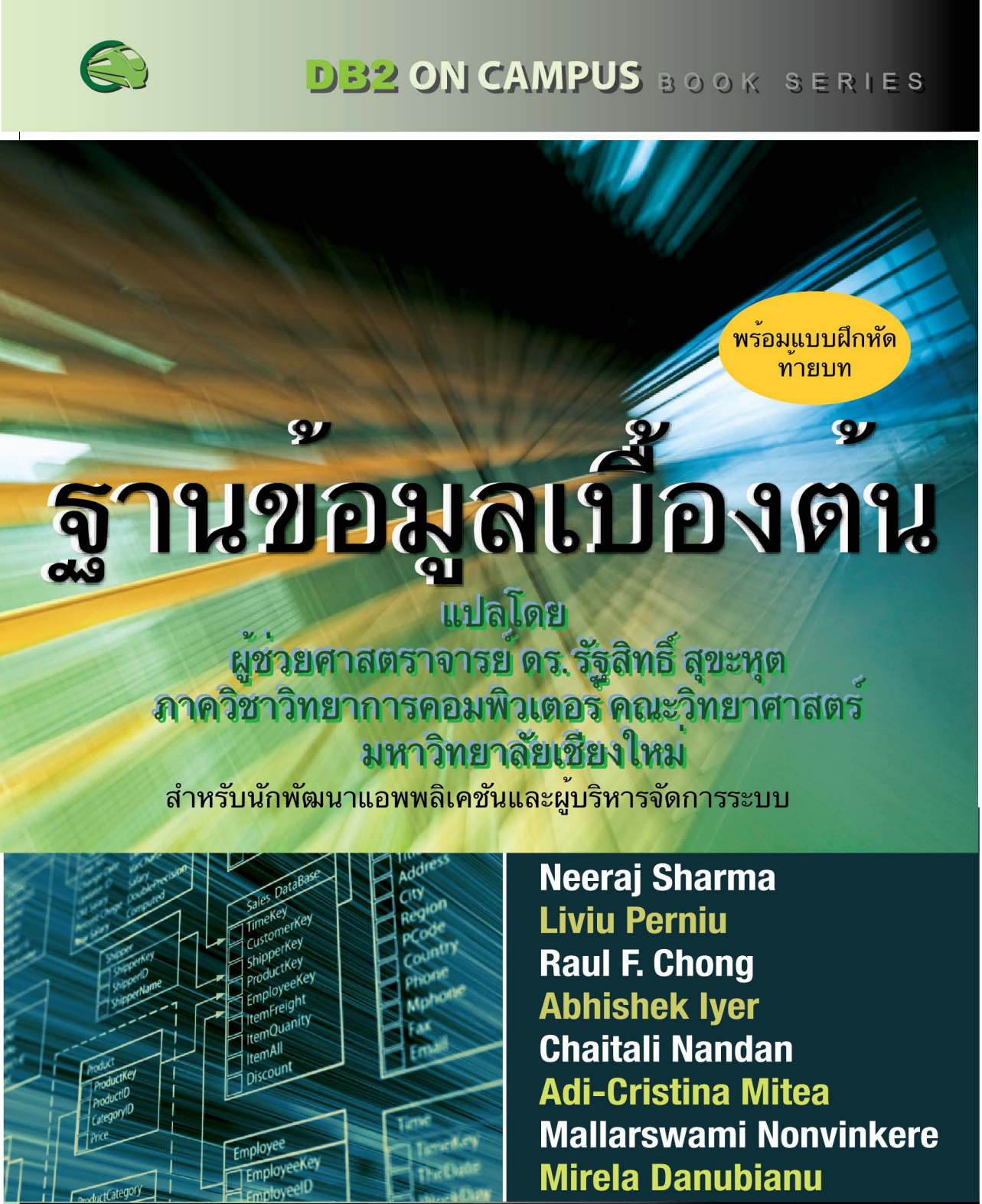


DB2 ON CAMPUS BOOK SERIES

พร้อมแบบฝึกหัดท้ายบท

ฐานข้อมูลเบื้องต้น

แปลโดย

ผู้ช่วยศาสตราจารย์ ดร. รัฐสิทธิ์ สุขะหุต
ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
มหาวิทยาลัยเชียงใหม่

สำหรับนักพัฒนาแอพพลิเคชันและผู้บริหารจัดการระบบ

Neeraj Sharma
Liviu Perniu
Raul F. Chong
Abhishek Iyer
Chaitali Nandan
Adi-Cristina Mitea
Mallarswami Nonvinkere
Mirela Danubianu

BigDataUniversity.com

ฐานข้อมูลเบื้องต้น

Database Fundamentals

A book for the community by the community

โดย: Neeraj Sharma, Liviu Perniu, Raul F. Chong, Abhishek Iyer, Adi-Cristina Mitea, Chaitali Nandan, Mallarswami Nonvinkere, Mirela Danubianu

แปล: ผู้ช่วยศาสตราจารย์ ดร. รัฐสิทธิ์ สุขะหุต
ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
มหาวิทยาลัยเชียงใหม่

พิมพ์ครั้งที่ 1

พิมพ์ครั้งที่ 1 (ตุลาคม 2555)

IBM Canada
8200 Warden Avenue
Markham, ON
L6G 1C7
Canada

หนังสือเล่มนี้ครอบคลุมถึงเนื้อหาซอฟต์แวร์ IBM® DB2® Express-C เวอร์ชั่น 9.7 สำหรับ Linux®, UNIX® และ Windows®.

สาส์นจากผู้บริหาร

ในโลกปัจจุบัน หน่วยงานต่าง ๆ ล้วนต้องหาวิธีบริหารจัดการข้อมูลขององค์กร เพื่อนำมาใช้ให้เกิดประโยชน์สูงสุด ดังนั้น ความรู้และทักษะทางด้านการบริหารจัดการฐานข้อมูลให้เกิดประสิทธิภาพแก่หน่วยงานและองค์กรจึงเป็นสิ่งสำคัญ ด้วยความจำเป็นดังกล่าวนี้เอง มีผลทำให้บุคลากรที่มีทักษะ ความรู้และความเชี่ยวชาญในด้านนี้ เป็นที่ต้องการอย่างยิ่งในตลาดแรงงานและองค์กรธุรกิจต่าง ๆ

บริษัท ไอบีเอ็ม ประเทศไทย สนับสนุนการสร้างเสริมทักษะ ความรู้โดยเฉพาะอย่างยิ่งทางด้านเทคโนโลยีสารสนเทศให้กับสังคมไทยมาอย่างยาวนาน และมีความยินดีเป็นอย่างยิ่งที่ได้มีโอกาสร่วมมือกับคณะวิทยาศาสตร์ มหาวิทยาลัยเชียงใหม่ เพื่อแปลหนังสือ IBM Database Fundamentals ซึ่งมีเนื้อหาเกี่ยวกับการเพิ่มพูนความรู้ ทักษะทางด้านการบริหารจัดการฐานข้อมูลเบื้องต้นเป็นภาษาไทย เพื่อเผยแพร่ความรู้ดังกล่าวสู่สาธารณชน และเปิดโอกาสให้คณาจารย์ นิสิต นักศึกษา และผู้สนใจทั่วไปสามารถนำความรู้จากหนังสือเล่มนี้ไปใช้เป็นส่วนหนึ่งในการเรียนการสอนได้โดยไม่เสียค่าใช้จ่าย

ในความริเริ่มและร่วมมือกันจัดทำหนังสือเล่มนี้ให้สำเร็จลุล่วง บริษัท ไอบีเอ็ม ประเทศไทย ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร. รัฐสิทธิ์ สุขะหุต จากภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยเชียงใหม่ เป็นอย่างสูงรวมทั้งผู้มีส่วนเกี่ยวข้องทุกท่านที่ช่วยกันจัดทำและเรียบเรียงเนื้อหาในบทต่าง ๆ ในหนังสือเล่มนี้ให้แล้วเสร็จสมบูรณ์ และหวังเป็นอย่างยิ่งว่าความรู้ที่ได้จากหนังสือเล่มนี้จะเป็นประโยชน์ในการสร้างเสริมความรู้ ทักษะทางด้านการบริหารจัดการฐานข้อมูลเบื้องต้นแก่คณาจารย์ นักเรียน นิสิต นักศึกษา และผู้สนใจโดยทั่วไปไม่มากก็น้อย

ด้วยความนับถือ

เจษฎา ไกรสิงขร
กรรมการ รองกรรมการผู้จัดการใหญ่ ธุรกิจซอฟต์แวร์
บริษัท ไอบีเอ็ม ประเทศไทย จำกัด

ประกาศ

ข้อมูลนี้ใช้สำหรับผลิตภัณฑ์และบริการที่เสนอในไอบีเอ็ม ประเทศสหรัฐอเมริกา ไอบีเอ็มอาจไม่เสนอสินค้า บริการ หรือคุณลักษณะที่กล่าวถึงในเอกสารนี้ในประเทศอื่นๆ ทั้งนี้ หากท่านมีข้อสงสัย เราขอแนะนำให้ท่านขอคำปรึกษาเกี่ยวกับผลิตภัณฑ์และบริการจากตัวแทนของบริษัทไอบีเอ็มในประเทศของท่าน การอ้างอิงใดๆที่เกี่ยวข้องกับผลิตภัณฑ์หรือโปรแกรมของไอบีเอ็มหรือบริการอื่นๆ ไม่ได้มีเจตนาว่าหมายถึงหรือบอกเป็นนัยว่าสินค้า โปรแกรม หรือ บริการดังกล่าวนั้นจะถูกนำมาใช้ ในทางตรงกันข้าม สินค้า โปรแกรม หรือบริการที่มีหน้าที่การใช้งานที่ใกล้เคียง และไม่ละเมิดสิทธิทรัพย์สินทางปัญญาของไอบีเอ็มอาจถูกนำมาใช้แทน อย่างไรก็ตาม ผู้ใช้งานจำเป็นต้องประเมินและตรวจสอบการทำงานของผลิตภัณฑ์ โปรแกรม หรือบริการที่ไม่ได้เป็นของไอบีเอ็มนั้น ๆ ด้วยตัวเอง

บริษัท ไอบีเอ็ม อาจมีการจดสิทธิบัตรหรือกำลังยื่นขอจดสิทธิบัตรกับสินค้า โปรแกรม หรือบริการที่ถูกอ้างถึงในหนังสือเล่มนี้ ดังนั้น การแก้ไขเนื้อหาใดๆ ในหนังสือเล่มนี้จะไม่ได้ทำให้คุณได้รับสิทธิใดๆทางด้านสิทธิบัตรต่างๆ แต่อย่างใด อย่างไรก็ตาม ท่านอาจส่งข้อสงสัยหรือคำถามเกี่ยวกับเรื่องสิทธิหรือสิทธิบัตรไปที่

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

สำหรับคำถามเกี่ยวกับใบอนุญาตที่เกี่ยวกับ Double-byte character set (DBCS) ขอให้ท่านติดต่อแผนกทรัพย์สินทางปัญญาของบริษัทไอบีเอ็มในประเทศของท่านโดยตรง หรือส่งคำถามไปที่:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
3-2-12, Roppongi, Minato-ku, Tokyo 106-8711

สำหรับรายละเอียดต่อไปนี้ไม่สามารถใช้กับประเทศสหราชอาณาจักรหรือประเทศอื่นๆที่ข้อบัญญัติไม่สอดคล้องกับกฎหมายท้องถิ่น บริษัท INTERNATIONAL BUSINESS MACHINES CORPORATION นำเสนอเนื้อหาในหนังสือเล่มนี้ตามที่นำเสนอโดยไม่มีการรับประกันใดๆ ทั้งโดยตรงหรือโดยนัยซึ่งรวมถึงแต่ไม่จำกัดเฉพาะการรับประกันโดยนัย การละเมิด ซื้อขาย หรือสำหรับวัตถุประสงค์เฉพาะด้าน บางรัฐในสหรัฐอเมริกาไม่อนุญาตให้มีการแจ้งขอยกเว้นในทางตรงและโดยนัยสำหรับบางธุรกรรม ดังนั้น คำชี้แจงนี้อาจไม่ได้ใช้กับท่าน

ข้อมูลที่รวบรวมในหนังสือเล่มนี้ อาจมีความไม่ถูกต้องทางด้านเทคนิค หรือพิมพ์ผิดพลาดบ้าง ดังนั้น จะมีการแก้ไขปรับปรุงอยู่เป็นระยะๆ และนำเสนอการแก้ไขนั้นๆ ในการจัดพิมพ์ครั้งต่อไป อย่างไรก็ตาม ไอบีเอ็มอาจจะทำการปรับปรุงและ/หรือเปลี่ยนแปลงผลิตภัณฑ์และ/หรือโปรแกรมที่อธิบายไว้ในหนังสือเล่มนี้ในเวลาใดก็ได้ โดยไม่ต้องแจ้งให้ทราบล่วงหน้า

การอ้างอิงใดๆถึงเว็บไซท์อื่นๆที่ไม่ได้เป็นของไอบีเอ็มในหนังสือเล่มนี้ มีขึ้นเพื่อเพิ่มความสะดวกแต่ไม่ได้หมายความว่า ไอบีเอ็มจะรับรองเนื้อหาต่างๆในเว็บไซท์เหล่านั้นแต่อย่างใด ทั้งนี้ เนื่องจากเนื้อหาในเว็บไซท์เหล่านั้นไม่ได้เป็นของไอบีเอ็ม ผู้ใช้งานที่ใช้เนื้อหาในเว็บไซท์เหล่านั้น จำเป็นต้องใช้วิจารณญาณและแบกรับความเสี่ยงเอง ไอบีเอ็มอาจใช้หรือเผยแพร่ข้อมูลใดๆที่ท่านนำเสนอมาให้ซึ่งไอบีเอ็มพิจารณาแล้วว่าเหมาะสม โดยไม่ได้หมายความหรือแสดงนัยว่าจะมีข้อผูกมัดใดๆกับท่าน

โปรแกรมและเอกสารต่างๆที่มีลิขสิทธ์และได้ถูกอ้างถึงในหนังสือเล่มนี้ ได้ถูกนำเสนอภายใต้เงื่อนไขข้อตกลงและเงื่อนไขสัญญาลูกค้าของไอบีเอ็ม หรือเงื่อนไขข้อตกลงระหว่างประเทศที่เกี่ยวข้องกับลิขสิทธ์ของไอบีเอ็มหรือเทียบเท่า

ข้อมูลใดๆที่เกี่ยวข้องกับประสิทธิภาพการทำงานได้ถูกนำเสนอมาจากสภาพแวดล้อมการทำงานที่ถูกควบคุม ดังนั้นผลลัพธ์ที่ได้จากสภาพแวดล้อมอื่นอาจแตกต่างกันอย่างมีนัยสำคัญ การวัดค่าบางอย่างที่ได้วัดมาจากระดับที่ระบบมีการพัฒนา จะไม่มีการรับประกันว่าค่าที่วัดได้จะเหมือนกันกับระบบอื่น นอกจากนี้ การวัดค่าในบางกรณีอาจได้มาจากการคาดการณ์ซึ่งผลลัพธ์ที่ได้อาจแตกต่างกันไป ผู้ใช้งานหนังสือเล่มนี้ควรตรวจสอบข้อมูลที่สามารถใช้งานได้กับสภาพแวดล้อมการทำงานของท่าน ข้อมูลหรือข่าวสารอื่นใดที่เกี่ยวข้องกับผลิตภัณฑ์ที่ไม่ใช่ของไอบีเอ็ม และเป็นข้อมูลที่ได้มาจากเจ้าของหรือผู้จำหน่ายผลิตภัณฑ์เหล่านั้น ไอบีเอ็มไม่ได้ทำการทดสอบผลิตภัณฑ์เหล่านั้นแต่อย่างใด และไม่สามารถรับรองประสิทธิภาพการทำงาน ความถูกต้อง ความเข้ากันได้หรือรับประกันเรื่องใดๆที่เกี่ยวข้องกับผลิตภัณฑ์เหล่านั้นที่ไม่ได้เป็นของไอบีเอ็ม ท่านสามารถสอบถามไปยังเจ้าของหรือผู้จำหน่ายผลิตภัณฑ์เหล่านั้นโดยตรง

ข้อมูลใดๆที่เกี่ยวข้องกับทิศทางของไอบีเอ็มในอนาคตหรือความตั้งใจในการทำกิจกรรมใดๆที่กล่าวถึงในหนังสือเล่มนี้ อาจมีการเปลี่ยนแปลงหรือเพิกถอนโดยไม่ต้องแจ้งให้ทราบล่วงหน้า และข้อมูลดังกล่าวอาจถูกนำเสนอเพื่อแสดงถึงเป้าหมายของไอบีเอ็มเท่านั้น การอ้างถึงตัวอย่างของข้อมูลหรือรายงานที่ใช้ในหนังสือเล่มนี้ บางครั้งจำเป็นต้องอ้างถึงชื่อบุคคล ชื่อบริษัท ชื่อสินค้า หรือชื่อผลิตภัณฑ์ เพื่อทำให้การนำเสนอตัวอย่างทำได้อย่างสมบูรณ์ที่สุดเท่าที่จะเป็นไปได้ อย่างไรก็ตาม ชื่อดังกล่าวล้วนเป็นชื่อที่สมมติขึ้น ซึ่งอาจบังเอิญไปคล้ายคลึงกับชื่อหรือที่อยู่ที่มีอยู่จริง ดังนั้น ให้ถือว่าการเกี่ยวโยงนั้นล้วนเป็นความบังเอิญทั้งสิ้น

ลิขสิทธิ์

ข้อมูลที่นำเสนอในหนังสือเล่มนี้ประกอบด้วยโปรแกรมตัวอย่างจากภาษาต้นฉบับ ซึ่งอธิบายเทคนิคการเขียนโปรแกรมบนระบบปฏิบัติการประเภทต่างๆ ท่านอาจจะคัดลอก ดัดแปลงแก้ไข และเผยแพร่โปรแกรมตัวอย่างเหล่านี้ในรูปแบบต่างๆโดยไม่ต้องเสนอค่าตอบแทนให้กับไอบีเอ็มภายใต้วัตถุประสงค์ของการพัฒนาการใช้งานเพื่อการตลาด หรือเพื่อการเผยแพร่โปรแกรมที่สอดคล้องกับอินเตอร์เฟซการเขียนโปรแกรมประยุกต์บนระบบปฏิบัติการที่โปรแกรมตัวอย่างนั้นถูกเขียนขึ้น ตัวอย่างเหล่านี้ยังไม่ได้รับการทดสอบอย่างละเอียดภายใต้เงื่อนไขใดๆ ไอบีเอ็มจึงไม่สามารถรับประกันหรือรับรองความน่าเชื่อถือของโปรแกรมเหล่านี้ นอกจากนั้นโปรแกรมตัวอย่างที่นำเสนอในหนังสือเล่มนี้ เป็นลักษณะให้ “ตามที่เป็น” โดยไม่มีการรับประกันใดๆ ไอบีเอ็มจะไม่รับผิดชอบต่อความเสียหายใดๆที่เกิดขึ้นจากการใช้งานโปรแกรมตัวอย่างเหล่านี้ทั้งสิ้น การอ้างอิงในหนังสือเล่มนี้กับผลิตภัณฑ์หรือบริการของไอบีเอ็มใดๆ ไม่ได้มีความหมายหรือมีนัยว่าไอบีเอ็ม จะนำเสนอผลิตภัณฑ์หรือบริการนั้นๆในทุกประเทศที่ไอบีเอ็มดำเนินธุรกิจ ในกรณีที่ท่านใช้งานหนังสือเล่มนี้ในลักษณะของซอฟต์ไฟล์ ภาพหรือรูปประกอบบางชิ้นอาจไม่ปรากฏขึ้น

เครื่องหมายการค้า

บริษัท ไอบีเอ็ม โลโก้ของไอบีเอ็ม และ ibm.com เป็นเครื่องหมายการค้าจดทะเบียนของบริษัท International Business Machines Corp. ซึ่งได้มีการจดทะเบียนตามเงื่อนไขต่างๆทั่วโลก ชื่อผลิตภัณฑ์และบริการอื่นๆ อาจเป็นเครื่องหมายทางการค้าของไอบีเอ็มหรือบริษัทอื่นๆ ข้อมูลเกี่ยวกับเครื่องหมายการค้าของบริษัท ไอบีเอ็ม สามารถตรวจสอบได้ภายใต้เนื้อหาในหัวข้อชื่อ "Copyright and trademark information" จากเว็บไซต์ www.ibm.com/legal/us/en/copytrade.shtml

Java และเครื่องหมายการค้าที่ใช้ภาษา Java เป็นเครื่องหมายการค้าของ Sun Microsystems, Inc. ในสหรัฐอเมริกา ในประเทศอื่น หรือทั้งสองกรณี

Microsoft และ Windows เป็นเครื่องหมายการค้าของ Microsoft Corporation ในสหรัฐอเมริกา ในประเทศอื่น หรือทั้งสองกรณี

Linux เป็นเครื่องหมายจดทะเบียนการค้าของ Linus Torvalds ในสหรัฐอเมริกา ในประเทศอื่น หรือทั้งสองกรณี

UNIX เป็นเครื่องหมายจดทะเบียนการค้าของ The Open Group ในสหรัฐอเมริกาและในประเทศอื่น ๆ

ชื่อบริษัท ผลิตภัณฑ์ หรือชื่อบริการอื่นๆ อาจเป็นเครื่องหมายการค้าหรือเครื่องหมายบริการของบริษัทอื่นๆ

สารบัญ

คำนิยม

ผู้ช่วยศาสตราจารย์ ดร.รัฐสิทธิ์ สุขะหุต จากภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยเชียงใหม่เป็นผู้ที่มีความรู้และความเชี่ยวชาญทางด้านฐานข้อมูลและระบบการจัดการฐานข้อมูลในระดับต้นๆของมหาวิทยาลัยเชียงใหม่ บัดนี้อาจารย์ได้ทุ่มเทเวลาในการแปลหนังสือ Database Fundamentals ซึ่งเป็นลิขสิทธิ์ของบริษัทไอบีเอ็ม ประเทศแคนาดา จากภาคภาษาอังกฤษมาเป็นภาคภาษาไทยในชื่อว่า 'ฐานข้อมูลเบื้องต้น' ซึ่งหนังสือเล่มนี้จะเป็นหนังสือคู่มือและเอกสารอ้างอิงที่มีประโยชน์ยิ่งสำหรับนักพัฒนาแอพพลิเคชันและผู้จัดการระบบ

หนังสือ Database Fundamentals เป็นหนังสือที่วางโครงสร้างเนื้อหาและคำอธิบายประกอบไว้ดีมาก และผู้ช่วยศาสตราจารย์ ดร.รัฐสิทธิ์ สุขะหุต ได้พยายามแปลโดยใช้ภาษาที่ช่วยให้ผู้อ่านชาวไทยเข้าใจได้ง่ายและสามารถเข้าถึงแก่นความรู้ที่หนังสือได้นำเสนอได้ลึกซึ้งมากขึ้นยิ่งกว่าการอ่านจากต้นฉบับภาษาอังกฤษ หนังสือเล่มนี้ได้เริ่มต้นอธิบายแนวคิดพื้นฐานและวิวัฒนาการของฐานข้อมูล ระบบการจัดการฐานข้อมูล และแบบจำลองข้อมูล และได้ขยายความอธิบายถึงภาพรวมของแบบจำลองข้อมูลเชิงสัมพันธ์ตลอดจนทฤษฎีทางคณิตศาสตร์ที่เป็นพื้นฐาน แนวคิดในการสร้างฐานข้อมูลและการออกแบบฐานข้อมูลเชิงสัมพันธ์ ระเบียบปฏิบัติและฟังก์ชันภาษา SQL และการใช้งานในโปรแกรมประยุกต์ ภาษาสำหรับการสอบถามข้อมูลสำหรับ XML และความปลอดภัยของฐานข้อมูล ซึ่งจะช่วยให้ผู้อ่านได้ใช้โปรแกรมฐานข้อมูลได้อย่างถูกต้องและมีประสิทธิภาพ นอกจากนี้หนังสือยังได้กล่าวถึงแนวโน้มเทคโนโลยีและฐานข้อมูล โดยเฉพาะการประมวลผลแบบกลุ่มเมฆและการพัฒนาซอฟต์แวร์บนโทรศัพท์เคลื่อนที่ซึ่งจะช่วยให้ผู้ใช้ได้รับความสะดวกในการใช้ประโยชน์จากเทคโนโลยีสารสนเทศและการสื่อสารเพิ่มมากขึ้น โดยภาพรวมแล้วอาจกล่าวได้ว่าหนังสือเล่มนี้ได้รวบรวมคำอธิบายเกี่ยวกับฐานข้อมูลได้อย่างกระชับและครบถ้วนสมบูรณ์

ในโอกาสนี้ คณะวิทยาศาสตร์จึงใคร่ขอแสดงความยินดีและชื่นชมต่อผู้ช่วยศาสตราจารย์ ดร.รัฐสิทธิ์ สุขะหุต ที่ได้กรุณาแปลหนังสือ Database Fundamentals นอกจากจะเป็นหนังสือคู่มือสำหรับนักพัฒนาแอพพลิเคชันและผู้จัดการระบบ ยังจะเป็นหนังสือสำหรับนักศึกษาและผู้สนใจทั่วไปได้อ่านเพื่อการเรียนรู้ รวมทั้งสถานศึกษาสามารถใช้เป็นเอกสารประกอบการเรียนการสอนในเรื่องฐานข้อมูลพื้นฐานต่อไป

รองศาสตราจารย์ ดร. สัมพันธ์ สิงหราชวราพันธ์
คณบดีคณะวิทยาศาสตร์
มหาวิทยาลัยเชียงใหม่

คำนำ

ปัจจุบันเทคโนโลยีทางด้านฐานข้อมูลเชิงสัมพันธ์มีการพัฒนาที่ก้าวหน้าเป็นอย่างมาก ทั้งประสิทธิภาพในการรองรับข้อมูลขนาดใหญ่ การรองรับชนิดข้อมูลต่างๆ และความสามารถในการทำงานร่วมกับเครื่องมือออกแบบและพัฒนาบนแพลตฟอร์มที่หลากหลาย ประกอบกับเครื่องมือที่มีมาให้ในผลิตภัณฑ์ระบบฐานข้อมูลต่างๆ เอื้ออำนวยให้นักออกแบบและพัฒนาฐานข้อมูลสามารถทำงานกับระบบฐานข้อมูลได้อย่างสะดวกรวดเร็วมากยิ่งขึ้น

หนังสือฐานข้อมูลเบื้องต้นเล่มนี้อธิบายถึงแนวคิดพื้นฐานที่สำคัญของฐานข้อมูลเชิงสัมพันธ์ ทฤษฎีพื้นฐาน ขั้นตอนการวิเคราะห์ ออกแบบ รวมถึงการพัฒนาชุดคำสั่งและภาษาที่ใช้ในการจัดการกับฐานข้อมูล ซึ่งหนังสือเล่มนี้ได้อธิบายอย่างเป็นขั้นตอนพร้อมทั้งการยกตัวอย่างประกอบเพื่อให้เห็นภาพของการนำไปใช้งานและการพัฒนามากยิ่งขึ้น

ผู้แปลเห็นว่าหนังสือเล่มนี้เป็นหนังสือมีประโยชน์ต่อวงการวิชาการที่น่าจะนำมาเผยแพร่ และมีประโยชน์ต่อสาขาเทคโนโลยีสารสนเทศ สาขาวิทยาการคอมพิวเตอร์ สาขาวิศวกรรมคอมพิวเตอร์ และสาขาอื่นๆที่เกี่ยวข้อง เหมาะสำหรับนักศึกษา คณาจารย์และผู้ที่สนใจที่จะศึกษาค้นคว้าเพิ่มเติม และเป็นข้อมูลอ้างอิงทางวิชาการเกี่ยวกับเทคโนโลยีฐานข้อมูล ผู้แปลหวังว่าหนังสือเล่มนี้จะเป็นประโยชน์และเป็นแหล่งข้อมูลความรู้ที่สำคัญต่อวงการไอทีของประเทศไทยต่อไป

ผู้ช่วยศาสตราจารย์ ดร. รัฐสิทธิ์ สุขะหุต
ภาควิชาวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ มหาวิทยาลัยเชียงใหม่

บทนำ

ด้วยความท้าทายในโลกปัจจุบันที่เทคโนโลยีมีการเปลี่ยนแปลงอย่างรวดเร็วและทำให้ท่านต้องพัฒนาความรู้ทักษะอยู่ตลอดเวลา DB2® on Campus Book Series ได้มีการพัฒนาอย่างต่อเนื่องเพื่อช่วยให้ท่านเสริมสร้างความรู้ทักษะให้ทันกับเทคโนโลยีเหลานี้

หนังสือเล่มนี้จะช่วยให้ผู้สนใจความรู้ทางด้านฐานข้อมูล สามารถเข้าใจแนวคิดทางด้านฐานข้อมูล ด้วยการผสมผสานเนื้อหาทั้งในแนวกว้างและแนวลึกเพื่อให้เป็นประโยชน์สูงสุดต่อผู้อ่าน

ใครควรอ่านหนังสือเล่มนี้?

หนังสือเล่มนี้เหมาะสำหรับผู้ที่สนใจความรู้พื้นฐานทางด้านเทคโนโลยีฐานข้อมูล นักพัฒนาโปรแกรม ผู้ดูแลระบบฐานข้อมูล คณาจารย์ นักเรียน นิสิต นักศึกษา รวมทั้งผู้สนใจทั่วไป

หนังสือเล่มนี้มีโครงสร้างอย่างไร?

หนังสือเล่มนี้แบ่งออกเป็นบทต่าง ๆ โดยในบทที่ 1 อธิบายถึงแนวคิดพื้นฐานของฐานข้อมูลและเนื้อหาเกี่ยวกับโมเดล บทที่ 2 เป็นเนื้อหาที่ครอบคลุมเกี่ยวกับโมเดลข้อมูลเชิงสัมพันธ์ บทที่ 3 และ 4 อธิบายการสร้างแบบจำลองเชิงแนวคิดและการออกแบบฐานข้อมูลเชิงสัมพันธ์ ในบทที่ 5, 6 และ 7 มุ่งเน้นเรื่อง SQL บทที่ 8 จะเน้นเรื่องการจัดเก็บข้อมูลแบบ XML และสืบค้นผ่าน SQL และ XQuery บทที่ 9 เน้นในเรื่องด้านความปลอดภัยของฐานข้อมูล และในบทที่ 10 ของหนังสือเล่มนี้จะสรุปภาพรวมของเทคโนโลยีต่างๆ และโปรแกรมที่เกี่ยวข้องที่เป็นที่นิยมในธุรกิจต่าง ๆ ในปัจจุบัน ทั้งนี้แบบฝึกหัดและคำถามท้ายบทจะอยู่ในตอนท้ายของแต่ละบท รวมทั้งคำเฉลยซึ่งอยู่ในภาคผนวก A

หนังสือเพื่อประโยชน์ต่อสาธารณชน

หนังสือเล่มนี้ แปลจากหนังสือต้นฉบับที่เป็นภาษาอังกฤษซึ่งจัดทำโดยอาจารย์จากมหาวิทยาลัย นักศึกษา ผู้เชี่ยวชาญ (รวมทั้งพนักงานไอบีเอ็ม) ผู้ทรงคุณวุฒิและผู้มีความรู้จากทั่วโลกได้มีส่วนร่วมในการร่วมกันพัฒนาเนื้อหาหนังสือเล่มนี้ และเปิดโอกาสให้บุคคลทั่วไปสามารถดาวน์โหลดหนังสือเล่มนี้ไปใช้ศึกษา เพิ่มพูนความรู้ได้ โดยไม่เสียค่าใช้จ่าย ในกรณีที่ท่านมีข้อเสนอแนะเนื้อหาใหม่ ๆ เพื่อเพิ่มเนื้อหาที่มีอยู่ในหนังสือต้นฉบับหรือสนใจแปลหนังสือเล่มนี้เป็นภาษาอื่น กรุณาส่งอีเมล์เป็นภาษาอังกฤษไปที่ db2univ@ca.ibm.com ด้วยหัวข้อ "Database fundamentals book feedback"

มาตรฐาน

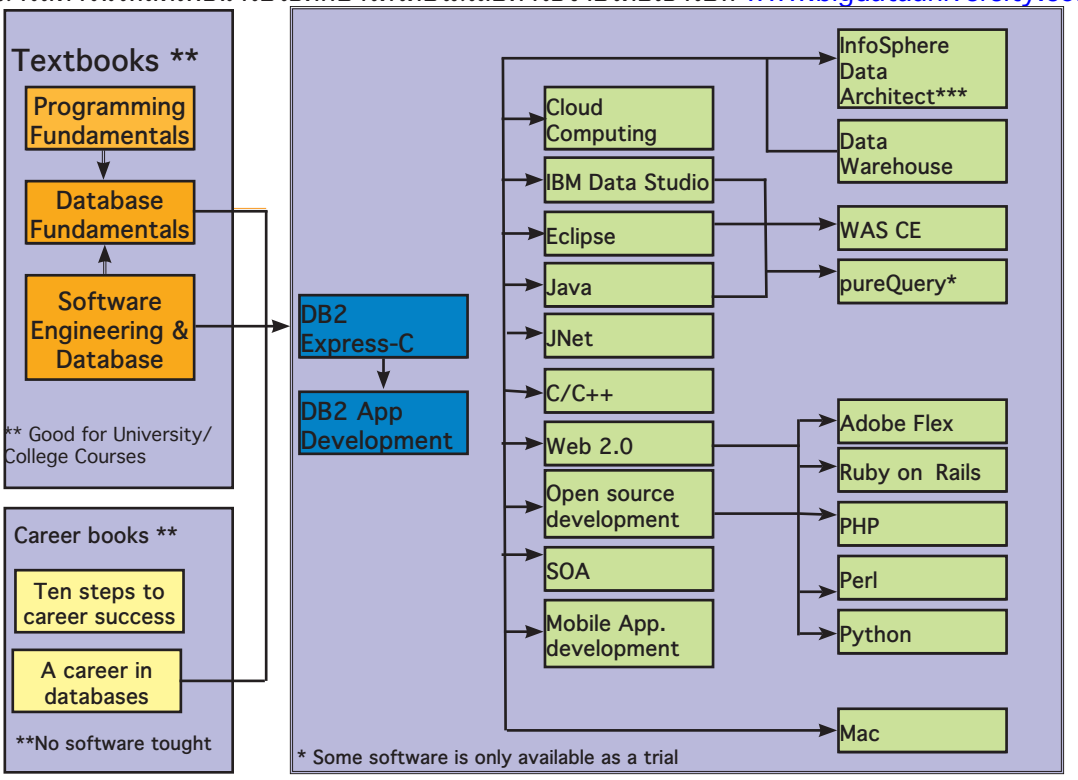
ตัวอย่างต่างๆ ของคำสั่ง SQL และโค๊ดจะรวมอยู่ในหนังสือเล่มนี้ตลอดทั้งเล่ม คำเฉพาะจะเป็นตัวพิมพ์ใหญ่หนา ตัวอย่างเช่น NULL หมายถึงสถานะที่เป็นค่าว่าง คำสั่งจะเป็นตัวพิมพ์เล็กหนา ตัวอย่างเช่น dir เป็นคำสั่งแสดงรายชื่อไฟล์และไดเรกทอรีย่อยทั้งหมดในวินโดวส์ คำสั่ง SQL จะเป็นตัวพิมพ์ใหญ่หนา ตัวอย่างเช่น ใช้คำสั่ง SELECT เป็นคำสั่งที่ใช้ในการสืบค้นข้อมูลจากตาราง ชื่อออบเจ็กต์จะเป็นตัวเอียงหนา ตัวอย่างเช่น flight และตัวเอียงจะใช้สำหรับชื่อตัวแปรในรูปแบบของคำสั่งหรือข้อความ ถ้าชื่อตัวแปรมีมากกว่าหนึ่งคำ จะเชื่อมชื่อตัวแปรเข้าด้วยกันด้วยขีดล่าง ตัวอย่างเช่น CREATE TABLE table_name

แนวทางการใช้ประโยชน์จากหนังสือเล่มนี้

ผู้จัดทำแนะนำให้ท่านอ่านหนังสือต่อไปนี้ประกอบเพิ่มเติมเพื่อใช้เป็นข้อมูลเสริมเกี่ยวกับหัวข้อต่างๆ ในหนังสือเล่มนี้

- Getting started with DB2 Express-C
- Getting started with InfoSphere Data Architect
- Getting started with data warehousing
- Getting started with DB2 application development

แผนภูมิด้านล่างนี้แสดงให้เห็นโครงสร้างของหนังสืออีบุ๊คทั้งหมดที่อยู่ใน DB2 on Campus book series ซึ่งท่านสามารถดาวน์โหลดเพื่อนำไปใช้ศึกษาได้โดยไม่เสียค่าใช้จ่ายโดยเข้าไปที่ www.bigdatauniversity.com



The DB2 on Campus book series

เกี่ยวกับผู้เขียน ต้นฉบับภาษาอังกฤษ

Neeraj Sharma จบการศึกษาระดับปริญญาตรีในสาขาวิศวกรรมอิเล็กทรอนิกส์และการสื่อสาร ปริญญาโทในด้านระบบซอฟต์แวร์และปัจจุบันเป็นผู้เชี่ยวชาญระดับสูงด้านไอทีที่ Dynamic Warehousing Center of Competency ณ ห้องปฏิบัติการซอฟต์แวร์ ไอบีเอ็ม อินเดีย หน้าที่หลักคือการออกแบบ กำหนด ตั้งค่าและดำเนินการกับคลังข้อมูลขนาดใหญ่ระหว่างโดเมนในอุตสาหกรรมต่างๆ ซึ่งดำเนินการตามแนวคิด Proof of Concept (POC) และทดสอบประสิทธิภาพตามคำร้องขอของลูกค้า

Liviu Perniu เป็นศาสตราจารย์ใน Automation Department at Transilvania University of Brasov ประเทศโรมาเนีย ปัจจุบันท่านสอนหลักสูตรที่เกี่ยวข้องกับความต้องการข้อมูล การวิเคราะห์ และการสร้างแบบจำลอง และได้รับรางวัล IBM 2006 Faculty Award ซึ่งเป็นส่วนหนึ่งของโครงการ Eclipse Innovation Awards

Raul F. Chong เป็นผู้จัดการโครงการ DB2 on Campus ที่ IBM Toronto Laboratory และเป็นผู้สอนทางด้านเทคนิคของ DB2 หน้าที่หลักคือเผยแพร่ความรู้เกี่ยวกับ DB2 แก่สาธารณชนต่างๆทั่วโลก Raul เริ่มต้นทำงานใน IBM ตั้งแต่ปี 1997 และดำรงตำแหน่งต่างๆสำคัญในบริษัทหลายตำแหน่ง เคยเป็นที่ปรึกษาด้าน DB2 ที่ให้การช่วยเหลือแก่บริษัทคู่ค้าของ IBM ในการโอนย้ายฐานข้อมูลจากระบบฐานข้อมูลอื่นๆที่มีการจัดการในลักษณะของฐานข้อมูลเชิงสัมพันธ์ไปยัง DB2 นอกจากนั้น Raul ยังเคยช่วยแก้ปัญหา DB2 บน OS/390®, z/OS®, Linux®, UNIX® และแพลตฟอร์มวินโดว์ สอนการอบรมเชิงปฏิบัติการ DB2 อีกทั้งได้ตีพิมพ์บทความเป็นจำนวนมาก และมีส่วนร่วมในการออกข้อสอบ DB2 Raul ได้สรุปประสบการณ์ DB2 ในหลายปีที่ผ่านมาไว้ในหนังสือ Understanding DB2 - Learning Visually with Examples 2nd Edition (ISBN-10: 0131580183) ที่ท่านเป็นหัวหน้าทีมในการเขียน เป็นผู้เขียนร่วมในหนังสือ DB2 SQL PL Essential Guide for DB2 UDB on Linux, UNIX, Windows, i5/OS, and z/OS (ISBN 0131477005) และเป็นหัวหน้าโครงการและผู้ร่วมเขียนในหนังสือหลายเล่มของ DB2 on Campus

Abhishek Iyer เป็นวิศวกรที่ Warehousing Center of Competency ที่ห้องปฏิบัติการซอฟต์แวร์ของไอบีเอ็ม อินเดีย หน้าที่หลักคือสร้างการพิสูจน์แนวคิดและดำเนินการวัดประสิทธิภาพเมื่อลูกค้าร้องขอ รวมถึงมีความเชี่ยวชาญในการดำเนินการด้านคลังข้อมูลและการทำเหมืองข้อมูล Abhishek จบการศึกษาระดับปริญญาตรีในสาขาวิทยาการคอมพิวเตอร์

Chaitali Nandan เป็นวิศวกรซอฟต์แวร์ในทีม DB2 Advanced Technical Support ที่ห้องปฏิบัติการซอฟต์แวร์ ไอบีเอ็ม อินเดีย หน้าที่หลักคือ ช่วยแก้ไขเบื้องต้นและให้การสนับสนุนลูกค้าองค์กรเกี่ยวกับ DB2 Chaitali จบปริญญาตรีทางด้านวิศวกรรมเทคโนโลยีสารสนเทศ

Adi-Cristina Mitea เป็นศาสตราจารย์ที่ภาควิชาวิทยาการคอมพิวเตอร์ “Hermann Oberth” คณะวิศวกรรมศาสตร์ “Lucian Blaga” University of Sibiu ประเทศโรมาเนีย ปัจจุบันสอนในสาขาวิชาฐานข้อมูล distributed systems, parallel and distributed algorithms, fault tolerant systems และอื่น ๆ รวมถึงได้ทำงานวิจัยเกี่ยวกับในเรื่องดังกล่าวด้วย Adi-Cristina จบปริญญาตรีและปริญญาเอกทางด้านวิทยาการคอมพิวเตอร์

Mallarswami Nonvinkere เป็นผู้เชี่ยวชาญด้าน pureXML® ที่ห้องปฏิบัติการซอฟต์แวร์ ไอบีเอ็ม อินเดียและทำงานในทีม DB2 pureXML ในอินเดีย ปัจจุบันช่วยสนับสนุนลูกค้าของไอบีเอ็ม และผู้พัฒนาซอฟต์แวร์ โดยช่วยให้พวกเขาเข้าใจการใช้เทคโนโลยี pureXML และพัฒนางานประสิทธิภาพสูงโดยใช้ XML นอกจากนั้น Mallarswami ยังเคยเป็นผู้บรรยายในการประชุมระหว่างประเทศรวมทั้ง IDUG Australasia, IDUG India, งาน IMTC เป็นต้น

Mirela Danubianu เป็นอาจารย์ที่ Stefan cel Mare University of Suceava, Faculty of Electrical Engineering and Computer Science Mirela ได้รับปริญญาโทสาขาวิทยาการคอมพิวเตอร์ที่มหาวิทยาลัย Craiova (1985 – Automatizations and Computers) และทางด้านเศรษฐศาสตร์ที่ Stefan cel Mare University of Suceava, (2009 - Management) ปริญญาเอกทางด้านวิทยาการคอมพิวเตอร์จาก Stefan cel Mare University of Suceava (2006 - มีส่วนร่วมในการพัฒนาวิธีการและเทคนิคการทำเหมืองข้อมูลและความรู้) งานวิจัยในปัจจุบัน ได้แก่ฐานข้อมูลและการดำเนินการตามทฤษฎี เหมืองข้อมูลและคลังข้อมูลประยุกต์ โดยใช้เทคโนโลยีสารสนเทศที่ทันสมัยในด้านเศรษฐศาสตร์และเรื่องการดูแลสุขภาพ Mirela เป็นผู้เขียนร่วมในหนังสือ 7 เล่ม รวมเขียนบทความมากกว่า 25 บทความ ได้เข้าร่วมงานประชุมมากกว่า 15 งาน และเป็นสมาชิกของคณะกรรมการบริหารหลักสูตรนานาชาติอื่น ๆ อีกด้วย

ผู้ที่มีส่วนร่วมในการเขียนหนังสือเล่มนี้

รายชื่อต่อไปนี้เป็นผู้ที่มีส่วนร่วมในการจัดทำหนังสือต้นฉบับของหนังสือเล่มนี้ โดยจะเรียงลำดับตามชื่อบุคคลที่ช่วยพัฒนาตรวจสอบ และจัดเรียงเนื้อหาและส่วนสำคัญต่าง ๆ ในหนังสือเล่มนี้

ชื่อ	บริษัท/มหาวิทยาลัย	ตำแหน่ง/อาชีพ	การมีส่วนร่วม
Agatha Colangelo	ION Designs, Inc	Data Modeler	พัฒนาโครงสร้างหลักและสารบัญในหนังสือเล่มนี้
Cuneyt Goksu	VBT Vizyon Bilgi-Teknolojileri	DB2 SME and IBM Gold Consultant	ช่วยตรวจสอบรายละเอียดทางเทคนิค
Marcus Graham	IBM US	Software developer	ตรวจสอบเนื้อหาทางเทคนิคในบทที่ 10
Amna Iqbal	IBM Toronto Lab	Quality Assurance -Lotus Foundations	ตรวจสอบเนื้อหาทางเทคนิคของหนังสือทั้งเล่มยกเว้นบทที่ 5 และ 7
Leon Katsnelson	IBM Toronto Lab	Program Director, IBM Data Servers	ตรวจสอบเนื้อหาทางเทคนิคและมีส่วนร่วมในเนื้อหาของบทที่ 10
Jeff (J.Y.) Luo	IBM Toronto Lab	Technical Enablement Specialist	ตรวจสอบเนื้อหาในบทที่ 7
Fraser McArthur	IBM Toronto Lab	Information Management Evangelist	ตรวจสอบเนื้อหาทางเทคนิค

Danna Nicholson	IBM US	STG ISV Enablement, Web Services	ตรวจสอบเนื้อหาของหนังสือทั้งเล่ม
Rulesh Rebello	IBM India	Advisory Manager - IBM Software Group Client Support	ตรวจสอบเนื้อหาทางเทคนิค
Suresh Sane	DST Systems, Inc	Database Architect	ตรวจสอบเนื้อหาทางเทคนิคโดยเฉพาะที่เกี่ยวข้องกับ SQL
Nadim Sayed	IBM Toronto Lab	User-Centered Design Specialist	ตรวจสอบเนื้อหาในบทที่ 1
Ramona Truta	University of Toronto	Lecturer	พัฒนาโครงสร้างหลักและสารบัญของหนังสือเล่มนี้

กิตติกรรมประกาศ

คณะผู้จัดทำขอขอบคุณเป็นอย่างสูงสำหรับบุคคลต่อไปนี้สำหรับความช่วยเหลือในการพัฒนาส่วนต่างๆในหนังสือเล่มนี้
Natasha Tolub สำหรับการออกแบบหน้าปกของหนังสือเล่มนี้
Susan Visser สำหรับความช่วยเหลือในการจัดพิมพ์หนังสือตัวต้นฉบับ

สำหรับการจัดทำหนังสือเล่มนี้ให้เป็นภาษาไทย ผู้แปลใคร่ขอขอบคุณบุคคลต่างๆ ต่อไปนี้
Raul F. Chong หัวหน้าทีมพัฒนาเนื้อหาต้นฉบับ และเป็นผู้จัดการโครงการ DB2 ณ IBM Toronto Lab ประเทศแคนาดา
คุณวีระกิจ โล่ทองเพชร ผู้จัดการโครงการส่งเสริมการศึกษาและสนับสนุนนักพัฒนา บริษัท ไอบีเอ็ม ประเทศไทย ที่คอยให้ความช่วยเหลือในด้านต่างๆ ทั้งการประสานงาน และการตรวจสอบเนื้อหา จนหนังสือเล่มนี้ได้รับการดำเนินการแปลเป็นภาษาไทยและจัดทำได้เป็นผลสำเร็จ
คุณธนะพงษ์ จารุเวคิน ผู้เชี่ยวชาญทางเทคนิค กลุ่มซอฟต์แวร์บริหารจัดการข้อมูล บริษัท ไอบีเอ็ม ประเทศไทย
คุณพรชัย เอื้อวัฒนะชาคร ผู้เชี่ยวชาญทางเทคนิค กลุ่มซอฟต์แวร์บริหารจัดการข้อมูล บริษัท ไอบีเอ็ม ประเทศไทย
คุณรุ่งทิวา เอี่ยมเตชะ ผู้เชี่ยวชาญทางเทคนิค กลุ่มซอฟต์แวร์บริหารจัดการข้อมูล บริษัท ไอบีเอ็ม ประเทศไทย ที่ช่วยตรวจสอบเนื้อหาในบทต่าง ๆ ของหนังสือเล่มนี้
คุณตรัยรัตน์ สุวรรณประทีป ผู้เชี่ยวชาญทางด้านเทคโนโลยี บริษัท ไอบีเอ็ม ประเทศไทย ที่ช่วยตรวจสอบเนื้อหาในบทที่ 10

1

บทที่ 1 - แบบจำลองข้อมูลและฐานข้อมูล

ข้อมูลถือได้ว่าเป็นทรัพย์สินที่สำคัญส่วนหนึ่งของธุรกิจ ที่ใช้สำหรับการบริหารและการจัดการภายในองค์กร โดยข้อมูลอาจจะมีการรวบรวมจากแหล่งข้อมูลต่างๆ ตัวอย่างเช่นการจัดเก็บและรวบรวมข้อมูลของผู้บริโภคที่มีการใช้บัตรเครดิตเพื่อหาแบบแผนการใช้บัตรเครดิตของลูกค้า หรือหน่วยงานทางด้านอวกาศยังมีการรวบรวมข้อมูลต่างๆ จากดาวพระเคราะห์เป็นต้น ข้อมูลขององค์กรจึงมีความสำคัญมาก และต้องการความเสถียรภาพ ความปลอดภัย อีกทั้งยังต้องการโปรแกรมที่มีความสามารถในการเรียกใช้งานข้อมูล จัดเก็บและประมวลผลข้อมูลได้อย่างรวดเร็ว คำตอบของความต้องการเหล่านี้คือฐานข้อมูลที่มีความเสถียรภาพและเชื่อถือได

ซอฟต์แวร์ฐานข้อมูล (Database System Software) ถูกใช้งานอย่างแพร่หลาย ตัวอย่างเช่นผู้ใช้ระบบงานหลายพันล้านคนทั่วโลกในการทำธุรกรรมการเงินผ่านเครื่องอัตโนมัติจากแหล่งต่างๆ มีการสงข้อมูลการทำธุรกรรมและการเข้าถึงข้อมูลในส่วนรักษาความปลอดภัยของสำนักงาน จากที่กล่าวมานี้เป็นคุณสมบัติและความสามารถของซอฟต์แวร์ฐานข้อมูล เพื่อให้เข้าใจถึงพื้นฐานของแบบจำลองข้อมูลและระบบการจัดการฐานข้อมูล ซึ่งจะอธิบายรายละเอียดในบทนี้

1.1 ฐานข้อมูลคืออะไร

ฐานข้อมูลมีต้นกำเนิดมาจากการค้นคว้าวิจัยในเชิงของวิทยาการคอมพิวเตอร์ ฐานข้อมูลเป็นพื้นที่จัดเก็บข้อมูล ที่ถูกออกแบบมาเพื่อการจัดเก็บข้อมูลอย่างมีประสิทธิภาพ สามารถให้ผู้ใช้จัดการข้อมูล การเรียกใช้และการเข้าถึงข้อมูลในลักษณะต่างๆ และรวมทั้งการบำรุงรักษาข้อมูลได้ ฐานข้อมูลมีหลายประเภทตามความเหมาะสมสำหรับความต้องการของแต่ละอุตสาหกรรม ฐานข้อมูลอาจมีความสามารถในการจัดเก็บข้อมูลประเภทแบบไบนารีไฟล์ เอกสาร รูปภาพ วีดีโอ ข้อมูลเชิงสัมพันธ์ ข้อมูลเชิงมิติ ข้อมูลทรานแซคชัน ข้อมูลเชิงวิเคราะห์ ข้อมูลภูมิศาสตร์และข้อมูลประเภทอื่นๆ

ข้อมูลสามารถจัดเก็บได้หลากหลายรูปแบบได้แก่ แบบตาราง แบบลำดับชั้น และแบบกราฟ กรณีที่ข้อมูลถูกจัดเก็บในรูปแบบของตารางจะถูกเรียกว่า ฐานข้อมูลเชิงสัมพันธ์ (Relational database) เมื่อเป็นการจัดเก็บข้อมูลแบบโครงฐานแบบทรี (Tree) จะถูกเรียกว่า ฐานข้อมูลแบบลำดับชั้น (Hierarchical database) ข้อมูลถูกจัดเก็บในรูปแบบของกราฟความสัมพันธ์ระหว่างออบเจ็กต์จะหมายถึง ฐานข้อมูลแบบเครือข่าย (Network database) สำหรับหนังสือเล่มนี้จะให้ความสำคัญกับฐานข้อมูลเชิงสัมพันธ์เป็นหลัก

1.2 ระบบการจัดการฐานข้อมูลคืออะไร

ระบบจัดการฐานข้อมูล (DataBase Management System: DBMS) หมายถึงซอฟต์แวร์ที่ใช้สำหรับเป็นเครื่องมือในการควบคุมการเข้าถึง จัดระเบียบ จัดเก็บ จัดการ เรียกใช้ และบำรุงรักษาข้อมูลในฐานข้อมูล การใช้งานฐานข้อมูลจำเป็นที่จะต้องติดตั้งซอฟต์แวร์บนเครื่องแม่ข่ายฐานข้อมูล (Database server) เพื่อให้สามารถทำงานและรองรับการให้บริการได้อย่างมีประสิทธิภาพ

ระบบจัดการฐานข้อมูลเป็นเครื่องมือที่ทำให้เราสามารถจัดเก็บข้อมูล และอำนวยความสะดวกในการเข้าถึง และการเรียกใช้ข้อมูล โดยข้อมูลที่จัดเก็บในฐานข้อมูลจะต้องมีความสอดคล้องกันของข้อมูล ซึ่งปกติระบบฐานข้อมูลจะมีความสามารถในการรองรับผู้ใช้งานหลายคนเข้ามาทำงานพร้อมกันได้ โดยระบบฐานข้อมูลจะต้องมีระบบการป้องกัน เช่น การเพิ่ม การปรับปรุง และการลบข้อมูลตัวเดียวกันได้โดยไม่มีผลกระทบต่อผู้ใช้งานอื่น หมายความว่าผู้ใช้งานแต่ละคนจะไม่ทำให้เกิดความไม่สอดคล้องกันของข้อมูล ข้อมูลจะไม่สูญหายโดยไม่ตั้งใจระหว่างการดำเนินการ นอกจากนี้เราจำเป็นต้องมีเครื่องมือที่ใช้ติดต่อกับระบบฐานข้อมูล (Interface) แบบมาตรฐานสำหรับการเข้าถึงข้อมูล เครื่องมือสำหรับการสำรองข้อมูล การคืนกลับข้อมูล และกู้คืน และระบบฐานข้อมูลที่ดี ควรมีวิธีการจัดการกับสิ่งที่นอกเหนือจากความคาดหมายอื่นๆ เช่น การรองรับการทำงานกับข้อมูลจำนวนมหาศาล และผู้ใช้จำนวนมาก ระบบจัดการฐานข้อมูลจึงถูกออกแบบเพื่อจัดการกับความท้าทายต่างๆที่ได้กล่าวมา

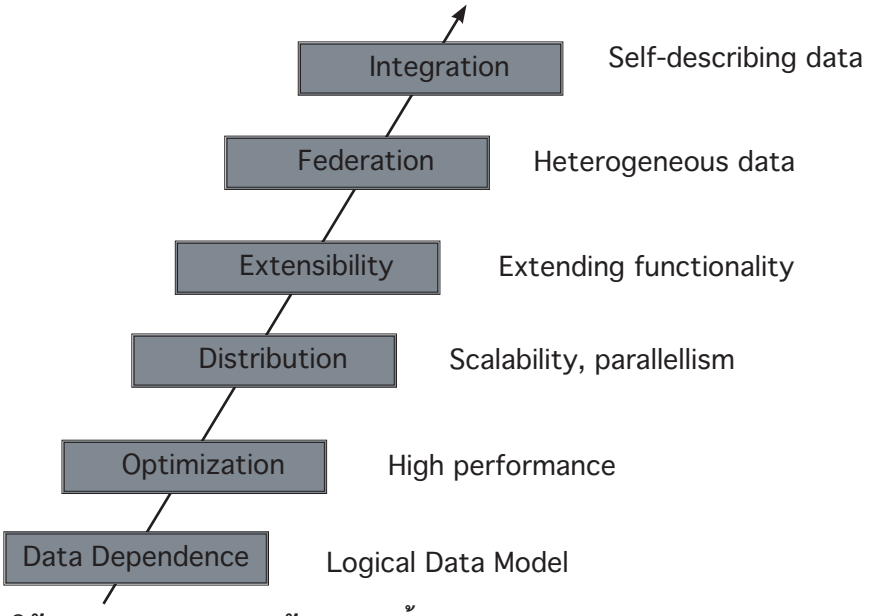
ผู้ผลิตซอฟต์แวร์ระบบฐานข้อมูลส่วนใหญ่ได้มีการพัฒนาระบบการจัดการฐานข้อมูลในลักษณะที่เป็นฐานข้อมูลเชิงสัมพันธ์ (Relational DataBase Management System: RDBMS) ซึ่งปัจจุบันได้กลายมาเป็นแกนหลักสำคัญในการรองรับข้อมูลขององค์กรต่างๆและทำงานร่วมกันกับแอพพลิเคชันในหลากหลายอุตสาหกรรม ตัวอย่างเช่น งานธนาคาร งานขนส่ง งานสาธารณสุขและอื่นๆ อีกทั้ง แอพพลิเคชันที่ทำงานบนเว็บที่มีปริมาณเพิ่มขึ้นทั้งในด้านของจำนวนผู้ใช้และขอบเขตของการใช้งาน เช่นในกรณีของงานธุรกรรมอิเล็กทรอนิกส์เพื่อรองรับข้อมูลของการทำธุรกิจและการค้าแบบออนไลน์

1.2.1 วิวัฒนาการของระบบการจัดการฐานข้อมูล

ในปี ค.ศ. 1960 ระบบแบบเครือข่ายและระบบแบบลำดับชั้นเป็นเทคโนโลยีที่ล้ำสมัยในขณะนั้นถูกนำมาใช้ทำธุรกรรมแบบอัตโนมัติของธนาคาร ด้านบัญชี ด้านการสั่งซื้อ และมีระบบการประมวลผลโดยใช้คอมพิวเตอร์แม่ข่ายที่เรียกว่า mainframe ระบบที่ใช้งานดังกล่าวเป็นพื้นฐานของระบบเบื้องต้น พื้นฐานของสถาปัตยกรรมแบบที่มีการทำงานร่วมกันระหว่างการจัดการระดับกายภาพ (Physical) ของข้อมูลและการจัดการระดับลอจิคัล (Logical) ของข้อมูล โดยเมื่อต้องการเปลี่ยนแปลงข้อมูลที่ใดที่หนึ่งบนดิสก์ไปยังอีกที่หนึ่ง แอพพลิเคชันก็จะถูกใช้สำหรับทำการเปลี่ยนแปลงข้อมูลเพื่ออ้างอิงไปยังตำแหน่งใหม่

จากเอกสารวิจัยของ E.F. Codd นักวิจัยของ ไอบีเอ็ม San Jose ในปี 1970 มีแนวความคิดใหม่ตามเอกสารที่ชื่อว่า “A relational model of data for large shared data banks” [1.1] ในแนวคิดเรื่องของความเป็นอิสระของข้อมูลโดยแบ่งระดับทางกายภาพของข้อมูล และระดับลอจิคัลของข้อมูลที่อ้างอิงในแอพพลิเคชัน โดยข้อมูลสามารถถูกเคลื่อนย้ายจากดิสก์ที่จัดเก็บจากที่หนึ่งไปยังอีกที่หนึ่ง หรือจัดเก็บในรูปแบบต่างๆ กัน ได้โดยปราศจากการที่แอพพลิเคชันจะต้องมีการเขียนใหม่ นักพัฒนาแอพพลิเคชันไม่จำเป็นต้องสนใจในเรื่องของการจัดการข้อมูลในระดับกายภาพ แต่สามารถมุ่งเน้นในการจัดการระดับลอจิคัล ของข้อมูลในเฉพาะโปรแกรมแอพพลิเคชันนั้นๆแทน

รูปที่ 1.1 แสดงให้เห็นถึงวิวัฒนาการของระบบการจัดการฐานข้อมูล



รูป 1.1 วิวัฒนาการของระบบการจัดการฐานข้อมูล

จากรูป 1.1 แสดงให้เห็นถึงวิวัฒนาการของระบบการจัดการฐานข้อมูลด้วยแบบจำลองเชิงสัมพันธ์ที่ให้ความเป็นอิสระของข้อมูล System R ของ ไอบีเอ็ม เป็นระบบแรกที่นำแนวคิดของ Codd มาประยุกต์ใช้ โดย System R เป็นพื้นฐานสำหรับ SQL/DS ซึ่งภายหลังได้ถูกพัฒนาเป็นโปรแกรม DB2 นอกจากนี้ยังมีคุณสมบัติที่รองรับภาษา SQL ซึ่งเป็นภาษามาตรฐานที่ใช้สำหรับฐานข้อมูลเชิงสัมพันธ์และเปิดโอกาสให้มีการพัฒนาระบบการจัดการฐานข้อมูลในเชิงพาณิชย์อีกด้วย

ปัจจุบันระบบจัดการฐานข้อมูลเชิงสัมพันธ์เป็นระบบที่ถูกใช้มากที่สุดในการจัดการฐานข้อมูล และถูกพัฒนาขึ้นโดยบริษัทผู้พัฒนาหลายแห่ง ไอบีเอ็ม เป็นหนึ่งในผู้นำตลาดด้วยผลิตภัณฑ์ DB2 database server รวมถึงผลิตภัณฑ์จากค่ายอื่นๆ เช่น Oracle, Microsoft SQL Server, INGRES, PostgreSQL, MySQL และ dBASE

ฐานข้อมูลเชิงสัมพันธ์กลายเป็นฐานข้อมูลที่ได้รับความนิยมเป็นอย่างมาก มีความต้องการในการจัดเก็บข้อมูลและการสอบถามข้อมูล (Query) ที่มีประสิทธิภาพและรวดเร็วมากขึ้น ในซอฟต์แวร์ระบบฐานข้อมูล DB2 เครื่องมือ DB2's optimizer เป็นส่วนประกอบที่ทันสมัยของฐานข้อมูล ในมุมมองของผู้ใช้ DB2's optimizer เสมือนว่าเป็นกล่องดำที่มีการส่งค่าอินพุทเป็นคำสั่ง SQL เพื่อสอบถามข้อมูล และ DB2's optimizer จะทำการคำนวณหาวิธีที่เร็วที่สุดในการเข้าถึงข้อมูลจากปัจจัยต่างๆ เช่น ความเร็วของ CPU และดิสก์ จำนวนข้อมูลที่เข้าถึง ตำแหน่งของข้อมูล ประเภทของข้อมูลดัชนีของข้อมูลที่มีอยู่และอื่นๆ DB2's optimizer จะใช้แนวคิดในเชิงประหยัดต้นทุนในการทำงานที่น้อยที่สุด (Cost–based) ในการประมวลผลจากปริมาณของข้อมูลที่มีการจัดเก็บมากขึ้น มีการจัดเก็บข้อมูลอยู่บนเครื่องแพลตฟอร์มที่หลากหลาย ระบบฐานข้อมูลในปัจจุบันได้มีการเพิ่มประสิทธิภาพการทำงาน เช่น ระบบฐานข้อมูลของ DB2 ที่มีความสามารถในการทำงานบนระบบปฏิบัติการ Linux, UNIX และ Windows อีกทั้งการเพิ่มคุณสมบัติของการแบ่งพาร์ติชัน (Database Partitioning Feature) ในการกระจายฐานข้อมูลไปยังเครื่องแม่ข่าย DB2 หลายเครื่องภายใต้แนวคิด Shared Nothing Architecture โดยในแต่ละเครื่องสามารถที่จะเพิ่ม CPU และดิสก์ คำสั่งในการสอบถามข้อมูลสามารถทำงานแบบขนาน (Parallelized) ได้ โดยแต่ละเครื่องสามารถดึงข้อมูลเฉพาะส่วนที่ต้องการมาประกอบเป็นข้อมูลที่ต้องการทั้งหมด

วิวัฒนาการต่อมาของระบบจัดการฐานข้อมูลคือ การขยายความสามารถ ภาษา SQL ที่ประดิษฐ์โดย ไอบีเอ็ม ตั้งแต่ต้นปี ค.ศ. 1970 ได้มีการปรับปรุงอย่างต่อเนื่อง ถึงแม้ภาษา SQL จะมีประสิทธิภาพสูงและผู้ใช้สามารถสร้างภาษา SQL เพิ่มเติมได้เองแล้วนั้น ตัวอย่างเช่น ในโปรแกรม DB2 ผู้ใช้สามารถเขียนฟังก์ชันและ store procedure ขึ้นมาใช้งานเองตามลักษณะของการทำงานที่ต้องการได้อีกด้วย อีกทั้งระบบจัดการฐานข้อมูลได้จัดการกับปัญหาความแตกต่างของชนิดข้อมูลและความแตกต่างของแหล่งข้อมูล ในจุดนี้เอง DB2 ได้เปลี่ยนชื่อโดยเอาคำว่า Universal เข้ามารวมด้วยเป็น DB2 Universal Database (DB2 UDB) แม้ภายหลังจะตัดคำว่า Universal เพื่อให้ง่ายต่อการเรียกชื่อแต่ก็ยังคงคุณสมบัติที่สามารถจัดเก็บข้อมูลได้ทุกชนิดไม่ว่าจะเป็นข้อมูลวิดีโอ เสียง ข้อมูลไบนารีและอื่นๆ นอกจากนี้ยังปฏิบัติตามมาตรฐานสหพันธ์ คือสามารถสอบถามข้อมูลจากผลิตภัณฑ์จากค่ายอื่นที่ไม่ใช่ผลิตภัณฑ์ของ ไอบีเอ็ม ในฐานข้อมูล DB2 ได้

สุดท้ายนี้ในการพัฒนาในลำดับต่อไปเป็นการเน้นเรื่องของการรวบรวมข้อมูล (Data Integration) ซึ่งทุกวันนี้หลายองค์กรจำเป็นต้องมีการแลกเปลี่ยนข้อมูลซึ่งกันและกัน ซึ่ง eXtensible Markup Language (XML) เป็นเทคโนโลยีหนึ่งที่ได้รับความสนใจเพื่อตอบวัตถุประสงค์ดังกล่าว XML เป็นภาษาที่มีการขยายความของข้อมูลตัวเอง การใช้งาน XML ได้มีการขยายตัวและใช้งานใน Web 2.0 และ Service-Oriented Architecture (SOA) ซึ่งทางบริษัท ไอบีเอ็ม เองก็ได้ให้ความสำคัญสำหรับ XML โดยมีการพัฒนาเทคโนโลยีที่ชื่อว่า pureXML มาใช้งานบนระบบจัดการฐานข้อมูล DB2 ซึ่งเป็นการพัฒนาเพิ่มเติมทำให้ DB2 สามารถจัดเก็บข้อมูล XML ในรูปแบบระดับชั้น ซึ่งเป็นรูปแบบของ XML และใน DB2 Engine ได้มีการพัฒนาภาษา ที่เรียกว่า XQuery ซึ่งเป็นภาษาสำหรับการสอบถามข้อมูลที่จัดเก็บในรูปแบบของ XML นอกจากนี้คุณสมบัติที่เรียกว่า pureXML ใน DB2 ซึ่งมีความสามารถในการจัดการข้อมูล XML ได้อย่างรวดเร็ว และในขณะเดียวกันก็ยังมีการรักษาความปลอดภัยของข้อมูล มีความเสถียรภาพและความยืดหยุ่นสำหรับการใช้งานข้อมูล

ปัจจุบันเทคโนโลยีที่กำลังได้รับความสนใจคือ Cloud Computing ซึ่งทาง ไอบีเอ็ม เองนั้นก็ได้มีการพัฒนา Cloud ซึ่งในความเป็นจริงได้เริ่มมีการใช้ DB2 images บน Amazon EC2 ไอบีเอ็ม Smart Business Development และทดสอบบน ไอบีเอ็ม Cloud เอง อีกทั้งระบบฐานข้อมูลของ DB2 ก็มีความสามารถในการแบ่งพาร์ติชันได้ดังที่กล่าวไว้ก่อนหน้านี้ซึ่งเหมาะสำหรับการทำ Cloud เมื่อเราต้องการแบ่งขนาดหรือความต้องการของเครื่องแม่ข่ายในการจัดกลุ่ม cluster หรือการทำ Data rebalancing ก็สามารถทำงานได้โดยอัตโนมัติ โดยโปรแกรม DB2 จะเป็นประโยชน์เมื่อต้องการข้อมูลจากเซิร์ฟเวอร์ฐานข้อมูลในการจัดการข้อมูลเชิงสรุปเป็นรายเดือนหรือสรุปรายปีจากข้อมูลทรานแซคชัน

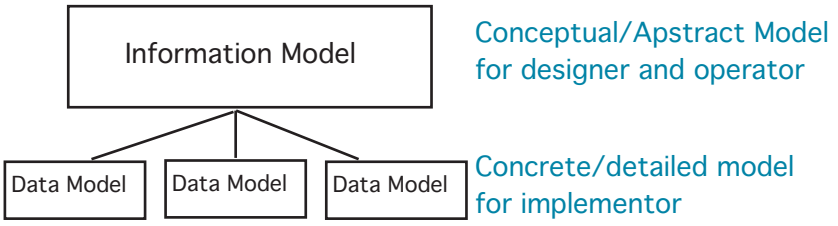
1.3 รูปแบบข้อมูล (Information Models) และแบบจำลองข้อมูล (Data Models)

รูปแบบข้อมูล (Information Models) เป็นการนำเสนอแบบจำลองโครงสร้างของข้อมูล โดยมีการนำเสนอเป็นลักษณะเอนทิตี (entity) ที่มีคุณสมบัติ (properties) ของตัวเอง มีความสัมพันธ์ (relationship) ระหว่างเอนทิตี และสิ่งที่สามารถกระทำได้กับเอนทิตี โดยเอนทิตีจะถูกสร้างเป็นแบบจำลองที่มาจากความต้องการของธุรกิจ และปัญหาที่เกิดในโลกของความเป็นจริง (Real-world problem) เช่น เอนทิตีของอุปกรณ์เครือข่าย หรือเป็นเอนทิตีที่เป็นนามธรรม เช่น เอนทิตีของระบบการเรียกเก็บเงิน แรงจูงใจที่อยู่เบื้องหลังเป็นแนวคิดที่ต้องการสร้างรูปแบบเพื่ออธิบายปัญหาให้เกิดความกระจ่างและสามารถนำไปพัฒนาได้จริงโดยระบบซอฟต์แวร์ การสร้างรูปแบบดังกล่าวนั้นเป็นการแปลงรูปแบบข้อมูลที่เรียกว่าการทำแบบจำลองข้อมูล ซึ่งอาจจะใช้แบบจำลองเชิงวัตถุ (Unified Modeling Language - UML) แบบจำลองอีอาร์ (Entity Relationship Models : E-R model) หรือ XML schemas

การสร้างแบบจำลองข้อมูลนั้นมีความสำคัญโดยต้องคำนึงถึงความยืดหยุ่นที่ต้องรองรับการเปลี่ยนแปลงในอนาคตและไม่ให้เกิดผลกระทบที่จะเกิดขึ้นต่อการทำงาน แบบจำลองที่สร้างขึ้นต้องเข้ากันได้และ

ต้องรองรับสำหรับส่วนต่อขยายในอนาคต รูปแบบข้อมูล และแบบจำลองข้อมูลนั้นมีความต่างกันเนื่องจากตอบวัตถุประสงค์คนละอย่างกัน จุดประสงค์หลักของรูปแบบข้อมูล นั้นก็คือการจัดการแบบจำลองในระดับแนวคิด (Conceptual level) ไม่ยึดติดกับวิธีการพัฒนาหรือโปรโตคอลในการส่งข้อมูล ระดับของรายละเอียดที่อธิบายรูปแบบข้อมูลนั้นจะขึ้นอยู่กับความต้องการของนักออกแบบ การที่จะทำให้การออกแบบโดยรวมเป็นไปได้อย่างชัดเจน รูปแบบข้อมูลต้องไม่แสดงโปรโตคอลและรายละเอียดการพัฒนา ลักษณะที่สำคัญอีกอย่างหนึ่งของรูปแบบข้อมูลคือการกำหนดความสัมพันธ์ระหว่างออบเจ็กต์ที่ดำเนินการ สำหรับแบบจำลองข้อมูลนั้นต่างออกไปแบบจำลองข้อมูลอยู่ในระดับที่สัมผัสได้มากขึ้นและมีการเพิ่มรายละเอียดต่างๆ ซึ่งเป็นประโยชน์ต่อนักพัฒนาซอฟต์แวร์รวมไปถึงโปรโตคอลเฉพาะต่างๆ แบบจำลองข้อมูลเป็นเหมือนพิมพ์เขียวของทุกระบบฐานข้อมูล

รูปที่ 1.1 แสดงให้เห็นถึงความสัมพันธ์ระหว่างแบบจำลองข้อมูลและรูปแบบข้อมูล



รูปที่ 1.1 – ความสัมพันธ์ระหว่างแบบจำลองข้อมูลและรูปแบบข้อมูล

เนื่องจากแบบจำลองระดับแนวคิดสามารถนำไปติดตั้งได้หลายแบบ แบบจำลองข้อมูลหลายแบบจำลองสามารถถูกแบ่งมาจากแบบจำลองข้อมูลระดับแนวคิดอันเดียวกันได้

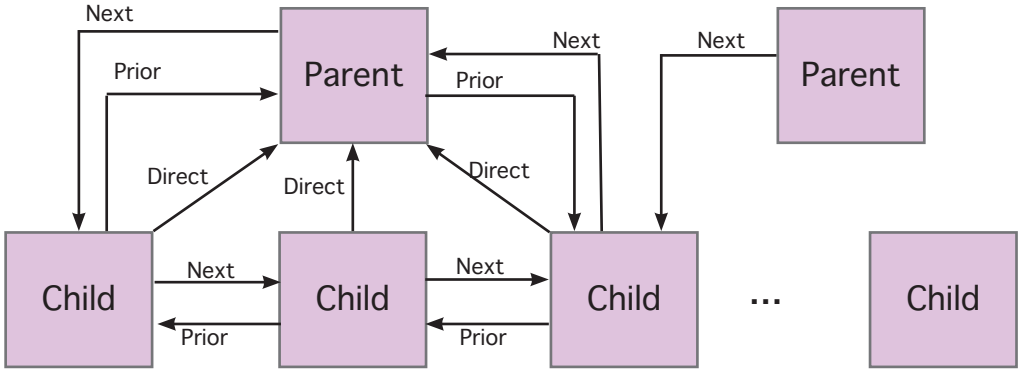
1.4 ประเภทของรูปแบบข้อมูล

ประเภทรูปแบบข้อมูลมีพัฒนาการตามยุคสมัยต่างๆซึ่งสามารถแบ่งออกเป็น 9 ยุคดังต่อไปนี้

- ยุค 1970 - Network (CODASYL)
- ปลายยุค 1960's และ ยุค 1970 - Hierarchical (IMS)
- ยุค 1970 และต้นยุค 1980 - Relational
- ยุค 1970 - Entity-Relationship
- ยุค 1980 - Extended Relational
- ปลายยุค 1970 และ ยุค 1980 - Semantic
- ปลายยุค 1980 และ ต้นยุค 1990 - Object-oriented
- ปลายยุค 1980 และ ต้นยุค 1990 - Object-relational
- ปลายยุค 1990 จนถึงปัจจุบัน - Semi-structured (XML)

1.4.1 แบบจำลองเครือข่าย (Network model)

ในปี 1969 CODASYL (Committee on Data Systems Languages) ได้ออกข้อกำหนดเกี่ยวกับแบบจำลองเครือข่าย และต่อมาในปี 1971 และ 1973 ออกข้อกำหนดสำหรับการประมวลผลครั้งละเรคอร์ด (record-at-a-time) ของภาษาที่ใช้จัดการข้อมูล ตัวอย่างของแบบจำลองเครือข่ายแสดงไว้รูปที่ 1.2

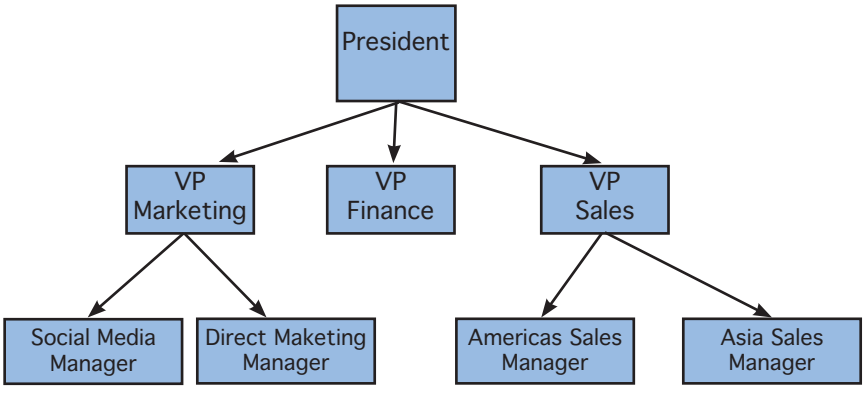


รูปที่ 1.2 - แบบจำลองเครือข่าย

จากภาพข้างต้นแสดงถึงชนิดของเรคอร์ดโดยใช้โครงสร้างแผนภูมิ เรคอร์ดเหล่านั้นสามารถระบุได้ด้วยคีย์ กลุ่มของชนิดของเรคอร์ดจะประกอบกันเป็นเครือข่าย CODASYL หรือ ฐานข้อมูล CODASYL โดยที่ตัวลูก (child) หนึ่งตัวสามารถมาจากตัวแม่ (parent) หลายตัวและเชื่อมต่อกันด้วยพอยน์เตอร์ไปยังตัวถัดไปและเชื่อมไปยังตัวก่อนหน้าโดยตรง

1.4.2 แบบจำลองลำดับชั้น (Hierarchical Model)

แบบจำลองลำดับชั้นมีโครงสร้างการจัดการโดยโครงสร้างแบบต้นไม้ รากของต้นไม้ (root) จะทำหน้าที่เป็นโหนดแม่ ซึ่งจะประกอบด้วยโหนดลูก โดยที่โหนดลูกไม่สามารถมีโหนดแม่ได้หลายโหนดพร้อมกัน แต่โหนดแม่สามารถมีโหนดลูกได้หลายตัว ดังแสดงในรูปที่ 1.3



รูปที่ 1.3 - แบบจำลองลำดับชั้น

ในแบบจำลองลำดับชั้นจะจัดเก็บชื่อฟิลด์ซึ่งเกี่ยวข้องกับชนิดข้อมูลเรียกว่าชนิดเรคอร์ด ชนิดของเรคอร์ดจะต้องอธิบายข้อมูลที่จัดเก็บข้างในว่าเป็นข้อมูลอะไร บางฟิลด์ในชนิดเรคอร์ดถูกใช้เป็นคีย์

ระบบการจัดการฐานข้อมูลแบบลำดับชั้นระบบแรกคือ IMS (Information Management System) นำเสนอโดย ไอบีเอ็ม ในปี ค.ศ. 1968 แต่เดิมนั้นเป็นฐานข้อมูลที่ใช้สำหรับโปรแกรมของยานอวกาศอพอลโลที่

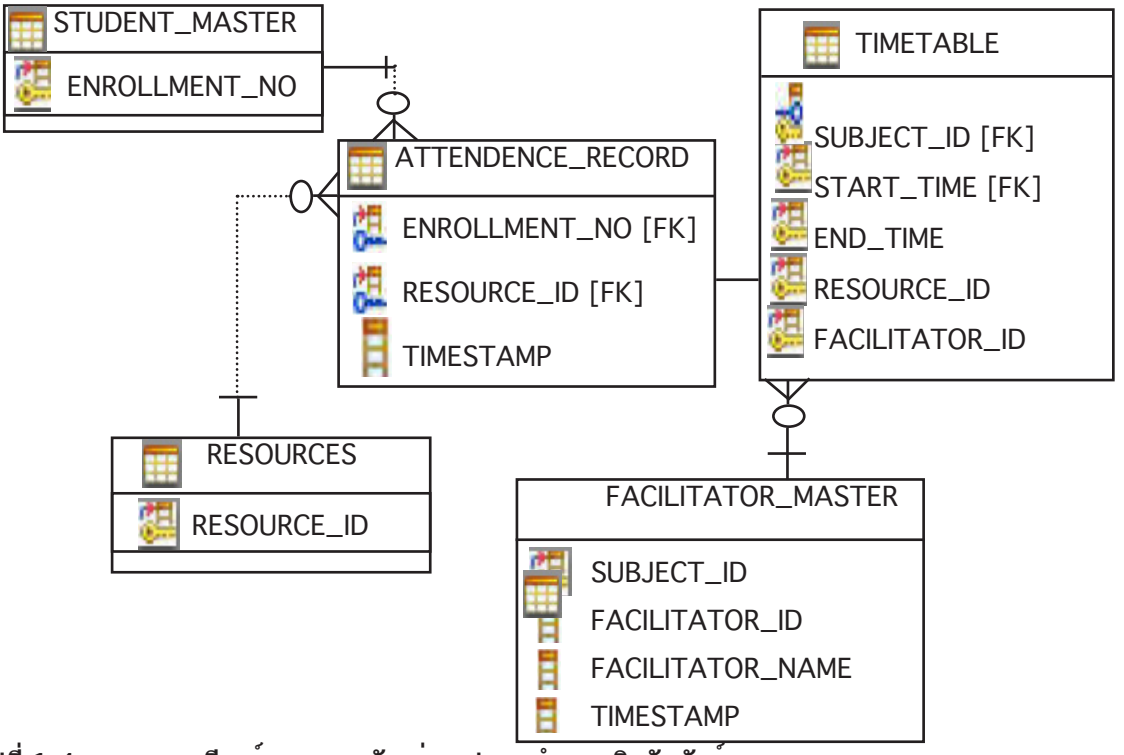
นำมนุษย์คนแรกลงสู่พื้นผิวดวงจันทร์ IMS ในปัจจุบันเป็นฐานข้อมูลที่มีเสถียรภาพสูงและยังมีการใช้แพร่หลายในหลายบริษัททั่วโลก

1.4.3 แบบจำลองเชิงสัมพันธ์ (Relational model)

แบบจำลองเชิงสัมพันธ์เป็นแบบจำลองที่เข้าใจง่ายและมีรูปแบบที่เป็นระบบ มีรากฐานมาจากคณิตศาสตร์บนทฤษฎีและการพิสูจน์แคลคูลัส แบบจำลองเชิงสัมพันธ์เป็นแบบจำลองที่ถูกใช้มากที่สุดในฐานข้อมูลปัจจุบัน สิ่งที่มีส่วนผลักดันจากงานวิจัยของ Codd นั่นก็คือการที่มองว่านักพัฒนาระบบ IMS ใช้เวลามากไปในการจัดการบำรุงรักษาแอพพลิเคชัน IMS เมื่อทำการเปลี่ยนแปลงตรรกะหรือลักษณะทางกายภาพ ดังนั้นจุดประสงค์หลักของของ Codd คือ สร้างแบบจำลองที่มีความเป็นอิสระของข้อมูล โดยได้มีข้อเสนอมา 3 ข้อคือ

- จัดเก็บข้อมูลบนโครงสร้างข้อมูลที่ง่าย (ตาราง)
- สามารถเข้าถึงการประมวลผลแบบกลุ่ม (set-at-a-time) โดยภาษาจัดการข้อมูล (data manipulation language) ซึ่งเป็นภาษาในระดับสูง
- เป็นอิสระจากการจัดเก็บทางกายภาพ

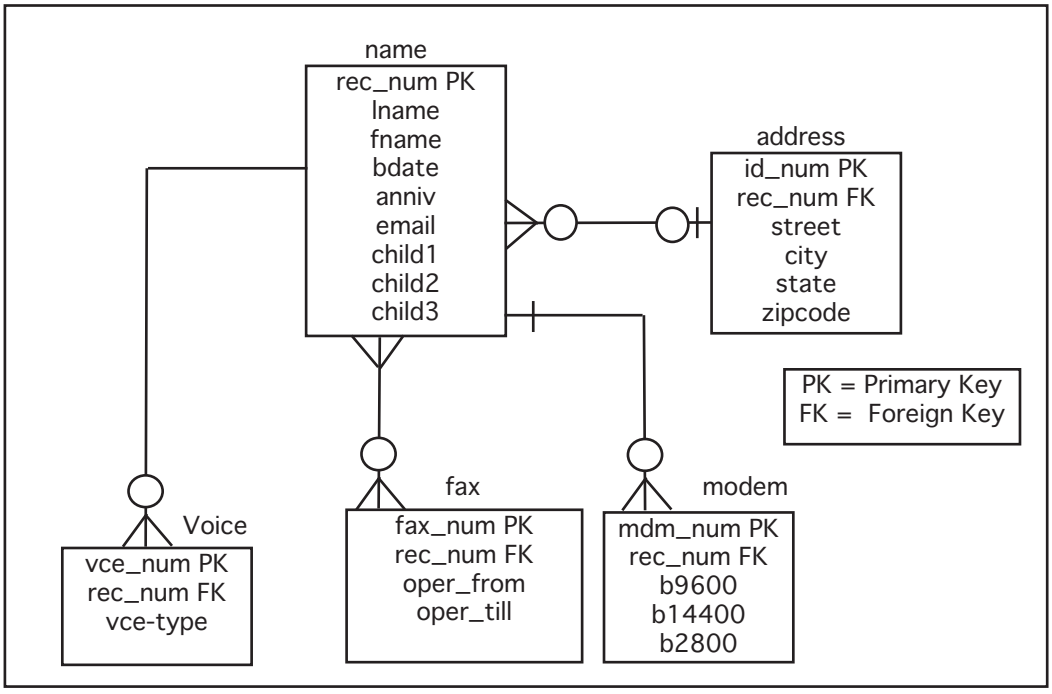
ด้วยโครงสร้างอย่างง่าย ข้อแรกทำให้เป็นโอกาสที่ดีที่ทำให้ข้อมูลเชิงตรรกะมีความเป็นอิสระ การใช้ภาษาระดับสูงในการจัดการทำให้ข้อมูลระดับกายภาพเป็นอิสระมากขึ้น แบบจำลองเชิงสัมพันธ์นี้ยังเอื้อสำหรับการจัดเก็บข้อมูลทางกายภาพ (Physical Data) ที่เป็นอิสระอีกด้วยซึ่งเป็นข้อจำกัดของ IMS และ CODASY ในรูปที่ 1.4 แสดงแผนภาพอีอาร์ (Entity-Relationship diagram) ที่มีเอนทิตี (ตาราง) และแสดงความสัมพันธ์ในแบบจำลอง ซึ่งจะได้อธิบายเพิ่มเติมเกี่ยวกับแผนภาพอีอาร์ในส่วนถัดไป



รูปที่ 1.4 – แผนภาพอีอาร์การแสดงตัวอย่างรูปแบบจำลองเชิงสัมพันธ์

1.4.4 แบบจำลองอีอาร์

ในกลางทศวรรษที่ 1970 Peter Chen ได้นำเสนอแบบจำลองอีอาร์ซึ่งเป็นอีกทางเลือกหนึ่งในการทำรูปแบบความสัมพันธ์นอกเหนือจาก CODASYL และแบบจำลองลำดับชั้น Peter Chen ได้เสนอแนวคิดของฐานข้อมูลที่เป็นกลุ่มของอินสแตนซ์ของเอนทิตี เอนทิตีเป็นออบเจ็กต์ที่เป็นอิสระจากเอนทิตีอื่นในฐานข้อมูล เอนทิตีมีแอตทริบิวต์ซึ่งแอตทริบิวต์แสดงถึงคุณสมบัติหรือลักษณะของเอนทิตี แอตทริบิวต์หนึ่งตัวหรือหลายตัวถูกกำหนดให้เป็นคีย์ สุดท้ายเอนทิตีสามารถมีความสัมพันธ์ระหว่างกันเป็นแบบ หนึ่งต่อหนึ่ง (1 to 1) หนึ่งต่อกลุ่ม (1 to many) กลุ่มต่อกลุ่ม (many to many) ขึ้นอยู่กับแต่ละเอนทิตีว่าจะมีความสัมพันธ์กันอย่างไร ความสัมพันธ์อาจจะมีแอตทริบิวต์เพิ่มเติมในการอธิบายถึงความสัมพันธ์ ในรูปที่ 1.5 แสดงตัวอย่างแผนภาพอีอาร์สำหรับแบบจำลองข้อมูลโทรศัพท์



รูปที่ 1.5 – แผนภาพอีอาร์สำหรับแบบจำลองข้อมูลหมายเลขโทรศัพท์

จากรูปที่ 1.5 เอนทิตีแทนด้วยรูปสี่เหลี่ยม ในภาพก็คือเอนทิตี name, address, voice, fax, และ modem ภายในเอนทิตีจะประกอบด้วยคุณสมบัติเรียกว่าแอตทริบิวต์ ตัวอย่างเอนทิตี voice ประกอบด้วยแอตทริบิวต์ vce_num rec_num และ vce-type PK เป็นสัญลักษณ์แทนคีย์หลัก FK เป็นสัญลักษณ์แทนคีย์นอก แนวคิดเรื่องคีย์จะได้พูดถึงในรายละเอียดต่อไป

แบบจำลองอีอาร์เป็นเครื่องมือในการออกแบบฐานข้อมูลเชิงสัมพันธ์ที่มีประสิทธิภาพ จากเอกสารของ Chen ได้ระบุถึงวิธีการสร้างแผนภาพอีอาร์และยังบอกถึงวิธีในการแปลงแผนภาพอีอาร์ให้เป็นกลุ่มของตารางข้อมูลใน third normal form สำหรับข้อมูลของ third normal form และทฤษฎีการทำบรรทัดฐาน (normalization theory) จะกล่าวถึงในส่วนท้ายของหนังสือเล่มนี้

หมายเหตุ: ปัจจุบัน เราสามารถสร้างแผนภาพอีอาร์ โดยใช้เครื่องมือในการออกแบบแบบจำลองข้อมูล เช่น IBM InfoSphere Data Architect ซึ่งสามารถศึกษารายละเอียดการใช้เครื่องมือดังกล่าวจาก eBook “Getting started with InfoSphere Data Architect” ซึ่งหนังสือเล่มนี้อยู่ในส่วนของ DB2 ในชุด Campus book series

1.4.5 แบบจำลองความสัมพันธ์เชิงวัตถุ (Object-relational model)

แบบจำลองความสัมพันธ์เชิงวัตถุมีความคล้ายคลึงกับแบบจำลองเชิงสัมพันธ์ แต่จะมองเอนทิตีเป็นเชิงวัตถุและมีความสัมพันธ์ที่เป็นแบบคุณสมบัติการสืบทอด คุณลักษณะและข้อดีของแบบจำลองความสัมพันธ์แบบวัตถุคือ

- ผู้ใช้สามารถสร้างชนิดของข้อมูลขึ้นมาเอง
- สนับสนุนคุณสมบัติการสืบทอดของออบเจ็กต์
- สนับสนุนส่วนขยายของออบเจ็กต์ เช่น สนับสนุนการจัดเก็บข้อมูลแผนที่

ฐานข้อมูลความสัมพันธ์เชิงวัตถุมีการจัดเก็บรูปแบบความสัมพันธ์ในรูปแบบของออบเจ็กต์

1.4.6 แบบจำลองข้อมูลอื่นๆ

ในช่วงทศวรรษที่ผ่านมาจะเห็นได้ว่าเราได้ให้ความสำคัญของการทำงานแบบกึ่งโครงสร้าง (semi-structured) การทำงานแบบมีโครงสร้าง (Semantic) และแบบจำลองข้อมูลเชิงวัตถุ (object oriented data models) สำหรับรูปแบบการจัดเก็บข้อมูลแบบกึ่งโครงสร้างที่ได้รับความนิยมมาก คือ XML โดยที่แบบจำลองพื้นฐานของ XML ได้รับความนิยมบน Web 2.0 และสถาปัตยกรรมเชิงบริการ (Service-Oriented Architecture: SOA) ส่วนแบบจำลองข้อมูลเชิงวัตถุ ได้รับความนิยมในมหาวิทยาลัย แต่ไม่ค่อยได้รับการยอมรับในเชิงธุรกิจ อย่างไรก็ตาม ในปัจจุบันมีเครื่องมือที่เป็น object-relational mapping (ORM) ที่ใช้ในการเชื่อมโยงโปรแกรม object-oriented กับ ฐานข้อมูลเชิงสัมพันธ์

1.5 บทบาททั่วไปและเส้นทางสำหรับผู้เชี่ยวชาญด้านฐานข้อมูล

ตามลักษณะการทำงานโดยทั่วไป การทำงานด้านฐานข้อมูลต้องมีการกำหนดบทบาทและหน้าที่ของผู้ปฏิบัติงานที่สัมพันธ์กัน ลำดับต่อไปจะเป็นการอธิบายถึงบทบาทหน้าที่ของผู้ที่เกี่ยวข้องกับระบบฐานข้อมูล

1.5.1 นักสถาปัตยกรรมข้อมูล (Data Architect)

นักสถาปัตยกรรมข้อมูลเป็นผู้รับผิดชอบในการออกแบบสถาปัตยกรรมที่เกี่ยวข้องกับการจัดการข้อมูลที่มีอยู่ในปัจจุบันรวมถึงความสามารถในการรองรับความต้องการข้อมูลในอนาคต สถาปัตยกรรมดังกล่าวควรครอบคลุมถึงฐานข้อมูล การรวบรวมข้อมูล และความหมายของข้อมูลที่จัดเก็บ โดยปกติเป้าหมายของนักสถาปัตยกรรมข้อมูลคือการกำหนดมาตรฐานข้อมูลขององค์กร นักสถาปัตยกรรมข้อมูลอาจหมายถึงนักออกแบบแบบจำลองข้อมูล (แต่ในความเป็นจริงแล้วอาจจะทำหน้าที่มากกว่าการออกแบบ)

สำหรับทักษะพื้นฐานของนักสถาปัตยกรรมข้อมูลที่ควรมี ประกอบด้วย

- สร้างแบบจำลองข้อมูลระดับตรรกะ
- สร้างแบบจำลองข้อมูลระดับกายภาพ

- กำหนดกระบวนการและกฎความสัมพันธ์ของข้อมูล
- สามารถเลือกระบบที่จะนำมาใช้ในการตอบสนองความต้องการการใช้ข้อมูลของธุรกิจ

1.5.2 นักสถาปัตยกรรมฐานข้อมูล (Database Architect)

นักสถาปัตยกรรมฐานข้อมูลนั้นจะมีบทบาทคล้ายกับนักสถาปัตยกรรมข้อมูล แต่จะจำกัดอยู่ในเรื่องของฐานข้อมูลเท่านั้น นักสถาปัตยกรรมฐานข้อมูลมีหน้าที่ดังต่อไปนี้

- รวบรวมเอกสารความต้องการจากผู้ใช้งานและจัดการข้อมูลเหล่านั้นให้ไปเป็นรูปแบบของสถาปัตยกรรม
- ถ่ายทอดความรู้ทางสถาปัตยกรรมให้แก่ผู้ใช้งานและวิธีการจัดการข้อมูล
- สร้างและกำหนดมาตรฐานของการใช้งานฐานข้อมูล มาตรฐานของกระบวนการการพัฒนาแอพพลิเคชัน
- สร้างและกำหนดข้อตกลงในการให้บริการ (Service Level Agreements: SLAs) สำหรับธุรกิจโดยเฉพาะการเตรียมการเพื่อให้ใช้ข้อมูลได้อย่างต่อเนื่อง (High Availability) การสำรอง/การกู้คืนข้อมูล รวมทั้งการรักษาความปลอดภัยของข้อมูล
- การเรียนรู้ผลิตภัณฑ์ใหม่ๆ การปรับเปลี่ยนและการเข้ากันได้ของเวอร์ชันต่างๆ เพื่อนำมาปรับปรุงความสามารถของฐานข้อมูล รวมทั้งให้คำแนะนำกับทีมพัฒนาในเรื่องการพัฒนาฐานข้อมูลและการจัดการข้อมูล
- มีความเข้าใจในเรื่องฮาร์ดแวร์ ระบบปฏิบัติการ ระบบฐานข้อมูล สถาปัตยกรรมแบบหลายชั้น (Multi-tier Component Architecture) และการทำงานร่วมกันขององค์ประกอบเหล่านี้
- จัดทำเอกสารความต้องการของผู้ใช้งานในระดับภาพรวม
- ตรวจสอบทบทวนรายละเอียดการออกแบบ (detailed designs) และรายละเอียดการพัฒนาในระดับการใช้งานจริง (implementation details)

สิ่งสำคัญสำหรับนักสถาปัตยกรรมฐานข้อมูลคือ ต้องก้าวทันความหลากหลายทางด้านเทคโนโลยีของเครื่องมือต่างๆในผลิตภัณฑ์ฐานข้อมูล รูปแบบของฮาร์ดแวร์และระบบปฏิบัติการจากหลากหลายค่าย ที่สามารถนำมาประยุกต์ใช้เพื่อพัฒนาและปรับปรุงงานของธุรกิจ

1.5.3 ผู้ดูแลระบบฐานข้อมูล (Database Administrator: DBA)

ผู้ดูแลระบบฐานข้อมูล (DBA) เป็นผู้รับผิดชอบเรื่องการบำรุงรักษา ประสิทธิภาพการทำงาน ความถูกต้องสมบูรณ์และรักษาความปลอดภัยของฐานข้อมูล รวมถึงหน้าที่อื่นๆเช่นการมีส่วนร่วมในการวางแผนการจัดเก็บข้อมูล การพัฒนาโปรแกรมประยุกต์ และการแก้ไขปัญหาต่างๆที่เกิดขึ้น

การทำงานของผู้ดูแลระบบฐานข้อมูลจะแตกต่างออกไปตามลักษณะขององค์กรและหน้าที่ที่ได้รับมอบหมาย บางครั้งอาจมีหน้าที่ในการบำรุงรักษาฐานข้อมูลอย่างเดียว หรืออาจจะต้องเป็นผู้เชี่ยวชาญในการให้คำปรึกษาในเรื่องการพัฒนาแอพพลิเคชัน เกี่ยวกับการใช้งานฐานข้อมูล ซึ่งความรับผิดชอบส่วนใหญ่เป็นดังต่อไปนี้

- กำหนดสิทธิ์การใช้งานฐานข้อมูลของแต่ละผู้ใช้งาน ตรวจสอบการเข้าใช้ฐานข้อมูลของผู้ใช้ และดูแลตรวจสอบเรื่องระบบการรักษาความปลอดภัยฐานข้อมูล

- ตรวจสอบประสิทธิภาพการทำงานและจัดการค่าพารามิเตอร์ของฐานข้อมูลซึ่งเป็นปัจจัยที่ส่งผลต่อความเร็วของการดึงข้อมูลของผู้ใช้
- แปลงการออกแบบระดับแนวคิดเพื่อใช้สำหรับสร้างฐานข้อมูลตามที่วางแผนไว้
- ดูแลจัดการข้อมูลที่จัดเก็บอยู่เบื้องหลังและการเรียกใช้ข้อมูลของผู้ใช้ที่อยู่เบื้องหน้า
- แก้ไขปรับปรุงการออกแบบเชิงตรรกะจนสามารถแปลงมาให้เป็นแบบจำลองข้อมูล
- แก้ไขปรับปรุงการออกแบบทางกายภาพเพื่อให้สอดคล้องกับเนื้อที่การจัดเก็บข้อมูล
- ติดตั้งและทดสอบระบบจัดการฐานข้อมูลเมื่อต้องการปรับเปลี่ยนเวอร์ชัน
- รักษาป้องกันความปลอดภัยของข้อมูลให้เป็นไปตามมาตรฐาน
- จัดทำเอกสารที่เกี่ยวกับฐานข้อมูล กระบวนการ (Procedures) และคำอธิบายความหมายของข้อมูล (data dictionary)
- ควบคุมและกำหนดสิทธิ์การเข้าถึงข้อมูลการจัดการข้อมูล
- วางรูปแบบจัดการและทดสอบแผนการสำรองข้อมูลและการกู้คืนข้อมูล
- ตรวจสอบเนื้อที่สำหรับจัดเก็บการสำรองข้อมูล ขั้นตอนการสำรองข้อมูลและการกู้คืนข้อมูลสามารถทำงานได้อย่างถูกต้อง
- จัดทำแผนรองรับการเพิ่มขึ้นของข้อมูลที่จัดเก็บ
- ทำงานร่วมกันอย่างใกล้ชิดกับหัวหน้าส่วนงานไอที โปรแกรมเมอร์ที่มีการใช้งานฐานข้อมูล นักพัฒนาเว็บไซต์
- มีการติดต่อสื่อสารระหว่างผู้เชี่ยวชาญด้านเทคนิค ผู้ใช้ระบบงานและพนักงานระดับปฏิบัติงาน เพื่อดูแลตรวจสอบในเรื่องความถูกต้องของข้อมูลที่ใช้งานร่วมกันและการรักษาความปลอดภัยของข้อมูล
- ทำการทดสอบการทำงานของระบบเดิมและระบบที่ถูกติดตั้งใหม่

เนื่องจากภัยคุกคามข้อมูลจากแฮกเกอร์ที่มีจำนวนเพิ่มขึ้น และข้อมูลที่มีการจัดเก็บเป็นข้อมูลที่มีความสำคัญและละเอียดอ่อน ดังนั้นผู้ดูแลระบบฐานข้อมูลควรให้ความสำคัญในประเด็นเรื่องของความปลอดภัยของข้อมูลที่จัดเก็บและการกู้คืนข้อมูลเมื่อมีความเสียหายที่อาจจะเกิดขึ้นได้

1.5.4 นักพัฒนาโปรแกรมประยุกต์ (Application Developer)

นักพัฒนาโปรแกรมประยุกต์เป็นผู้พัฒนาโปรแกรมประยุกต์ที่มีการเข้าถึงข้อมูลในฐานข้อมูล นักพัฒนาโปรแกรมประยุกต์ต้องมีความรู้ต่อไปนี้

- โปรแกรมประยุกต์สำหรับการพัฒนาระบบงานที่มีการเข้าถึงข้อมูลในฐานข้อมูล (Integrated database application Development Environments: IDEs)
- ฐานข้อมูลที่สามารถทำงานร่วมกับโปรแกรมประยุกต์
- เครื่องมือการพัฒนาด้วยภาษา SQL (Structure Query Language)
- การตรวจสอบประสิทธิภาพการทำงานของฐานข้อมูลและการตรวจแก้จุดบกพร่อง

- สภาพแวดล้อมของแอพพลิเคชันเซิร์ฟเวอร์ การปรับใช้แอพพลิเคชัน การตรวจสอบประสิทธิภาพ การทำงานแอพพลิเคชันและการตรวจแก้จุดบกพร่อง

ตัวอย่าง เครื่องมือ IDE ของบริษัท ไอบีเอ็ม ได้แก่ IBM Data Studio ซึ่งช่วยให้นักพัฒนาสามารถเข้าถึงข้อมูลในฐานข้อมูล DB2 เช่น ตารางข้อมูล วิว อินเด็กซ์ ชุดคำสั่ง (stored procedures) การเขียนฟังก์ชันและเว็บเซอร์วิส นอกจากนั้นยังมีคุณสมบัติที่สามารถตรวจแก้จุดบกพร่อง (Debug) รวมถึงการพัฒนาด้วย SQL และ XQuery การทำงานร่วมกันระหว่างแอพพลิเคชันเซิร์ฟเวอร์ที่ต่างกัน เช่น WebSphere® นอกจากนี้ โปรแกรม DB2 ยังสามารถทำงานร่วมกับ Microsoft® Visual Studio ได้โดยมีชุดเครื่องมือที่สามารถจัดการเกี่ยวกับอ็อบเจ็กต์ของ DB2 (ตารางข้อมูล วิว ชุดคำสั่ง การเขียนฟังก์ชัน ฯลฯ) นักพัฒนาที่พัฒนาด้วย .NET ไม่จำเป็นต้องเปิดการทำงานสลับไปมาระหว่างสองโปรแกรม ซึ่งบทบาทและหน้าที่ความรับผิดชอบที่ได้อธิบายมานั้นเป็นการแบ่งแยกอย่างกว้าง ๆ ซึ่งแต่ละองค์กรจะมีการแบ่งบทบาทและหน้าที่ภายในองค์กรเอง

1.6 สรุป

ในบทนี้เราได้อธิบายถึงแนวคิดพื้นฐานเกี่ยวกับฐานข้อมูล และระบบจัดการฐานข้อมูล ด้วยคำจำกัดความที่ง่าย จากนั้นเราได้กล่าวถึงรูปแบบของข้อมูลและแบบจำลองข้อมูลแบบต่างๆ เช่นแบบจำลองเครือข่าย แบบจำลองแบบลำดับชั้นและแบบจำลองเชิงสัมพันธ์ ส่วนท้ายของบทกล่าวถึงผู้ที่มีบทบาทหน้าที่ที่เกี่ยวข้องกับฐานข้อมูล สำหรับบทต่อไปนั้นจะมีการอธิบายรายละเอียดเพิ่มเติมเกี่ยวกับแนวคิดฐานข้อมูลให้เพิ่มมากขึ้น

1.7 แบบฝึกหัด

1. เราสามารถเรียนรู้เพิ่มเติมเกี่ยวกับฐานข้อมูลด้วยแบบฝึกภาคปฏิบัติ DB2 Express-C สำหรับเซิร์ฟเวอร์ฐานข้อมูล DB2 เป็นฟรีเวอร์ชัน ซึ่งสามารถดาวน์โหลดได้ที่ ibm.com/db2/express
2. เราสามารถเรียนรู้เพิ่มเติมเกี่ยวกับ IDEs ด้วยแบบฝึกภาคปฏิบัติ IBM Data Studio เป็นฟรีเวอร์ชัน สามารถดาวน์โหลดได้ที่ ibm.com/db2/express

1.8 คำถามท้ายบท

1. ฐานข้อมูลคืออะไร
2. ระบบการจัดการฐานข้อมูลคืออะไร
3. ความแตกต่างระหว่างรูปแบบข้อมูลกับแบบจำลองข้อมูลคืออะไร
4. ข้อดีของแบบจำลองเชิงสัมพันธ์ที่ดีกว่าแบบจำลองอื่น ๆ คืออะไร
5. จงบอกหน้าที่หลักๆของ DBA มาสองข้อ
6. ข้อใดต่อไปนี้ไม่ใช่รูปแบบข้อมูล
   A. แบบจำลอง pureXML
   B. แบบจำลองเชิงสัมพันธ์
   C. แบบจำลองลำดับชั้น
   D. แบบจำลองเครือข่าย
   E. ไม่มีข้อใดถูก

7. วิวัฒนาการระบบการจัดการฐานข้อมูล ในเรื่องของ Optimization ควรคำนึงถึงอะไรเป็นสิ่งสำคัญ

A. ความพร้อมการใช้งาน (High availability)

B. ความปลอดภัย (Security)

C. ประสิทธิภาพการทำงาน (Performance)

D. การขยายต่อเติมระบบ(Scalability)

E. ไม่มีข้อถูก

8. ข้อใดต่อไปนี้ไม่ใช่วิวัฒนาการระบบการจัดการฐานข้อมูล

A. การกระจายข้อมูล (Distribution)

B. ความเป็นอิสระของข้อมูล (Data Independence)

C. การใช้งานข้อมูลร่วมกัน(Integration)

D. การจัดระเบียบการใช้งานโดยรวม(Federation)

E. ไม่มีข้อถูก

9. ในวิวัฒนาการระบบการจัดการฐานข้อมูล ในขั้นตอนใดควรใช้ pureXML

A. ความเป็นอิสระของข้อมูล

B. ส่วนขยายต่อเติม (Extensibility)

C. ปรับให้เหมาะสม (Optimization)

D. การใช้งานข้อมูลร่วมกัน

E. ไม่มีใดข้อถูก

10. DB2 บน Cloud มีอะไรใหม่ที่ถูกเพิ่มเข้ามา

A. Spatial Extender

B. Database Partitioning Feature

C. เทคโนโลยี pureXML

D. ถูกทุกข้อ

E. ไม่มีข้อใดถูก

2

บทที่ 2 – แบบจำลองข้อมูลเชิงสัมพันธ์

ในบทนี้จะกล่าวถึงพื้นฐานของแบบจำลองข้อมูลเชิงสัมพันธ์ ซึ่งประกอบด้วยแอตทริบิวต์ ทูเปิล รีเลชัน (relation) โดเมน (domains) โครงร่าง (schema) และคีย์ (key) รวมถึงอธิบายถึงความแตกต่างระหว่างการควบคุมข้อมูล (constraints) ประเภทต่างๆ และยังกล่าวถึงพีชคณิต (algebra) แคลคูลัส ที่เกี่ยวข้องกันกับ ข้อมูลเชิงสัมพันธ์ ซึ่งบทนี้จะเป็นพื้นฐานที่สำคัญสำหรับการออกแบบฐานข้อมูลซึ่งจะกล่าวในบทที่ 3 ได้แก่ แบบจำลองข้อมูลระดับแนวคิดและการแปลงแบบจำลองระดับแนวคิดไปเป็นโครงร่างฐานข้อมูลเชิงสัมพันธ์ และบทที่ 4 จะกล่าวถึงการออกแบบฐานข้อมูลเชิงสัมพันธ์ นอกจากนี้ยังจะเน้นการทำความเข้าใจเกี่ยวกับภาษา SQL ให้มากขึ้น สำหรับประเด็นหัวข้อสำคัญที่จะเรียนในบทนี้ประกอบด้วย

- ภาพรวมของแบบจำลองข้อมูลเชิงสัมพันธ์
- คำนิยามของแอตทริบิวต์ ทูเปิล รีเลชัน โดเมน แบบโครงร่างและคีย์
- การควบคุมข้อมูลของแบบจำลองเชิงสัมพันธ์
- พีชคณิตเชิงสัมพันธ์
- แคลคูลัสเชิงสัมพันธ์

2.1 ภาพรวมของแบบจำลองข้อมูลเชิงสัมพันธ์

แบบจำลองสำหรับข้อมูลสารสนเทศ (Information Models) เป็นการนำเอาข้อมูลที่อยู่ในโลกความเป็นจริง ที่มีความซับซ้อนมาแปลงให้เป็นโครงร่างการทำงาน (framework) ที่เข้าใจง่าย แบบจำลองข้อมูลจะนำเอาโครงสร้างข้อมูล คุณลักษณะของข้อมูล ความสัมพันธ์ระหว่างข้อมูล การควบคุมข้อมูล กฎการตรวจสอบความถูกต้องข้อมูลและอื่นๆที่เกี่ยวข้องในการเชื่อมโยงข้อมูล แบบจำลองข้อมูลนี้เป็นเสมือนเครื่องมือที่ช่วยให้นักออกแบบ นักพัฒนาโปรแกรมและผู้ใช้งานฐานข้อมูลสามารถสื่อสารเพื่อความเข้าใจระหว่างกัน แบบ

จำลองข้อมูลมีอยู่หลายแบบในปัจจุบันและมีคุณสมบัติที่แตกต่างกัน อย่างไรก็ดีในบทนี้จะเน้นที่แบบจำลองข้อมูลเชิงสัมพันธ์ซึ่งเป็นแบบจำลองที่มีการใช้งานอย่างแพร่หลายในทุกวันนี้ รูปที่ 2.1 นำเสนอส่วนประกอบหลักที่เกี่ยวข้องกับแบบจำลองข้อมูลเชิงสัมพันธ์

ส่วนประกอบของแบบจำลองข้อมูลเชิงสัมพันธ์ ซึ่งสามารถอธิบายในรายละเอียดต่อไปนี้

- ข้อมูลที่เกี่ยวข้องกับแนวคิดแบบจำลองข้อมูล เช่น แอตทริบิวต์ ทูเปิล โดเมน รีเลชัน โครงร่าง และคีย์
- การควบคุมข้อมูลแบบจำลองเชิงสัมพันธ์ เช่น entity integrity referential integrity semantic constraints ที่ใช้ในการควบคุมข้อมูลในฐานข้อมูลเชิงสัมพันธ์
- พีชคณิตเชิงสัมพันธ์ ซึ่งประกอบด้วย

  union

  intersection

  difference

  cartesian product

  selection

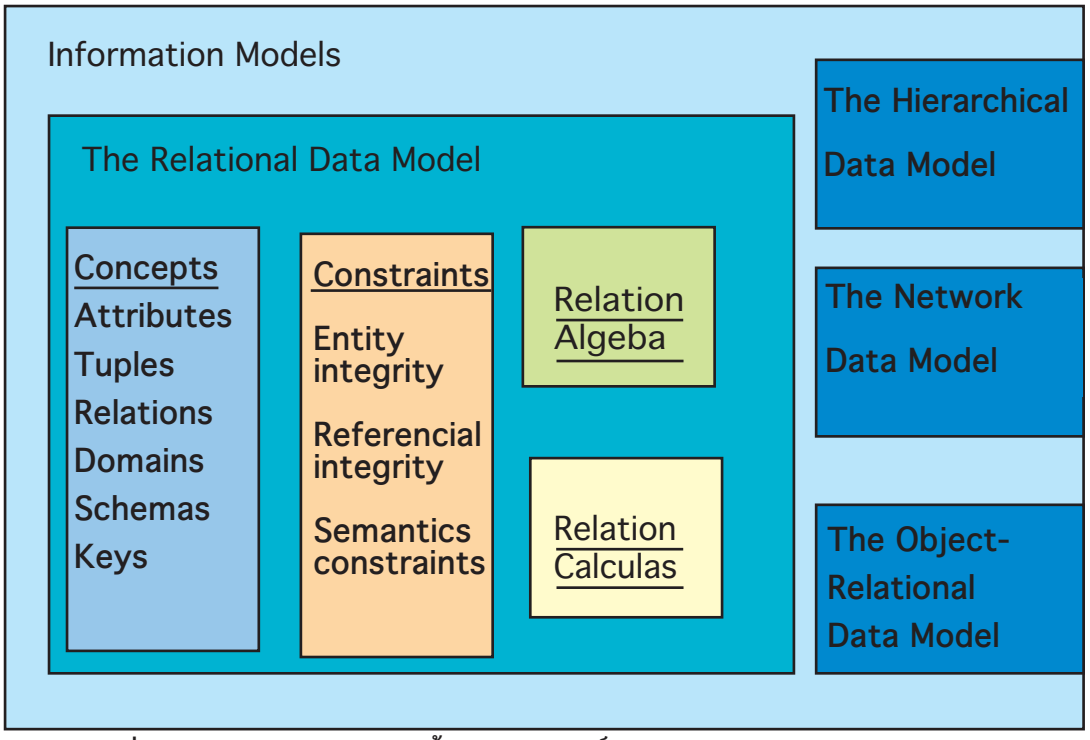
  projection

  join

  division

  ที่ช่วยในการจัดการความสัมพันธ์ ในแบบจำลองข้อมูลเชิงสัมพันธ์
- แคลคูลัสเชิงสัมพันธ์ ต่างจากพีชคณิตเชิงสัมพันธ์ จะเป็นการจัดการข้อมูลบางส่วน เป็นหลักการทางตรรกะคณิตศาสตร์เพื่อการจัดการข้อมูล

รูป 2.1 – ภาพรวมแบบจำลองข้อมูลเชิงสัมพันธ์ซึ่งอยู่ในกรอบของโครงสร้างระบบสารสนเทศ



รูปที่ 2.1 แสดงส่วนประกอบของแบบจำลองข้อมูลเชิงสัมพันธ์

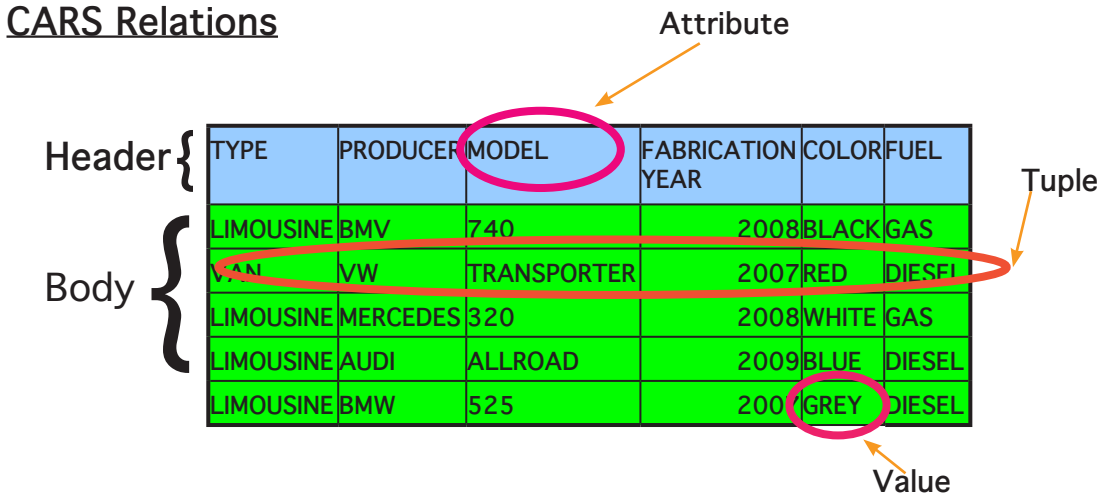
2.2 พื้นฐานแนวคิดของแบบจำลองข้อมูลเชิงสัมพันธ์

การสร้างแบบจำลองข้อมูลเชิงสัมพันธ์มีองค์ประกอบที่สำคัญหลายส่วนด้วยกันเช่น แอตทริบิวต์ โดเมน ทูเปิล รีเลชัน โครงร่างคีย์ แคนดิเดตคีย์ (candidate key) คีย์หลัก (primary key) และคีย์นอก (foreign key) ซึ่งจะได้อธิบายศัพท์เหล่านี้ในรายละเอียดต่อไป

2.2.1 แอตทริบิวต์ (Attributes)

แอตทริบิวต์ คือการกำหนดคุณลักษณะของข้อมูล ตัวอย่างเช่น บุคคล จะประกอบไปด้วยแอตทริบิวต์ คือ ชื่อ เพศ วันเดือนปีเกิด แอตทริบิวต์ ในมุมของแบบจำลองเชิงสัมพันธ์ คือ คอลัมน์ ในตารางข้อมูล หรือ ฟิลด์ ในแฟ้มข้อมูล

รูป 2.2 แสดงถึงแอตทริบิวต์ของรถยนต์ประกอบด้วย Type, Producer, Model, FabricationYear, Color, Fuel



รูปที่ 2.2 รีเลชันรถยนต์ (ตารางข้อมูลรถยนต์) ส่วนหัวตารางแสดงแอตทริบิวต์และส่วนรายละเอียดในตารางแต่ละแถวแสดงด้วยทูเพิล

2.2.2 โดเมน (Domains)

โดเมน คือเซตของค่าที่เป็นค่าเดี่ยวไม่สามารถแบ่งแยกย่อยลงไปได้อีก (atomic value) ซึ่งข้อมูลในโดเมนจะต้อง เป็นประเภทเดียวกัน ค่า (value) ถือว่าเป็นหน่วยเล็กที่สุดของข้อมูลในแบบจำลองเชิงสัมพันธ์ ตัวอย่างเช่น BMW Mercedes Audi และ VW เป็นค่า สำหรับแอตทริบิวต์ Producer ค่าของแอตทริบิวต์ที่จัดเก็บเป็น atomic value ซึ่งหมายถึงค่านั้นไม่สามารถแบ่งแยกย่อยลงไปได้อีก โดเมนสำหรับ Producer คือเซตของชื่อผู้ผลิตรถยนต์ที่เป็นไปได้ทั้งหมด แอตทริบิวต์จะมีความเกี่ยวข้องกับโดเมนเสมอ กล่าวคือ โดเมนเป็นการกำหนดเซตของค่าที่เป็นไปได้สำหรับแต่ละแอตทริบิวต์ และแอตทริบิวต์มากกว่าหนึ่งแอตทริบิวต์สามารถมีโดเมนที่เหมือนกันได้

โดเมนจะมีชื่อ และสามารถกำหนดข้อมูลอ้างอิงภายใน รวมถึงกำหนด dimension (จำนวนของค่าที่โดเมนนั้นมี) ตัวอย่างเช่น แอตทริบิวต์โดเมน Fuel มี ค่า เพียงแค่ 2 ตัวคือ GAS และ DIESEL เราสามารถมองโดเมน เหมือนเป็นการรวมกันของ ค่า (Value)

การดำเนินการที่สำคัญของโดเมนคือ หากมี 2 แอตทริบิวต์ที่มีโดเมนเหมือนกัน การดำเนินการเปรียบเทียบก็ใช้ค่าเดียวกันและเป็นสิ่งที่สมเหตุสมผล ในทางกลับกัน หาก 2 แอตทริบิวต์ใช้โดเมนที่ต่างกัน การเปรียบเทียบจะไม่สามารถทำได้

โดเมนเป็นแนวคิดหลักที่สำคัญ แต่โดยทั่วไปมักจะไม่ได้จัดเก็บอยู่ในฐานข้อมูล ดังนั้นจึงควรระบุถึงโดเมนให้เป็นส่วนหนึ่งของคำจำกัดความของฐานข้อมูล และควรกำหนดให้แอตทริบิวต์ สามารถอ้างอิงถึงโดเมนที่เกี่ยวข้อง ทั้งนี้จะทำให้ระบบคำนึงถึงผลที่จะเกิดขึ้นจากการเปรียบเทียบแอตทริบิวต์ดังที่ได้กล่าวมาข้างต้น

การกำหนดชื่อแอตทริบิวต์เป็นชื่อเดียวกับโดเมนเป็นกรณีที่ควรหลีกเลี่ยงเนื่องจากจะทำให้เกิดความสับสน อย่างไรก็ตามก็เป็นไปได้ที่จะนำเอาชื่อโดเมนไปต่อท้ายชื่อของแอตทริบิวต์เมื่อต้องการอธิบายรายละเอียดเพิ่มเติมในชื่อแอตทริบิวต์ เช่น Color_Character หมายถึงแอตทริบิวต์สีที่มีโดเมนเป็นตัวอักษร

2.2.3 ทูเปิล (Tuples)

ทูเปิล คือเซตของค่าที่อธิบายถึงลักษณะของข้อมูลในช่วงเวลาหนึ่ง ดังแสดงในรูป 2.2 ซึ่งแสดงถึง

ตัวอย่างของทูเปิล คำศัพท์อื่นที่ใช้แทนทูเปิลคือ แถว (row) ในตารางข้อมูล หรือ เรคอร์ดในแฟ้มข้อมูล

2.2.4 รีเลชัน (Relations)

รีเลชันเป็นส่วนหลักของฐานข้อมูลเชิงสัมพันธ์ จากความรู้เบื้องต้นเกี่ยวกับระบบฐานข้อมูล [2.1] รีเลชันบนโดเมน D1, D2, ..., Dn ประกอบด้วยส่วนหัว (heading) และส่วนตัวตาราง (body) หรือรายละเอียดที่เก็บข้อมูลในรีเลชัน ส่วนหัวจะประกอบด้วยเซตของแอตทริบิวต์ A1, A2, ..., An จากชุดแอตทริบิวต์ดังกล่าว แอตทริบิวต์ที่ Ai จะสอดคล้องกับโดเมน Di (i=1, 2, ..., n)

ส่วนตัวตารางหรือรายละเอียดข้อมูล ประกอบด้วยเซตของทูเปิลที่แปรผันตามเวลา เซตของทูเปิลจัดเก็บแอตทริบิวต์คู่กับค่า (Ai:vi) (i=1, 2, ..., n) เช่น คู่ของแอตทริบิวต์ Ai ในส่วนหัวตาราง จัดเก็บค่าเป็น (Ai:vi) vi เป็นค่าจากโดเมน Di ซึ่งมีความสัมพันธ์กับแอตทริบิวต์ Ai

ในรูปที่ 2.2 แสดงถึงรีเลชัน CARS ซึ่งส่วนหัวของรีเลชันประกอบด้วยแอตทริบิวต์ 6 แอตทริบิวต์ คือ Type, Producer, Model, Fabrication, Year, Color, Fuel แต่ละแอตทริบิวต์จะมีโดเมนที่สอดคล้องกัน ในส่วนของตัวตาราง รายละเอียดที่เก็บในรีเลชันจะประกอบด้วยเซตของทูเปิล (5 ทูเปิลแสดงดังรูป 2.2 แต่ละทูเปิลมีค่าที่แตกต่างกันในแต่ละช่วงเวลา) แต่ละทูเปิลประกอบไปด้วยเซตของแอตทริบิวต์จำนวน 6 แอตทริบิวต์ พร้อมกับค่า

ดีกรีของรีเลชัน (relation degree) เทียบเท่ากันกับจำนวนของแอตทริบิวต์ในรีเลชัน จากรูปที่ 2.2 มีดีกรีของรีเลชันเท่ากับ 6 สำหรับกรณีที่ดีกรีของรีเลชันเท่ากับ 1 เรียกว่า unary หรือดีกรีของรีเลชันเท่ากับ 2 เรียกว่า binary ดีกรีของรีเลชันเท่ากับ 3 เรียกว่า ternary และสุดท้ายดีกรีของรีเลชันเท่ากับ n เรียกว่า nary

รีเลชันคาร์ดินัลลิตี (relation cardinality) เทียบเท่ากันกับจำนวนของทูเปิลสำหรับรีเลชัน จากรีเลชันในรูปที่ 2.2 มีคาร์ดินัลลิตีเท่ากับ 5 คาร์ดินัลลิตีจะเปลี่ยนไปตามระยะเวลา แต่ดีกรีนั้นจะไม่มีการเปลี่ยนแปลงบ่อยนัก

จากรีเลชันที่กล่าวมาจะเห็นได้ว่าส่วนตัวตารางของรีเลชันหรือส่วนรายละเอียดของรีเลชันจะประกอบไปด้วยชุดของทูเปิล ณ เวลาหนึ่ง รีเลชันคาร์ดินัลลิตีอาจเท่ากับ m เมื่อเวลาผ่านไปรีเลชันเดิมอาจมีรีเลชันคาร์ดินัลลิตีเท่ากับ n เราจะเรียกสถานะของรีเลชัน ณ เวลาหนึ่ง ว่า รีเลชันอินสแตนซ์ หรือข้อมูลจริงของตารางความสัมพันธ์ (relation instance) ดังนั้นในช่วงชีวิตของรีเลชันจึงมีรีเลชันอินสแตนซ์หลายอินสแตนซ์ รีเลชันมีส่วนประกอบและคุณสมบัติที่สำคัญดังที่ได้อธิบายไว้ข้างต้น ดังนั้นสามารถสรุปคุณสมบัติของรีเลชันได้ 4 ข้อดังนี้

- ข้อมูลในแต่ละทูเปิลต้องไม่ซ้ำกันในรีเลชัน
- ทูเปิล ไม่มีการเรียงลำดับ (บนลงล่าง)
- แอตทริบิวต์ไม่มีการเรียงลำดับ (ซ้ายไปขวา)
- ค่าแอตทริบิวต์ทั้งหมดเป็นค่า Atomic

หมายเหตุ ศัพท์อย่างไม่เป็นทางการที่ใช้สำหรับเรียกรีเลชัน ว่า ‘ตาราง’ หรือ ‘แฟ้มข้อมูล’

2.2.5 โครงร่าง (Schemas)

โครงร่างของฐานข้อมูลเป็นรูปแบบมาตรฐานที่อธิบายถึงรายละเอียดของฐานข้อมูลเชิงสัมพันธ์และความสัมพันธ์ระหว่างรีเลชันที่อยู่ภายใน ในบทที่ 3 การออกแบบแบบจำลองข้อมูลในระดับแนวคิด (Concep-

tual data modeling) และบทที่ 4 การออกแบบฐานข้อมูลเชิงสัมพันธ์ (Relational database design) ในหัวข้อต่อไป จะได้เรียนรู้เกี่ยวกับโครงร่างสำหรับฐานข้อมูลเชิงสัมพันธ์ในรายละเอียด

2.2.6 คีย์ (Keys)

แบบจำลองข้อมูลเชิงสัมพันธ์ใช้คีย์เพื่อบ่งบอกถึงความแตกต่างระหว่างทูเปิลของรีเลชัน คีย์ถูกใช้สำหรับบังคับใช้เป็นกฎ และ/หรือ การควบคุมข้อมูลในฐานข้อมูล การควบคุมข้อมูลดังกล่าวใช้จัดการความสอดคล้องกันและความถูกต้องของข้อมูล ระบบจัดการฐานข้อมูล (DBMS) อนุญาตให้มีการใช้คีย์และเป็นส่วนที่ระบบจัดการฐานข้อมูลเชิงสัมพันธ์ใช้ในการตรวจสอบและรักษาความสอดคล้องกันและความถูกต้องของข้อมูลในฐานข้อมูล ซึ่งสามารถระบุคีย์ตามประเภทต่างๆ ดังนี้

2.2.6.1 แคนดิเดตคีย์ (Candidate Keys)

แคนดิเดตคีย์ คือสิ่งที่สามารถใช้ระบุทูเปิลที่ไม่ซ้ำกันในรีเลชัน ตามคำจำกัดความ รีเลชันหนึ่งมีแคนดิเดตคีย์อย่างน้อย 1 ตัว (คุณสมบัติหนึ่งของรีเลชัน) ในทางปฏิบัติแล้วรีเลชันส่วนมากจะมีแคนดิเดตคีย์มากกว่า 1 ตัว โดย C. J. Data [2.2] ให้คำนิยามสำหรับแคนดิเดตคีย์ ดังต่อไปนี้

รีเลชัน R ประกอบด้วยแอตทริบิวต์ A1, A2, ..., An เซตของ K=(Ai, Aj, ..., Ak) ในรีเลชัน R ถูกเรียกว่าแคนดิเดตคีย์ ถ้ามีคุณสมบัติ 2 ข้อต่อไปนี้

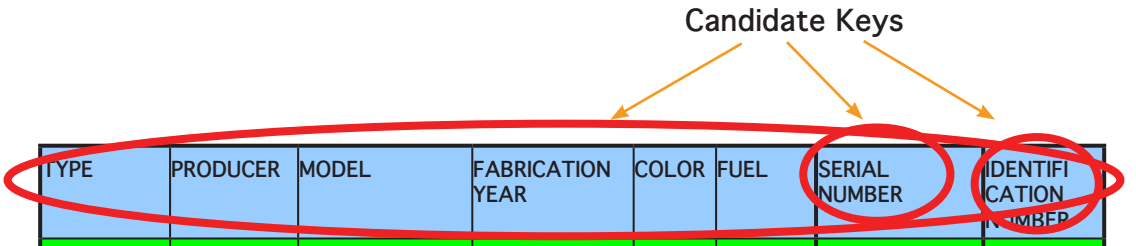
ยูนิคเนส (Uniqueness)

ณ เวลาหนึ่ง จะไม่สามารถมีทูเปิล 2 ทูเปิลในรีเลชัน R ที่มี ค่าของ Ai,Aj,...Ak ที่เหมือนกัน

มินิมอลลิตี้ (Minimality)

ต้องไม่มีแอตทริบิวต์ Ai,Aj,...Ak ที่อยู่นอกเซตของ K นอกจากจะยกเลิกคุณสมบัติยูนิคเนส หมายถึงเซตของ K ต้องเป็นเซตของแอตทริบิวต์ที่น้อยที่สุดที่สามารถระบุถึงทูเปิลได้

สำหรับทุกรีเลชันจะมีแคนดิเดตคีย์อย่างน้อย 1 ตัว เนื่องจากแคนดิเดตคีย์เป็นการรวมกันของแอตทริบิวต์ที่ทำให้ทูเปิลมีค่าไม่ซ้ำกัน (คุณสมบัติหนึ่งของรีเลชัน) แต่มักจะเลือก แอตทริบิวต์ไม่กี่ตัวในรีเลชันเพื่อนำมาเป็นแคนดิเดตคีย์ ตัวอย่างเช่น รีเลชัน CAR ที่แสดงในรูป 2.2 แคนดิเดตคีย์คือ K=(Type, Producer, Model, FabricationYear, Color, Fuel) ลองพิจารณาว่าเราสามารถมีรถหลายคันที่มีคุณลักษณะที่เหมือนกันได้ในรีเลชัน หากเราสร้างรีเลชัน CAR ใหม่โดยเพิ่มแอตทริบิวต์ SerialNumber (หมายเลขเครื่องยนต์) และ IdentificationNumber (ทะเบียนรถยนต์) ซึ่งประกอบด้วย 3 แคนดิเดตคีย์



TYPE	PRODUCER	MODEL	FABRICATION YEAR	COLOR	FUEL	SERIAL NUMBER	IDENTIFICATION NUMBER
LIMOUSINE	BMV	740	2008	BLACK	GAS	WBADL9105 GW65796	SB24MEA
VAN	VW	TRANSPORTER	2007	RED	DIESEL	QASMD8209 NF37590	AB08DGF
LIMOUSINE	MERCEDES	320	2008	WHITE	GAS	XEFAR2096 WM19875	SB06GHX
LIMOUSINE	AUDI	ALLROAD	2009	BLUE	DIESEL	AKLMD8064 MW79580	SB52MAG
LIMOUSINE	BMW	525	2007	GREY	DIESEL	QMXAS4390 WQ21998	AB02AMR

รูปที่ 2.3 รีเลชัน CARS และแคนดิเดตคีย์

แคนดิเดตคีย์ในบางครั้งเรียกว่า ยูนิคคีย์ (unique key) ยูนิคคีย์ถูกใช้ในภาษากำหนดโครงสร้าง (DDL) ที่มีการใช้พารามิเตอร์ UNIQUE ตามหลังชื่อของแอตทริบิวต์ ถ้ารีเลชันมีแคนดิเดตคีย์มากกว่าหนึ่งตัว หนึ่งในนั้นจะถูกเลือกให้เป็นคีย์หลัก (primary key) ที่เหลือจะถูกใช้เป็นคีย์ทดแทนเรียกว่า อัลเทอเนทคีย์ (alternate keys)

หมายเหตุ:

เพื่อตรวจสอบความถูกต้องของแคนดิเดตคีย์ คุณต้องคำนึงถึงรีเลชันอินสแตนซ์และความหมายของแอตทริบิวต์ เพื่อให้สามารถตรวจสอบความซ้ำซ้อนของข้อมูลในรีเลชัน

2.2.6.2 คีย์หลัก (Primary Key)

คีย์หลักเป็นสิ่งที่ใช้แสดงความเป็นหนึ่งเดียวของแต่ละทูเปิลในรีเลชัน ดังที่ได้กล่าวไว้ คีย์หลักคือหนึ่งในแคนดิเดตคีย์ที่ถูกเลือกของรีเลชันในฐานข้อมูลเพื่อระบุความเป็นหนึ่งเดียวของแต่ละทูเปิลในรีเลชัน รีเลชันในฐานข้อมูลควรมีการระบุคีย์หลักเสมอ

ระบบจัดการฐานข้อมูล (DBMS) สามารถกำหนดคีย์หลักในขั้นตอนการสร้างรีเลชัน (ตารางข้อมูล) ภาษาที่ใช้สำหรับกำหนดโครงสร้าง (DDL) จะใช้คำว่า PRIMARY KEY ในการกำหนดคีย์หลัก ตัวอย่างเช่น การสร้างรีเลชัน CAR ในรูปที่ 2.3 เป็นการเลือกแคนดิเดตคีย์ IdentificationNumber เพื่อเป็นคีย์หลัก โดยที่ค่าของแอตทริบิวต์ดังกล่าวต้องเป็นค่ายูนิคและไม่เป็นค่าว่าง (NOT NULL) สำหรับทุกทูเปิลในรีเลชัน

ในสถานการณ์จริง การสร้างรีเลชันอาจจะมีกรณีที่ไม่มีแอตทริบิวต์ใดๆเลยที่เป็นยูนิคดังตัวอย่างในภาพที่ 2.2 ซึ่งอาจจะสร้างความลำบากในการเลือก กรณีดังกล่าวคีย์หลักอาจจะเกิดขึ้นจากการรวมกันของทุกแอตทริบิวต์ในรีเลชัน ซึ่งอาจจะทำให้เกิดความไม่สะดวกในการใช้งานในทางปฏิบัติ และต้องเสียพื้นที่ในการจัดเก็บคีย์นั้นจำนวนมาก รวมถึงความยากในการจัดการ ความสัมพันธ์ระหว่างรีเลชัน สำหรับกรณีนี้การแก้ปัญหาสามารถทำได้โดยการสร้าง แอตทริบิวต์ใหม่ขึ้นมาชื่อว่า ID ซึ่งเป็นค่าที่ไม่มีความหมาย แต่ ID เป็นค่าที่กำหนดขึ้นมาเพื่อความเป็นหนึ่งเดียวและสามารถใช้เป็นคีย์หลัก ซึ่งแอตทริบิวต์ที่ถูกสร้างขึ้นมาเพื่อใช้เป็นคีย์หลักนี้เรียกว่าเซอโรเกตคีย์ (surrogate key) บางครั้งในบทความเอกสารเกี่ยวกับฐานข้อมูลจะเรียกว่าอาร์ติฟิเชียลคีย์ (artificial key) โดยปกติแล้วเซอโรเกตคีย์ (surrogate key) จะเป็นตัวเลขที่มีการเพิ่มหรือลดค่าโดย

อัตโนมัติ (เพิ่มหรือลดทีละ 1)

2.2.6.3 คีย์นอก (Foreign keys)

คีย์นอกเป็นแอตทริบิวต์หรือการรวมตัวของแอตทริบิวต์ในรีเลชัน R2 ที่มีค่าตรงกันกับคีย์หลักในรีเลชัน R1 โดยใช้สำหรับการสร้างความสัมพันธ์ระหว่างรีเลชัน เพื่อให้ R1 และ R2 มีความสัมพันธ์กัน ซึ่งคีย์นอกและคีย์หลักที่เกี่ยวข้องกันควรกำหนดให้อยู่ในกรอบโดเมนเดียวกัน สำหรับตัวอย่างในรูป 2.4 มีรีเลชันชื่อ OWNERS ซึ่งเก็บข้อมูลเจ้าของรถ และมีความสัมพันธ์กับรีเลชัน CARS

Primary key

Foriegn key

TYPE	PRODUCER	MODEL	FABRICATION YEAR	COLOR	FUEL	SERIAL NUMBER	IDENTIFICATION NUMBER
LIMOUSINE	BMV	740	2008	BLACK	GAS	WBADL9105 GW65796	SB24MEA
VAN	VW	TRANSPORTER	2007	RED	DIESEL	QASMD8209 NF37590	AB08DGF
LIMOUSINE	MERCEDES	320	2008	WHITE	GAS	XEFAR2096 WM19875	SB06GHX
LIMOUSINE	AUDI	ALLROAD	2009	BLUE	DIESEL	AKLMD8064 MW79580	SB52MAG
LIMOUSINE	BMW	525	2007	GREY	DIESEL	QMXAS4390 WQ21998	AB02AMR

รูป 2.4 – รีเลชัน OWNERS และคีย์หลักและคีย์นอกของรีเลชัน

IdentificationNumber เป็นคีย์นอกของรีเลชัน OWNERS ที่ใช้อ้างอิงไปถึง IdentificationNumber ซึ่งเป็นคีย์หลักของรีเลชัน CARS ในลักษณะดังกล่าวทำให้เราทราบว่าใครเป็นเจ้าของรถคันไหน

ความสัมพันธ์คีย์นอกไปยังคีย์หลัก (Foreign-to-primary-key) ที่ตรงกันเป็นการอ้างอิงรีเลชันหนึ่งไปยังอีกรีเลชันหนึ่ง เหมือนเป็นกาวที่ยึดโยงรีเลชันต่างๆไว้ด้วยกันในฐานข้อมูล อีกนัยหนึ่งสามารถกล่าวได้ว่าความสัมพันธ์คีย์นอกไปยังคีย์หลักเป็นความสัมพันธ์ที่เชื่อมโยงกันระหว่างทูเปิล แต่สิ่งที่ควรระวังคือไม่ใช่ทุกความสัมพันธ์จะสามารถแสดงด้วยความสัมพันธ์คีย์นอกไปยังคีย์หลักได้เสมอ สำหรับภาษาที่ใช้ในการกำหนดโครงสร้าง (DDL) จะใช้คีย์เวิร์ดว่า FOREIGN KEY ในการกำหนดคีย์นอก โดยต้องมีการระบุว่าเชื่อมโยงไปยังคีย์หลักชื่ออะไรและรีเลชันไหน

2.3 การควบคุมข้อมูลของแบบจำลองข้อมูลเชิงสัมพันธ์

ในการสร้างแบบจำลองข้อมูลเชิงสัมพันธ์ กฎควบคุมความถูกต้อง (integrity rules) หรือเงื่อนไขการควบคุมข้อมูล (constraint) เป็นส่วนหนึ่งที่สำคัญในการกำหนดความถูกต้องของข้อมูล ถ้าต้องการให้ฐานข้อมูลเชิงสัมพันธ์มีความถูกต้องเชิงโครงร่าง (schema level) โดยการควบคุมข้อมูลถูกใช้ในระดับการออกแบบโครงร่างฐานข้อมูล ซึ่งถ้าผู้ใช้พยายามที่จะละเมิดการควบคุมข้อมูลดังกล่าวระบบจะทำการปฏิเสธการดำเนินการนั้นหรือหาทางเลือกอื่นให้กับผู้ใช้เพื่อไม่ให้กระทบต่อข้อมูลในฐานข้อมูล ในลำดับต่อไปจะอธิบายถึงการควบคุมข้อมูลต่างๆที่ใช้ในฐานข้อมูลเชิงสัมพันธ์

2.3.1 การควบคุมข้อมูลเอนทิตีอินทิกริตี (Entity integrity constraints)

การควบคุมข้อมูลเอนทิตีอินทิกริตี เป็นการกำหนดว่าแอตทริบิวต์ที่ถูกใช้เป็นคีย์หลักในแต่ละรีเลชันต้อง

ไม่อนุญาตให้เป็นค่าว่าง (null values) คำว่า Null หมายถึง ไม่มีค่า (property inapplicable) หรือค่าที่ไม่รู้จักมาก่อน (information unknown) Null เป็นเหมือนเครื่องหมายแสดงการขาดหายไปของค่า หรือเป็นค่าที่ไม่สามารถระบุได้ ตัวอย่างเช่น ค่าว่างในแอตทริบิวต์สีของรีเลชัน CAR หมายถึงว่าในขณะนั้นเราไม่สามารถทราบได้ว่ารถคันนั้นสีอะไร การใช้การควบคุมข้อมูลเอนทิตีอินทิกริตี แสดงรายละเอียด ได้ดังต่อไปนี้

- รีเลชันของฐานข้อมูลเชิงสัมพันธ์สอดคล้องกันกับเอนทิตี้ โดยคำจำกัดความ เราสามารถมองเห็นความแตกต่างของเอนทิตี้ได้ ทำให้เราสามารถระบุถึงความเป็นหนึ่งเดียวของเอนทิตี้
- คีย์หลัก ทำหน้าที่บ่งชี้ความเป็นหนึ่งเดียวของทูเปิลในรีเลชันในแบบจำลองเชิงสัมพันธ์
- คีย์หลักที่เป็นค่าว่างจึงขัดแย้งกับสิ่งที่กล่าวมา 2 ข้อ เพราะไม่สามารถระบุทูเปิลในรีเลชันได้

2.3.2 การควบคุมข้อมูลเรเฟอเรนเชียลอินทริกริตี (Referential integrity)

การควบคุมข้อมูลสำหรับเรเฟอเรนเชียลอินทริกริตีเป็นการกำหนดเงื่อนไขสำหรับความสัมพันธ์ของรีเลชัน กล่าวว่า ถ้ารีเลชัน R2 มีคีย์นอกคือ FK อ้างอิงหรือตรงกับคีย์หลัก PK ของรีเลชันอื่นคือ รีเลชัน R1 ดังนั้นทุก FK ใน R2 ต้องเท่ากับค่า PK ใน R1 เสมอ หรือเป็นค่าว่าง (ค่าแอตทริบิวต์ที่เป็น FK เป็นค่าว่าง) เรเฟอเรนเชียลอินทริกริตีมีที่มาได้จากหลายสาเหตุดังต่อไปนี้

- ถ้าทูเปิล t2 จากรีเลชัน R2 อ้างอิงถึงบางทูเปิล t1 จากรีเลชัน R1 ดังนั้นต้องมีค่า t1 อยู่ก่อน ไม่เช่นนั้นการอ้างอิงก็ไม่สามารถทำได้
- ดังนั้นค่าของคีย์นอกต้องตรงกับค่าของคีย์หลัก ในการอ้างอิงผ่านความสัมพันธ์ คีย์นอกจึงควรเป็นค่าที่ไม่ใช่ค่าว่าง
- บางครั้งในทางปฏิบัติจะอนุญาตให้คีย์นอกเป็นค่าว่างได้

ตัวอย่างเช่น รีเลชัน OWNERS มีคีย์นอกคือ IdentificationNumber ค่าของแอตทริบิวต์ดังกล่าวที่เป็นคีย์นอกต้องเชื่อมความสัมพันธ์กับค่าในรีเลชัน CARS เพราะบุคคลต้องเป็นเจ้าของรถยนต์ ถ้าคีย์นอกเป็นค่าว่างแสดงว่าบุคคลนั้นไม่ได้เป็นเจ้าของรถยนต์ แต่ก็สามารถที่จะซื้อรถยนต์มาเป็นเจ้าของได้ สำหรับการระบุค่าคีย์นอกในฐานข้อมูลนั้น นักออกแบบฐานข้อมูลต้องพิจารณาประเด็นสำคัญ 3 ข้อ ดังต่อไปนี้

1. สามารถยอมรับค่าว่าง (Null Value) ได้หรือไม่ ตัวอย่างเช่นกรณีสำหรับผู้ที่เป็นเจ้าของรถอาจจะไม่ทราบว่ารถยนต์ของตนมีสีอะไร สำหรับคำตอบในข้อนี้นักออกแบบฐานข้อมูลไม่ต้องการให้เป็นค่าว่าง แต่ในความเป็นจริงอาจเกิดขึ้นโดยการมีการจัดเก็บค่าว่างในฐานข้อมูล

2. ประเด็นที่ต้องพิจารณาคือจะเกิดอะไรขึ้นหากคีย์หลักที่ คีย์นอกอ้างอิงถึงถูกลบไป ตัวอย่างเช่น เกิดการลบรถยนต์ ที่เป็นเจ้าของโดยบุคคล โดยปกติการกระทำดังกล่าว มี 3 แนวทางที่เป็นไปได้

   - CASCADE การดำเนินการลบแบบ cascades จะทำการลบทูเปิลที่เกี่ยวข้องหรืออ้างอิงไปถึงด้วย (ทูเปิลที่มีความสัมพันธ์ผ่านคีย์นอก) ในกรณีนี้ถ้ารถยนต์ถูกลบทูเปิลของเจ้าของรถก็จะถูกลบทิ้งด้วย
   - RESTRICT การดำเนินการลบแบบ restricted ทำการลบข้อมูลเมื่อไม่มีการอ้างอิงถึงทูเปิล (จะทำการยกเลิกการลบทันทีในกรณีที่มีการอ้างอิงข้อมูลผ่านคีย์นอก) ในกรณีนี้สามารถลบรถยนต์ได้หากรถยนต์นั้นไม่มีใครเป็นเจ้าของ
   - NULLIFIES จะเปลี่ยนค่าคีย์นอกให้เป็นค่าว่างในกรณีที่มีการอ้างอิงและทูเปิลที่เป็นคีย์หลักจะถูก

ลบ (ในกรณีนี้ไม่สามารถทำได้หากไม่มีการกำหนดให้คีย์นอกยอมรับค่าว่าง) ในกรณีนี้สามารถลบรถยนต์ได้หาก ค่าแอตทริบิวต์ IdentificationNumber ในรีเลชัน OWNERS ถูกกำหนดเป็นค่าว่าง

3. เกิดอะไรขึ้นหากมีการเปลี่ยนแปลงค่า (update) คีย์หลักที่มีค่าคีย์นอกอ้างอิงถึง

- CASCADE การดำเนินการอัพเดตแบบ cascades ทำการอัพเดตค่าของคีย์นอกที่เกี่ยวข้องหรืออ้างอิงมายังทูเปิล(รวมไปถึงทูเปิลจากความสัมพันธ์ของคีย์นอก) ในกรณีนี้ถ้า identification number ของรถยนต์ถูกอัพเดต identification number ของเจ้าของรถในรีเลชัน OWNER ก็จะถูกอัพเดตตามไปด้วย
- RESTRICT การดำเนินการอัพเดตแบบ restricted ทำการอัพเดตข้อมูลเมื่อไม่มีการอ้างอิงถึงทูเปิล (ทำการยกเลิกการอัพเดตหากมีการอ้างอิง) ในกรณีนี้ identification number ของรีเลชัน CARS จะถูกปรับปรุง เมื่อไม่มีบุคคลใดเป็นเจ้าของรถยนต์คันดังกล่าวเลย
- NULLIFIES จะเปลี่ยนค่าคีย์นอกให้เป็นค่าว่างในกรณีที่มีการอ้างอิงและทูเปิลที่มีคีย์หลักจะถูกอัพเดต (ในกรณีนี้ไม่สามารถทำได้หากไม่มีการกำหนดให้คีย์นอกยอมรับค่าว่าง) ในกรณีนี้สามารถอัพเดต IdentificationNumber ของรถยนต์หลังจากค่าแอตทริบิวต์ IdentificationNumber ในรีเลชัน OWNERS ถูกกำหนดเป็นค่าว่าง

2.3.3 การควบคุมข้อมูลซีเมนทิคอินทิกริตี (Semantic integrity constraints)

ซีเมนทิคอินทิกริตีเป็นการควบคุมข้อมูลที่แสดงถึงความถูกต้องของความหมายข้อมูล ขอบเขตและชนิดของข้อมูล ตัวอย่างเช่น แอตทริบิวต์ เลขที่ถนน จากรีเลชัน OWNERS ต้องเป็นเลขจำนวนเต็มบวกเพราะในความจริงนั้นเลขที่ถนนต้องเป็นเลขจำนวนเต็มบวกเท่านั้น ซีเมนทิคอินทิกริตีถือเป็นสิ่งที่ควรปฏิบัติเพื่อความถูกต้องในสถานะของรีเลชันตามความต้องการการจัดเก็บข้อมูลของฐานข้อมูล ถ้าผู้ใช้พยายามที่จะละเมิดการควบคุมข้อมูลดังกล่าว ระบบจะปฏิเสธการดำเนินการดังกล่าวทันที หรือดำเนินการอย่างอื่น เพื่อให้คงความถูกต้องของข้อมูลในฐานข้อมูล ดังนั้นจึงต้องมีภาษาในการจัดการความถูกต้อง รวมไปถึงการจัดการไม่ให้ดำเนินการใดๆกับข้อมูลโดยพลการและอำนวยความสะดวกสำหรับการดำเนินการที่เหมาะสม

การควบคุมข้อมูลซีเมนทิคอินทิกริตี โดยปกติเป็นหน้าที่ของผู้ดูแลฐานข้อมูลที่จะทำการระบุการควบคุมข้อมูลนี้เพื่อใช้จัดการฐานข้อมูล โดยจัดทำไว้ในพจนากรมข้อมูล (system catalog หรือ dictionary) ระบบจัดการฐานข้อมูลจะทำหน้าที่เฝ้าดูการทำงานของผู้ใช้ให้เป็นไปตามการควบคุมข้อมูล ระบบจัดการฐานข้อมูลเชิงสัมพันธ์อนุญาตให้มีการควบคุมข้อมูลซีเมนทิคอินทิกริตี สำหรับการจัดการความถูกต้องของข้อมูล เช่นการควบคุมข้อมูลโดเมน (domain constraint) การควบคุมข้อมูลค่าว่าง (Null constraint) การควบคุมข้อมูลการตรวจสอบข้อมูลซ้ำ (unique constraint) และการควบคุมข้อมูลการตรวจสอบเงื่อนไข (check constraint)

2.3.3.1 การควบคุมข้อมูลโดเมน (Domain constraint)

การควบคุมข้อมูลโดเมนเปรียบเสมือนเป็นการกำหนดคุณลักษณะของแอตทริบิวต์ที่ปรากฏในรีเลชัน การควบคุมข้อมูลโดเมนจะเป็นการระบุค่าเพื่อเป็นการตรวจสอบว่าค่านั้นอยู่ในเซตที่กำหนดไว้ในโดเมนหรือไม่ ตัวอย่างเช่น แอตทริบิวต์ โดเมน ชื่อ Street ในรีเลชัน OWNERS คือ CHAR(20) เนื่องจากชื่อถนนเป็นตัวอักษร และแอตทริบิวต์ โดเมน Number เป็น NUMERIC เพราะเลขที่ถนนเป็นตัวเลข

การควบคุมข้อมูลโดเมน อาจจะเป็นการควบคุมข้อมูลแบบกำหนดชื่อ (namely format) และการควบคุมข้อมูลแบบขอบเขตของข้อมูล (range) การควบคุมข้อมูลแบบกำหนดชื่อ จะคล้ายกับการกำหนดรูป

แบบของข้อมูล ตัวอย่างเช่นค่าของแอตทริบิวต์ IdentificationNumber จะต้องอยู่ในรูปแบบ XX99XXX โดยที่ X แทนตัวอักษร และ 9 แทนตัวเลข ส่วนการควบคุมข้อมูลแบบขอบเขตของข้อมูล ค่าของแอตทริบิวต์ต้องอยู่ในช่วงที่กำหนดไว้เท่านั้น ตัวอย่างเช่น ค่าของแอตทริบิวต์ FabricationYear ควรมีค่าระหว่าง 1950 ถึง 2010

2.3.3.2 การควบคุมข้อมูลค่าว่าง (Null value)

การควบคุมข้อมูลค่าว่างเป็นการระบุให้แอตทริบิวต์ไม่สามารถเป็นค่าว่าง สำหรับทูเปิลทุกตัวที่จัดเก็บไว้ในรีเลชัน ซึ่งจะเป็นการกำหนดให้แอตทริบิวต์มีค่าเสมอภายใต้ขอบเขตของข้อมูลในโดเมนของแอตทริบิวต์ ตัวอย่างเช่น แอตทริบิวต์ FirstName และ LastName ต้องไม่เป็นค่าว่าง นั่นย่อมหมายถึงว่าเจ้าของรถยนต์แต่ละคันต้องมีชื่อและนามสกุล

การควบคุมข้อมูลค่าว่างใช้คำว่า NOT NULL ชื่อแอตทริบิวต์ที่กำหนดจะต้องตามด้วยคำว่า NOT NULL รวมถึงการใช้คำว่า WITH DEFAULT เป็นทางเลือกให้ระบบสามารถระบุค่าเริ่มต้นให้กับแอตทริบิวต์ในกรณีการเพิ่มค่าเข้ามาแล้วค่านั้นเป็นค่าว่าง WITH DEFAULT ใช้สำหรับค่าประเภทตัวเลข (integer decimal float ฯลฯ) และ ค่าประเภทเวลา (date time timestamp) สำหรับค่าประเภทอื่นๆ เช่น character ที่จะใช้เป็นค่าเริ่มต้นให้ระบุค่าดังกล่าวในภาษากำหนดโครงสร้าง (Data Definition Language: DDL)

2.3.3.3 การควบคุมข้อมูลยูนิค (Unique constraint)

การควบคุมข้อมูลยูนิคเป็นการระบุให้ค่าแอตทริบิวต์ที่จัดเก็บไม่ซ้ำกัน ซึ่งจำเป็นต้องพิจารณาถึงลักษณะของข้อมูลที่จัดเก็บว่าในกรณีใดบ้างที่ค่าที่จัดเก็บนั้นจะไม่มีค่าที่ซ้ำกัน ตัวอย่างเช่นสำหรับรีเลชัน CARS ซึ่งมีแอตทริบิวต์ชื่อ SerialNumber ค่าที่จัดเก็บจะต้องเป็นค่ายูนิคหรือเป็นค่าที่ไม่ซ้ำ เนื่องจากเป็นไปไม่ได้ที่รถยนต์จะมีค่าหมายเลขเครื่องยนต์ที่เหมือนกัน 2 คัน การกำหนดค่ายูนิคทำได้โดยระบุคีย์เวิร์ดคำว่า UNIQUE ตามหลังชื่อแอตทริบิวต์

หมายเหตุ:

ค่าว่างไม่ได้เป็นส่วนหนึ่งของโดเมนใดๆ เพราะฉะนั้นการเปรียบเทียบใน SQL จะส่งค่ากลับเป็น unknown หรือค่าที่ไม่ทราบ เมื่อมีค่าว่างในการดำเนินการใดเข้ามา ดังนั้นการจัดการกับค่าว่างดังกล่าว SQL มีเครื่องมือจัดการในกรณีที่ อนุญาตให้เป็นค่าว่าง (IS NULL) และไม่อนุญาตให้เป็นค่าว่าง (IS NOT NULL) เพื่อใช้ในการตรวจสอบค่าว่าเป็นค่าว่างหรือไม่ ตารางสรุปด้านล่างเป็นการจัดการค่าว่าง (“Unknown”) โดย SQL ฐานข้อมูลของผู้ผลิตรายอื่นหรือรูปแบบอื่นก็จะมีวิธีการจัดการกับค่าว่างที่แตกต่างกันออกไป

A	B	A OR B	A AND B	A = B
True	True	True	True	True
True	False	True	False	False
True	Unknown	True	Unknown	Unknown
False	True	True	False	False
False	False	False	False	False
False	Unknown	Unknown	False	False
Unknown	True	True	Unknown	Unknown
Unknown	False	Unknown	False	False
Unknown	Unknown	Unknown	Unknown	Unknown

A	B
True	True
True	False
True	Unknown

2.3.3.4 การควบคุมข้อมูลตามเงื่อนไข (Check constraint)

การควบคุมข้อมูลการตรวจสอบเงื่อนไขเป็นการตรวจสอบข้อมูลภายในรีเลชัน โดยที่การควบคุมข้อมูลนี้จะเกิดขึ้นทุกครั้งเมื่อมีการเกี่ยวข้องกับข้อมูล ลักษณะของการทำงานจะมีการตรวจสอบถึงเงื่อนไข (predicate) ที่กำหนดไว้ ซึ่งระบบจะทำการตรวจสอบทุกครั้งว่าเป็นไปตามการควบคุมข้อมูลที่ระบุไว้หรือไม่ในกรณีที่ข้อมูลมีการเปลี่ยนแปลง หากถูกต้องตามการควบคุมข้อมูลก็สามารถดำเนินการต่อไปได้ ตัวอย่างเช่น

- เงินเดือนของพนักงานจะต้องไม่เกินเงินเดือนของผู้จัดการ
- หัวหน้าส่วนงานต้องดูแลพนักงานในแผนกไม่เกินกว่า 20 คน

สำหรับรีเลชันก่อนหน้าที่มีการยกตัวอย่างไปเช่นรีเลชัน CARS แอตทริบิวต์ fabrication year (ปีที่ผลิต) จะมากกว่า current year (ปีที่เป็นเจ้าของรถยต์) ไม่ได้

ประเภทของการควบคุมข้อมูลนี้สามารถระบุไว้ในฐานข้อมูลโดยใช้คำว่า CHECK หรือใช้ทริกเกอร์ (trigger) การควบคุมข้อมูลการตรวจสอบเงื่อนไข จะถูกตรวจสอบโดยระบบก่อนและหลังการดำเนินงานต่อไปนี้ คือ การเพิ่ม การลบ และการปรับปรุง

2.4 พีชคณิตเชิงสัมพันธ์

พีชคณิตเชิงสัมพันธ์เป็นแนวทางการดำเนินการสำหรับจัดการรีเลชัน โดยที่แต่ละตัวดำเนินการของพีชคณิตเชิงสัมพันธ์จะนำเข้ารีเลชันหนึ่งหรือสองรีเลชันแล้วได้ผลลัพธ์เป็นรีเลชันที่สร้างขึ้นมาใหม่ ซึ่ง Codd [2.3] ได้กำหนดตัวดำเนินการ 8 ตัว ประกอบด้วย 2 กลุ่มโดยแต่ละกลุ่มมี 4 ตัวดำเนินการ ดังนี้

- ตัวดำเนินการที่เกี่ยวกับเซต คือ Union, Intersection, Difference, และ Cartesian product
- ตัวดำเนินการที่เกี่ยวกับรีเลชัน คือ Select , Project, Join, และ Divide

2.4.1 Union

Union เป็นการดำเนินการที่เชื่อมรีเลชัน 2 ตัว เข้าด้วยกัน โดย R1 UNION R2 คือเซตของทุกทูเปิล t ของ R1 หรือ R2 หรือ ของทั้งคู่ ซึ่งทั้งสองรีเลชันสามารถทำการ union กันได้ (union-compatible) รีเลชันทั้งสองต้องมีดีกรีที่เท่ากันและ มีโดเมนของแอตทริบิวต์ที่เหมือนกัน

- สัญลักษณ์ที่ใช้แทน Union คือ U
- Union เป็นการดำเนินการที่มีคุณสมบัติสามารถสลับที่ได้และเปลี่ยนกลุ่มได้

รูปที่ 2.5 แสดงการดำเนินการ Union ตัวที่ถูกดำเนินการคือรีเลชัน R1 และ R2 ผลลัพธ์ได้ออกมาเป็นรีเลชัน R3 ประกอบด้วยทูเปิลจำนวน 5 แถว

R1

Name	Age	Sex
A	20	M
C	21	M
B	21	F

R2

Name	Age	Sex
D	20	F
A	20	M
E	21	F

R3 = R1 U R2

Name	Age	Sex
A	20	M
C	21	M
B	21	F
D	20	F
E	21	F

รูปที่ 2.5 – ตัวอย่างของการดำเนินการ UNION รีเลชัน: R1 และ R2

2.4.2 Intersection

Intersection เป็นการดำเนินการกับรีเลชันโดยที่ R1 INTERSECT R2 จะได้ผลลัพธ์เป็นเซตของทุกทูเปิล t ที่เหมือนกันของรีเลชันทั้งคู่คือ R1 และ R2 สัญลักษณ์ที่ใช้แทนสำหรับการดำเนินการ intersect คือ ∩

intersect เป็นการดำเนินการที่มีคุณสมบัติสามารถสลับที่ได้และเปลี่ยนกลุ่มได้

รูปที่ 2.6 แสดงการดำเนินการ INTERSECT ตัวที่ถูกดำเนินการคือรีเลชัน R1 และ R2 ผลลัพธ์ได้ออกมาเป็นรีเลชัน R3 มีทูเปิลจำนวน 1 แถว

R1

Name	Age	Sex
A	20	M
C	21	M
B	21	F

R2

Name	Age	Sex
D	20	F
A	20	M
E	21	F

R3 = R1 ∩ R2

Name	Age	Sex
A	20	M

รูปที่ 2.6 ตัวอย่างของการดำเนินการ INTERSECT รีเลชัน R1 และ R2

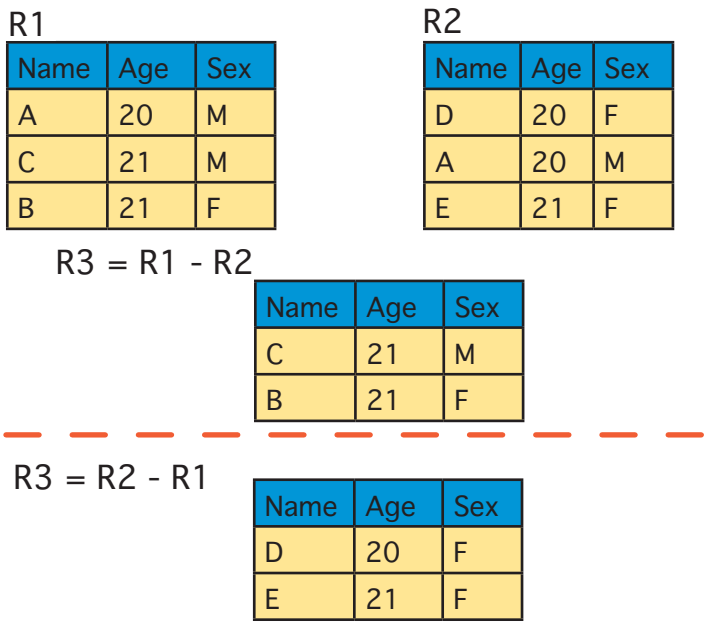
2.4.3 Difference

Difference เป็นการดำเนินการของรีเลชันโดยที่ R1 MINUS R2 คือเซตของทุกทูเปิลของรีเลชัน R1 ที่

ไม่อยู่ใน R2 สัญลักษณ์ที่ใช้แทนสำหรับการดำเนินการ difference คือ '–'

Difference เป็นการดำเนินการที่ไม่สามารถสลับที่และเปลี่ยนกลุ่มได้

รูปที่ 2.7 ตัวอย่างการดำเนินการของ DIFFERENCE ระหว่างตัวถูกดำเนินการรีเลชัน R1 และ R2 ผลลัพธ์เป็น R3 ที่มีทูเปิล 2 แถว ซึ่งผลลัพธ์ของ R1 - R2 จะได้ผลลัพธ์ที่แตกต่างจาก R2 - R1



รูปที่ 2.7 ตัวอย่างการดำเนินการ DIFFERENCE ระหว่างรีเลชัน R1 และ R2

2.4.4 Cartesian product

Cartesian product ระหว่างรีเลชัน R1 และ R2 โดยใช้ R1 TIMES R2 คือ เซตของทุกทูเปิล t โดยที่ t เกิดจากการเชื่อมต่อกันของทูเปิล r ของรีเลชัน R1 กับทูเปิล s ของรีเลชัน R2 การเชื่อมต่อกันของทูเปิล r = $(r_1, r_2, ..., r_m)$ และทูเปิล s = $(s_{m+1}, s_{m+2}, ..., s_{m+n})$ คือทูเปิล t = $(r_1, r_2, ..., r_m, s_{m+1}, s_{m+2}, ..., s_{m+n})$

R1 และ R2 ไม่มีความจำเป็นที่ต้องเหมือนกัน (ไม่จำเป็นต้อง union กันได้ คือแอตทริบิวต์ไม่ต้องเหมือนกัน) สัญลักษณ์ที่ใช้แทนสำหรับการดำเนินการ Cartesian product คือ '×'

ถ้า R1 มีดีกรีเท่ากับ n และมีคาร์ดินัลลิตีเท่ากับ N1 และ R2 มีดีกรีเท่ากับ m มีคาร์ดินัลลิตีเท่ากับ N2 จะได้ผลลัพธ์เป็นรีเลชัน R3 มีดีกรีเท่ากับ n-m และคาร์ดินัลลิตีเท่ากับ N1*N2* แสดงดังรูปที่ 2.8

R1

Name	Age	Sex
A	20	M
C	21	M

R2

Name	Age	Sex
D	20	F
E	21	F

R3 = R1 X R2

Name	Age	Sex	Name	Age	Sex
A	20	M	D	20	F
C	21	M	D	20	F
A	20	M	E	21	F
C	21	M	E	21	F

รูปที่ 2.8 ตัวอย่างการดำเนินการ CARTESIAN PRODUCT ของรีเลชัน 2 รีเลชัน

2.4.5 Selection

ตัวดำเนินการ select เป็นการเลือกทูเปิลที่เป็นซับเซต (subset) ในรีเลชัน เป็นตัวดำเนินการทางคณิตศาสตร์ ที่ใช้กับรีเลชันเดียว เพื่อให้ได้ซับเซตที่เป็นจริงตามเงื่อนไขการเลือกหรือเป็นจริงตามเงื่อนไขที่กำหนด

รูปแบบของการใช้การดำเนินการ select คือ

σ <select condition> (<relation>)

โดยที่ <select condition> คือ
<attribute> <comparison operator> <constant value>/<attribute>[AND/OR/NOT] <attribute> <comparison operator> <constant value>/<attribute>...]

เครื่องหมายเปรียบเทียบที่สามารถใช้ได้ในเงื่อนไข <, >, <=, >=, =, <> ใช้กับโดเมนของแอตทริบิวต์หรือค่าคงที่ที่จัดเก็บ

ดีกรีของรีเลชันที่ได้เป็นผลลัพธ์จะเท่ากับดีกรีของรีเลชันที่ตัวดำเนินการระบุหรือทำการเลือก คาร์ดินัลลิตีของรีเลชันผลลัพธ์จะน้อยกว่าหรือเท่ากับคาร์ดินัลลิตีรีเลชันเริ่มต้นหรือที่ดำเนินการ ถ้าคาร์ดินัลลิตีที่ได้จำนวนน้อยแสดงว่าการเลือกนั้นมีเงื่อนไขที่เข้มงวด แต่ในทางกลับกันถ้าคาร์ดินัลลิตีที่ได้มีจำนวนน้อยแสดงว่าเงื่อนไขที่เลือกนั้นจะไม่ได้เข้มงวด

หมายเหตุ Selection เป็นการดำเนินการที่มีคุณสมบัติของการสลับที่

รูปที่ 2.9 เป็นตัวอย่างการดำเนินการ select สองแบบกับรีเลชัน R อันแรกมีเงื่อนไขการเลือกเป็น Age = 20 ได้ผลลัพธ์ดังรีเลชัน R1 และแบบที่สองมีเงื่อนไขการเลือกเป็น (Sex = M) AND (Age >19) ได้ผลลัพธ์คือรีเลชัน R2

R

Name	Age	Sex
A	20	M
M	21	F
B	20	F
F	19	M
A	20	F
R	21	F
C	21	M

R1= σ (Age=20)(R)

Name	Age	Sex
A	20	M
B	20	F
A	20	F

R2= σ (Sex=M AND Age>19))(R)

Name	Age	Sex
A	20	M
C	21	M

รูปที่ 2.9 ตัวอย่างตัวดำเนินการ SELECT (แสดงเงื่อนไขของการ Select ที่ต่างกัน)

2.4.6 Projection

การดำเนินการ Project สร้างรีเลชันใหม่โดยการ เลือกซับเซตของแอตทริบิวต์ในรีเลชัน ที่มีค่าของทูเปิลที่ไม่ซ้ำกัน เป็นตัวดำเนินการทางคณิตศาสตร์

รูปแบบของตัวดำเนินการ project คือ

π <attribute list> (<relation>)

โดยที่ <attribute list> เป็นซับเซตของแอตทริบิวต์ที่อยู่ในรีเลชัน

รีเลชันดีกรีของผลลัพธ์เท่ากับจำนวนของแอตทริบิวต์ที่กำหนดจาก <attribute list> เพราะจะปรากฏเฉพาะแอตทริบิวต์ที่เลือกในรีเลชันผลลัพธ์ หากแอตทริบิวต์ที่เลือกเป็นแคนดิเดตคีย์ คาร์ดินัลลิตีของรีเลชันผลลัพธ์จะน้อยกว่าหรือเท่ากับคาร์ดินัลลิตีของรีเลชันเริ่มต้น เนื่องจากมีทูเปิลที่ซ้ำกันซึ่งถูกการดำเนินการดังกล่าวตัดออก

Projection เป็นการดำเนินการที่ไม่มีคุณสมบัติการเปลี่ยนกลุ่ม

ในรูปที่ 2.10 แสดงตัวอย่างการดำเนินการ project กับรีเลชัน R จำนวน 2 ตัวอย่าง โดยตัวอย่างแรกทำการ projection แอตทริบิวต์ Name และ Sex ผลลัพธ์ที่ได้คือรีเลชัน R1 ตัวอย่างที่ 2 ทำการ projection แอตทริบิวต์ Age และ Sex ผลลัพธ์คือรีเลชัน R2

R

Name	Age	Sex
A	20	M
M	21	F
B	20	F
F	19	M
A	20	F
R	21	F
C	21	M

R1= π (Name, Sex)(R)

Name	Sex
A	M
M	F
B	F
F	M
A	F

R2= π (Age, Sex)(R)

Age	Sex
20	M
21	F
20	F
19	M

รูปที่ 2.10 – ตัวอย่างของการดำเนินการ PROJECT (ได้แอตทริบิวต์ที่แตกต่างกัน)

2.4.7 Join

Join เป็นตัวดำเนินการในการเชื่อมรีเลชันสองรีเลชัน บนเงื่อนไขการเชื่อม(join condition or predicate) รีเลชันที่จะทำการเชื่อมกันต้องมีอย่างน้อย 1 แอตทริบิวต์ที่เป็นแอตทริบิวต์กลางที่ใช้ร่วมกัน อยู่ภายใต้โดเมนเดียวกัน การ join ต้องมีการระบุแอตทริบิวต์ที่จะใช้ในการ join

รูปแบบของการดำเนินการ join คือ

R <join condition> ►◄ S

โดยที่ <join condition> คือ

<attribute from R> <comparison operator> < <attribute from S>

เครื่องหมายเปรียบเทียบที่สามารถใช้ได้ในเงื่อนไข <, >, <=, >=, =, <> ใช้กับโดเมนของแอตทริบิวต์ ถ้ารีเลชัน R ประกอบด้วยแอตทริบิวต์ A1, A2, ...,An รีเลชัน S ประกอบด้วยแอตทริบิวต์ B1, B2,...., Bm แอตทริบิวต์ Ai และ แอตทิบิวต์ Bj เป็นแอตทริบิวต์ที่อยู่ภายใต้โดเมนเดียวกัน เราสามารถทำการ join รีเลชัน R และ S โดยใช้แอตทริบิวต์ในการ join คือ Ai และ Bj ผลลัพธ์จะได้รีเลชัน T ที่ประกอบด้วยทูเปิล t ซึ่งการเชื่อมต่อกันระหว่างทูเปิล r จากรีเลชัน Rและทูเปิล s จากรีเลชัน S โดยที่ตัวที่ใช้ในการ join นั้นมีเงื่อนไขที่เป็นจริง จะเรียกการดำเนินการ join นี้ว่า theta-join ถ้ารีเลชันผลลัพธ์การ join มาจากจุดที่แอตทริบิวต์เหมือนกัน

โดยรีเลชันผลลัพธ์การ join จะประกอบด้วยทั้งสองแอตทริบิวต์ที่เหมือนกัน ถ้าทำการรวมแอตทริบิวต์ที่เหมือนกันจาก 2 แอตทริบิวต์ให้เหลือ 1 แอตทริบิวต์

เราจะเรียกการ join ดังกล่าวว่า natural join

รูปแบบที่ใช้เรียกการดำเนินการ join มักใช้คำว่า equijoin โดยแทนด้วยเครื่องหมาย ‘=’

ผลลัพธ์การ join ระหว่างรีเลชัน R และ S ในทางปฏิบัติบางครั้งเราต้องการผลลัพธ์ทุก ทูเปิลโดยไม่สนใจว่าจะ

มีค่าตรงกันหรือไม่ของทั้งสองรีเลชันที่จะทำการ join เราสามารถใช้ outer join

สำหรับการทำ outer join จะมีอยู่ 3 รูปแบบ คือ left outer join เมื่อต้องการทุกทูเปิลในรีเลชัน R เป็นผลลัพธ์ และ right outer join เมื่อต้องการทุกทูเปิลในรีเลชัน S เป็นผลลัพธ์ full outer join เมื่อต้องการทุกทูเปิลจากรีเลชัน R และ S เป็นผลลัพธ์ ในทูเปิลที่มีค่าแอตทริบิวต์ไม่ตรงกัน ระบบจะทำการพิจารณาใส่ค่าสมมติให้โดยให้เป็นค่าว่าง รวมอยู่ในทูเปิลเดียวกัน

ในรูปที่ 2.11 รีเลชันสองรีเลชันคือ R1 และ R2 ทำการ join โดยใช้แอตทริบิวต์ LastName ของรีเลชัน R1 เท่ากับแอตทริบิวต์ LastName ของรีเลชัน R2 ผลลัพธ์ได้รีเลชัน R3

R1

First Name	Last Name
A	Mary
B	John
C	Ann

R2

Last Name	Sex
Ann	F
John	M
Mary	F
Bill	M

R3=R1(Last Name=Last Name) R2

First Name	Last Name
A	Mary
B	John
C	Ann

รูปที่ 2.11 – ตัวอย่างของการดำเนินการ JOIN

ในรูปที่ 2.12 จะเห็นถึงผลลัพธ์การดำเนินการ join แบบ natural join ระหว่างรีเลชัน R1 และ R2 และผลลัพธ์สำหรับ right outer join

Natural Join

First Name	Last Name	Sex
A	Mary	F
B	John	M
C	Ann	F

Right Outer Join

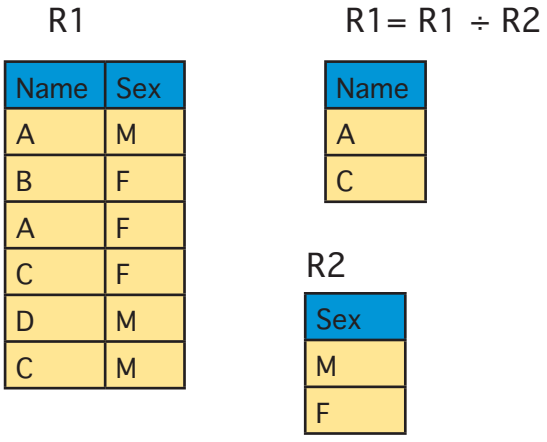
First Name	Last Name	Last Name	Sex
A	Mary	Mary	F
B	John	John	M
C	Ann	Ann	F
NULL	NULL	Bill	M

รูปที่ 2.12 –ตัวอย่างการดำเนินการ NATURAL JOIN และ RIGHT OUTER JOIN

2.4.8 Division

ตัวดำเนินการ Division เป็นการแบ่งรีเลชันหรือการหารรีเลชัน รีเลชัน R1 มีดีกรีเท่ากับ (n + m) หารด้วยรีเลชัน R2 มีดีกรีเท่ากับ m ผลลัพธ์ที่ได้คือรีเลชันที่มีดีกรีเท่ากับ n ลำดับแอตทริบิวต์ (n+i) ของ รีเลชัน R1 และ ลำดับแอตทริบิวต์ i จากรีเลชัน R2 ควรกำหนดในโดเมนเดียวกัน ผลลัพธ์การดำเนินการ division ระหว่าง R1 และ R2 จะได้รีเลชันผลลัพธ์ใหม่ที่ประกอบด้วยทูเปิลทั้งหมดที่เชื่อมต่อด้วยทูเปิลทั้งหมดของ R2 ที่เป็นส่วนหนึ่งของรีเลชัน R1

รูป 2.13 แสดงตัวอย่างการดำเนินการ Division ระหว่าง R1 และ R2



รูปที่ 2.13 – ตัวอย่างการดำเนินการ DIVISION

2.5 แคลคูลัสเชิงสัมพันธ์

แคลคูลัสเชิงสัมพันธ์เป็นการจัดการกับข้อมูลในระบบฐานข้อมูลอีกรูปแบบหนึ่ง ซึ่งต่างจากพีชคณิตเชิงสัมพันธ์ (Relational algebra) ความแตกต่างของทั้งสองคือ

- พีชคณิตเชิงสัมพันธ์จัดเตรียมตัวดำเนินการต่างๆ ไว้ เช่น union, intersect, difference, select, project, join ฯลฯ ซึ่งจะใช้ สร้างรีเลชันผลลัพธ์ จากรีเลชันในฐานข้อมูล
- แคลคูลัสเชิงสัมพันธ์ เป็นการวางกฎเกณฑ์ คำนิยามของรีเลชัน ตัวอย่างเช่น การสอบถามข้อมูลเจ้าของรถยนต์ซึ่งประกอบด้วยชื่อนามสกุล ชื่อเมืองที่อยู่ของเจ้าของรถยนต์ที่เป็นเจ้าของรถยนต์สีแดง

ตัวอย่างของพีชคณิตเชิงสัมพันธ์สำหรับการสอบถามข้อมูลที่ต้องการ คือ

- Join รีเลชัน OWNERS ด้วย รีเลชัน CARS โดยใช้แอตทริบิวต์ IdentificationNumber
- เลือกทูเปิลให้ได้รีเลชันผลลัพธ์ ที่มีค่า Colour = “RED”
- Project ผลลัพธ์โดยให้แสดงเฉพาะแอตทริบิวต์ FirstName, LastName และ City

ในส่วนของแคลคูลัสเชิงสัมพันธ์ จะใช้วิธีต่อไปนี้ คือ

- การนำเอา FirstName, LastName และ City ของเจ้าของรถยนต์ในรีเลชัน OWNERS โดยใช้แอตทริบิวต์ IdentificationNumber ที่เหมือนกันของสองรีเลชัน (OWNERS และ CARS) และ

เลือก RED จากรีเลชัน CARS ซึ่งเพียงผู้ใช้ระบุลักษณะของเซตของทูเปิลที่ต้องการ ระบบจะทำการประมวลผลให้ตามที่ระบุ

จากที่ได้กล่าวข้างต้นแคลคูลัสเชิงสัมพันธ์เป็นการบรรยายลักษณะของข้อมูลที่ต้องการ โดยไม่สนใจวิธีการ แต่พีชคณิตเชิงสัมพันธ์เป็นการทำอย่างไรเพื่อให้ได้มาซึ่งข้อมูลที่ต้องการ

ในความเป็นจริงแคลคูลัสเชิงสัมพันธ์และพีชคณิตเชิงสัมพันธ์มีความสามารถทำในสิ่งที่เหมือนกัน ทุกสิ่งที่ทำได้ในพีชคณิตเชิงสัมพันธ์เราก็สามารถทำได้ในแคลคูลัสเชิงสัมพันธ์เพียงแต่ต่างรูปแบบกันเท่านั้น โดยทั่วไปพีชคณิตเชิงสัมพันธ์จะมีลักษณะคล้ายภาษาโปรแกรมและแคลคูลัสเชิงสัมพันธ์มีลักษณะคล้ายภาษาธรรมชาติมากกว่า

แคลคูลัสเชิงสัมพันธ์อยู่บนพื้นฐานในสาขาหนึ่งของตรรกทางคณิตศาสตร์ (mathematical logic) ที่เรียกว่า Predicate Calculus (หรือ First-Order Logic) Kuhns [2.4] ซึ่งถือได้ว่าเป็นบิดาในการใช้ Predicate Calculus ซึ่งเป็นพื้นฐานของภาษาในฐานข้อมูล แต่ Codd เป็นคนแรกที่นำเสนอแนวคิดแคลคูลัสเชิงสัมพันธ์ในการประยุกต์ใช้ Predicate Calculus โดยระบุไว้ในฐานข้อมูลเชิงสัมพันธ์ข้อ [2.3] ส่วนภาษาที่ระบุไว้ในพื้นฐานแคลคูลัสเชิงสัมพันธ์ซึ่งนำเสนอโดย Codd ใน [2.5] ถูกเรียกว่า data sublanguage ALPHA ภาษา QUEL จาก INGRES มีความคล้ายกับภาษา data sublanguage ALPHA นอกจากนี้ Codd ยังใช้อัลกอริทึม reduction algorithm ของ Codd เอง เพื่อปรับปรุงนิพจน์ให้ใช้ได้เทียบเท่ากับพีชคณิตเชิงสัมพันธ์

เราสามารถจำแนกประเภทของแคลคูลัสเชิงสัมพันธ์ได้ออกเป็น 2 ประเภท ดังนี้

1. Tuple-oriented relational calculus การใช้ตัวแปรในพื้นฐานของทูเปิล
2. Domain-oriented relational calculus การใช้ตัวแปรในพื้นฐานของโดเมน

2.5.1 ทูเปิลเชิงแคลคูลัสเชิงสัมพันธ์ (Tuple-oriented relational calculus)

ตัวแปรทูเปิล (tuple variable) เป็นตัวแปรในขอบเขตของทูเปิลในรีเลชัน โดยเป็นการกำหนดเงื่อนไขของค่าที่เป็นไปได้ของทูเปิลในรีเลชัน หรืออีกนัยหนึ่ง ถ้าตัวแปรทูเปิล T มีขอบเขตครอบคลุม รีเลชัน R ดังนั้น T จะเป็นตัวแทนของทูเปิลบางทูเปิล t ที่อยู่ในรีเลชัน R

การกำหนด tuple variable

RANGE OF T IS X1; X2; ...; Xn

โดย T หมายถึง tuple variable และ X1; X2; ...; Xn เป็น tuple calculus expressions ซึ่งเป็นตัวแทนของรีเลชัน R1, R2, ..., Rn รีเลชัน R1, R2, ..., Rn มีความสัมพันธ์กันและมีการกำหนดชื่อแอตทริบิวต์ในทุกรีเลชันที่เหมือนกัน tuple variable ชื่อ T มีขอบเขตครอบคลุมทุกรีเลชันที่กล่าวมา (union of relations) ถ้ากำหนดให้ tuple calculus expressions มาจากรีเลชัน R (ในกรณีปกติ) tuple variable ชื่อ T ดังกล่าวจะมีขอบเขตแทนทูเปิลทั้งหมดในรีเลชันนั้น

การประกาศตัวแปรทูเปิลนั้นสามารถทำได้สองรูปแบบ คือ เป็นการประกาศ ตัวแปรแบบอิสระ (Free variable) และประกาศ ตัวแปรแบบไม่เป็นอิสระ (Bound variable) ตัวแปรทูเปิลสามารถอยู่ในรูปแบบ T.A โดย A เป็นแอตทริบิวต์ของรีเลชันในขอบเขต T โดยจะเรียกว่าการประกาศตัวแปรแบบอิสระ แต่หากตัวแปรทูเปิลนั้นตามด้วย the existential quantifier ∃ หรือ the universal quantifier ∀ จะเรียกว่าตัวแปรแบบไม่เป็นอิสระ

รูปแบบของ tuple calculus expression

T.A, U.B, ..., V.C WHERE f

โดยที่ T, U, ..., V เป็น tuple variables และ A, B, ..., C เป็นแอตทริบิวต์ของรีเลชันที่เกี่ยวข้อง ส่วน f เป็น relational calculus formula ที่มี T, U, ..., V เป็นตัวแปรแบบอิสระ (Free variable) ค่าดังกล่าวมาจากการ ทำ projection ของซับเซตที่เกิดขึ้นจาก Cartesian product T×U×...×V (เมื่อ T, U, ..., V เป็นค่าที่เป็นไปได้) โดยที่ f มีค่าที่เป็นจริง (true) หรือ WHERE f เกิดจากการ ละเว้น projection ของ Cartesian product

ตัวอย่างการสอบถามข้อมูลโดยใช้ฟิลด์ข้อมูล FirstName, LastName และ City ของเจ้าของรถ (รีเลชัน OWNERS) โดยที่รถยนต์ (รีเลชัน CARS) ต้องมี IdentificationNumber ที่เหมือนกันและมีสีแดง สามารถทำได้โดยใช้นิพจน์ ดังต่อไปนี้

```
RANGE OF OWNERS IS
    OWNERS.FirstName, OWNERS.LastName, OWNERS.City WHERE
        ∃ CARS(CARS.IdentificationNumber=OWNERS.IdentificationNumber
                                AND CARS.Color='RED')
```

tuple calculus มีลักษณะการดำเนินการที่เหมือนกับพีชคณิตเชิงสัมพันธ์

ภาษา QUEL ของ INGRES อยู่บนพื้นฐานของ tuple-oriented relational calculus

2.5.2 โดเมนแคลคูลัสเชิงสัมพันธ์ (Domain-oriented relational calculus)

Lacroix และ Pirotte [2.6] ได้นำเสนออีกรูปแบบหนึ่งของแคลคูลัสเชิงสัมพันธ์ เรียกว่าแคลคูลัสโดเมน (domain calculus) ซึ่งเป็นการแทนค่าตัวแปรทูเปิล (tuple variables) ด้วยตัวแปรโดเมน (domain variables) โดยที่ตัวแปรโดเมน จะครอบคลุมโดเมนแทนที่จะเป็น รีเลชัน

ตัวแปรโดเมนสามารถเป็นตัวแปรแบบอิสระและตัวแปรแบบไม่เป็นอิสระ ตัวแปรโดเมนแบบไม่เป็นอิสระจะตามด้วย the universal quantifier ∀ หรือ the existential quantifier ∃ ในกรณีอื่นนอกจากนี้จะถือว่าเป็นตัวแปรแบบอิสระ

โดเมนแคลคูลัสเชิงสัมพันธ์ จะประกอบด้วยสมาชิกและข้อกำหนด รูปแบบข้อกำหนดของสมาชิก (membership conditions) ได้แก่

R (term, term, ...)

โดยที่ R เป็นชื่อรีเลชัน และ term คือคู่ลำดับในรูปแบบ A:v ซึ่ง A เป็นแอตทริบิวต์ ของ R และ v เป็นตัวแปรโดเมนหรือค่าคงที่ เงื่อนไขจะเป็นจริง (true) เมื่อทูเปิล ในรีเลชัน R มีค่าที่มีอยู่จริงสำหรับแอตทริบิวต์ที่ระบุ

ตัวอย่างเช่น นิพจน์ OWNERS (IdentificationNumber: 'SB24MEA', City: 'SIBIU') เป็น membership condition ซึ่งผลการเปรียบเทียบ จะมีค่าเป็นจริงก็ต่อเมื่อทูเปิลในรีเลชัน OWNERS มีแอตทริบิวต์ IdentificationNumber ที่มีค่าเท่ากับ SB24MEA และ แอตทริบิวต์ City มีค่าเท่ากับ SIBIU ในทำนองเดียวกันกับ

R (A:AX, B:BX, ...)

ผลลัพธ์จะเป็นจริงก็ต่อเมื่อรีเลชัน R ประกอบด้วยทูเปิลที่มีแอตทริบิวต์ A ซึ่งมีค่าเท่ากับตัวแปรโดเมน AX (ค่าอะไรก็ตามที่เป็นไปได้) ค่าแอตทริบิวต์ B มีค่าเท่ากับตัวแปรโดเมน BX (ค่าอะไรก็ตามที่เป็นไปได้) และอื่นๆ

ตัวอย่างการสอบถามข้อมูล โดยที่ต้องการ FirstName, LastName และ City ของเจ้าของรถ (รีเลชัน OWNERS) โดยที่รถยนต์ (รีเลชัน CARS) ต้องมี IdentificationNumber ที่เหมือนกันและมีสีแดง สามารถเขียนได้ดังนี้

FirstNameX, LastNameX, CityX WHERE → IdentificationNumberX

(OWNERS (IdentificationNumber:IdentificationNumberX,

FirstName:FirstNameX, LastName:LastNameX, City:CityX)

AND CARS(IdentificationNumber:IdentificationNumberX, Color:'RED')

โดเมนเชิงแคลคูลัสมีลักษณะรูปแบบการดำเนินการที่เหมือนกับพีชคณิตเชิงสัมพันธ์

ภาษา ที่เรียกว่า ILL อยู่บนพื้นฐานของแคลคูลัสที่นำเสนอ โดย Lacroix และ Pirotte ใน [2.7] ภาษาอื่นที่อยู่บนพื้นฐานของโดเมนเชิงแคลคูลัสสัมพันธ์คือ Query-By-Example (QBE)

2.6 สรุป

ในบทที่ 2 นี้ เราได้กล่าวถึงพื้นฐานแนวคิดเกี่ยวกับแบบจำลองเชิงสัมพันธ์ ซึ่งประกอบไปด้วย แอตทริบิวต์ ทูเปิล รีเลชัน โดเมน โครงร่าง แคนดิเดตคีย์ คีย์หลัก อัลเทอเนทคีย์ (alternate keys) และคีย์นอก โดยได้มีการอธิบายและยกตัวอย่างประกอบ เพื่อให้เกิดความเข้าใจแนวคิดและส่วนประกอบของฐานข้อมูลเชิงสัมพันธ์ที่ดีขึ้น

ในบทนี้ยังได้นำเสนอการควบคุมข้อมูลของแบบจำลองข้อมูลเชิงสัมพันธ์ ประเภทของการควบคุมข้อมูลของแบบจำลองข้อมูลเชิงสัมพันธ์ เช่น การควบคุมข้อมูลเอนทิตีอินทิกริตี การควบคุมข้อมูลเรเฟอเรนเชียลอินทริกริตี การควบคุมข้อมูลซีเมนทิคอินทิกริตี เราได้อธิบายถึงการควบคุมข้อมูลต่างๆที่กล่าวมารวมถึงบทบาทหน้าที่และประโยชน์ต่อฐานข้อมูลเชิงสัมพันธ์

พีชคณิตเชิงสัมพันธ์ประกอบด้วยตัวดำเนินการ เช่น union intersection difference Cartesian product selection projection join และ division ซึ่งได้อธิบายรายละเอียดให้ทราบแล้วในบทนี้ สิ่งที่สำคัญคือต้องมีความเข้าใจเกี่ยวกับตัวดำเนินการของพีชคณิตเชิงสัมพันธ์ที่เกี่ยวข้องกับรีเลชัน เพราะรูปแบบการสอบถามข้อมูลจากฐานข้อมูลเชิงสัมพันธ์ต้องใช้ตัวดำเนินการเหล่านั้น

แคลคูลัสเชิงสัมพันธ์ถูกนำเสนอเป็นอีกรูปแบบหนึ่งนอกจากพีชคณิตเชิงสัมพันธ์ในการจัดการกับข้อมูลในแบบจำลองข้อมูลเชิงสัมพันธ์ ได้มีการอธิบายความแตกต่างระหว่างแคลคูลัสเชิงสัมพันธ์กับพีชคณิตเชิงสัมพันธ์ ทูเปิลเชิงแคลคูลัสเชิงสัมพันธ์ที่มีพื้นฐานมาจากแนวคิดของตัวแปรทูเปิล โดเมนแคลคูลัสเชิงสัมพันธ์ที่มีพื้นฐานมาจากแนวคิดตัวแปรโดเมน

2.7 แบบฝึกหัด

ในบทนี้เราได้เรียนรู้แนวคิดพื้นฐานของแบบจำลองเชิงสัมพันธ์ เพื่อให้เข้าใจมากยิ่งขึ้น ให้พิจารณา

สถานการณ์จริงและรูปแบบของข้อมูลในฐานข้อมูลเชิงสัมพันธ์ดังต่อไปนี้

> บริษัทแห่งหนึ่งมีหลายแผนก แต่ละแผนกมีหมายเลขของแผนก ชื่อแผนก ผู้จัดการแผนก ที่อยู่และงบประมาณ แต่ละแผนกจะไม่มีหมายเลขแผนกที่ซ้ำกัน ไม่สามารถมีผู้จัดการคนเดียวกันได้ แต่ละแผนกจะมีผู้จัดการได้คนเดียว แต่ละแผนกจะมีพนักงานแยกจากกันแผนกไหนแผนกนั้นไม่มีพนักงานซ้ำกัน ข้อมูลของพนักงานที่ต้องการคือ ชื่อพนักงาน ตำแหน่งงาน เงินเดือน วันเดือนปีเกิด และรหัสพนักงาน ซึ่งเป็นค่าที่ไม่ซ้ำกัน

จากข้อมูลสามารถสร้างแบบจำลองของฐานข้อมูลเชิงสัมพันธ์ สำหรับข้อมูลดังกล่าวโดยมีรีเลชัน 2 รีเลชัน คือ

- DEPARTMENTS
- EMPLOYEES

รีเลชัน DEPARTMENTS จะประกอบด้วยแอตทริบิวต์ คือ DeptNo DepName Manager Address Budget

รีเลชัน EMPLOYEES จะประกอบด้วยแอตทริบิวต์ คือ ID EmpName Job Salary BirthDate DepNo แต่ละแอตทริบิวต์จะต้องมีโดเมน

สำหรับตัวอย่างนี้ รีเลชันควรจะมีโดเมนเป็นดังนี้

DEPARTMENTS		EMPLOYEES	
DepNo	Numeric(2,0)	ID	Numeric(3,0)
DepName	Character(20)	EmpName	Character(30)
Manager	Numeric(3,0)	Job	Character(10)
Address	Character(50)	Salary	Numeric(7,2)
Budget	Numeric(10,2)	BirthDate	Date
		DepNo	Numeric(2,0)

แต่ละรีเลชันต้องมีคีย์หลัก สำหรับรีเลชัน DEPARTMENTS มีแคนดิเดตคีย์ คือ DepNo และ Manager หนึ่งในแคนดิเดตคีย์จะถูกเลือกให้เป็นคีย์หลักและที่ไม่ถูกเลือกจะเป็นอัลเทอเนทคีย์ ในตัวอย่าง DepNo เป็นคีย์หลัก Manager เป็นอัลเทอเนทคีย์ สำหรับคีย์หลักของรีเลชัน EMPLOYEE คือ ID ซึ่งต้องเป็นไปตามข้อกำหนดคีย์หลักคือ แอตทริบิวต์ที่ใช้เป็นคีย์หลักจะต้องไม่มีค่าเป็นค่าว่าง ดังนั้นเราต้องกำหนดให้ ID ไม่สามารถเป็นค่าว่างได้ (NOT NULL)

ในโปรแกรม DB2 สามารถสร้างรีเลชันได้จากคำสั่งดังนี้

```
CREATE TABLE Departments (
        DepNo Numeric(2,0) NOT NULL PRIMARY KEY,
        DepName Char(20) NOT NULL,
        Manager Numeric(3,0) NOT NULL,
        Address Char(50),
        Budget Numeric(10,2)  );

CREATE TABLE Employees (
        ID Numeric(3,0) NOT NULL PRIMARY KEY,
        EmpName Char(30) NOT NULL,
        Job Char(10) NOT NULL,
        Salary Numeric(7,2),
        BirthDate Date NOT NULL,
```

```
DepNo Numeric(2,0));
```

ความสัมพันธ์ระหว่างรีเลชัน DEPARTMENTS และ EMPLOYEES พนักงาน 1 คนทำงานใน 1 แผนก มีการใช้การควบคุมข้อมูลคีย์นอก โดยที่มี DepNo เป็นคีย์นอกสำหรับรีเลชัน EMPLOYEES และอ้างอิงไปยังคีย์หลัก DepNo ของรีเลชัน DEPARTMENTS

ในโปรแกรม DB2 การใช้ การควบคุมข้อมูลเรเฟอเรนเชียลอินทริกริตี (referential integrity constraint) สามารถแสดงได้ ดังนี้

```
ALTER TABLE Employees ADD FOREIGN KEY (DepNo)
REFERENCES Departments (DepNo)
ON DELETE RESTRICT
ON UPDATE RESTRICT
ENFORCED ENABLE QUERY OPTIMIZATION;
```

ทูเปิลของรีเลชัน EMPLOYEES และ DEPARTMENTS และค่าของทูเปิล สามารถแสดงได้ดังรูปที่ 2.14

DEPARTMENTS Relation

DepNo	DepName	Manager	Address	Budget
2	Software	211	Bucharest	2,000,000.00
1	Accounting	422	Bucharest	500,000.00
3	Hardware	111	Bucharest	4,000,000.00

EMPLOYEES Relation

ID	EmpName	Job	Salary	BirthDate	DepNo
211	John Smith	Engineer	Bucharest	04/05/1966	2
123	Ann Adams	Accoutant	Bucharest	11/05/1975	1
311	Bill Johns	Programmer	Bucharest	09/03/1980	

รูปที่ 2.14 – รีเลชัน DEPARTMENT และ EMPLOYEE

2.8 คำถามทบทวน

1. รีเลชัน supplier ประกอบด้วย 4 แอตทริบิวต์คือ Id - เลขรหัสของ supplier เป็นค่าไม่ซ้ำและไม่เป็นค่าว่าง (unique,not null) Name - ชื่อ Supplier ไม่เป็นค่าว่าง (not null) Addess - ที่อยู่ supplier Discount - ส่วนลดของ supplier ไม่เป็นค่าว่าง ค่าอยู่ระหว่าง 0%-50% (not null, values between 0 % and 50 %) จงบอกถึงการควบคุมข้อมูลที่ใช้ในการตรวจสอบความถูกต้องของข้อมูลในรีเลชัน
2. ตัวดำเนินการหลัก 5 ตัวของการดำเนินการพีชคณิตศาสตร์ คือ union difference, Cartesian product, selection และ projection ซึ่งสามารถดำเนินการร่วมกับการดำเนินการอื่นๆได้ จงอธิบายการใช้ intersection, join และ division ในเชิงการดำเนินการในแบบของพีชคณิตเชิงสัมพันธ์ทั้ง 5 ตัวดำเนินการ

3. สำหรับความสัมพันธ์ที่กำหนดไว้ในข้อ 1 ต้องการหาชื่อ supplier จาก New York ที่ใช้ส่วนลดมากกว่า 5% จากตัวดำเนินการพีชคณิตศาสตร์ (relational algebra operations)

4. ผลลัพธ์ที่ต้องการเหมือนข้อ 3 โดยใช้ทูเปิลเชิงแคลคูลัสสัมพันธ์ (tuple-oriented relational calculus)

5. ผลลัพธ์ที่ต้องการเหมือนข้อ 3 โดยใช้โดเมนเชิงแคลคูลัสสัมพันธ์ (domain-oriented relational calculus)

6. ข้อใดต่อไปนี้อธิบายถึงแอตทริบิวต์ในมุมมองของแบบจำลองเชิงสัมพันธ์

   A. แบบจำลองฐานข้อมูลที่สร้างจากข้อมูลจริง

   B. เซตของค่าที่แบ่งแยกไม่ได้ (atomic values)

   C. คุณลักษณะของข้อมูล (data characteristic)

   D. เซตของคำสั่งที่อธิบายคุณลักษณะของข้อมูล (an ordered set of values that describe data characteristics)

   E. ไม่มีข้อใดถูก

6. ข้อใดต่อไปนี้เป็นคุณลักษณะของรีเลชัน

   F. คีย์หลักไม่สามารถเป็นค่าว่างได้

   G. ค่าทูเปิลในรีเลชันต้องเป็นค่าที่ไม่ซ้ำ

   H. แอตทริบิวต์เป็นค่าที่แบ่งแยกไม่ได้(atomic values)

   I. ค่าทูเปิลในรีเลชันเป็นค่าที่ซ้ำกันได้

   J. ไม่มีข้อใดถูก

7. สิ่งที่สามารถมีในรีเลชัน

   A. โดเมน (domain)

   B. อินสแตนท์ (instance)

   C. ค่า (value)

   D. ดีกรี (degree)

   E. ไม่มีข้อใดถูก

8. ข้อใดต่อไปนี้ถูกต้อง

   A. คีย์หลักคือคีย์แคนดิเดตคีย์ (candidate)

   B. แต่ละรีเลชันมีคีย์นอกอย่างน้อยหนึ่งคีย์

C. คีย์นอกไม่สามารถมีค่าว่างได้

D. คีย์หลักคือคีย์การสำรองหรืออัลเทอเนทคีย์ (alternate key)

E. ไม่มีข้อใดถูก

9. เมื่อมีการลบทูเปิลจากรีเลชันที่มีการกำหนดคีย์หลัก ทางเลือก (options) ของการกำหนด คีย์นอกใดต่อไปนี้ ที่จะทำให้เกิดการลบทุกทูเปิลที่มีค่าเดียวกันในรีเลชัน ที่เป็นคีย์นอก

A. Restrict

B. ตั้งค่าเป็นค่าว่าง (Null)

C. Delete

D. Cascade

E. ไม่มีข้อใดถูก

3

บทที่ 3 – แนวคิดเกี่ยวกับแบบจำลองข้อมูล

สำหรับเนื้อหาในบทนี้จะอธิบายถึงแนวคิดของการพัฒนาแบบจำลองฐานข้อมูล และอธิบายถึงมุมมองในการนำเอาแบบจำลอง ไปใช้ในการติดตั้ง ซึ่งหลังจากได้ศึกษารายละเอียดใน บทนี้เราจะสามารถ:

- ให้ข้อมูลที่จำเป็นเพื่ออธิบายถึงความต้องการทางธุรกิจได้อย่างถูกต้อง
- ป้องกันความผิดพลาดและความเข้าใจผิดเกี่ยวกับแบบจำลองข้อมูล
- สามารถให้รายละเอียดเกี่ยวกับกฎทางธุรกิจ (business rules) ที่จำเป็นได้
- ให้ข้อมูลพื้นฐานสำหรับการออกแบบฐานข้อมูลในระดับลอจิคัล (logical)
- มีความเข้าใจเกี่ยวกับข้อระเบียบและกฎหมายสำหรับการควบคุมองค์กร

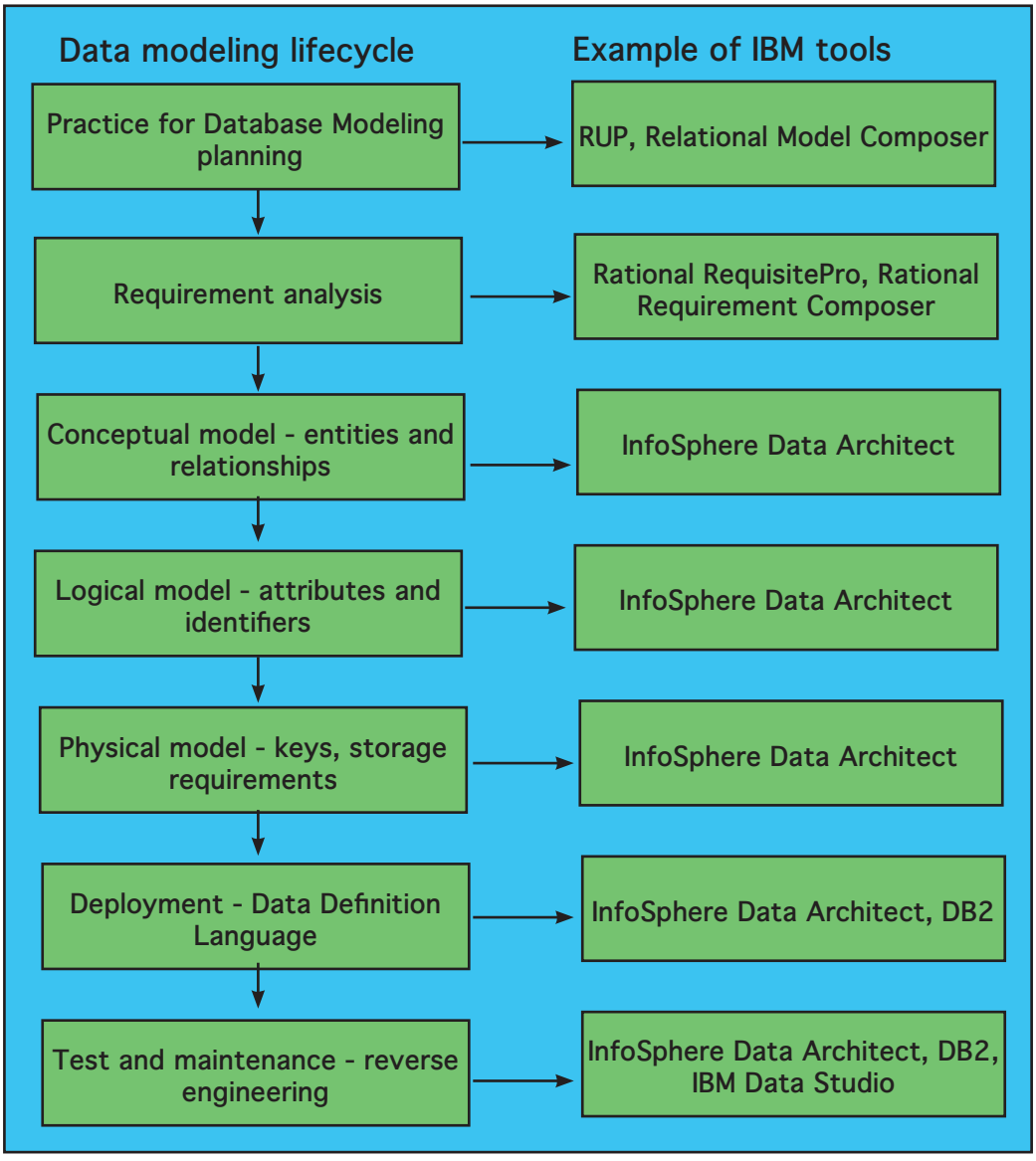
3.1 ภาพรวมการสร้างแบบจำลองข้อมูล

แบบจำลองข้อมูลที่มักจะกล่าวถึง ได้แก่แบบจำลองระดับแนวคิด (Conceptual modeling) แบบจำลองทางตรรกะ (Logical modeling) และแบบจำลองทางกายภาพ (Physical modeling) ซึ่งเป็นแบบจำลองสำคัญที่นำมาใช้สำหรับการทำงานกับฐานข้อมูล สำหรับแบบจำลองระดับแนวคิดนั้นจะมุ่งเน้นรายละเอียดจากที่เห็นในมุมของธุรกิจ ซึ่งใช้สำหรับเป็นข้อมูลที่จะกำหนดเอนทิตี และความสัมพันธ์ในเชิงธุรกิจ แบบจำลองทางตรรกะจะขึ้นอยู่กับแบบจำลองทางคณิตศาสตร์ ซึ่งจะนำเสนอข้อมูลและเอนทิตีในลักษณะที่เป็นมาตรฐานโดยที่ไม่มีความซ้ำซ้อนของข้อมูล แบบจำลองทางกายภาพจะเป็นการติดตั้งแบบจำลองทางตรรกะ ซึ่งจะถูกสร้างขึ้นมาเฉพาะเจาะจงกับฐานข้อมูลในแต่ละผลิตภัณฑ์และเฉพาะรุ่น

สำหรับบทนี้มุ่งเน้นการออกแบบแบบจำลองข้อมูลในระดับแนวคิด (Conceptual design) แต่ในตอนท้ายของบทนี้ก็จะมีหัวข้อที่เกี่ยวข้องกับการสร้างแบบจำลองทางตรรกะและทางกายภาพ ซึ่งจะแสดงกรณีศึกษาสำหรับแบบจำลองระดับแนวคิด และแปลงไปเป็นแบบจำลองเชิงตรรกะ (ในบทที่ 4) และแบบจำลองทางกายภาพ (ในบทที่ 5) ต่อไป

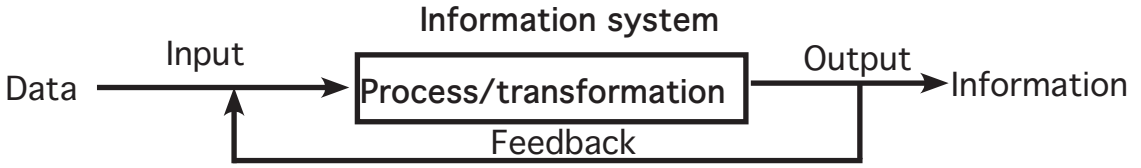
สิ่งสำคัญของการพัฒนาแบบจำลองจะต้องมีการวิเคราะห์ความต้องการของผู้ใช้ (User requirement)

เพื่อทำความเข้าใจในวัตถุประสงค์ และ ป้องกันการเข้าใจที่ผิดที่อาจจะเกิดจากการสื่อสารระหว่างกัน ตัวอย่างของเครื่องมือที่สามารถใช้เพื่อวัตถุประสงค์ดังกล่าวนี้ คือเครื่องมือที่ ชื่อว่า IBM's Rational RequisitePro และ IBM's InfoSphere Data Architect รูปที่ 3.1 แสดงถึงแบบจำลองระดับแนวคิด แบบจำลองทางตรรกะ และแบบจำลองทางกายภาพ ที่เข้ากันกับวงจรชีวิตของแบบจำลองข้อมูล และ เครื่องมือต่างๆ ของ ไอบีเอ็ม ที่เกี่ยวข้อง และสามารถนำมาใช้เพื่อการพัฒนาแบบจำลอง



รูปที่ 3.1 - ขั้นตอนการออกแบบแบบจำลองข้อมูลและความสัมพันธ์กับเครื่องมือของไอบีเอ็ม

แบบจำลองข้อมูลอธิบายถึงวิธีการแสดงและการเข้าถึงข้อมูล โดยจะแสดงองค์ประกอบและความสัมพันธ์ระหว่างข้อมูล ข้อมูลเป็นกลุ่มของตัวอักษร ตัวเลข ข้อความและเอกสารที่สามารถตีความได้หลายวิธี ข้อมูลที่ผ่านการประมวลผลซึ่งเรียกว่า ข้อมูลสารสนเทศ (information) จะถือว่าเป็นข้อมูลที่มีประโยชน์ เมื่อเทียบกับข้อมูลที่มีการจัดเก็บไว้เพียงอย่างเดียวที่เรียกว่า data รูป 3.2 แสดงถึงภาพรวมของความสัมพันธ์ระหว่าง data และ information



รูปที่ 3.2 – แสดงขั้นตอนการประมวลผลข้อมูล

จากรูป 3.2 แสดงถึงการนำเข้าข้อมูลดิบ (data) ที่มีการประมวลผล และแปลงเป็นผลลัพธ์ ซึ่งก็คือ ข้อมูลสารสนเทศ (information) ตัวอย่างของข้อมูลในบริบทของมหาวิทยาลัยจะเป็น ชื่อของนักเรียน และตัวอย่างของข้อมูลสารสนเทศในบริบทเดียวกันจะเป็นหมายเลขของนักศึกษา

3.2 แบบจำลองคืออะไร?

แบบจำลองเป็นนามธรรม (abstraction) ซึ่งเป็นตัวแทนที่แสดงให้เห็นถึงคุณสมบัติ ความสนใจเกี่ยวกับข้อมูลสารสนเทศของผู้ใช้ แบบจำลองถูกสร้างขึ้นมา เพื่อให้ผู้ใช้งานสามารถเข้าใจในกระบวนการปรากฏการณ์ หรือกิจกรรมต่างๆ ที่เกิดขึ้น

เราใช้แบบจำลอง เพื่อแสดงให้เห็นภาพของข้อมูลที่ต้องการจัดเก็บ เหตุการณ์ที่สามารถเกิดขึ้นได้ และการแสดงถึงความสัมพันธ์ซึ่งกันและกัน เพื่อให้เกิดเป็นแนวคิดพื้นฐานและกฎกติกาที่ใช้สำหรับการสื่อสารที่ดีระหว่างผู้ใช้งานกับนักพัฒนาระบบฐานข้อมูล

3.2.1 แบบจำลองข้อมูล

แบบจำลองข้อมูลเป็นการกำหนดข้อมูลสารสนเทศที่จำเป็นทั้งหมด เพื่อให้แน่ใจว่าข้อมูลที่ถูกนำเสนอนั้นมีความเป็นหนึ่งเดียว ดังที่ Connolly [3] ได้กล่าวไว้ว่า “แบบจำลองข้อมูลเป็นเครื่องมือสำหรับ ข้อมูลในเชิงนามธรรม ซึ่งใช้เป็นตัวแทนขององค์กร และช่วยให้ผู้ใช้ข้อมูลสามารถสื่อสารได้อย่างชัดเจน แม่นยำ เพื่อความเข้าใจที่ดีในข้อมูลขององค์กร” ซึ่งการสร้างแบบจำลองข้อมูลนับว่าเป็นขั้นที่สำคัญในกระบวนการพัฒนาฐานข้อมูล

3.2.2 แบบจำลองฐานข้อมูล

แบบจำลองฐานข้อมูลถูกใช้เพื่ออธิบายข้อมูลของข้อมูล หรือเรียกว่า ‘เมตาดาต้า’ (metadata) แบบจำลองฐานข้อมูลเป็นการรวบรวมแนวคิดสำหรับคำอธิบายข้อมูล (data description) ความสัมพันธ์ของข้อมูล (data relationships) ความหมายข้อมูล (data semantics) และการควบคุมข้อมูล (data constraints) โดยปกติแล้วโครงสร้างของแบบจำลองฐานข้อมูลจะถูกอธิบายโดยโครงร่างฐานข้อมูล (database schema)

แบบจำลองฐานข้อมูลแบ่งเป็นประเภทต่างๆดังนี้

- แบบจำลองข้อมูลภายนอก (External data model) รูปแบบนี้ใช้เพื่อแสดงให้เห็นภาพของผู้ใช้ทุกคน โดยมีแบบจำลองหลัก คือ แบบจำลองสำหรับระเบียนเรคอร์ด (Records-based Logical Model)
- แบบจำลองระดับแนวคิด (Conceptual data model) สำหรับแบบจำลองระดับแนวคิดนี้ใช้สำหรับมุมมองทั่วไปของข้อมูล และไม่ได้เจาะจงว่าจะเป็นระบบฐานข้อมูลใดๆ ซึ่งการนำเสนอแบบจำลองลักษณะนี้จะใช้แบบจำลองเชิงวัตถุ (Object-based Logical Model)

- แบบจำลอง ข้อมูลภายใน (Internal data model) เป็นแบบจำลองข้อมูลที่แปลงมาจากแบบจำลองในระดับแนวคิด (conceptual) เพื่อให้แบบจำลองนั้นมีความเป็นรูปธรรม และอาจเจาะจงไปยังระบบฐานข้อมูลแต่ละผลิตภัณฑ์ เพื่อสามารถนำเอาแบบจำลองนี้ไปพัฒนาเป็นแบบจำลองทางกายภาพ (Physical data model) ต่อไป

สำหรับแบบจำลองที่นิยมใช้กันอย่างกว้างขวางมากที่สุดทุกวันนี้ ได้แก่ แบบจำลองของฐานข้อมูลเชิงสัมพันธ์ ซึ่งประกอบด้วยแบบจำลองความสัมพันธ์เอนทิตี (Entity-Relationship model) ที่อยู่บนพื้นฐานของแบบจำลองระดับแนวคิด ในการสร้างกฎทางธุรกิจ คุณลักษณะที่สำคัญที่สุดเกี่ยวกับแบบจำลองเชิงสัมพันธ์คือสามารถอธิบายให้เข้าใจได้ง่าย และใกล้เคียงกับ ความเป็นจริง

องค์ประกอบ สำหรับแบบจำลองฐานข้อมูล ประกอบด้วย

- ส่วนประกอบโครงสร้าง (Structural component) – หมายความถึงกฎกติกาทั่วไปสำหรับการพัฒนาฐานข้อมูล
- ส่วนประกอบการจัดการ (Manipulation component) – กำหนดการดำเนินการที่ใช้กับฐานข้อมูล (สำหรับค้นหา หรือปรับปรุงข้อมูลในฐานข้อมูล หรือ การปรับเปลี่ยนโครงสร้างฐานข้อมูล)
- ความสมบูรณ์ขององค์ประกอบข้อมูล (Data Integrity component) – ชุดของกฎที่สมบูรณ์ซึ่งรับประกันความถูกต้องของข้อมูล

3.2.3 แนวคิดของแบบจำลองข้อมูลระดับแนวคิด

ในกระบวนการของการสร้างแบบจำลองข้อมูล แบบจำลองข้อมูลระดับแนวคิดควรจะถูกสร้างขึ้นเป็นขั้นตอนแรก สำหรับแบบจำลองข้อมูลระดับแนวคิดนั้นเป็นภาพที่แสดงถึงองค์ประกอบต่างๆโดยรวม และไม่เจาะจงว่าจะเป็นแบบจำลองสำหรับฐานข้อมูลใดๆ แต่เป็นการแสดงแผนภาพกว้างๆ เพื่อแสดงถึงขอบเขตข้อมูลที่องค์กรต้องการจัดเก็บ และความสัมพันธ์ระหว่างออบเจ็กต์ต่างๆ

วัตถุประสงค์ของการออกแบบฐานข้อมูลระดับแนวคิดเพื่อที่จะสร้างเป็นแบบจำลองข้อมูลระดับแนวคิด ซึ่งจะต้องทำขั้นตอนดังต่อไปนี้

- สร้างไดอะแกรมความสัมพันธ์ระหว่างเอนทิตี (Entity-Relationship Diagram) โดยการสร้างเอนทิตี เพื่อระบุ แอตทริบิวต์ และสร้างความสัมพันธ์ระหว่างเอนทิตีที่เหมาะสม
- ทำการระบุเงื่อนไขและการควบคุมข้อมูลต่างๆ (Constraint) โดยเขียนเป็นเอกสารกำกับที่ชัดเจนว่ามีการควบคุมข้อมูลอะไรบ้าง เช่นเรเฟอเรนเชียลอินทริกริตี การควบคุมข้อมูลโดเมน การควบคุมข้อมูลเอนเตอร์ไพรซ์ และ เอนทิตีอินทิกริตีการทบทวนแบบจำลองที่ออกแบบ โดยการลบความสัมพันธ์แบบ M:N ลบความสัมพันธ์แบบวนซ้ำ (recursive relationships) การพิจารณากลุ่มเอนทิตีที่มีความสัมพันธ์แบบ 1 ต่อ 1 เนื่องจากปรกติความสัมพันธ์ลักษณะนี้มักจะไม่ปรากฏในฐานข้อมูล

3.2.3.1 แบบจำลองความสัมพันธ์เอนทิตี (Entity-Relationship Model)

ตามที่ได้กล่าวในบทที่ 1 แบบจำลอง E-R (E-R model) เป็นเครื่องมือที่ประสบความสำเร็จที่ใช้ ในการออกแบบฐานข้อมูลเชิงสัมพันธ์ ซึ่งปรกติจะถูกเรียกว่า Entity Relationship Diagram (ERD) เพื่อใช้ในการแสดงความสัมพันธ์ระหว่างเอนทิตีที่เกิดขึ้นในฐานข้อมูล และที่สำคัญคือใช้สำหรับเป็นแบบจำลองที่แสดงโครงสร้างความสัมพันธ์ของข้อมูลอีกด้วย

แนวคิดที่ใช้ในแบบจำลองความสัมพันธ์สำหรับเอนทิตี (Entity-Relationship model) นั้นประกอบด้วย

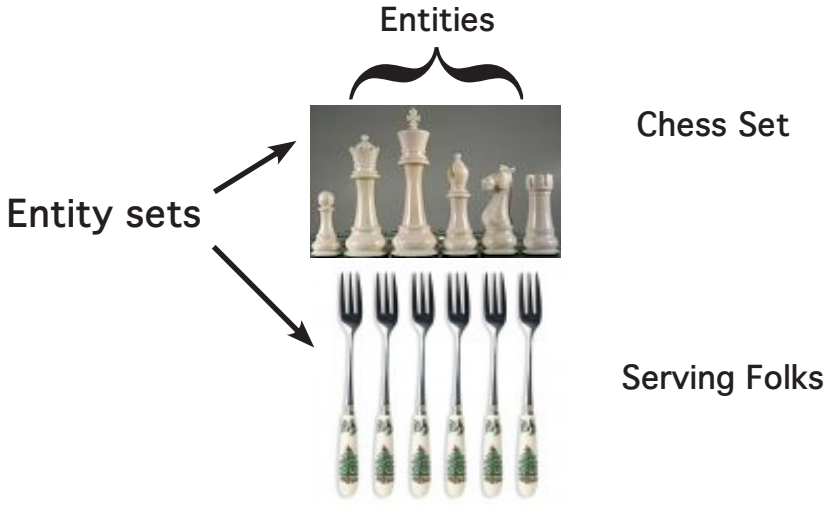
- ชุดเอนทิตี (Entity set)
- แอตทริบิวต์ (Attribute)
- ชุดความสัมพันธ์ (Relationship set)
- การควบคุมข้อมูล (Constraint)
- โดเมน (Domain)
- เอกซเทนชั่น (Extension)
- อินเทนชั่น (Intension)

3.2.3.2 เอนทิตีและชุดเอนทิตี

ชุดเอนทิตี ประกอบด้วยเอนทิตีที่มีชนิดเดียวกันและมีคุณสมบัติเหมือนกัน เอนทิตีเป็นอินสแตนซ์ของชุดเอนทิตี ตัวอย่างเช่น

- กลุ่มข้อมูลที่มีความหมายในตัวเอง เช่น Teacher, Student
- กลุ่มข้อมูลที่เป็นส่วนในส่วนหนึ่งของข้อมูล เช่น Grade
- กลุ่มข้อมูลที่เป็นเหตุการณ์ที่เกิดขึ้น เช่น Exam

รูป 3.3 แสดงตัวอย่างของเอนทิตีและชุดเอนทิตี โดยรูปที่เห็นเป็นสิ่งที่อธิบายตัวมันเอง



รูปที่ 3.3 - ตัวอย่างการกลุ่มเอนทิตีและเอนทิตี

ทั้งนี้การกำหนดเอนทิตีนั้นขึ้นอยู่กับบริบท คำนามที่กำหนดเช่น ครู อาจเป็นเอนทิตีหรือกำหนดเป็นกลุ่มของ เอนทิตี ตัวอย่างเช่นกรณีที่ต้องการแสดงความหมายของบุคคลโดยทั่วไป เช่น ครู (TEACHER) หรือ นักเรียน (STUDENT) เราอาจจะกำหนดชุดเอนทิตีที่มีความหมายครอบคลุมในลักษณะกลุ่มเช่น ชุดเอนทิตี PERSON เป็นต้น

สำหรับในบริบทของมหาวิทยาลัย สามารถกำหนดชุดเอนทิตี TEACHER โดยมีเอนทิตี เป็น PROFESSOR, ASSOCIATE PROFESSOR, ASSISTANT PROFESSOR หรืออื่นๆได้เช่นเดียวกัน

หมายเหตุ:

ในรูปแบบของแบบจำลองทางตรรกะ เราจะเรียก เอนทิตี ว่า ทูเปิล (tuples) และชุดเอนทิตีจะเรียกว่ารีเลชัน ตัวอย่างเช่น คุณสามารถสร้างรีเลชัน ที่เรียกว่า PERSON มีสองแอตทริบิวต์: NAME และ ADDRESS ในกรณีนี้สามารถเขียนได้เป็น PERSON=(NAME, ADDRESS)

3.2.3.3 แอตทริบิวต์

แอตทริบิวต์เป็น รายการข้อมูลที่อธิบายถึงคุณสมบัติของชุดเอนทิตี เราใช้แอตทริบิวต์ในการกำหนด อธิบาย และการจัดประเภทชุดเอนทิตี

แอตทริบิวต์คือ ค่า (value) ซึ่งเป็น ชนิดของข้อมูลต่างๆ เช่นหมายเลข (numbers) อักขระ (character) ข้อความ (string) วัน (dates) รูปภาพ (images) เสียง (sounds) และอื่นๆ โดยที่แอตทริบิวต์ สามารถมีได้เพียงหนึ่งค่าสำหรับแต่ละอินสแตนซ์ของชุดเอนทิตี

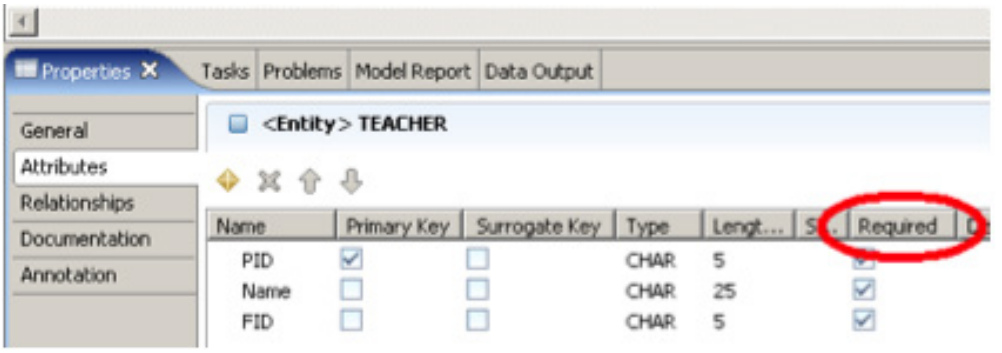
ในแบบจำลองทางกายภาพ แอตทริบิวต์จะใช้เป็นชื่อคอลัมน์ในตาราง และมีโดเมน ซึ่งเป็นชนิดของข้อมูลที่เป็นไปได้ แต่ละตารางข้อมูลประกอบด้วยรายการของแอททริบิวต์ (หรือคอลัมน์)

ชนิดของแอตทริบิวต์ประกอบด้วย

- แอตทริบิวต์ที่มีคุณลักษณะแบบอะตอม (simple atomic attribute) – แอตทริบิวต์ในลักษณะนี้จะมีคอมโพเนนต์เดียว ตัวอย่างเช่น แอตทริบิวต์ Gender จะมีคอมโพเนนต์เดียว ที่มีค่าได้เพียงสองค่า
- แอตทริบิวต์แบบคอมโพสิต (composite attribute) – แอตทริบิวต์แบบผสมที่ประกอบด้วยคอมโพเนนต์แบบต่างๆ เช่น แอตทริบิวต์ ชื่อ (Name) ประกอบด้วยคอมโพเนนต์ ชื่อสกุล (last Name) และ ชื่อตัว (first Name)
- แอตทริบิวต์ที่มีค่าเดียว (single-valued attribute) – แอตทริบิวต์ชนิดนี้มีค่าหนึ่งค่าสำหรับหนึ่งเอนทิตี ตัวอย่างเช่น แอตทริบิวต์ Title มีเพียงค่าเดียวสำหรับแต่ละเอนทิตี teacher
- แอตทริบิวต์แบบหลายค่า (multi-valued attribute) – เป็นแอตทริบิวต์ที่สามารถมีได้หลายค่าสำหรับ เอนทิตีหนึ่ง ตัวอย่างเช่นคนหนึ่งคนสามารถมีเบอร์โทรศัพท์ได้หลายเลขหมาย
- แอตทริบิวต์ที่มีค่ามาจากแอตทริบิวต์อื่น (derived attribute) – เป็นแอตทริบิวต์ที่ มีค่ามาจากการคำนวณจากแอตทริบิวต์ อื่น เช่นผลรวมของนักศึกษาทุกคนในคณะ แอตทริบิวต์ที่มีค่ามาจากแอตทริบิวต์อื่น ไม่ได้เป็นส่วนหนึ่งส่วนใดของตารางข้อมูลจากฐานข้อมูล แต่เพื่อให้เกิดความชัดเจนในการออกแบบ แต่ก็ยังเป็นค่าที่มีความสำคัญกับนักพัฒนาโปรแกรมประยุกต์ที่จะสามารถนำค่านี้ไปใช้ประโยชน์
- แอตทริบิวต์ที่ไม่เสถียร (unstable attributes) – เป็นแอตทริบิวต์ที่มีค่าที่เปลี่ยนแปลงเสมอ ตัวอย่างเช่น ปีการศึกษาของนักเรียน
- แอตทริบิวต์ที่เสถียรภาพ (stable attributes) – เป็นแอตทริบิวต์ที่มี การเปลี่ยนแปลงน้อยครั้ง
- แอตทริบิวต์ที่บังคับให้ต้องมีค่า (mandatory attributes) – เป็นแอตทริบิวต์ที่ถูกบังคับว่าจะต้องมีค่า ตัวอย่างเช่นในองค์กรธุรกิจจำเป็นต้องมีการอ้างถึงชื่อบุคคล ดังนั้นจำเป็นต้องเก็บชื่อของบุคคล (Name) ให้มีค่าอยู่เสมอ

หมายเหตุ:

โปรแกรม InfoSphere Data Architecture เราสามารถเลือกออพชัน required ของแท็บ Attributes ภายในเพจ properties เมื่อต้องการกำหนด แอตทริบิวต์ที่บังคับให้ต้องมีค่า เมื่อทำการเลือกชุดเอนทิตีจากแผนภาพแสดงตารางความสัมพันธ์ของฐานข้อมูล (Entity-Relationship Diagram) ดังแสดงในภาพที่ 3.4



รูปที่ 3.4 - การตั้งค่าแอตทริบิวต์สำหรับฟิลด์ที่บังคับต้องมีค่า

- แอตทริบิวต์ที่เป็นออบชัน (optional attribute) - เป็นแอตทริบิวต์ที่ไม่บังคับว่าจะต้องมีค่าหรือไม่ ซึ่งปรกติแล้วจะสามารถมีค่าเป็นค่าว่างได้ (null value)
- การระบุค่าที่ไม่ซ้ำกัน (unique identifier) - ชนิดของแอตทริบิวต์นี้จะเป็นแอตทริบิต์ที่ต่างจากเอนทิตีปรกติ เนื่องจากมีวัตถุประสงค์เพื่อใช้สำหรับแอตทริบิวต์เพื่อจำแนกความแตกต่างระหว่างข้อมูล ตัวอย่างเช่นข้อมูลของบุคคลในห้องเรียน สามารถจำแนกได้เป็นนักเรียนนาย ก หรือนาย ข โดยอาศัยแอตทริบิวต์รหัสนักเรียน (Student Id)

การสร้างแบบจำลองทางตรรกะจำเป็นต้องมีการระบุค่าที่ไม่ซ้ำกัน ของข้อมูล ซึ่งสิ่งที่ใช้ในการระบุนั้นจะเรียกว่าคีย์ (Key) โดยที่คีย์ ก็คือ ฟิลด์ หรือ กลุ่มของฟิลด์ซึ่ง จะมีค่าที่ไม่ซ้ำกันในแต่ละเรคคอร์ดในรีเลชัน ซึ่งการกำหนดคีย์ในฐานข้อมูลจะทำให้เรามั่นใจได้ว่าข้อมูลที่จัดเก็บนั้นจะไม่เกิดความซ้ำซ้อน สำหรับคีย์ที่อยู่ในรีเลชันนั้นมีอยู่หลายประเภทด้วยกัน ดังรายละเอียดต่อไปนี้

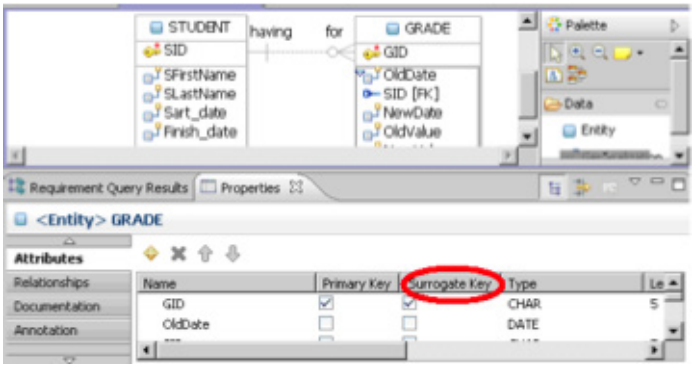
- แคนดิเดตคีย์ (candidate key) – แคนดิเดตคีย์เป็นแอตทริบิวต์ หรือกลุ่มของแอตทริบิวต์ที่ทำให้เรคอร์ดใดเรคอร์ดหนึ่งในรีเลชันมีค่าที่ไม่ซ้ำกัน
- คีย์หลัก (primary key) – คีย์หลักคือคีย์ที่ถูกคัดเลือกมาจากแคนดิเดตคีย์ (candidate key) ในรีเลชัน ในแต่ละรีเลชันควรจะต้องกำหนดให้มีคีย์หลัก สำหรับคุณสมบัติของคีย์หลักควรประกอบด้วย

  Stable - ค่าของคีย์หลักไม่ควรมีการเปลี่ยนแปลง หรือมีค่าเป็นค่าว่าง (null value) ตลอดการ

  ใช้งานของเอนทิตี ตัวอย่างเช่น พิจารณาระเบียนนักเรียน ฟิลด์อายุ Age ไม่เหมาะที่จะเป็นคีย์หลักเนื่องจาก จะมีการเปลี่ยนแปลงค่าเมื่อเวลาผ่านไป

  Minimal - ควรมีค่าที่น้อยที่สุด คีย์หลักควรประกอบด้วยจำนวนฟิลด์ที่น้อยที่สุด เพื่อให้แน่ใจว่าข้อมูลนี้จะไม่เกิดขึ้นซ้ำกัน

- คีย์ทดแทน (alternate key) – คีย์ทดแทนหมายถึงคีย์ใดๆซึ่งอาจจะเป็นหนึ่งในแคนดิเดตคีย์ ที่ไม่ได้ถูกคัดเลือกให้เป็นคีย์หลัก ซึ่ง อาจจะถูกเลือกเพื่อให้เป็นคีย์หลักในอนาคต ถ้าคีย์หลักที่ใช้ไม่มีความเหมาะสม
- คีย์ตัวแทนสำหรับคีย์หลัก - (surrogate key) – คีย์ตัวแทนสำหรับคีย์หลัก ทำหน้าที่ เสมือนเป็นคีย์หลัก แต่คีย์นี้อาจจะไม่ได้มีอยู่จริง แต่เป็นคีย์ที่ถูกสร้างขึ้นมาเพื่อทำหน้าที่แทนคีย์หลักเพื่อใช้สำหรับการอ้างอิงชุดข้อมูลใดข้อมูลหนึ่ง เช่นในรีเลชัน EMPLOYEE จะมีการเก็บข้อมูลอายุ ซึ่งเราอาจจะดึงเอาข้อมูลนี้มาสร้างเป็นรีเลชันใหม่สำหรับเก็บเป็นช่วงอายุแทน และสร้างคีย์ ตัวแทนสำหรับคีย์หลัก นี้สำหรับการอ้างอิงในแต่ละช่วงอายุเป็นต้น ซึ่ง คีย์ตัวแทนสำหรับคีย์หลักโดยปรกติจะไม่นิยมใช้เนื่องจากจะทำให้พื้นที่ในการจัดเก็บมากยิ่งขึ้น สำหรับ InfoSphere Data Architecture มีเครื่องมือสำหรับสร้างคีย์ตัวแทนสำหรับคีย์หลัก นี้ ดังแสดงในรูปที่ 3.5.



รูปที่ 3.5 - ตัวอย่างการเพิ่ม คีย์ตัวแทนสำหรับคีย์หลัก

- Simple Key – คีย์ที่ประกอบด้วยแอตทริบิวต์เดี่ยว
- Composite Key – คีย์ที่ประกอบด้วยแอตทริบิวต์หลายตัวประกอบกัน
- คีย์นอก (Foreign Key) – คีย์นอกที่ใช้สำหรับการอ้างอิงเมื่อมีการสร้างความสัมพันธ์ระหว่างรีเลชันโดยที่คีย์นอกนี้จะปรากฏในส่วนของรีเลชันที่มีความสัมพันธ์ด้วยกับรีเลชันหลัก

3.2.3.4 ชุดความสัมพันธ์

ชุดความสัมพันธ์ เป็นชุดของความสัมพันธ์ระหว่างสองชุดเอนทิตีหรือมากกว่า โดยการเชื่อมความสัมพันธ์ระหว่างสองรีเลชันมักจะอธิบายในลักษณะของคำกริยา (verb) แสดงถึงความสัมพันธ์ของอินสแตนซ์ของชุดความสัมพันธ์และความสัมพันธ์ระหว่างเอนทิตีที่เกี่ยวข้อง

ชุดความสัมพันธ์นั้นจะเกิดขึ้นระหว่างชุดเอนทิตีอย่างน้อยสองชุดเสมอ (หรือความสัมพันธ์ที่เกิดภายในชุดเอนทิตีเอง -recursive relationship)

ชุดความสัมพันธ์สามารถถูกกำหนดโดยสัญลักษณ์ตามที่กำหนดความสัมพันธ์ทางคณิตศาสตร์สำหรับชุดเอนทิตีดังนี้:

ชุดเอนทิตี: E

เอนทิตี: e

ชุดความสัมพันธ์: R

ความสัมพันธ์: r

กำหนดให้ชุดของเอนทิตี E_1, E_2... E_k รีเลชัน R กำหนดกฎที่สอดคล้องกันระหว่างชุดเอนทิตี อินสแตนซ์ $R(E_1, E_2,..., E_K)$ ของรีเลชัน R หมายถึงเอนทิตี้ E_1, E_2... E_k จะอยู่ในรีเลชัน R ของอินแสตนซ์นี้

3.2.3.5 การควบคุมข้อมูล (Constraints)

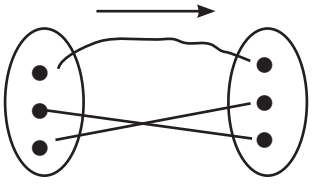
ในแต่ละธุรกิจมักจะมีข้อกำหนดว่าสามารถมีค่าของแอตทริบิวต์เป็นอะไรได้บ้าง หรือยอมให้เกิดความสัมพันธ์ระหว่างกันในรูปแบบใด ซึ่งในแบบจำลองข้อมูลระดับแนวคิดนี้สามารถจัดการกับข้อจำกัดต่างๆเหล่านี้ โดยใช้การควบคุมข้อมูล เพื่อการออกแบบเป็นกฎความสัมพันธ์สำหรับชุดของเอนทิตี ซึ่งการควบคุมข้อมูลเหล่านี้อาจจะหมายถึงกฎที่บังคับใช้ภายในรีเลชัน ตัวอย่างเช่น “ครูทุกคนต้องทำงานสังกัดอยู่ในแผนกเพียงแค่หนึ่งแผนกเดียวเท่านั้น” ซึ่งก็จะมีผลบังคับต่อการสร้างความสัมพันธ์ระหว่างเอนทิตี TEACHER และ DEPARTMENT

ประเภทของการควบคุมข้อมูล คือ:

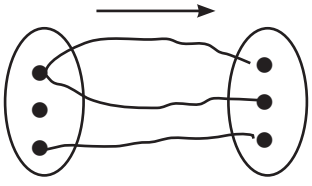
- Cardinalities – คาร์ดินัลลีตีเป็นการกำหนดจำนวนของชุดความสัมพันธ์ที่เป็นไปได้ในการเชื่อมโยงระหว่างชุดเอนทิตี ซึ่งคาร์ดินัลลิตีแต่ละประเภทจะสามารถอธิบายได้ตามความสัมพันธ์ข้างล่าง

กำหนดให้รีเลชัน R เป็นความสัมพันธ์ระหว่าง E (เอนทิตีที่ปรากฏอยู่ทางด้านซ้ายมือ) และ F (เอนทิตีที่ปรากฏอยู่ทางด้านขวามือ) ดังนั้นเราสามารถอธิบายรีเลชัน R ได้ดังต่อไปนี้

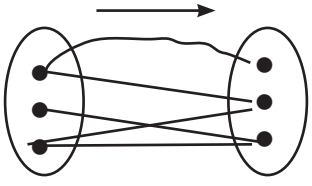
- ความสัมพันธ์ แบบหนึ่งต่อหนึ่ง (1:1) - ถ้าทั้ง E และ F มีค่าเดี่ยวที่เชื่อมโยงระหว่างกัน ดังแสดงในรูปต่อไปนี้



- ความสัมพันธ์แบบหนึ่งต่อกลุ่ม (1:M) - ถ้า E มีค่าเดียว และ F มีส่วนร่วมหลายค่าตามที่อธิบายในรูปด้านล่าง



- ความสัมพันธ์แบบกลุ่มต่อกลุ่ม (M:M) - ถ้าทั้ง E และ F มีค่าที่ร่วมกันหลายค่า ดังรูปต่อไปนี้



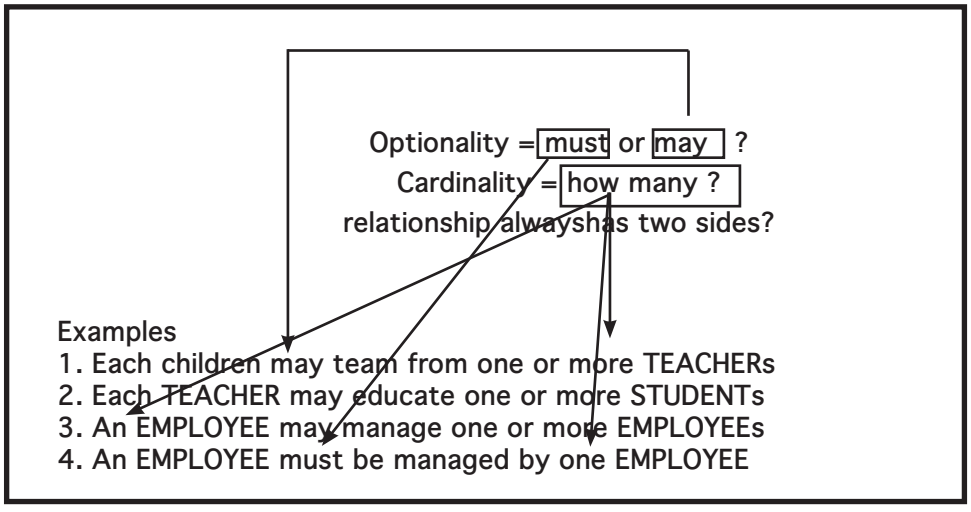
สำหรับความสัมพันธ์แบบกลุ่มต่อกลุ่มไม่สามารถนำไปใช้ในแบบจำลองเชิงสัมพันธ์ ซึ่งจะต้องทำการแก้ไขโดยแบ่งชุดความสัมพันธ์ให้เป็นแบบหนึ่งต่อกลุ่ม 1:M เป็นจำนวนสองชุด

- Participation Cardinalities

Participation cardinalities สามารถกำหนดเป็นเงื่อนไขเพิ่มเติมเพื่อใช้ควบคุมข้อมูลโดยการระบุถึงชุดเอนทิตี ที่เกี่ยวข้องกับชุดเอนทิตีอื่น ผ่านชุดความสัมพันธ์ โดยสามารถจำแนกได้เป็น 2 ประเภท ดังต่อไปนี้

- Total or mandatory เป็นการบังคับว่าชุดเอนทิตีนั้นจะต้องเข้าร่วมในความสัมพันธ์แบบทั้งหมด เราเรียกความสัมพันธ์แบบนี้ว่าแบบ compulsory
- Partial or Optional เป็นการกำหนดว่าชุดเอนทิตีนั้นมีส่วนร่วมในความสัมพันธ์แบบบางส่วน หรือเป็นตัวเลือก หรือเรียกว่าแบบ non-compulsory

รูป 3.6 สรุปทั้ง cardinality และข้อจำกัดของ optionality



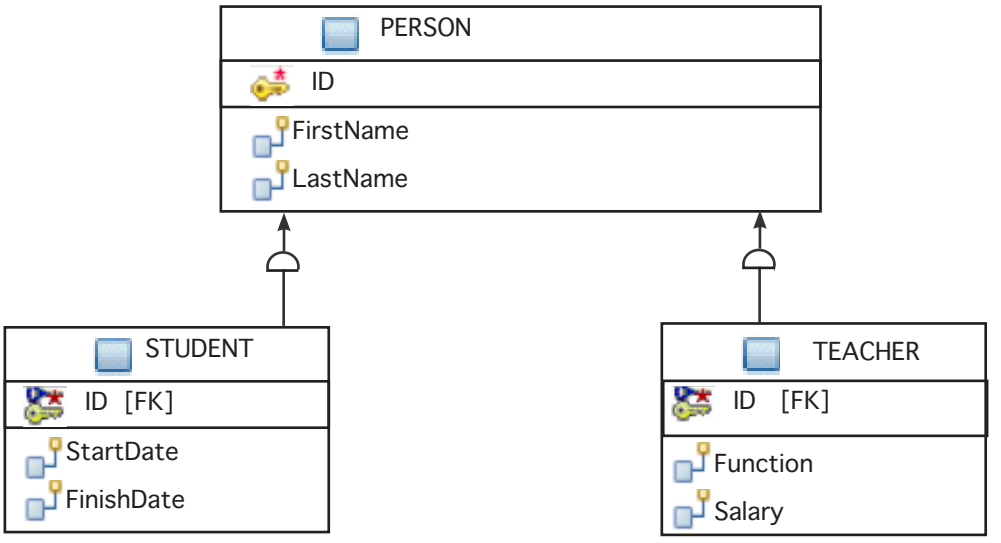
รูปที่ 3.6 - รูปแบบความสัมพันธ์แบบ Cardinality และ optionality

- ซับเซตและซูเปอร์เซต

เมื่อกลุ่มของอินสแตนซ์มีคุณสมบัติพิเศษเช่น มีแอตทริบิวต์ หรือ ชุดความสัมพันธ์ที่ค่าสำหรับกลุ่มของอินสแตนซ์เท่านั้น ทำให้เราสามารถแบ่งชุดเอนทิตี้ดังกล่าว เป็น ซูเปอร์เซตและซับเซต โดยที่ชุดเอนทิตีที่เป็นซูเปอร์เซตเรียกว่าเอนทิตีแม่ (Parent) และชุดเอนทิตีที่เป็นซับเซต เรียกว่าเอนทิตีลูก (Child) โดยที่ความสัมพันธ์ของชุดเอนทิตีลักษณะนี้เรียกว่าเป็นความสัมพันธ์แบบแม่-ลูก (parent-child relationship) โดยที่ชุดเอนทิตีเหล่านี้จะมีคุณสมบัติที่เหมือนหรือคล้ายกัน เช่นคุณสมบัติของเอนทิตีลูกอาจจะเหมือนกับคุณสมบัติของเอนทิตีแม่ โดยที่โครงสร้างของความสัมพันธ์จะสามารถแสดงเป็นลำดับชั้น (hierarchical structure)

ตัวอย่างเช่น

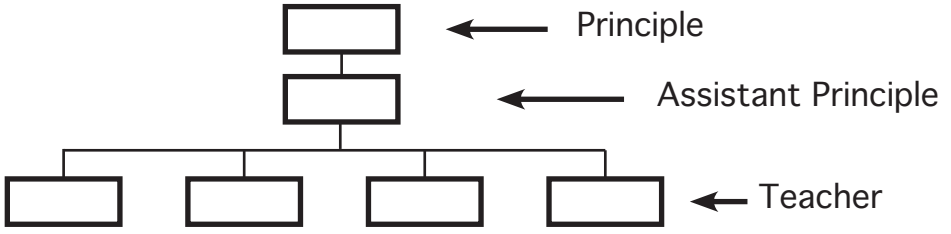
การแสดงโครงสร้างความสัมพันธ์แบบ parent-child สำหรับชุดเอนทิตี PERSON สามารถแบ่งออกเป็น ซับเซต STUDENT และ TEACHER ตามที่กล่าวถึงในรูปที่ 3.7 ด้านล่าง



รูปที่ 3.7- ซูเปอร์เซต PERSON และซับเซต STUDENT และ TEACHER

- ลำดับชั้น (hierarchy)

  การแสดงรายการต่างๆที่เป็นแบบลำดับชั้น ดังตัวอย่างเช่นโครงสร้างของตำแหน่งของครูในโรงเรียน ประกอบด้วย Principle, Assistant principle และ Teacher ดังแสดงใน รูปที่ 3.8

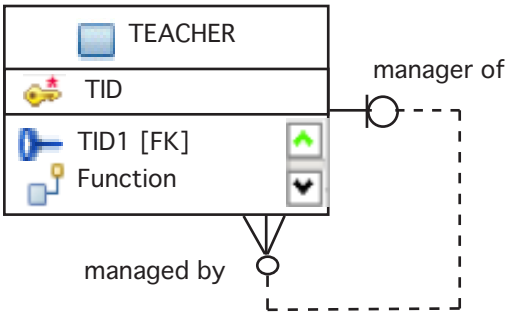


รูปที่ 3.8 - ตัวอย่างของลำดับชั้นในโรงเรียน

- ความสัมพันธ์ที่อยู่ภายในชุดเดียวกัน (Unary relationship set) - ความสัมพันธ์ในลักษณะนี้เป็นความสัมพันธ์ที่เกิดขึ้นภายในตัวเอง ซึ่งเราเรียกความสัมพันธ์ในลักษณะนี้ว่าเป็น recursive relationship set

  ตัวอย่างเช่น

ตำแหน่ง Assistant teacher ซึ่งที่ จริงแล้วก็คือ Teacher ดังนั้นจึงสามารถแสดงเป็นความสัมพันธ์ภายในรีเลชัน Teacher เป็นความสัมพันธ์ที่เกิดขึ้นภายในตัวเอง ดังแสดงในรูป 3.9 จะเห็นว่าเป็นความสัมพันธ์ที่เกิดขึ้นระหว่างหน้าที่ของครู ซึ่งหมายความถึงครูนั้นทำหน้าที่อยู่หลายบทบาท (roles) หมายถึงเป็นทั้งครูผู้สอน (Teacher) และเป็นทั้งผู้ช่วยผู้อำนวยการของโรงเรียน (Assistant principal) ด้วย



รูปที่ 3.9 – ตัวอย่างของชุดความสัมพันธ์ unary

ในบางกรณี เราต้องการตรวจสอบและยืนยันความถูกต้องของข้อมูลที่เกิดขึ้นในฐานข้อมูลในระหว่างที่ทำงาน เนื่องจากเมื่อค่าของแอตทริบิวต์มีการเปลี่ยนแปลง ก็อาจจะมีผลกระทบต่อการเปลี่ยนแปลงในชุดความสัมพันธ์ด้วย ดังนั้นเราสามารถใช้ข้อมูลประวัติ เป็นเสมือนการควบคุมข้อมูลผ่านเงื่อนไขของเวลา ตัวอย่างเช่น การกำหนดเงื่อนไขให้วันที่สิ้นสุดนั้นจะต้องเกิดขึ้นหลังจากวันที่เริ่มต้นเสมอ ในแบบจำลองข้อมูลระดับกายภาพ (physical data model) เราสามารถใช้เงื่อนไข CHECK สำหรับควบคุมและตรวจสอบเงื่อนไขของข้อมูลให้เป็นไปตามที่กำหนด เช่นการตรวจสอบวันเริ่มต้นและวันสิ้นสุดให้ไม่สามารถที่จะเป็นวันเดียวกันได้ เป็นต้น

ตัวอย่างเช่น

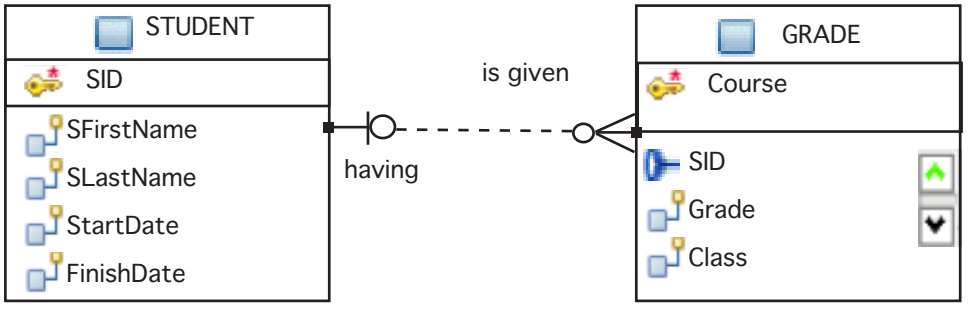
วันที่นักเรียนสำเร็จการศึกษาจะต้องเป็นวันที่หลังจากเริ่มต้นการศึกษาเท่านั้น ในตัวอย่างนี้วันที่สิ้นสุดการศึกษาจะต้องมีค่ามากกว่าวันที่เริ่มต้นการศึกษา

หมายเหตุ: การเพิ่มการควบคุมข้อมูล check สามารถดูรายละเอียดเพิ่มเติมได้ใน eBook เรื่อง Getting Started with InfoSphere Data Architect ซึ่งเป็นส่วนหนึ่งของชุดหนังสือ eBook

นอกจากนี้ในกรณีที่จำเป็นที่จะต้องเก็บค่าในอดีตไว้ ก่อนที่จะมีการเปลี่ยนแปลงข้อมูลใดๆ โดยเฉพาะอย่างยิ่งกรณีที่เป็นข้อมูลสำคัญ เช่นเมื่อนักศึกษามาขอแก้ไขเกรด เป็นต้น

ตัวอย่างเช่น

กรณีที่เกรดของนักศึกษาถูกแก้ไข เราจำเป็นต้องมีการบันทึกเป็นเรคอร์ดว่าข้อมูลเกรดนี้ถูกเปลี่ยนแปลงเมื่อไร เกรดเดิมเป็นอะไร เกรดใหม่ และใครที่เป็นผู้แก้ไขข้อมูลชุดนี้ ดังแสดงในรูปที่ 3.10 แสดงถึงการจัดเก็บประวัติของการเปลี่ยนแปลงนี้



รูปที่ 3.10 - ตัวอย่างการบันทึกประวัติการเปลี่ยนแปลง

3.2.3.6 โดเมน (Domain)

โดเมนหมายถึงชุดของค่าของข้อมูลสำหรับแอตทริบิวต์ที่สามารถเป็นไปได้ ผู้ใช้งานสามารถกำหนดโดเมนขึ้นมาเอง ซึ่งโดเมนจะยอมให้ผู้ใช้กำหนดเป็นชนิดของข้อมูลขึ้นมาใหม่เพื่อใช้ให้เหมาะสมกับธุรกิจหนึ่งๆ ได้

หมายเหตุ: ในโปรแกรม InfoSphere Data Architect เราสามารถใช้โดเมนเพื่อจำกัดค่าและรูปแบบของชนิดข้อมูลที่เหมาะสมกับชนิดข้อมูลเชิงธุรกิจได้

สำหรับการกำหนดเงื่อนไข CHECK ในคำสั่งมาตรฐาน SQL เป็นการกำหนดให้การตรวจสอบโดเมนมีความเข้มงวดมากยิ่งขึ้น เงื่อนไข CHECK ยอมให้นักออกแบบฐานข้อมูลระบุว่าค่าที่จะป้อนเข้ามานั้นต้องเป็นไปตามค่าที่กำหนดเอาไว้ในโดเมนเท่านั้น

3.2.3.7 เอกซเทนชั่น (Extension)

ระบบฐานข้อมูลนั้นมีการเปลี่ยนแปลงข้อมูลอยู่ตลอดเวลา ข้อมูลในฐานข้อมูล ณ เวลาใดเวลาหนึ่งนั้นถูกเรียกว่าเอกซเทนชั่น ของฐานข้อมูล เอกซเทนชั่นจะอ้างถึงชุดของทูเปิลปัจจุบันในรีเลชัน

3.2.3.8 อินเทนชั่น (Intension)

อินเทนชั่น หรือ แบบโครงร่างของฐานข้อมูล(schema) เป็นแบบจำลองทางตรรกะของฐานข้อมูล ซึ่งจะแสดงถึงชุดเอนทิตี อินแสตนซ์ จะอธิบายถึงโครงสร้างของทูเปิลที่ยอมให้เกิดขึ้นในรีเลชันได้ ซึ่งอินแสตนซ์นั้นจะนำเสนอโดยชุดเอนทิตีในรูปของแบบจำลองของฐานข้อมูล โดยที่การจัดการกับข้อมูลบนทูเปิลจะสามารถทำได้ถ้าไม่ขัดต่อกฎเงื่อนไขความสัมพันธ์ข้อมูล

3.3 กรณีศึกษาที่เกี่ยวข้องกับการจัดการระบบห้องสมุด - ส่วนที่ 1 ของ 3

สำหรับในกรณีศึกษานี้จะอธิบายถึงการจัดการระบบห้องสมุดอย่างง่าย โดยสมมุติว่าความต้องการของระบบ ถูกกำหนดขึ้นมาโดยผู้ใช้งานระบบ ดังนี้

> “ระบบจะจัดการข้อมูลเกี่ยวกับผู้แต่งหนังสือและผู้ยืมหนังสือ โดยจะเก็บข้อมูลรายละเอียดของหนังสือที่ยืม ข้อมูลของผู้ยืมประกอบด้วย ชื่อ ที่อยู่ อีเมล์ และเบอร์โทรศัพท์ รวมถึงรายละเอียดสำหรับผู้แต่งหนังสือประกอบด้วยชื่อ ที่อยู่ และอีเมล์
>
> ข้อมูลของหนังสือใหม่ ชื่อผู้แต่ง ชื่อผู้ยืม จะต้องถูกบันทึกลงไปในระบบ เมื่อผู้ยืมทำการยืมหนังสือ ระบบจะต้องทำการบันทึกวันที่ยืมหนังสือและคำนวณจำนวนวันที่หนังสือเล่มนั้นๆสามารถถูกยืมได้ รวมถึงวันที่ต้องคืน ซึ่งหากผู้ยืมทำการคืนหนังสือช้ากว่าที่กำหนด ผู้ยืมหนังสือคนนั้นจะต้องจ่ายค่าปรับตามจำนวนวันที่ยืมหนังสือเกินไป”

3.3.1 การพัฒนาแบบจำลองในระดับแนวคิด (Conceptual model)

ขั้นตอนต่อไปนี้เป็นขั้นตอนมาตรฐานสำหรับการพัฒนาแบบจำลองในระดับแนวคิด

ขั้นตอนที่ 1 - การระบุเอนทิตี

สำหรับขั้นตอนของการระบุชุดเอนทิตี เราจำเป็นต้องมีการวิเคราะห์ความต้องการของผู้ใช้งานอย่างรอบคอบ และมุ่งเน้นคำที่เป็นคำนาม เราสามารถสร้างคำนามเหล่านี้ในลักษณะของรายการ และแสดงทุกรายการรวมถึงที่ซ้ำกัน (ซึ่งอาจจะมีการตัดออกในภายหลังสำหรับสิ่งที่ไม่เกี่ยวข้อง) ซึ่งตัวอย่างสำหรับโจทย์ของ

ห้องสมุดนี้ เราจะได้รายการของคำที่เป็นคำนาม ดังนี้

books, authors, name, address, e-mail, loaner, client, borrowers, name, address, e-mail, phone, loan date, return date, loan days, fine

สำหรับคำนามแต่ละตัวนั้นสามารถที่จะนำไปสร้างเป็นชุดเอนทิตีได้ แต่บางตัวจะสามารถเป็นเพียงแค่แอตทริบิวต์ ซึ่งเราจะต้องตัดสินใจโดยเริ่มต้นจากการวิเคราะห์หาความสัมพันธ์และความขึ้นอยู่กับ (dependencies) เช่นตัวอย่างนี้

name, address, e-mail จะขึ้นอยู่กับ authors

name, address, e-mail และ phone จะขึ้นอยู่กับ borrowers

สำหรับการวิเคราะห์เบื้องต้น เราได้กำหนดให้แบบจำลองประกอบด้วยเอนทิตีต่อไปนี้ LIBRARY, BOOK, AUTHOR, BORROWER, CLIENT และ LOANER ซึ่งจะได้ชุดเอนทิตีเป็น

No.	Entity set
1	LIBRARY
2	BOOKS
3	AUTHORS
4	BORROWERS
5	CLIENTS
6	LOANERS

ขั้นตอนที่ 2 - การตัดเอนทิตีที่ซ้ำกันออก

เมื่อวิเคราะห์ความต้องการจากผู้ใช้งาน จะพบว่ามีคำศัพท์หลายๆ คำ ที่อาจจะเกิดขึ้น โดยคำเหล่านั้นมีความหมายเดียวกัน ซึ่งเราจะต้องวิเคราะห์และแสดงให้เห็นถึงความแตกต่างของคำศัพท์ที่อยู่ในแต่ละชุดเอนทิตี ตัวอย่างเช่นกลุ่มคำที่มีความหมายเหมือนกัน เช่น borrowers, loaners และ clients ซึ่งล้วนแล้วแต่สื่อความหมายเดียวกัน ซึ่งในกรณีนี้เราเลือกที่จะใช้คำศัพท์ borrowers

ทั้งนี้เราไม่ควรนำเอา สิ่งที่เป็นภาพรวม มารวมอยู่ในชุดเอนทิตี ดังกรณีนี้เรากำลังสร้างแบบจำลองสำหรับห้องสมุด เพราะฉะนั้นเราไม่ควรจะมี library อยู่ในชุดเอนทิตีของเรา

ในขั้นตอนสุดท้ายนี้จะต้องตัดสินใจกำหนดประเภท ของชุดเอนทิตีว่าเป็นแบบ weak หรือ strong ความหมายของชุดเอนทิตีแบบ weak นั้นจะหมายถึงว่าการเกิดของชุดเอนทิตีนั้นจะต้องขึ้นอยู่กับ ชุดเอนทิตีอื่น และชุดเอนทิตีแบบ strong นั้นจะหมายถึงชุดเอนทิตีที่สามารถเกิดขึ้นได้ด้วยตัวเอง ไม่ขึ้นอยู่กับชุดเอนทิตีใดๆ ซึ่งชนิดของชุดเอนทิตีนั้นจะมีผลต่อการวิเคราะห์ถึงประเภทของความสัมพันธ์ต่อไป

สำหรับผลลัพธ์จากการวิเคราะห์ในขั้นตอนนี้ สามารถแสดงในรูปของเอนทิตีได้ดังนี้

No.	Entity set	Type
1	BOOK	Strong
2	AUTHOR	Strong
3	BORROWER	Strong

ขั้นตอนที่ 3 - แสดงรายการแอตทริบิวต์สำหรับในแต่ละชุดเอนทิตี

สำหรับในขั้นตอนนี้ เพื่อให้มั่นใจว่าชุดเอนทิตีนี้ เป็นสิ่งที่ต้องมีจริงๆ เราจะต้องทำการวิเคราะห์ว่าคำนามที่พบนั้นควรเป็นชุดเอนทิตีหรือแอตทริบิวต์ เช่น เบอร์โทรศัพท์ PHONE ควรจะเป็นชุดเอนทิตีหรือเป็นแอตทริบิวต์ที่อยู่ในชุดเอนทิตี AUTHOR?

ในที่นี้ เราควรมีการหารือกับผู้ใช้งานเกี่ยวกับแอตทริบิวต์อื่นๆที่จำเป็น ซึ่งจากตารางตัวอย่างต่อไปนี้เป็นการแสดงถึงรายการแอตทริบิวต์ของทุกๆ ชุดเอนทิตีที่ได้วิเคราะห์มาทั้งหมด

BORROWER entity set

Attribute name	Type	Domain	Optional
BORROWER_ID	Unique identifier	Text	No
NAME	Composite attribute	Text	No
EMAIL	Single valued attribute	Text	Yes
PHONE	Multi-valued attribute	Text	Yes
ADDRESS	Composite attribute	Text	Yes
BOOK_ID	Single valued attribute	Text	No
LOAN_DATE	Single valued attribute	Text	No
DUE_DATE	Derived attribute	Text	No
RETURN_DATE	Derived attribute	Text	No

AUTHOR entity set

Attribute name	Type	Domain	Optional
AUTHOR_ID	Unique identifier	Text	No
NAME	Composite attribute	Text	No
EMAIL	Single valued attribute	Text	Yes
PHONE	Multi-valued attribute	Text	Yes
ADDRESS	Composite attribute	Text	Yes

BOOK entity set

Attribute name	Type	Domain	Optional
BOOK_ID	Unique identifier	Text	No
TITLE	Single valued attribute	Text	No
EDITION	Single valued attribute	Numeric	Yes
YEAR	Single valued attribute	Numeric	Yes
PRICE	Single valued attribute	Numeric	Yes
ISBN	Single valued attribute	Text	Yes

PAGES	Single valued attribute	Numeric	Yes
AISLE	Single valued attribute	Text	Yes
DECRIPTION	Single attribute	Text	Yes

ตามที่กล่าวไว้ในหัวข้อที่ 3.2.3.3 มีบางสิ่งที่ต้องพิจารณาเกี่ยวกับแอตทริบิวต์ที่ระบุไว้ในตารางข้างต้น:

- ทำการตัดสินใจเกี่ยวกับแอตทริบิวต์แบบคอมโพสิต (Composite attribute) โดยมองว่าจะรวมกันไว้หรือแยกออกจากกัน ทั้งนี้จะต้องขึ้นอยู่กับการนำไปใช้ประโยชน์ ในกรณีตัวอย่างนี้ลูกค้าต้องการที่จะสามารถสืบค้นข้อมูลจากชื่อหรือนามสกุลได้ ดังนั้นการจัดเก็บข้อมูลชื่อนี้ควรจะเก็บแยกจากกัน โดยแยก NAME ออกเป็น FIRST_NAME และ LAST_NAME
- เมื่อพบค่าแอตทริบิวต์แบบหลายค่า (Multi-value attribute) เราจำเป็นต้องแยกแอตทริบิวต์นี้ให้เป็นชุดเอนทิตีใหม่ ซึ่งในกรณีนี้เราอาจจะไม่จำเป็นต้องมีเบอร์โทรศัพท์หลายเบอร์ ดังนั้นจึงต้องทำการเปลี่ยนชนิดของแอตทริบิวต์จากหลายค่า (Multi-value) ให้เป็นค่าเดียว (Single value)
- แอตทริบิวต์ DUE_DATE ถูกสร้างมาจากแอตทริบิวต์ LOAN_DATE ซึ่งเรียกว่า แอตทริบิวต์ที่มีค่ามาจากแอตทริบิวต์อื่น จากตัวอย่างหนังสือภายในห้องสมุดจะสามารถถูกยืมได้ไม่เกิน 10 วัน ดังนั้นแอตทริบิวต์ DUE_DATE สามารถคำนวณได้จากแอตทริบิวต์ LOAN_DATE + 10

หลังจากการวิเคราะห์นี้ เราจะได้ผลลัพธ์ที่เป็นแอตทริบิวต์สำหรับทุกชุดเอนทิตีดังต่อไปนี้

BORROWER entity set

Attribute name	Type	Domain	Optional
BORROWER_ID	Unique identifier	Text	No
FIRST_NAME	Single valued attribute	Text	No
LAST_NAME	Single valued attribute	Text	No
EMAIL	Single valued attribute	Text	Yes
PHONE	Single valued attribute	Text	Yes
ADDRESS	Composite attribute	Text	Yes
BOOK_ID	Single valued attribute	Text	No
LOAN_DATE	Single valued attribute	Text	No
RETURN_DATE	Single valued attribute	Text	No

AUTHOR entity set

Attribute name	Type	Domain	Optional
AUTHOR_ID	Unique identifier	Text	No
FIRST_NAME	Single valued attribute	Text	No
LAST_NAME	Single valued attribute	Text	No
EMAIL	Single valued attribute	Text	Yes
PHONE	Single valued attribute	Text	Yes
ADDRESS	Composite attribute	Text	Yes

BOOK entity set

Attribute name	Type	Domain	Optional
BOOK_ID	Unique identifier	Text	No
TITLE	Single valued attribute	Text	No
EDITION	Single valued attribute	Numeric	Yes
YEAR	Single valued attribute	Numeric	Yes
PRICE	Single valued attribute	Numeric	Yes
ISBN	Single valued attribute	Text	Yes
PAGES	Single valued attribute	Numeric	Yes
AISLE	Single valued attribute	Text	Yes
DECRIPTION	Single valued attribute	Text	Yes

ขั้นตอนที่ 4 – การคัดเลือกตัวระบุ (identifier) ที่ไม่ซ้ำกัน

สำหรับชุดเอนทิตี BOOK เราสามารถเลือกแอตทริบิวต์ BOOK_ID หรือ ISBN เพื่อใช้เป็นคีย์อ้างอิง ซึ่งจากตัวอย่างนี้ เราควรเลือก BOOK_ID เนื่องจากเป็นคีย์ที่ไม่มีการซ้ำกันในระบบห้องสมุด นอกจากนี้การยืม-คืนหนังสือถ้าต้องมีการพิมพ์เลข ISBN ซึ่งต้องพิมพ์หมายเลขทั้งหมด จะทำให้ไม่สะดวกต่อการทำงานของเจ้าหน้าที่ห้องสมุด

ขั้นตอนที่ 5 - การกำหนดชุดความสัมพันธ์

ในขั้นตอนนี้เป็นการตรวจสอบความสัมพันธ์ระหว่างชุดเอนทิตี ซึ่งเราสามารถหาชุดความสัมพันธ์สำหรับแต่ละคู่ ของชุดเอนทิตี ตัวอย่างเช่น หนังสือ (BOOK) จะถูกเขียนหรือแต่งขึ้นโดยบุคคล (PERSON) หรือหนังสือ (BOOK) จะสามารถถูกยืม/คืนโดยบุคคล (PERSON) เช่นเดียวกัน

จะต้องอธิบายถึงคาร์ดินัลลิตี (cardinality) ทั้งแบบเข้มงวด (mandatory) และยืดหยุ่น (optionally) ซึ่งสามารถนำมาใช้ประกอบการวิเคราะห์เพื่อลดปัญหาในเรื่องของความซ้ำซ้อน และดีกรีของความสัมพันธ์แบบ strong หรือ weak ซึ่งความสัมพันธ์ที่เป็นแบบ weak นั้นจะเป็นความสัมพันธ์ระหว่างชุดเอนทิตีแบบ strong และชุดเอนทีตีแบบ weak ในขณะเดียวกัน ความสัมพันธ์ที่เป็นแบบ strong นั้นก็จะเป็นความสัมพันธ์ระหว่างชุดเอนทิตีสองชุดที่เป็นแบบ strong ในโปรแกรม InfoSphere Data Architect เราเรียกความสัมพันธ์ของทั้งสองลักษณะนี้ว่าเป็นแบบ identifying และ non-identifying สำหรับชุดความสัมพันธ์แบบ identifying นั้นเป็นการกำหนดว่า child เป็นเอนทิตีที่ต้องขึ้นอยู่กับอีกเอนทิตีหนึ่ง ส่วน non-identifying นั้นเป็นลักษณะที่ทั้งสองเอนทิตีไม่ได้ขึ้นอยู่ต่อกันและกันเลย

ในการระบุชุดความสัมพันธ์จากกรณีศึกษาระบบห้องสมุด เราสามารถวิเคราะห์ความสัมพันธ์ของชุดเอนทิตีตามสถานการณ์ (scenarios) ตามตารางต่อไปนี้:

	Relationship set	Identifying	Left verb	Right verb	Cardinality	Optionality
1	BORROWER -> BOOK	No	Borrows	Be borrowed	Many-to-many	May
2	AUTHOR - > BOOK	No	Write	Is written	Many-to-many	May
3	AUTHOR - > BOOK	No	Borrows	Be borrowed	Many-to-many	May

จากตารางข้างต้น แสดงชุดความสัมพันธ์แบบ กลุ่มต่อกลุ่ม (many to many) สามชุดซึ่งจะต้องทำการแก้ไข โดย ทำการแยกย่อย (decompose) ชุดความสัมพันธ์ชุดแรกที่เป็นกลุ่มต่อกลุ่ม ให้อยู่ในรูปของ หนึ่งต่อกลุ่ม (one to many) สองชุด โดยเราจะได้ผลลัพธ์ดังนี้

1. BORROWER -> BOOK

ชุดเอนทิตี COPY ถูกสร้างขึ้นมาเพื่อเป็นชุดเอนทิตีกลางที่จะลบความสัมพันธ์แบบกลุ่มต่อกลุ่ม ระหว่าง BORROWER และ BOOK โดยจะมีลักษณะดังนี้

COPY entity set

Attribute name	Type	Domain	Optional
COPY_ID	Unique identifier	Text	No
STATUS	Single valued attribute	Text	No

2. AUTHOR -> BOOK

ชุดเอนทิตี AUTHOR_LIST ถูกสร้างขึ้นมาเพื่อลบชุดความสัมพันธ์แบบกลุ่มต่อกลุ่ม-ระหว่าง AUTHOR และ BOOK ชุดเอนทิตีใหม่จะมีลักษณะดังนี้:

AUTHOR_LIST entity set

Attribute name	Type	Domain	Optional
ROLE	Single valued attribute	Text	No

ดังนั้นผลลัพธ์ที่เกิดจากการปรับปรุงจะได้ตารางที่มีการกำหนดค่าความสัมพันธ์:

Relationship set	Identifying	Left verb	Right verb	Cardinality	Optionality
BORROWER - > COPY	No	Borrows	Be borrowed	One-to-many	May
BOOK - > COPY	No	Has	Is created	One-to-many	May
AUTHOR - > AUTHOR_LIST	No	Appear	Has	One-to-many	May
AUTHOR_LIST - > BOOK	No	Is created	Has	Many-to-many	May

3. AUTHOR -> BOOK

ชุดความสัมพันธ์ที่สามที่จะต้องทำการแยกย่อย ได้ผลลัพธ์ดังนี้

Relationship set	Identifying	Left verb	Right verb	Cardinality	Optionality
AUTHOR - > BOOK	No	Borrows	Be borrowed	Many-to-many	May

สำหรับกรณีศึกษานี้ เราอาจสงสัยว่าทำไมต้องมีชุดเอนทิตี AUTHORS และ BORROWERS ทั้งๆที่แอตทริบิวต์ส่วนใหญ่เหมือนกัน ทำไมแบบจำลองไม่ใช้เพียงหนึ่ง เอนทิตีคือ PERSONS ซึ่งเป็นซูเปอร์เซตของ AUTHORS และ BORROWERS ซึ่งสามารถเป็นได้ทั้งผู้เขียนและผู้ยืมหนังสือ

จริงๆ แล้วเราสามารถใช้แบบจำลองสำหรับเอนทิตีเดียว PERSONS แต่อย่างไรก็ตามมีเหตุผลอยู่สองประการในการที่จะต้องแยก ออกเป็นสองชุดเอนทิตี:

- ความง่ายในการทำความเข้าใจกับแบบจำลองนี้ (ส่วนใหญ่เป็น pedagogical เหตุผลสำหรับผู้อ่านเพื่อให้เข้าใจแนวคิดได้ง่ายขึ้น)
- ความง่าย ในการพัฒนาโปรแกรมประยุกต์สำหรับรูปแบบนี้ สมมุติว่า ระบบนี้มีไว้สำหรับห้องสมุดของเมืองขนาดเล็กมาก และโอกาสของผู้ที่เขียนหนังสือจะอาศัยอยู่ในชุมชนนี้มีน้อย ซึ่งหมายความว่าโอกาสของผู้ที่จะยืมหนังสือจะน้อยตามไปด้วย ในสถานการณ์เช่นนี้คงเป็นเรื่องแปลกที่บรรณารักษ์ของห้องสมุดจะมีการสร้างบัญชีในแก่ผู้เขียนหนังสือเป็นผู้ยืมหนังสือของตนเอง ซึ่งแน่นอนว่าปัญหาเหล่านี้บางอย่างอาจจะต้องมีการวิเคราะห์เพิ่มเติมเกี่ยวกับกฎทางธุรกิจอีกด้วย

สำหรับข้อควรพิจารณาอื่น ๆ ที่จะกล่าวถึงในภายหลังจะเกี่ยวข้องกับการควบคุมข้อมูลที่เก็บรวบรวม ตัวอย่างเช่น ควรมีการควบคุมข้อมูล ในการ ตรวจสอบว่า วันที่คืนหนังสือจะต้องเป็นวันหลังจากที่ยืมเสมอเป็นต้น ซึ่งเราจะกล่าวเพิ่มเติมเกี่ยวกับเรื่องนี้เมื่ออธิบายเกี่ยวกับการควบคุมข้อมูลในแบบจำลองทางกายภาพ ซึ่งจะกล่าวถึงในบทที่ 5

ขั้นตอนที่ 6 - กำหนดกฎทางธุรกิจ (business rules)

กฎทางธุรกิจสามารถนำมาใช้ได้ โดยโปรแกรมประยุกต์ ในตัวอย่างของเราจะพิจารณากฎต่อไปนี้:

- ผู้ดูแลระบบเท่านั้นที่สามารถเปลี่ยนแปลงข้อมูล
- อนุญาตให้มีระบบจัดการการจองหนังสือ ผู้ยืมแต่ละคนที่จองหนังสือสามารถตรวจสอบสถานะของหนังสือที่สนใจได้
- โปรแกรมประยุกต์ส่งอีเมลเตือนการยืม เพื่อแจ้งเตือนทางอีเมล์ก่อนล่วงหน้า 3 วันก่อนครบกำหนดคืนหนังสือ
- ถ้าผู้ยืมหนังสือไม่ทำการส่งคืนหนังสือทันเวลา จะต้องถูกปรับเป็นเงิน 0.1% ของราคาหนังสือในแต่ละวัน

กรณีศึกษานี้นำเสนอและให้ข้อแนะนำเพื่อสามารถนำไปใช้เป็นแนวปฏิบัติ อย่างไรก็ตามแนวคิดการสร้างแบบจำลองเป็นกระบวนการที่ต้องทำซ้ำ ดังนั้นเราอาจจะต้องสร้างเป็นแบบจำลองขึ้นมาหลายเวอร์ชันด้วยกัน โดยอาจจะไม่มีคำตอบที่ดีที่สุด หรือถูกต้องกับการแก้ไขปัญหาที่สุด แต่ก็จะทำให้เราเห็นว่าบางแบบจำลองอาจจะมีข้อดีและข้อเสียที่แตกต่างกันไป

กรณีศึกษานี้ยังคงใช้ในบทที่ 4 เมื่อเรากล่าวถึงการออกแบบแบบจำลองทางตรรกะสำหรับระบบห้องสมุดนี้

3.4 บทสรุป

บทนี้กล่าวถึงการสร้างแบบจำลองระดับแนวคิด ซึ่งจะอธิบายวิธีการทำงานกับไดอะแกรมความสัมพันธ์ของเอนทิตี และอธิบายแนวคิดที่แตกต่างกันเช่นชุดเอนทิตี แอตทริบิวต์ ความสัมพันธ์ การควบคุมข้อมูล โดเมน และอื่นๆ สามารถใช้เครื่องมือกราฟิกเช่น InfoSphere Data Architect โดยเฉพาะเมื่อเรามีโครงการที่ซับซ้อน มีการสร้าง แบบจำลองระดับแนวคิด และ ใช้แบบจำลองดังกล่าวร่วมกัน สำหรับทีมงานที่ดูแลแบบจำลองทางกายภาพและแบบจำลองทางตรรกะ แนวคิดพื้นฐานของ การสร้างแบบจำลองระดับแนวคิด สามารถดูรายละเอียด เพิ่มเติมใน ebook การเริ่มต้นการใช้งาน InfoSphere Data Architect

นอกจากนี้ในบทนี้ยังกล่าวถึงวิธีการจัดการแบบจำลองข้อมูล โดยการใช้กฎเพื่อตรวจสอบแบบจำลอง ER และการสร้างไดอะแกรม แม้ว่านี่จะไม่ใช่หนังสือการพัฒนาโปรแกรมประยุกต์ แต่บทนี้ให้ความเข้าใจเกี่ยวกับแนวคิดการสร้างแบบจำลอง ด้วยการใช้ เครื่องมือ InfoSphere Data Architect

3.5 แบบฝึกหัด

กำหนดชุดเอนทิตีต่อไปนี้ในบริบทมหาวิทยาลัย (university):

- Faculty
- Teacher
- Function
- Course
- Student

ให้แสดงภาพของความสัมพันธ์และชุดเอนทิตี โดยแสดงภาพทั้งหมดและไดอะแกรมความสัมพันธ์ของเอนทิตีโดยใช้เครื่องมือ InfoSphere Data Architect สำหรับข้อมูลเพิ่มเติมเกี่ยวกับเครื่องมือนี้ สามารถอ้างอิงได้จาก eBook: InfoSphere Data Architect ซึ่งเป็นส่วนหนึ่งของตำราชุดนี้

3.6 คำถามท้ายบท

1. ข้อใดต่อไปนี้ตรงกับความหมายของชุดเอนทิตี

   A. คอลัมน์ในตาราง

   B. คอลเลกชันของวัตถุจริงหรือแนวคิด

   C. ข้อมูลที่อธิบายคุณลักษณะของวัตถุหรือแนวคิด

   D. ชุดของเอนทิตีของชนิดเดียวกันที่คุณสมบัติเดียวกันที่ใช้ร่วมกัน

   E. ไม่มีข้อใดถูกต้อง

2. ข้อใดถือว่าเป็นแอตทริบิวต์ที่ไม่เสถียร

   A. มีการเปลี่ยนแปลงค่าบ่อยครั้ง

   B. มีการเปลี่ยนแปลงค่าในบางโอกาส

   C. มีค่าที่มาจากแอตทริบิวต์อื่นหรือคุณลักษณะอื่น ๆ

   D. มีค่าหลายค่าสำหรับเอนทิตีหนึ่ง

   E. ไม่มีข้อใดถูกต้อง

3. เราควรแก้ไขชุดความสัมพันธ์แบบ M:M โดยกำหนดค่าความสัมพันธ์ใหม่อย่างไร

   A. กำหนดเป็นตัวเลือก (optional) ในฝั่ง Many

   B. กำหนดเป็นข้อบังคับ (mandatory) ในฝั่ง One

C. กำหนดเป็นข้อบังคับในฝั่ง Many

D. มีความซ้ำซ้อนในฝั่ง Many

E. มีการวนซ้ำ (Recursive) ในฝั่ง One

4. ข้อใดต่อไปนี้เป็นความจริงเกี่ยวกับการสร้างแบบจำลองระดับแนวคิด

A. ชุดเอนทิตีควรเกิดขึ้นเพียงครั้งเดียวบนแบบจำลองข้อมูล

B. แบบจำลองระดับแนวคิด (Conceptual model) ควรประกอบด้วยแอตทริบิวต์ที่มีค่ามาจากแอตทริบิวต์อื่น

C. ต้องสามารถแสดงข้อมูลทั้งหมดบนแบบจำลองข้อมูล

D. ถูกต้องทั้งหมด

E. ไม่มีข้อใดถูก

5. ข้อใดต่อไปนี้ไม่ใช่คำจำกัดความของชุดเอนทิตี

A. ชุดเอนทิตีเป็นคอลเลกชันของวัตถุจริงหรือแนวคิดที่มีลักษณะเดียวกัน

B. ชุดเอนทิตีมีอินสแตนซ์ของวัตถุจริง

C. ชุดเอนทิตีเป็นสิ่งที่เกิดขึ้นจริงและแตกต่างจากเอนทิตีอื่นๆ

D. ถูกต้องทั้งหมด

E. ไม่มีข้อใดถูก

6. ข้อใดต่อไปนี้ไม่ใช่คำจำกัดความของชุดความสัมพันธ์

A. ชุดความสัมพันธ์แสดงความเกี่ยวข้องระหว่างเอนทิตี

B. ชุดความสัมพันธ์เป็นตัวกำหนดความสัมพันธ์ระหว่างชุดเอนทิตีสองชุด

C. ชุดความสัมพันธ์สามารถอ่านได้ (should be read in double sense)

D. ชุดความสัมพันธ์เป็นคำนาม

E. ทั้งหมดข้างต้น

7. ข้อใดต่อไปนี้เป็นจริงเกี่ยวกับข้อบังคับในการกำหนดค่าความสัมพันธ์

A. จะแสดงว่าเอนทิตีเกี่ยวข้องกับเอนทิตีอื่นอย่างไร

B. แต่ละชุดเอนทิตีที่ต้องเข้าร่วมในความสัมพันธ์

C. แต่ละชุดเอนทิตีอาจมีส่วนร่วมในความสัมพันธ์

D. มีการกำหนดจำนวนที่เป็นไปได้สำหรับชุดความสัมพันธ์สำหรับทุกชุดเอนทิตี

E. ไม่มีข้อใดถูกต้อง

8. กลุ่มของอินสแตนซ์ ซึ่งมีแอตทริบิวต์ หรือชุดความสัมพันธ์ที่มีอยู่เฉพาะกลุ่ม เรียกว่า

A. Constraint

B. Cardinality

C. Superset

D. Subset

E. ถูกต้องทั้งหมด

9. คำสั่งใดในมาตรฐาน SQL ที่ช่วยให้โดเมนถูกจำกัด

A. Relationship

B. Primary Key

C. Check

D. Constraint

E. ไม่มีข้อใดถูกต้อง

10. ข้อมูลในฐานข้อมูล ณ เวลาใดเวลาหนึ่งเรียกว่า

A. Intension

B. Extension

C. Schema

D. Instance

E. ไม่มีข้อใดถูกต้อง

4

บทที่ 4 – การออกแบบฐานข้อมูลเชิงสัมพันธ์

ในบทนี้เป็นการศึกษาแนวคิดของการออกแบบฐานข้อมูลเชิงสัมพันธ์ เพื่อจะช่วยในการสร้างแบบจำลองสำหรับฐานข้อมูลเชิงสัมพันธ์ โดยอ้างถึงหลักเกณฑ์ในการกำหนดตารางคอลัมน์และสร้างความสัมพันธ์ระหว่างตารางเพื่อลดความซ้ำซ้อนของข้อมูล ซึ่งในบทนี้เราจะได้เรียนรู้เกี่ยวกับ

- การสร้างแบบจำลองจากปัญหาจริงเป็นตารางที่สัมพันธ์กัน
- การระบุถึงปัญหาและการลดความซ้ำซ้อนของข้อมูลที่จัดเก็บ
- การระบุการอ้างอิง และการรวมตัวกันในการออกแบบฐานข้อมูลเชิงสัมพันธ์
- การปรับแต่งตารางในฐานข้อมูลเชิงสัมพันธ์เพื่อให้การออกแบบมีความเหมาะสมมากที่สุด

4.1 ปัญหาการซ้ำซ้อนของข้อมูล

ความซ้ำซ้อนข้อมูลหมายถึงข้อมูลประเภทเดียวกันที่มีการจัดเก็บมากกว่าหนึ่งชุดข้อมูล ซึ่งอาจจะหมายถึงการจัดเก็บภายในตารางเดียวกันของฐานข้อมูล ซึ่งการซ้ำซ้อนข้อมูลนี้เกิดจากการออกแบบฐานข้อมูลเชิงสัมพันธ์ที่ไม่เหมาะสม ทำให้เกิดความผิดปกติของข้อมูลในฐานข้อมูลดังต่อไปนี้

- ความผิดปกติจากการบันทึกข้อมูล
- ความผิดปกติจากการลบข้อมูล
- ความผิดปกติจากการปรับปรุงแก้ไขข้อมูล

ตารางที่ 4.1 แสดงให้เห็นถึงตัวอย่างของการจัดเก็บข้อมูลที่ซ้ำซ้อนของนักศึกษาในมหาวิทยาลัยซึ่งในตารางนี้ข้อมูลได้ถูกออกแบบให้จัดเก็บข้อมูลทั้งหมดในตารางเดียวกัน ภายใต้การออกแบบนี้มีการจัดเก็บข้อมูลที่ซ้ำซ้อนกัน สำหรับคอลัมน์ COLLEGE และ COLLEGE_LEVEL ตัวอย่างเช่นข้อมูล COLLEGE_LEVEL ระดับ 1 มีซ้ำสองครั้ง คือหนึ่งสำหรับนักเรียนชื่อ George Smith และอีกครั้ง สำหรับนักเรียนชื่อ Will Brown

STUDENT_ID	STUDENT	RANK	COLLEGE	COLLEGE_LEVEL
0001	Ria Sinha	6	Fergusson	4
0002	Vivek Kaul	15	PICT	5
0003	George Smith	9	IIT	1
0004	Will Brown	1	IIT	1

ตาราง 4.1 – ตัวอย่างข้อมูลซ้ำซ้อน (Student schema)

หมายเหตุ: ตาราง 4.1 จะถูกใช้อ้างเป็นตัวอย่างในบทนี้ คอลัมน์ STUDENT_ID แสดงรหัสนักศึกษาในมหาวิทยาลัย ซึ่งเป็น primary key ใน student schema

4.1.1 ความผิดปกติเนื่องของการแทรกข้อมูล

ความผิดปกติที่เกิดขึ้นเมื่อมีการแทรก Record ของข้อมูลใหม่ ซึ่งปัญหาอาจจะเกิดขึ้นในกรณีที่ฟิลด์ข้อมูลไม่สมบูรณ์ ดังตัวอย่างต่อไปนี้จะเห็นว่าการแทรกข้อมูลเกี่ยวกับนักเรียนนั้นไม่สามารถกระทำได้เนื่องจากฟิลด์ข้อมูล COLLEGE_LEVEL นั้นยังมีค่าเป็นค่าว่างอยู่

STUDENT_ID	STUDENT	RANK	COLLEGE	COLLEGE_LEVEL
0005	Susan Fuller	10	Fergusson	

ตาราง 4.2 –ตัวอย่างการแทรกข้อมูลที่ไม่ปกติ

เพราะการแทรกข้อมูลให้กับตารางระเบียนนี้อาจทำให้เกิดการซ้ำซ้อนของข้อมูลขึ้นได้ เนื่องจากในฐานข้อมูลนี้ยังมีการบันทึก COLLEGE_LEVEL แยกสำหรับนักเรียนแต่ละคนด้วย

4.1.2 ความผิดปกติเนื่องจากการลบข้อมูล

ความผิดปกติที่เกิดจากลบข้อมูลนั้นอาจจะมีผลมาจากการสูญเสียคุณสมบัติของความสัมพันธ์บางอย่างระหว่างข้อมูลของ Record ระเบียนในตาราง ตัวอย่างเช่นการลบแถว (Row) ทั้งหมดที่อ้างอิงถึงนักเรียนจาก COLLEGE ที่มีการกำหนดไว้ ซึ่งจะทำให้สูญเสียคุณสมบัติในการเชื่อมโยงระหว่าง COLLEGE และ COLLEGE_LEVEL สำหรับ COLLEGE นั้น ดังภาพประกอบในตาราง 4.3 จะเกิดการสูญเสียความสัมพันธ์ของข้อมูลที่เกี่ยวกับ COLLEGE 'IIT' ในกรณีถ้ามีการลบข้อมูลของนักเรียนทั้งสองคน คือ George Smith และ Will Brown ออกจากตาราง

STUDENT_ID	STUDENT	RANK	COLLEGE	COLLEGE_LEVEL
0003	George Smith	9	IIT	1
0004	Will Brown	1	IIT	1

ตาราง 4.3 ตัวอย่างความผิดปกติที่เกิดจากการลบข้อมูล

4.1.3 ความผิดปกติเนื่องจากการปรับปรุงข้อมูล (Update Anomalies)

เป็นความผิดปกติที่พบหลังจากที่มีการปรับปรุงข้อมูลเกิดขึ้น โดยข้อมูลของเอนทิตีเดียวกันอาจมีความ

ไม่สอดคล้องกัน และอาจจะพบความซ้ำซ้อนของข้อมูลที่อยู่ในจุดอื่นๆของตารางตัวอย่างเช่นกรณีที่ปรับปรุง COLLEGE_LEVEL = 1 สำหรับ STUDENT_ID = 0003 ซึ่งจะทำให้วิทยาลัย 'Fergusson' มีข้อมูลที่ขัดแย้งกันของ COLLEGE_LEVEL มีค่าเท่ากับ 1 และ 4

STUDENT_ID	STUDENT	RANK	COLLEGE	COLLEGE_LEVEL
0001	Ria Sinha	6	Fergusson	4
0003	George Smith	9	Fergusson	1

ตาราง 4.4 ตัวอย่างความผิดปกติในการปรับปรุงข้อมูล

ในส่วนต่อไป ขอแนะนำวิธีการแก้ปัญหาที่เกิดจากความซ้ำซ้อนของข้อมูลข้างต้น ดังต่อไปนี้

4.2. การกระจายความสัมพันธ์ (Decompositions)

การกระจายความสัมพันธ์ในการออกแบบฐานข้อมูลเชิงสัมพันธ์แสดงถึงการแบ่งโครงร่าง (Shcema) ที่เกี่ยวข้องให้มีความสัมพันธ์ที่มีขนาดเล็กลง มีความซับซ้อนที่น้อยลง เพื่อหลีกเลี่ยงความซ้ำซ้อน และสามารถสืบค้นได้อย่างถูกต้อง โดยสร้างความสัมพันธ์ที่มีขนาดเล็กสำหรับข้อมูลกลุ่มต่างๆ ตัวอย่างเช่น student schema รูป 4.1 แสดงวิธีขั้นตอนการกระจายความสัมพันธ์

STUDENT_ID	STUDENT	RANK	COLLEGE	COLLEGE_LEVEL
0001	Ria Sinha	6	Fergusson	4
0002	Vivek Kaul	15	PICT	5
0003	George Smith	9	IIT	1
0004	Will Brown	1	IIT	1

(a) Student - College Relation

COLLEGE	COLLEGE_LEVEL
Fergusson	4
PICT	5
IIT	1
IIT	1

(b) College Relation

STUDENT_ID	STUDENT	RANK	COLLEGE
0001	Ria Sinha	6	Fergusson
0002	Vivek Kaul	15	PICT
0003	George Smith	9	IIT
0004	Will Brown	1	IIT

(c) Student Relation

รูป 4.1 – ตัวอย่างการกระจายความสัมพันธ์

จากรูป 4.1 แสดงถึงรายละเอียดโครงสร้างข้อมูลนักเรียนในตาราง student ดังแสดงใน (a) และการแจงความสัมพันธ์ของตารางโดยจำแนกเป็น college และความสัมพันธ์ของ student ตามที่ปรากฏในแบบ (b) และ (c) ดังนั้น COLLEGE_LEVEL จะไม่ถูกเก็บข้อมูลซ้ำกันไว้ แต่ประกอบด้วยเพียง เรคอร์ด (Record) เดียวในสำหรับแต่ละ COLLEGE โดยที่ในตาราง COLLEGE จะแสดง student ซึ่งเป็นส่วนหนึ่งของแอตทริบิวต์ของ student relation เพื่อสามารถใช้แทนข้อมูลต้นฉบับของ COLLEGE ได้ ซึ่งสามารถกระจายความสัมพันธ์ทั้งสองได้ ดังนี้

COLLEGE	COLLEGE_LEVEL

STUDENT_ID	STUDENT	RANK

โดยความสัมพันธ์นี้จะมีผลต่อการสูญเสียคุณสมบัติของความสัมพันธ์ของข้อมูลเช่นในกรณีที่เราไม่สามารถโยงความสัมพันธ์ระหว่างข้อมูล college กับ student ได้ อีกตัวอย่างหนึ่งของการกระจายความสัมพันธ์สามารถแสดงได้ดังนี้

COLLEGE	COLLEGE_LEVEL	STUDENT_ID

STUDENT_ID	STUDENT	RANK

ซึ่งจากตัวอย่างนี้จะทำให้เกิดความซ้ำซ้อนของข้อมูลใน COLLEGE_LEVEL ของนักเรียนแต่ละคน ในขณะที่การแบ่งโครงราง (Schema) เชิงสัมพันธ์กำหนดช่วยหลีกเลี่ยงความซ้ำซ้อนของข้อมูล ข้อควรระวังคือควรจะสร้างแบบแผนเชิงสัมพันธ์ต้นฉบับย้อนกลับจากความสัมพันธ์ที่มีขนาดเล็กลงได้ โดยใช้ Functional dependencies ช่วยให้เราสามารถแปลงข้อมูลกลับได้ ซึ่งจะกล่าวในรายละเอียดต่อไป

4.3. Functional Dependencies

Functional Dependencies (FD) เป็นชนิดของความสัมพันธ์เพื่อสร้างเงื่อนไขให้เกิดความถูกต้องของข้อมูล โดยนำแนวคิดของการมี Super key มาใช้ โดยกำหนดให้มีการพึ่งพาระหว่าง subset ของแอตทริบิวต์ของความสัมพันธ์ที่กำหนดไว้ คำจำกัดความของ functional dependency ได้ให้นิยามไว้ดังนี้

นิยาม: กำหนด schema เชิงสัมพันธ์ R กับ subset ของแอตทริบิวต์ของ A และ B กล่าวได้ว่ามี functional dependency A→B ปรากฏอยู่ ถ้าค่าใน A แสดงถึงค่าที่ B สามารถอ้างถึงได้ในทุกอินสแตนซ์ r ของความสัมพันธ์ R

ดังนั้น เราจึงกล่าวได้ว่า มี FD generalizes ตามหลักของ Super Key ซูเปอร์คีย์เนื่องจาก แอตทริบิวต์ในเซต A จะเป็นตัวกำหนดค่าของแอตทริบิวต์ในเซต B สำหรับความสัมพันธ์ R ที่กำหนด นอกจากนี้ การอ้างอิงฟังก์ชัน A→B มีอยู่เมื่อค่าของ B จะขึ้นอยู่กับ A อย่างเป็นฟังก์ชัน สามารถกล่าวได้ว่าเป็น ‘functionally’ เนื่องจากใกล้เคียงกับแนวคิดของฟังก์ชัน โดย FD เชื่อมโยงความสัมพันธ์ของเซตแอตทริบิวต์ A ไปยังเซต B

การรักษา FD บนความสัมพันธ์ R ถ้ามีสำหรับทุกๆอินสแตนซ์ r ของความสัมพันธ์ R

ในขณะเดียวกัน การตรวจสอบว่า อินสแตนซ์ r ใดตรงกับการ FD เราต้องตรวจสอบว่า ทุกค่าของข้อมูลในแต่ละแถวในอินสแตนซ์ r จะตรงตาม FD ที่กำหนดเช่น A→B (A กำหนด B หรือ A สัมพันธ์กับ B สำหรับแต่ละแถวของข้อมูลที่จัดเก็บอยู่ในตารางที่มีอินสแตนซ์)

Functional Dependency หมายถึง “A กำหนด B”, “B คือขึ้นอยู่กับ A” หรือ “A สัมพันธ์กับ B” และกล่าวได้ว่า “A→B”

ถ้า $A_1 \rightarrow B_1$, $A_2 \rightarrow B_2$, $A_3 \rightarrow B_3$ ... $A_n \rightarrow B_n$, แล้วเราถึงจะเป็น

$A_1\ A_2\ A_3 \ldots A_n \rightarrow B_1\ B_2\ B_3 \ldots B_n$

ตารางที่แสดงต่อไปนี้คืออินสแตนซ์ของความสัมพันธ์ R (A, B, C, D) ที่ A→D

A	B	C	D
a1	b1	c1	d1
a2	b2	c2	d1
a3	b3	c3	d1
a4	b3	c4	d2

ตาราง 4.5 – Functional Dependency

FDs ด้านหนึ่งจากตาราง 4.5 คือ A→D เพื่อแสดงถึง D ขึ้นอยู่กับ A แต่ หากพิจารณา D→A จะไม่สามารถย้อนกลับได้ เนื่องจากกำหนดค่าเดียวกันใน D ส่งผลต่อการเปลี่ยนแปลงของ A ดูจากทูเปิล (a1, b1, c1, d1) และ (a2, b2, c2, d1)

ตัวอย่าง

การใช้ Student schema จากตัวอย่างที่แสดงในตาราง 4.7 FD, STUDENT_ID→ COLLEGE ใน Student schema เนื่องจาก STUDENT_ID นั้นไม่มีการซ้ำ ขณะที่ FD, STUDENT→COLLEGE อาจจะไม่สามารถครอบคลุมความสัมพันธ์ได้ เนื่องจากอาจจะมีกรณีที่นักศึกษาที่มีชื่อเดียวกันซ้ำกันในหลายๆ COLLEGE

STUDENT_ID	STUDENT	RANK	COLLEGE
0001	Ria Sinha	6	Fergusson
0002	Vivek Kaul	15	PICT
0003	George Smith	9	IIT
0004	Will Brown	1	IIT

ตาราง 4.7 – ตัวอย่าง Functional Dependency

เซตต่อไปนี้ของ FDs ยังคงเป็นจริง

{ STUDENT_ID → COLLEGE , STUDENT_ID → STUDENT,
STUDENT_ID → STUDENT → COLLEGE
STUDENT_ID → STUDENT → RANK }

Trivial Functional Dependencies - functional dependency ที่ยังคงเป็นจริงสำหรับทุกค่าของแอตทริบิวต์ที่กำหนด เป็นแบบ FD trivial

ตัวอย่าง

(First-name, last-name) → first-name

โดยทั่วไป functional dependency A→B ในกรณีที่ B คือ subset ของ A หมายถึง B บรรจุอยู่ใน A (ด้านขวามือจะเป็นส่วนหนึ่งของทางด้านซ้ายมือ)

ในการออกแบบฐานข้อมูลเชิงสัมพันธ์โดยทั่วไปเราสนใจ non-trivial FDs ที่ช่วยกำหนดความถูกต้องของเงื่อนไขความสัมพันธ์ในขณะที่ trivial FDs เป็นเพียงสิ่งที่ปรากฏชัดเจน และจะมีอยู่ในกรณีใดกรณีหนึ่งเท่านั้น

4.4 คุณสมบัติของ Functional Dependencies

คุณสมบัติของ functional dependencies จากเซตที่กำหนดเพื่อการอ้างอิงที่ใช้งานได้ จะกำหนดคุณสมบัติสำคัญของ FD เรียกว่า Closure Set of functional dependencies

Closure Set of Functional Dependencies - functional dependencies ทั้งหมดที่มีจากเซตที่กำหนดของ functional dependencies S จะเรียกว่า Closure Set of Function Dependency, S^+.

ดังนั้น ทุกๆอินสเตนซ์ของความสัมพันธ์ r ที่เป็นจริงจากเซตที่กำหนดของ FD, S จะยังเป็นจริงตาม closure set FD, S^+.

สำหรับหัวข้อย่อยต่อไปนี้เราจะพิจารณาจากกฎการคำนวณ closure set of functional dependencies.

4.4.1 Armstrong's Axioms

Armstrong's Axioms หรือที่เรียกว่า 'Inference Rules' เป็นกฎสำคัญที่จะช่วยให้การอ้างถึงทั้งหมดของ functional dependencies ในเซต FDs

สำหรับ Inference Rules ประกอบด้วย 3 กฎ ดังนี้

1. Reflexivity ถ้า B เป็น subset ของแอตทริบิวต์ในเซต A แล้ว A→B (โดย trivial FD)

2. Augmentation ถ้า A→B และ C เป็นแอตทริบิวต์อื่น แล้ว AC→BC การใช้ reflexivity สำหรับกฎนี้สามารถกล่าวได้อีกว่า AC→B

3. Transitivity ถ้า A→B และ B→C ดังนั้น A→C

นอกจากนี้ เรายังมีกฎเพิ่มเติมที่ได้จากการพิสูจน์จาก axioms ในข้อที่ 3 ข้างต้น ดังต่อไปนี้ เพื่อให้ง่ายต่อการสร้าง closure FD จากเซตของ FD ที่กำหนด ดังต่อไปนี้

1. Union: ถ้า A→B และ A→C แล้ว A→BC

2. Decomposition: ถ้า A→BC แล้ว →AB และ A→C

Armstrong's Axioms เป็นกฎที่สมเหตุสมผล และกฎนี้ถูกสร้างขึ้นมาเฉพาะ FDs ใน closure set ของ FD, S^+ ที่กำหนด และสร้างเซตที่สมบูรณ์ของ FDs ใน S^+.

4.4.2 การสร้าง closure set of attributes

วิธีนี้เป็นอีกวิธีการหนึ่งเพื่อสร้าง closure set ของ functional dependencies จาก FD ที่กำหนด ยังสามารถใช้เพื่อดูว่าการตั้งค่าแอตทริบิวต์ที่กำหนด คือการใช้ Super key ในการค้นหาความสัมพันธ์

สำหรับ Closure set ของแอตทริบิวต์ที่กำหนด A คือ เซตของแอตทริบิวต์ทั้งหมดในความสัมพันธ์ R ที่จะกำหนดโดย A ที่อยู่ใน FDs ที่กำหนด

หมายเหตุ: กำหนดให้ closure (A), และ A^+ เป็นเซตที่มีคุณลักษณะทั้งหมดในความสัมพันธ์ที่กำหนด R แล้วเราสามารถพูดได้ว่าแอตทริบิวต์ A เป็น Super key สำหรับความสัมพันธ์ทาง R

การคำนวณ

กำหนดความสัมพันธ์ R ด้วยเซตของแอตทริบิวต์ เราคำนวณชุด closure set ของแอตทริบิวต์สำหรับ A, closure (A) เป็นดังนี้:

1. เริ่มกำหนด closure (A) = A

2. สำหรับแต่ละ FD ถ้า A→B แล้ว เพิ่ม B closure (A), กล่าวคือ closure (A) U B

3. สำหรับเซตย่อยใดๆ ของ A, (ให้ C เป็นเซตย่อยของ A), A→C (โดย trivial FD) และ ถ้า C→D ที่

   D ไม่เป็นเซตย่อยของ A แล้วเพิ่ม D closure (A)

4. ทำซ้ำขั้นตอนที่ 3 จนกว่าจะไม่มีเซตของแอตทริบิวต์เพิ่มเติมที่จะเพิ่มไป closure (A)

ตัวอย่าง

พิจารณาความสัมพันธ์ R (A, B, C, D, E) ด้วยการกำหนด FDs A→B, B→DE และ D→C

การคำนวณ

ขั้นตอนที่ 1 Closure (A) = A

ขั้นตอนที่ 2 A→B, ดังนั้น closure (A) = A U B สามารถกล่าวได้เป็น AB

ขั้นตอนที่ 3

1st Iteration: B→DE และ B คือเซตย่อยของ closure (A), เพราะ closure (A) = ABDE

2nd Iteration: AD→C, D คือเซตย่อยของ closure (A) และ C ในไม่เป็นเซตย่อยของ closure (A), ดังนั้น closure(A), A^+ = ABDEC

ในทำนองเดียวกัน closure (B), B^+= BDEC

Closure (C), C^+ = C
Closure (D), D^+= DC
Closure (E), E^+= E

4.4.3 ความสัมพันธ์ระหว่างกลุ่ม (Entailment)

Functional Dependencies (FDs) เมื่อข้อมูลถูกบันทึกลงในฐานข้อมูล ข้อมูลจำเป็นต้องมีความสอดคล้องกับเงื่อนไขหรือข้อกำหนดที่ระบุไว้ นอกจากเงื่อนไขอื่นๆที่กำหนด ความถูกต้องของข้อมูลยังจำเป็นต้องมีความสอดคล้องกับ functional dependencies อีกด้วย
คุณสมบัติของ functional dependencies ช่วยให้ความหมายเกี่ยวกับ closure set ของ functional dependencies เพื่อให้มีสูตรโครงสร้างการสืบทอดมาเป็นเซตของข้อจำกัดได้อย่างมีประสิทธิภาพใน dependencies ในการทำงานจริง

Armstrong's Axioms ช่วยให้สามารถทำงานได้อย่างมีประสิทธิภาพเช่นความครบถ้วนสมบูรณ์ Closure ของแอตทริบิวส์ สำหรับการกำหนดเซตของ functional dependencies ไม่เพียงแต่ให้ทางเลือกวิธีการเพื่อคำนวณ closure set ของ FDs แต่ยังช่วยให้เรากำหนด Super key ของความสัมพันธ์และการตรวจสอบว่ามี functional dependency X→Y เป็น closure set ของ functional dependency

4.5 นอร์มอลฟอร์ม (Normal Forms)

กระบวนการหนึ่งที่สำคัญในการออกแบบฐานข้อมูลคือ กระบวนการ Normalization ซึ่งเป็นกระบวนการในการออกแบบฐานข้อมูลเชิงสัมพันธ์เพื่อให้มีการปรับปรุงเปลี่ยนแปลงแบบแผนเชิงสัมพันธ์ให้อยู่ในรูปแบบที่เหมาะสม คือการกำจัดความซ้ำซ้อนในความสัมพันธ์และลดปัญหาที่จะตามมา อันได้แก่ความผิดปกติอันเนื่องมาจากการเพิ่ม ลบ และปรับปรุง ข้อมูล เป็นต้น

ขั้นตอน Normal forms เพื่อการออกแบบที่ดีที่สุดในการทำ Normalization กระบวนการจะเป็นแบบ step-wise ซึ่งเป็นลำดับขั้นตอนแต่ละขั้นจะแปลงสกีมาของความสัมพันธ์เพื่อให้อยู่ในรูปของ normal form ที่สูงขึ้น แต่ละ normal form จะประกอบด้วย normal form ก่อนหน้านี้ทั้งหมดและมีการเพิ่มประสิทธิภาพโดยเพิ่มเติมบางอย่างซึ่งทำให้มีประสิทธิภาพมากกว่าเดิม

4.5.1 นอร์มอลฟอร์ม ระดับที่ 1 (First Normal Form (1NF))

ความสัมพันธ์ของรีเลชันในอันดับแรกจะถือว่าความสัมพันธ์นั้นอยู่ในรูปของ First Normal Form โดยแอตทริบิวต์ของทั้งหมด มีโดเมนที่ไม่สามารถแบ่งออกเป็นส่วนๆ ได้ หรือ การทำให้เป็นหน่วยที่เล็กที่สุด
แนวคิดของหน่วยที่เล็กที่สุดสำหรับแอตทริบิวต์ เพื่อให้แน่ใจว่า ไม่มี 'กลุ่มซ้ำ' เนื่องจากระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ คือความสามารถในการจัดเก็บค่าเดียวเท่านั้น ที่จุดตัดของแถวและคอลัมน์ กลุ่มซ้ำคือเมื่อกลุ่ม

เราพยายามที่จะเก็บค่าหลายค่าที่จุดตัดของแถว และคอลัมน์ และตารางที่จะประกอบด้วยเป็นค่าที่ซ้ำกันโดยไม่มีการจำกัดความสัมพันธ์อย่างเคร่งครัด

ข้อกำหนดเพิ่มเติม C. J. Date's [4.8], table จะอยู่ในรูป 1NF ก็ต่อเมื่อตรงกับเงื่อนไข 5 ข้อ ต่อไปนี้:

- ไม่มีการเรียงลำดับด้านบนลงล่างในแถว
- ไม่มีการเรียงลำดับจากซ้ายไปขวาในคอลัมน์
- ไม่มีแถวที่ซ้ำกัน
- ทุกจุดตัดของแถว และคอลัมน์ประกอบด้วยค่าเพียงหนึ่งค่าเท่านั้นจากโดเมนที่ใช้
- คอลัมน์ทั้งหมดเป็นปกติ (เช่นแถวไม่มีคอมโพเนนต์ที่ซ่อนอยู่เช่น row IDs , object IDs หรือ timestamps ที่ซ่อนอยู่ เป็นต้น)

คอลัมน์สำหรับจัดเก็บ "Relatives of a family" ตัวอย่างเช่น ไม่ใช่ค่าที่เล็กที่สุด ลักษณะของข้อมูลประกอบไปด้วยกลุ่มของชื่อ ในขณะที่คอลัมน์ เช่น หมายเลขพนักงาน ซึ่งไม่สามารถจะแบ่งย่อยเพิ่มเติมให้มีค่าเล็กลงไปได้อีก

ตัวอย่าง

จากตารางต่อไปนี้ ให้พิจารณาถึงความสัมพันธ์ของตาราง Movie ประกอบด้วย {Movie_Title, Year} อยู่ในรูปแบบ candidate key

Movie_Title	Year	Type	Director	Director_DOB	Yr releases cnt	Actors
Notting Hill	1999	Romantic	Roger M	05/06/1956	30	Hugh G Rhys I
Lagaan	2000	Drama	Ashutosh G	15/02/1968	50	Aamir K Gracy S

ตาราง 4.8 – ความสัมพันธ์ของตาราง Movie ที่ยังไม่อยู่ในรูปของ normal form

ความสัมพันธ์จากตารางดังกล่าวยังไม่ได้อยู่ในรูปของ 1NF และยังไม่ใช่ความสัมพันธ์อย่างเคร่งครัด เนื่องจากประกอบด้วยแอตทริบิวต์ Actors ด้วยค่าที่สามารถแจงเพิ่มเติมได้ เมื่อต้องการแปลงเป็นความสัมพันธ์ 1NF เรากระจายตารางเพิ่มเติมลงในตาราง Movie Table และ Cast Table ดังแสดงในรูป 4.2 ด้านล่าง

Movie_Title	Year	Type	Director	Director_DOB	Yr releases cnt
Notting Hill	1999	Romantic	Roger M	05/06/1956	30
Lagaan	2000	Drama	Ashutosh G	15/02/1968	50

Movie Table

Movie_Title	Year	Actors
Notting Hill	1999	Hugh G
Notting Hill	1999	Rhys I
Lagaan	2000	Aamir K
Lagaan	2000	Gracy S

Cast Table

รูป 4.2 – การแปลงเป็นรูปแบบ First Normal Form ตัวอย่างของความสัมพันธ์ใน 1NF

ในรูป 4.2 การตัดกันของแถวและคอลัมน์ในตารางแต่ละตาราง แต่ละตอนเก็บค่าที่ย่อยที่สุดที่ไม่สามารถถูกแบ่งย่อยเพิ่มเติม และดังนั้น การกระจายตัวได้สร้างความสัมพันธ์ใน 1NF สมมุติว่า ชื่อนักแสดงในคอลัมน์ Actors ไม่สามารถแบ่งเป็น ‘ชื่อ’ และ ‘นามสกุล’ ได้

4.5.2 นอร์มอลฟอร์ม ระดับที่ 2 (Second Normal Form (2NF))

ความสัมพันธ์ของตารางจะถือว่าอยู่ในรูปของ Second Normal Form เมื่ออยู่ใน 1NF และไม่มี non-key attribute ที่ขึ้นอยู่กับบางส่วนของ candidate key แต่ ขึ้นอยู่กับทั้งหมดของ candidate key จากข้อกำหนดข้างต้น ความสัมพันธ์ที่มีแอตทริบิวต์เดียวเป็น candidate key จะถือว่าอยู่ในรูป 2NF เสมอ

ตัวอย่าง

การหาความสัมพันธ์จากตาราง movie ที่อยู่ในรูปของ 1NF และพยายามแปลงให้เป็น 2NF โดยตัดความสัมพันธ์ในการอ้างอิงใดๆในบางส่วนของ candidate key ดังตัวอย่าง Yr_releases_cnt ขึ้นกับ Year กล่าวคือ Year → Yr_releases_cnt แต่คีย์ candidate key คือ {Movie_Title, Year}

ดังนั้น เพื่อให้อยู่ในรูปของ 2NF จำเป็นต้องกระจายตารางข้างต้นเพิ่มเติมเป็น Movie relation, Yearly releases relation และ Cast relation ดังแสดงในรูป 4.3.

Movie_Title	Year	Type	Director	Director_DOB
Notting Hill	1999	Romantic	Roger M	05/06/1956
Lagaan	2000	Drama	Ashutosh G	15/02/1968

(a) Movie Relation

Year	Yr releases cnt
1999	30
2000	50

(b) Year Relrease Relation

Movie_Title	Year	Actors
Notting Hill	1999	Hugh G
Notting Hill	1999	Rhys I
Lagaan	2000	Aamir K
Lagaan	2000	Gracy S

(c) Cast Relation

รูป 4.3 – การแปลงเป็น Second Normal Form

ในรูป 4.3 แต่ละ non–key แอตทริบิวต์จะขึ้นอยู่กับ candidate key ทั้งหมด ดังนั้น การกระจายตัวข้างบนนี้ สร้างความสัมพันธ์ให้อยู่ใน 2NF

4.5.3 นอร์มอลฟอร์ม ระดับที่ 3 (Third Normal Form (3NF))

ความสัมพันธ์ที่อยู่ในรูปของ Third Normal Form หมายถึงตารางที่อยู่ในรูปของ 2NF และไม่มี non-key attribute ที่ขึ้นกับ candidate transitively key ที่มีแอตทริบิวต์ทุกตัวขึ้นโดยตรงกับ primary key และไม่ผ่าน transitive relation ที่แอตทริบิวต์ Z อาจขึ้นกับ non-key attribute Y และ Y ขึ้นกับ primary key X ตามลำดับ โดยอ้างถึงกฎความสัมพันธ์แบบ Transitivity หมายถึง เมื่อ X→Y และ Y→Z แล้ว X→Z

จากความสัมพันธ์ 3NF แสดงให้เห็นว่า non-key แอตทริบิวต์จะเป็นอิสระต่อกันและกัน

ตัวอย่าง

การหาความสัมพันธ์ 2NF ของ movie ข้างต้น เราพยายามแปลงเป็น 3NF โดยการตัดการอ้างอิงใด ๆ ออก transitive dependency ของ non-prime attribute ใน primary key ในรูป 4.3, Director_DOB ขึ้นกับ Director ซึ่ง Director → Director_DOB

อย่างไรก็ตาม candidate key คือ {Movie_Title, Year } ดังนั้น {Movie_Title, Year } → Director และ Director → Director_DOB ดังนั้น เป็น transitive dependency

ดังนั้น เพื่อให้อยู่ในรูป 3NF เรากระจายตารางข้างต้นเป็น Movie relation, Director Relation, Yearly releases relation และ Cast relation ดังแสดงในรูป 4.4.

Movie_Title	Year	Type	Director	Director_DOB
Notting Hill	1999	Romantic	Roger M	05/06/1956
Lagaan	2000	Drama	Ashutosh G	15/02/1968

(a) Movie Relation

Director	Director_DOB
Roger M	05/06/1956
Ashutosh G	15/02/1968

(b) Director Relation

Year	year_release_cnt
1999	30
2000	50

(c) Movie Relation

Movie_Title	Year	Actors
Notting Hill	1999	Hugh G
Notting Hill	1999	Rhys I
Lagaan	2000	Aamir K
Lagaan	2000	Gracy S

(d) Cast Relation

รูป 4.4 – การแปลงเป็น Third Normal Form

จากรูปข้างบน แต่ละ non–key แอตทริบิวต์ เป็นอิสระแก่กันและกัน และขึ้นกับ candidate key ทั้งหมดดังนั้น จากขั้นตอนการแจงความสัมพันธ์ decomposition ดังกล่าวนี้จะเป็นการสร้างความสัมพันธ์ของตารางให้อยู่ในรูปของ 3NF

4.5.4 นอร์มอลฟอร์มแบบ Boyce-Codd Normal Form (BCNF)

ลักษณะของนอร์มอลฟอร์มแบบ Boyce-Codd Normal Form (BCNF) จะเป็น 3NF แบบเข้มงวด ที่นำไปใช้ต่อความสัมพันธ์ที่อาจมีการ overlapping ของ candidate keys ความสัมพันธ์นี้เรียกว่าเป็น Boyce-Codd normal form ถ้าเป็น 3NF และทุก non-trivial FD สำหรับความสัมพันธ์นี้มี candidate key เป็นดีเทอร์มิแนนต์ กล่าวคือ สำหรับทุก X→ Y, X คือ candidate key

ตัวอย่าง

พิจารณา Guest Lecture relation สำหรับ college ดังแสดงในตาราง 4.9ด้านล่าง สมมติว่า ครูแต่ละคนสอนเรื่องเดียวเท่านั้น

Candidate Keys: {Subject, Lecture_Day}, {Lecture_Day, Teacher}

Subject	Lecture_Day	Teacher
Graphics	Monday	Dr. Arindham Singh
Databases	Monday	Dr. Emily Jose
Java	Wednesday	Dr. Prasad
Graphics	Tuesday	Dr. Arindham Singh
Java	Thursday	Dr. George White

ตาราง 4.9 - Guest Lecture relation

จากความสัมพันธ์ที่แสดงในตาราง 4.9 แสดงให้เห็นว่าไม่มีค่าใดที่เป็นค่าที่ย่อยที่สุด ดังนั้นจะถือว่าอยู่ในรูป 1NF แอตทริบิวต์ทั้งหมดเป็นส่วนประกอบของแคนดิเดตคีย์ ดังนั้นอยู่ใน 2NF และ 3NF และมี FD ระหว่าง Teacher→Subject ปรากฏอยู่ในความสัมพันธ์ข้างต้น อย่างไรก็ตามแคนดิเดตคีย์ไม่ได้มีเพียง Teacher เท่านั้น ดังนั้น ความสัมพันธ์ข้างต้นไม่ได้อยู่ในรูป BCNF การแปลงให้เป็นความสัมพันธ์ BCNF เรา decompose เป็น ความสัมพันธ์ Subject area experts และ Lecture timetable ดังแสดงในรูป 4.5.

Subject	Teacher
Graphics	Dr.Arindham Singh
Databases	Dr. Emily Jose
Java	Dr.Prasad
Java	Dr.George White

(a) Subject area experts' relation

Subject		Teacher
Graphics	Monday	Dr.Arindham Singh
Databases	Monday	Dr. Emily Jose
Java	Wednesday	Dr.Prasad
Graphics	Tuesday	Dr.Arindham Singh
Java	Thursday	Dr.George White

(b) Lecture timetable

รูป 4.5 – การแปลงเป็น Boyce-Codd Normal Form

ตาราง 4.9 แสดงความสัมพันธ์ของการแจงตารางในระดับ BCNF สำหรับ non–trivial FD Teacher→Subject จากตารางนี้แสดงถึงว่า Teacher เป็นแคนดิเดตคีย์ ในรูป 4.5 (a) Subject area experts' relation

4.6 คุณสมบัติของ Decompositions

ในส่วนนี้จะอธิบาย Decomposition เพื่อพิจารณาคุณสมบัติของการกระจาย ซึ่งการแบ่งการกระจาย relational schema ให้มีความสัมพันธ์ที่มีขนาดเล็กลง แต่ละแอตทริบิวต์มีอยู่อย่างน้อยหนึ่งอย่างในความสัมพันธ์ใหม่ และได้ปรับปรุงประสิทธิภาพของการออกแบบมากขึ้น เพื่อมุ่งเน้นเป้าหมายการกำจัดความซ้ำซ้อนที่จะเกิดขึ้น

4.6.1 Lossless and Lossy Decompositions

การกระจายความสัมพันธ์จากรีเลชัน R ลงในความสัมพันธ์ X และ Y จะต้องคงคุณสมบัติที่เรียกว่า lossless ซึ่งหมายถึงการคงไว้ซึ่งคุณสมบัติของความสัมพันธ์ หลังจากที่มีการกระจายความสัมพันธ์โดยวิธีการ Normalization กล่าวอีกนัยหนึ่งคือความสามารถในการสร้างความสัมพันธ์แบบเดิมกลับมาจากความสัมพันธ์ที่ถูกกระจาย และไม่มีแถวของข้อมูลพิเศษถูกเพิ่มไปเป็นข้อมูลไปยังแถวที่เป็นผลลัพธ์

หลังจากนั้นถ้าผลลัพธ์ของการกระจายความสัมพันธ์ มีการสูญหายของข้อมูลจากต้นฉบับ ความสัมพันธ์ R แล้ว การกระจายความสัมพันธ์ที่เป็นแบบ lossy และแน่นอนเป็นคุณสมบัติที่ไม่ควรจะเกิดขึ้น

ให้ R มี functional dependency F ที่มีกระจายความสัมพันธ์ เป็น R_1 และ R_2 แล้ว กระจายความสัมพันธ์ เป็นแบบ lossless ถ้า FDs ต่อไปนี้เป็นจริงสำหรับ decomposed relations R_1 และ R_2 : $R_1 \rightarrow R_2 \rightarrow R_1$ หรือ $R_1 \rightarrow R_2 \rightarrow R_2$

หากมี common attributes ใน R_1 และ R_2 (เช่น$R_1 \rightarrow R_2$) อยู่ในรูป super key ใด R_1 หรือ R_2 แล้ว ความสัมพันธ์ R ต้นฉบับได้รับ lossless decomposition ลงใน R_1 และ R_2

คุณสมบัติสำคัญของการกระจายความสัมพันธ์ในฐานข้อมูลเชิงสัมพันธ์ควรเป็น lossless การ join lossless ของ decomposed relation X และ Y เป็น R ผลลัพธ์จะต้องไม่สูญเสียข้อมูลและนอกจากนั้นต้องไม่มีข้อมูลเพิ่มเติม เมื่อ join X และ Y ไปเป็นความสัมพันธ์เดิม R

ตัวอย่าง

พิจารณาจากความสัมพันธ์ของ employee เป็นดังนี้:

EMP_ID	EMP_NAME	WORK_EXP	DEPT_ID	DEPT_NAME

การ lossless decomposition สำหรับพนักงานสัมพันธ์ข้างต้นจะเป็นดังนี้:

DEPT_ID	DEPT_NAME	EMP_NAME

Department relation

EMP_ID	EMP_NAME	WORK_EXP

Employee relation

ในเบื้องต้น (Department) → (Employee) = DEPT_ID and DEPT_ID → { DEPT_ID , DEPT_NAME} กล่าวคือ DEPT_ID เป็น super key สำหรับ Department relation และมีการ decomposition แบบ lossles

การ lossy decomposition สำหรับด้านบนเป็น:

DEPT_ID	DEPT_NAME

Department relation

EMP_ID	EMP_NAME	WORK_EXP	DEPT_ID

Employee relation

ในข้างต้น (Department) → (Employee) = EMP_NAME ซึ่งไม่ใช่เป็น super key สำหรับ department หรือ employee table ดังนั้นการ decomposition เป็น lossy

จากตัวอย่างข้างต้น EMP_NAME อาจจะเกิดการซ้ำกันได้ และดังนั้นเราไม่สามารถบอกได้ว่าพนักงานทำงานในแผนกใดได้อย่างชัดเจน ซึ่งตัวอย่างนี้เป็นการแสดงให้เห็นถึงผลลัพธ์การสูญหายของข้อมูล

4.6.2 การกระจายความสัมพันธ์โดย Dependency-Preserving

การกระจายความสัมพันธ์ R ในrelationของ X และ Y คือการกระจายความสัมพันธ์โดยการอ้างอิงหลัก Dependency เมื่อ FDs ทั้งหมดที่เก็บบน X และ Y เมื่อใส่ข้อมูลตรงกันทั้งหมด FDs ที่มีอยู่ในการใช้งาน Dependency ของ F^+ ก็จะเก็บไว้ในความสัมพันธ์เดิมของ R

ในการจัดเก็บชุดคำสั่งของ FD ใน relation R, F คือชุดคำสั่งของแอตทริบิวต์ X คือชุดคำสั่งของ FDs ของฟอร์ม C->D ซึ่ง C และ D เป็นชุดคำสั่งของ X นี่คือแสดงถึงสถานะของ F_x

ดังนั้นการกระจายความสัมพันธ์ของ relation R ไป X และ Y คือการกระจายความสัมพันธ์โดยการอ้างอิงหลัก Dependency เพื่อทำการ union กับ ชุดคำสั่ง FDs ใน X และY ก็จะเท่ากับ การทำงาน Depen-

dency ของ F^+ ที่เก็บใน R กล่าวคือ F_x U F_y แสดงถึง F^+ ซึ่ง F_x คือชุดคำสั่งของ FDs ใน F^+ ที่สามารถตรวจสอบได้ในเฉพาะ X เท่านั้น และ F_y คือชุดคำสั่งของ FDs ที่สามารถตรวจสอบได้เฉพาะใน Y เท่านั้น

จากที่กล่าวข้างต้น ถ้าสามารถตรวจสอบ constraints ทั้งหมดเทียบกับตารางที่ decomposed ได้โดยที่ไม่ต้องตรวจสอบกับตารางต้นฉบับ

การกระจายความสัมพันธ์โดยการอ้างอิงหลัก Dependency อาจจะไม่ได้แสดงถึงการเชื่อมความสัมพันธ์ที่ขาดหายไปได้ ขณะที่การเชื่อมระหว่างความสัมพันธ์ของการกระจายความสัมพันธ์โดยการอ้างอิงหลัก Dependency ไม่สามารถอ้างอิงสำเร็จทุกครั้ง

4.7 ชุดคำสั่ง Minimal Cover

Minimal Cover, F_c คือชุดคำสั่งที่ dependencies กันน้อยที่สุด เช่นชุดคำสั่งที่ dependencies กันทำให้เกิด Minimal Cover และมีชุดคำสั่งของ FDs F ไปยัง relation R กล่าวคือ $F^+=F_c^+$

วัตถุประสงค์ของ Minimal Cover คือทำให้ constraint ในการตรวจสอบง่ายขึ้นเมื่อมีการป้อนข้อมูลในแบบ RDBMS ข้อมูลที่ป้อนลงในตารางจะต้องเป็นไปตาม constraint ที่มีอยู่ใน relation ที่กำหนดใน Minimal Cover ดังนั้นถ้ามีจำนวนต่ำสุดของการตรวจสอบการคำนวณมาเปรียบเทียบกับชุดเดิมของ FDs ที่มีอยู่ใน relation และ constraint ที่มี Minimal Cover

Minimal Cover F_c ที่มาจาก F เช่น:

1. RHS สำหรับแต่ละ FD ใน Minimal Cover คือ แอตทริบิวต์เดียว
2. ถ้าลดแอตทริบิวต์ใดๆจาก LHS สำหรับแต่ละ FD, จะมีการเปลี่ยนแปลงน้อยลง
3. เอา FD ใด ๆ ออกจาก Minimal Cover ทำให้เกิดการเปลี่ยนแปลงของ Minimal Cover

Minimal Cover สำหรับการกำหนดชุดของ FDs ที่ unique

แอตทริบิวต์ที่ไม่เกี่ยวข้องกัน

สำหรับแต่ละ $\alpha \rightarrow \beta$ β ที่มีอยู่ใน F คือแอตทริบิวต์ที่ไม่เกี่ยวข้องเช่นถ้าเอาแอตทริบิวต์จาก α และ β ชุดของFDs ไม่สามารถจะเปลี่ยนแปลงได้

กล่าวคือ กำจัด A จาก α เมื่อ F_c แสดงถึง $(F_c - \{\alpha \rightarrow \beta\}) \cup (\{\alpha - A\} \rightarrow \beta)$ และ

กำจัด B จาก β เมื่อ F_c แสดงถึง $(F_c - \{\alpha \rightarrow \beta\}) \cup (\alpha \rightarrow \{\beta - B\})$

เมื่อต้องการคำนวณ Minimal Cover ของF_cทำตามขั้นตอนดังนี้

1. ใช้กฎการกระจายความสัมพันธ์ ของ Armstrong's มากระจาย FD ที่มีแอตทริบิวต์เดียวใน RHS
2. ลดระดับ LHS สำหรับแต่ละ FD ใน F_cเพื่อเอาแอตทริบิวต์ที่ไม่เกี่ยวข้องใด ๆ
3. ใช้ กฎ Armstrong's เพื่อลด FDs ซ้ำซ้อนที่คงเหลือใน Minimal Cover ซึ่งไม่สามารถเปลี่ยนแปลงได้ กล่าวคือ ต้องรักษา $F^+ = F_c^+$

ตัวอย่าง

พิจารณา relation Rดังนี้: R (A, B, C, D)ด้วยการกำหนดชุดคำสั่งของ FDs F:

A→BC,

B→C,

A→B,

AB→C,

AC→D

เมื่อต้องการคำนวณ minimal cover F_c ที่มีขั้นตอนอธิบายไว้ก่อนหน้านี้ หมายเหตุการเปลี่ยนแปลงที่ถูกเน้นในรายละเอียด ดังต่อไปนี้:

ขั้นตอนที่ 1: ทำการลด FD ในชุดคำสั่ง RHS ที่ประกอบด้วยแอตทริบิวต์เดียว

(ใช้การกระจายความสัมพันธ์: ถ้า X→YZ แล้ว X→Y และ X→Z)

A→B, A→C,

B→C,

A→B,

AB→C,

AC→D

ขั้นตอนที่ 2: ลด LHS เพื่อเอาแอตทริบิวต์ที่ไม่เกี่ยวข้องที่มี ถ้ามี B→C และ AB→C ดังนั้น A จะไม่เกี่ยวข้องใน AB→C มาแทนที่ด้วย B→C ซึ่งมีอยู่แล้วในชุดคำสั่งของ FD ในทำนองเดียวกัน A→C และ AC→D และ C จะไม่เกี่ยวข้องใน AC→D มาแทนที่ด้วย →AD

ดังนั้น ชุดคำสั่งที่น้อยที่สุดคือ:

A→B, A→C,

B→C,

A→B,

B→C,

A→D

ขั้นตอนที่ 3: การลดการเก็บข้อมูลที่ซ้ำซ้อน FDs

ถ้ามีข้อมูลซ้ำ A→B→C และ B จากนั้น มีการส่งค่าผ่าน relation

A→C,

จาก นั้นจะได้ชุดคำสั่งที่น้อย:

A→B,

B→C,

A→D

Minimal Cover F_c= {A→B, B→C, A→D}

4.8 Synthesis of 3NF schemas

Synthesis of 3NF schemas เป็นการสร้างแบบจุดต่ำสุดขึ้น lossless join ร่วมกับ dependency preserving decompositions ของ relation ไปยัง 3NF การทำ Synthesis จากจุดต่ำสุดขึ้นไปเพราะว่าเราเริ่มจาก minimum set ของ FDs และสร้างการกระจายความสัมพันธ์ของ schemasโดยตรง โดยการเพิ่ม การรวมนี้และอ้างอิงรักษา decompositions ของความสัมพันธ์ลงใน 3NF การสังเคราะห์เป็นสายด้านล่างเนื่องจากเราเริ่มจากการตั้งค่าต่ำสุดของ FDs และสร้าง schema ที่ decomposed โดยตรง โดยการเพิ่ม schema ไปยัง list ดังนั้นต้องสร้างการกระจายความสัมพันธ์ใน3NF

การสร้างความสัมพันธ์โดยตรงของ schemas จาก minimal cover ในชุดคำสั่งของ FDs ใน relation R และตรวจสอบว่า schema นั้นเป็น lossless join decomposition หรือไม่ ถ้าไม่ก็ต้องเพิ่มอีก schema หนึ่งเข้าไปโดยเพิ่มเพียง candidate key ก็จะทำให้กลายเป็น lossless join decomposition ภายใน 3NF

ขั้นตอนการ Synthesis of 3NF schemas ไปยังความสัมพันธ์ R ลงใน $R_1,R_2,R_3.....,R_n$ มีดังต่อไปนี้:

1. สำหรับแต่ละ FD $\alpha \rightarrow \beta$ ใน Minimal Cover F_cถ้าไม่มีข้อมูล schemas $R_1,R_2,R_3.....,R_n$มี contains α,ในขณะ β เพิ่ม schema $R_i = (\alpha, \beta)$.
2. สำหรับแต่ละ R_i ลงในรายการ $R_1,R_2,R_3.....,R_n$ ตรวจสอบ relation ที่น้อยที่สุดของ contains a candidate key ใน R. ในขณะนั้นถ้ายังไม่เพิ่ม relation ไปยังชุดคำสั่งของการกระจายความสัมพันธ์ของ schemas R_{n+1} ดังเช่น R_{n+1}คือ candidate key ของ R

4.9 การกระจายความสัมพันธ์ในรูป 3NF decomposition

นอร์มอลฟอร์มในระดับ 3NF decomposition สามารถทำได้โดยใช้วิธีการกระจายความสัมพันธ์แบบบนลงล่างตามกระบวนการของ normalization ซึ่งการกระจายความสัมพันธ์ตามคำนิยามของ 3NF และจากนั้นให้แน่ใจว่าการอ้างอิงหลัก Dependency โดยทำการตรวจสอบบางอย่างในรีเลชันที่สร้างขึ้นดังแสดงในหัวข้อ 4.5.3 ที่ให้ภาพประกอบของวิธีการนี้

ในส่วนนี้ จะพิจารณาตัวอย่างของการกระจายความสัมพันธ์ 3NF ลงในรีเลชันสำหรับโครงสร้างสกีมาที่เกี่ยวข้องตามวิธีการแบบล่างขึ้นบนเช่นการอ้างอิงหลัก Dependency โดยใช้อัลกอริทึมดังที่อธิบายไว้ข้างต้น

ตัวอย่าง

พิจารณาโครงสร้างรีเลชัน Book (book_name, author, auth_dob, sales, rating)ดังชุดคำสั่งของ FDs:

(book_name author) → sales

author → auth_dob

sales → rating
and candidate key { book_name, author }

โดยใช้การวิเคราะห์อัลกอริทึมที่อธิบายไว้ในส่วนสุดท้าย สามารถกระจายความสัมพันธ์ Bookไปยัง schema ที่ 3NF ดังนี้:

ขั้นตอนที่ 1: minimal coverของชุดคำสั่ง FDs ข้างต้น

ขั้นตอนที่ 2: จากแต่ละ FD ในชุดคำสั่งจะได้ relation ดังต่อไปนี้

(book_name, author, sales)
(author, auth_dob)
(sales, rating)

ขั้นตอนที่ 3: ขณะที่ (book_name, author, sales) มี contains ของ candidate key แล้วไม่สามารถเพิ่ม relations ไปยังการความสัมพันธ์ของชุดคำสั่งของ schemas

ดังนั้น การกระจายความสัมพันธ์ของ 3NF คือ:

(book_name, author, sales)
(author, auth_dob)
(sales, rating)

4.10 The Fourth Normal Form (4NF)

นอร์มอลฟอร์มในระดับ fourth normal form ที่สามารถเข้าใจในเงื่อนไขของ dependencies ทั้งหลาย 4NF สำหรับ relational schema ที่กล่าวมา เมื่อไม่เกี่ยวข้องกับ Multi-valued dependencies นั้น จะมี relational ไม่เกินกว่าที่กำหนด

4.10.1 Multi-valued dependencies

แอตทริบิวต์ A multi- determines แอตทริบิวต์ B เมื่อกำหนดค่าของ A ก็คือค่าของ B ที่มีอยู่

Multi-valued Dependency (MVD) คือการแสดงถึง A→→B. ซึ่งหมายความ ว่า A multi- determines B, B คือ multi-dependent ใน A หรือ A→→B.

รีเลชันใน 4 NF เมื่อไม่มีสองสิ่งหรือมากกว่า MVDs ในrelationที่มีเช่น multi-valued แอตทริบิวต์ทั้งหลายเป็นอิสระต่อกันอย่างเป็นทางการ ของ relation R(A,B,C) ใน 4NF ถ้า MVDs ในรีเลชันที่มีเช่นนั้น A→→B และ A→→C แล้ว B และ C ไม่เป็นอิสระแก่กัน กล่าวคือ relation ทั้งหลายเกี่ยวข้องกัน

ตัวอย่าง

พิจารณา relation ของ ice cream ดังที่แสดงในตาราง4.10.

Vendor	I_Type	I_Flavour
Amul	Scoop	Vanilla
Amul	Softy	Vanilla
Amul	Scoop	Chocolate
Amul	Softy	Chocolate
Baskin Robbins	Scoop	Chocolate
Baskin Robbins	Sundae	Chocolate
Baskin Robbins	Scoop	Strawberry

Baskin Robbins	Sundae	Strawberry
Baskin Robbins	Scoop	Butterscotch
Baskin Robbins	Sundae	Butterscotch

ตาราง 4.10 ice cream relation in BCNF

สำหรับรีเลชันด้านบนใน BCNF เป็นแอตทริบิวต์ทั้งหมดเป็นส่วนหนึ่ง candidate key

The following MVDs exist

Vendor →→ I_Type

Vendor →→ I_Flavour

ดังนั้น คือผลรวมที่ดีของความซ้ำซ้อนในตารางข้างต้นที่จะนำไปสู่การ update anomalies เพราะฉะนั้นต้องแปลง relation 4NF ดังแสดงในรูป 4.6

Vendor	I_Type
Amul	Scoop
Amul	Softy
Baskin Robbins	Scoop
Baskin Robbins	Sundae

Ice cream type relation

Vendor	I_Flavour
Amul	Vanilla
Amul	Chocolate
Baskin Robbins	Chocolate
Baskin Robbins	Strawberry
Baskin Robbins	Butterscotch

Ice cream flavour relation

รูปที่ 4.6 - รีเลชันที่อยู่ในรูปของ 4NF

รีเลชันในรูป 4.6 จะกระจายความสัมพันธ์ลงใน 4NF มี MVD เป็นอิสระแก่กันไม่เกินหนึ่งในรีเลชัน ของการกระจายความสัมพันธ์

4.11 รูปแบบนอร์มอลฟอร์มอื่นๆ

นอร์มอลฟอร์มทั้งสามสิ่งที่เพิ่มขึ้นคือ fifth normal form [4.10], domain key normal form (DKNF) [4.11] และ sixth normal form [4.12, 4.13]. เป็นการประยุกต์มาจากพฤติกรรมทางธรรมชาติ เพราะฉะนั้น จะไม่มีการอธิบายต่อไป การเรียนรู้เพิ่มเติมเกี่ยวกับสิ่งเหล่านั้น สามารถทำตามการอ้างอิงที่ให้มา

4.12 กรณีศึกษาที่เกี่ยวข้องกับ Library Management System - Part 2 of 3

ในส่วนนี้ของกรณีศึกษานี้ จุดมุ่งหมายของเราคือการ พัฒนา logical model based ใน conceptual model และ logical model จะตรวจสอบ conceptual model โดยใช้เทคนิคของการทำ normalization จะทดสอบรีเลชันกับชุดคำสั่ง normal forms เป็นประโยชน์ในการลดความซ้ำซ้อน ซึ่งในการออกแบบระดับ conceptual model จะยังไม่สามารถตรวจพบความซ้ำซ้อนนี้ได้

ตารางแสดงความเกี่ยวข้องระหว่าง conceptual model และ the logical model:

Conceptual modeling concept	Logical modeling concept
Name of entity set	Relation variable, R
Entity set	Relation
Entity	Tuple
Attribute	Attribute, A1, A2, etc.
Relationship set	A pair of primary key – foreign key
Unique identifier	Primary key

ในตัวอย่าง การแปลง conceptual model เป็น logical model จะหมายถึง schema

(ส่วนขีดเส้นใต้คือ primary key):

BORROWER = {BORROWER_ID, FIRST_NAME, LAST_NAME, EMAIL, PHONE, ADDRESS, BOOK_ID, LOAN_DATE, RETURN_DATE}
AUTHOR = {AUTHOR_ID, FIRST_NAME, LAST_NAME, EMAIL, PHONE, ADDRESS)
BOOK = {BOOK_ID, TITLE, EDITION, YEAR, PRICE, ISBN, PAGES, AISLE, DECRIPTION}
COPY = {COPY_ID, STATUS}
AUTHOR_LIST = {ROLE}

เมื่อต้องการรักษาค่าความสัมพันธ์สำหรับ data integrity และหลีกเลี่ยงการสูญหายของข้อมูล data loss จำเป็นต้องใส่ foreign keys ดังนั้น ส่วนที่ขีดเส้นใต้ด้วยเส้นประคือ foreign key):

BORROWER = {BORROWER_ID, COPY_ID, FIRST_NAME, LAST_NAME, EMAIL, PHONE, ADDRESS, BOOK_ID, LOAN_DATE, RETURN_DATE}

AUTHOR = {AUTHOR_ID, FIRST_NAME, LAST_NAME, EMAIL, PHONE, ADDRESS)

BOOK = {BOOK_ID, TITLE, EDITION, YEAR, PRICE, ISBN, PAGES, AISLE, DECRIPTION}

COPY = {COPY_ID, BORROWER_ID, BOOK_ID, STATUS}

AUTHOR_LIST = {AUTHOR_ID, BOOK_ID, ROLE}

เนื่องจากกฎระบุว่า borrower จะสามารถเฉพาะยืมหนังสือได้ไม่เกินจำนวนเล่มที่กำหนดต่อวันเท่านั้น ดังนั้นรีเลชันของ BORROWER ควรมี primary key a composite key อันประกอบด้วย: {BORROWER_ID, COPY_ID, LOAN_DATE }

ต่อไปเป็นการทดสอบ relation กับ normal forms

เมื่อต้องการตรวจสอบ relation กับ first normal forms ต้องแน่ใจว่า ไม่มี 'กลุ่มการทำซ้ำ' และแอตทริบิวต์เหล่านั้นไม่สามารถแบ่งออกเป็นส่วนๆได้ ในตัวอย่างนั้นไม่ใช่แอตทริบิวต์ทั้งหมด ที่ไม่สามารถแบ่งออกเป็นส่วนๆได้ เช่นแอตทริบิวต์ของ ADDRESS ในรีเลชัน BORROWER และรีเลชัน AUTHORS คุณสามารถกระจายความสัมพันธ์นั้นลงใน {ADDRESS, CITY, COUNTRY} ในเวลาเดียวกัน คุณจะพบว่ามีการทำซ้ำภายใน BORROWER เพราะว่า borrower อาจมีการยืมหนังสือหลายเล่มเกินเวลาที่กำหนด ดังนั้นคุณจำเป็นต้องกระจายความสัมพันธ์ของ BORROWER รีเลชันใน BORROWER และ LOAN relations ดังนี้:

BORROWER = {BORROWER_ID, FIRST_NAME, LAST_NAME, EMAIL, PHONE, ADDRESS}

LOAN = {BORROWER_ID, COPY_ID, LOAN_DATE, RETURN_DATE}

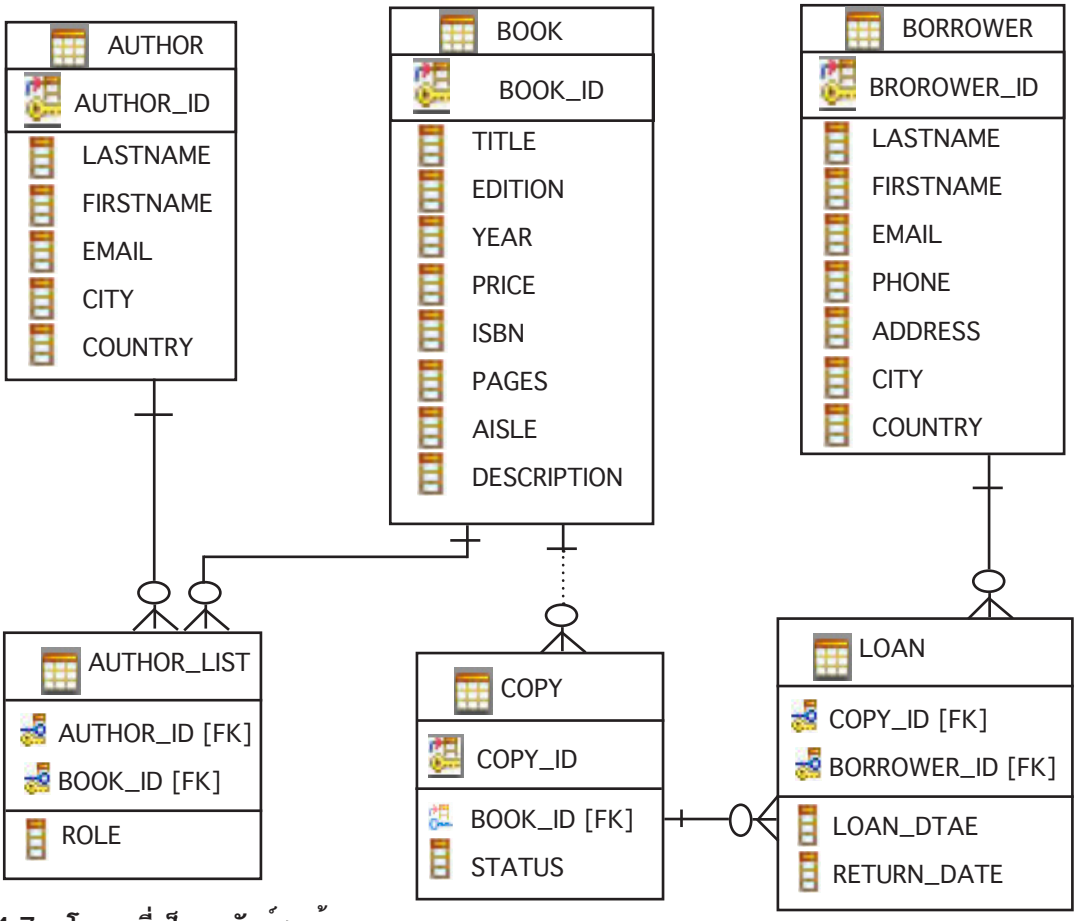
เมื่อสิ้นสุดของกระบวนการนี้ สามารถสรุปได้ว่ารีเลชันนี้อยู่ใน first normal form

ขณะนี้ มาลองทดสอบเปรียบเทียบกับ 2nd normal form กฎของ 2nd normal form บอกว่า แอตทริบิวต์ทั้งหมดของรีเลชันนั้นขึ้นกับคีย์ทั้งหมดที่ไม่อยู่ในส่วนใดส่วนหนึ่ง ในตัวอย่างของเราทุกรีเลชันมี primary key ประกอบด้วยแอตทริบิวต์เดียว ได้โดยอัตโนมัติใน 2nd normal form ดังนั้นคุณจำเป็นต้องทดสอบรีเลชันเพียงอย่างเดียวที่มี composite key ดังนั้น {LOAN_DATE, RETURN_DATE} ขึ้นอยู่กับทั้ง BORROWER_ID และ COPY_ID และบทบาทของ AUTHOR_ID และ BOOK_ID นั้นคือรีเลชันทั้งหมดที่อยู่ใน 2nd normal form

สำหรับ 3rd normal form ถ้าจำเป็นต้องทดสอบ 3rd normal form หากมีแอตทริบิวต์ที่ไม่มีคีย์หลักซึ่งขึ้นกับแอตทริบิวต์ที่ไม่มีคีย์อื่นๆ ซึ่งจากตัวอย่างที่แสดงนั้นไม่มีสถานการณ์ในลักษณะนั้น ดังนั้นไม่มีเหตุผลต่อการเพิ่มการกระจายความสัมพันธ์ของรีเลชันทั้งหมดที่อยู่ใน 3rd normal form

ขณะนี้ สามารถดูที่รูปแบบแนวคิดที่สอง และทำการเปลี่ยนแปลงที่เหมาะสม: สร้างชุดคำสั่ง Loan entity สร้างความสัมพันธ์แล้วดู foreign keys และจัดเรียงชุดคำสั่งทั้งหมดของentity แล้วก็มีเครื่องมือมีประโยชน์เช่นเดียวกับ InfoSphere Data Architect (IDA) จะช่วยสำหรับขั้นตอนนี้

ภาพด้านล่างสร้างขึ้น ด้วย IDA แสดง final logical model:



รูปที่ 4.7 – โมเดลที่เป็นผลลัพธ์สุดท้าย

อีกวิธีหนึ่งเพื่อสร้าง final logical model คือเริ่มต้นจากจุดเริ่มต้นกับ final logical model ที่รวมรีเลชันและแอตทริบิวต์ทั้งหมดที่พบในส่วนที่ 1 ของกรณีศึกษาในระหว่างการวิเคราะห์ conceptual model การปรับปรุงรูปแบบ โดยการระบุ specifying primary keys และ proceed ไปยัง normalize model สุดท้าย ควรจะเข้าถึง final logical model

ส่วนที่ 3 ของกรณีศึกษานี้ยังคงดำเนินต่อไปในบทที่ 5 ใน ส่วนที่ 3 นี้ได้อธิบายวิธีที่สามารถเปลี่ยน logical model ลงใน physical model

4.13 สรุป

ในบทนี้อธิบายถึง model ของ real-world objects เริ่มจากโดเมนทางธุรกิจที่มีการออกแบบอย่างเหมาะสมในตารางต่างๆ ภายใน relational database ซึ่งมีคุณสมบัติในการเชื่อมโยงออบเจ็กต์เหล่านี้ให้เป็นของคอลัมน์และกล่าวถึงวิธีการสร้าง relate tables เพื่อสร้างแบบจำลองความสัมพันธ์ที่ดีที่สุดเพื่อใช้ในสถานการณ์จริง

การจัดเก็บข้อมูลที่ซ้ำซ้อนสำหรับในตารางที่เกี่ยวข้องนั้น relational database ทำให้เกิดความผิดปกติเช่นการเพิ่ม ลบและ แก้ไข ข้อมูล ดังนั้น การพยายามปรับปรุง relational schema เพื่อทำให้เกิดความซ้ำซ้อนน้อยที่สุด

การทบทวน normal forms สำหรับ relational tables และ กระบวนการของ normalization ไปยังการออกแบบ relational database ที่เหมาะสมที่สุด แต่ละ normal form ที่สูงขึ้นไป คือ normal form ที่ละเอียดกว่า normal form ก่อนหน้านี้ การกระจายความสัมพันธ์ของรีเลชันเป็นวิธีการทำ higher normal forms

Functional ของ การกระจายความสัมพันธ์ระหว่างแอตทริบิวต์ของ table guide จะเป็นวิธีการที่ดีที่สุดในการกระจายความสัมพันธ์ของตาราง

คุณสมบัติของ function dependencies: Armstrong's Axioms และ Closure set ของแอตทริบิวต์ช่วยในการทำงานได้อย่างมีประสิทธิภาพมากที่สุดผ่านชุดคำสั่งของ functional dependencies ไปยังการตรวจสอบที่เร็วที่สุด เราพยายามกระจายความสัมพันธ์ ของ relations เช่น คุณสมบัติของการกระจายความสัมพันธ์ - lossless เข้ากับอ้างอิงหลักของ dependency ไว้ในการออกแบบฐานข้อมูล

หลังจากการทำความเข้าใจกับบทนี้ สามารถทำการออกแบบ relational database เพื่อให้เป็นแบบจำลองภายในฐานข้อมูล และทำการเปลี่ยนแปลงให้เหมาะสมเพื่อการเก็บข้อมูลที่มีประสิทธิภาพมากที่สุด โดยเกิดความซ้ำซ้อนของข้อมูลน้อยที่สุด และยังอำนวยความสะดวกในการเรียกดูข้อมูล

4.14 แบบฝึกหัด

1. คำนวณ Closure ของชุดแอตทริบิวต์ต่อไปนี้ สำหรับแต่ละแอตทริบิวต์จากรีเลชันที่กำหนด R(A,B,C,D,E) สำหรับการกำหนดชุดของ FDs {AB→C, A→DE, B→D, A→B, E→C} และให้ทำการค้นหา super key จากรีเลชันนี้

2. ให้ทำการหาว่ารีเลชันต่อไปนี้จัดอยู่ในรูปของนอร์มอลฟอร์มที่เท่าไร

   Order (product_id, customer_id , price, quantity, order_type)
   โดยที่ (product_id, customer_id) เป็นแคนดิเดตคีย์ และ order_type ถูก กำหนดให้เป็น
   'luxury' ในกรณีที่ price > 1000$ และเป็น 'regular' ถ้า price <1000$
   รวมถึงให้ทำการแปลงให้อยู่ในรูปของนอร์มอลฟอร์มที่ 3 (3NF)

3. คำนวณ minimal cover ของrelation R (A, B, C, D) จากชุดคำสั่งของ FDs

   AB→C, B→C, A→CD

4. สำหรับรีเลชัน Library Book (Book_id, bookname, author, subject)การสังเคราะห์รีเลชันที่อยู่ 3NF นอร์มอลฟอร์มที่ยังคงคุณสมบัติของความสัมพันธ์เอาไว้

5. คำนวณชุด Closure ของ Functional Dependencies F+ สำหรับรีเลชันที่กำหนด R(A,B,C,D,E) มีการกำหนดชุดคำสั่งของ FDs {AB→C, A→DE, B→D, A→B, E→C}

4.15 คำถามทบทวนความจำ

1. ให้แสดงการกระจายความสัมพันธ์ต่อไปนี้เพื่อให้เห็นถึงคุณสมบัติของ lossless-join สำหรับ wedding organizer:

   Order (customer_id, bride, groom, budget)
   Wedding Category (budget, wedding_type)

2. กำหนดรีเลชันสำหรับ Cell phone ต่อไปนี้:

mobile	brand	head_office
N93	Nokia	New Delhi
Diamond	HTC	Hyderabad
N97	Nokia	New Delhi
MotoSlim	Motorola	Mumbai
3315	Nokia	New Delhi
ZN50	Motorola	Mumbai

Cell phone relation

การ update statement สำหรับรีเลชันที่กำหนดในข้อใดต่อไปนี้ จะส่งผลให้การ update anomaly ผิดพลาด

A. UPDATE cellphone set mobile = '6600' where mobile = 'N97'

B. UPDATE cellphone set brand = 'Samsung' where mobile = 'ZN50'

C. UPDATE cellphone set head_office = 'Bangalore' where mobile = 'N97'

D. UPDATE cellphone set brand = 'Samsung' , mobile = 'X210' where mobile = 'MotoSlim'

E. ไม่มีข้อใดถูก

3. ระบุ normal form สำหรับ library - book relation ด้านล่างนี้:
Library Book (Book_id, bookname, author, subject)

A. First Normal Form

B. Second Normal Form

C. Third Normal Form

D. Boyce - Codd Normal Form

E. Fourth Normal Form

4. ข้อใดต่อไปนี้ไม่สามารถกำหนดจากการอ้างอิงคุณสมบัติของ Functional

A. Closure set of functional dependencies, F+

B. Decomposition is lossless

C. $X \rightarrow Y$ belongs to F+

D. Super key

E. ไม่มีข้อใดถูก

5. สำหรับรีเลชันในระดับ BCNF เมื่อมีการอ้างอิง (MVDs) อยู่หลาย Multi-valued จะถือว่าอยู่ในรูป 4NF อย่างไร

   A. MVDs เป็นอิสระต่อกัน

   B. MVDs เป็นรีเลชันที่เกี่ยวข้องกัน

   C. Candidate key กำหนด MVDs

   D. Candidate keys จะซ้อนทับกัน

   E. ไม่มีข้อใดถูก

6. ข้อใดต่อไปนี้เป็นประโยคที่ไม่ถูกต้อง

   A. Lossless เชื่อมกับการกระจายความสัมพันธ์ สามารถเชื่อมได้ตลอดเวลา

   B. การอ้างอิง Functional คือชนิดของ integrity constraints

   C. Dependency preserving implies lossless join and vice-versa

   D. BCNF ไม่ประสบความสำเร็จเสมอ

   E. ไม่มีข้อใดถูก

5

บทที่ 5 - ภาษา SQL เบื้องต้น

ภาษา Structured Query Language (SQL) หรือเรียกว่า เอส-คิว-แอล เป็นภาษาพื้นฐานที่ช่วยให้ผู้ใช้สามารถจัดการกับข้อมูลในฐานข้อมูลเชิงสัมพันธ์ ภาษา SQL มีจุดแข็งอย่างหนึ่งคือให้ผู้ใช้ระบุข้อมูลที่สนใจหรือต้องการเรียกใช้ โดยไม่จำเป็นต้องเข้าใจหรือรู้ถึงวิธีการที่จะเรียกใช้ข้อมูล ซึ่งระบบการจัดการฐานข้อมูลจะทำหน้าที่ในการหากรรมวิธีในการเข้าถึงข้อมูลเหล่านี้ โดยผ่านทางการใช้คำสั่ง SQL นี้ให้เอง คำสั่ง SQL นั้นจะทำงานกับข้อมูลที่เป็นตาราง โดยสามารถทำงานได้กับตารางอย่างน้อยหนึ่งตารางหรือมากกว่าเป็นต้นไป

คำสั่ง SQL สามารถแบ่งเป็น 3 กลุ่ม ตามประเภทฟังก์ชันของการใช้งานดังต่อไปนี้:

- DDL (Data Definition Language) ประกอบด้วยกลุ่มคำสั่งที่เกี่ยวข้องกับการนิยามข้อมูล ใช้ในการกำหนด เปลี่ยนแปลงโครงสร้าง หรือลบ ออบเจ็กต์ใดๆในฐานข้อมูล
- DML (Data Manipulation Language) ประกอบด้วยกลุ่มคำสั่งที่เกี่ยวข้องกับการเรียกใช้ ปรับเปลี่ยน และลบข้อมูลในฐานข้อมูล
- DCL (Data Control Language) ประกอบด้วยกลุ่มคำสั่งที่ใช้สำหรับการควบคุมสิทธิ์การใช้งาน และการเข้าถึงฐานข้อมูล

ในบทนี้เราจะเรียนรู้ถึงประวัติของ SQL และวิธีการทำงานกับภาษาที่มีประสิทธิภาพสูง โดยมุ่งเน้นพื้นฐานการทำงานของ SQL ที่มักจะมีการใช้งานสำหรับโปรแกรมประยุกต์ นั่นคือ การสร้าง (Create) การอ่าน (Read) การปรับปรุง (Update) และการลบ (Delete) ซึ่งจะเรียกโดยรวมว่า CRUD

5.1 ประวัติของ SQL

ในปี 1970's Don Chamberlin และ Ray Boyce จากบริษัทไอบีเอ็ม ได้พัฒนาภาษา SQL ซึ่งเป็นส่วนหนึ่งของโครงการพัฒนาระบบ System R โครงการนี้ได้แสดงความมุ่งมั่นที่จะนำแบบจำลองเชิงสัมพันธ์ตามแนวคิดของ Codd เพื่อนำไปใช้ในเชิงปฏิบัติจริง

แต่เดิมภาษานี้ถูกตั้งชื่อว่า "Structured English Query Language" หรือ SEQUEL แต่ภายหลังถูก

เปลี่ยนเป็น SQL เนื่องจากว่า SEQUEL เป็นชื่อที่ถูกจดทะเบียนทางการค้าไว้แล้วโดยบริษัทแห่งหนึ่งในประเทศสหราชอาณาจักร (United Kingdom)

ปัจจุบันภาษา SQL ได้รับการยอมรับให้เป็นภาษามาตรฐานสำหรับฐานข้อมูลเชิงสัมพันธ์ โดยถูกประกาศให้ใช้เป็นภาษามาตรฐานในปี 1986 โดย American National Standards Institute (ANSI) และโดย International Standards Organization (ISO) ในปี 1987 และภาษา SQL ได้ถูกปรับปรุงมาแล้ว 6 ครั้ง นับตั้งแต่ภาษานี้ได้รับการยอมรับให้เป็นภาษามาตรฐาน โดยมีการปรับปรุงครั้งล่าสุดในปี ค.ศ. 2008 และเรามักจะเรียกตามชื่อและปีที่ได้รับการยอมรับความเป็นมาตราฐาน เรียกว่า SQL:2008

ภาษา SQL เป็นภาษาที่มีโครงสร้างที่คล้ายอังกฤษ โดยมีลักษณะการทำงานที่เหมาะสมสำหรับฐานข้อมูล มีรูปแบบคำสั่งและคำสั่งเฉพาะต่างๆที่ง่ายต่อการใช้งานและการเข้าใจ รวมถึงมีกลไกที่สนับสนุนการออกแบบและการใช้งานตามความต้องการของผู้ใช้ ซึ่งเราจะอธิบายวิธีการและกลไกเหล่านี้ต่อไป

5.2 กำหนดโครงสร้างฐานข้อมูลเชิงสัมพันธ์ใน SQL

ตามที่กล่าวไว้ในบทต่างๆก่อนหน้า โครงสร้างฐานข้อมูลเชิงสัมพันธ์คือการอธิบายอย่างมีรูปแบบของข้อมูลทั้งหมด รวมทั้งความสัมพันธ์ฐานระหว่างข้อมูลทั้งหมดด้วย คุณสามารถสร้างโครงสร้างทางกายภาพนี้ (หรือที่เรียกว่า Physical Data Model หรือแบบจำลองข้อมูลเชิงกายภาพ) ด้วยภาษา SQL อย่างไรก็ตามผู้จัดจำหน่ายฐานข้อมูลส่วนใหญ่จะสนับสนุนมาตรฐาน ANSI และ ISO SQL ซึ่งไวยากรณ์ภาษาของแต่ละค่ายนั้นอาจจะมีไวยากรณ์ที่แตกต่างกันเพียงเล็กน้อย แต่แบบจำลองข้อมูลเชิงกายภาพจะเป็นแบบเฉพาะแต่ละผลิตภัณฑ์ฐานข้อมูล ในหนังสือเล่มนี้เราใช้ฐานข้อมูล DB2 Express-C ของ ไอบีเอ็ม ซึ่งเป็นเวอร์ชันหนึ่งของ IBM DB2 Server ที่แจกจ่ายให้ใช้ฟรี

สำหรับองค์ประกอบและคุณสมบัติต่าง ๆ ที่เป็นส่วนหนึ่งของแบบจำลองข้อมูลเชิงกายภาพ ซึ่งเราจะกล่าวในรายละเอียดเพิ่มเติมต่อไป และรวมทั้งวิธีการนำไปใช้งานโดยใช้คำสั่ง SQL

5.2.1 ประเภทของข้อมูล (Data Types)

ในลักษณะเดียวกับภาษาที่ใช้สำหรับการพัฒนาโปรแกรม ในระบบจัดการฐานข้อมูลมีการสนับสนุนชุดของประเภทของข้อมูล(data type)จำกัดชุดหนึ่ง ซึ่งสามารถใช้เพื่อกำหนดประเภทของข้อมูล(data type)ของข้อมูลที่จัดเก็บในแต่ละคอลัมน์ ประเภทของข้อมูล(data type)ของข้อมูลพื้นฐาน ตัวอย่างเช่น เลขจำนวนเต็ม (integer) เลขทศนิยม (float) อักขระ (char) วันที่ (date) เวลา (time) บล็อบ (blob) และอื่น ๆ

นอกจากนี้ ในระบบจัดการฐานข้อมูลเรายังสามารถเลือกที่จะสร้างประเภทของข้อมูล(data type)ที่ผู้ใช้กำหนดเองเรียกว่า User-Defined Data Type หรือ UDT อย่างไรก็ตามข้อกำหนดของประเภทของข้อมูลที่ผู้ใช้กำหนดเองนี้ ยังขึ้นอยู่กับประเภทของข้อมูลพื้นฐานที่สนับสนุนโดยระบบจัดการฐานข้อมูล ตัวอย่าง UDT เช่น address, country, phone number, social security number, postal zip code

5.2.1.1 วันและเวลา (Date and Time)

ปัจจุบันฐานข้อมูลส่วนใหญ่จะสนับสนุน วันและเวลา (date and time) ที่เป็นประเภทของข้อมูล และฟังก์ชันที่ใช้สำหรับทำงานร่วมกันกับประเภทของข้อมูลเหล่านี้ ซึ่งโปรแกรม DB2 สนับสนุนประเภทของข้อมูลดังกล่าว ดังต่อไปนี้

- Date (YYYY-MM-DD)

- Time (HH:MM:SS)
- Timestamp (YYYY-MM-DD-HH:MM:SS:ssssss)

โดยที่ ssssss จะหมายถึงการประทับเวลาในระดับเสี้ยววินาที (microseconds) และต่อไปนี้จะเป็นฟังก์ชันชุดหนึ่งที่ใช้สำหรับประเภทของข้อมูล วันและเวลา

- Year
- Month
- Day
- Dayname
- Hour
- Minute
- Second
- Microsecond

5.2.2 การสร้างตาราง (Create Table)

ตาราง (Table) ประกอบไปด้วยชุดข้อมูลที่มีการจัดการและจัดเก็บไว้ในแถว (Rows) และคอลัมน์ (Column) ตัวอย่างตารางที่ใช้สำหรับเก็บข้อมูลเช่น รายการนักศึกษา อาจารย์ วิชา หนังสือฯลฯ

จากการสร้างแบบจำลองข้อมูล เอนทิตี (Entities) จะถูกนำมาสร้างเป็นตารางในฐานข้อมูล และแอตทริบิวต์ของเอนทิตีก็จะถูกนำไปสร้างเป็นคอลัมน์ต่างๆ ภายในตาราง ตัวอย่างของคำสั่ง SQL สำหรับสร้างตารางอย่างง่าย ๆ

```
create table myTable (col1 integer)
```

คำสั่งข้างต้นจะทำการสร้างตารางชื่อ myTable ประกอบด้วยคอลัมน์หนึ่งชื่อ col1 ที่สามารถเก็บข้อมูลที่มีประเภทของข้อมูล เป็นจำนวนเต็ม คอลัมน์นี้จะสามารถรับค่าจำนวนเต็มใดๆ หรือยอมรับค่าที่เป็นค่าว่าง หรือค่า NULL เป็นค่าถูกต้อง สำหรับค่า NULL จะอธิบายหัวข้อนี้ในภายหลัง

5.2.2.1 ค่าเริ่มต้น (Default Values)

เมื่อข้อมูลถูกเพิ่มลงในตาราง เราอาจจะต้องการสร้างค่าเริ่มต้นสำหรับบางคอลัมน์โดยอัตโนมัติ ตัวอย่างเช่น เมื่อผู้ใช้เว็บไซต์ลงทะเบียนในเว็บไซต์ ถ้าผู้ใช้เว็บไซต์ไม่ได้ใส่ค่าในฟิลด์ที่เก็บข้อมูลอาชีพ เราอาจจะไม่ต้องการให้ฟิลด์นี้ มีค่าเป็นค่าว่าง ดังนั้นในตาราง Users เราจะต้องกำหนดค่าเริ่มต้น (ค่า Default) ให้เป็น ‘Student’ ด้วยคำสั่งต่อไปนี้

```
CREATE TABLE USERS
(NAME      CHAR(20),
 AGE       INTEGER,
 PROFESSION VARCHAR(30) with default ‘Student’)
```

เมื่อต้องการกำหนดค่าคอลัมน์หมายเลขแผนก (DEPTNO) เป็นค่าหมายเลขแผนกล่าสุด เราสามารถ

กำหนดในคำสั่งได้ดังต่อไปนี้

```
CREATE TABLE DEPT
  (DEPTNO     SMALLINT      NOT NULL
              GENERATED ALWAYS AS IDENTITY
               (START WITH 500, INCREMENT BY 1),
   DEPTNAME   VARCHAR(36)   NOT NULL,
   MGRNO      CHAR(6),
   ADMRDEPT   SMALLINT      NOT NULL,
   LOCATION   CHAR(30))
```

คำสั่ง SQL ข้างต้นสร้างตาราง DEPT ที่กำหนดค่าเริ่มต้นให้แก่คอลัมน์ DEPTNO โดยใช้คำสั่ง GENERATED ALWAYS AS IDENTITY โดยมีค่าเริ่มต้นจาก 500 และเพิ่มขึ้นให้ทีละ 1 อัตโนมัติเมื่อมีการเพิ่มแถวในตารางนี้ โดยที่เราไม่ต้องใส่ค่าให้กับคอลัมน์ DEPTNO ซึ่งเป็นการป้องกันมิให้คอลัมน์นี้เป็นค่าว่าง

5.2.2.2 ค่าว่าง (NULL Values)

ค่า NULL Values หรือค่าว่าง เป็นการแสดงถึงสถานะของข้อมูลที่เป็น Unknown คือไม่สามารถระบุได้ว่าเป็นค่าอะไรหรือไม่มีการใส่ค่าให้แก่คอลัมน์นั้น ตัวอย่างเช่น ตารางที่มีการเก็บข้อมูลคอลัมน์คะแนนของนักเรียน สามารถมีค่าเป็นค่าว่าง NULL ได้ ทั้งนี้การยอมให้มีการจัดเก็บค่าว่างนี้ อาจจะหมายความว่าในความเป็นจริง นักเรียนยังไม่ได้ส่งการบ้าน หรือยังไม่ได้ทำข้อสอบ ซึ่งยังไม่สามารถระบุคะแนนได้ ซึ่งจะแตกต่างจากค่าคะแนนที่เป็นศูนย์ในกรณีที่นักเรียนทำข้อสอบผิดทุกข้อ อย่างไรก็ตาม บางครั้งเราอาจจะมีสถานการณ์ที่ไม่ต้องการให้คอลัมน์มีข้อมูลเป็นค่าว่าง ตัวอย่างเช่น กรณีที่ไม่ต้องการให้ฟิลด์ข้อมูลสำหรับประเทศ (country) ในใบรับสมัครเป็นค่าว่าง เราสามารถระบุคำสั่ง NOT NULL สำหรับฟิลด์ข้อมูลนั้นๆเมื่อมีการสร้างตาราง อย่างไรก็ตาม ฟิลด์ข้อมูลสำหรับ country นี้ยอมให้มีการซ้ำกันของข้อมูลเกิดขึ้นได้

```
create table myTable (name varchar(30), country varchar(20) NOT NULL)
```

5.2.2.3 ข้อกำหนด (Constraints)

ในฐานข้อมูลเชิงสัมพันธ์เราสามารถระบุข้อกำหนด (Constraints) เป็นการกำหนดเงื่อนไขซึ่งเป็นกฎที่ใช้บังคับในตารางฐานข้อมูล โดยแต่ละตารางอาจจะมีข้อกำหนดที่แตกต่างกัน ข้อกำหนดมีหลายประเภท ดังรายละเอียดต่อไปนี้

- A UNIQUE เป็นข้อกำหนดที่ใช้สำหรับป้องกันความซ้ำกันของข้อมูลในตาราง โดย UNIQUE สามารถกำหนดโดยสร้างเป็นดัชนีที่ป้องกันการซ้ำกัน (unique indexes) หรือจะระบุคีย์เวิร์ด UNIQUE ในคำสั่งสร้างตาราง ซึ่งค่า NULL เป็นส่วนหนึ่งของค่าที่ไม่ซ้ำกันในโดเมน UNIQUE

- A PRIMARY KEY เป็นข้อกำหนดสำหรับคีย์หลัก ที่คล้ายคลึงกับข้อกำหนดของการไม่ซ้ำกัน อย่างไรก็ตามสำหรับคีย์หลักนี้จะทำให้ข้อมูลมีคุณสมบัติทั้งสองอย่างคือไม่ซ้ำ UNIQUE และ ข้อมูลจะต้องไม่เป็นค่าว่าง ข้อกำหนดคีย์หลัก จะมีดัชนีที่ป้องกันการซ้ำกัน (unique indexes) ให้ร่วมด้วย

- A REFERENTIAL เป็นข้อกำหนดที่ใช้สำหรับสนับสนุนแนวคิดกฎความสัมพันธ์ referential integrity ซึ่งจะช่วยให้สามารถจัดการความสัมพันธ์ระหว่างตาราง ซึ่งในรายละเอียดจะกล่าวเพิ่มเติมในส่วนถัดไป

- A CHECK เป็นข้อกำหนดที่ใช้สำหรับการตรวจสอบว่าข้อมูลที่บันทึกลงในตารางเป็นไปตาม

เงื่อนไขที่ระบุไว้ในข้อกำหนด CHECK

ตัวอย่างต่อไปนี้แสดงถึงข้อกำหนดของตารางที่มีการตรวจสอบเงื่อนไขการบันทึกข้อมูลโดยใช้ข้อกำหนด CHECK ในหลายรูปแบบ และแสดงถึงข้อกำหนดคีย์หลัก PRIMARY KEY

```
CREATE TABLE EMPLOYEE
    (ID             INTEGER     NOT NULL  PRIMARY KEY,
     NAME           VARCHAR(9),
     DEPT           SMALLINT    CHECK (DEPT BETWEEN 10 AND 100),
     JOB            CHAR(5)     CHECK (JOB IN ('Sales','Mgr','Clerk')),
     HIREDATE       DATE,
     SALARY         DECIMAL(7,2),
     CONSTRAINT  YEARSAL  CHECK ( YEAR(HIREDATE) > 1986
                                  OR SALARY > 40500 )
    )
```

สำหรับตัวอย่างที่แสดงในตารางนี้ ประกอบด้วย 4 ข้อกำหนด (constrains) ที่จะใช้สำหรับตรวจสอบข้อมูลตามเงื่อนไข ก่อนที่ข้อมูลนั้นๆจะสามารถบันทึกลงในตารางได้ ข้อกำหนดเหล่านี้ประกอบด้วย

- PRIMARY KEY การกำหนดคีย์หลักสำหรับคอลัมน์ ID เพื่อกำหนดว่าคอลัมน์นี้จะต้องไม่มีค่าซ้ำกันหรือค่าว่าง nulls
- CHECK เป็นการกำหนดเงื่อนไข DEPT BETWEEN 10 AND 100 เพื่อตรวจสอบว่าคอลัมน์ DEPT จะต้องมีค่าอยู่ระหว่าง 10 ถึง 100 เท่านั้น
- CHECK เป็นการกำหนดเงื่อนไข JOB IN ('Sales','Mgr','Clerk') ข้อมูลอาชีพ JOB นั้นจะต้องเป็นค่าที่อนุญาตหรือระบุไว้ในลิสต์ คือ Sales, Mgr หรือ Clerk เท่านั้น
- CHECK เป็นการตรวจสอบเงื่อนไข YEAR(HIREDATE) > 1986 OR SALARY > 40500 หมายถึงข้อมูลที่จะสามารถบันทึกลงในตาราง คอลัมน์วันที่เริ่มจ้าง HIREDATE จะต้องมีค่าปีมากกว่า 1986 หรือ เงินเดือน SALARY จะต้องมีค่ามากกว่า 40500 เท่านั้น

5.2.2.4 ความสัมพันธ์ (Referential Integrity)

ตามที่ได้กล่าวไว้ในบทที่ 2 ฐานข้อมูลเชิงสัมพันธ์มีคุณลักษณะหนึ่งที่สำคัญคือกฎความสัมพันธ์ referential integrity ที่ใช้สำหรับสร้างความสัมพันธ์ระหว่างตาราง ซึ่งเป็นการสร้างความเชื่อมโยงกันระหว่างคีย์หลัก(Primary Key)และคีย์นอก (Foreign Key) เพื่อเป็นการบังคับให้ข้อมูลที่จัดเก็บในตารางต่างๆนั้นมีความสัมพันธ์กัน ซึ่งนั่นก็ย่อมหมายถึงการเพิ่มประสิทธิภาพในเรื่องความถูกต้องของข้อมูลด้วย การสร้าง referential integrity จะช่วยลดความซับซ้อนของการพัฒนาโปรแกรมประยุกต์โดยลดการตรวจสอบความถูกต้องของข้อมูลในระดับของโปรแกรมประยุกต์

สำหรับตารางที่มีคอลัมน์ของข้อมูลที่มีค่าขึ้นอยู่กับ ค่าข้อมูลของตารางอื่น จะเรียกว่าตารางพึ่งพา (dependant) หรือเราเรียกตารางนี้ว่าเป็นตารางลูก (child table) และตารางที่จะถูกอ้างอิงถึงเรียกว่า ตารางพื้นฐาน (base) หรือตารางแม่ (parent table) ซึ่งตารางที่มีการกำหนดคอลัมน์ให้เป็นคีย์หลัก (primary key) หรือ ไม่ซ้ำ (unique) เท่านั้น ที่จะสามารถถูกอ้างอิงในตารางอื่น ๆสำหรับใช้เป็นคีย์นอกเพื่ออ้างอิงตามกฎความสัมพันธ์ (referential integrity)

การกำหนดความสัมพันธ์ระหว่างตารางนั้นสามารถกำหนดขึ้นได้เมื่อมีการสร้างตารางใหม่ขึ้นมา หรือกำหนด

หลังจากที่ตารางได้ถูกสร้างขึ้นมาแล้ว ดังแสดงในตัวอย่างคำสั่ง (Syntax 1-3) ต่อไปนี้

Syntax 1:

```
CREATE TABLE DEPENDANT_TABLE
    (ID         INTEGER REFERENCES BASE_TABLE(UNIQUE_OR_PRIMARY_KEY),
     NAME       VARCHAR(9),
     :
     :
     :
    );
```

Syntax 2:

```
CREATE TABLE DEPENDANT_TABLE
    (ID         INTEGER,
     NAME       VARCHAR(9),
     :
     :
     :,
     CONSTRAINT constraint_name FOREIGN KEY (ID)
          REFERENCES BASE_TABLE(UNIQUE_OR_PRIMARY_KEY)
    );
```

Syntax 3:

```
CREATE TABLE DEPENDANT_TABLE
    (ID         INTEGER,
     NAME       VARCHAR(9),
     :
     :
     :
    );


ALTER TABLE DEPENDANT_TABLE
   ADD CONSTRAINT constraint_name FOREIGN KEY (ID)
        REFERENCES BASE_TABLE(UNIQUE_OR_PRIMARY_KEY);
```

จากตัวอย่างคำสั่ง SQL ข้างต้น กรณีที่ชื่อของข้อกำหนด (constraint name) ไม่ได้ระบุไว้ DB2 จะทำการสร้างชื่อของข้อกำหนดเพื่อใช้ในการอ้างอิงขึ้นมาให้อัตโนมัติ โดยชื่อที่สร้างขึ้นมานี้จะเป็นอักขระที่มีความยาว 15 ตัวอักษรตัวอย่างเช่น 'CC1288717696656'

ตามที่ได้อธิบายในบทที่ 2 เกี่ยวกับลักษณะของความสัมพันธ์ของตาราง กรณีที่มีการลบ (delete) หรือปรับปรุง (update) ข้อมูลในบางแถวจากตารางแม่ ซึ่งจะส่งผลกระทบต่อข้อมูลในตารางลูกที่มีความสัมพันธ์กัน ซึ่งสามารถกำหนดกฎการปฏิบัติต่างๆได้หลายรูปแบบสามารถอธิบายได้ดังต่อไปนี้

- CASCADE

  การกำหนดตัวเลือกCASECADE จะมีความหมายว่าการเปลี่ยนแปลงข้อมูลกรณีที่มีการลบหรือการปรับปรุงข้อมูลแต่ละแถวในตารางแม่ ก็จะส่งผลกระทบต่อทุกแถวในตารางลูกที่อ้างอิงแถวใดๆ ในตารางแม่นั้น ด้วยเช่นกัน

- SET NULL

  การกำหนดตัวเลือก SET NULL นั้นจะมีผลทำให้ค่าในคอลัมน์ที่มีการอ้างอิงในตารางลูกถูกกำหนดค่าให้เป็นเป็นค่าว่าง NULL

- NO ACTION

  สำหรับในตัวเลือกนี้ NO ACTION จะทำให้ไม่เกิดการกระทำใดๆตราบใดที่ยังมีการอ้างค่าระหว่างตารางอยู่

- RESTRICT

  การกำหนดตัวเลือก RESTRICT การปรับปรุง หรือลบแถวในตารางแม่ที่มีข้อมูลอ้างอิงในตารางลูก ไม่อนุญาตให้ดำเนินการได้

คำสั่งต่อไปนี้แสดงวิธีการกำหนดตัวเลือกสำหรับการลบและการปรับปรุงเมื่อมีการกำหนดความสัมพันธ์

```
ALTER TABLE DEPENDANT_TABLE
  ADD CONSTRAINT constraint_name
   FOREIGN KEY column_name
   ON DELETE <delete_action_type>
   ON UPDATE <update_action_type>
;
```

สำหรับกรณีลบข้อมูล เราสามารถเลือกกำหนดตัวเลือกเป็น CASCADE, SET NULL, NO ACTION หรือ RESTRICT ได้ แต่สำหรับการปรับปรุงข้อมูลนั้น จะสามารถกำหนดตัวเลือกเป็น NO ACTION หรือ RESTRICT เท่านั้น

5.2.3 การสร้างสกีมา (Schema)

การจัดเก็บข้อมูลในลักษณะแฟ้มข้อมูล ในไดเรกทอรี หรือโฟลเดอร์บนเครื่องคอมพิวเตอร์ เป็นลักษณะของการจัดเก็บแฟ้มข้อมูลให้เป็นหมวดหมู่ ในลักษณะเดียวกันกับการจัดการฐานข้อมูล โปรแกรม DB2 ใช้สกีมา (schema) ช่วยให้เราจัดกลุ่มของออบเจ็กต์ฐานข้อมูล (database objects) ที่เกี่ยวข้องกันไว้ด้วยกัน ทุกออบเจ็กต์ใน DB2 จะแบ่งออกเป็น 2 ส่วน ซึ่งประกอบด้วย 1) ชื่อสกีมา และ 2) ชื่อของออบเจ็กต์ ดังนั้นเมื่อต้องการสร้างสกีมา เราสามารถใช้คำสั่ง

```
create schema mySchema
```

เมื่อต้องการสร้างตารางโดยระบุชื่อของสกีมา เราสามารถเขียนเป็นคำสั่งในการสร้างตาราง

```
create table mySchema.myTable (col1 integer)
```

กรณีที่ไม่มีการระบุชื่อของสกีมา โปรแกรม DB2 จะใช้ชื่อของผู้ใช้งาน(user ID) ที่ใช้ติดต่อกับฐานข้อมูล DB2 เป็นชื่อสกีมาให้โดยปริยาย (implicit schema) เช่นกรณีที่เราติดต่อใช้งานฐานข้อมูลโดยชื่อผู้ใช้งาน คือ john ชื่อสกีมาจะเป็นชื่อ john โดยอัตโนมัติ แต่ทั้งนี้เราสามารถเปลี่ยนชื่อของสกีมาสำหรับการติดต่อเซสชันปัจจุบันด้วยคำสั่งการตั้งค่า SCHEMA ปัจจุบัน ดังนี้

```
set current schema = myschema
```

5.2.4 การสร้างวิว (View)

การสร้างวิว (View) เป็นการสร้างมุมมองของตารางหนึ่งหรือมากกว่าหนึ่งตารางขึ้นมาใหม่ในลักษณะที่เป็นตารางเสมือน ตารางเสมือนในที่นี้หมายถึงตารางที่ไม่ได้ถูกสร้างขึ้นมาจริง ไม่มีข้อมูลอยู่ในตารางเสมือนหรือวิว แต่ข้อมูลที่แสดงจากวิวจะถูกสร้างขึ้นมาเมื่อมีการเรียกใช้งานวิวตามคำสั่ง Select ในคำสั่งการสร้างวิว ตัวอย่างการสร้างวิวชื่อ MYVIEW จากตารางพนักงาน สามารถใช้คำสั่งดังนี้

```
CREATE VIEW MYVIEW AS
        SELECT LASTNAME, HIREDATE FROM EMPLOYEE
```

จากตัวอย่างนี้วิว MYVIEW จะถูกสร้างขึ้นมา เมื่อมีการสร้างวิวขึ้นมาแล้ว เราสามารถเรียกใช้งานวิวได้ดังเช่นตารางปกติ ตัวอย่างเช่นการแสดงข้อมูลที่อยู่ใน MYVIEW สามารถใช้คำสั่ง select ได้ดังตัวอย่างนี้

```
SELECT * FROM MYVIEW
```

นอกจากนี้การใช้งานวิวยังสามารถใช้งานได้ในลักษณะของการปรับเปลี่ยนมุมมอง จำกัดการเข้าถึงหรือซ่อนข้อมูลในบางคอลัมน์ หรือแม้กระทั่งการใช้วิวเพื่อวัตถุประสงค์การรักษาความปลอดภัยของฐานข้อมูลอีกด้วย

5.2.5 สร้างออบเจ็กต์อื่นๆของฐานข้อมูล

ดังที่ได้กล่าวมาข้างต้นเรามีคำสั่งในการสร้างตาราง (CREATE TABLE) ในระบบฐานข้อมูลเชิงสัมพันธ์นั้นยังมีคำสั่งในการสร้างออบเจ็กต์อื่นๆของฐานข้อมูล (Database Objects) ตามประเภทของออบเจ็กต์ เช่น ดัชนี (index) ฟังก์ชัน (function) โพรซีเยอร์ (procedure) ทริกเกอร์ (trigger) และอื่นๆ สำหรับรายละเอียดเพิ่มเติมเกี่ยวกับคำสั่งเหล่านี้สามารถอ้างอิงได้จาก DB2 9.7 Information Center

5.2.6 การปรับเปลี่ยนโครงสร้างหรือคุณสมบัติของออบเจ็กต์ฐานข้อมูล

หลังจากที่ออบเจ็กต์ฐานข้อมูลถูกสร้างขึ้นมา อ็อบเจ็กต์ฐานข้อมูลเหล่านี้จำเป็นต้องมีการเปลี่ยนแปลงคุณสมบัติให้เหมาะสมกับความต้องการทางธุรกิจที่เปลี่ยนแปลง การลบและการสร้างอ็อบเจ็กต์ฐานข้อมูลขึ้นมาใหม่จัดว่าเป็นวิธีการหนึ่งในการปรับเปลี่ยนนี้ อย่างไรก็ตามการลบออบเจ็กต์อาจจะส่งผลกระทบต่อระบบงานของผู้ใช้ ทางเลือกวิธีการปรับเปลี่ยนคุณสมบัติของอ็อบเจ็กต์ฐานข้อมูลเราสามารถใช้คำสั่ง SQL ALTER ตัวอย่างเช่นสมมุติว่าเราต้องการเปลี่ยนข้อกำหนดของตารางเพื่อให้บางคอลัมน์ไม่สามารถมีค่าว่าง NULLs ได้โดยคำสั่ง SQL ต่อไปนี้

```
alter table myTable alter column col1 set not null
```

ในทำนองเดียวกันการปรับเปลี่ยนโครงสร้างหรือเงื่อนไขอื่นๆของตาราง เช่นการเพิ่มหรือลบคอลัมน์ การกำหนดหรือลบคีย์หลัก เราก็ใช้คำสั่ง ALTER นอกจากนี้เรายังสามารถใช้คำสั่ง ALTER กับออบเจ็กต์อื่นๆของฐานข้อมูลได้เช่นเดียวกัน

5.2.7 การเปลี่ยนชื่อออบเจ็กต์ฐานข้อมูล

การเปลี่ยนชื่อของออบเจ็กต์ฐานข้อมูลสามารถกระทำได้โดยอาศัยคำสั่ง RENAME ในภาษา SQL ดังนี้

```
RENAME <object type> <object name> to <new name>
```

หลังจากที่ได้สร้างฐานข้อมูลขึ้นมาแล้ว ออบเจ็กต์บางประเภทสามารถเปลี่ยนชื่อได้ เช่น ตาราง (table) พื้นที่จัดเก็บออบเจ็กต์(table space) หรือดัชนี (index) โดยไม่ใช่ว่าออบเจ็กต์ทุกประเภทจะสามารถเปลี่ยนชื่อได้ ซึ่งจะต้องแก้ปัญหาโดยวิธีการลบ Drop และสร้างขึ้นมาใหม่เท่านั้น

กรณีที่ต้องการเปลี่ยนชื่อคอลัมน์ในตาราง ก็สามารถใช้คำสั่ง ALTER ร่วมกับคำสั่ง RENAME ดังตัวอย่างต่อไปนี้

```
ALTER TABLE <table name> RENAME COLUMN <column name> TO <new name>
```

5.3 การจัดการข้อมูลโดยอาศัยคำสั่ง SQL

ตามที่ได้อธิบายไปในตอนต้นเกี่ยวกับกลุ่มคำสั่งที่ใช้ในการจัดการข้อมูลในฐานข้อมูล ซึ่งเราจะเรียกกลุ่มคำสั่งนี้ว่า Data Manipulation Language หรือเรียกโดยย่อว่า DML ในส่วนนี้จะอธิบายถึงวิธี การอ่าน การปรับปรุง และการลบข้อมูลโดยใช้คำสั่ง SQL

5.3.1 การอ่านข้อมูล (Select)

การอ่านข้อมูลโดยใช้คำสั่ง SQL เป็นการอ่านหรือการดึงข้อมูล (retrieve) ที่เป็นแถวและคอลัมน์จากตารางในฐานข้อมูลเชิงสัมพันธ์ ในที่นี้ การอ่านข้อมูลจะดำเนินการโดยใช้คำสั่ง select ตัวอย่างสมมุติจากตารางชื่อ myTable คำสั่งที่ง่ายที่สุดในการอ่านข้อมูลจากตารางนี้คือ

```
select * from myTable
```

อักขระพิเศษ '*' ในที่นี้เป็นสัญญลักษณ์ที่ให้ผลลัพธ์ที่ได้แสดงข้อมูลทุกคอลัมน์จากตาราง myTable การใช้ '*' ในการอ่านข้อมูลอาจจะไม่จำเป็นต้องใช้ในทุกกรณี เนื่องจากในการใช้งานจริงบางครั้งเราต้องการที่จะสอบถามข้อมูลเพียงบางคอลัมน์เท่านั้น ในกรณีนี้การเลือกที่จะแสดงข้อมูลเพียงบางคอลัมน์ควรระบุ ดังตัวอย่างคำสั่งต่อไปนี้

```
select col1, col2 from myTable
```

ซึ่งจะเป็นการเลือกดึงเอาเฉพาะข้อมูลในคอลัมน์ col1 และ col2 จากตาราง myTable มาแสดง

5.3.1.1 การจัดลำดับผลลัพธ์ (Order)

ในเบื้องต้น คำสั่ง select ที่ใช้ในการอ่านข้อมูลจะส่งค่าผลลัพธ์ในลักษณะที่ไม่ได้จัดลำดับ การเรียกใช้คำสั่ง select เดียวกันในแต่ละครั้ง จะทำให้ได้ผลลัพธ์เดียวกันแต่อาจจะมีการจัดลำดับที่แตกต่างกัน ซึ่งหากเราต้องการแสดงผลและให้มีการจัดลำดับตามความต้องการ เช่นแสดงผลเรียงลำดับจากน้อยไปหามาก หรือมากไปหาน้อยนั้น เราสามารถใช้คำสั่ง ORDER BY ใน SQL ดังตัวอย่างต่อไปนี้จะส่งชุดผลลัพธ์ตามลำดับของ col1 จากน้อยไปหามาก

```
SELECT col1 FROM myTable ORDER BY col1 ASC
```

ASC (ascending) ในที่นี้จะหมายถึงการจัดลำดับของผลลัพธ์จากน้อยไปมาก ซึ่งเป็นค่าโดยปริยายของการจัดลำดับ และสำหรับกรณีที่ต้องการจัดลำดับของผลลัพธ์จากมากไปหาน้อย สามารถใช้คำสั่ง DESC (decending) ดังตัวอย่างต่อไปนี้

```
SELECT col1 FROM myTable ORDER BY col1 DESC
```

5.3.1.2 เคอร์เซอร์ (Cursor)

เคอร์เซอร์ (Cursor) ใช้ในการชี้ตำแหน่งของแถวที่เป็นผลลัพธ์ที่ได้จากการทำงานของคำสั่ง select คำสั่งสำหรับการประกาศ เปิด ดึงแถวของผลลัพธ์และปิดเคอร์เซอร์ แสดงดังต่อไปนี้

```
DECLARE <cursor name> CURSOR [WITH RETURN <return target>]
        <SELECT statement>;
OPEN  <cursor name>;
FETCH <cursor name> INTO <variables>;
CLOSE <cursor name>;
```

ลักษณะการทำงานของเคอร์เซอร์คือการจัดเก็บผลลัพธ์ที่ได้จากคำสั่ง select มาเก็บในหน่วยความจำ และโปรแกรมประยุกต์สามารถเรียกใช้แต่ละแถวของผลลัพธ์จากเคอร์เซอร์ทีละแถว โดยใช้คำสั่ง FETCH ภายในการวนซ้ำ (loop) ในโปรแกรมประยุกต์ ซึ่งทำให้นักพัฒนาสามารถใช้เคอร์เซอร์เพื่ออ่านข้อมูลแต่ละแถวและอาศัยกระบวนการของการทำงาน (logic) ของโปรแกรมประยุกต์ในการตรวจสอบหรือทำงานกับข้อมูลแต่ละแถว ตัวอย่างชุดคำสั่งย่อยดังต่อไปนี้แสดงการรวมเงินเดือนทั้งหมดของพนักงานโดยใช้เคอร์เซอร์

```
...
 DECLARE p_sum INTEGER;
 DECLARE p_sal INTEGER;
 DECLARE c CURSOR FOR
         SELECT SALARY FROM EMPLOYEE;
 DECLARE SQLSTATE CHAR(5) DEFAULT '00000';
 SET p_sum = 0;
 OPEN c;
 FETCH FROM c INTO p_sal;
 WHILE(SQLSTATE = '00000') DO
   SET p_sum = p_sum + p_sal;
   FETCH FROM c INTO p_sal;
 END WHILE;
 CLOSE c;

...
```

เคอร์เซอร์เป็นวิธีที่ได้รับความนิยมและใช้กันอย่างแพร่หลายกรณีที่ในโปรแกรมประยุกต์ มีการอ่านหรือการเข้าถึงข้อมูลทีละแถวจากตารางโดยการ FETCH และมีการประมวลผลข้อมูลแถวนั้น ๆ ในคราวเดียวกัน

5.3.2 การบันทึกข้อมูล (Insert)

การบันทึกข้อมูลหรือการนำข้อมูลใส่ลงในตารางสามารถทำได้โดยใช้คำสั่ง INSERT ซึ่งภาษา SQL เราสามารถบันทึกข้อมูลทีละแถว หรือทีละหลายแถวต่อหนึ่งคำสั่ง INSERT หรือนำเอาผลลัพธ์ของการอ่านจาก SELECT มาบันทึกลงในตาราง ดังแสดงในตัวอย่างต่อไปนี้

ในตัวอย่างแรกนี้ เป็นคำสั่งที่ใช้สำหรับการบันทึกข้อมูลทีละแถวเข้าในตาราง myTable

```
insert into myTable values (1);
```

```
insert into myTable values (1, 'myName', '2010-01-01');
```

ตัวอย่างที่สองนี้เป็นคำสั่งที่ใช้สำหรับบันทึกข้อมูลหลายแถว (3 แถว)เข้าในตาราง myTable.

```
insert into myTable values (1),(2),(3);
insert into myTable values (1, 'myName1','2010-01-01'),
                           (2, 'myName2','2010-02-01'),
                           (3, 'myName3','2010-03-01');
```

ตัวอย่างสุดท้ายนี้ เป็นคำสั่งที่ใช้ในการบันทึกข้อมูล จากผลลัพธ์ที่ได้จากการอ่านข้อมูลตาราง myTable2 โดยใช้ sub-query “select * from myTable2” เพื่อมาบันทึกเข้าในตาราง myTable

```
insert into myTable (select * from myTable2)
```

5.3.3 การลบข้อมูล (Delete)

การลบข้อมูลออกจากตารางสามารถกระทำได้โดยอาศัยคำสั่ง delete โดยสามารถลบเถวใดแถวหนึ่งออกจากตาราง หรือทีละหลายแถวในหนึ่งคำสั่ง โดยระบุเงื่อนไขสำหรับการลบแถวของข้อมูล โดยการระบุเงื่อนไข where ของคำสั่ง delete ตัวอย่างคำสั่ง delete ต่อไปนี้ เป็นการลบข้อมูลทุกแถวที่มีค่าภานใน col1 มากกว่า 1000

```
DELETE FROM myTable WHERE col1 > 1000
```

หมายเหตุ เมื่อมีการใช้คำสั่ง delete เพื่อทำการลบข้อมูล จะทำการตรวจสอบเงื่อนไขภายใน where ก่อนที่จะทำการลบข้อมูล แต่หากไม่ได้ระบุเงื่อนไข where ข้อมูลทุกแถวในตารางจะถูกลบทิ้งทั้งหมด

5.3.4 การปรับปรุงข้อมูล (Update)

การปรับปรุงข้อมูลที่จัดเก็บอยู่ในตารางสามารถกระทำได้โดยใช้คำสั่ง UPDATE ซึ่งข้อมูลสามารถถูกปรับปรุงทีละหนึ่งแถวหรือทีละหลายแถวในหนึ่งคำสั่ง โดยระบุเงื่อนไขข้อมูลที่ต้องการปรับปรุง โดยการระบุเงื่อนไข where ของคำสั่ง UPDATE สำหรับแต่ละแถวของข้อมูลที่ต้องการปรับปรุง เราสามารถปรับปรุงค่าในคอลัมน์ทีละหลายคอลัมน์พร้อมกันดังตัวอย่างต่อไปนี้

```
UPDATE myTable SET col1 = -1 WHERE col2 < 0 ;

UPDATE myTable SET col1 = -1, col2 = 'a', col3 = '2010-01-01'
WHERE col4 = '0';
```

หมายเหตุ เมื่อมีการใช้คำสั่ง update เพื่อทำการปรับปรุงข้อมูล จะทำการตรวจสอบเงื่อนไขภายใน where ก่อนที่จะทำการปรับปรุงข้อมูล แต่หากไม่ได้ระบุเงื่อนไข where ข้อมูลทุกแถวในตารางจะถูกปรับปรุงทั้งหมด

5.4 การอ่านข้อมูลจากตารางมากกว่าหนึ่งตาราง (Table Joins)

คำสั่ง Select ที่ใช้สำหรับการอ่านหรือเรียกดูข้อมูลในภาษา SQL นั้นเป็นคำสั่งที่ใช้ในการเลือกข้อมูลหนึ่งคอลัมน์หรือหลายคอลัมน์จากตารางเดียว ซึ่งเป็นคำสั่งที่ง่ายและไม่ซับซ้อน จะมีระดับของความซับซ้อนเพิ่มขึ้นเมื่อเราต้องทำการดึงข้อมูลมากกว่าหนึ่งตารางขึ้นไป สำหรับผลลัพธ์ของข้อมูลที่ได้จากคำสั่ง select ที่ใช้สำหรับการเรียกดูข้อมูลมากกว่าหนึ่งตารางขึ้นไปนั้นขึ้นอยู่กับวิธีการเชื่อมโยงตาราง ที่เรียกว่าประเภทของการจอยน์ ซึ่งประเภทของการจอยน์ตาราง มีอยู่ 2 ประเภทใน SQL คือ

1. อินเนอร์จอยน์ (Inner Join)
2. เอาน์เตอร์จอยน์ (Outer Join)

ซึ่งเราจะอธิบายในรายละเอียดต่อไป

5.4.1 การเชื่อมโยงตารางแบบอินเนอร์จอยน์ Inner Join

ลักษณะของการเชื่อมโยงแบบอินเนอร์จอยน์ (Inner Join) นั้นเป็นรูปแบบการเชื่อมโยงแบบปรกติในคำสั่ง SQL ซึ่งสามารถแบ่งการเชื่อมโยงได้เป็น 3 ประเภท

- Equi-join การเชื่อมโยงแบบอิควิ
- Natural join การเชื่อมโยงแบบธรรมชาติ
- Cross join การเชื่อมโยงแบบไขว้

5.4.1.1 Equi-join

การเชื่อมโยงตารางแบบ Equi-join ในรูปแบบนี้จะเป็นการเชื่อมโยงในลักษณะที่ทั้งสองตารางนั้นมีค่าของข้อมูลในคอลัมน์ที่เท่ากัน ดังตัวอย่างต่อไปนี้

```
SELECT *
  FROM student, enrollment
  WHERE student.enrollment_no=enrollment.enrollment_no

        OR
SELECT *
  FROM student
      INNER JOIN enrollment
      ON student.enrollment_no=enrollment.enrollment_no
```

จากตัวอย่างจะเห็นได้ว่า การเชื่อมโยงตาราง student และ enrollment นั้นจะมีการเทียบค่าของคอลัมน์ enrollment_no จากตารางทั้งสองที่มีค่าเท่ากัน

5.4.1.2 Natural Join

การเชื่อมโยงตารางแบบ Natural นั้น จะมีการปรับปรุงเพิ่มเติมจาก equi-join โดยที่การเชื่อมโยงระหว่างตารางนั้น ไม่จำเป็นต้องมีการระบุชื่อคอลัมน์ที่ใช้ในการเชื่อมโยง โดยที่ DB2 จะทำการเลือกคอลัมน์ที่

มีชื่อเหมือนกันมาทำเทียบค่าของคอลัมน์ที่มีค่าเท่ากันให้โดยอัตโนมัติ ซึ่งการเชื่อมแบบ Natural นั้นจะทำการตัดคอลัมน์ที่เหมือนกันออกให้เหลือคอลัมน์เดียว ดังตัวอย่างของคำสั่งที่ใช้ในการเชื่อมตารางแบบ Natural ต่อไปนี้

```
SELECT *
  FROM STUDENT
  NATURAL JOIN ENROLLMENT
```

การเชื่อมโยงแบบ Natural นั้นอาจจะทำให้เกิดข้อสงสัยและความกำกวมในบางครั้ง เช่นในกรณีที่มีจำนวนคอลัมน์ที่มีชื่อเหมือนกันเกิดขึ้นมากกว่าหนึ่งคอลัมน์ในแต่ละตาราง หรือในความเป็นจริงนั้นอาจเป็นไปได้ที่ผู้ใช้งานมีความจำเป็นต้องการเชื่อมโยงคอลัมน์ที่อาจจะชื่อไม่เหมือนกัน ซึ่งจากประเด็นต่างๆเหล่านี้เป็นเหตุผลที่ทำให้ระบบฐานข้อมูลส่วนใหญ่จะไม่สนับสนุนการเชื่อมโยงแบบ Natural Join นี้

5.4.1.3 Cross Join

การเชื่อมโยงตารางที่เป็นลักษณะของ Cross นั้นเป็นการเชื่อมโยงแบบการไขว้ตารางโดยไม่มีการระบุคอลัมน์ที่ใช้ในการเชื่อมโยง ซึ่งจะได้ผลลัพธ์ของการเชื่อมโยงโดยที่แต่ละแถวของตารางที่หนึ่งรวมกับทุกแถวของตารางที่สอง โดยเรียกผลลัพธ์นี้ว่า คาร์ทีเซียนโพรดักท์ (Cartesian Product) ดังตัวอย่างต่อไปนี้

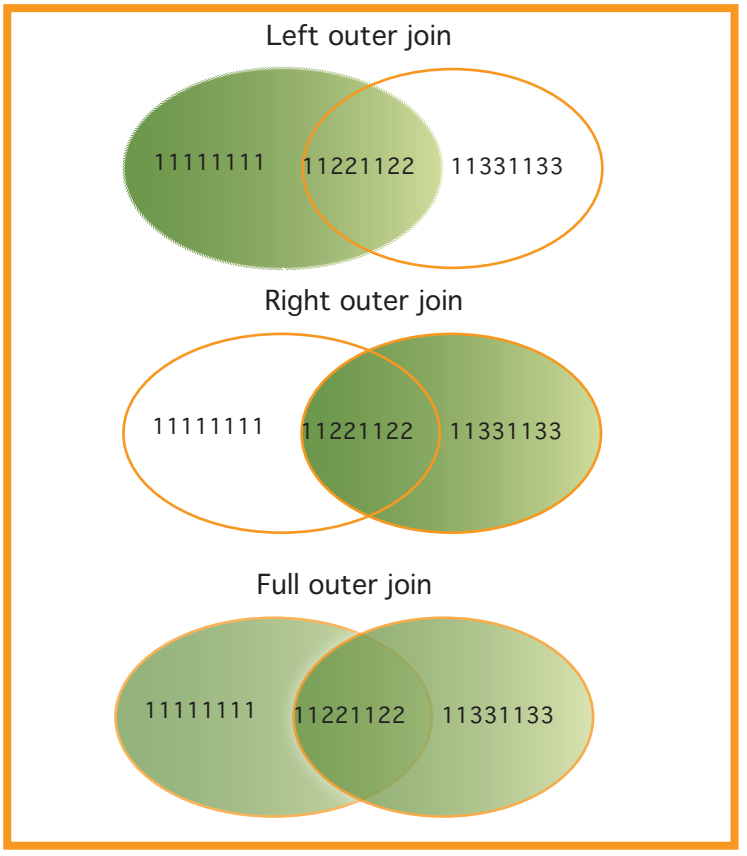
```
SELECT *
FROM STUDENT, ENROLLMENT
```

5.4.2 การเชื่อมโยงตารางแบบเอาท์เทอร์จอยน์ Outer Joins

การเชื่อมโยงตารางในลักษณะของ Outer Join นั้นเป็นรูปแบบคำสั่งแบบพิเศษที่อยู่ในภาษา SQL ซึ่งเชื่อมโยงตารางรูปแบบนี้ ชื่อตารางแรกที่เราจะต้องกำหนดใน FROM ของคำสั่ง SQL ซึ่งเราจะถือว่าเป็นตารางทางด้านซ้าย (LEFT table) และตารางที่จะเชื่อมโยงด้วยเรียกว่าตารางด้านขวา (RIGHT table) โดยประเภทของ Outer Join นั้นสามารถแบ่งออกได้เป็น 3 ประเภท ดังนี้

- Left outer join - การเชื่อมโยงตารางโดยยึดแถวของตารางทางซ้ายเป็นหลัก
- Right outer join – การเชื่อมโยงตารางโดยยึดแถวของตารางทางขวาเป็นหลัก
- Full outer join – การเชื่อมโยงตารางโดยยึดแถวของตารางทั้งทางซ้ายและทางขวาเป็นหลัก

รูป 5.1 แสดงรูปแบบของการเชื่อมโยงตารางแบบ Outer join ทั้ง 3 ประเภท



รูปที่ 5.1 – แสดงรูปแบบของการเชื่อมโยงตารางแบบ Outer join

ในส่วนต่อไปนี้จะอธิบายถึงรายละเอียดเพิ่มเติมของการเชื่อมโยงตารางแบบ Outer ในแต่ละประเภท เพื่อให้เกิดความชัดเจนมากยิ่งขึ้นในแต่ละกรณี ตัวอย่างของข้อมูลจากสองตาราง Student และ ตาราง Enrollment แสดงในรูปที่ 5.2

Student Table

Name	Year	Enrollment no
John Smith	1	11221122
Raul Chong	2	11331133

Enrollment table Table

Name	SubjectID	Enrollment no
Physics	100	11221122
Mathemethics	200	11111111

รูปที่ 5.2 – ตัวอย่างข้อมูลในตารางเพื่อใช้ในการเชื่อมโยงตารางแบบ Outer Join

5.4.2.1 Left outer joins

ผลลัพธ์ของการเชื่อมโยงคือเซตของผลลัพธ์จากการทำยูเนี่ยน (Union)ของผลลัพธ์การทำ equi-join รวมกับแถวใด ๆ จากตารางด้านซ้าย ที่ค่าในคอลัมน์ไม่มีในตารางที่มาเชื่อมโยงด้วย จากตัวอย่างคำสั่งต่อไปนี้ จะส่งผลลัพธ์แสดงในรูปที่ 5.3.

```
SELECT *
  FROM STUDENT
  LEFT OUTER JOIN ENROLLMENT
  ON STUDENT.ENROLLMENT_NO = ENROLLMENT_NO
```

Left outer join

Name	Year	Enrollment no	Name	SubjectID
John Smith	1	11221122	Physics	100
Raul Chong	2	11331133	-	-

รูปที่ 5.3 – ผลลัพธ์จากการเชื่อมโยง Left outer join

5.4.2.2 Right outer join

ผลลัพธ์ของการเชื่อมโยงคือเซตของผลลัพธ์จากการทำยูเนี่ยน (Union)ของผลลัพธ์การทำ equi-join รวมกับแถวใด ๆ จากตารางด้านขวา ที่ค่าในคอลัมน์ไม่มีในตารางที่มาเชื่อมโยงด้วย จากตัวอย่างคำสั่งต่อไปนี้ จะส่งผลลัพธ์แสดงในรูปที่ 5.4.

```
SELECT *
  FROM STUDENT
  RIGHT OUTER JOIN ENROLLMENT
  ON STUDENT.ENROLLMENT_NO = ENROLLMENT_NO
```

Right outer join

Name	Year	Enrollment no	Name	SubjectID
John Smith	1	11221122	Physics	100
-	-	11111111	Mathematics	200

รูปที่ 5.4 – ผลลัพธ์จากการเชื่อมโยง Right outer join

5.4.2.3 Full outer join

ผลลัพธ์ของการเชื่อมโยงคือเซตของผลลัพธ์จากการทำยูเนี่ยน (Union)ของผลลัพธ์การทำ equi-join รวมกับแถวใด ๆ จากตารางด้านซ้ายและตารางด้านขวา ที่ค่าในคอลัมน์ไม่มีในตารางที่มาเชื่อมโยงด้วย จากตัวอย่างคำสั่งต่อไปนี้จะส่งผลลัพธ์แสดงในรูปที่ 5.5.

```
SELECT *
  FROM STUDENT
  FULL OUTER JOIN ENROLLMENT
  ON STUDENT.ENROLLMENT_NO = ENROLLMENT_NO
```

Full outer join

Name	Year	Enrollment no	Name	SubjectID
John Smith	1	11221122	Physics	100
Raul Chong	2	11331133	-	-
-	-	11111111	Mathematics	200

รูป 5.5 – ผลลัพธ์จากการเชื่อมโยง Full outer join

การเชื่อมโยงของตาราง Outer Join แต่ละประเภทนั้นจะให้ผลลัพธ์ที่แตกต่างกันไป ดังนั้นการเลือกใช้แต่ละประเภท ควรคำนึงถึงความต้องการทางธุรกิจ (Business Requirement) เป็นหลัก ตัวอย่างเช่น กรณีที่เราต้องการเลือกข้อมูลรายการของนักเรียนที่มีการลงทะเบียนในแต่ละวิชา รวมถึงนักเรียนที่ไม่ได้ลงทะเบียน เราควรเลือกใช้วิธีการเชื่อมโยงตารางแบบ Left Outer Join

5.5 การรวมผลลัพธ์โดย Union, Intersection, Difference

ภาษา SQL สนับสนุนทฤษฎี เกี่ยวกับ SET OPERATORS คือ การทำ Union, Intersection, Difference

5.5.1 Union

การทำ Union สามารถใช้สำหรับรวมผลลัพธ์ของข้อมูลสองชุดเข้าด้วยกัน โดยมีเงื่อนไขว่าผลลัพธ์ของข้อมูลสองชุดนั้นจะต้องมีจำนวนคอลัมน์ที่เท่ากัน และประเภทของข้อมูลตามลำดับของคอลัมน์นั้นจะต้องสอดคล้องกัน โดยที่การ Union นั้นจะลบแถวที่มีค่าของข้อมูลทุกคอลัมน์ที่ซ้ำกันออกจากผลลัพธ์ ดังแสดงในตัวอย่างรูป 5.6

```
SELECT * FROM student_table_a
UNION
SELECT * FROM student_table_b
```

Student Table A

Name	Year	Enrollment no
John Smith	1	11221122
Raul Chong	2	11331133

Student Table B

Name	Year	Enrollment no
John Smith	1	11221122
Alan Doster	2	44556677

A Union B

Name	Year	Enrollment no
John Smith	1	11221122
Raul Chong	2	11331133
Alan Doster	2	44556677

รูป 5.6 - ตัวอย่างของการ Union

จากรูปที่ 5.6 จะเห็นได้ว่าการใช้ Union นั้นจะลบแถวที่มีค่าของข้อมูลทุกคอลัมน์ซ้ำกันออกจากผลลัพธ์ แต่ในบางสถานการณ์ เราอาจจำเป็นต้องแสดงแถวที่ซ้ำกัน ซึ่งจำเป็นต้องระบุ UNION ALL แทน UNION ดังนี้

```
SELECT * FROM student_table_a
UNION ALL
SELECT * FROM student_table_b
```

รูป 5.7 แสดงตัวอย่างผลลัพธ์ของการใช้ Union all

A Union All B

Name	Year	Enrollment no
John Smith	1	11221122
Raul Chong	2	11331133
John Smith	1	11221122
Alan Doster	2	44556677

รูปที่ 5.7 – ผลลัพธ์ของการทำ Union all

5.5.2 Intersection

การทำ intersaction นั้นจะส่งกลับแถวของผลลัพธ์ที่ข้อมูลของทั้งสองชุดนั้นเหมือนกัน แสดงตัวอย่างคำสั่งดังต่อไปนี้

```
select * from student_table_a
INTERSECT
select * from student_table_b
```

รูป 5.8 แสดงตัวอย่างผลลัพธ์จากคำสั่งข้างบน

A Intersect B

Name	Year	Enrollment no
John Smith	1	11221122

รูปที่ 5.8 – ตัวอย่างแสดงผลลัพท์ของการทำ INTERSECT

จากตัวอย่างดังกล่าว จะเห็นว่าการทำ intersect นั้นจะได้ผลลัพท์ของข้อมูลที่อยู่ในทั้งสองตาราง คือ ตาราง A และ B อย่างไรก็ตามแถวของข้อมูลที่ซ้ำหรือเหมือนกันจะปรากฏเพียงแถวเดียว ถึงแม้ว่าจะมีหลายแถวที่ซ้ำกันในตาราง A หรือ B ก็ตาม ทั้งนี้กรณีที่เราต้องการแสดงแถวของข้อมูลทั้งหมดที่ซ้ำกัน เราสามารถใช้ INTERSECT ALL ดังแสดงตามตัวอย่าง

```
select * from student_table_a
INTERSECT ALL
select * from student_table_b
```

5.5.3 Difference (Except)

การทำ Difference (EXCEPT) ส่งกลับแถวของผลลัพธ์ของข้อมูลมีอยู่ในตารางด้านซ้ายเท่านั้น ซึ่งสามารถอธิบายตามคำพูดที่ว่าA EXCEPT B = A MINUS [A INTERSECT B]

ดังตัวอย่าง

```
select * from student_table_a
EXCEPT
select * from student_table_b
```

ซึ่งจะส่งกลับผลลัพธ์ดังที่แสดงในรูป 5.9.

A Except B

Name	Year	Enrollment no
Alan Doster	2	44556677

รูปที่ 5.9 – ตัวอย่างแสดงผลลัพท์ของการทำ EXCEPT

การทำ EXCEPT นั้นจะส่งกลับแถวของผลลัพธ์ของข้อมูลที่มีอยู่ในตาราง A แต่ไม่อยู่ในตาราง B อย่างไรก็ตามแถวของข้อมูลที่ซ้ำหรือเหมือนกันจะปรากฏเพียงแถวเดียว ถึงแม้ว่าจะมีหลายแถวที่ซ้ำกันในตาราง A ทั้งนี้กรณีที่เราต้องการแสดงแถวของข้อมูลทั้งหมดที่ซ้ำกัน เราสามารถใช้ EXCEPT ALL ดังแสดงตามตัวอย่าง

```
select * from student_table_a
EXCEPT ALL
select * from student_table_b
```

5.6 ตัวดำเนินการเชิงสัมพันธ์ (Relational Operators)

ตัวดำเนินการเชิงสัมพันธ์ (Relational Operators) เป็นตัวที่ใช้สำหรับการทดสอบค่าของข้อมูล และการดำเนินการที่สามารถกระทำกับข้อมูล ตัวดำเนินการเหล่านี้รวมถึง

- การดำเนินการทางคณิตศาสตร์พื้นฐาน เช่น ‘+’, ‘ -’, ‘ *’ และ ‘/’
- ตัวดำเนินการเชิงตรรกะ เช่น ‘AND’, ‘OR’ และ ‘ NOT’
- ตัวดำเนินการที่จัดการกับข้อมูลประเภทตัวอักษร เช่น ‘CONCATENATE’, ‘LENGTH’ ‘SUBSTRING’
- ตัวดำเนินการที่ใช้ในการเปรียบเทียบ เช่น ‘=’, ‘ <’, ‘ >’, ‘ > =’, ‘ < = ‘และ’ ! =’
- ตัวดำเนินการที่ใช้ในการจัดกลุ่ม (grouping) และการรวมค่า (aggregate)
- ตัวดำเนินการอื่น ๆ เช่น DISTINCT

เราจะข้ามการอธิบาย การดำเนินการทางคณิตศาสตร์ ตัวดำเนินการที่ใช้ในการเปรียบเทียบ และตัวดำเนินการเชิงตรรกะ ซึ่งเป็นพื้นฐาน แต่จะอธิบายรายละเอียดตัวดำเนินการอื่นๆ

5.6.1 ตัวดำเนินการจัดกลุ่ม (Grouping Operators)

ตัวดำเนินการสำหรับการจัดกลุ่มนั้นสามารถกระทำได้กับข้อมูลที่มีจำนวนแถวตั้งแต่สองแถวขึ้นไป และเพื่อทำการหาผลลัพธ์ในลักษณะของการรวบรวมค่าของข้อมูล ตัวอย่างเช่น กรณีที่มีข้อมูลของนักเรียนที่ลง

ทะเบียนในหลักสูตรต่างๆ นักเรียนหนึ่งคนสามารถลงทะเบียนเรียนได้ในหลายหลักสูตร และในแต่ละสูตรก็จะมีนักเรียนลงทะเบียนเรียนได้หลายคนเช่นเดียวกัน ดังนั้นเราสามารถนับจำนวนของนักเรียนทั้งหมดที่ลงทะเบียนได้ดังนี้

```
select count(*) from students_enrollment
```

อย่างไรก็ตามถ้าเราต้องการนับจำนวนนักเรียนที่ได้ลงทะเบียนสำหรับแต่ละหลักสูตร เราจำเป็นต้องใช้ตัวดำเนินการสำหรับการจัดกลุ่ม Group by เพื่อทำการจัดกลุ่มโดยเรียงลำดับของข้อมูลตามแต่ละหลักสูตรและนับจำนวนของนักเรียนที่ลงทะเบียนเรียนในแต่ละหลักสูตร ดังตัวอย่างต่อไปนี้

```
select course_enrolled, count(*)
from students_enrollment
group by course_enrolled
```

----------ผลลัพธ์----------

COURSE_ENROLLED	STUDENT_COUNT
English	10
Mathematics	30
Physics	60

นอกจากนี้ การจัดกลุ่มยังสามารถทำมากกว่าหนึ่งคอลัมน์ได้เช่นเดียวกัน ซึ่งในกรณีนี้การจัดลำดับของการจัดกลุ่มนั้นจะทำจากคอลัมน์ซ้ายสุดก่อน

5.6.2 ตัวดำเนินการ Aggregation

ตัวดำเนินการ Aggregation นั้นสามารถกระทำได้กับข้อมูลที่มีจำนวนแถวตั้งแต่สองแถวขึ้นไปโดยผลลัพธ์ที่ได้จากตัวดำเนินการ Aggregation นั้นจะแสดงผลลัพธ์

สเกลา (Scalar result set) คือจะเป็นผลลัพธ์ที่เกิดจากข้อมูลหลาย ๆ แถวรวมกันเป็นหนึ่งค่า ตัวอย่างเช่นCOUNT, SUM, AVERAGE, MINIMUM, MAXIMUM และอื่นๆ ซึ่งตัวดำเนินการเหล่านี้จะต้องใช้ร่วมกับตัวดำเนินการ GROUP BY ตามที่แสดงในคำสั่ง SQL ข้างต้น

5.6.3 ตัวดำเนินการ HAVING

ตัวดำเนินการ HAVING นั้นเป็นตัวดำเนินการพิเศษที่ใช้ทำงานร่วมกับตัวดำเนินการ GROUP BY เท่านั้น โดยที่ HAVING นั้นใช้เพื่อระบุเงื่อนไขในการเลือกกลุ่มของ ข้อมูลที่มีการจัดกลุ่มแล้ว ดังตัวอย่างข้างต้นที่แสดงชื่อหลักสูตรและจำนวนที่นักเรียนลงทะเบียนในแต่ละหลักสูตร ถ้าต้องการทราบว่ามีหลักสูตรใดบ้างที่จำนวนของนักเรียนที่ลงทะเบียนในหลักสูตรน้อยกว่า 5 คน สามารถทำได้ดังตัวอย่างดังนี้

```
SELECT course_enrolled, count(*)
FROM students_enrollment
GROUP BY course_enrolled
HAVING count(*) < 5
```

การคัดเลือกผลลัพธ์ของข้อมูลออกไปนั้น เราสามารถระบุเงื่อนไขใน WHERE แต่ WHERE ไม่สามารถใช้ในการคัดเลือกผลลัพธ์ลักษณะที่เป็นกลุ่มของข้อมูลออกไป

5.7 ซับคิวรี (Sub-queries)

ในภาษา SQL นั้นเราสามารถเขียนเป็นคำสั่งเพื่อใช้ในการเรียกดูข้อมูลหรือสอบถามข้อมูลซึ่งใช้คำสั่ง Select ซึ่งเราเรียกว่า “คิวรี” (Query) เมื่อเราเขียนคิวรีอีกคิวรีหนึ่งซ้อนคิวรี หลัก (Main Query หรือ Parent Query) คิวรีที่อยู่ภายในที่ซ้อนนั้นเราจะเรียกว่า“ซับคิวรี” (Sub-query) หรือ Inner Query ซับคิวรีนี้อาจส่งผลลัพธ์คืนเป็นค่าสเกลาค่าเดียว หรือ ค่าข้อมูลปกติค่าเดียวหรือหลายค่า หรือค่าว่าง NULL สำหรับลำดับของการประมวลผลนั้น ซับคิวรีมีการประมวลผลก่อน และจึงทำการประมวลผล คิวรีหลักโดยใช้ข้อมูลที่ส่งออกมาจากซับคิวรี

5.7.1 ซับคิวรีที่ส่งกลับค่ากลับแบบสเกลา

ค่าสเกลา (Scalar) คือ ค่าผลลัพธ์ที่เกิดจากข้อมูลในคอลัมน์ใดๆ หลาย ๆ แถว หรือทุกแถวในตารางรวมกันเป็นหนึ่งค่า ตัวอย่างจากคอลัมน์ ชื่อ อายุ หลักสูตร ปี และอื่นๆ ตัวอย่างคิวรีต่อไปนี้ใช้ซับคิวรีที่ส่งผลลัพธ์กลับเป็นค่าสเกลา ซึ่งผลลัพธ์ที่ได้จะถูกนำไปใช้ในคิวรีหลักเพื่อนำไปประมวลผลต่อเพื่อได้ผลลัพธ์ตามที่ต้องการ

```
SELECT name FROM students_enrollment
    WHERE age = ( SELECT min(age) FROM students )
```

ตัวอย่างคิวรีข้างต้นนี้แสดงรายการชื่อของนักเรียนทั้งหมดที่มีอายุเท่ากับอายุของนักเรียนที่มีอายุน้อยที่สุด (min) ซึ่งเป็นผลลัพธ์ที่ได้มาจากซับคิวรี

5.7.2 ซับคิวรีที่ส่งค่ากลับแบบเว็กเตอร์

การส่งค่ากลับของซับคิวรีในลักษณะที่เป็นชุดข้อมูลหลายค่าของคอลัมน์หนึ่ง เช่นรายชื่อของนักเรียน หรือข้อมูลหลายค่าของคอลัมน์หลายคอลัมน์ เช่น รายชื่อของนักเรียน อายุ และ วันเกิด เราเรียกชุดของผลลัพธ์นี้ว่าผลลัพธ์ที่อยู่ในรูปของเว็กเตอร์ (Vector) ตัวอย่างเช่น เพื่อแสดงรายชื่อนักเรียนที่มีการลงทะเบียนหลักสูตรต่าง ๆของภาควิชาวิทยาการคอมพิวเตอร์ ซึ่งจะแสดงได้ดังตัวอย่างต่อไปนี้

```
SELECT name FROM students
    WHERE course_enrolled IN
    (
        SELECT distinct course_name
        FROM courses
        WHERE department_name = ‘Computer Science’
    )
```

จากตัวอย่างข้างต้นจะเห็นว่าซับคิวรี จะทำการส่งรายชื่อหลักสูตรทั้งหมดที่เป็นของภาควิชา “วิทยาศาสตร์คอมพิวเตอร์” และคิวรีหลักแสดงรายชื่อนักเรียนที่มีการลงทะเบียนหลักสูตรต่าง ๆ ที่เป็นผลลัพธ์จากซับคิวรี

หมายเหตุ การเรียกใช้งานคำสั่ง SQL อาจจะเขียนได้หลายรูปแบบซึ่งจะให้ผลลัพธ์ของคำตอบที่เหมือนกัน ซึ่งตัวอย่างสาธิตวิธีใช้งานคำสั่ง SQL ในที่นี้อาจจะเป็นวิธีหนึ่ง ซึ่งอาจจะไม่ใช่วิธีที่มีประสิทธิภาพดีที่สุดก็ได้

5.7.3 Correlated Sub-query

เมื่อซับคิวรีมีการทำงานและมีการอ้างค่าจากข้อมูลของคิวรีหลัก เราจะเรียกคิวรีนี้ว่าเป็น correlated sub-query ดังตัวอย่างต่อไปนี้

```
SELECT dept, name, marks
FROM final_result a WHERE marks = (SELECT max(marks) FROM final_result WHERE
dept = a.dept)
```

จากตัวอย่างคำสั่งข้างต้นผลลัพธ์จะแสดงรายชื่อของนักเรียน แผนก และค่าคะแนนที่ได้ โดยคะแนนที่ได้นี้จะเป็นคะแนนสูงสุดของแผนกที่นักเรียนศึกษาอยู่ แต่ละแถวจากตาราง final_result ซับคิวรีจะหาค่าคะแนนที่สูงสุด max(marks)ของแผนกของแถวนั้น และถ้าคะแนนของนักเรียนในแถวนั้นมีค่าเท่ากับผลลัพธ์จากซับคิวรี ข้อมูลของนักเรียนแถวนั้นจะเป็นผลลัพธ์ของคิวรีหลัก

5.7.4 การใช้ซับคิวรีใน FROM ของคำสั่ง Select

การใช้คิวรีเพื่อแสดงข้อมูล เราจะระบุชื่อของตารางข้อมูลในส่วนของ FROM แต่สำหรับกรณีที่เราต้องการให้แหล่งที่มาของข้อมูลได้ผลลัพธ์มาจากซับคิวรี เราสามารถใช้ซับคิวรี ระบุในส่วนของ FROM ของคำสั่ง Select ตามตัวอย่างต่อไปนี้

```
SELECT dept, max_marks, min_marks, avg_marks
FROM
    (
     SELECT dept,
          max(marks) as max_marks,
          min(marks) as min_marks,
          avg(marks) as avg_marks
     FROM final_result GROUP BY dept
    )
WHERE (max_marks – min_marks) > 50 and avg_marks < 50
```

จากคิวรีข้างต้นจะเห็นว่ามีการใช้ซับคิวรี ระบุใน from ในที่นี้ ซับคิวรี จะทำการส่งกลับค่าคะแนนสูงสุด คะแนนต่ำสุด และคะแนนเฉลี่ยสำหรับแต่ละแผนก ซึ่งผลลัพธ์จากซับคิวรีนี้จะถูกนำไปใช้ต่อสำหรับคิวรีหลักและมีการเลือกข้อมูลเพิ่มเติมโดยการเพิ่มเงื่อนไขในส่วน WHERE ของคิวรีหลัก

5.8 การเปรียบเทียบระหว่างแนวคิดเชิงวัตถุ (Object-oriented concept) กับแนวคิดเชิงสัมพันธ์ (Relational concept)

ปัจจุบันนักพัฒนาโปรแกรมประยุกต์ส่วนใหญ่ใช้ภาษาที่พัฒนาโปรแกรม ที่เรียกว่าภาษาพัฒนาโปรแกรมเชิงวัตถุ (Object-oriented programming languages) อย่างไรก็ตามพวกเขาทั้งหลายยังมีความจำเป็นที่ต้องมีการเข้าถึงฐานข้อมูลเชิงสัมพันธ์อยู่ ตารางต่อไปนี้จะแสดงการเปรียบเทียบแนวคิดระหว่างแนวคิดเชิงวัตถุและแนวคิดของฐานข้อมูลเชิงสัมพันธ์ แต่ไม่ได้แสดงการเปรียบเทียบข้อมูลทั้งหมดในที่นี้

Object-oriented concept - Class elements	Relational database concept
Name	Table name
Attribute	Column name
Method	Stored procedure
Constructor/Destructor	Triggers
Object identifier	Primary Key

ตาราง 5.1 การเปรียบเทียบแนวคิดเชิงวัตถุกับแนวคิดของฐานข้อมูลเชิงสัมพันธ์

ไลบรารีการแมปเชิงวัตถุ (Object-relational mapping - ORM) เช่น ไฮเบอร์เนต เป็นสิ่งที่ได้รับความนิยมในการเชื่อมโยง ระหว่างเทคโนโลยีในโลกเชิงวัตถุ (object-oriented) กับฐานข้อมูลเชิงสัมพันธ์ (relational) หรือ ในกรณีของ pureQuery ซึ่งเป็นเทคโนโลยีจาก ไอบีเอ็ม ซึ่งจะช่วยเพิ่มประสิทธิภาพในการทำงานในส่วนของโปรแกรมประยุกต์ภาษา Java สำหรับข้อมูลเพิ่มเติมเกี่ยวกับ pureQuery สามารถดาวน์โหลด eBook “Getting started with pureQuery” ได้ฟรี และถือได้ว่าเป็นซีรีย์หนึ่งของหนังสือเล่มนี้

5.9 กรณีศึกษา – การจัดการระบบห้องสมุด – ส่วนที่ 3

ในช่วงสุดท้ายนี้จะเป็นการแสดงถึงกรณีศึกษา การจัดการฐานข้อมูลสำหรับระบบห้องสมุด ในการแปลงแบบจำลองลอจิคัลให้เป็นแบบจำลองทางกายภาพ สำหรับการนำไปสร้างเป็นฐานข้อมูลในโปรแกรม DB2 ตารางต่อไปนี้ แสดงให้เห็นถึงความเกี่ยวข้องระหว่างแนวคิดของการออกแบบ ตั้งแต่ระดับคอนเซ็ปต์ (conceptual) ระดับลอจิคัล (logical) และระดับกายภาพ (physical)

Conceptual modeling concept	Logical modeling concept	Physical modeling concept
Name of entity set	Relation variable, R	Table name
Entity set	Relation	Table
Entity	Tuple	Row
Attribute	Attribute, A1, A2, etc.	Column
Relationship set	A pair of primary key – foreign key	Constraint
Unique identifier	Primary key	Primary key

การแปลงแบบจำลองในระดับลอจิคัลไปสู่แบบจำลองทางกายภาพนั้นจะไม่ค่อยซับซ้อน เนื่องจากแบบ

จำลองในระดับลอจิคัลทำให้มองเห็นว่ามีตารางใดบ้างที่จะนำไปสร้างเป็นฐานข้อมูลของระบบการจัดการห้องสมุด ซึ่งในส่วนที่ต้องทำต่อไปคือการระบุประเภทของข้อมูล (sub domain) ของแต่ละแอตทริบิวต์ภายในตาราง และกำหนดข้อกำหนด (constraint) ที่จำเป็น โดยการระบุชื่อของข้อกำหนดต่างๆนั้นสามารถระบุคำนำหน้าชื่อของข้อกำหนด ซึ่งเราแนะนำให้มีการใช้คำนำหน้าชื่อของข้อกำหนด ดังตัวอย่างต่อไปนี้

- PRIMARY KEY: pk_
- UNIQUE: uq_
- DEFAULT: df_
- CHECK: ck_
- FOREIGN KEY: fk_

ในที่นี้ ให้พิจารณาในแต่ละตารางที่ได้มีการกำหนดประเภทของข้อมูล (sub domain) และข้อกำหนด (constraint)

ตาราง BORROWER

Attribute name	Domain	Sub-domain	Optional	Constraints
BORROWER_ID	Text	CHAR(5)	No	Pk_
FIRST_NAME	Text	VARCHAR(30)	No	
LAST_NAME	Text	VARCHAR(30)	No	
EMAIL	Text	VARCHAR(40)	Yes	
PHONE	Text	VARCHAR(15)	Yes	
ADDRESS	Text	VARCHAR(75)	Yes	
CITY	Text	CHAR(3)	No	
COUNTRY	Text	DATE	No	

ตาราง AUTHOR

Attribute name	Domain	Sub-domain	Optional	Constraints
AUTHOR_ID	Text	CHAR(5)	No	Pk_
FIRST_NAME	Text	VARCHAR(30)	No	
LAST_NAME	Text	VARCHAR(30)	No	
EMAIL	Text	VARCHAR(40)	Yes	

PHONE	Text	VARCHAR(15)	Yes	
ADDRESS	Text	VARCHAR(75)	Yes	
CITY	Text	VARCHAR(40)	Yes	
COUNTRY	Text	VARCHAR(40)	Yes	

ตาราง BOOK

Attribute name	Domain	Sub-domain	Optional	Constraints
BOOK_ID	Text	CHAR(5)	No	Pk_
TITLE	Text	VARCHAR(40)	No	
EDITION	Numeric	INTEGER	Yes	
YEAR	Numeric	INTEGER	Yes	
PRICE	Numeric	DECIMAL(7,2)	Yes	
ISBN	Text	VARCHAR(20)	Yes	
PAGES	Numeric	INTEGER	Yes	
AISLE	Text	VARCHAR(10)	Yes	
DECRIPTION	Text	VARCHAR(100)	Yes	

ตาราง LOAN

Attribute name	Domain	Sub-domain	Optional	Constraints
BORROWER_ID	Text	CHAR(5)	No	Pk_, fk_
COPY_ID	Text	VARCHAR(30)	No	Pk_, fk_
LOAN_DATE	Text	DATE	No	< RETURN_DATE
RETURN_DATE	Text	DATE	No	

ตาราง COPY

Attribute name	Domain	Sub-domain	Optional	Constraints
COPY_ID	Text	CHAR(5)	No	Pk_
BOOK_ID	Text	VARCHAR(30)	No	Fk_
STATUS	Text	VARCHAR(30)	No	

ตาราง AUTHOR_LIST

Attribute name	Domain	Sub-domain	Optional	Constraints
AUTHOR_ID	Text	CHAR(5)	No	Pk_, fk_
BOOK_ID	Text	VARCHAR(30)	No	Pk_, fk_
ROLE	Text	VARCHAR(30)	No	

และสำหรับในขั้นตอนต่อไป เราสามารถสร้างตารางโดยอาศัยคำสั่ง SQL ต่อไปนี้

```
CREATE TABLE AUTHOR
(
        AUTHOR_ID CHAR(5) CONSTRAINT AUTHOR_PK PRIMARY KEY(AUTHOR_ID)
NOT     NULL,
        LASTNAME VARCHAR(15) NOT NULL,
        FIRSTNAME VARCHAR(15) NOT NULL,
        EMAIL VARCHAR(40),
        CITY VARCHAR(15),
        COUNTRY CHAR(2)
)

CREATE TABLE AUTHOR_LIST
(
        AUTHOR_ID CHAR(5) NOT NULL CONSTRAINT AUTHOR_LIST_AUTHOR_FK
        FOREIGN                 KEY(AUTHOR_ID) REFERENCES AUTHOR(AUTHOR_ID),

    BOOK_ID CHAR(5) NOT NULL,
    ROLE VARCHAR(15) CONSTRAINT AUTHOR_LIST_PK PRIMARY KEY
    (AUTHOR_ID,BOOK_ID) NOT NULL
)
```

```
CREATE TABLE BOOK
(
        BOOK_ID CHAR(3) CONSTRAINT BOOK_PK PRIMARY KEY(BOOK_ID) CON-
        STRAINT  AUTHOR_LIST_BOOK_FK FOREIGN KEY(BOOK_ID) REFERENCES
        BOOK (BOOK_ID) NOT NULL,
        TITLE VARCHAR(40) NOT NULL,
        EDITION INTEGER,
        YEAR INTEGER,
        PRICE DECIMAL(7 , 2),
        ISBN VARCHAR(20),
        PAGES INTEGER,
        AISLE VARCHAR(10),
        DESCRIPTION VARCHAR(100)
)

CREATE TABLE COPY
(
        COPY_ID CHAR(5) CONSTRAINT COPY_PK PRIMARY KEY(COPY_ID) NOT
NULL,
        BOOK_ID CHAR(5) CONSTRAINT COPY_BOOK_FK FOREIGN KEY(BOOK_ID)
        REFERENCES
     BOOK(BOOK_ID) NOT NULL,
        STATUS VARCHAR(10)
)



CREATE TABLE LOAN
(
        COPY_ID CHAR(5) CONSTRAINT LOAN_COPY_FK FOREIGN KEY(COPY_ID)
        REFERENCES
     COPY(COPY_ID) NOT NULL,
        BORROWER_ID CHAR(5) CONSTRAINT LOAN_BORROWER_FK FOREIGN
        KEY(BORROWER_ID) REFERENCES BORROWER (BORROWER_ID) NOT
        NULL,
        LOAN_DATE DATE NOT NULL,
        LOAN_DAYS INTEGER NOT NULL,
        RETURN_DATE DATE CONSTRAINT LOAN_PK PRIMARY KEY(COPY_ID, BOR-
        ROWER_ID)
)
```

```
CREATE TABLE BORROWER
(
        BORROWER_ID CHAR(5) NOT NULL CONSTRAINT BORROWER_PK PRIMARY
KEY
    (BORROWER_ID),
        LASTNAME VARCHAR(15) NOT NULL,
        FIRSTNAME VARCHAR(15) NOT NULL,
        EMAIL VARCHAR(40),
        PHONE VARCHAR(15),
        ADDRESS VARCHAR(60),
        CITY VARCHAR(15),
        COUNTRY CHAR(2)
)
```

สำหรับโปรแกรม InfoSphere Data Architect ซึ่งเป็นส่วนหนึ่งของผลิตภัณฑ์ของบริษัท ไอบีเอ็ม มีเครื่องมือที่จะให้เราแปลงจากแบบจำลองลอจิคัลให้เป็นแบบจำลองทางกายภาพโดยอัตโนมัติ และยังสร้างคำสั่ง SQL ที่เป็น DDL อีกด้วย

5.10 สรุป

ในบทนี้เราได้แสดงถึงความสามารถโดยรวมของภาษา SQL สำหรับคุณลักษณะเฉพาะพิเศษหรือฟังก์ชันเพิ่มเติม ที่เจ้าของผลิตภัณฑ์ต่างๆได้พัฒนาเพิ่มเติมซึ่งอาจจะแตกต่างกันออกไป ซึ่งคุณลักษณะเพิ่มเติมเหล่านี้เป็นการเพิ่มประสิทธิภาพของการออกแบบผลิตภัณฑ์และสถาปัตยกรรม เพื่อให้ผลิตภัณฑ์มีความแตกต่างและเป็นการปรับปรุงประสิทธิภาพของผลิตภัณฑ์ให้มีความรวดเร็วของการทำงานเพิ่มจากฟังก์ชันพื้นฐานของภาษา SQL มาตรฐาน ตัวอย่างหนึ่งได้แก่คุณสมบัติของการทำดัชนีในระบบฐานข้อมูลเป็นต้น ซึ่งโดยปกติแล้วลักษณะการทำงานพื้นฐานของดัชนีนั้นจะเหมือนกันในทุกระบบจัดการฐานข้อมูล อย่างไรก็ตาม ผลิตภัณฑ์ระบบจัดการฐานข้อมูลของแต่ละผู้ผลิตจะพยายามสร้างคุณลักษณะเพิ่มเติมเพื่อปรับปรุงวิธีการอ่าน/เขียนข้อมูลโดยอัลกอริทึมที่เป็นเทคนิคเฉพาะของตนเอง สำหรับข้อมูลรายละเอียดของคำสั่ง SQL ที่สำคัญ รวมถึงไวยากรณ์คำสั่ง สามารถดูรายละเอียดเพิ่มเติมได้จาก SQL Reference Guide [5.3]

5.11 แบบฝึกหัด

1. ให้ทำการสร้างตารางประกอบด้วยคอลัมน์ที่มีประเภทของข้อมูลเป็น เลขจำนวนเต็ม วันที่ และตัวอักษร โดยมีการกำหนดค่าเริ่มต้น (default value)
2. ทำการเพิ่มข้อมูล 10 แถวในตารางนี้ โดยใช้คำสั่ง SQL
3. ให้เขียนคำสั่ง SQL โดยการใช้ซับคิวรี พร้อมกับการทำอินเนอร์จอยน์
4. ให้เขียนคำสั่ง SQL โดยการใช้ correlated sub-query พร้อมกับGROUP BY
5. ให้เขียนคำสั่ง SQL โดยการใช้ฟังก์ชันการรวม (aggregate function) และกำหนดเงื่อนไข WHERE,

HAVING

6. ให้เขียนคำสั่ง SQL โดยการใช้ ORDER BY เพื่อการจัดลำดับมากกว่าหนึ่งคอลัมน์

5.12 คำถามท้ายบท

1. กลุ่มคำสั่งที่เป็นฟังก์ชันพื้นฐานสำหรับภาษา SQL คืออะไร
   A. การนิยามข้อมูล (Data definition)
   B. การเปลี่ยนแปลงข้อมูล (Data modification)
   C. การควบคุมสิทธิการเข้าถึงข้อมูล (Data control)
   D. ถูกทุกข้อ
   E. ไม่มีข้อใดถูกต้อง

2. ใครเป็นผู้ที่ให้กำเนิดภาษา SQL
   A. Raymond F. Boyce
   B. E. F. Codd
   C. Donald D. Chamberlin
   D. A และ C
   E. ไม่มีข้อใดถูกต้อง

3. ภาษา SQL ได้ถูกกำหนดให้เป็นภาษามาตรฐานโดยหน่วยงานใด
   A. American National Standards Institute
   B. Bureau of International Standards
   C. International Standards Organizations
   D. ถูกทุกข้อ
   E. ไม่มีข้อใดถูกต้อง

4. ข้อใดต่อไปนี้ เป็นการเปรียบเทียบระหว่างแนวคิดเชิงวัตถุไปสู่แนวคิดเชิงสัมพันธ์
   A. Name – Table Name
   B. Attribute – Column name
   C. Method – Stored procedure
   D. ถูกทุกข้อ
   E. ไม่มีข้อใดถูกต้อง

5. ฟังก์ชันใดต่อไปนี้ เป็นฟังก์ชันเฉพาะสำหรับการจัดการกับวันและเวลา

   A. Year

   B. Dayname

   C. Second

   D. ถูกทุกข้อ

   E. ไม่มีข้อใดถูกต้อง

6. การเรียงลำดับข้อมูลของผลลัพธ์ใน SQL โดยปริยาย คืออะไร

   A. จากน้อยไปหามาก (Ascending)

   B. จากมากไปหาน้อย (Descending)

   C. เรียงลำดับแบบสุ่ม (Random)

   D. ถูกทุกข้อ

   E. ไม่มีข้อใดถูกต้อง

7. จริงหรือไม่ที่คำสั่ง INSERT ไม่สามารถใช้ในการเพิ่มแถวหลายๆ แถวเข้าในตารางในคำสั่งเดียว

   A. จริง

   B. เท็จ

8. ข้อใดต่อไปนี้ จัดว่าเป็นประเภทของการเชื่อมโยงแบบอินเนอร์จอยน์ระหว่างตาราง

   A. Equi-join

   B. Natural-join

   C. Cross-join

   D. ถูกทุกข้อ

   E. ไม่มีข้อใดถูกต้อง

9. ข้อใดต่อไปนี้ จัดว่าเป็นประเภทของการเชื่อมโยงแบบเอาน์เตอร์จอยน์ระหว่างตาราง

   A. Left outer join

   B. Right outer join

C. Full outer join

D. ถูกทุกข้อ

E. ไม่มีข้อใดถูกต้อง

10. จริงหรือไม่ที่ตัวดำเนินการ Union จะเก็บค่าแถวที่ซ้ำกันในชุดผลลัพธ์

A. จริง

B. เท็จ

6

บทที่ 6 – Stored Procedure และฟังก์ชัน

Stored Procedure และฟังก์ชัน (Function) เป็นออบเจ็กต์ ของระบบฐานข้อมูล ซึ่งทำให้เราสามารถเขียนคำ สั่ง SQL ในการประมวลผล รวมถึงตรรกะทางธุรกิจ แล้วจัดเก็บในระบบฐานข้อมูลทำให้การเชื่อมต่อกันระหว่างโปรแกรมประยุกต์ต่าง ๆ กับระบบฐานข้อมูล ผ่านระบบเครือข่ายเป็นไปอย่างมีประสิทธิภาพ เพราะทำให้ปริมาณการส่งผ่านข้อมูล ไปมาในระบบเครือข่ายลดลง นอกจากนี้การจัดเก็บคำสั่งในการประมวลผลต่าง ๆ ในฐานข้อมูลไว้เป็นส่วนกลาง ทำให้โปรแกรมประยุกต์ อื่น ๆ สามารถเรียกใช้คำสั่งในการประมวลผลดังกล่าว ร่วมกันได้ ในบทนี้เราจะเรียนรู้เรื่องราวที่เกี่ยวข้องกับเนื้อหาต่อไปนี้

- วิธีการใช้งาน IBM Data Studio สำหรับการพัฒนาฟังก์ชัน และ Stored Procedure
- วิธีการทำงานกับฟังก์ชัน SQL
- วิธีการทำงานกับ Stored Procedure

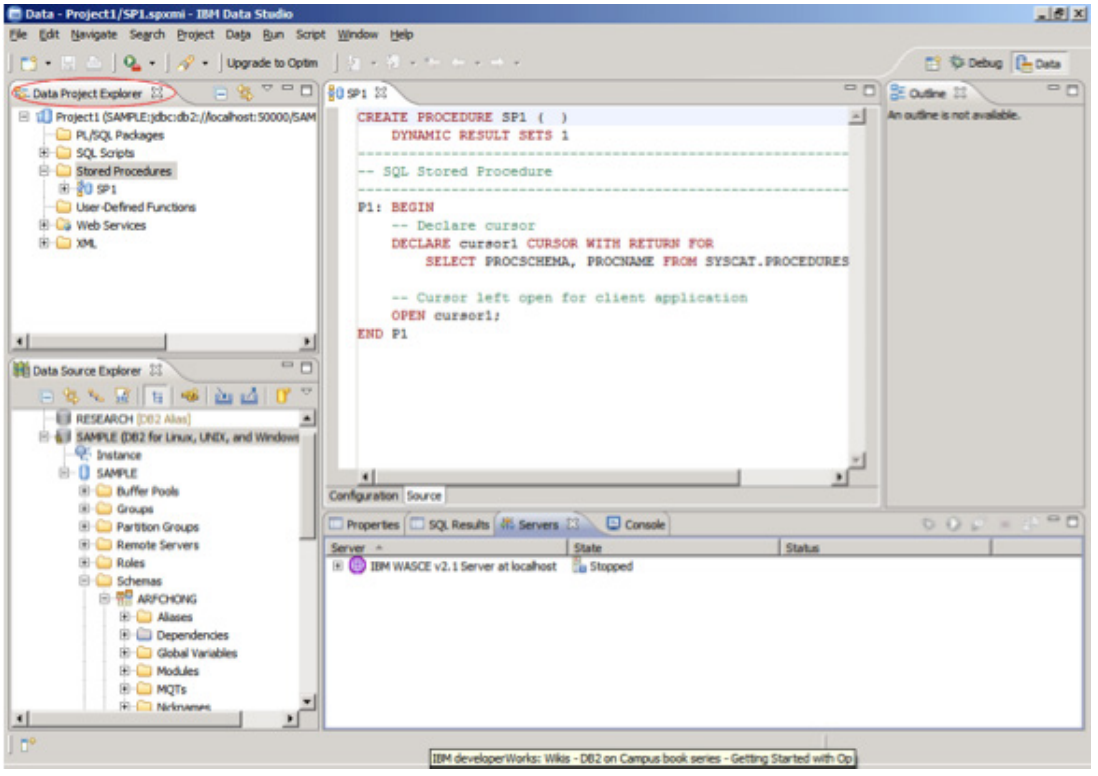
6.1 วิธีการใช้งาน IBM Data Studio

ใน ส่วนนี้ จะกล่าวถึง IBM Data Studio ที่ใช้ในการพัฒนา User-Defined Functions (UDFs) และ Stored Procedure ซึ่ง IBM Data Studio เป็นซอฟต์แวร์ที่ใช้สำหรับการพัฒนา และเป็นเครื่องมือสำหรับผู้ดูแลระบบ ได้แก่โปรแกรมมีชื่อว่า Eclipse และเป็นซอฟต์แวร์ที่ไม่ต้องเสียค่าใช้จ่าย ซึ่งซอฟต์แวร์นี้ไม่ได้มาพร้อมกับ DB2 แต่เป็นซอฟต์แวร์ที่แยกต่างหาก โดยผลิตภัณฑ์สามารถแบ่งตามคุณสมบัติของการทำงาน เป็น 2 ประเภทได้แก่

- IDE: เป็นผลิตภัณฑ์รุ่นที่อนุญาตให้สามารถทำงานร่วมกัน กับผลิตภัณฑ์ ในรูปแบบ Eclipse (shell sharing) อื่น ๆ ได้ เช่น InfoSphere Data Architect และ ผลิตภัณฑ์ต่าง ๆ ของ Rational ซึ่งผลิตภัณฑ์นี้ยังมีคุณสมบัติรองรับการทำงานของ Data Web Services
- Stand-Alone: ผลิตภัณฑ์รุ่นนี้จะมีฟังก์ชันการทำงานเกือบจะคล้ายคลึงกับผลิตภัณฑ์รุ่น IDE แต่จะไม่มีคุณสมบัติในการรองรับ Data Web Services และไม่สามารถทำงานรวมกัน กับผลิตภัณฑ์ ในรูปแบบ Eclipse (shell sharing) อื่น ๆ ได้ โดยผลิตภัณฑ์รุ่นนี้จะมีการใช้ทรัพยากรในหน่วยความจำสำรอง (ฮาร์ดดิสก์) ในปริมาณเพียงเล็กน้อยเท่านั้น

หมายเหตุ: สำหรับวิธีการใช้งาน IBM Data Studio อย่างละเอียด และครอบคลุม สามารถศึกษาได้จาก eBook Getting started with IBM Data Studio for DB2 ซึ่งเป็นส่วนหนึ่งของโครงการ DB2 on Campus free book

ในบทนี้จะใช้ผลิตภัณฑ์รุ่น Stand-Alone ซึ่งสามารถดาวน์โหลดได้ที่ ibm.com/db2/express ดังแสดงในรูปที่ 6.1 รูปแสดง IBM Data Studio 2.2

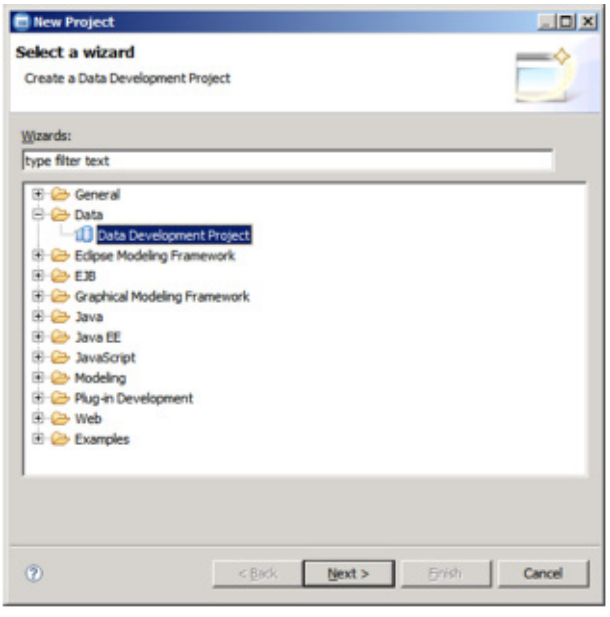


รูปที่ 6.1 IBM Data Studio 2.2

ในบทนี้จะมุ่งเน้นไปที่การใช้งานมุมมอง Data Project Explorer ซึ่งสามารถเห็นได้จากวงกลมสีแดงที่มุมซ้ายบนของรูปที่ 6.1 โดยจะเป็นมุมมองใน การพัฒนาโปรแกรมบนเครื่องแม่ข่าย ฐานข้อมูล

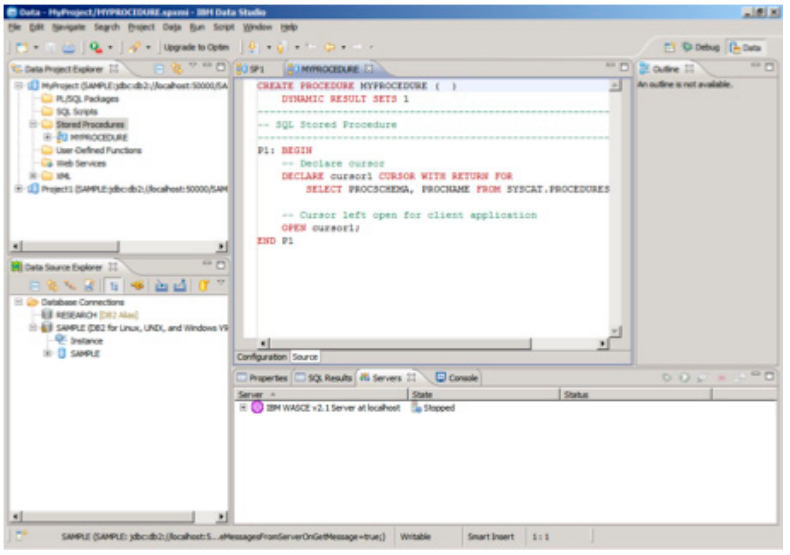
6.1.1 การสร้างโปรเจ็ค

ก่อนที่จะสามารถพัฒนา Stored Procedure หรือ UDFs ใน Data Studio ได้ จะต้องทำการสร้างโปรเจ็คขึ้นมาก่อน โดยวิธีการสร้างโปรเจ็ค ให้ไปที่ Data Studio เมนู แล้วคลิกเลือกเมนู File -> New -> Project และทำการเลือก Data Development Project ดังที่แสดงในรูปที่ 6.2



รูปที่ 6.2 การสร้าง Data Development Project

ทำตามขั้นตอนจากตัวช่วยสำหรับการสร้างโปรเจ็คโดยการกำหนดชื่อโปรเจ็ค และเลือกฐานข้อมูลที่ต้องการเชื่อมต่อ ถ้าต้องการเชื่อมต่อไปยังฐานข้อมูลใหม่ที่ไม่มีอยู่ในระบบ ให้คลิกที่ปุ่ม New ในส่วนของ Select Connection โดยจะปรากฏจอภาพดังรูปที่ 6.3

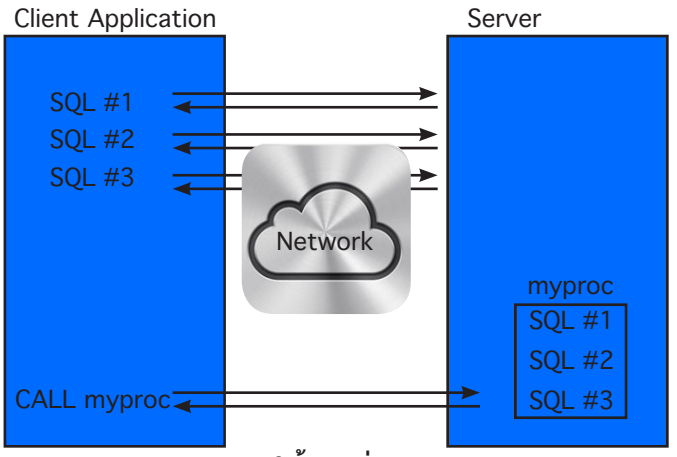


รูปที่ 6.3 แสดงพารามิเตอร์ในการเชื่อมต่อฐานข้อมูลใหม่

จากรูปที่ 6.3 เลือก DB2 for Linux, UNIX and Windows (DB2 LUW) ในส่วนของ Select a database manager ที่อยู่ทางด้านซ้ายของรูป สำหรับส่วน JDBC driver ค่า deault สำหรับ DB2 (LUW) คือ JDBC type 4 ซึ่งก็คือ IBM Data Server Driver for JDBC and SQLJ (JDBC 4.0) Default จากนั้นให้กำหนดข้อมูลต่าง ๆ ให้ครบถ้วนในส่วนของ ฟิลด์ต่างๆ ซึ่งในส่วนของ Host สามารถกำหนดเป็นหมายเลขไอพี หรือชื่อเครื่องแม่ขาย แต่ จากตัวอย่างในรูป IBM Data Studio และฐานข้อมูล DB2 ถูกติดตั้งอยู่บนเครื่องคอมพิวเตอร์เครื่องเดียวกัน ดังนั้นจึงสามารถใช้คำว่า localhost ในส่วนของ Host ได้ จากนั้นเพื่อเป็นการยืนยันการเชื่อมต่อฐานข้อมูลว่าสามารถทำการเชื่อมต่อได้ให้คลิกที่ปุ่ม Test Connection ที่ อยู่ทางด้านล่างซ้ายของรูป ถ้าสามารถเชื่อมต่อกับฐานข้อมูลได้แล้ว ให้คลิกที่ปุ่ม Finish จากนั้นชื่อของฐานข้อมูลจะถูกเพิ่มเข้าไปในรายการของการเชื่อมต่อ สำหรับโปรเจ็ค ให้เลือกฐานข้อมูลที่เพิ่งเพิ่มเข้าไปใหม่ แล้วคลิกที่ปุ่ม Finish โปรเจ็คที่เราสร้างขึ้นจะปรากฏในหน้าต่าง Data Project Explorer ซึ่ง ถ้า เราคลิกที่เครื่องหมาย “+” ที่ชื่อโปรเจ็คจะสามารถแสดงโฟลเดอร์ต่าง ๆ เช่น PL/SQL packages, SQL scripts, Stored Procedures เป็นต้น

6.2 การทำงานกับ Stored Procedures

Stored Procedure เป็นออบเจ็กต์ของฐานข้อมูลที่สามารถจัดเก็บคำสั่ง SQL ในการประมวลผลข้อมูลและตรรกะทางธุรกิจ ซึ่งจะช่วยเพิ่มประสิทธิภาพการทำงานโดยการลดปริมาณข้อมูลที่ส่งผ่านไปมาของระบบเครือข่าย ดังรูป 6.2 จะเป็นรูปแสดงการทำงานของ Stored Procedures ว่าทำงานอย่างไร



รูปที่ 6.2 แสดงการลดปริมาณการใช้เครือข่ายของ Stored Procedures

จากรูปที่ 6.2 จะเห็นว่ามีการประมวลผลคำสั่ง SQL หลายครั้งโดยที่ในการประมวลผลแต่ละครั้งเครื่องลูกข่ายจะส่งข้อมูลไปยังเครื่องแม่ข่าย และหลังสิ้นสุดการประมวลผลเครื่องแม่ข่ายจะทำการส่งผลลัพธ์ของการประมวลผลกลับมายังเครื่องลูกข่าย ถ้ามีการประมวลผลคำสั่ง SQL หลายๆคำสั่ง ปริมาณการใช้งานเครือข่ายจะเพิ่มขึ้นอย่างมาก แต่จากทางด้านล่างของรูป จะ เห็นได้ว่าปริมาณการใช้งานเครือข่ายมีเพียงเล็กน้อย โดยวิธีที่สองนี้เป็นการเรียกใช้งาน Stored Procedure ที่ชื่อว่า myproc ที่ถูกจัดเก็บไว้บนเครื่องแม่ข่ายซึ่งบรรจุคำสั่ง SQL ในการประมวลผลเหมือนที่ถูกประมวลผลในเครื่องลูกข่าย (ทางด้านซ้ายมือ) ซึ่งวิธีที่สองนี้จะ ใช้คำสั่ง Call ในการเรียกใช้ Stored Procedure ซึ่งจะมีประสิทธิภาพในการทำงานมากกว่า โดยปริมาณการใช้งานเครือข่ายจะถูกใช้งานเพียงครั้งเดียวในการเรียกใช้งาน Stored Procedure และถูกใช้งานเพียงครั้งเดียวในการส่งผลลัพธ์ในการประมวลผลคืนกลับสู่เครื่องลูกข่าย

นอกจากนี้ Stored Procedures ยังมีประโยชน์ในด้านการรักษาความปลอดภัยของฐานข้อมูลอีกด้วย ตัวอย่าง เช่น สามารถอนุญาตให้ผู้ใช้เข้าถึงตารางหรือวิวผ่านทาง Stored Procedures ได้เท่านั้น เป็นการช่วยทำให้สามารถป้องกันการเข้าถึงข้อมูลจากผู้ใช้ที่ไม่มีสิทธิในการเข้าถึงข้อมูลนั้น เพราะว่าผู้ใช้ไม่จำเป็นที่จะต้องมีสิทธิในการใช้งานตาราง หรือวิวเพียงแค่ มีสิทธิในการเรียกใช้งาน Stored Procedures

6.2.1 ประเภทของ Procedures

ประเภทของ Stored Procedures จะแบ่งออกได้เป็นสองประเภท: SQL Procedures และ External Procedures สำหรับ SQL Procedures จะเขียนคำสั่งในการประมวลผลด้วยคำสั่ง SQL ส่วน External Procedures จะเขียนคำสั่งในการประมวลผลด้วย host language แต่อย่างไรก็ตามเราสามารถพิจารณาความแตกต่างที่สำคัญได้จากการทำงานและขั้นตอนการพัฒนา
SQL Procedures และ External Procedures จะประกอบด้วยส่วนรายละเอียดของ Procedure (procedure definition) และคำสั่งในการประมวลผล ซึ่งในส่วนของรายละเอียดของ Procedure ทั้ง SQL Procedure และ External Procedure จะมีข้อกำหนดดังต่อไปนี้

- ชื่อของ Procedure
- พารามิเตอร์ที่ใช้ส่งข้อมูลเข้าและออก Procedure
- ภาษาที่ใช้เขียน Procedure ซึ่งสำหรับ SQL Procedure จะเป็นภาษา SQL
- ข้อมูลที่จะต้องใช้เมื่อ Procedure ถูกเรียกใช้งาน เช่น runtime options เวลาที่ Procedure สามารถใช้ในการประมวลผลและผลลัพธ์ที่ถูกส่งคืนกลับมาจาก Procedure

ตัวอย่างต่อไปนี้แสดงข้อกำหนดสำหรับ SQL Procedure

```
CREATE PROCEDURE UPDATESALARY          (1)
  (IN EMPNUMBR CHAR(10),               (2)
```

```
IN RATE DECIMAL(6,2))
LANGUAGE SQL                    (3)
UPDATE EMP                      (4)
SET SALARY = SALARY * RATE
WHERE EMPNO = EMPNUMBR
```

จากตัวอย่าง:

(1) ชื่อของ Procedure คือ UPDATESALARY

(2) มีพารามิเตอร์สองตัวคือ EMPNUMBR เป็นข้อมูลประเภท CHAR(10) และ RATE เป็นข้อมูลประเภท DECIMAL(6,2) ทั้งคู่เป็นพารามิเตอร์ประเภทส่งเข้า

(3) LANGUAGE SQL เป็นส่วนที่บอกว่า Procedure เป็นประเภท SQL Procedure

(4) เป็นส่วนคำสั่งในการประมวลผล (procedure body)ซึ่งในตัวอย่างจะเป็นคำสั่ง UPDATE ของภาษา SQL ซึ่งทำการปรับปรุงข้อมูลของแถวในตาราง employee

ตัวอย่างต่อไปนี้จะเป็นการแสดงข้อกำหนดสำหรับ External Stored Procedure ซึ่งมีคำสั่งในการประมวลผลเป็นคำสั่งในภาษา COBOL เป็น Stored Procedure ที่ทำการปรับปรุงเงินเดือนของพนักงานโดยการเรียกใช้โปรแกรม UPDSAL

```
CREATE PROCEDURE UPDATESALARY     (1)
 (IN EMPNUMBR CHAR(10),           (2)
 IN RATE DECIMAL(6,2))
 LANGUAGE COBOL                   (3)
 EXTERNAL NAME UPDSAL;            (4)
```

จากตัวอย่าง:

(1) ชื่อของ Procedure คือ UPDATESALARY

(2) มีพารามิเตอร์สองตัวคือ EMPNUMBR เป็นข้อมูลประเภท CHAR(10) และ RATE เป็นข้อมูลประเภท DECIMAL(6,2) ทั้งคู่เป็นพารามิเตอร์ประเภทส่งเข้า

(3) LANGUAGE COBOL เป็นส่วนที่บอกว่า Procedure เป็นประเภท External Procedure และคำสั่งที่ใช้ประมวลผลใน Procedure จะเป็นคำสั่งในภาษา COBOL โดยจะไปเรียกโปรแกรม COBOL ที่มีการเขียนเตรียมไว้

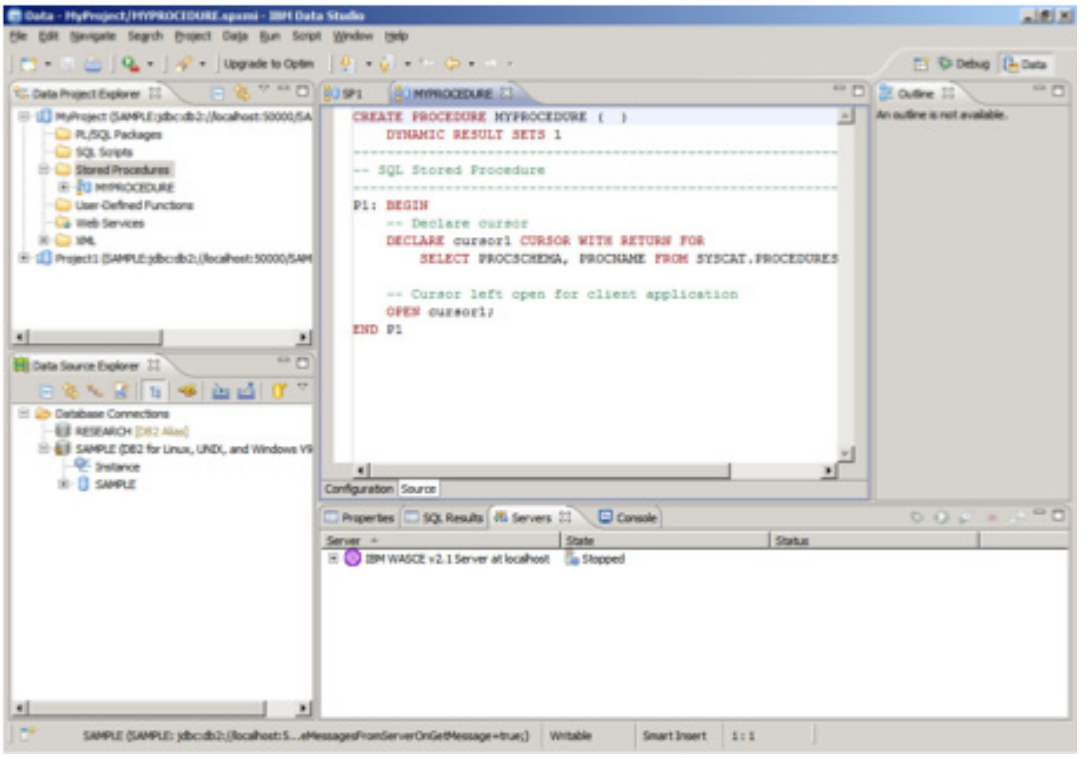
(4) เป็นชื่อโมดูลที่จัดเก็บคำสั่งในการประมวลผลที่เป็นภาษา COBOL ในตัวอย่างก็คือ UPDSAL

6.2.2 การสร้าง Stored Procedure

การสร้าง Stored Procedure ด้วยภาษา Java, PL/SQL หรือ SQL PL ใน Data Studio จะต้องทำตามขั้นตอนต่อไปนี้ การสร้าง Stored Procedures ในภาษาอื่น ๆ ไม่สามารถสร้างได้จาก Data Studio ในขั้นตอนการสร้างต่อไปนี้ได้เลือกการสร้าง Stored Procedure แบบ SQL หรือ SQL PL โดยสามารถประยุกต์ใช้ขั้นตอนแบบเดียวกันในการสร้าง Stored Procedure ในแบบภาษา Java และ PL/SQL

<u>ขั้นตอนที่ 1: การเขียนหรือสร้างคำสั่ง Stored Procedure</u>

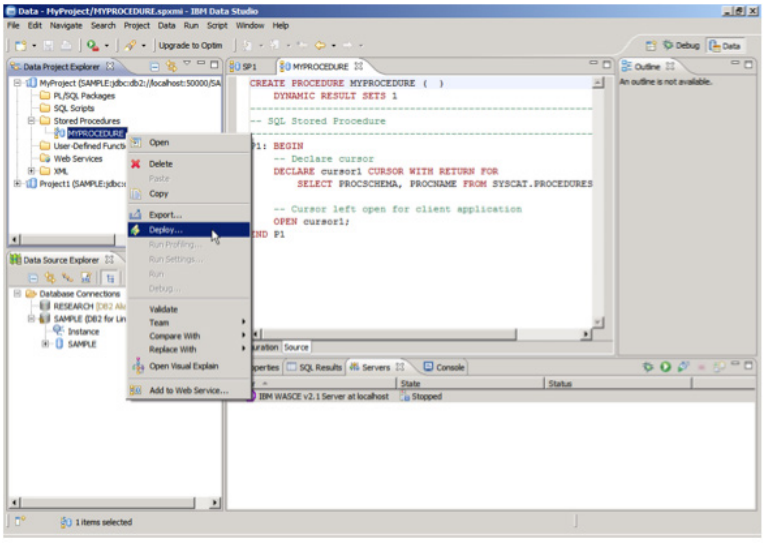
เมื่อต้องการสร้าง Stored Procedure ให้ไปคลิกขวาที่โฟลเดอร์ Stored Procedures และเลือก New -> Stored Procedure ทำการกำหนดข้อมูลที่ปรากฏในตัวช่วยสร้าง Stored Procedure เช่น ชื่อโปรเจ็ค ชื่อ Stored Procedure ภาษาที่จะใช้ และคำสั่ง SQL ที่จะใช้ใน Procedure โดย Data Studio จะมีตัวอย่างของคำสั่ง SQLให้ จากนั้นให้เลือกค่า default สำหรับหน้าจออื่นๆ แล้วให้คลิกที่ Finish และ Stored Procedure จะถูกสร้างขึ้นมาโดยใช้รูปแบบมาตรฐาน(template code) และตัวอย่างคำสั่ง SQL ดังที่แสดงในรูป 6.3



รูปที่ 6.3 ตัวอย่าง Stored Procedure

ในรูปที่ 6.3 เป็นคำสั่งของ Stored Procedure ตัวอย่างที่ชื่อ MYPROCEDURE ที่ถูกสร้างขึ้นมา โดยเราสามารถแทนที่คำสั่งทั้งหมดด้วยคำสั่งที่เราต้องการได้ เพื่อความง่ายในบทนี้เราจะใช้ตัวอย่างข้างบนเป็นเหมือนกับ Stored Procedure ที่เราเขียนขึ้นมา

<u>ขั้นตอนที่ 2: การติดตั้ง Stored Procedure</u>
หลังจากที่ Stored Procedure ได้ถูกสร้างขึ้น เมื่อจะทำการติดตั้งเพื่อนำไปใช้งาน ให้ไปคลิกขวาที่ชื่อ Stored Procedure ที่ต้องการในหน้าต่าง Data Project Explorer แล้วเลือก Deploy หลังจากเลือก Deploy แล้ว ระบบจะทำการประมวลผลคำสั่ง CREATE PROCEDURE ทำการ compile แล้วนำ Stored Procedure ไปจัดเก็บไว้ในฐานข้อมูล ดังรูป 6.4

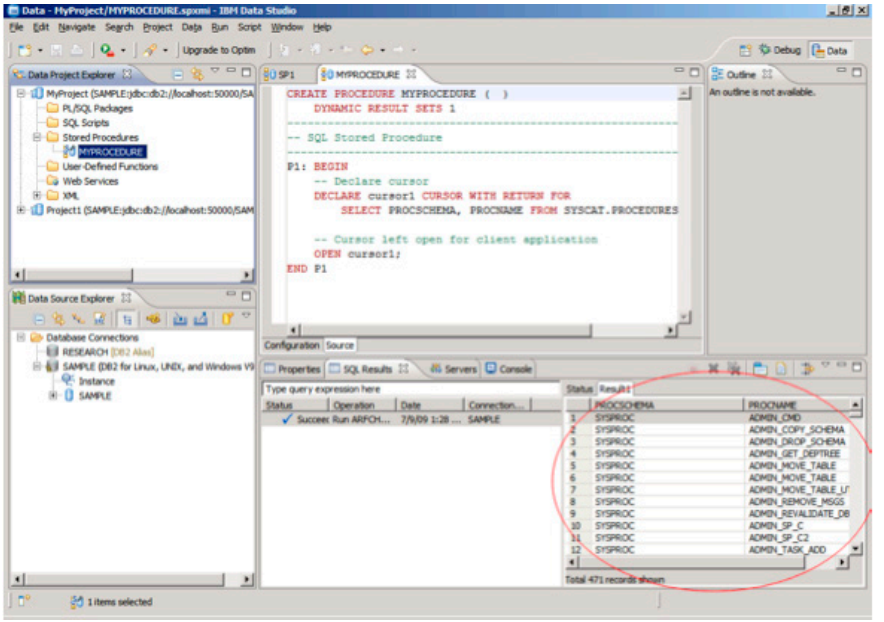


รูปที่ 6.4 การติดตั้ง Stored Procedure

หลังจากคลิก Deploy จะปรากฏหน้าจอสำหรับกำหนดค่าการติดตั้ง ให้เลือกเป็นค่า default แล้วให้คลิก Finish

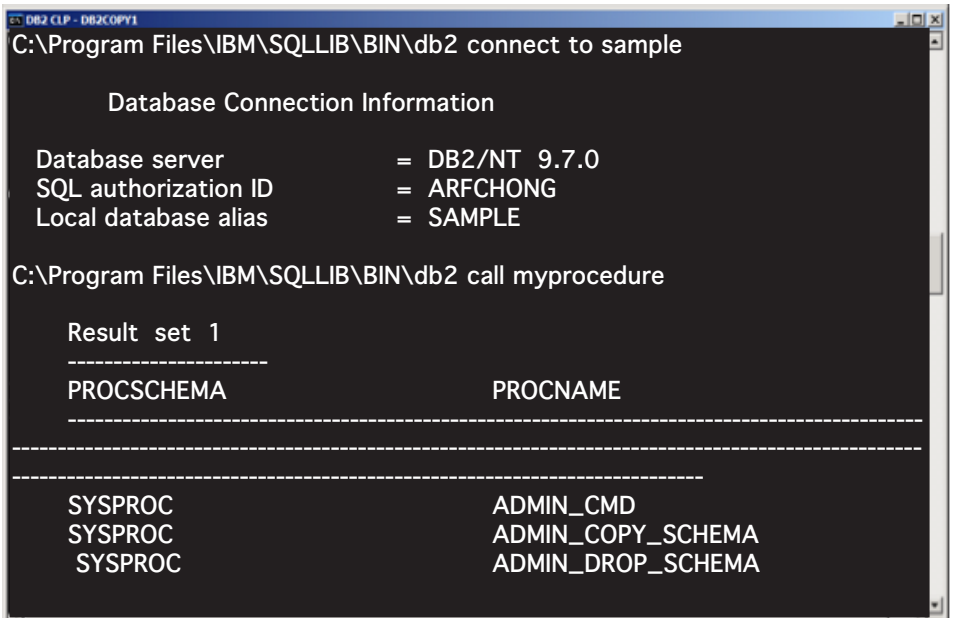
ขั้นตอนที่ 4: การเรียกใช้งาน Stored Procedure

หลังจากทำการติดตั้ง Stored Procedure เป็นที่เรียบร้อยแล้ว สามารถที่จะเรียกใช้งานได้โดยการคลิกขวาที่ชื่อ Stored Procedure และเลือก Run ผลลัพธ์ที่ได้จากการประมวลผลจะปรากฏที่หน้าต่าง Results ที่ด้านล่างทางขวาของ Data Studio ดังรูป 6.5



รูปที่ 6.5 ผลลัพธ์จากการเรียกใช้ Stored Procedure

สำหรับการเรียกใช้งาน Stored Procedure จาก DB2 Command Window หรือ Command Editor สามารถเรียกใช้งานได้โดยคำสั่ง CALL <ชื่อ Procedure> แต่จะต้องทำการเชื่อมต่อกับฐานข้อมูลที่จัดเก็บ Store Procedure นั้น ๆ ก่อน ดังรูป 6.6



รูปที่ 6.6 การเรียกใช้งาน Stored Procedure จาก DB2 Command Window

การเรียกใช้งาน Stored Procedure จากภาษาอื่น ๆ เช่น Java, C, Visual Basic สามารถทำได้เหมือนการเรียกใช้จาก DB2 Command Window แต่จะต้องเรียกใช้งานให้ถูกต้องตามหลักการของแต่ละภาษา

6.2.3 การแก้ไขและลบ Stored Procedure

การแก้ไข Stored Procedure ที่ถูกจัดเก็บไว้มีสองวิธีได้แก่:

1. ลบ และสร้าง Procedure ขึ้นมาใหม่อีกครั้งหนึ่ง
2. ใช้คำสั่ง 'CREATE OR REPLACE PROCEDURE' แทนที่การใช้คำสั่ง 'CREATE PROCEDURE'

นอกจากนี้ยังมีคำสั่ง ALTER PROCEDURE สำหรับการแก้ไขอยู่อีก แต่ก็สามารถใช้ได้เมื่อต้องการเปลี่ยนแปลงคุณสมบัติเฉพาะของ Procedure เท่านั้น
การลบ Procedure จะต้องระบุชื่อของ Procedure ให้ถูกต้องครบถ้วน (fully qualified name) และใช้คำสั่ง 'DROP PROCEDURE' ดังที่แสดงไว้ในตัวอย่างนี้

```
drop procedure myschema.EMPLOYEE_COUNT
```

การแก้ไข Procedure จะใช้คำสั่ง 'CREATE OR REPLACE PROCEDURE' มากกว่าที่จะลบ และสร้าง Procedure ขึ้นมาใหม่ ทั้งนี้เป็นเพราะว่าการลบ Procedure อาจจะทำให้เกิดผลกระทบกับ ออบเจ็กต์อื่นๆ ที่เกี่ยวข้องกับ Procedure ที่ถูกลบ แต่คำสั่ง CREATE OR REPLACE PROCEDURE จะไม่ทำให้เกิดผลกระทบ ดังกล่าว

6.3 การทำงานกับฟังก์ชัน

SQL ฟังก์ชันเป็นคำสั่ง หรือเป็นวิธีที่รับค่าพารามิเตอร์หรือไม่รับค่าพารามิเตอร์ก็ได้ และจะคืนค่าผลลัพธ์กลับมาเพียงแค่ค่าเดียวเท่านั้น ฐานข้อมูลทั้งหมดที่ปรากฏอยู่ในท้องตลาดส่วนใหญ่จะมีฟังก์ชันที่เป็นมาตราฐานอยู่แล้ว เช่นฟังก์ชันสำหรับการคำนวณทางคณิตศาสตร์, ฟังก์ชันในการจัดการข้อมูลประเภทตัวอักษร และฟังก์ชันในการจัดการเรื่องอื่นๆ ซึ่งผู้ใช้งานสามารถสร้างฟังก์ชันของตนเองได้ ซึ่งจะถูกเรียกว่า User-Defined Functions เพื่อเพิ่มเติมจาก Functions Library พื้นฐานที่อาจจะไม่ตอบสนองความต้องการได้ทั้งหมด

6.3.1 ประเภทของฟังก์ชัน

ฟังก์ชันสามารถแบ่งประเภทได้ตามลักษณะการทำงาน และข้อมูลที่นำเข้าได้ดังต่อไปนี้:

- Scalar (ตัวแปรเดี่ยว)
  - Aggregate (การคำนวณผลรวม)
  - String (สตริงข้อความ)
- Table (ตาราง)

นอกจากประเภทที่กล่าวมาข้างต้นแล้วยังมีฟังก์ชันที่เป็นฟังก์ชันพื้นฐานที่มากับตัวฐานข้อมูล และฟังก์ชันที่ผู้ใช้งานสร้างขึ้นเอง (UDFs) โดย UDF จะใช้ภาษา SQL ในการพัฒนา ซึ่งเป็นโปรแกรมที่มีขนาดเล็กคล้าย ๆ กับโปรแกรมย่อย หรือฟังก์ชัน host language อย่างไรก็ตาม User-Defined Function มักจะเป็นทางเลือกที่ดีกว่าสำหรับการเขียนโปรแกรมด้วยภาษา SQL เนื่องจากสามารถเรียกใช้งานได้จากคำสั่ง SQL ใน DB2 สามารถสร้าง Scalar หรือ Table UDFs โดยใช้ภาษา SQL PL, PL/SQL, C/C++, Java, CLR (Common Language Runtime) และ OLE (Object Linking and Embedding) ได้

6.3.1.1 ฟังก์ชันแบบตัวแปร (Scalar Function)

ฟังก์ชันแบบตัวแปรเป็นฟังก์ชันที่สามารถรับค่าพารามิเตอร์ได้มากกว่าหนึ่ง และคืนค่าผลลัพธ์ออกมาเพียงค่าเดียว ฟังก์ชันแบบตัวแปรส่วนมากจะใช้สำหรับการจัดการตัวอักษร หรือการคำนวณพื้นฐานทางคณิตศาสตร์ด้วยคำสั่ง SQL ซึ่งฟังก์ชันแบบตัวแปรจะไม่สามารถใช้คำสั่ง SQL ประเภท INSERT, UPDATE และ DELETE ได้ ตัวอย่างเช่น ฟังก์ชัน LENGTH ซึ่งเป็นฟังก์ชันแบบตัวแปรพื้นฐานที่มากับฐานข้อมูลซึ่งจะคืนค่าความยาวของตัวอักษรทั้งหมด ดังตัวอย่างต่อไปนี้

```
SELECT length('Mary')
   FROM sysIBM.sysdummy1
```

คำสั่ง SQL นี้ จะทำการประมวลผลโดยที่เราจะต้องเชื่อมต่อกับฐานข้อมูล DB2 เสียก่อนซึ่งหลังจากประมวลผล

แล้วจะคืนค่า 4 ซึ่งเป็นจำนวนตัวอักษรทั้งหมดของคำว่า 'Mary' กลับมา

ฟังก์ชันแบบตัวแปรสามารถถูกอ้างอิง หรือถูกเรียกใช้งานจากคำสั่ง SQL ทั้งในส่วน SELECT-LIST หรือส่วนของ FROM ได้ เช่น

```
SELECT EMPNO, LASTNAME, YEAR(CURRENT DATE - BIRTHDATE)
    FROM EMPLOYEE
    WHERE WORKDEPT = 'D01'
```

จากตัวอย่างข้างบนฟังก์ชันที่ชื่อ YEAR จะถูกเรียกใช้เพื่อให้คืนค่าของจำนวนปีที่เกิดจากการนำเอา "CURRENT DATE" ลบด้วย "BIRTHDATE"

ฟังก์ชันแบบตัวแปร ที่มากับฐานข้อมูลจะทำงานได้อย่างมีประสิทธิภาพเนื่องจากตรรกะของฟังก์ชันจะถูกประมวลผลพร้อมกันกับคำสั่ง SQL ที่เรียกใช้ฟังก์ชันดังกล่าว บนเครื่องแม่ข่ายฐานข้อมูล ดังนั้นเมื่อใช้งานฟังก์ชันในส่วนที่เป็น where clause (predicates) จึงสามารถเพิ่มขีดความสามารถในการสอบถาม (query) ข้อมูลได้ดียิ่งขึ้น เพราะเมื่อฟังก์ชันแบบตัวแปรทำงานจะสามารถทำหน้าที่เป็นตัวกรองข้อมูล ทำให้จำกัดจำนวนแถวของข้อมูลที่ต้องถูกส่งกลับไปยังเครื่องลูกข่าย

ฟังก์ชันแบบการคำนวณผลรวม (Aggregate functions)

ฟังก์ชันแบบการคำนวณผลรวมเป็นฟังก์ชันที่สามารถรับค่าพารามิเตอร์ได้มากกว่าหนึ่ง และคืนค่ากลับได้เพียงค่าเดียว ตัวอย่างเช่น AVG(COL_NAME) จะคืนค่าของผลรวมของคอลัมน์ COL_NAME ที่ส่งค่าเข้าไปในฟังก์ชัน

ฟังก์ชันของตัวอักษร (Strings functions)

ฟังก์ชันของตัวอักษรเป็นฟังก์ชันที่สามารถรับค่าพารามิเตอร์ที่เป็นตัวอักษรได้มากกว่าหนึ่ง หรือพารามิเตอร์ที่เป็นตัวเลขจำนวนเต็มบวก หรือจำนวนเต็มศูนย์ ตัวอย่างเช่น SUBSTR('abcdefghi',3,4) มีพารามิเตอร์สามตัวด้วยกันคือ (ตัวอักษรที่ต้องการประมวลผล, ตำแหน่งที่เริ่มต้น, และจำนวนตัวอักษรนับจากตำแหน่งที่เริ่มต้น) และจะส่งค่า 'cdef' ในการตัดตัวอักษรกลับคืนมา

ฟังก์ชันแบบตาราง (Table functions)

ฟังก์ชันแบบตารางจะเป็นฟังก์ชันที่ส่งค่าเป็นตารางกลับคืน ซึ่งสามารถเรียกใช้งานได้จากส่วนของ FROM โดยที่ฟังก์ชันแบบตารางจะแตกต่างจากฟังก์ชันแบบตัวแปรคือสามารถเรียกใช้คำสั่งประเภท INSERT, UPDATE และ DELETE ในภาษา SQL ได้ ตัวอย่างฟังก์ชันแบบตารางพื้นฐานที่มากับ DB2 เช่น SNAPSHOT_DYN_SQL() และ MQREADALL() ซึ่งฟังก์ชันแบบตารางจะทำงานคล้ายกับ วิว (Views) แต่เนื่องจากสามารถทำการปรับปรุงแก้ไขข้อมูลได้ (INSERT, UPDATE, และ DELETE) ทำให้ฟังก์ชันแบบตารางมีประสิทธิภาพในการทำงานมาก

ตัวอย่างต่อไปนี้เป็นตัวอย่างของฟังก์ชันแบบตารางที่ระบุชุดข้อมูลของแผนกของพนักงาน

```
CREATE FUNCTION getEnumEmployee(p_dept VARCHAR(3))
RETURNS TABLE
  (empno CHAR(6),
   lastname VARCHAR(15),
   firstnme VARCHAR(12))
SPECIFIC getEnumEmployee
RETURN
  SELECT e.empno, e.lastname, e.firstnme
    FROM employee e
    WHERE e.workdept=p_dept
```

6.3.2 การสร้างฟังก์ชัน

การสร้างฟังก์ชันจะคล้ายกับการสร้าง Stored Procedure ซึ่งสามารถใช้ IBM Data Studio ในการสร้าง User-Defined Function ได้ต่างกันตรงที่จะคลิกขวาที่โฟลเดอร์ user-defined functions แทนโฟลเดอร์ Stored procedures จากนั้นก็ทำตามขั้นตอนเหมือนกับการสร้าง Stored Procedure คำสั่ง CREATE FUNCTION จะใช้ในการสร้าง user-defined function หรือแม่แบบของฟังก์ชัน (function template)

ที่เครื่องแม่ข่ายฐานข้อมูล จะมีไวยากรณ์ของคำสั่งดังนี้

```
>>-CREATE--+-----------------+--FUNCTION--function-name-------------->
           '-OR REPLACE-'

        .-IN------.
>--(--+---------+--parameter-name--| data-type1 |--+-------------------+--|-)-->
      |         |                                  '-| default-clause  |-'
      +-OUT-----+
      '-INOUT---'
>-- RETURNS--+-| data-type2 |-------------+--| option-list |----->
             '-+-ROW-----+-----|   column-list   |-'
               '--TABLE---'

>--| SQL-function-body |---------------------------------------><
```

สำหรับตัวอย่างต่อไปนี้เป็นการสร้างฟังก์ชันโดยใช้ภาษา SQL PL โดยเป็นการคืนค่าเป็นตัวอักษรกลับมา

```
CREATE FUNCTION REVERSE(INSTR VARCHAR(40))
    RETURNS VARCHAR(40)
    DETERMINISTIC NO EXTERNAL ACTION CONTAINS SQL
    BEGIN ATOMIC
     DECLARE REVSTR, RESTSTR VARCHAR(40) DEFAULT '';
     DECLARE LEN INT;
     IF INSTR IS NULL THEN
       RETURN NULL;
     END IF;
     SET (RESTSTR, LEN) = (INSTR, LENGTH(INSTR));
     WHILE LEN > 0 DO
      SET (REVSTR, RESTSTR, LEN)
        = (SUBSTR(RESTSTR, 1, 1) CONCAT REVSTR,
           SUBSTR(RESTSTR, 2, LEN - 1),
           LEN - 1);
     END WHILE;
     RETURN REVSTR;
END
@
```

สำหรับข้อมูลที่ครอบคลุมเกี่ยวกับไวยากรณ์ของการสร้างฟังก์ชันสามารถอ้างอิงได้จาก DB2 v9.7 Information Center

6.3.3 การเรียกใช้งานฟังก์ชัน

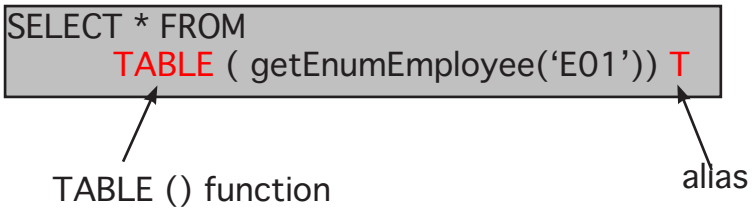
ฟังก์ชันสามารถเรียกใช้งานได้ในทุก ๆ คำสั่ง SQL หรือในการดำเนินการกับข้อมูลต่างๆ โดยฟังก์ชันไม่สามารถเรียกใช้งานผ่านทางคำสั่ง 'CALL' ได้ โดยทั่วไปแล้วฟังก์ชันมักจะถูกเรียกใช้งานผ่านทางคำสั่ง SELECT หรือ VALUES ตัวอย่างเช่น ฟังก์ชัน REVERSE ที่แสดงไว้ก่อนหน้านี้

```
SELECT reverse(col_name) from myschema.mytable

                OR
VALUES reverse('abcd')
```

ในกรณีของฟังก์ชันแบบตาราง สามารถเรียกใช้งาน ในส่วนของ FROM ในคำสั่ง SQL โดยจะทำการส่งค่าคืนมาเป็นตาราง การใช้คำสั่ง TABLE() โดยปกติแล้ว จะต้องทำการกำหนดนามแฝง (alias) ทุกครั้ง ตัวอย่างเช่น

การเรียกใช้งานฟังก์ชันแบบตารางที่ชื่อ getEnumEmployee โดยการใช้คำสั่ง SELECT ในรูปที่ 6.1 ด้านล่าง



รูปที่ 6.7 การเรียกใช้งานฟังก์ชันแบบตาราง

6.3.4 การแก้ไข และการลบฟังก์ชัน

การแก้ไข user-defined function ที่มีอยู่แล้วมีสองวิธี ดังนี้

1. ทำการลบฟังก์ชัน และสร้างฟังก์ชันขึ้นมาใหม่อีกครั้ง
2. ใช้คำสั่ง 'CREATE OR REPLACE FUNCTION' แทนการใช้คำสั่ง 'CREATE FUNCTION' และยังมีคำสั่ง ALTER FUNCTION สำหรับการแก้ไขอยู่อีก แต่ก็สามารถใช้ได้เมื่อต้องการเปลี่ยนแปลงคุณสมบัติเฉพาะของ FUNCTION เท่านั้น

การลบ Function จะต้องใช้ชื่อของ Function ให้ถูกต้องครบถ้วน (fully qualified name) และใช้คำสั่ง 'DROP FUNCTION' ดังที่แสดงไว้ในตัวอย่างข้างล่าง

```
DROP FUNCTION myschema.reverse
```

การแก้ไข Function จะใช้คำสั่ง 'CREATE OR REPLACE FUNCTION' มากกว่าที่จะลบ และสร้าง Function ขึ้นมาใหม่ ทั้งนี้เป็นเพราะว่าการลบ Function อาจจะทำให้เกิดผลกระทบกับออบเจ็กต์ อื่นๆที่เกี่ยวข้องกันกับ Function ที่ถูกลบอยู่ ซึ่งคำสั่ง CREATE OR REPLACE FUNCTION จะไม่ส่งผลกระทบดังกล่าว

6.4 บทสรุป

Stored Procedures และฟังก์ชันเป็นเครื่องมือที่มีความสำคัญ และมีประโยชน์มากใน กรณีที่เราต้องการการทำงานบางอย่างซึ่งไม่ได้มากับฐานข้อมูล ซึ่งผู้ใช้งาน Stored Procedures และฟังก์ชันจะต้องมีสิทธิ 'EXECUTE' จึงจะสามารถทำงานกับ Stored Procedure หรือ ฟังก์ชันได้ ซึ่งการใช้งาน Stored Procedure และฟังก์ชัน ทำให้ปริมาณการใช้งานเครือข่ายลดลงส่งผลให้การทำงานมีประสิทธิภาพมากขึ้นโดยการเก็บชุดคำสั่งในการประมวลผลไว้ที่ฐานข้อมูล รวมถึงทำให้ฐานข้อมูลมีความปลอดภัยมากขึ้นอีกด้วย

6.5 แบบฝึกหัด

1. สร้าง SQL PL Stored Procedure โดยการใช้ IBM Data Studio โดย ให้ใช้ฐานข้อมูล SAMPLE ของ DB2 โดย procedure จะต้องรับค่ารหัสพนักงานเป็นพารามิเตอร์ และทำการคืนค่าคอลัมน์ของพนักงานทั้งหมดของรหัสพนักงานนั้น ๆ จากตาราง EMPLOYEE
2. สร้าง UDF ที่ให้ผลลัพธ์เหมือนกับแบบฝึกหัดข้อที่ (1)
3. สร้าง SQL Function ซึ่งไม่มีการรับค่าพารามิเตอร์ใด ๆ ทั้งสิ้น และมีการคืนค่าชื่อของวันเป็นตัวอักษร (เช่น Monday, Tuesday และอื่น ๆ) โดยดึงข้อมูลมาจากวันที่ปัจจุบันของระบบ
4. สร้าง SQL Function ซึ่งรับค่าพารามิเตอร์ที่เป็นวันที่ และมีการคืนค่าชื่อของวันเป็นตัวอักษร (เช่น Monday, Tuesday และอื่น ๆ) จากพารามิเตอร์ที่รับค่าเข้าไป
5. สร้าง procedure ที่ไม่รับค่าพารามิเตอร์ใด ๆ ทั้งสิ้น และมีการคืนค่าจำนวนตารางทั้งหมดในฐานข้อมูล
6. สร้าง procedure ที่รับค่าพารามิเตอร์เป็นชื่อตาราง และมีการคืนค่าจำนวนแถวของข้อมูลทั้งหมดในตาราง

6.6 คำถามท้ายบท

1. ฟังก์ชันใดต่อไปนี้ เป็นฟังก์ชันที่มีอยู่จริง
   A. Select
   B. Update
   C. Count
   D. Delete
   E. ไม่มีข้อใดถูก

2. ฟังก์ชันใดต่อไปนี้เป็นฟังก์ชันที่ไม่มีอยู่จริง
   A. avg
   B. count
   C. insert
   D. substr
   E. ไม่มีข้อใดถูก

3. ฟังก์ชันใดต่อไปนี้เป็นฟังก์ชันแบบตัวแปร
   A. avg
   B. count
   C. max
   D. substr
   E. ถูกทุกข้อ

4. ฟังก์ชันใดต่อไปนี้ไม่เป็นฟังก์ชันแบบตัวแปร
   A. trim
   B. upper
   C. min
   D. substr
   E. ไม่มีข้อใดถูก

5. ฟังก์ชันใดต่อไปนี้เป็นฟังก์ชันแบบผลรวม
   A. lcase
   B. year
   C. max
   D. substr
   E. ไม่มีข้อใดถูก

6. ฟังก์ชันใดต่อไปนี้ไม่เป็นฟังก์ชันแบบผลรวม
   A. sum
   B. min
   C. max
   D. len
   E. ไม่มีข้อใดถูก

7. ฟังก์ชันใดต่อไปนี้เป็นฟังก์ชันสำหรับตัวอักษร
   A. avg
   B. year
   C. max
   D. substr
   E. ไม่มีข้อใดถูก

8. ภาษาใดต่อไปนี้ สามารถใช้ในการพัฒนา Stored Procedures ใน Data Studio
   A. SQL PL
   B. PL/SQL
   C. Java
   D. ถูกทุกข้อ
   E. ไม่มีข้อใดถูก

9. เราสามารถเรียกใช้งานฟังก์ชันได้จากคำสั่ง 'VALUES' ใช่หรือไม่
   A. ใช่
   B. ไม่ใช่

10. เราสามารถเรียกใช้งาน Procedure ได้ด้วยคำสั่ง 'CALL' ใช่หรือไม่
   A. ใช่
   B. ไม่ใช่

7

บทที่ 7 – การเรียกใช้งาน SQL ในโปรแกรมประยุกต์

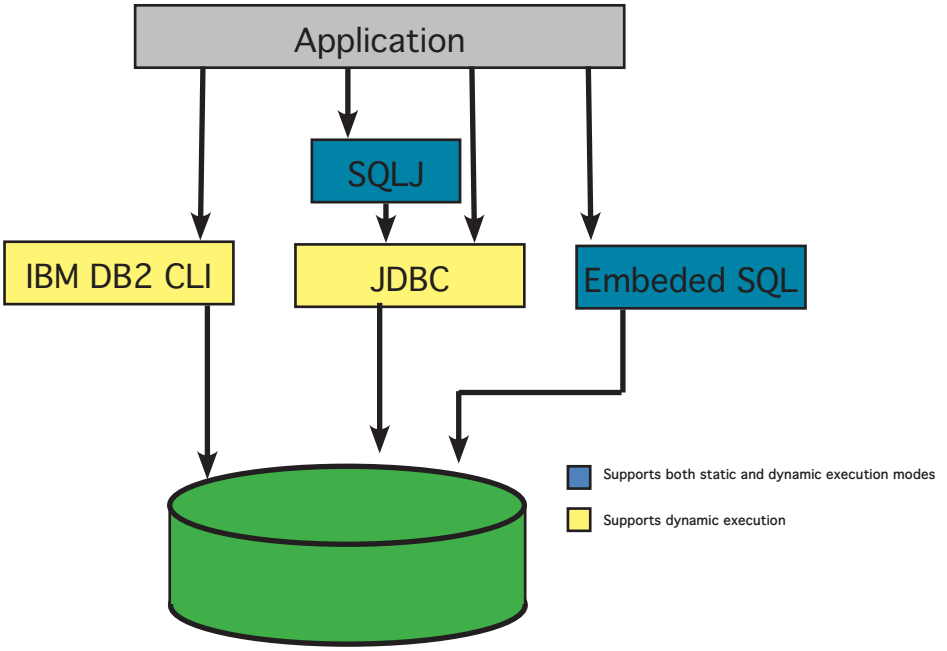
จากบทที่ผ่านมาได้นำเสนอเรื่องราวที่เกี่ยวกับ Structured Query Language (SQL) ซึ่งเป็นภาษาหรือคำสั่งมาตรฐานในการเชื่อมต่อ, จัดการออบเจ็กต์ และเรียกใช้ข้อมูลจากฐานข้อมูล ซึ่งในบทเรียนนี้จะนำเสนอแนวคิด และวิธีการเรียกใช้งาน SQL จากโปรแกรมประยุกต์ที่ถูกพัฒนาโดยภาษาที่เป็นที่นิยมต่างๆ เช่น C, C++, Java, .NET และอื่น ๆ ว่าสามารถทำได้อย่างไร
โดยบทเรียนนี้จะศึกษาเรื่องต่าง ๆ เกี่ยวกับแนวคิดดังต่อไปนี้

- แนวคิดของการทำ Transaction
- การทำงานร่วมกับ embedded SQL
- ความแตกต่างระหว่าง static และ dynamic SQL
- APIs ที่ใช้เชื่อมต่อกับฐานข้อมูล เช่น ODBC, CLI และ JDBC
- เริ่มต้นการทำงานแบบ pureQuery

7.1 ภาพรวมของการใช้งาน SQL ในโปรแกรมประยุกต์

SQL เป็นภาษาหรือคำสั่งมาตรฐานที่อนุญาตให้ผู้ใช้งานสามารถติดต่อกับฐานข้อมูลได้ แต่อย่างไรก็ตามการเขียนคำสั่งหรือฟังก์ชันต่างๆ ที่ใช้ในโปรแกรมประยุกต์ เพื่อทำงานกับฐานข้อมูลโดยอาศัยคำสั่ง SQL เพียงอย่างเดียวยังไม่เพียงพอ ดังนั้นภาษาที่ใช้ในการพัฒนาโปรแกรมประยุกต์ต่าง ๆ เช่น C, C++ หรือ Java จึงอนุญาตให้ผู้ใช้งานสามารถควบคุม ตรรกะของฟังก์ชัน (functional logic) ได้ โดยภาษาเหล่านี้จะเป็นที่รู้จักกันในชื่อ Host Language ซึ่งสามารถรวมเอาตรรกะของฟังก์ชัน และคำสั่ง SQL เข้าด้วยกันภายใต้ตัวโปรแกรมประยุกต์ เพื่อใช้ในการติดต่อฐานข้อมูลในกรณีนี้ SQL ถูกฝังตัว (embedded) ใน host application

ส่วนวิธีการอื่นๆ คืออนุญาตให้ใช้งาน Application Programming Interface (API) ในการเชื่อมต่อ และเข้าถึงฐานข้อมูล ตัวอย่างเช่น ODBC, CLI และ JDBC โดยที่เทคนิคของโปรแกรมประยุกต์ที่ใช้งาน SQL เพื่อเชื่อมต่อกับฐานข้อมูลจะแสดงในรูปที่ 7.1 ด้านล่าง โดยเราจะศึกษาในรายละเอียดของเทคนิคต่างๆ ซึ่งจะกล่าวในหัวข้อต่อไป



รูปที่ 7.1 – เทคนิคของโปรแกรมประยุกต์ที่ใช้งาน SQL

7.2 Transaction คืออะไร?

ก่อนที่จะอธิบายรายละเอียดของเทคนิคต่างๆของโปรแกรมประยุกต์ที่ใช้งาน SQL ควรจะมีความเข้าใจเกี่ยวกับแนวคิดของ Transaction กอน Transaction หรือหนวยของการทำงาน (unit of work) เป็นชุดของคำสั่ง ตาง ๆ ของฐานข้อมูล เช่น คำสั่ง insert, update หรือ delete ที่จะต้องประมวลผลทุกคำสั่งในชุดของคำสั่งนั้นให้เสร็จสิ้นทั้งหมด จึงจะถือได้วาทำ Transaction นั้นสำเร็จ

ตัวอย่างเช่น การโอนเงินจากธนาคารจำนวน 1,000 บาทจากบัญชี A ไปยังบัญชี B จะต้องทำตามขั้นตอนต่างๆ ตอไปนี้ จึงจะถือว่าการโอนเงินทำได้สำเร็จ

- ทำการลดจำนวนเงิน 1,000 บาทจากบัญชี A
- ทำการเพิ่มจำนวนเงิน 1,000 บาทในบัญชี B

ในมุมของ SQL จากตัวอย่างดังกล่าวมีการ ใช้คำสั่ง SQL อยู่ 2 คำสั่ง ดังนั้นถ้าคำสั่งใดคำสั่งหนึ่งเกิดประมวลผลไม่สำเร็จ จะต้องถือว่า การโอนเงิน นั้นทำไม่สำเร็จ ซึ่งคำสั่ง 2 คำสั่งที่แสดงในตัวอย่างนี้จะประกอบกันเป็นชุดของคำสั่ง ที่เราเรียกว่า Transaction โดย SQL จะเป็นตัวจัดการในการยืนยัน (Commit) หรือ ยกเลิก (Rollback) Transaction เพื่อให้ข้อมูลมีความถูกต้อง สำหรับตัวอย่างอื่น ๆ ของการใช้ Transaction ได้แก ระบบการจองตั๋วออนไลน์ การซื้อสินค้าออนไลน์ และอื่น ๆ

7.3 Embedded SQL

Embedded SQL เป็นการเขียนคำสั่ง SQL ฝังไว้ที่ host application ซึ่งจะถูกเขียนด้วยภาษาในการพัฒนาโปรแกรม ขั้นสูง เช่น C, C++ หรือ COBOL ซึ่ง host application เหล่านี้จะ ประกอบไปด้วย ตรรกะทางธุรกิจที่จำเป็น และจะเรียกใช้งานคำสั่ง SQL ในการเข้าถึงข้อมูลต่างๆ ในฐานข้อมูล มีคำถามเกิดขึ้นว่า จะทำอย่างไรเมื่อทำการแปล (compile) โปรแกรมประยุกต์ต่างๆ ที่พัฒนาด้วยภาษาในการพัฒนาโปรแกรม ขั้นสูงซึ่งทำการฝังคำสั่ง SQL ไว้ในโปรแกรมประยุกต์เหล่านั้น ให้เป็นภาษาคอมพิวเตอร์ ? เพราะ ตัวแปลภาษา (compiler) ของภาษาในการพัฒนาโปรแกรมเหล่านั้นไม่ได้รองรับการตรวจสอบข้อผิดพลาดของไวยากรณ์ในคำสั่ง SQL ซึ่งคำตอบของคำถามคือ การทำการแปลเป็นภาษาคอมพิวเตอร์ล่วงหน้า (pre-compilation) ในกรณีของ DB2 จะมีตัวแปลภาษาล่วงหน้า (pre-compiler) ซึ่งทำหน้าที่ในการ แปลงไวยากรณ์ของคำสั่ง SQL

ให้อยู่ในรูปแบบ API ที่เป็น runtime services API calls ผลที่ได้จากการแปลภาษาล่วงหน้า จะถูกแปลและเชื่อมโยง (link) โดยเครื่องมือในการพัฒนาของ host language ต่างๆ

คำสั่ง SQL ที่ถูกฝังไว้ใน host language ต่างๆ จะต้องมีสิ่งที่ทำให้ทราบว่าคำสั่งดังกล่าว เป็น embedded SQL ในกรณีของ C, C++ และ COBOL คำสั่งจะเริ่มต้นด้วย EXEC SQL ซึ่งจะตอดด้วยคำสั่ง SQL ที่ต้องการทำการประมวลผลและปิดท้ายคำสั่งด้วยเครื่องหมายเซมิโคลอน (;) ตัวอย่างเช่น

```
EXEC SQL
UPDATE employee.details
      SET emp_desig = 'Mgr' WHERE emp_desig = 'Asst Mgr';
```

คำสั่ง SQL ก็สามารถฝังไว้ในโปรแกรมประยุกต์ที่พัฒนาด้วยภาษา Java ได้ โดยเราจะเรียกว่า โปรแกรมประยุกต์ SQLJ โดยจะเริ่มต้นด้วย #sql แล้วตามด้วยคำสั่ง SQL ที่ต้องการประมวลผล ซึ่งคล้ายกับการเริ่มต้นด้วย EXEC SQL ในกรณีของ host language

ดังตัวอย่างสำหรับการใช้งานคำสั่ง UPDATE ของ SQL ถ้าเขียนด้วย SQLJ จะมีรูปแบบดังนี้:

```
#sql {
 UPDATE employee.details
      SET emp_desig = 'Mgr' WHERE emp_desig = 'Asst Mgr';
```

ตัวอย่างทั้งสองจะเป็นคำสั่ง SQL ที่ทำงานแบบ Static เพราะชื่อตาราง, ชื่อคอลัมน์ และ ออบเจ็กต์อื่น ๆ ในฐานข้อมูลที่อยู่ในไวยากรณ์ของ SQL จะอ้างถึงชื่อตาราง หรือ ชื่อคอลัมน์ โดยตรงในแต่ละครั้งที่เรียกใช้งาน หรือประมวลผลโปรแกรม แต่ถ้ามีการกำหนดตัวแปรให้มีการรับค่าที่เป็น ชื่อตาราง หรือ ชื่อคอลัมน์ในระหว่างการประมวลผล เราจะเรียกว่า การทำงาน ในรูปแบบ Dynamic SQL ซึ่งจะกล่าวถึงในหัวข้อต่อไป

7.3.1 Static SQL

Static SQL เป็นวิธีการแบบดั้งเดิมในการสร้างโปรแกรมประยุกต์แบบฝังคำสั่ง SQL (embedded SQL) โดยที่โปรแกรมประยุกต์แบบ Static SQL นั้นจะถูกออกแบบสำหรับโปรแกรมประยุกต์ที่ต้องการใช้คำสั่ง SQL ที่เหมือน ๆ กันทุกครั้งที่ทำการติดต่อกับฐานข้อมูล ตัวอย่างเช่น โปรแกรมซึ่งทำการปรับปรุงจำนวนคงเหลือของวัตถุดิบ มักจะปรับปรุงข้อมูลในตารางเดิม และคอลัมน์เดิมเสมอ ในทำนองเดียวกันกับระบบการสำรองที่นั่งออนไลน์ที่มักจะปรับปรุงตารางเดียวกันและทำเครื่องหมายการสำรองที่นั่งไว้ในคอลัมน์เดียวกันสำหรับการสำรองที่นั่งใหม่ และการตรวจสอบสถานะการสำรองที่นั่งล่าสุด ก็จะใช้คำสั่ง SELECT ในการสอบถามข้อมูลในตารางเดียวกันเสมอ โปรแกรมประยุกต์ที่มีการฝังคำสั่ง SQL จะทราบรายละเอียดเกี่ยวกับไวยากรณ์ของคำสั่ง SQL อยู่แล้ว เนื่องจากมีการกำหนดค่าตายตัว (hard-code) ไว้ในตัวโปรแกรม โดยเราจะเรียกโปรแกรมประยุกต์ในลักษณะนี้ว่า โปรแกรมประยุกต์ที่ฝังคำสั่ง SQL ไว้แบบคงที่ (static embedded SQL application) ซึ่งการส่งผ่านข้อมูลจากโปรแกรมประยุกต์ไป ยังคำสั่ง SQL ไม่ว่าจะเป็นค่าของข้อมูลที่ต้องเพิ่มเข้าไปในตาราง หรือ ค่าของข้อมูลที่จะใช้ใน where clause ของคำสั่ง SQL จะสามารถทำได้โดยการใช้มีเพียงข้อมูลที่ต้องการจะทำการเพิ่มลงไปในตารางเท่านั้น ซึ่งข้อมูลที่จะเติมลงไปในคำสั่ง SQL นั้นจะใช้ ตัวแปรแบบโฮสต์ (host variables)

7.3.1.1 ตัวแปรแบบโฮส (Host variables)

ตัวแปรแบบโฮส เป็นตัวแปรที่ใช้ในการพัฒนาโปรแกรมซึ่งจะสามารถใช้ได้ในคำสั่ง static SQL เท่านั้น โดยตัวแปรเหล่านี้จะต้องทำการประกาศตัวแปรก่อนใช้งาน ซึ่งในทางปฏิบัติ จะกำหนดค่าเริ่มต้น (default) ของตัวแปรในขณะทำการประกาศตัวแปรด้วย โดยในการตั้งชื่อตัวแปรที่ดีควรจะมีคำว่า '_hv' ตามหลังชื่อตัวแปร ซึ่งจะแสดงความแตกต่างจากตัวแปรอื่นๆ ตัวอย่างในการเขียนคำสั่ง SQL แบบฝังในโปรแกรมประยุกต์ภาษา C แสดงในรายการที่ 7.1

```
EXEC SQL
  SELECT emp_name, emp_dept, emp_salary
    INTO :name_hv, :dept_hv, :salary_hv
    FROM employee.details
```

```
    WHERE emp_id = :id_hv ;

  EXEC SQL
    UPDATE employee.details
        SET emp_salary = :new_salary_hv
            WHERE emp_id = :id_hv ;
```

รายการที่ 7.1 – บางส่วนของคำสั่ง SQL แบบฝังในภาษา C

ในคำสั่งทั้งสองจะปรากฏชื่อตาราง (employee.details) และชื่อคอลัมน์ (emp_name, emp_dept, etc.) ซึ่งมีการกำหนดค่าตายตัว โดยข้อมูลที่จะส่งไปยังคำสั่ง SQL ในขณะทำการประมวลผล จะทำโดยผ่าน ตัวแปรแบบโฮสต่าง ๆ (id_hv, dept_hv, ฯลฯ) โดยเครื่องหมายโคลอน (:) ก่อนชื่อตัวแปรแบบโฮสเป็นส่วนหนึ่งของไวยากรณ์คำสั่ง SQL แบบฝัง

7.3.1.2 โครงสร้างโปรแกรมประยุกต์แบบฝังคำสั่ง SQL

โปรแกรมประยุกต์แบบฝังคำสั่ง SQL จะประกอบไปด้วยสามส่วนหลัก ซึ่งจะต้องทำการกำหนด และประมวลผลคำสั่ง SQL ดังต่อไปนี้

- ส่วนของการประกาศตัวแปรโฮสต่าง ๆ (DECLARE SECTION)
- ส่วนหลักของโปรแกรม ที่ทำการกำหนด และการประมวลผลคำสั่ง SQL
- ตรรกะที่ใช้ในการยืนยัน (commit) และ ยกเลิก (rollback) การเปลี่ยนแปลงที่กระทำโดยคำสั่ง SQL

รายการที่ 7.2 เป็นการแสดงโครงสร้างทั้งสามส่วนของคำสั่ง SQL แบบฝัง โดยเป็นการเขียนในรูปแบบของภาษา C

```
int getDetails( int employee_id, double new_salary)
{
   int ret_code = 1;
   EXEC SQL INCLUDE SQLCA;

   EXEC SQL BEGIN DECLARE SECTION;
   sqlint32 id_hv = 0;          // employee id
   char name_hv[129] = {0};          // employee name
   char dept_hv[129] = {0}; // employee department
   double salary_hv = 0;       // employee salary

EXEC SQL END DECLARE SECTION;
// Copy the employee id and salary passed to this function
// into the host variables
id_hv = employee_id;
salary_hv = new_salary;

// Issue the UPDATE statement to set the new salary of an employee
EXEC SQL
 UPDATE employee.details
       SET emp_salary = :salary_hv WHERE emp_id = :id_hv;

if (SQLCODE < 0)
```

```
    {
     printf("\n UPDATE SQL Error:%ld\n",SQLCODE);
     EXEC SQL ROLLBACK; // Rollback the transaction
     ret_code = 0; // error
    }
  else
    {
     EXEC SQL COMMIT; // Commit the transaction

  // Issue a SELECT to fetch updated salary information
    EXEC SQL
    SELECT emp_name, emp_dept, emp_salary
      INTO :name_hv, :dept_hv, :salary_hv
      FROM employee.details
     WHERE emp_id  = :id_hv;

   if (SQLCODE < 0)
   {
     printf("\n SELECT SQL Error:%ld\n",SQLCODE);
     Ret_code = 0;
   }
   else
   {
    // Display the updated salary information
       printf("\n Employee name: %s",name_hv);
       printf("\n Employee Id: %d",id_hv);
       printf("\n Employee Department: %s",dept_hv);
       printf("\n Employee New Salary: Rs. %ld p.a",salary_hv);
   }
  }
  return ret_code;

  int getDetails( int employee_id, double new_salary)
  {
     int ret_code = 1;
     EXEC SQL INCLUDE SQLCA;

     EXEC SQL BEGIN DECLARE SECTION;
     sqlint32 id_hv = 0;          // employee id
     char name_hv[129] = {0};          // employee name
     char dept_hv[129] = {0}; // employee department
     double salary_hv = 0;       // employee salary

  EXEC SQL END DECLARE SECTION;
```

```
// Copy the employee id and salary passed to this function
// into the host variables
id_hv = employee_id;
salary_hv = new_salary;

// Issue the UPDATE statement to set the new salary of an employee
EXEC SQL
 UPDATE employee.details
       SET emp_salary = :salary_hv WHERE emp_id = :id_hv;
if (SQLCODE < 0)
   {
    printf("\n UPDATE SQL Error:%ld\n",SQLCODE);
    EXEC SQL ROLLBACK; // Rollback the transaction
    ret_code = 0; // error
   }
else
   {
    EXEC SQL COMMIT; // Commit the transaction

// Issue a SELECT to fetch updated salary information
   EXEC SQL
   SELECT emp_name, emp_dept, emp_salary
     INTO :name_hv, :dept_hv, :salary_hv
     FROM employee.details
    WHERE emp_id  = :id_hv;

  if (SQLCODE < 0)
  {
    printf("\n SELECT SQL Error:%ld\n",SQLCODE);
    Ret_code = 0;
  }
  else
  {
   // Display the updated salary information
      printf("\n Employee name: %s",name_hv);
      printf("\n Employee Id: %d",id_hv);
      printf("\n Employee Department: %s",dept_hv);
      printf("\n Employee New Salary: Rs. %ld p.a",salary_hv);
  }
 }
 return ret_code;
}
```

รายการที่ 7.2 -- การแสดงโครงของคำสั่ง SQL แบบฝัง โดยเป็นการเขียนในรูปแบบของภาษา C

จากตัวอย่าง ข้างต้นแสดงให้เห็นถึงคำสั่ง SQL ที่นำไปเขียนแบบฝังตัวไว้ในคำสั่งของภาษา C host language อื่นๆเช่น ภาษา C++, COBOL, Java ก็ประกอบไปด้วยองค์ประกอบสามส่วน เช่นเดียวกันกับตัวอย่างข้างต้น

7.3.1.3 การใช้ SQL communications area SQLCODE และ SQLSTATE

SQL Communications Area (SQLCA) เป็นโครงสร้างของข้อมูลที่ใช้ในการติดต่อสื่อสารระหว่างฐานข้อมูลกับไคลเอ็นต์ (client) โดยโครงสร้างของ SQLCA จะประกอบไปด้วยตัวแปรที่ใช้สำหรับเก็บค่าสถานะของการประมวลผลในแต่ละคำสั่ง SQL โดยที่ SQLCODE เป็นตัวแปรหนึ่งในโครงสร้างข้อมูลที่จะถูกกำหนดค่าไว้เป็น 0 (ศูนย์) ในกรณีที่การประมวลผลคำสั่ง SQL ทำได้สำเร็จ ถ้าคำสั่ง SQL ที่ทำการประมวลผลทำได้สำเร็จ แต่มี คำเตือน (Warning) เกิดขึ้น ค่าของ SQLCODE ที่ส่งกลับจะเป็น ค่าบวกที่ไม่ใช่ศูนย์ และกรณีที่คำสั่ง SQL เกิดความผิดพลาด (error) ค่าของ SQLCODE ที่ส่งกลับจะเป็น ค่าที่ติดลบ

การตรวจสอบค่าของ SQLCODE เป็นการตรวจสอบถึงการประมวลผลของคำสั่ง SQL ว่ามีข้อผิดพลาดหรือไม่ ซึ่งค่าของ SQLCODE อาจจะเป็นค่าที่บ่งบอกถึงปัญหาทางด้านฮาร์ดแวร์หรือปัญหาทางด้านระบบปฏิบัติการ ตัวอย่างเช่น เมื่อระบบไฟล์ซิสเต็ม เต็ม หรือมีข้อผิดพลาดในการเข้าถึงไฟล์ เป็นแนวคิดที่ดีที่จะมีการตรวจสอบค่าของ SQLCODE หลังจากที่มีการประมวลผลคำสั่ง SQL

SQLSTATE เป็นตัวแปรอีกตัวหนึ่งใน SQLCA ซึ่งทำหน้าที่ในการจัดเก็บค่าในรูปแบบ String ที่ส่งกลับคืนมาหลังจากการประมวลผลคำสั่ง SQL ล่าสุด แต่อย่างไรก็ตาม SQLSTATE จะแสดงค่าข้อความที่เป็นมาตรฐานที่ใช้ในผลิตภัณฑ์ฐานข้อมูลแต่ละยี่ห้อ

7.3.1.4 ขั้นตอนในการ แปลภาษาโปรแกรมประยุกต์ Static SQL

จากที่กล่าวมาข้างต้นว่าโปรแกรมประยุกต์ SQL แบบฝังตัวนั้น จำเป็นต้องมีการ แปลภาษาล่วงหน้าโดยใช้ ตัวแปลภาษาล่วงหน้าของ DB2 (DB2 pre-compiler) ซึ่งหน้าที่ของตัวแปลภาษาล่วงหน้า คือจะทำการตรวจสอบคำสั่ง SQL ที่อยู่ในซอสโค๊ด และทำการแทนที่โค๊ดเหล่านั้นด้วย DB2 runtime APIs และทำการเขียนเป็นคำสั่ง (ที่มีการ comment คำสั่ง SQL ไว้) ไปยังไฟล์ใหม่ซึ่งหลังจากนั้น สามารถที่จะแปลภาษาและเชื่อมโยง (link) โดยเครื่องมือในการพัฒนาของ host language ต่างๆ

ตัวแปลภาษาลวงหน้าของ DB2 จะถูกเรียกใช้งานด้วยคำสั่ง PRECOMPILE (หรือ PREP) ซึ่ง ตัวแปลภาษาล่วงหน้าของ DB2 จะทำงานตามขั้นตอนต่าง ๆ ดังนี้

1. ทำการตรวจสอบไวยากรณ์สำหรับคำสั่ง SQL แต่ละคำสั่ง และยืนยันว่าชนิดของข้อมูลที่ใช้มีความเหมาะสมสำหรับตัวแปรแบบโฮสต์โดยทำการเปรียบเทียบกับชนิดของข้อมูลของคอลัมน์ตามลำดับ และยังกำหนดวิธีการแปลงข้อมูลที่จะนำมาใช้ในการ fetch หรือการเพิ่มข้อมูลในฐานข้อมูล

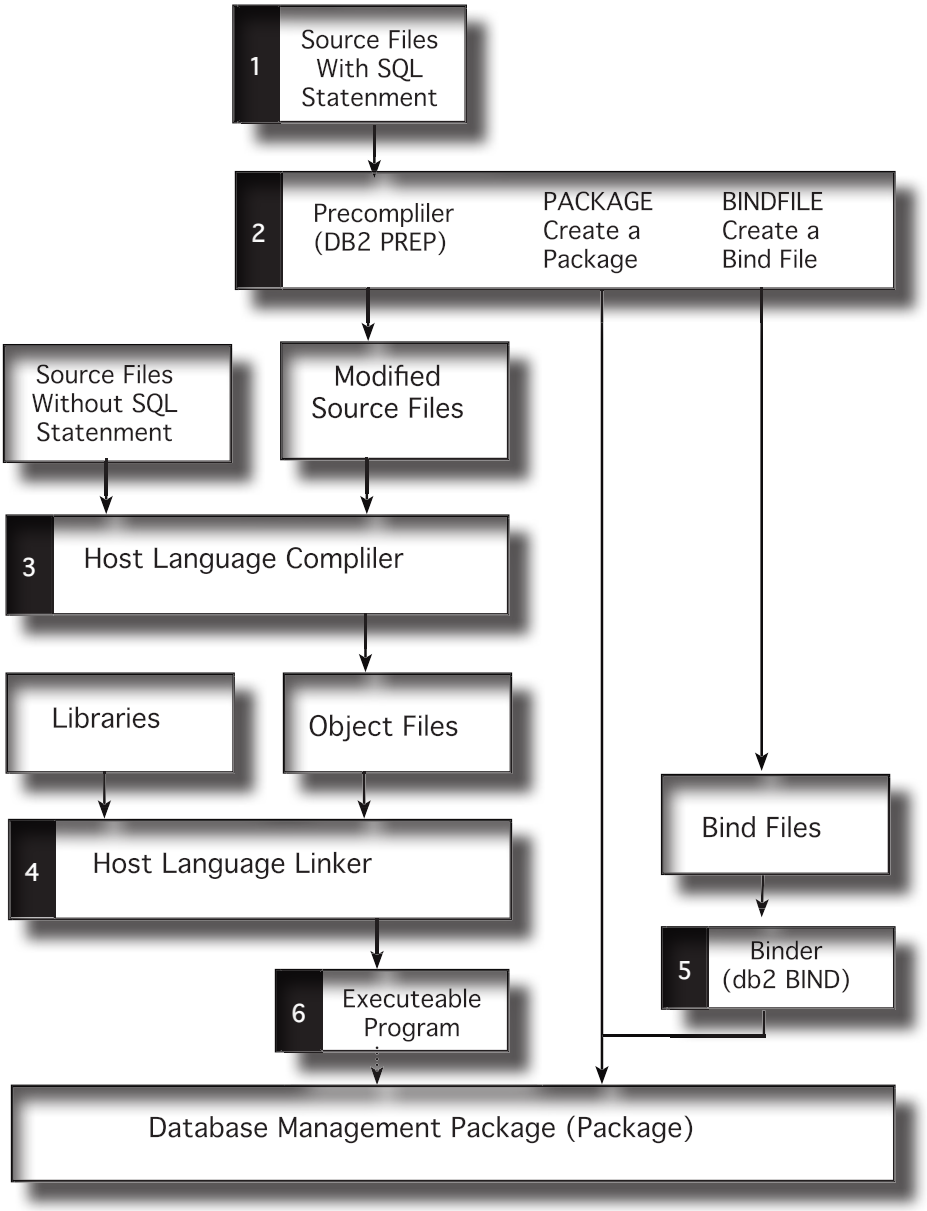
2. ทำการประเมิน โดยอ้างอิงถึงออบเจ็กต์ของฐานข้อมูล สร้างแผนการเข้าถึงข้อมูล (access plan) โดยจะเก็บแผนการเข้าถึงข้อมูลเหล่านั้นไว้ในแพคเกจในฐานข้อมูล แผนเข้าถึงข้อมูลของคำสั่ง SQL จะเป็นวิธีการที่ดีที่สุดในการเข้าถึงข้อมูลที่คำสั่ง SQL จะใช้อ้างอิง โดย DB2 optimizer จะทำการประเมินแผนการเข้าถึงข้อมูล นี้ในขณะที่ทำการแปลภาษาล่วงหน้า ของคำสั่ง SQL ฝังตัวแบบ Static (static embedded SQL statement) การประเมินนี้จะยึดตามข้อมูลที่มีอยู่และถูกส่งไปยัง optimizer ในขณะที่ทำการแปลภาษาล่วงหน้า ซึ่งโดยทั่วไปจะเป็นข้อมูลจากแค็ตตาล็อกของระบบฐานข้อมูล (system catalog)

นอกจากนั้นแต่ละโปรแกรมประยุกต์จะถูก ผูกไว้ กับแพคเกจที่เกี่ยวข้องที่เก็บอยู่ในฐานข้อมูล ประโยชน์ของการเก็บแผนในการเข้าถึงข้อมูลเหล่านั้นไว้ในฐานข้อมูล คือทุกครั้งที่โปรแกรมประยุกต์ถูกเรียกใช้งาน จะมีการใช้แผนการเข้าถึงข้อมูลสำหรับคำสั่ง SQL ที่ดึงมาจาก แพคเกจ ซึ่งเป็นผลให้การ ประมวลผล SQL ทำได้อย่างรวดเร็วในฐานข้อมูล เนื่องจาก optimizer ทราบถึงวิธีการที่เหมาะสมมากที่สุดในการเข้าถึง ออบเจ็กต์ของฐานข้อมูลล่วงหน้า ดังนั้นสำหรับคำสั่ง SQL แบบ Static ที่จะฝังตัวในโปรแกรมประยุกต์ ไวยากรณ์ของคำสั่งเหล่านั้น จะถูกรู้ล่วงหน้าในขณะที่ทำการแปลภาษาล่วงหน้า (pre-compile)

เมื่อโปรแกรมประยุกต์ SQL แบบฝังตัว ถูกแปลภาษาล่วงหน้า และถูกผูก (bind) ไว้กับฐานข้อมูล โปรแกรมประยุกต์ SQL แบบฝังตัวดังกลาว จะสามารถถูกแปลภาษา และเชื่อมโยง (link) ด้วยเครื่องมือการ

พัฒนา host language แต่เราสามารถผัดผ่อนการผูก (bind) โปรแกรมประยุกต์ โดยการระบุ BINDFILE <bindfile> clause ในคำสั่ง PRECOMPILE ซึ่งจะทำการสร้าง bindfile อันประกอบไปด้วยข้อมูลที่จำเป็นในการสร้างแพคเกจบนฐานข้อมูล ซึ่ง <bindfile> นี้จะสามารถใช้กับ DB2 BIND Utility ในการสร้างแพคเกจในฐานข้อมูล รวมถึงใช้ในการผูกโปรแกรมประยุกต์เข้ากับแพคเกจ โดยจะเรียกวิธีการดังกล่าวว่า deferred binding

ในรูปที่ 7.2 ด้านล่างนี้อธิบายถึง กระบวนการ แปลภาษา Static SQL แผนภาพนี้แสดงถึงการรับส่งการกระทำทั่วไปของการดำเนินการในขั้นตอนการ แปลภาษาซึ่งอาจแตกต่างกันไปสำหรับหน่วย แปลภาษาที่มีลักษณะเฉพาะ



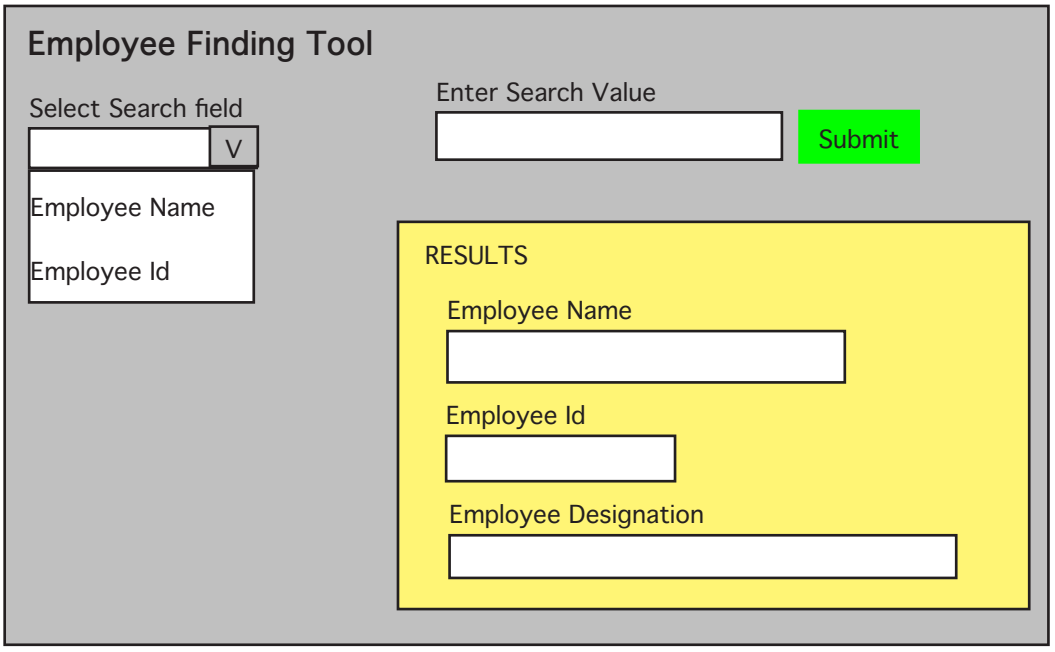
รูปที่ 7.2 - กระบวนการแปลภาษา Static SQL

สำหรับโปรแกรมประยุกต์ที่พัฒนาด้วย SQLJ จะมีกระบวนการคล้ายกับตัวแปลภาษาล่วงหน้าของ DB2 สำหรับโปรแกรมประยุกต์ SQL แบบฝังตัว โดย SQLJ Translator จะ ตรวจจับคำสั่ง SQL ภายใน โปรแกรมประยุกต์ SQLJ เพื่อแปลงข้อมูลเหล่านั้น ให้เป็นคำสั่ง JDBC ซึ่งรายละเอียดต่างๆของ JDBC จะกล่าวถึง ต่อไปในบทนี้

7.3.2 คำสั่ง SQL แบบ Dynamic

คำสั่ง SQL แบบ Dynamic ประกอบด้วยพารามิเตอร์ต่างๆที่ไม่ทราบค่าล่วงหน้า จนกว่าจะทำการ

ป้อนข้อมูลในโปรแกรมประยุกต์เพื่อทำการประมวลผล ซึ่งคำสั่ง SQL แบบ Dynamic จะสะดวกสำหรับผู้ใช้งานโปรแกรมประยุกต์ที่ให้บริการแก่ผู้ใช้งานแบบโต้ตอบ (Interactive) โดยที่ข้อมูลต่างๆ ที่ผู้ใช้งานป้อนเข้าไปจะนำมาใช้ในการสร้างคำสั่ง SQL ดังตัวอย่างที่แสดงในรูปที่ 7.3 ,"Employee finder tool" ซึ่งเป็นส่วนหนึ่งของโปรแกรมประยุกต์ HR



รูปที่ 7.3 – โปรแกรม Employee finder tool

จากเครื่องมือสำหรับค้นหาที่อยู่ในโปรแกรม HR ข้างต้น ทำให้ผู้ใช้สามารถเรียกดูรายละเอียดต่างๆของพนักงาน สามารถค้นหาพนักงานโดยใช้ ชื่อพนักงานหรือ รหัสพนักงาน ซึ่งโปรแกรมประยุกต์นี้จะยังไม่ทราบรูปแบบคำสั่ง SQL ที่แน่นอนจนกว่าจะถึงเวลาที่จะเริ่มต้นทำการค้นหาข้อมูลตามพารามิเตอร์ที่ผู้ใช้ระบุ ซึ่งเป็นตัวอย่างของการประมวลผล SQL แบบ Dynamic ซึ่งจะต่างจากเทคนิคของ Static SQL ตรงที่แผนในการเข้าถึงข้อมูลของ Dynamic SQL จะถูกสร้างขึ้น ณ ขณะที่โปรแกรมทำงาน (Runtime) เท่านั้น

7.3.2.1 โครงสร้างโปรแกรมประยุกต์ SQL แบบฝังตัว

จากโปรแกรมตัวอย่างที่ 7.3 ข้างล่างเป็นการแสดงคำสั่งที่ใช้สร้างเป็นเครื่องมือสำหรับการค้นหาข้อมูลพนักงาน Employee Finder Tool ซึ่งเป็นการสาธิตให้เห็นถึงวิธีการเรียกใช้คำสั่ง SQL แบบ Dynamic จากโปรแกรมประยุกต์ SQL แบบฝังตัวที่พัฒนาขึ้นด้วยภาษา C

```
int findEmployee(char * field, char * value, char * emp_name, char * emp_id, char *
emp_desig)
{
 int ret_code = 1;
 char sqlstmt[250] = {0};
 EXEC SQL INCLUDE SQLCA;

 EXEC SQL BEGIN DECLARE SECTION;
  char name_hv[129] = {0};  // employee name
  char desig_hv[129] = {0}; // employee designation
  char id_hv[10] = {0};          // employee id
  char value_hv[250] = {0}; // Field value
```

```
 EXEC SQL END DECLARE SECTION;
// Copy the Search Field passed to this function into
// the sqlstmt string
sprintf(sqlstmt,"SELECT emp_name, emp_id, emp_desig
                 FROM employee.details
                WHERE %s = ?",field);
// Copy the field value passed to this function into the
// host variable
    strcpy(value_hv,value) ;

// Prepare the dynamic SQL statement first. This statement would
// create the access plan at runtime
  EXEC SQL PREPARE dynsqlstmt FROM :sqlstmt;
  if (SQLCODE <0)
  {
    printf("\n Error during Prepare statement:%ld",SQLCODE);
    ret_code = 0; //error
  }
  else
  {
    EXEC SQL DECLARE cur1 CURSOR FOR :dynsqlstmt;
    if (SQLCODE <0)
    {
     printf("\n Error during Declare Cursor:%ld",SQLCODE);
     ret_code = 0; //error
     }
     else
     {
       EXEC SQL OPEN cur1 USING :value_hv;
       if (SQLCODE <0)
       {
         printf("\n Error during Open Cursor:%ld",SQLCODE);
         ret_code = 0; //error
       }
       else
       {
         EXEC SQL FETCH cur1 INTO :name_hv, :id_hv, :design_hv;
         if (SQLCODE <0)
         {
           printf("\n Error during Fetch cursor:%ld",SQLCODE);
           ret_code = 0; //error
         }
         else
         {
```

```
            EXEC SQL CLOSE cur1;
             if (SQLCODE <0)
             {
               printf("\n Error during Close cursor:%ld",SQLCODE);
               Ret_code = 0; //error
             }
            else
            {
             // Copy the fetched results into the target variables
             strcpy(emp_name,:name_hv);
             strcpy(emp_id,:id_hv);
             strcpy(emp_desig,:desig_hv);
             }
           }
         }
        }
      }
    return ret_code;
```

รายการที่ 7.3 – ตัวอย่าง SQL แบบฝังตัวที่ทำงานร่วมกับโปรแกรมภาษา C โดยการใช้งาน SQL แบบ static และแบบ Dynamic

จากตัวอย่าง ข้างต้น ข้อมูล ที่ระบุในฟิลด์เพื่อใช้ในการค้นหา พนักงานใน Employee Finder Tool (ซึ่งผู้ใช้งานจะระบุเงื่อนไขในขณะใช้งาน) จะถูกเก็บอยู่ในตัวแปรเรียกว่า field โดยข้อมูลนี้จะถูก นำไปใช้ในคำสั่ง SQL ซึ่งจะถูกเก็บในตัวแปรสตริง (จากคำสั่ง sqlstmt ข้างต้น)

สมมติว่า ผู้ใช้เลือก Employee Id เป็นเงื่อนไขในการค้นหา ในกรณีนี้คำสั่ง SQL (โดยไม่มีค่า predicate – ค่าของคอลัมน์ที่อยู่หลัง where) จะมีรูปแบบคำสั่งดังต่อไปนี้

SELECT emp_name, emp_id, emp_desig from employee.details WHERE emp_id =?

เครื่องหมายคำถาม (?) ที่ใช้ในคำสั่งนี้ เรียกว่า parameter marker เครื่องหมายนี้จะใช้แทน ค่าของคอลัมน์ที่ผู้ใช้ระบุให้เป็นเงื่อนไขในการค้นหา (ซึ่งในที่นี้คือ ค่าของคอลัมน์ emp_id) ที่ผู้ใช้ป้อนเป็นข้อมูลเข้ามาในขณะที่โปรแกรมทำงาน (runtime) เนื่องจากค่าที่ ป้อนเข้ามาจากผู้ใช้เหล่านี้ ไม่จำเป็นต้อง ใช้ในการสร้างแผนการเข้าถึงข้อมูล การสร้างแผนการเข้าถึงข้อมูล ทำได้โดยการ 'เตรียม' (prepare) คำสั่ง SQL แบบDynamic (ดูคำสั่ง EXEC SQL PREPARE ในตัวอย่างข้างต้น)

ในกรณีที่ผู้ใช้เลือก ชื่อพนักงาน เป็น เงื่อนไขในการค้นหา SQL จะมีรูปแบบคำสั่งต่อไปนี้

SELECT emp_name, emp_id, emp_desig from employee.details WHERE emp_name =?

เนื่องจากรูปแบบของคำสั่ง SQL ที่สมบูรณ์ จะถูกสร้างขึ้นในขณะกำลังโปรแกรมทำงานเท่านั้น ในกรณีนี้ การ 'เตรียม' (prepare) คำสั่ง SQL แบบ Dynamic จะใช้แผนในการเข้าถึงข้อมูลที่แตกต่างจากตัวอย่างแรก (ที่ใช้ รหัสพนักงาน เป็นเงื่อนไขในการค้นหา)

เมื่อคำสั่งถูกจัดเตรียม (prepare) เรียบร้อยแล้ว คำสั่ง SQL ก็พร้อมที่จะทำงาน โดยการรับค่าของข้อมูลที่ป้อนจากผู้ใช้ ในกรณีของคำสั่ง SELECT ขั้นตอนจะเริ่มจากการประกาศเคอร์เซอร์ (declare cursor) สำหรับคำสั่ง SQL นั้นๆ แล้วทำการเปิดใช้เคอร์เซอร์ โดยใช้ค่าของข้อมูลที่ผู้ใช้ป้อนเข้ามา (open cursor) หลังจากนั้นจะทำการ fetch ผลลัพธ์ที่ได้จากการสอบถามข้อมูล ไปจัดเก็บในตัวแปรแบบ โฮสต์ และสุดท้าย เมื่อสิ้นสุดการทำงาน โปรแกรมก็จะทำการปิดการทำงานของเคอร์เซอร์ (Close cursor)

1. จากโค้ด (code) ตัวอย่างข้างต้นแสดงให้เห็นถึงบางคำสั่งที่เป็น Static SQL ที่จำเป็นสำหรับการจัดเตรียมคำสั่ง (Statement preparation) การประกาศเคอร์เซอร์ (cursor declaration), และคำสั่งอื่นๆ ซึ่งหมายความว่าโปรแกรมประยุกต์ดังกล่าวยังจำเป็นต้องใช้ การแปลภาษาล่วงหน้า หรือ ใช้ตัวแปลง SQL (สำหรับโปรแกรมประยุกต์ ที่เป็น SQLJ) เนื่องจากโปรแกรมยัง มีความจำเป็นต้องใช้ คำสั่ง SQL แบบ static อยู่

2. ถ้ามีวิธีที่จะแทนที่ Static SQL ทั้งหมดดังกล่าว โดย การใช้ APIs, การแปลภาษาล่วงหน้า/ตัวแปลง SQL ของ โปรแกรมประยุกต์จะไม่จำเป็นต้องใช้อีกต่อไป คำถามก็คือ เราจะมีวิธีการอย่างไรที่จะเขียนคำสั่ง SQL แบบ Dynamic โดยไม่ใช้คำสั่ง SQL แบบ Static เลย (รวมไปถึงคำสั่ง PREPARE, EXECUTE, การประกาศ CURSOR ,ฯลฯ) คำตอบสำหรับคำถามนี้จะอธิบายในหัวข้อต่อไป

7.3.3 ความแตกต่างระหว่าง Static และ dynamic SQL

ต่อไปนี้เป็นความแตกต่างหลักๆ ระหว่างโหมดการ ประมวลผลของ Static SQL และ Dynamic SQL ในโปรแกรมประยุกต์ที่มีคำสั่ง SQL แบบฝังตัว

1. คำสั่ง SQL แบบ Dynamic แตกต่างจากคำสั่ง SQL แบบ Static เนื่องจากแผนในการเข้าถึงข้อมูล สำหรับคำสั่งแบบ Dynamic จะถูกสร้างขึ้นเฉพาะในเวลาที่ประมวลผล (runtime) ดังนั้นคำสั่งแบบ Dynamic จะต้องถูกเตรียม (prepare) ในโปรแกรมประยุกต์

2. การสร้างแผนการเข้าถึงข้อมูล ณ เวลาที่ประมวลผลทำให้โปรแกรมประยุกต์ที่ใช้งาน SQL แบบ Dynamicใช้เวลาในการทำงานที่มากกว่า Static SQL เล็กน้อย อย่างไรก็ตาม Dynamic SQL ก็มีความยืดหยุ่นมากกว่าในการพัฒนา ซึ่งทำให้ Dynamic SQL มีความน่าสนใจมากกว่า Static SQL

3. บางครั้งคำสั่ง SQL แบบ Dynamic จะมีประสิทธิภาพดีกว่า Static SQL เนื่องจากมันสามารถที่จะใช้ประโยชน์จากข้อมูลสถิติล่าสุดที่มีอยู่ในฐานข้อมูลในเวลาที่มันประมวลผล แผนเข้าถึงข้อมูลที่สร้างขึ้นสำหรับคำสั่ง SQL แบบ Static ถูกเก็บไว้ล่วงหน้าและอาจกลายเป็นล้าสมัยในกรณีที่ข้อมูลสถิติของฐานข้อมูลเกิดการเปลี่ยนแปลง ซึ่งกรณีนี้จะไม่เกิดขึ้นถ้าใช้คำสั่ง SQL แบบ Dynamic

4. ข้อดีของ Dynamic SQL ที่แตกต่างจาก Static SQL อีกประการหนึ่งคือ เมื่อใดก็ตามที่มีการแก้ไขหรืออัพเกรดโปรแกรมประยุกต์ ถ้าส่วนที่เป็น SQL Static ของโปรแกรมประยุกต์มีการแก้ไขแล้ว แผนการเข้าถึงข้อมูลจำเป็นที่จะต้องมีการสร้างขึ้นใหม่ทุกครั้ง ซึ่งหมายความว่าจะต้องทำการแปลภาษาล่วงหน้า ของโปรแกรมประยุกต์ และ จะต้องทำ rebinding ของแพคเกจซ้ำอีกครั้ง แต่สำหรับในกรณีของ Dynamic SQL เนื่องจากแผนการเข้าถึงข้อมูลจะถูกสร้าง ณ เวลาประมวลผล (runtime), การแปลภาษาล่วงหน้า และการทำ rebinding จึงไม่มีความจำเป็น

7.4 Application Programming Interface (APIs) สำหรับระบบฐานข้อมูล

จากที่ได้อธิบายไปก่อนหน้านี้ แม้ว่าคำสั่ง SQL ที่ทำงานแบบฝังตัวจะมีความสามารถในการประมวลผลคำสั่ง SQL แบบ Dynamic แต่ก็ยังคงมีบางคำสั่ง SQL ที่เป็นแบบ Static ซึ่งจำเป็นต้องผ่านขั้นตอนการแปลภาษา ลวงหน้า (Pre-compilation) หรือขั้นตอนการแปลคำสั่ง SQL (SQL translation) นอกจากนี้ยังมีความจำเป็นที่จะเชื่อมต่อไปยังฐานข้อมูล ในขณะที่มีการแปลภาษา ลวงหน้าสำหรับโปรแกรมประยุกต์ที่ มี SQL แบบฝังตัว และเพื่อสร้างแผนการเข้าถึงข้อมูล จะต้องใช้ข้อมูลสถิติจากแคตตาล็อกของฐานข้อมูล (Database catalog) ซึ่งสิ่งต่างๆที่กล่าวมานี้ ทำให้เกิดรูปแบบใหม่ ของการพัฒนาโปรแกรมประยุกต์ โดยใช้โปรแกรมอินเตอร์เฟซเพื่อเชื่อมต่อกับฐานข้อมูล (Database Application Programming Interfaces) โดยมีลักษณะการทำงานเป็นแบบ Dynamic และสามารถใช้เป็นเครื่องมือสำหรับการพัฒนาที่เป็นอิสระจากซอฟต์แวร์ฐานข้อมูลที่ใช้ โดยไม่จำเป็นต้องมีการแปลภาษา ล่วงหน้า

API สำหรับฐานข้อมูล ประกอบด้วยกลุ่มของคำสั่ง APIs ที่พัฒนาขึ้นโดยผู้ผลิตระบบฐานข้อมูล เพื่อ ให้รองรับการพัฒนาซอฟต์แวร์โดยภาษาต่างๆ เช่นภาษา C, C++, Java และอื่นๆ ซึ่งช่วยให้นักพัฒนาโปรแกรมประยุกต์ มีกลไกที่จะช่วยให้การทำงานกับฐานข้อมูล ทำได้ง่ายขึ้นโดยผ่านทางอินเตอร์เฟซ ที่เรียกว่า SQL callable interfaces โดย API นี้จะเป็นเลเยอร์ที่อยู่ กึ่งกลางระหว่างโปรแกรมประยุกต์และฐานข้อมูล โดยการเชื่อมต่อกับฐานข้อมูลนั้นจะอาศัยไดรเวอร์ เรียกว่า database connectivity driver ซึ่งผู้ผลิตฐานข้อมูลจะมีการพัฒนาไดรเวอร์ ซึ่งจะมีส่วน libraries ของไดร์เวอร์ที่ใช้เชื่อมกับ libraries ในส่วนของ source code ทำให้สามารถทำการแปลภาษาและประมวลผลโปรแกรมประยุกต์ ทั้งนี้การพัฒนาโปรแกรม จะเป็นอิสระจากระบบฐานข้อมูล โดยไม่ต้องทำการเชื่อมต่อกับฐานข้อมูลในกรณีทำการแปลภาษา ซึ่งทำให้นักพัฒนาโปรแกรมประยุกต์มีความยืดหยุ่น สามารถเรียกใช้งานผ่าน API โดยที่ไม่จำเป็นต้องเข้าใจรายละเอียดของภาษา SQL ที่ทำงานแบบฝังตัวอีกด้วย ซึ่งในลำดับต่อไปจะกล่าวถึงรายละเอียดของไดร์เวอร์สำหรับการเชื่อมต่อฐานข้อมูล

7.4.1 ODBC และ IBM Data Server CLI driver

เพื่อให้การเชื่อมต่อกับฐานข้อมูลของผู้ผลิตฐานข้อมูลในค่ายต่างๆมีความเป็นมาตรฐาน บริษัท X/Open และ SQL Access Group ได้ร่วมมือกันพัฒนาอินเตอร์เฟซ ที่เป็นมาตรฐาน เรียกว่า X/Open Call Level Interface โดยมีวัตถุประสงค์เพื่อให้การพัฒนาโปรแกรมประยุกต์ ที่เชื่อมต่อกับฐานข้อมูลมีความยืดหยุ่นและไม่ผูกติดกับฐานข้อมูลของค่ายใดๆ โดยที่อินเตอร์เฟซนี้ได้รับการยอมรับเป็นส่วนหนึ่งของมาตรฐานสากล (ISO/IEC 9075-3:1995 SQL/CLI)

นอกจากนี้ทางบริษัทไมโครซอฟท์ได้พัฒนาอินเตอร์เฟซที่เรียกว่า SQL callable Open Database Connectivity (ODBC) สำหรับระบบปฏิบัติการไมโครซอฟท์ซึ่งอยู่บนพื้นฐานของมาตรฐาน X/Open CLI อย่างไรก็ตามในปัจจุบัน ODBC ไม่ได้จำกัดเพียงแค่ระบบปฏิบัติการของไมโครซอฟท์เท่านั้น แต่ยังสามารถใช้งานกับระบบปฏิบัติการอื่นๆได้อีกด้วย

สำหรับบริษัท ไอบีเอ็ม ได้พัฒนาไดรเวอร์สำหรับเชื่อมต่อกับฐานข้อมูล DB2 เรียกว่า IBM Data Server CLI driver ซึ่งเป็น DB2 Call level interface ที่อยู่บนพื้นฐานข้อกำหนดของ Microsoft ® ODBC และมาตรฐานสากลสำหรับ SQL/CLI ซึ่งคุณสมบัติเหล่านี้ถูกเลือกให้เป็นพื้นฐานสำหรับ DB2 Call Level Inteface เพื่อปฏิบัติตามมาตราฐานอุตสาหกรรมและเพื่อให้เกิดการเรียนรู้ที่สั้นลงสำหรับผู้เขียนโปรแกรมประยุกต์ ที่คุ้นเคยกับ Call Level Interface อื่นๆอยู่แล้ว นอกจากนี้บริษัท ไอบีเอ็ม ยังได้มีการเพิ่มเติมความสามารถของ CLI เพื่อช่วยให้ นักพัฒนาสามารถพัฒนาโปรแกรมประยุกต์เพื่อใช้งานคุณสมบัติเฉพาะบางอย่างของ DB2 โดย DB2 CLI เป็น API สำหรับภาษา C และ C++ ที่ใช้ในการเข้าถึงฐานข้อมูล โดยใช้ การเรียกฟังก์ชัน (function call) ในการส่งผ่านคำสั่ง SQL แบบ Dynamic โดยเป็นอาร์กิวเมนต์ (arguments) ของฟังก์ชัน และแม้กระทั่งคำสั่ง PREPARE และ EXECUTE ก็ยังมีคำสั่งที่เทียบเท่าเช่น SQLPrepare() และ SQLExecute ซึ่งยอมรับคำสั่ง SQL เป็นอาร์กิวเมนต์ ซึ่งหมายถึง เราไม่จำเป็นต้องใช้คำสั่ง EXEC SQL ในโปรแกรมประยุกต์ ตัวอย่างเช่น พิจารณาคำสั่ง SQL แบบ Dynamic ที่จะต้องมีการเตรียมการใช้ (prepare) โดยใช้ SQLPrepare () API

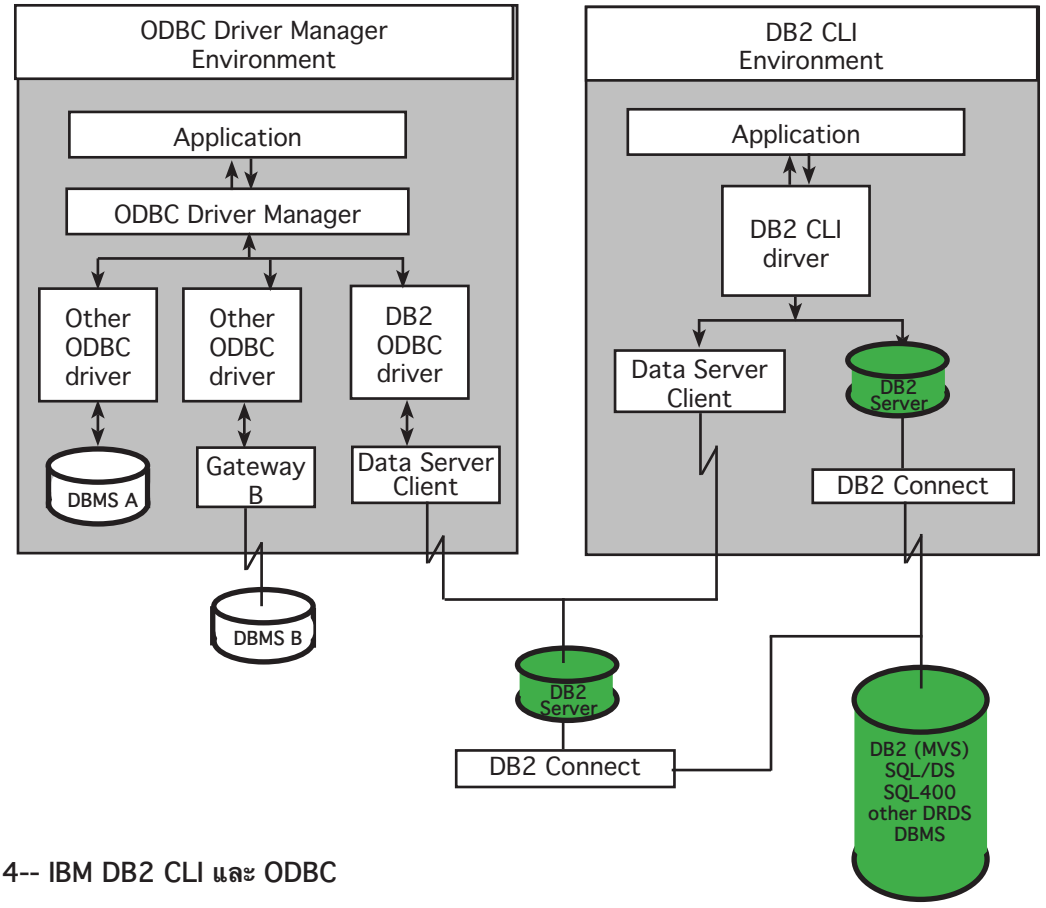
```
SQLCHAR *stmt = (SQLCHAR *)"UPDATE employee.details SET emp_id = ? WHERE
emp_name = ? ";

/* prepare the statement */
int rc = SQLPrepare(hstmt, stmt, SQL_NTS);
```

ในทำนองเดียวกัน IBM CLI ยังได้นำเสนอ callable interface อื่น ๆ เช่น SQLConnect (), SQLFetch (), SQLExecute และอื่นๆ

ตามขอกำหนดของ ODBC ซึ่ง IBM DB2 Call Level Interface นั้นจะสอดคล้องกับมาตรฐาน ODBC 3.51

รูปที่ 7.4 แสดงให้เห็นถึงที่ IBM DB2 CLI driver ในสภาพแวดล้อมแบบ Dynamic SQL สำหรับการพัฒนาโปรแกรมประยุกต์ นอกจากนี้ยังแสดงให้เห็นถึงวิธีการที่ IBM DB2 CLI driver สามารถทำหน้าที่เป็น ODBC driver อื่น ๆ เมื่อโหลดผ่าน ODBC Driver Manager



รูป 7.4-- IBM DB2 CLI และ ODBC

7.4.2 JDBC

JDBC ย่อมาจาก Java Database Connectivity ซึ่งเป็นโปรแกรมอินเตอร์เฟซสำหรับการพัฒนาโปรแกรมภาษา SQL เพื่อเชื่อมต่อกับระบบฐานข้อมูล ซึ่งคล้ายกับ ODBC และ CLI แต่สำหรับ JDBC เป็นไดรเวอร์สำหรับการใช้งานในภาษาจาวา (Java) การพัฒนาโปรแกรมประยุกต์โดยใช้ CLI จะต้องมีการเชื่อม (link) CLI libraries เข้ากับโปรแกรมประยุกต์ ในทำนองเดียวกันใน JDBC แพคเกจของภาษาจาวา ที่เกี่ยวข้องกับ JDBC APIs จะต้องถูก import เข้ามาในโปรแกรมก่อน

ตัวอย่างของ การใช้คำสั่ง SELECT ในการพัฒนา โปรแกรมประยุกต์ JDBC ซึ่งแสดงในรายการที่ 7.4ดังต่อไปนี้

```
// SQL for SELECT. The name, country, street and province information
// which has been provided by the user are stored in respective variables.
String sqlSel = “select “+name+”, ” +country+”, “+street+”, “+province+”, “+zip+”
from CUSTOMER where Customer = ?”;

//prepare the SELECT statement
try {
        PreparedStatement pstmt=con.prepareStatement(sqlSel);
```

```
            pstmt.setString (1, “custCountry”);      //set the Input parameter
            pstmt.execute();                                    //execute SELECT statement
            ResultSet result = pstmt.getResultSet ();  //get the results and set
                                                //values

            List<Customer> custList = new ArrayList<Customer>();

            while (result.next ()) {
                    Customer cust = new Customer();
                    cust.name = result.getString (1);
                    cust.country = result.getString (2);
                    cust.street = result.getString (3);
                    cust.province = result.getString (4);
                    cust.zip = result.getString (5);
                    custList.add (cust);
              }
            } catch (SQLException e) {e.pringStackTrace ();
      }
```

รายการที่ 7.4 ตัวอย่างโปรแกรมโค๊ดสำหรับ JDBC

จากตัวอย่างโปรแกรม ข้างต้น จะเห็นว่า

- คำสั่ง SQL ที่สมบูรณ์จะจัดเก็บอยู่ในตัวแปรสตริง (sqlSel)
- คำสั่ง SQL แบบ Dynamic จะถูกเตรียมโดยคำสั่ง prepare โดยการใช้ parameter marker (เครื่องหมาย ?) แทนที่การใช้ค่าของคอลัมน์
- ก่อนที่จะ ประมวลผลคำสั่ง SQL นี้ ค่าของคอลัมน์ จะถูกนำไปแทนที่ parameter marker เช่นในตัวอย่างคำสั่ง JDBC pstmt.setString(1, “custCountry”) จะแทนที่ parameter marker ตัวแรก ค่าของตัวแปร custCountry
- และในขั้นตอนสุดท้าย หลังจากที่มีการ ประมวลผลคำสั่งแล้ว ผลลัพธ์ที่ได้จะถูกนำไปเก็บในออบเจ็กต์ ResultSet ที่ชื่อว่า result

7.5 pureQuery

ถึงแม้ว่า Dynamic SQL ช่วยเพิ่ม ความยืดหยุ่นสำหรับนักพัฒนาโปรแกรม แต่อย่างไรก็ตามมันจะเกิด overhead ขึ้น เพราะในแต่ละคำสั่ง SQL ที่ ประมวลผลนั้น จะต้องถูกจัดเตรียมด้วยคำสั่ง prepare ซึ่ง ค่าของคอลัมน์ จะแทนที่ parameter marker และผลลัพธ์ที่ส่งกลับโดย JDBC จะถูกจับคู่กับออบเจ็กต์ของโปรแกรมประยุกต์จาวา

pureQuery เป็นรูปแบบหนึ่งที่ช่วยให้ นักพัฒนาโปรแกรมภาษาจาวา ใช้ประโยชน์จากคำสั่ง SQL แบบ Dynamic โดยที่ไม่ต้องคำนึงถึง ขั้นตอนของการเตรียมคำสั่ง SQL รวมถึงการจับคู่ ระหว่าง ค่าของตัวแปรและออบเจ็กต์ของโปรแกรมประยุกต์ภาษาจาวา เปรียบเทียบกับตัวอย่างโปรแกรมประยุกต์ JDBC ในรายการที่ 7.4 คำสั่ง SELECT เดียวกันที่พัฒนาขึ้นโดยใช้ pureQuery จะมีจำนวนบรรทัดของโค๊ดที่น้อยกว่า ดังแสดงใน รายการที่ 7.5 ดังต่อไปนี้

```
//The name, country, street and province are variables which would be
//populated at runtime using user inputs.
String sqlSel = “select “+name+”, ” +country+”, “+street+”, “+province+”, “+zip+”
from CUSTOMER where Customer = ?”;

Data data = DataFactory.getData (con);

//execute the Select and get the list of customer
List<Customer> customerList = data.queryList (sqlSel, Customer.class, “custCoun-
try”);
```

รายการที่ 7.5 ตัวอย่างโปรแกรมที่ใช้ pureQuery

จากตัวอย่างข้างต้นจะเห็นว่า pureQuery API queryList จะ ประมวลผลคำสั่ง sqlSel ด้วยค่าที่ระบุใน “custCountry” และจะส่งผลลัพธ์จากการ สอบถามข้อมูลกลับเข้าไปใน cutomerList คล้ายกับการทำงานของ JDBC โดยที่คำสั่ง SQL จะถูกสร้างขึ้นขณะประมวลผล (runtime) แต่จำนวนบรรทัดของโปรแกรมจะน้อยกว่าเมื่อเทียบกับการเขียนโปรแกรมประยุกต์โดยใช้ JDBC ซึ่งไม่เพียงแต่ช่วยเพิ่มความเร็วในการพัฒนาโปรแกรมประยุกต์ แต่ยังช่วยในการลดความซับซ้อนของโปรแกรมประยุกต์ อีกด้วย

วิธีการของ pureQuery ข้างต้นจะเรียกว่าเป็นวิธีการแบบอินไลน์ (inline method) ซึ่งสนับสนุนการประมวลผล SQL แบบ Dynamic นอกจากนี้ pureQuery ยังสนับสนุนการประมวลผล SQL แบบ static โดยใช้วิธีการเขียนคำสั่งแบบย่อ (annotated method)

ด้วยวิธีการแบบอินไลน์ คำสั่ง SQL ที่ถูกสร้างขึ้นเป็นออบเจ็กต์สตริงในภาษาจาวา และส่งผ่านไปยัง API ของ pureQuery และในทางกลับกัน วิธีการเขียน คำสั่งแบบย่อ ทำให้ SQL string จะถูกกำหนดให้เป็น pureQuery annotation ดังตัวอย่างต่อไปนี้

- @Select (สำหรับการ สอบถามข้อมูล)
- @Update (สำหรับคำสั่ง SQL ที่ใช้จัดการข้อมูล Data Manipulation Language: DML)
- @Call (สำหรับคำสั่ง SQL ในการเรียกคำสั่ง CALL)

พิจารณาจากคำสั่ง SELECT ต่อไปนี้

SELECT Name, Country, Street, Province, Zip FROM customer where Customer = ?

สิ่งที่จำเป็นสำหรับ pureQuery มีดังต่อไปนี้

- วางคำสั่ง SQL Select ให้อยู่ใน annotation ที่เหมาะสม
- ทำการประกาศฟังก์ชันที่ผู้ใช้กำหนด (user-defined function) ที่ใช้สำหรับการรันคำสั่ง SQL

ตัวอย่าง ใน รายการที่ 7.6 แสดงคำสั่ง SQL SELECT ในรูปแบบ @Select annotation

```
public interface CustomerData
{
    //Select PDQ_SC.CUSTOMER by parameters and populate Customer bean with
```

```
        results
        @Select(sql="select Name, Country, Street, Province, Zip from CUSTOMER where
        Customer =?")
      Customer getCustomer(int cid);
    }
```

รายการที่ 7.6 ตัวอย่างการเขียนคำสั่ง @Select

เมื่อทำการประมวลผลคำสั่ง SQL ข้างต้นโปรแกรมประยุกต์ไม่จำเป็นต้องติดตั้งอินเตอร์เฟซ CustomerData เอง โดย pureQuery จะทำการสร้างอินเตอร์เฟซ ด้วยการใช้ยูทิลิตี้ที่มีมาให้ใน pureQuery ที่ชื่อว่า pureQuery generator

โปรแกรมประยุกต์ สามารถสร้างตัวแปรอินสแตนซ์ (instance variable) สำหรับ CustomerData โดยใช้ pureQuery API และทำการเรียก getCustomer() โดยตรงเพื่อ ประมวลผลคำสั่ง SQL ดังกล่าว ดังจะปรากฏในตัวอย่างของรายการที่ 7.7 ต่อไปนี้

```
// use the DataFactory to instantiate the user defined interface
CustomerData cd = DataFactory.getData(CustomerData.class, con);
// execute the SQL for getCustomer() and get the results in Customer beans
Iterator<Customer> cust = cd.getCustomer();
```

รายการที่ 7.7 – การประมวลผลคำสั่ง SQL ที่ใช้ pureQuery และ annotation method

ผลลัพธ์ที่ได้จากเครื่องมือ pureQuery generatorจะเป็น ไฟล์จาวา (CustomerDataImpl.java ดังแสดงในตัวอย่างข้างต้น) ซึ่งเป็นส่วนหนึ่งของอินเตอร์เฟซที่สร้างขึ้นมา (user-defined interface - CustomerData) โดยในไฟล์จาวาดังกล่าว จะมีคำสั่ง SQL และเมทธอดที่ถูกสร้างขึ้นมา (declare method) ชื่อว่า getCustomer

รูปแบบของการพัฒนาโปรแกรมลักษณะนี้ทำให้นักพัฒนาสามารถระบุคำสั่ง SQL และ เมทธอดที่เกี่ยวข้องภายในอินเตอร์เฟซ โดยเมทธอดเหล่านี้จะใช้สำหรับการ ประมวลผลคำสั่ง SQL ในโปรแกรมประยุกต์ วิธีนี้จะทำให้คำสั่ง SQL ถูกแยกออกจากตรรกะ ทางธุรกิจ (Business Logic) ในโปรแกรมประยุกต์ นอกจากนี้การเขียนโปรแกรมโดยใช้การเขียนแบบย่อของ pureQuery ยังสนับสนุนการทำงานของโปรแกรมแบบ Static ในขณะที่เมทธอดแบบอินไลน์นั้นจะยังคงสนับสนุนการทำงานแบบ Dynamicอยู่ ซึ่งทำให้นักพัฒนาโปรแกรมสามารถพัฒนาโปรแกรม ประยุกต์ได้ตรงตามความต้องการของผู้ใช้งาน

สำหรับรายละเอียดเพิ่มเติมเกี่ยวกับเทคนิคการเขียนโปรแกรม pureQuery ให้ดูที่ลิงก์ต่อไปนี้ :
http://publib.boulder.ibm.com/infocenter/idm/v2r2/index.jsp?topic=/com.ibm.datatools.javatool.runtime.overview.doc/topics/helpindex_pq_sdf.html

7.5.1 IBM pureQuery Client Optimizer

คุณสมบัติของ pureQuery ที่น่าสนใจอีกอย่างหนึ่งคือ pureQuery Client Optimizer ซึ่งเราได้นำเสนอไปแล้วว่า โปรแกรมประยุกต์ที่ใช้ SQL แบบ Dynamic จะมีขั้นตอนของการเตรียมคำสั่ง SQL รวมถึงต้องจับคู่ ค่าของตัวแปร กับออบเจ็กต์ของโปรแกรมประยุกต์จาวา ซึ่ง pureQuery Client Optimizer มีเทคนิคที่น่าสนใจในการลด ประเด็นดังกล่าว โดยโปรแกรมประยุกต์ JDBC แบบ Dynamic ที่มีอยู่จะสามารถ สร้าง

ประโยชน์ได้สูงสุด โดยสามารถประมวลผลคำสั่ง SQL แบบ Static โดยที่ไม่จำเป็นต้องมีการเปลี่ยนแปลงใดๆ ในโปรแกรมประยุกต์ ซึ่งการใช้ pureQuery Client Optimizer จะต้องมีการดำเนินการต่อไปนี้

- เมื่อมีการ ประมวลผลโปรแกรมแบบไดนามิก เป็นครั้งแรก เครื่องมือ pureQuery Client Optimizer จะทำการดักจับคำสั่ง SQL ที่ส่งมาจากโปรแกรม และทำการบันทึกลงในไฟล์ชื่อว่า pureQueryXml
- คำสั่ง SQL ที่ อยู่ในไฟล์ XML จะถูกแบ่งออกเป็นแพค เกจต่างๆ โดยมีการเรียกคำสั่งที่เป็น Command Line ชื่อว่า Configure
- หลังจากนั้นจะ เรียก staticBinder ซึ่งจะสร้างแพค เกจใหม่ในฐานข้อมูล และโปรแกรมประยุกต์จะเชื่อมโยง ไปยัง แพคเกจดังกล่าว
- ขั้นตอนสุดท้าย เป็นการกำหนดการประมวลผลของรันโปรแกรมประยุกต์ ให้ทำงานแบบ static เพื่อเป็นการยอมให้คำสั่ง SQL บางคำสั่งทำงานในโหมด Static

โดยทั่วไป คำสั่ง SQL ที่ถูก ส่งมาจากโปรแกรมประยุกต์จะถูกนำไป เปรียบเทียบกับคำสั่ง SQL ที่จัดเก็บอยู่ในไฟล์ pueryQueryXml ซึ่งหากพบว่าคำสั่งที่ถูกส่งมา ตรงกันกับ คำสั่งที่จัดเก็บอยู่ในไฟล์ ก็จะทำการเรียกแพค เกจที่ จัดเก็บอยู่ในไฟล์นั้นขึ้นมาทำงาน ซึ่ง จะทำให้คำสั่ง SQL สามารถทำงานแบบ Static ได้ แต่ถ้าหากเปรียบเทียบแล้วคำสั่ง SQL ดังกล่าวไม่เหมือนกับคำสั่ง SQL ที่จัดเก็บในไฟล์ ก็จะใช้วิธีการ ประมวลผลคำสั่งนั้นๆแบบ Dynamic แทน

7.6 บทสรุป

ในบทนี้ได้กล่าวถึงเทคนิคต่างๆสำหรับการใช้คำสั่ง SQL ในโปรแกรมประยุกต์ ซึ่งได้เริ่มต้นจากการอธิบายแนวคิดของ transaction ในฐานข้อมูล ตามด้วยการเขียนคำสั่ง SQL แบบฝังตัวในโปรแกรมประยุกต์ และได้อธิบายถึงความแตกต่างระหว่างการทำงานของคำสั่ง SQL แบบ Static และแบบ Dynamic เพื่อให้เห็นถึงประสิทธิภาพการทำงาน จะเห็นว่า SQL แบบ Static นั้นจะทำให้ได้ประสิทธิภาพที่ดี ในขณะที่ SQL แบบ Dynamic นั้นจะช่วยทำให้การพัฒนาโปรแกรมประยุกต์มีความยืดหยุ่นในการทำงานมากยิ่งขึ้น อีกทั้งความสามารถ พัฒนาและ ประมวลผลโปรแกรมประยุกต์โดยที่ไม่ ต้องทำการแปลภาษาล่วงหน้า และต้องเชื่อมต่อกับฐานข้อมูล ซึ่งการเลือกที่จะใช้งาน SQL ทั้งสองโหมดนี้ขึ้นอยู่กับรูปแบบการออกแบบ การพัฒนา และความต้องการของโปรแกรมประยุกต์ ซึ่งไม่มีกฎที่แน่นอนว่าเราต้องเลือกการทำงานของ SQL ในโหมดไหน

โปรแกรม SQL แบบฝังตัว และ SQLJ สนับสนุนทั้งแบบ Static และ Dynamic โปรแกรมประยุกต์ส่วนใหญ่ ยังคงทำงานเป็นแบบ Dynamic แต่ก็ยังคงต้องใช้คำสั่ง SQL แบบ Static อยู่ ซึ่ง หมายความว่าโปรแกรมประยุกต์ยังจำเป็นต้องมีการ แปลภาษาล่วงหน้า และต้องเชื่อมต่อกับระบบฐานข้อมูล

เรายังสามารถใช้วิธีการแบบอื่น เช่น การใช้ API เช่น ODBC, CLI และ JDBC ซึ่งเป็นการพัฒนาโปรแกรม SQL แบบ Dynamic ทั้งหมด ซึ่งวิธีเหล่านี้ช่วยแก้ไขปัญหาที่พบใน การพัฒนาคำสั่ง SQL แบบฝังตัว สิ่งที่เป็นข้อได้เปรียบคือนักพัฒนาโปรแกรมสามารถเขียนโปรแกรมประยุกต์ที่เป็นมาตรฐานสำหรับเชื่อมต่อกับระบบฐานข้อมูลเชิงสัมพันธ์ใดๆ ก็ได้โดยที่ แก้ไขโปรแกรม เพียงเล็กน้อย

ในส่วนสุดท้ายของบทนี้ ได้อธิบายถึงการทำงานของ pureQuery ซึ่งเป็นเทคโนโลยีหนึ่งของบริษัทไอบีเอ็ม ที่ช่วยทำให้เราได้ใช้ประโยชน์จากการพัฒนาโปรแกรม SQL แบบ Dynamic โดยที่นักพัฒนาไม่ต้องกังวลเกี่ยวกับการเตรียมคำสั่ง (prepare) และปัญหาของทำออบเจ็กต์แม็บระหว่างฐานข้อมูลกับโปรแกรมซอฟต์แวร์

7.7 แบบฝึกหัด

ให้ทำการออกแบบระบบการจัดการห้องสมุด เพื่อให้สามารถรองรับการดำเนินการดังต่อไปนี้

A. การปรับปรุงหนังสือเล่มใหม่ลงในฐานข้อมูล
B. การค้นหาหนังสือตามชื่อผู้แต่งหรือชื่อเรื่อง
C. การปรับปรุงข้อมูลหนังสือที่ถูกยืมและคืนในฐานข้อมูล โดยการระบุวันที่คืน
D. สามารถแสดงรายชื่อหนังสือที่ต้องคืน โดยการระบุวันที่ที่ถึงกำหนดคืน

และในระบบนี้ ให้ทำการเลือกดูว่ามี การสอบถามข้อมูลใดบ้างที่เหมาะสำหรับการทำงานแบบDynamic และแบบใดที่เหมาะกับการดำเนินการแบบ Static

7.8 คำถามท้ายบท

1. การทำงานของ SQL แบบ Static จะหมายความถึงแผนการเข้าถึงข้อมูลแบบใด
   A. จะได้รับการสร้างเมื่อโปรแกรมประยุกต์ ทำการประมวลผล (runtime)
   B. จะถูกสร้างขึ้นในช่วงเวลาที่ทำการแปลภาษาล่วงหน้า (pre-compile)
   C. ถูกทั้งหมด
   D. ไม่มีข้อใดถูก

2. การทำงานแบบไดนามิก SQL เป็นอย่างไร
   A. เป็นคำสั่ง SQL ที่ไม่มีไวยากรณ์ที่สมบูรณ์มาก่อน
   B. เป็นคำสั่ง SQL ที่ต้องมีไวยากรณ์สมบูรณ์ในขณะที่ ทำการแปลภาษาล่วงหน้า
   C. ถูกทั้งหมด
   D. ไม่มีข้อใดถูก

3. โปรแกรม C และ C++ ที่มีคำสั่ง SQL แบบฝังตัวเป็นอย่างไร
   A. ไม่จำเป็นต้องใช้ DB2 pre-compiler
   B. จำเป็นต้องใช้ DB2 pre-compiler

4. SQLJ คืออะไร
   D. เทคนิคการทำงานของ SQL แบบฝังตัวสำหรับโปรแกรมจาวา
   E. เป็น Call Level Interface สำหรับการพัฒนา SQL

5. ODBC ย่อมาจากอะไร
   E. Open Database Community
   F. Open Database Connectivity
   G. Open source database community
   H. Open database connection
   I. ไม่มีข้อใดถูก

6. ข้อใดดังต่อไปนี้ที่สนับสนุนการทำงานทั้งแบบ Static และแบบ Dynamic
   A. JDBC
   B. SQLJ
   C. DB2 CLI

D. ถูกทั้งหมด
E. ไม่มีข้อใดถูก

7. เราจำเป็นต้อง เชื่อมต่อกับฐานข้อมูลในกรณีที่ทำการแปลภาษาล่วงหน้า สำหรับโปรแกรมประยุกต์แบบใด
A. JDBC
B. SQLJ
C. DB2 CLI
D. ถูกทั้งหมด
E. ไม่มีข้อใดถูก

8. เครื่องหมายของพารามิเตอร์ ('?') สามารถใช้เป็นการจัดเก็บค่าสำหรับข้อใด
H. ค่าของคอลัมน์ที่มีการระบุค่าในขณะที่ทำการประมวลผล (runtime)
I. ชื่อคอลัมน์ และชื่อตาราง
J. สำหรับคำสั่ง SQL
K. ถูกทั้งหมด
L. ไม่มีข้อใดถูก

9. pureQuery ที่ใช้วิธีการเขียนคำสั่งแบบย่อ (annotated method) สนับสนุน
I. คำสั่ง SQL แบบ Dynamic
J. คำสั่ง SQL แบบ Static
K. ถูกทั้งหมด
L. ไม่มีข้อใดถูก

10. pureQuery Client Optimizer จะต้อง
J. ต้องมีการปรับเปลี่ยนโปรแกรมโค๊ด JDBC เพื่อให้สามารถทำงานแบบ Static
K. ไม่ต้องมีการปรับเปลี่ยนโปรแกรมโค๊ดที่ทำงานแบบ JDBC ใดๆ เพื่อให้สามารถทำงานแบบ Static

8

บทที่ 8 – ภาษาในการสอบถามข้อมูลสำหรับ XML

โลกทุกวันนี้ ข้อมูลถูกนำเสนอในรูปแบบต่างๆ ไม่ว่าจะเป็นแบบจำลองเชิงสัมพันธ์ (relational model) ซึ่งเป็นแบบจำลอง ดั้งเดิมที่ถูกใช้งานมามากกว่าทศวรรษ แต่ด้วยความต้องการที่เปลี่ยนไป ทำให้มีความเกี่ยวข้องกับข้อมูลจากแหล่งข้อมูลอื่น ๆ ซึ่งข้อมูลดังกล่าวอาจไม่ได้อยู่ในรูปแบบโครงสร้าง เชิงสัมพันธ์ (structure relational) เสมอไป แต่ในความเป็นจริงข้อมูลดังกล่าว จะเป็นแบบกึ่งโครงสร้าง (semi-structure) หรือรูปแบบที่ไม่มีโครงสร้าง (unstructured) ซึ่งข้อมูลที่ไม่มีโครงสร้างเหล่านี้สามารถเป็นได้ทั้งรูปภาพและภาพเคลื่อนไหว ในขณะข้อมูลที่มีโครงสร้างเชิงสัมพันธ์ จะมาพร้อมกับโครงสร้างของข้อมูล ส่วนข้อมูลที่อยู่ในรูปแบบกึ่งโครงสร้างจะไม่มีโครงสร้างของข้อมูลที่แน่นอน เหมือนกับแถวในตาราง และการจัดเก็บข้อมูลมีความยืดหยุ่นมากขึ้น เพื่อให้สามารถนำเสนอโครงสร้างของข้อมูลได้หลากหลายมากขึ้น

XML เป็นวิธีหนึ่งที่ใช้ในการนำเสนอข้อมูลที่อยู่ในรูปแบบกึ่งโครงสร้าง (semi-structure) และมีการใช้งานที่ประสบผลสำเร็จ ปัจจุบัน XML ได้กลายเป็นมาตรฐานสำหรับการแลกเปลี่ยนข้อมูลบนอินเทอร์เน็ต เพื่อรองรับการทำธุรกรรมต่างๆแบบออนไลน์ทั่วทั้งอินเทอร์เน็ต ซึ่งธุรกรรมที่เกิดขึ้นนั้นต้องมีการตรวจสอบติดตาม และเก็บข้อมูลของธุรกรรมทั้งหมดเหล่านี้ และเก็บรักษาข้อมูลเหล่านี้ไว้สำหรับการใช้งานในอนาคต ในบางองค์กรมีความต้องการจัดเก็บข้อมูลสำหรับการตรวจสอบ และข้อมูลในการปฏิบัติงาน หรือข้อมูลอื่น ๆ เพื่อใช้ในการวิเคราะห์ข้อมูล สำหรับการแข่งขันในเชิงธุรกิจ จึงทำให้เกิดความจำเป็นที่จะต้องมีระบบฐานข้อมูลที่มีประสิทธิภาพสูงที่ สนับสนุนในการจัดเก็บ และการสอบถามข้อมูลเหล่านั้น ซึ่งอยู่ในรูปแบบของ XML ที่มีขนาดใหญ่ได้อย่างมีประสิทธิภาพ DB2 เป็น ฐานข้อมูลที่ให้การสนับสนุน ทั้งข้อมูลที่เป็นโครงสร้างแบบจำลอง เชิงสัมพันธ์ และข้อมูลที่เป็นโครงสร้างแบบ XML ซึ่งเรียกว่า เป็นฐานข้อมูลลูกผสม (hybrid data server) ได้อย่างผสมผสาน

8.1 ภาพรวมของ XML

XML ย่อมาจาก eXtensible Markup Language XML เป็นรูปแบบของข้อมูลแบบลำดับชั้น (hierarchical) ประกอบด้วย nodes หลายชนิดที่เชื่อมโยงเข้าด้วยกันผ่านทางความสัมพันธ์แบบ parent/child แบบจำลองข้อมูล XML ยังสามารถแสดงในรูปแบบของข้อความ หรือในรูปแบบของไบนารีได้

8.1.1 XML element และออบเจ็กต์ฐานข้อมูล

XML element เป็นส่วนประกอบที่สำคัญของเอกสาร XML (XML document) โดยทุกเอกสาร XML ต้องมี XML element อย่างน้อยหนึ่ง XML element ซึ่งเราเรียกว่า root หรือ document element element เหล่านี้สามารถ ที่จะมีแอตทริบิวต์ (attribute) หรือ child element

ตัวอย่างต่อไปนี้เป็นตัวอย่างแบบง่ายๆ ของ XML element

```
<name>I am an XML element</name>
```

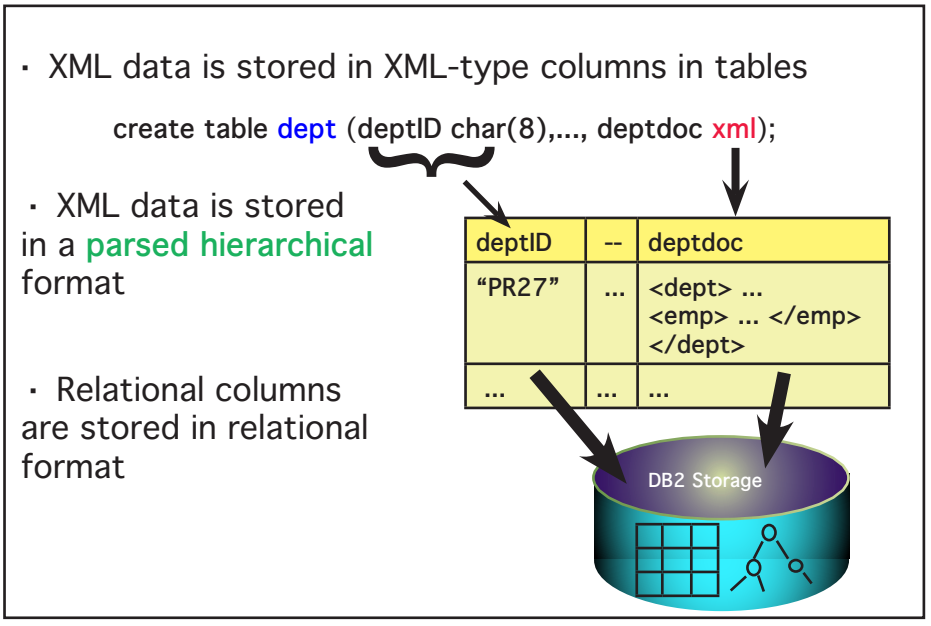
ทุก XML element จะต้องมีแท็กสำหรับจุดเริ่มต้น (start tag) และแท็กสำหรับจุดสิ้นสุด (end tag) ในตัวอย่างข้างต้น <name> คือ แท็กสำหรับจุดเริ่มต้น และ </name> คือแท็กสำหรับจุดสิ้นสุด element จะมีค่าที่เป็นข้อความ (text value) ว่า “I am an XML element” ตัวอย่างต่อไปนี้เป็นเอกสาร XML ประกอบด้วย XML element ต่าง ๆ และแอตทริบิวต์

```
<employees>
        <employee id=”121”>
        <firstname>Jay</firstname>
        <lastname>Kumar</lastname>
        <job>Asst. manager</job>
        <doj>2002-12-12</doj>
        </employee>
</employees>
```

จากตัวอย่างข้างต้น <employees> เป็น document element ของเอกสาร XML ที่มี child element <employee> ซึ่ง element <employee> มี child element หลายๆ element และยังมี แอตทริบิวต์ที่มีชื่อว่า id ซึ่งมีค่าเท่ากับ 121 โดยใน DB2 เอกสาร XML ทั้งหมด จะถูกจัดเก็บไว้ในคอลัมน์เดียว ตัวอย่างเช่น โครงสร้างตารางด้านล่างนี้

```
department(id integer, deptdoc xml)
```

คอลัมน์ *id* จะจัดเก็บค่าจำนวนเต็มบวกของรหัสแต่ละแผนก ในขณะที่คอลัมน์ **deptdoc** ซึ่งเป็น ชนิดของข้อมูล XML จะจัดเก็บเอกสาร XML หนึ่งเอกสารสำหรับแต่ละแผนก ดังนั้น เอกสาร XML ทั้งหมดจะถูก มองเสมือนเป็นออบเจ็กต์เดียว ซึ่งรูปที่ 8.1 ต่อไปนี้แสดงถึงวิธีการจัดเก็บข้อมูล XML ใน DB2



รูปที่ 8.1 – การจัดเก็บข้อมูล XML ใน DB2

ใน DB2 เอกสาร XML จะถูกวิเคราะห์ไวยากรณ์ในประโยค (parsed) ในระหว่างการเก็บข้อมูล และจะถูกแยกจัดเก็บในรูปแบบลำดับชั้น (hierarchical format) ภายในฐานข้อมูล จึงทำให้ DB2 ไม่จำเป็นต้องวิเคราะห์ไวยากรณ์ในประโยคของเอกสาร XML ในระหว่างการสอบถามข้อมูล ด้วยเหตุนี้จึงทำให้ประสิทธิภาพในการสอบถามข้อมูลได้ดีขึ้น

8.1.2 แอตทริบิวต์ของ XML (XML attribute)

แอตทริบิวต์เป็นส่วนหนึ่งของ element เสมอ และเป็นส่วนที่ให้ข้อมูลเพิ่มเติมต่างๆที่เกี่ยวข้องกับ element นั้นๆ ตัวอย่างต่อไปนี้แสดง element ที่มีสอง แอตทริบิวต์ คือ id และ uid ซึ่งมีค่า 100-233-03 และ 45 ตามลำดับ

```
<product id=”100-233-03” uid=”45”/>
```

หมายเหตุ สำหรับแอตทริบิวต์ ของแต่ละ element จะต้องมีชื่อที่ไม่ซ้ำกัน กล่าวคือ element เดียวกันจะต้องไม่มี แอตทริบิวต์สองตัวที่มีชื่อซ้ำ กัน

ตัวอย่าง การประกาศ element ต่อไปนี้จะทำให้เกิดข้อผิดพลาด

```
<product id=”100-233-03” id=”10023303”/>
```

หมายเหตุ ค่าของ แอตทริบิวต์ ต้องอยู่ในเครื่องหมายคำพูดเสมอ

ในข้อกำหนดโครงสร้างของ XML (XML schema definition) แอตทริบิวต์จะถูกกำหนดหลังจากที่มีการกำหนด element ทั้งหมดแล้ว ตัวอย่างต่อไปนี้เป็นตัวอย่างของ element ที่ซับซ้อนที่ประกอบด้วยหนึ่งแอตทริบิวต์และสอง child element

```
<xs:element name=”employee”>
  <xs:complexType>
    <xs:sequence>
      <xs:element name=”firstname” type=”xs:string”/>
```

```
    <xs:element name="lastname" type="xs:string"/>
   </xs:sequence>
  <xs:attribute name="empid" type="xs:integer"/>
 </xs:complexType>
</xs:element>
```

8.1.3 Namespaces

XML namespaces เป็นกลไกที่ช่วย กำหนดความเหมาะสมในการกำหนดชื่อของ แอตทริบิวต์ และ element ให้มีประสิทธิภาพ เพื่อหลีกเลี่ยงความขัดแย้งในการตั้งชื่อในเอกสาร XML ตัวอย่างเช่น ถ้าบริษัทประกันสุขภาพได้รับข้อมูลผู้ประกันตนจากบริษัทอื่นเป็นเอกสาร XML ซึ่งเป็นไปได้ค่อนข้างสูงว่า ข้อมูลที่ได้รับจากหลายๆ บริษัทนั้นจะมีชื่อ element ที่เหมือนกันแต่อาจจะ เป็น สิ่ง หรือ รูปแบบที่ต่างกัน การกำหนดความเหมาะสมของชื่อ element ด้วย namespace นี้จะเข้ามาแก้ไขปัญหาการกำหนดชื่อ element ที่ซ้ำซ้อน ใน XML ชื่อสามารถกำหนดความเหมาะสมโดย namespace ชื่อที่มีความเหมาะสม จะมีองค์ประกอบอยู่สองส่วน คือ

- Namespace Uniform Resource Identifier (URI)
- Local name

ตัวอย่างเช่น **http://www.acme.com/names** เป็น namespace URI และ customer เป็น local name โดยส่วนใหญ่การกำหนดชื่อที่เหมาะสมจะใช้ prefix แทนที่ URI ตัวอย่างเช่น customer element ที่เป็นส่วนหนึ่งของ **http://www.acme.com/names** URI สามารถเขียนได้เป็น acme:customer โดยที่ acme ที่อยู่นำหน้าจะเป็น namespace prefix สำหรับ **http://www.acme.com/names** ซึ่ง namespace prefix สามารถถูกประกาศในภาษา XQuery Prolog ดังที่แสดงด้านล่าง

Declare namespace acme "http://www.acme.com/names"

ทั้งนี้ namespace prefix สามารถถูกประกาศใน element constructors ดังนี้

<book xmlns:acme="**http://www.acme.com/names**">

หมายเหตุ การประกาศ namespace ใช้ขอบเขตเดียวกันกับการประกาศ element โดยที่ child element ทั้งหมดสามารถ อ้างอิงถึง namespace ที่ประกาศนี้ได้

โดย namespace prefix ดังต่อไปนี้ไม่สามารถ ใช้เป็น namespace prefix ที่ผู้ใช้กำหนดขึ้นมาเองได้:

Xml, xs, xsi, fn, xdt

นอกจากนี้ยังสามารถประกาศ namespace ที่เป็นค่าเริ่มต้น (default) โดยเป็น namespace ที่ไม่มี prefix ด้วยวิธีการดังต่อไปนี้:

```
 declare default element namespace 'http://www.acme.org/names'
(ในภาษา XQuery Prolog)
<book xmlns = "http://www.acme.com/names">
(ใน element constructor )
```

8.1.4 ข้อกำหนดของชนิดเอกสาร (Document Type Definition: DTD)

ข้อกำหนดของชนิดเอกสาร เป็นข้อกำหนดรายละเอียดที่จะมีในเอกสาร XML โดยจะกำหนดโครงสร้าง

ของเอกสารกับรายการของ element และ แอตทริบิวต์ทั้งหมดที่มีในเอกสาร XML ซึ่ง DTD สามารถประกาศภายในเอกสาร XML หรือการอ้างอิงจากภายนอก ตัวอย่างต่อไปนี้เป็นตัวอย่างของ DTD

```
<!DOCTYPE TVSCHEDULE [

<!ELEMENT PRODUCTS (PRODUCT+)>
<!ELEMENT PRODUCT (NAME,PRICE,DESCRIPTION)>
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT PRICE (#PCDATA)>
<!ELEMENT DESCRIPTION (#PCDATA)>

<!ATTLIST PRODUCTS ID CDATA #REQUIRED>
]>
```

ซึ่งเอกสาร XML ข้างล่างนี้จะมีความถูกต้องเมื่ออ้างอิงกับ DTD ข้างต้น

```
<PRODUCTS>
  <PRODUCT ID=”100-200-43”>
    <NAME>Laptop</NAME>
    <PRICE>699.99</PRICE>
     <DESCRIPTION>This is a Laptop with 15 inch wide screen, 4 GB RAM, 120 GB HDD </
DESCRIPTION>
  </PRODUCT>
</PRODUCTS>

<PRODUCTS>
   <PRODUCT ID=”100-200-56”>
     <NAME>Printer</NAME>
     <PRICE>69.99</PRICE>
     <DESCRIPTION>This is a line printer </DESCRIPTION>
   </PRODUCT>
   <PRODUCT ID=”100-200-89”>
     <NAME>Laptop</NAME>
     <PRICE>699.99</PRICE>
      <DESCRIPTION>This is a Laptop with 13 inch wide screen, 4 GB RAM, 360 GB HDD </
DESCRIPTION>
   </PRODUCT>
</PRODUCTS>
```

8.1.5 XML Schema

XML Schema จะเป็นการกำหนด โครงสร้าง เนื้อหา และชนิดของข้อมูลสำหรับเอกสาร XML ซึ่งสามารถมี schema document ได้มากกว่าหนึ่ง โดยที่เรา สามารถกำหนด namespace ใน schema document ได้อีกด้วย

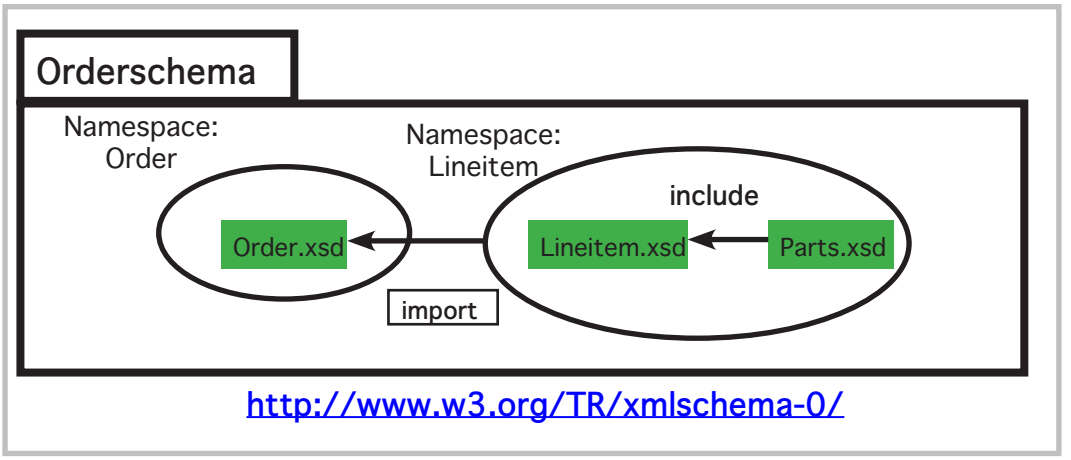
```
<xsd:schema targetNamespace="http://www.mycompany/products"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:simpleType name="PriceType">
        <xsd:restriction base="xsd:decimal">
            <xsd:minInclusive value="0"/>
            <xsd:maxInclusive value="100000"/>
            <xsd:totalDigits value="9"/>
            <xsd:fractionDigits value="3"/>
        </xsd:restriction>
      </xsd:simpleType>
      <xsd:complexType name="StockPriceType">
        <xsd:sequence>
            <xsd:element name="Ask"type="PriceType"/>
            <xsd:element name="Bid" type="PriceType"/>
            <xsd:element name="P50DayAvg" type="PriceType"/>
      </xsd:sequence>
      <xsd:complexType>
      <xsd:element name="StockPrice" type="StockPriceType"/>
 </xsd:schema>
```

XML Schema Namespace

รูปที่ 8.2 – ตัวอย่าง XML Schema

XML Schema เป็นการใช้คำสั่ง XML แทนการใช้ DTDs ซึ่ง XML Schema จะอธิบายถึงโครงสร้างของเอกสาร XML โดยภาษาที่ใช้สำหรับ XML Schema จะเรียกว่า XML Schema Definition (XSD) โดยที่ XML Schema จะมีประสิทธิภาพมากกว่า DTDs เนื่องจาก XML Schema จะมีการควบคุมที่ดีกว่าโดยจะครอบคลุมทั้งเอกสาร XML การใช้งาน XML Schema ไม่เพียงทำให้สามารถใช้งานชนิดของข้อมูลพื้นฐาน เช่น integer date decimal และ datetime เท่านั้น แต่ยังสามารถใช้งานชนิดข้อมูลที่ผู้ใช้งานกำหนดขึ้นเอง หรือชนิดของข้อมูลที่เป็น complex element ได้อีกด้วย โดยที่เราสามารถระบุ ความยาว ค่าสูงสุด ค่าต่ำสุด รูปแบบของตัวอักษรที่อนุญาตให้ใช้ และการแจกแจงรายละเอียด (enumeration) นอกจากนี้ยังสามารถระบุลำดับของ element ในเอกสาร XML ประโยชน์อีกอย่างหนึ่งของการใช้ XML Schema แทน DTDs คือ ความสามารถในการใช้ XML Namespaces นอกจากนี้ XML Schema ยังให้การสนับสนุนการสืบทอด (inheritance) ซึ่งเป็นข้อกำหนดของ W3C ซึ่งกำหนดไว้เมื่อเดือน พฤษภาคม 2544

ข้างล่างเป็นตัวอย่างของ XML Schema ซึ่งประกอบด้วยสาม schema documents และสอง namespaces



รูปที่ 8.3 - namespace หลายๆ namespace ใน XML Schema

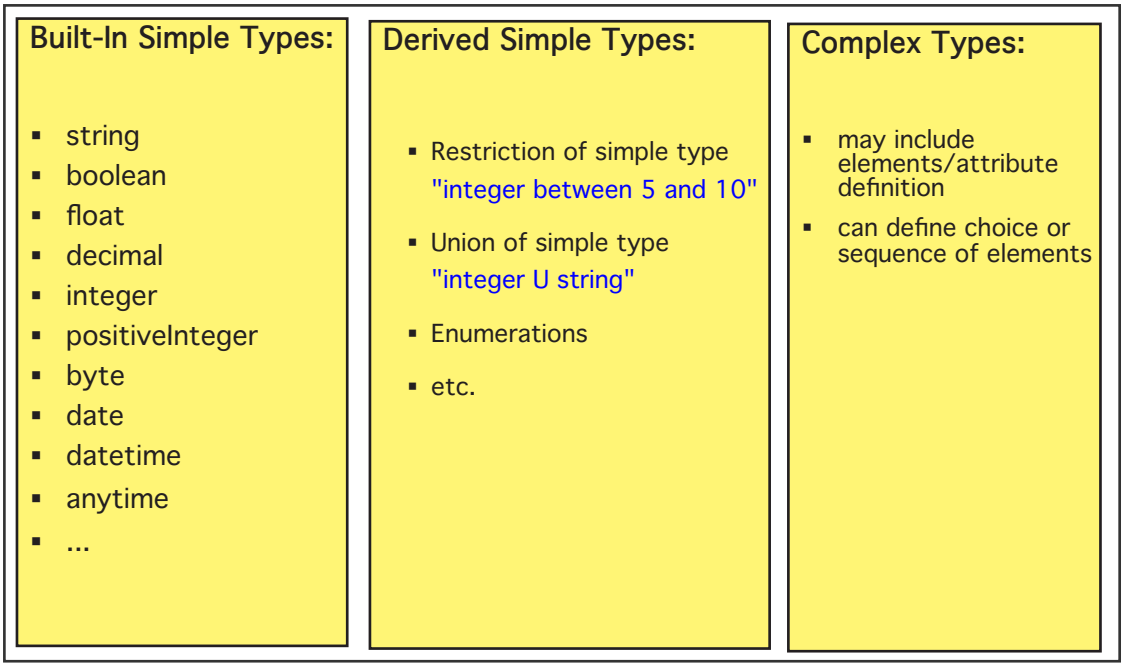
ใน DB2 การใช้งาน XML Schema เป็นทางเลือกหนึ่ง ขึ้นอยู่กับ แต่ละเอกสาร นั่นหมายถึง เราสามารถเลือกใช้คอลัมน์ XML เดียวกันจัดเก็บ เอกสาร XML ทั้งสองประเภท คือ เอกสาร XML ที่มี XML Schema และ ที่ไม่มี XML Schema ได้ ดังนั้น ไม่มีความจำเป็นสำหรับการใช้ fixed schema สำหรับแต่ละคอลัมน์ XML ซึ่งการตรวจสอบความถูกต้อง (validate) ของเอกสาร XML สามารถทำได้สำหรับแต่ละเอกสาร XML (นั่นคือทำได้ ต่อแถว) ดังนั้นเราจะไม่กำหนดให้มี XML Schema หรือ กำหนดให้มี หนึ่ง หรือมากกว่าหนึ่ง XML Schema ต่อคอลัมน์ XML ก็ได้ รวมถึง ยังสามารถเก็บเอกสาร XML ที่ไม่มี หรือ มีการตรวจสอบความถูกต้องผสมกันในคอลัมน์ XML เดียวกันได้ โดย DB2 ยังอนุญาตให้ผู้ใช้สามารถ ป้องกันการ เก็บเอกสาร XML ที่ไม่ได้ผ่านการตรวจสอบความถูกต้องโดยใช้ส่วนของคำสั่งย่อย (clause) 'IS VALIDATED' ในคำสั่ง select ได้

8.2 ภาพรวมของ XML Schema

XML เป็นส่วนหนึ่งของ Standard Generalized Markup Language (SGML) ที่มีการใช้งานที่ง่ายและมีประสิทธิภาพมาก ซึ่ง XML ไม่ได้จำกัดจำนวน namespace หรือจำกัดความซับซ้อนของโครงสร้าง โดย XML เป็นภาษาที่สนับสนุนการทำงานในตลาดอุตสาหกรรมด้านต่างๆ ซึ่ง XML จะสนับสนุนชนิดข้อมูลแทบจะทุกชนิดข้อมูล และสนับสนุน integrity constraints ซึ่งจะกล่าวถึง ต่อไป

8.2.1 ชนิดของข้อมูลทั่วไป (simple types)

XML Schema ที่กำหนดขึ้นโดย W3C ได้ระบุประเภทของชนิดข้อมูลทั่วไป ดังแสดงในรูปที่ 8.4



รูปที่ 8.4 – ชนิดของข้อมูลใน XML Schema

มีชนิดของข้อมูลทั่วไปอยู่หลายชนิดที่สามารถใช้ในการระบุชนิดของข้อมูลสำหรับ element ในเอกสาร XML นอกจากการกำหนดชนิดของข้อมูลทั่วไปสำหรับ element แล้ว ผู้ใช้ยังสามารถกำหนดชนิดของข้อมูลขึ้นมาเอง (user-defined type) ได้ โดยสร้างขึ้นจากชนิดของข้อมูลทั่วไป

ดังตัวอย่างที่แสดงไว้ด้านล่างจะเป็นชนิดของข้อมูลที่ผู้ใช้กำหนดขึ้นเอง 'myInteger' ที่สร้างจากชนิดของข้อมูล xs:integer ซึ่งอนุญาตให้ใส่เฉพาะค่าในช่วง -2 ถึง 5

```
<xs:simpleType name= “myInteger” >
<xs:restriction base= “xs:integer” >
<xs:minInclusive value = “-2” />
<xs:maxExclusive value = “5” />
</xs:restriction>
</xs:simpleType>
```

ชนิดของข้อมูลข้างต้นนี้ เรียกว่าการสร้างชนิดของข้อมูลโดยมีข้อจำกัด (derivation by restriction) ตัวอย่างต่อไปนี้ เป็นอีกตัวอย่างหนึ่งสำหรับการสร้างชนิดของข้อมูลขึ้นมาเองโดยใช้ การแจกแจงรายละเอียด (enumeration) ใน schema element :

```
<xs:simpleType name= “passGrades” >
<xs:restriction base= “xs:string” >
<xs:enumeration  value = “A” />
<xs:enumeration  value = “B” />
<xs:enumeration  value = “C” />
</xs:restriction>
</xs:simpleType>
```

สำหรับ element ใด ๆ ที่ถูกกำหนดเป็นชนิดของข้อมูล passGrades จะสามารถมีค่าได้เพียงหนึ่งในสามค่า (A, B หรือ C) ส่วนค่าอื่น ๆ สำหรับ element นี้จะถือว่าเป็นข้อผิดพลาด ที่เกี่ยวกับ XML Schema อีกตัวอย่างหนึ่งซึ่งมีการระบุรูปแบบของค่า (pattern of value) ที่สามารถจัดเก็บได้ใน element ดังแสดงไว้ในตัวอย่าง

ด้านล่างนี้:

```
<xs:simpleType name= “CapitalNames” >
<xs:restriction base= “xs:string” >
<xs:pattern  value = “([A-Z]( [a-z]*)?)+” />
</xs:restriction>
</xs:simpleType>
```

สำหรับตัวอย่างอีกสองตัวอย่างในการกำหนดชนิดของข้อมูลแบบ derivation by list และแบบ derivation by union ตัวอย่างต่อไปนี้เป็นลักษณะของ derivation by list

```
<xs:simpleType name= “myintegerList” >
<xs:list itemType= “xs:integer” />
</xs:simpleType>
```

ชนิดของข้อมูลข้างต้นนี้สามารถใช้เพื่อแอตทริบิวต์ หรือ element เพื่อให้ยอมรับเลขจำนวนเต็มที่มีช่องว่างคั่นได้ เช่น “1 234 333 -32321” ต่อไปนี้เป็นตัวอย่างของการใช้ชนิดของข้อมูลแบบ derivation by union

```
<xs:simpleType name= “intordate” >
<xs:union memberTypes= “xs:integer xs:date” />
</xs:simpleType>
```

ชนิดของข้อมูลนี้สามารถใช้เพื่อกำหนดแอตทริบิวต์ หรือ element ให้ยอมรับ เลขจำนวนเต็มที่มีช่องว่างคั่น เช่น “223 1 2001-10-26” หมายเหตุ ในกรณีนี้มีการใช้ชนิดของข้อมูลต่างชนิดกัน (integer และ date)

8.2.2 ชนิดของข้อมูลที่มีความซับซ้อน (complex types)

ชนิดของข้อมูลที่มีความซับซ้อน จะ ใช้ชนิดของข้อมูลทั่วไป ในการสร้างแต่ละ element หรือแอตทริบิวต์ที่มีความซับซ้อนขึ้น โดย element ที่มีความซับซ้อนจะประกอบ ขึ้นจาก element/แอตทริบิวต์อื่นๆ โดย element ที่มีความซับซ้อนจะสามารถเป็นค่าว่าง(empty)ได้ นอกจากนี้มันยังสามารถ ประกอบด้วย element อื่นๆ ข้อความ (text) หรือทั้งสองอย่างพร้อมด้วยแอตทริบิวต์ ตัวอย่างข้างล่างเป็นชนิดของข้อมูลที่มีความซับซ้อน

```
<xs:complexType name=”employeeType>
  <xs:sequence>
    <xs:element name=”firstname” type=”xs:string”/>
    <xs:element name=”lastname” type=”xs:string”/>
  </xs:sequence>
</xs:complexType>
```

element ที่มีความซับซ้อนข้างต้น สามารถนำไปใช้งานได้ตามตัวอย่างที่แสดงไว้ด้านล่าง

```
<xs:element name=”employee” type=“employeeType”>
```

ตัวอย่างที่แสดงไว้ด้านล่างเป็นอีกวิธีหนึ่งในการใช้งานชนิดของข้อมูลที่มีความซับซ้อนเพื่อให้ได้ผลลัพธ์เดียวกัน

```
<xs:element name=”employee”>
  <xs:complexType>
    <xs:sequence>
      <xs:element name=”firstname” type=”xs:string”/>
      <xs:element name=”lastname” type=”xs:string”/>
    </xs:sequence>
  </xs:complexType>
```

```
</xs:element>
```

ความแตกต่างของวิธีการทั้งสองคือ การใช้งานวิธีแรกจะมีความเป็นอิสระมากกว่า และสามารถนำไปใช้ซ้ำในการ กำหนด element อื่นๆได้อีก

8.2.3 Integrity Constraints

XML Schema อนุญาตให้มีการระบุและการอ้างอิงบางส่วนของข้อมูล เช่น เราสามารถ อ้างอิงถึงแอตทริบิวต์ ID และ IDREFs จาก XML DTDs โดยการใช้ XML Schema ประเภท xs:ID และ xs:IDREFs แต่อย่างไรก็ตาม XML Schema, xs:ID และ xs:IDREFs ยังสามารถใช้ได้กับ element ไม่ใช่ใช้ได้แค่แอตทริบิวต์ เหมือนกันกับกรณีของ DTDs นอกจากนี้เราสามารถ บังคับให้ element มีค่าที่ไม่ซ้ำซ้อนโดยการใช้ที่ ชนิดของ element ที่เป็น xs:unique ดังที่แสดงไว้ในตัวอย่างด้านล่าง

```
<xs:element name =”book” >
<xs:complexType>
....
</xs:complexType>
<xs:unique name=”book”>
<xs:selector xpath=”book”/>
<xs:field xpath=”isbn”/>
</xs:unique>
</xs:element>
```

ซึ่งจะทำให้มั่นใจได้ว่า ISBN จะไม่ซ้ำซ้อนกันสำหรับหนังสือแต่ละเล่มในเอกสาร XML และเรายังสามารถใช้ xs:key และ xs:keyref ในการบังคับ integrity constraints โดยที่ key จะเป็น unique constraint ที่มีข้อจำกัดอื่นๆ ซึ่งจะคล้ายกับข้อกำหนดของ element แบบ unique

```
<xs:element name =”book” >
<xs:complexType>
....
</xs:complexType>
<xs:key name=”book”>
<xs:selector xpath=”book”/>
<xs:field xpath=”isbn”/>
</xs:key>
</xs:element>
```

สำหรับ xs:keyref ใช้เพื่ออ้างถึง key นี้จากจุดปัจจุบัน (current scope)

หมายเหตุ การอ้างแอตทริบิวต์ ของ element xs:keyref ควรจะอ้างถึง xs:key หรือ element xs:unique ที่มีการกำหนด ภายใต้ element เดียวกัน หรือภายใต้ element ที่อยู่เหนือกว่า element นี้ (ancestor)

8.2.4 วิวัฒนาการของ XML Schema

เหตุผลหนึ่งที่ XML มีการใช้งานที่เพิ่มขึ้น ได้แก่ความยืดหยุ่นในการเป็นแบบจำลองข้อมูล ที่สามารถรองรับการเปลี่ยนแปลงโครงสร้างได้อย่างง่ายดาย ทุกวันนี้ อุตสาหกรรมต่างๆ ได้มีข้อกำหนดมาตรฐานอุตสาหกรรมในรูปแบบของเอกสาร XML Schema เพื่อให้เป็นเอกสารมาตรฐานของอุตสาหกรรม ซึ่งต้องสามารถปรับแก้ไขได้อย่างรวดเร็ว โดยการเปลี่ยนแปลงเหล่านี้จะต้องไม่ส่งผลกระทบใดๆต่อโปรแกรมประยุกต์ ซึ่ง DB2 นั้นมีความสามารถที่จะทำให้เกิดความยืดหยุ่นในการจัดการการเปลี่ยนแปลง XML Schema เหล่านี้ DB2 สามารถจัดเก็บเอกสาร XML ที่มี หรือไม่มี XML Schema ไว้ ภายในคอลัมน์เดียวกัน นอกจากนี้ DB2 ยัง

สามารถเก็บเอกสาร XML ที่มี XML Schema ที่แตกต่างกันได้ภายในคอลัมน์ XML เดียวกัน และ DB2 ยังมีฟังก์ชันในการตรวจสอบความเข้ากันได้ (compatibility) ของสองเวอร์ชันของ XML Schema ถ้าพบว่าเข้ากันได้ DB2 สามารถ ปรับปรุง Schema ปัจจุบัน ด้วย Schema ที่ใหม่กว่า

ถ้าคุณ ต้องการตรวจสอบความถูกต้องของเอกสาร XML ที่จัดเก็บไว้ เทียบกับ Schema ที่มีการเปลี่ยนแปลงไป สามารถทำได้สองวิธี ใน DB2 pureXML:

- หาก Schema ทั้งสองมีความเหมือนกันพอสมควร (compatible) เราสามารถ register schema ใหม่ใน XML Schema Repository (XSR) โดยการแทนที่ schema เดิม และดำเนินการตรวจสอบความถูกต้องต่อไป ชื่อ ของ schema (ชื่อ SQL และโครงสร้างที่อยู่ใน URI) จะยังคง เป็นชื่อเดิม สำหรับทั้งสอง schema
- ในกรณีที่ XML Schema ทั้งสองไม่เหมือนกัน (not compatible) เราสามารถ register schema ใหม่ ด้วยชื่อ SQL ใหม่และตำแหน่งโครงสร้าง ใน URI ใหม่

หลังจากทำการ ปรับปรุง schema ใหม่ที่เข้ากันได้เมื่อใช้ XMLVALIDATE แล้วเรายังสามารถดำเนินการต่อเพื่ออ้างถึง XML SChema ใหม่โดยใช้ชื่อ SQL ที่มีอยู่เดิม หรือสามารถใช้ตำแหน่งโครงสร้างใน URI ใน XML instance document โดยที่ URI ยังคงไม่เปลี่ยนแปลงทั้งในส่วน XML instance document ที่มีอยู่เดิม หรือของใหม่ โดยทั่วไปแล้ว วิธีข้างต้นนี้จะถูกใช้ก็ต่อเมื่อมีการเปลี่ยนแปลงใน schema เพียงเล็กน้อย

ลองพิจารณาถึงกรณีที่มีการเปลี่ยนแปลง schema เพียงเล็กน้อย ขั้นตอนในการดำเนินงานจะเป็นการแทนที่โครงสร้างที่มีอยู่เดิม ด้วยโครงสร้างที่แก้ไขใหม่เพื่อเป็นการ ปรับปรุง XML Schema ใน XSR ซึ่งสามารถทำได้ดังนี้:

1. การเรียกใช้งาน stored procedure XSR_REGISTER หรือเรียกใช้คำสั่ง REGISTER XMLSCHEMA ในการลงทะเบียน XML Schema ใหม่ใน XSR

   หมายเหตุ จะไม่มีเอกสารที่ จะถูกตรวจสอบความถูกต้องกับ XML Schema ที่มีการลงทะเบียนใหม่ถ้าขั้นตอนคือการแทนที่ schema ที่มีอยู่แล้ว ด้วย schema ใหม่ตามที่จะกล่าวถึงในขั้นตอนถัดไป

2. การเรียกใช้งาน stored procedure XSR_UPDATE หรือเรียกใช้คำสั่ง UPDATE XMLSCHEMA ในการปรับปรุง XML Schema ใหม่ใน XSR โดยแทนที่ schema เดิม ที่มีอยู่

หลังจากที่มีการปรับปรุง XML Schema แล้ว Schema ใหม่จะถูกนำไปแทนที่ XML Schema เดิม ซึ่ง หลังจากทำสำเร็จแล้ว จะมีเพียง XML Schema ใหม่เท่านั้นที่สามารถ ใช้งานได้

ถ้ามีการใช้ตัวเลือก dropnewschema เมื่อมีการเรียกใช้ stored procedure XSR_UPDATE หรือ คำสั่ง UPDATE XMLSCHEMA schema ใหม่จะ สามารถใช้งานได้ภายใต้ชื่อ schema ที่มีอยู่เดิมเท่านั้น จะไม่สามารถใช้งานภายใต้ชื่อที่ใช้ในการลงทะเบียนใหม่ดังกล่าว

8.3 XPath

XPath 2.0 เป็น expression language สำหรับการประมวลผลค่าที่สอดคล้องกับ XQuery/XPath Data Model (XDM) โดย XPath จะใช้ path expression ในการเข้าถึง เอกสาร XML โดยเป็นองค์ประกอบสำคัญใน XSLT และ XQuery และเป็นสิ่งที่ W3C ให้คำรับรอง

8.3.1 แบบจำลองข้อมูลสำหรับ XPath (XPath data model - XDM)

XDM จะแสดงเอกสาร XML ในรูปแบบโครงสร้างต้นไม้ โดย ส่วนต่างๆ ใน XDM จะเป็นลำดับ (sequence) ที่เป็นไปได้ทั้ง การไม่มีรายการ (zero item) ไปจนถึง การมีหลายๆ รายการ (more item) โดยที่รายการ (item) หมายถึงสิ่งต่างๆ ดังต่อไปนี้:

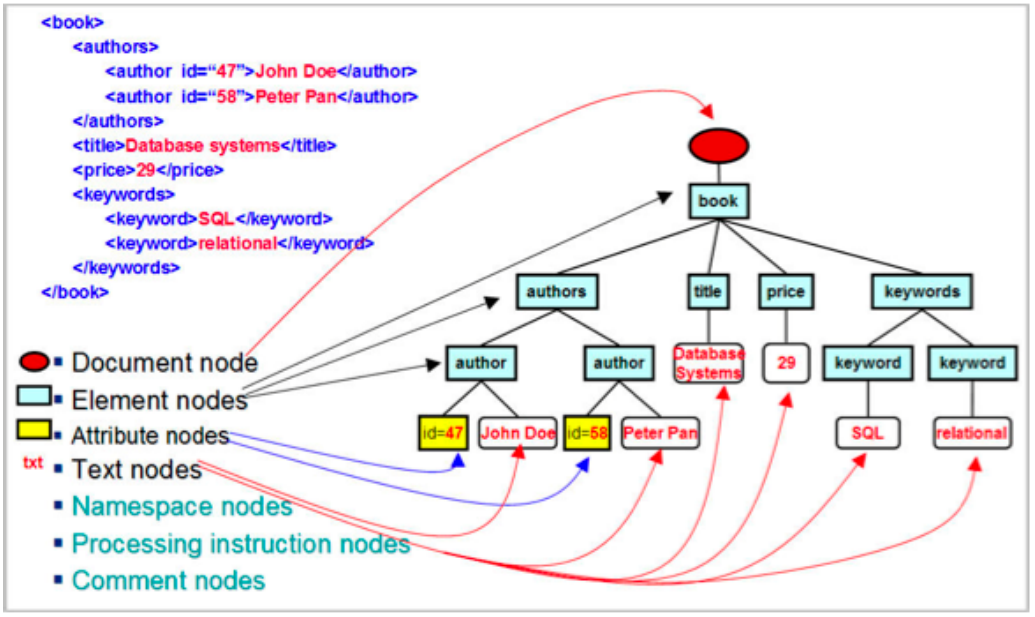
- Atomic values เช่น integers strings หรือ Booleans
- XML nodes เช่น Document, element, attribute หรือ text

XPath หรือ XQuery expression จะทำงานกับ instance ของ XDM โดย ลำดับ (sequence) คือการเรียงรายการต่างๆ ตั้งแต่การไม่มีรายการ หรือการมีหลายๆรายการ ซึ่งรายการ (item) จะหมายถึง atomic values หรือ nodes ดังได้กล่าวไว้ข้างต้น ถ้าเราต้องการรวมลำดับ สองลำดับเข้าด้วยกัน ผลลัพธ์ที่ได้ จะเป็นลำดับที่ประกอบไปด้วยรายการทั้งหมด ของทั้งสองลำดับ

ตัวอย่างเช่น การ แทรกลำดับ (<x/>, <y/>, 45) ระหว่างสองรายการในลำดับ (“beta”, 2) จะได้ผลลัพธ์ คือลำดับ (“beta”, <x/>, <y/>, 45, 2) เครื่องหมายต่างๆ ที่ใช้ในตัวอย่างข้างต้น เช่น วงเล็บเครื่องหมายคำพูด จะสอดคล้องกันกับไวยากรณ์ที่ใช้ในการสร้างลำดับใน XQuery โดยลำดับจะหมายถึงรายการ ทั้งหมด ที่อยู่ในเครื่องหมายวงเล็บ ซึ่งแต่ละรายการจะถูกคั่นด้วยเครื่องหมายจุลภาค (comma)

8.3.2 Document node

Document node เป็น node ที่มีอยู่ในทุกเอกสาร XML โดยเป็นจุดเริ่มต้นของเอกสาร ทุกๆเอกสาร XML ต้องมี document node และจะต้องไม่มี document node มากกว่าหนึ่งในเอกสาร XML เดียวกันใน DB2 ลำดับ (sequence) ที่ถูกส่งค่ากลับมาโดยคำสั่ง XQuery หรือ XPath จะอยู่ในรูปแบบของเอกสาร XML รูปที่ 8.5 แสดงชนิดต่างๆของ nodes ซึ่งเป็นส่วนหนึ่งของ XML data model



รูป 8.5 – XML Data Model: ชนิดของโหนด

ตัวอย่างคำสั่งต่อไปนี้ประกอบด้วย content expression ที่ส่งกลับมาเป็นเอกสาร XML ประกอบด้วย root element ชื่อว่า customer-list:

```
document
{
        <customer-list>
                {db2-fn:xmlcolumn('MYSCHEMA.CUSTOMER.INFO')/ns1:customerinfo/
        name}
        </customer-list>
}
```

ในรูปแบบมาตราฐานของเอกสาร XML ที่มีความถูกต้อง (well-formed) จะมี node แรก เป็น document node

ตัวอย่างเช่น ในเอกสาร XML ด้านล่าง element <products> เป็น document node ซึ่งใน บางครั้งเราจะเรียก document node ว่า root node หรือโหนดเริ่มต้น

```
<products>
<product pid="100-201-01"><description>
<name>Ice Scraper, Windshield 4 inch</name>
<price>3.99</price></description>
</product>
</products>
```

8.3.3 Path Expressions

Path expressions เป็น expression ที่ใช้บ่อยที่สุดใน XPath ซึ่งจะประกอบด้วยหนึ่งขั้นตอน (step) หรือ หลายๆ ขั้นตอน โดยที่แต่ละขั้นตอนจะถูกคั่นด้วยเครื่องหมายทับ (/) หรือเครื่องหมายทับสองอัน (//) ผลลัพธ์ของ path expression จะเป็นลำดับของรายการที่สร้างจากขั้นตอนสุดท้าย (final step) ใน path expression

ถ้าเราเขียน expression โดยเริ่มต้นด้วยเครื่องหมาย "/" หมายถึง path จะเริ่มต้นจาก root node จุดเริ่มต้น แต่ถ้าเราเขียน expression โดยเริ่มต้นด้วยเครื่องหมาย "// " หมายถึง path จะ เริ่มต้นจาก root node รวมไปถึง node ที่อยู่ภายใต้ทั้งหมด (all child nodes) ตัวอย่างเช่น //name หมายถึง ให้แสดง element name ทั้งหมดที่มีอยู่ในเอกสาร XML โดยไม่สนใจว่าจะอยู่ที่ตำแหน่งไหน

8.3.4 Advanced Navigation in XPath

แต่ละขั้นตอน (step) จะสามารถประกอบไปด้วย axis step หรือ filter expression โดยที่ axis step จะประกอบด้วยสามส่วน ต่อไปนี้:

- เป็น optional axis ซึ่งระบุทิศทางของการ เคลื่อนที่ในเอกสาร XML
- node test ที่กำหนด เงื่อนไขที่ใช้ในการเลือก node และ
- Predicates ซึ่งจะทำการกรองลำดับ (sequence) ที่สร้างขึ้นโดยขั้นตอน (step)

รูปแบบของ axis step คือ axisname::nodetest[predicate]

ตัวอย่างเช่น

- Child::book หมายถึง ให้ทำการเลือก book node ทั้งหมดที่เป็น child node ของ node ปัจจุบัน
- Attribute::id หมายถึง ให้ทำการเลือก id attribute ของ node ปัจจุบัน

ผลลัพธ์ของ axis step คือ ลำดับของ nodes และแต่ละ node จะถูกกำหนด context position ที่สอดคล้องกับตำแหน่งของมันในลำดับ context position จะอนุญาตให้ทุกโหนดสามารถเข้าถึงได้ โดยการใช้ตำแหน่งของมัน

8.3.5 XPath Semantics

Context node คือ node อะไรก็ตามที่ถูกทดสอบเพื่อดูว่าเข้ากันได้ (match) กับเงื่อนไขที่ใช้ในการเลือก node หรือไม่ นอกจากนี้ context node ยังมีความหมายพิเศษเมื่อใช้ใน nested XPath expression (ส่วนใน เครื่องหมายก้ามปู [ ])

ตารางต่อไปนี้แสดง axes ต่าง ๆ ที่ สามารถใช้งานได้ใน DB2

Axis	Description	Direction
Self	Returns the context node.	Forward
Child	Returns the children of the context node	Forward
descendant	Returns the descendants (children, grand-children) of the context node	Forward
descendant-or-self	Returns the context node and its descen-dants	Forward
Parent	Returns the parent of the context node	Reverse
Attribute	Returns the attributes of the context node	Forward

node test คือเงื่อนไขที่ใช้ในการทดสอบ nodes โดยจะต้องมีค่าเป็นจริง (true) สำหรับแต่ละ node ที่ถูกเลือกไว้ โดย axis step ซึ่ง node test สามารถ เป็นได้ทั้ง name test และ kind test

name test จะทำการเลือก node โดยใช้ชื่อของ node เอง ซึ่งประกอบด้วย QName หรือ เครื่องหมายดอกจันทน์ (*) และ เมื่อใช้งาน จะทำการเลือกnode (element หรือ แอตทริบิวต์) ที่เข้ากันได้ (match) กับ QNames ซึ่งจะถือว่า QName เข้ากันได้ถ้า QName ของ node เท่ากันกับ QName ของ name test นอกจากนี้ QNames สองตัว จะมีค่าเท่ากันในกรณีที่อยู่ใน namespace เดียวกันและมี local names ที่เหมือนกัน

ตาราง ด้านล่างนี้อธิบายถึง name test ทั้งหมดที่ใช้งานได้ใน DB2

Test	Description
QName	Matches all nodes whose QName is equal to the specified QName
NCName.*	Matches all nodes whose namespace URI is the same as the namespace to which the specified prefix is bound
*.NCName	Matches all nodes whose local name is equal to the specified NCName
*	Matches all nodes

Kind test จะเลือก nodes ตามชนิดของ nodes ตาราง ด้านล่างนี้อธิบายถึง kind test ทั้งหมดที่ทำงานใน DB2

Test	Description
node()	Matches any node
text()	Matches any text node
comment()	Matches any comment node
processing-instruction()	Matches any processing instruction node
element()	Matches any element node
attribute()	Matches any attribute node
Document-node()	Matches any document node

จะมีไวยากรณ์ (syntax) อยู่ 2 แบบ สำหรับ axis step คือ unabbreviated และ abbreviated โดย ไวยากรณ์ unabbreviated จะประกอบด้วยชื่อของ axis และ node test ที่คั่น ด้วยเครื่องหมายโคลอนสองอัน (::) และไวยากรณ์ abbreviated ซึ่งจะถูกเขียน โดยการใช้เครื่องหมาย

ตาราง ด้านล่างนี้อธิบายไวยากรณ์ abbreviated ที่ใช้งานได้ ใน DB2

Abbreviated syntax	Description
No Axis specified	child:: except when the node test is attribute(). In that case, omitted axis shorthand for attribute::
@	attribute::
//	/descedent-or-self::node() except when it appear in the beginning of path expression. In that case, axes step selects the root of the tree plus all nodes that are its descendents
.	self::node()
..	Parent::node()

ตัวอย่างเช่น ตารางต่อไปนี้แสดง path expressions

Path expression	Path Expression using abbreviated syntax
/dept/emp/firstname/child::node()	/dept/emp/firstname
/dept/emp//firstname/parent::node()/@id	/dept/emp/firstname/../@id

8.3.6 XPath Queries

ใน DB2 เราสามารถเขียน XPath expression โดยการฝัง path expressions ใน XQuery

ตัวอย่างเช่น XQuery ต่อไปนี้ส่งค่ากลับมาเป็นชื่อ (ชื่อของ element) ของผลิตภัณฑ์ทั้งหมดที่จัดเก็บไว้ในคอลัมน์ DESCRIPTION ของตาราง

```
XQuery db2-fn:xmlcolumn('PRODUCT.DESCRIPTION')/product/description/name
Execution result
1
--------------------------------------------------
<name>Snow Shovel, Basic 22 inch</name>
<name>Snow Shovel, Deluxe 24 inch</name>
<name>Snow Shovel, Super Deluxe 26 inch</name>
<name>Ice Scraper, Windshield 4 inch</name>
  4 record(s) selected.
```

ดังที่แสดงไว้ข้างต้น xmlcolumn คือฟังก์ชันใน XQuery ที่ใช้ string argument ของแบบฟอร์ม SCHEMA-NAME.TABLENAME.XMLCOLUMNNAME ถ้าตารางที่ต้องการจะ สอบถามข้อมูลอยู่ใน default schema จะ สามารถระบุเพียง TABLENAME.XMLCOLUMNNAME เท่านั้น โดย Db2-fn เป็นชื่อของ namespace ที่ฟังก์ชัน xmlcolumn เป็นสมาชิก ฟังก์ชัน xmlcolumn จะส่งกลับค่าที่เป็นเอกสาร XML ทั้งหมดที่จัดเก็บไว้ในคอลัมน์ XML ที่ระบุ

DB2 ยังมีฟังก์ชัน XQuery อื่นที่เรียกว่า ***sqlquery*** ที่ประมวลผลแล้วได้ผลลัพธ์เดียวกันกับ ฟังก์ชัน ***xmlcolumn*** ความแตกต่างระหว่างสองฟังก์ชันอยู่ที่ ความสามารถของฟังก์ชัน ***sqlquery*** ซึ่งสามารถระบุให้ทำงานเฉพาะเอกสาร XML ที่ต้องการ ซึ่งตรงข้ามกับการต้องทำงานกับเอกสาร XML ทั้งหมด ในกรณีของฟังก์ชัน xmlcolumn ตัวอย่างของ XQuery ด้านล่างเป็นการแสดงชื่อของผลิตภัณฑ์เฉพาะที่มี pid เป็น 100-201-01 โดยที่ pid เป็นคอลัมน์เชิงสัมพันธ์ (relational column)

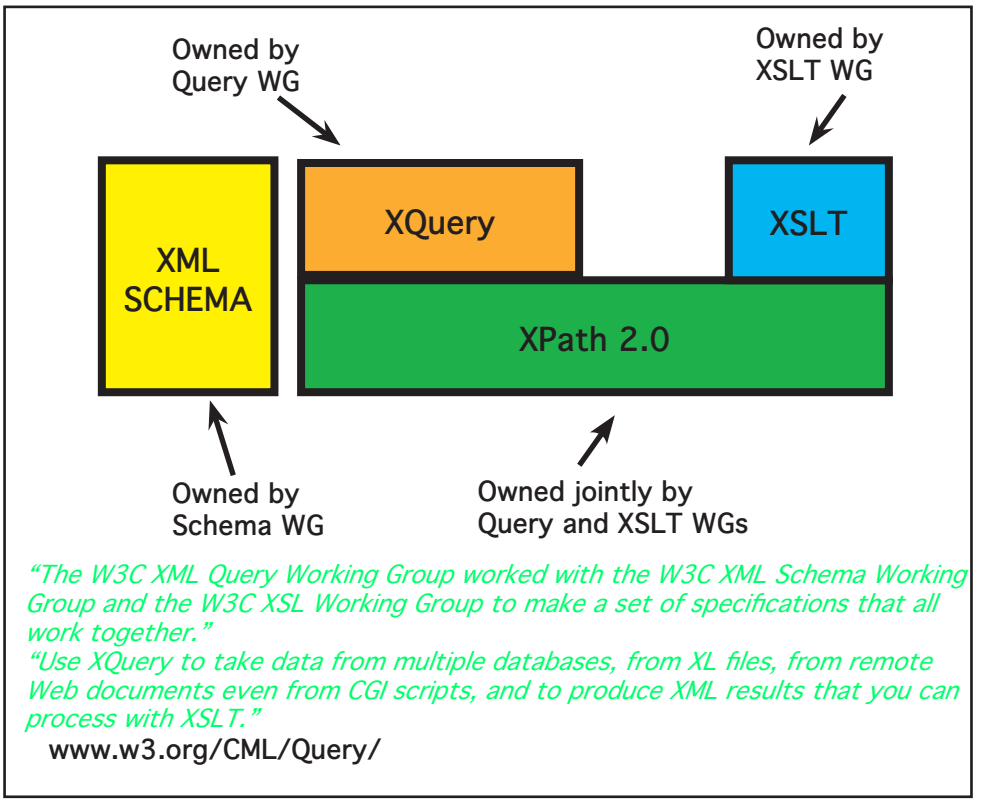
```
XQuery db2-fn:sqlquery("select DESCRIPTION from PRODUCT where pid= '100-201-
01'")/product/description/name

Execution result :
1
--------------------------------------------------
<name>Ice Scraper, Windshield 4 inch</name>

1 record(s) selected.
```

8.4 XQuery

XQuery เป็นภาษา ในการสอบถามข้อมูลที่ออกแบบมาสำหรับแหล่งข้อมูล XML โดยเฉพาะ โดย XQuery ได้รับการพัฒนาโดยได้รับการสนับสนุนจาก W3C และขณะนี้อยู่ในสถานะที่ได้รับการรับรอง ซึ่ง XQuery ถูกใช้เป็นมาตรฐานของภาษาในการสอบถามข้อมูลสำหรับ XML XQuery สามารถเข้าถึงข้อมูล XML ได้เช่นเดียวกันกับ SQL เข้าถึงข้อมูลเชิงสัมพันธ์ รูปที่ 8.6 แสดง XQuery และ มาตรฐานอื่นๆ ที่เกี่ยวกับ XML โดยรูปดังกล่าว 8.6 แสดงให้เห็นว่าทั้ง XPath และ XQuery ใช้ XPath เป็นพื้นฐาน และมีการใช้งาน XPath expressions



รูป 8.6 - XQuery และมาตรฐานที่เกี่ยวข้อง

8.4.1 XQuery พื้นฐาน

ทุก XQuery ประกอบด้วยสองส่วนต่อไปนี้

- prolog ซึ่งเป็นทางเลือก ประกอบด้วยการประกาศ (declarations) ที่จะกำหนดสภาพแวดล้อมการทำงานของการสอบถามข้อมูล (query)
- ส่วนประกอบหลักของการ สอบถามข้อมูล (query body) ประกอบด้วย expression ที่แสดงผลลัพธ์ของ การสอบถามข้อมูล

ข้อมูลนำเข้า (input) และ ผลลัพธ์ (output) ของ XQuery จะเป็น instance ของ XDM เสมอ

ด้านล่างเป็นตัวอย่างของ XQuery ที่ซึ่งการประกาศ element namespace ที่เป็น default จะเป็นส่วนของ prolog และ และส่วนประกอบหลัก ซึ่งเป็น FLWOR expression โดย FLWOR expression จะถูกอธิบายในส่วนถัดไป

```
XQUERY
declare default element namespace "http://posample.org";
for $cust in db2-fn:xmlcolumn('CUSTOMER.DESCRIPTION')
return $cust/Name/LastName;
```

ส่วนประกอบหลัก (body) ของ XQuery สามารถประกอบไปด้วยส่วนใดส่วนหนึ่งหรือทั้งหมดของ expression ต่อไปนี้:

- Literals and variables
- Path expressions
- Predicates
- If ..then..else
- Constructors
- Comparisons
- FLWOR expressions

8.4.2 FLWOR expressions

FLWOR expressions เป็นส่วนสำคัญของภาษา XQuery โดยที่ FLWOR จะย่อมาจาก FOR LET WHERE ORDER BY และ RETURN บ่อยครั้งที่ FLWOR จะถูกเปรียบเทียบ กับคำสั่ง SELECT FROM WHERE ORDER BY ใน SQL ตัวอย่างด้านล่างเป็นตัวอย่างการใช้ FLWOR expression ใน XQuery

```
XQuery
for $x in db2-fn:xmlcolumn('PRODUCT.DESCRIPTION')
let $p := $x/product/description/name
where $x/product/description/price < 3
return <cheap_products> {$p} </cheap_products>
```

คำสั่งย่อย for เป็นการทำซ้ำโดยจะทำงานครั้งละหนึ่งเอกสาร ผ่านการใช้งาน ตัวแปร (จากตัวอย่างคือ $x) ซึ่งตัวแปร $x จะถูกแทนที่ด้วยเอกสาร XML ที่จัดเก็บในคอลัมน์ DESCRIPTION ของตาราง PRODUCT ในระหว่างการทำซ้ำแต่ละครั้งของคำสั่งย่อย for $x จะถูกแทนที่ด้วยเอกสาร XML ครั้งละหนึ่งเอกสาร โดยที่เอกสาร XML ดังกล่าวจะถูกนำไปใช้ในการกำหนดค่าของตัวแปร $p ของ XQuery ตัวอย่างเช่น ถ้าให้เอกสาร XML ที่แทนที่ตัวแปร $x มีชื่อผลิตภัณฑ์อยู่ 4 ชื่อ ซึ่งชื่อผลิตภัณฑ์ดังกล่าวจะเป็นส่วนหนึ่งของลำดับ (sequence) และถูกนำมาแทนที่ในการ กำหนด ค่าตัวแปร $p

คำสั่งย่อย where จะคล้ายกับคำสั่งย่อย where ใน SQL คือเป็นการกรองผลลัพธ์จากเงื่อนไขที่ระบุ จากตัวอย่างข้างต้น ได้กำหนดให้เลือกข้อมูลเฉพาะสินค้าที่มีราคาน้อยกว่า 3

คำสั่งย่อย return เป็นการกำหนด XDM instance ที่จะคืนค่าออกมา ตัวอย่างต่อไปนี้แสดงรายละเอียดเพิ่มเติมเกี่ยวกับ XQuery FLWOR

1. XQuery for $i in (1 to 3) return $i
   OR
2. XQuery for $x in db2-fn:xmlcolumn('PRODUCT.DESCRIPTION')//text()

Execution Result

```
1
-----
1
2
3
```

จากตัวอย่าง การสอบถามข้อมูลด้านบน จะทำการคืนค่า text node ทั้งหมดจากเอกสาร XML ที่จัดเก็บในคอลัมน์ DESCRIPTION ในตารางข้อมูล PRODUCT

8.4.3 การ Joins ใน XQuery

ใน XQuery เราสามารถที่จะ Join ข้อมูล XML จากหลายๆ แหล่งข้อมูลได้ ซึ่งจะเป็นประโยชน์ เมื่อต้องการสอบถามข้อมูล XML จากคอลัมน์ XML หนึ่งหรือมากกว่าหนึ่งคอลัมน์ ตัวอย่างด้านล่างเป็นตัวอย่างการ Join ใน XQuery

```
for $book in db2-fn:xmlcolumn('BOOKS.DOC')/book
for $entry in db2-fn:xmlcolumn('REVIEWS.DOC')/entry
where $book/title = $entry/title
return <review>
{$entry/review/text()}
</review>;
```

จากตัวอย่างเป็นการ join ของสองคอลัมน์ XML ชื่อ DOC จากตารางข้อมูลที่ต่างกันคือ ตาราง BOOKS และตาราง REVIEWS คอลัมน์ DOC ในตาราง BOOKS จัดเก็บข้อมูลพื้นฐานเกี่ยวกับหนังสือ เช่น ชื่อเรื่อง ชื่อผู้แต่งและราคาของหนังสือเล่มนี้ในรูปแบบ XML คอลัมน์ DOC ของตาราง REVIEWS เก็บข้อมูลความคิดเห็นเกี่ยวกับหนังสือ เช่น ชื่อเรื่องของหนังสือและความคิดเห็นในหนังสือพร้อมกับคำอธิบายความคิดเห็น XQuery ข้างต้นจะคืนค่าข้อมูลความคิดเห็น (text nodes เท่านั้น) เกี่ยวกับหนังสือเหล่านั้น ที่มีรายการทั้งในตาราง BOOKS และตาราง REVIEWS

8.4.4 ฟังก์ชันที่ผู้ใช้งานกำหนดขึ้นมาเอง (User-defined functions)

ตามข้อกำหนดของ XQuery ผู้ใช้สามารถสร้างและกำหนดฟังก์ชันขึ้นมาเองภายในขอบเขตของ XQuery และสามารถเรียกใช้ฟังก์ชันดังกล่าวในช่วงระยะเวลาการดำเนินการของ XQuery โดยที่ DB2 จะมี built-in functions มากมายที่สามารถใช้งานภายใน XQuery เช่น ฟังก์ชันที่มีหน้าที่ในการจัดการกับสตริง ตัวเลขและวันที่ นอกจากนี้ยังมีฟังก์ชันที่จะดำเนินการในการจัดการกับลำดับ (sequence) เช่นการยอนกลับลำดับ ฯลฯ ตัวอย่าง ด้านลางเป็น XQuery ที่ใช้ฟังก์ชัน “contains” ใน DB2 :

```
XQuery
declare default element namespace “http://posample.org”;
for $x in db2-fn:xmlcolumn('PRODUCT.DESCRIPTION')
let $p := $x/product/description/name
where $x/product/description/name/fn:contains(text(),'Scraper')
return $p

Execution Result
:
1
--------------------------------------------------------------------
```

```
<name>Ice Scraper, Windshield 4 inch</name>
    1   record(s) selected.
```

8.4.5 XQuery และ XML Schema

XQuery และ XML Schema เป็นมาตรฐานที่พัฒนาโดยคณะทำงานของ W3C, โดยที่ XML schema ถูกใช้ในการ อธิบายความสัมพันธ์ของชนิดของข้อมูลใน XDM instance
นอกจากนั้น DB2 ยังมี built-in functions ซึ่งช่วยในการแปลงค่า nodeต่างๆไปเป็นรูปแบบของ XML Schema ตัวอย่าง XQuery ด้านล่างแสดงการใช้งาน ฟังก์ชัน xs:double

```
XQuery
declare default element namespace “http://posample.org”;
for $x in db2-fn:xmlcolumn(‘PRODUCT.DESCRIPTION’)
let $p := $x/product/description/name
where $x/product/description/xs:double(price) = 3.99
return $p
```

8.4.6 การจัดกลุ่มและการรวมตัว (Grouping and aggregation)

XQuery สนับสนุนการจัดกลุ่มและการรวมตัวของข้อมูล XML ตัวอย่างด้านล่างเป็นตัวอย่างของการจัดกลุ่มข้อมูล XML จากสองคอลัมน์ XML ที่มาจากตาราง สองตาราง คำสั่งในการสอบถามขอมูล ดานลางนี้เป็นการรวมกลุ่มของชื่อของลูกค้าในตาราง CUSTOMER

คำสั่งย่อย “for” เป็นการ ทำซ้ำเอกสาร XML ในคอลัมน์ info ของตาราง customer โดยการแทนที่แต่ละ element “city” ผ่าน ตัวแปร $city สำหรับแต่ละ city คำสั่งย่อย let กำหนดค่าตัวแปร $cust - names โดยการแทนที่ด้วยชื่อของลูกค้าทั้งหมดที่อยู่ในเมืองนั้น การสอบถามข้อมูล จะ ได้ผลลัพธ์เป็น element “city” ซึ่งประกอบด้วยชื่อเมืองและข้อมูลลูกค้าทั้งหมดที่อาศัยอยู่ในเมืองนั้น

```
XQuery
for $city in fn:distinct-values(db2-fn:xmlcolumn(‘CUSTOMER.INFO’)
   /customerinfo/addr/city)
let $cust-names := db2-fn:xmlcolumn(‘CUSTOMER.INFO’)
   /customerinfo/name[../addr/city = $city]
order by $city
        return <city>{$city, $cust-names} </city>
```

จากการสอบถามข้อมูล จะได้ผลลัพธ์ดังต่อไปนี้

```
Execution result :
1
<city>Aurora
   <name>Robert Shoemaker</name>
</city>
<city>Markham
   <name>Kathy Smith</name>
   <name>Jim Noodle</name>
```

```
</city>
<city>Toronto
   <name>Kathy Smith</name>
   <name>Matt Foreman</name>
   <name>Larry Menard</name>
</city>
```

ตัวอย่างด้านล่างต่อไปนี้ เป็นตัวอย่างการรวมตัว (aggregation) ใน XQuery ซึ่งจะทำการ ทำซ้ำสำหรับแต่ละ element “PurchaseOrder” โดยใช้เงื่อนไข คือ วันที่สั่งซื้อในปี 2005 และ ทำการแทนที่ element ดังกล่าวในตัวแปร $po สำหรับคำสั่งย่อย for และ path expression $po/item/ จะถูกเคลื่อนย้ายไปยังตำแหน่งของแต่ละ element “item” ที่อยู่ภายใต้ element “PurchaseOrder” ส่วนของ nested expression (price * quantity) จะ ทำการคำนวณรายได้ทั้งหมดสำหรับแต่ละ item ฟังก์ชัน fn:sum จะทำการรวมรายได้ทั้งหมด สำหรับแต่ละ item หลังจากนั้น คำสั่งย่อย let จะทำการแทนที่ผลลัพธ์ของฟังก์ชัน fn:sum ไปที่ตัวแปร $revenue ส่วนคำสั่งย่อย order by จะทำการเรียงลำดับผลลัพธ์ โดยรายได้ทั้งหมด สำหรับแต่ละคำสั่งซื้อ

```
for $po in db2-fn:xmlcolumn('PURCHASEORDER.PORDER')/
    PurchaseOrder[fn:starts-with(@OrderDate, "2005")]
  let $revenue := sum($po/item/(price * quantity))
  order by $revenue descending
  return
   <tr>
     <td>{string($po/@PoNum)}</td>
     <td>{string($po/@Status)}</td>
     <td>{$revenue}</td>
   </tr>
```

8.4.7 Quantification

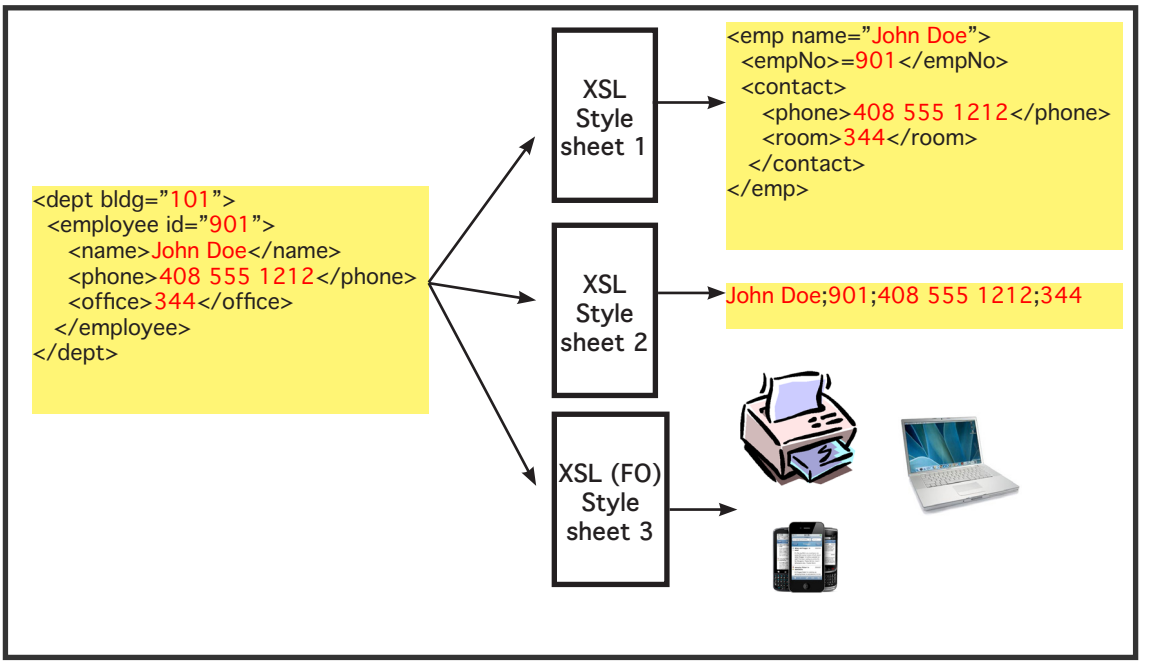
Quantified expressions จะทำการคืนค่า true หรือ false ขึ้นอยู่กับว่า รายการบางส่วนหรือทุกรายการในลำดับ (sequence) ตรงกันกับเงื่อนไขที่ระบุไว้หรือไม่ ดังตัวอย่างต่อไปนี้

```
some $i in (1 to 10) satisfies $i mod 7 eq 0
every $i in (1 to 5) , $j in (6, 10) satisfies $i < $j
```

Quntified expression ขึ้นต้นด้วย quantifier คือ some หรือ every โดยที่ quantifier จะตามด้วยคำสั่งย่อยหนึ่ง หรือมากกว่าหนึ่งคำสั่ง ที่ซึ่งมีการแทนที่ตัวแปรด้วยลำดับ (sequence) ในตัวอย่างแรกข้างต้น $i เป็นตัวแปรและมี (1 to 10) เป็น ลำดับ ตัวอย่างที่สองข้างต้น มีสองตัวแปรคือ $i และ $j โดยมี (1 to 5) และ (6 to 10)เป็น ลำดับ จากนั้นจะตามด้วย expression ที่ใช้ในการทดสอบ โดยจะมีการอ้างอิงถึงตัวแปร expression ที่ใช้ในการทดสอบ จะใช้ในการทดสอบว่า ค่าบางส่วน หรือ ค่าทั้งหมดที่แทนที่ในตัวแปร เป็นจริงตามเงื่อนไขที่ทดสอบหรือไม่ ในตัวอย่างแรก มีเงื่อนไขว่า ถ้า $i mod 7 เท่ากับ 0 โดยที่ qualifier คือ some ซึ่งหมายความว่า ถ้ามีค่าอย่างน้อยหนึ่งค่าที่ทำให้เงื่อนไขดังกล่าว เป็นจริง จะทำให้ผลลัพธ์ของ Quntified expression เป็นจริง ในตัวอย่างที่สองเงื่อนไขคือถ้า $i มีค่าน้อยกว่า $j โดยที่ qualifier คือ every หมายความว่า ทุกๆ ค่าจะต้องเป็นไปตามเงื่อนไขที่ระบุ จึงจะทำให้ Quntified expression เป็นจริง

8.5 XSLT

XSLT ย่อมาจาก eXtensible Stylesheet Language Transformations ซึ่งเป็นส่วนหนึ่งของมาตราฐาน XSL ที่อธิบายถึงวิธีการที่จะ แปลงโครงสร้างของเอกสาร XML ไปเป็นเอกสาร XML ที่มีโครงสร้างที่แตกต่างกัน เป็นการช่วยในการแสดงข้อมูลเดียวกันในรูปแบบที่แตกต่างกันออกไป ตัวอย่างเช่นรายละเอียดของการสั่งซื้อซึ่งเก็บไว้เป็นเอกสาร XML สามารถแปลงโดยการใช้ style sheets ต่างๆ เอกสารการสั่งซื้อบางคำสั่งซื้อสามารถแสดงบนเว็บไซต์ในรูปแบบตารางโดยการฝัง HTML tag รูปที่ 8.7 แสดงวิธีการทำงานของ XSLT



รูป 8.7 – eXtensible Stylesheet Language (XML) หนึ่งแหล่งที่มาและหลายๆเป้าหมาย

ใน DB2 9.7 เพื่อดำเนินการแปลง XSLT มีอยู่สองสิ่งที่ต้องใช้

1. ข้อมูลเอกสาร XML
2. XSLT Stylesheet

XSLT style sheet เป็นเอกสาร XML ที่อยู่ในรูปแบบที่ถูกต้อง (well-formed) และสามารถจัดเก็บไว้ในคอลัมน์ XML ของ DB2 ด้านล่างเป็นตัวอย่างของเอกสาร XSLT style sheet

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <h2>Product Details</h2>
    <table border="1">
      <tr >
        <th>Name</th>
        <th>Price</th>
      </tr>
      <xsl:for-each select="product/description">
        <tr>
          <td><xsl:value-of select="name"/></td>
          <td><xsl:value-of select="price"/></td>
        </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

ตัวอย่าง ของข้อมูล XML ที่จะทำการแปลง :

```
<products>
    <product pid="100-201-01">
      <description>
        <name>Ice Scraper, Windshield 4 inch</name>
        <details>Basic Ice Scraper 4 inches wide, foam handle</details>
        <price>3.99</price>
      </description>
    </product>
   </products>
```

DB2 9.7 มี ฟังก์ชัน XslTransform ที่ จะรับเอกสาร XML และ XSLT stylesheet เป็น ข้อมูลนำเข้า และจะได้ผลลัพธ์ที่ได้จากการแปลง (ในรูปแบบของเอกสาร XML)
ตัวอย่าง คำสั่ง SELECT ดังต่อไปนี้ใช้ฟังก์ชัน XslTransform ซึ่งจะได้ผลลัพธ์เป็น HTML document ที่สามารถบันทึกเป็นไฟล์ .html

```
SELECT XSLTRANSFORM (description USING stylesheet AS CLOB (10M)) FROM
product where pid like '100-201-01'
```

เมื่อประมวลคำสั่งจะได้ผลลัพธ์ดังนี้ :

```
1
-------------------------------------------------------
<html>
<body>
<h2>Product Details</h2>
<table border="1">
<tr>
<th>Name</th><th>Price</th>
</tr>
</table>
</body>
</html>

  1 record(s) selected.
```

8.6 SQL/XML

SQL / XML จะกำหนดกลไกมาตรฐานสำหรับการใช้ข้อมูล XML ด้วย คำสั่ง SQL SQL/XML เป็นไปตามมาตรฐาน ANSI / ISO ที่กำหนดการทำงานกับ SQL based extensions (ฟังก์ชัน) ซึ่งจะช่วยในการดำเนินการต่างๆเกี่ยวกับข้อมูลเชิงสัมพันธ์และข้อมูล XML นอกจากนี้ยังมีการกำหนด XML ให้เป็นชนิดของข้อมูล โดยฟังก์ชันเหล่านี้จะถือว่าเป็นส่วนขยายของคำสั่ง SQL นอกจากนี้ยังมีการกำหนดฟังก์ชันเพื่อตรวจสอบไวยากรณ์ในเอกสาร XML และการตรวจสอบเอกสาร XML โดยเทียบกับ XML schema รวมถึงมีฟังก์ชันอื่น ๆ เช่น XMLCONCAT และ XMLAGG ที่สามารถใช้ในการรวม XML ให้เป็น เอกสาร XML ที่มีขนาดใหญ่ขึ้น

8.6.1 ความสัมพันธ์ในการเข้ารหัส XML Document

SQL / XML มีฟังก์ชันในการแปลงข้อมูลเชิงสัมพันธ์ให้อยู่ในรูปแบบ XML และแปลงกลับจาก XML ไปเป็นข้อมูลเชิงสัมพันธ์ และยังมีฟังก์ชัน ที่ใช้ตรวจสอบ การ วิเคราะห์โครงสร้าง (parse) และการลำดับ (serialize) เอกสาร XML ฟังก์ชันการเผยแพร่ (Publishing) จะเป็นประโยชน์เมื่อมีความ ต้องการส่งข้อมูลในรูปแบบ XML ไปยังโปรแกรมประยุกต์ ตัวอย่างเช่น ถ้ามีการเรียกใช้ web service โดยต้องการคำที่ส่งคืนมาเป็นข้อมูลสต็อกโดยใช้รหัสสต็อกสินค้าในการอ้างอิงสต็อก (เป็นรูปแบบเอกสาร XML) ดังนั้นข้อมูลของสต็อกที่ถูกจัดเก็บไว้ในตารางข้อมูลเชิงสัมพันธ์ สามารถแปลง ให้อยู่ใน รูปแบบ XML และ web service สามารถส่ง XML ที่ได้ ไปยังโปรแกรมประยุกต์ที่ร้องขอ ในหลายโอกาส อาจมีการใช้งาน ขอมูลบางส่วนของ XML เช่น อาจจะมีการสร้างรายงาน หรือเครื่องมือ business intelligence ที่เรียกใช้ข้อมูล บางส่วนของ XML และนำข้อมูลที่ได้ดัง

กล่าวไปประมวลผลต่อ

8.6.2 การจัดเก็บและเผยแพร่เอกสาร XML

ฟังก์ชันการเผยแพร่ (Publishing) เป็นฟังก์ชัน SQL / XML ที่ใช้ในการแปลงข้อมูลเชิงสัมพันธ์เป็นรูปแบบ XML และ สามารถแปลงย้อนกลับจาก XML มาเป็นข้อมูลเชิงสัมพันธ์ได้อีกด้วย

8.6.3 ฟังก์ชัน SQL / XML

มาดูถึงส่วนประกอบของฟังก์ชันและไวยากรณ์ของ SQL / XML

8.6.3.1 XMLELEMENTและ XMLATTRIBUTES

XMLELEMENT และ XMLATTRIBUTES เป็นสองฟังก์ชันที่ใช้กันมากที่สุดของฟังก์ชันการเผยแพร่ ดังที่กล่าวข้างต้น ฟังก์ชัน XMLELEMENT จะ ทำการสร้าง XML element จากข้อมูลเชิงสัมพันธ์ที่ได้รับ ตัวอย่างเช่นคำสั่ง SELECT ต่อไปนี้

```
Select XMLELEMENT(NAME “firstname”, firstnme) from employee
```

เมื่อเรียกคำสั่งจะได้ผลลัพธ์ดังนี้ : 1

```
----------------------------------------
<FIRSTNAME>CHRISTINE</FIRSTNAME>
<FIRSTNAME>MICHAEL</FIRSTNAME>
<FIRSTNAME>SALLY</FIRSTNAME>
<FIRSTNAME>JOHN</FIRSTNAME>
<FIRSTNAME>IRVING</FIRSTNAME>
<FIRSTNAME>EVA</FIRSTNAME>

6 record(s) selected.
```

คีย์เวิร์ด NAME ในคำสั่งจะเป็นตัวบอกว่าข้อความหรือสตริงที่ตามมาจะเป็นชื่อของ XML element ที่จะสร้างขึ้น สตริงที่ระบุชื่อ XML element จะตามด้วยชื่อคอลัมน์ในตารางข้อมูลที่สัมพันธ์กัน ซึ่งในตัวอย่างคือ firstname ฟังก์ชัน XMLATTRIBUTES ใช้เพื่อสราง แอตทริบิวต์จากข้อมูลเชิงสัมพันธ์ ตัวอย่างเช่นคำสั่ง SELECT ต่อไปนี้

```
Select XMLELEMENT(NAME “emp” , XMLATTRIBUTES(EMPNO AS “employee_num” ))
from employee
```

เมื่อเรียกคำสั่งจะได้ผลลัพธ์ดังนี้:

```
1
-----------------------------------------------------
<EMP EMPLOYEE_NUM=”000010”/>
<EMP EMPLOYEE_NUM=”000020”/>
<EMP EMPLOYEE_NUM=”000030”/>
<EMP EMPLOYEE_NUM=”000050”/>
<EMP EMPLOYEE_NUM=”000060”/>
6 record(s) selected.
```

ฟังก์ชัน XMLATTRIBUTES มีการรับค่า 2 ค่าคือ ชื่อแอตทริบิวต์และค่าของแอตทริบิวต์ โดย เป็นพารามิเตอร์เดียวในรูปแบบ A AS “B” ซึ่ง A เป็นค่าของแอตทริบิวต์ ที่เป็นคอลัมน์ ในตารางขอมูล (empno) ตามด้วย คีย์เวิร์ด AS และตามด้วยสตริงที่ระบุชื่อของแอตทริบิวต์ (employee_num) ให้สังเกตว่า EMP เป็น XML element ที่ไม่ได้มีค่าข้อความ (text value) ใดๆ ด้านล่างนี้เป็นอีกตัวอย่างหนึ่งของฟังก์ชัน XMLATTRIBUTES ที่ EMP เป็น XML element ที่มีสองแอตทริบิวต์ (NUM และ SALARY) นอกจากนี้ยังมี element ลูกอีกสอง element (FIRST และ LAST) ซึ่งแสดงชื่อและนามสกุลของพนักงาน

```
Select
XMLELEMENT(NAME “emp” ,
 XMLATTRIBUTES(EMPNO AS “NUM”,SALARY as “salary” ),
XMLELEMENT(NAME “first” , firstnme),
XMLELEMENT(NAME “last”, lastname) )
 from employee
```

เมื่อเรียกคำสั่งจะได้ผลลัพธ์ดังนี้ :

```
1
-----------------------------------------------------------
<EMP NUM=”000010” SALARY=”152750.00”><FIRST>CHRISTINE</
FIRST><LAST>HAAS</LAST></EMP>
<EMP NUM=”000020” SALARY=”94250.00”><FIRST>MICHAEL</
FIRST><LAST>THOMPSON</LAST></EMP>
<EMP NUM=”000030” SALARY=”98250.00”><FIRST>SALLY</
FIRST><LAST>KWAN</LAST></EMP>
<EMP NUM=”000050” SALARY=”80175.00”><FIRST>JOHN</
FIRST><LAST>GEYER</LAST></EMP>
<EMP NUM=”000060” SALARY=”72250.00”><FIRST>IRVING</
FIRST><LAST>STERN</LAST></EMP>
```

8.6.3.2 XMLQUERY

หนึ่งในข้อดีของการใช้ SQL / XML เมื่อเทียบกับ การใช้คำสั่ง XQuery เพียงอย่างเดียวคือทำให้สามารถดึงข้อมูลเชิงสัมพันธ์และ XML จากตารางข้อมูลได้ในเวลาเดียวกัน XMLQUERY เป็นฟังก์ชัน SQL / XML ที่ช่วยให้การดำเนินการดังกล่าวสามารถทำได้โดยง่าย ฟังก์ชัน XMLQUERY จะนำเข้า XQuery-expression-constant ซึ่งเป็นนิพจน์ (expression) ที่ ได้ผลลัพธ์ ในรูปแบบ XML ตัวอย่างเช่น คำสั่ง SELECT ต่อไปนี้แสดงรหัสและชื่อของผลิตภัณฑ์ทั้งหมดจากตารางผลิตภัณฑ์

```
SELECT pid , XMLQUERY(‘$DESCRIPTION/product/description/name’  ) AS
“PRODUCTNAME”   FROM product
```

เมื่อเรียกคำสั่งจะได้ผลลัพธ์ดังนี้ :

```
PID       PRODUCTNAME
-----------------------------------------------------------
100-100-01 <name>Snow Shovel, Basic 22 inch</name>
100-101-01 <name>Snow Shovel, Deluxe 24 inch</name>
100-103-01 <name>Snow Shovel, Super Deluxe 26 inch</name>
100-201-01 <name>Ice Scraper, Windshield 4 inch</name>
  4 record(s) selected.
```

ชื่อผลิตภัณฑ์เป็นส่วนหนึ่งของคอลัมน์ DESCRIPTION ที่ จัดเก็บเอกสาร XML ที่อธิบายผลิตภัณฑ์ ในกรณีที่มีคอลัมน์ DESCRIPTION มากกว่าหนึ่งคอลัมน์ (ซึ่งเป็นผลมาจากการ join มากกว่าสองตาราง) เราสามารถใช้คำสั่ง PASSING ดังตัวอย่างที่แสดงด้านล่าง

```
SELECT pid , XMLQUERY(‘$D/product/description/name’ PASSING
prod.description as “D”  ) AS “PRODUCTNAME”   FROM product prod,
purchaseorder po
```

และถ้าในเอกสาร XML มี namespaces อยู่ด้วย จะต้องมีการเขียน Query ใหม่ตามตัวอย่างดังต่อไปนี้

```
SELECT pid , XMLQUERY('$DESCRIPTION/*:product/*:description/*:name' ) AS
"PRODUCTNAME" FROM product
```

เมื่อเรียกคำสั่งจะได้ผลดังนี้ :

```
PID        PRODUCTNAME
------------------------------------------------------------------
100-100-01 <name xmlns="http://posample.org">Snow Shovel, Basic 22 inch</
name>
100-101-01 <name xmlns="http://posample.org">Snow Shovel, Deluxe 24 inch</
name>
100-103-01 <name xmlns="http://posample.org">Snow Shovel, Super Deluxe 26
inch</name>
100-201-01 <name xmlns="http://posample.org">Ice Scraper, Windshield 4 inch</
name>

  4 record(s) selected.
```

XMLQUERY จะทำการส่งกลับ XML sequence จากเอกสาร XML ทั้งหมดในคอลัมน์ดังกล่าว หากต้องการจะ เรียกชื่อผลิตภัณฑ์ที่เฉพาะเจาะจง โดยขึ้นอยู่กับ predicate แล้วสามารถใช้ ฟังก์ชัน XMLEXISTS พร้อมกับ XMLQUERY
ตัวอย่างเช่นคำสั่ง SELECT ดังต่อไปนี้จะเรียกชื่อของผลิตภัณฑ์ที่มี PID 100-103-01 และมีราคาของผลิตภัณฑ์เป็น 49.99

```
SELECT pid , XMLQUERY('$DESCRIPTION/*:product/*:description/*:name' ) AS
"PRODUCTNAME"  FROM product
 where
 XMLEXISTS('$DESCRIPTION/*:product[@pid="100-103-01"]/*:description[*:pri
ce="49.99"]')
```

เมื่อเรียกคำสั่งจะได้ผลดังนี้ :

```
PID        PRODUCTNAME
------------------------------------------------------------------
100-103-01 <name xmlns="http://posample.org">Snow Shovel, Super Deluxe 26
inch</name>
  1 record(s) selected.
```

ความสำคัญอีกอย่างหนึ่งของการใช้ XMLEXISTS พร้อมกับ XMLQUERY สำหรับ predicate คือจะช่วยในการกำจัด empty sequences เพราะ XMLEXISTS จะส่งกลับค่า จริงหรือเท็จ (true หรือ false) ขึ้นอยู่กับว่าตรงกับ predicate ที่ระบุหรือไม่ XMLQUERY จะส่งกลับเฉพาะ sequence ที่มีค่า เป็นจริง จากฟังก์ชัน XMLEXISTS ทำให้สามารถกำจัด empty sequence ได้ดังกล่าว
ตัวอย่างเช่นถ้าคำสั่ง SELECT มี predicate ที่เขียนโดยไม่ใช้ XMLEXISTS ผลลัพธ์ที่ได้ ทำให้เกิดความสับสนในคำสั่ง SELECT ด้านล่าง predicate เป็นส่วนหนึ่งของพารามิเตอร์ XQuery-expression-constant ของฟังก์ชัน XMLQUERY

```
SELECT pid , XMLQUERY('
$DESCRIPTION/*:product[@pid="100-103-01"]/*:description[*:price="49.99"]/*:n
ame')
AS "PRODUCTNAME"
 FROM product
```

เมื่อเรียกคำสั่งจะได้ผลดังนี้ :

```
PID        PRODUCTNAME
-------------------------------------------------------------------------
100-100-01
100-101-01
100-103-01 <name xmlns="http://posample.org">Snow Shovel, Super Deluxe 26
inch</name>
100-201-01
4 record(s) selected.
```

เหตุผลที่ทำให้มี 3 empty sequences เป็น เพราะ XMLQUERY จะคืนค่า sequence เสมอ โดยจะคืนค่า seguence ในกรณีที่ตรงกับ Predicate และจะคืนค่า empty sequence ในกรณีที่ไม่ตรงกันกับ Predicate ดังนั้น จึงแนะนำ ให้ใช้ ฟังก์ชัน XMLEXISTS ควบคู่กับ XMLQUERY สำหรับการค้นหาโดยใช้ Predicate

8.6.3.3 XMLAGG

ฟังก์ชัน XMLAGG จะส่งกลับ XML sequence ที่ไม่ใช่ null ในรูปแบบของ กลุ่มของ XML values ตัวอย่าง เช่น คำสั่ง SELECT ดังต่อไปนี้จะทำการรวมพนักงานที่อยู่ในแผนกให้เป็นหนึ่งกลุ่ม

```
SELECT
   XMLELEMENT (NAME "Department",
      XMLATTRIBUTES (e.workdept AS "name" ),
      XMLAGG ( XMLELEMENT (NAME "emp", e.firstnme)  )
        ) AS "dept_list"
FROM employee e
GROUP BY  e.workdept;
```

เมื่อเรียกคำสั่งจะได้ผลดังนี้ :

```
dept_list
-------------------------------------------------------------------
<Department name="A00">
<emp>CHRISTINE</emp>
<emp>VINCENZO</emp>
<emp>SEAN</emp>
<emp>GREG</emp>
</Department>
<Department name="B01">
<emp>MICHAEL</emp>
</Department>
<Department name="C01">
<emp>SALLY</emp>
<emp>DELORES</emp>
<emp>HEATHER</emp>
<emp>KIM</emp>
</Department>
<Department name="D21">
<emp>EVA</emp>
<emp>MARIA</emp>
</Department>
<Department name="E01">
<emp>JOHN</emp>
</Department>
```

8.7 การสอบถามข้อมูลจากเอกสาร XML ที่จัดเก็บในตารางข้อมูล

นอกเหนือจากการสร้างเอกสาร XML จาก ข้อมูลเชิงสัมพันธ์โดยใช้ฟังก์ชัน SQL / XML แล้ว เรายังสามารถใช้ XQuery เพื่อทำการ สอบถามขอมูลจาก เอกสาร XML ที่เก็บอยู่ในคอลัมน์ ที่เป็น XML หากเราต้องการจะ ใช้คำสั่ง SELECT ของ SQL เป็นภาษาหลักแล้ว เรา สามารถใช้ฟังก์ชัน XMLQUERY ของ SQL / XML และส่งผาน XQuery เป็นพารามิเตอร์ไปยังฟังก์ชัน ดังกล่าวได้

ตัวเลือกของภาษาในการสอบถามข้อมูล ขึ้นอยู่กับ ว่าต้อง การดึงข้อมูลชนิดใด ถ้าข้อมูลที่จะต้องใช้มีเพียงข้อมูลในรูปแบบของ XML แล้ว ทางเลือกหนึ่งคือการใช้ XQuery หรือการใช้ฟังก์ชัน XMLQUERY ที่เป็นส่วนหนึ่งของคำสั่ง SQL SELECT ในขณะเดียวกัน ถาขอมูลที่จะต้องใช้มีทั้งขอมูลเชิงสัมพันธ์และ XML แล้ว การใช้คำสั่ง SELECT ของ SQL ควบคู่ไปกับ XMLQUERY จะเป็นตัวเลือกที่เหมาะสมที่สุด

8.8 การเปลี่ยนแปลงข้อมูลในเอกสาร XML

การจัดเก็บเอกสาร XML ในฐานข้อมูลเป็นสิ่งที่สำคัญในธุรกิจปัจจุบัน เพราะข้อมูลดังกล่าวจะต้องสอดคล้องกับมาตราฐาน และสามารถตอบสนองกับความต้องการที่เปลี่ยนแปลงของลูกค้า นอกจาก DB2 จะมีความสามารถในการจัดการข้อมูล XML ที่มีประสิทธิภาพ โดยการจัดเก็บในรูปแบบที่เป็น native storage แล้ว DB2 ยังมีความสามารถในการจัดการข้อมูล XML โดยมีฟังก์ชัน XMLPARSE และ XMLSERIALIZE ซึ่งจะช่วยในการแปลงเอกสาร XML ที่อยู่ในรูปแบบ text ให้อยู่ในรูปแบบที่เป็น native XML รวมถึงสามารถแปลง native XML ให้เป็น XML ที่อยู่ในรูปแบบ text ได้อีกด้วย

8.8.1 XMLPARSE

ดังที่ได้กล่าวข้างต้น XMLPARSE เป็นฟังก์ชันที่ ใช้วิเคราะห์ข้อความ XML ที่กำหนดและทำการแปลง XML ดังกลาวให้เป็นรูปแบบ native XML (ตัวอย่างเช่น รูปแบบที่เป็นลำดับชั้นของต้นไม้ – hierarchical tree format) การ insert ขอมูลใน DB2 โดย default แลวจะทำการวิเคราะห์ขอความ XML แบบ implicit แต่ถาผู้ใช้ต้องการวิเคราะห์ข้อความ XML แบบ explicit ก็สามารถทำได้โดยใช้ XMLPARSE

ตัวอย่างเช่นคำสั่ง INSERT ต่อไปนี้ จะทำการ วิเคราะห์ข้อความ XML ที่กำหนดก่อน หากพบว่าเอกสาร XML ดังกล่าวเป็นเอกสาร XML ที่อยู่ในรูป well-formed ก็จะทำงานต่อด้วยการ insert เอกสาร XML ดังกล่าวในคอลัมน์ที่ระบุ

```
INSERT INTO PRODUCT(PID,DESCRIPTION) VALUES('100-345-01', XMLPARSE(
DOCUMENT '<product xmlns="http://posample.org" pid="100-100-01">
<description><name>Snow Shovel, Basic 22 inch</name>
<details>Basic Snow Shovel, 22 inches wide, straight handle with D-Grip</details>
<price>9.99</price>
<weight>1 kg</weight>
</description>
        </product> '))
```

อีกเหตุผลหนึ่งสำหรับการใช้ XMLPARSE คือ เป็นวิธีในการจัดการกับ whitespace โดย default แล้ว DB2 จะทำการตัดชองวางระหวาง element แต่ในกรณีที่ผู้ใช้ต้องการที่จะเก็บรักษาชองวางดังกลาวเอาไว้ จะต้องใช้คำสั่ง PRESERVE WHITESPACE ดังตัวอย่างที่แสดงด้านลาง

```
INSERT INTO PRODUCT(PID,DESCRIPTION) VALUES('100-345-01', XMLPARSE(
DOCUMENT '<product xmlns="http://posample.org" pid="100-100-01">
<description><name>Snow Shovel, Basic 22 inch</name>
<details>Basic Snow Shovel, 22 inches wide, straight handle with D-Grip</details>
<price>9.99</price>
<weight>1 kg</weight>
</description>
        </product> ' PRESERVE WHITESPACE))
```

8.8.2 XMLSERIALIZE

ฟังก์ชัน XMLSERIALIZE ถูกใช้เพื่อแปลงโครงสร้าง XML (ซึ่งเป็นโครงสร้างแบบลำดับชั้นของต้นไม้) ให้อยู่ในรูปแบบสตริงหรือไบนารี โดยจะสามารถ แปลงรูปแบบลำดับชั้นของต้นไม้ ให้เป็นรูปแบบ CHAR / CLOB / BLOB ซึ่งการแปลงรูปแบบดังกล่าวนี้เป็นสิ่งจำเป็นเมื่อคุณต้องการที่จะจัดการกับ XML ในรูปแบบข้อความ (text) เนื่องจากข้อมูลใด ๆ ที่ส่งกลับโดย XQuery จากชุดของเอกสาร XML โดยปกติแล้วจะอยู่ในรูปแบบของ XML ตัวอย่างเช่นคำสั่ง SELECT ดังต่อไปนี้จะเรียกเอกสาร XML จากคอลัมน์ DESCRIPTION และส่งข้อมูลกลับต่อเนื่องกันเป็น CLOB โดยมีสินค้าที่มี PID เป็น ‘10010001’

```
SELECT XMLSERIALIZE(DESCRIPTION AS CLOB(5K)) FROM PRODUCT WHERE PID LIKE
‘10010001’
```

เมื่อเรียกคำสั่งจะได้ผลดังนี้ :

```
1
---------------------------------------------------------------------------
<product xmlns=”http://posample.org” pid=”100-100-01”>
<description><name>Snow Shovel, Basic 22 inch</name>
<details>Basic Snow Shovel, 22 inches wide, straight handle with D-Grip</
details><price>9.99</price><weight>1 kg</weight>
</description>
</product>
1 record(s) selected.
```

8.8.3 TRANSFORM expression

TRANSFORM expression เริ่มใช้ใน DB2 9.5 ทำให้ผู้ใช้สามารถทำการแก้ไขเอกสาร XML ที่มีอยู่ ที่จัดเก็บไว้ในคอลัมน์ XML TRANSFORM expression เป็นส่วนหนึ่งของ XQuery ที่ถูกปรับปรุงขึ้น ซึ่งอำนวยความสะดวกในการดำเนินการเกี่ยวกับอินสแตนซ์ XDM ดังต่อไปนี้

- การแทรกโหนด (insert of a node)
- การลบโหนด (deletion of the node)
- การเปลี่ยนแปลงของโหนดโดยการเปลี่ยนบางส่วนของคุณสมบัติ (properties) ในขณะที่รักษาเอกลักษณ์ (identity) ของโหนดนั้นๆเอาไว้
- การสร้างสำเนาการแก้ไขของโหนดด้วยการสร้างโหนดใหม่

ด้านล่างจะเป็นตัวอย่างการใช้ TRANSFORM expression ในการปรับปรุงเอกสาร XML ของลูกค้าที่มี cid= 1000 TRANSFORM expression จะทำการปรับปรุงทั้งค่าข้อความของ element ที่เป็นหมายเลขโทรศัพท์ และค่าของแอตทริบิวต์ ‘Type’ ด้วย ‘Home’
เอกสาร XML ก่อนที่จะดำเนินการคำสั่งการปรับปรุง :

```
<customerinfo>
      <name>John Smith</name>
      <addr country=“Canada”>
            <street>Fourth</street>
            <city>Calgary</city>
            <state>Alberta</state>
            <zipcode>M1T 2A9</zipcode>
      </addr>
      <phone type=”work”>963-289-4136</phone>
</customerinfo>
```

คำสั่ง update ที่มี TRANSFORM expression :

```
update customer
set info = xmlquery( ‘copy $new := $INFO
              modify (
          do replace value of $new/customerinfo/phone with “416-123-4567”,
          do replace value of $new/customerinfo/phone/@type with “home”    )
                    return  $new’)
where cid = 1000;
```

เอกสาร XML หลังจากที่ดำเนินการคำสั่งการปรับปรุง :

```
<customerinfo>
      <name>John Smith</name>
      <addr country=“Canada”>
            <street>Fourth</street>
            <city>Calgary</city>
            <state>Alberta</state>
            <zipcode>M1T 2A9</zipcode>
      </addr>
      <phone type=”work”>963-289-4136</phone>
</customerinfo>
```

8.9 บทสรุป

XML เป็นรูปแบบข้อมูลที่มีความยืดหยุ่นมากและเหมาะที่สุดสำหรับการนำไปใช้งาน XML เป็นทางเลือกสำหรับโปรแกรมประยุกต์ที่เหมาะกับระบบที่มักจะมีการเปลี่ยนแปลง หรือใช้สำหรับออบเจ็กต์ที่มีความซับซ้อนหรือเป็นลำดับชั้น ความสามารถในการแสดงข้อมูลที่เป็นกึ่งโครงสร้าง (semi-structured) XML เป็นทางเลือกที่ดีสำหรับใช้ในการแลกเปลี่ยนข้อมูลและบูรณาการข้อมูลจากหลายๆแหล่ง ซึ่ง IBM DB2 ให้การสนับสนุนสำหรับการจัดเก็บเอกสาร XML (โดยใช้ XML Native) ทำให้สามารถจัดการข้อมูล XML ได้อย่างมีประสิทธิภาพและช่วยอำนวยความสะดวกในการสอบถามข้อมูล ซึ่งผู้ใช้มีทางเลือกอยู่ 2 ทางเลือกคือ XQuery หรือ SQL/XML ขึ้นอยู่กับข้อมูลที่จะทำการเข้าถึง นอกจากนี้ DB2 ยังมีความสามารถในการจัดการ XML schema รวมถึงการตรวจสอบ (validate) เอกสาร XML โดยใช้ XML schema ฟังก์ชันในการ TRANSFORM ของ DB2 เป็นวิธีที่ง่ายและสามารถทำการเปลี่ยนแปลงเอกสาร XML โดยไม่ต้องปรับแก้ไขในระดับของโปรแกรมประยุกต์

8.10 แบบฝึกหัด

1. ให้ทำการสร้างตารางข้อมูลที่มีคอลัมน์เชิง สัมพันธ์ หนึ่งคอลัมน์ (cid) และอีกหนึ่งคอลัมน์ XML จากนั้นให้ทำการ insert เอกสาร XML ดังโครงสร้าง XML ต่อไปนี้

```
<customer id=”C62”>
  <firstname>Pete</firstname>
  <lastname>Bush</lastname>
  <address>
     <door>No 34</door>
<building>Galaxy Apartment</building>
<road>Meera road</road>
```

```
<city>Mumbai</city>
<zip>411202</zip>
 </address>
</customer>
```

2. INSERT รายละเอียดของลูกค้า 5 คน ซึ่งเป็นเอกสาร XML โดยใช้โครงสร้าง XML เช่นเดียวกับที่กล่าวถึงข้างต้นโดยใช้คำสั่ง INSERT โดยให้แน่ใจว่า CID ของลูกค้าคือ 10, 13, 15, 20, 23 ตามลำดับ
3. ทำการ SELECT ในตารางข้อมูลเพื่อดึงข้อมูลทั้งหมดที่เป็นข้อมูลเชิงสัมพันธ์และข้อมูล XML
4. เขียน XQuery เพื่อเรียกเอกสาร XML ทั้งหมดที่เก็บไว้ในตารางข้อมูล
5. เขียน XQuery เพื่อเรียกเอกสาร XML โดยเลือกเอกสาร XML ที่มี cid ระหว่าง 10 และ 20 เท่านั้น

8.11 คำถามท้ายบท

1. ฟังก์ชันของ SQL / XML ใดต่อไปนี้ไม่ได้เป็น ฟังก์ชัน Publishing?
   A. XMLELEMENT
   B. XMLATTRIBUTE
   C. XMLCONCAT
   D. XMLPARSE
   E. ไม่มีข้อใดถูกต้อง

2. กำหนดโครงสร้างของตารางดังต่อไปนี้

```
create table clients(
id              int primary key not null,
name            varchar(50),
status  varchar(10),
contactinfo     xml )
```

และข้อมูล XML ในคอลัมน์ contactinfo

```
<customerinfo>
        <name>Kathy Smith>
        <addr country=”Canada>
                <city>Toronto</city>
                <prov-state>Ontario</prov-state>
                <zip>M5H-4C9</zip>
        </addr>
</customerinfo>
<customerinfo>
        <name>Amit Singh>
        <addr country=”Canada>
```

```
                    <city>Markham</city>
                    <prov-state>Ontario</prov-state>
                    <zip>N9C-3T6</zip>
                </addr>
            </customerinfo>
```

ผลลัพธ์ของการ Query ต่อไปนี้คืออะไร?

```
select xmlquery('$c/customerinfo/addr/city[1]'
        passing info as "c") \
from xmlcustomer \
where xmlexists('$c/customerinfo/addr[prov-state="Ontario"]' \
passing xmlcustomer.info as "c")
```

A. Toronto
   Markham
B. <City>Toronto</City>
   <City>Markham</City>
C. <City>Toronto
   Markham</City>
D. ไม่มีข้อใดถูกต้อง

3. XML parsing เป็นกระบวนการของการแปลงเรื่องใด
   A. ข้อมูลเชิงสัมพันธ์จากรูปแบบเชิงสัมพันธ์ (relational format) ไปเป็น รูปแบบลำดับชั้น (hierarchical format)
   B. ข้อมูล XML จากรูปแบบสตริงที่ต่อเนื่อง (serialized string format) ไปเป็น รูปแบบลำดับชั้น
   C. ข้อมูล XML จากรูปแบบลำดับชั้นไปเป็นการจัดรูปแบบสตริงที่ต่อเนื่อง
   D. ข้อมูล XML จากรูปแบบลำดับชั้นไปเป็น รูปแบบเชิงสัมพันธ์
   E. ไม่มีข้อใดถูกต้อง

4. FLWOR ย่อมาจากอะไร
   A. For, Lower, Where, Or, Run
   B. From, Let, Where, Or, Reset
   C. For, Let, Where, Order by, Reset
   D. For, Let, Where, Order by, Return

5. เมื่อเรียกใช้ XQuery ภายใน SQL สิ่งที่ควรระวังคืออะไร
   A. XQuery เป็น Case sensitive ในขณะที่ SQL ไม่ได้เป็น Case Sensitive
   B. ทั้ง XQuery และ SQL เป็น Case Sensitive
   C. SQL เป็น Case sensitive ในขณะที่ XQuery ไม่ได้เป็น Case Sensitive
   D. ไม่จำเป็นต้องระวัง

E. ไม่มีข้อใดถูกต้อง

6. คำสั่ง XQuery ต่อไปนี้หมายถึงอะไร

" xquery db2-fn:xmlcolumn('NNXML1.XMLCOL')/a/b "

A. ดึงข้อมูลจาก element b ซึ่งเป็น element ลูกของ element a จากคอลัมน์ XMLCOL ในตารางข้อมูล NNXML1

B. ดึงข้อมูลจาก element a ซึ่งเป็น element ลูกของ root node ชื่อ b จากคอลัมน์ XMLCOL ในตารางข้อมูล NNXML1

C. ดึงข้อมูลจาก element b ซึ่งเป็น element ลูกของroot node a จากคอลัมน์ XMLCOL ในตารางข้อมูล NNXML1

D. ดึงข้อมูลจาก element a จากคอลัมน์ XMLCOL ในตารางข้อมูล NNXML1

7. ถ้าตารางข้อมูลต่อไปนี้มีคอลัมน์ที่เก็บเอกสาร XML คอลัมน์เดียว คือ คอลัมน์ DOC ดังแสดงไว้ด้านล่าง

Table description: CONTEST (DOC XML)

```
<dept bldg=»111»>
   <employee id=»901»>
<name>Ajit Patil</name>
        <phone>567 789 1342</phone>
        <office>124</office>
  </employee>
  <employee id="922">
        <name>Peter Jose</name>
        <phone>121 768 3456</phone>
        <office>213</office>
  </employee>
</dept>
```

คำสั่ง Query ใดต่อไปนี้จะได้ element ที่เป็น <name>Peter Jose</name> ?

A. db2-fn:xmlcolumn('CONTEST.DOC')/dept/employee[@id="922"]/name

B. select xmlquery('$d/dept/employee[@id="922"]/name' passing DOC as "d") from contest

C. ทั้ง A และ B

D. ไม่มีข้อใดถูกต้อง

8. คำสั่ง XQuery ใดที่ให้ผลลัพธ์เดียวกันกับ XQuery ต่อไปนี้

xquery db2-fn:xmlcolumn('CONTEST.DOC')/dept/employee[@id="922"]/name

A. xquery db2-fn:xmlcolumn('CONTEST.DOC') /dept/employee/name[../@id="922"]

B. xquery db2-fn:xmlcolumn('CONTEST.DOC') /dept/employee[../@id="922"]/name

C. xquery db2-fn:xmlcolumn('CONTEST.DOC') /dept/employee/name[@id="922"]

D. ไม่มีข้อใดถูกต้อง

9. ใน DB2 9.7 ข้อใดต่อไปนี้เป็นจริงสำหรับคำสั่ง update ต่อไปนี้ ถ้า XPath expression '$new/customer/phone' ส่งข้อมูล phone element กลับมามากกว่าหนึ่งชุด

```
update customer
set info =
 xmlquery('copy $new := $information
   modify do replace value of $new/customer/phone with "091-454-8654"
   return  $new')
where cid = 67;
```

A. คำสั่ง UPDATE จะล้มเหลวและข้อความผิดพลาดจะปรากฏขึ้นที่หน้าจอ

B. คำสั่ง UPDATE จะแทนที่ phone element ทั้งหมด ด้วย phone element ใหม่ ที่มีข้อความเป็น "091-454-8654"

C. คำสั่ง UPDATE จะแทนที่เฉพาะ phone element แรกที่เจอ ด้วย phone element ที่มีข้อความเป็น "091-454-8654"

D. ไม่มีข้อใดถูกต้อง

9

บทที่ 9 – ความปลอดภัยของฐานข้อมูล

เทคโนโลยีสารสนเทศในปัจจุบันมีการพัฒนาไปอย่างรวดเร็ว องค์กรต่างๆมีการจัดเก็บข้อมูลที่เกี่ยวข้องกับกิจกรรมและการดำเนินงานในด้านต่างๆที่มีปริมาณมากยิ่งขึ้น ซึ่งแน่นอนว่าข้อมูลเหล่านี้มีประโยชน์ในการประกอบการวิเคราะห์และการตัดสินใจและเป็นทรัพยากรอันมีค่าสำคัญขององค์กร ดังนั้นการให้ความสำคัญของความปลอดภัยของข้อมูลจึงเป็นเรื่องที่จำเป็น ซึ่งทุกองค์กรควรให้ความสำคัญต่อภัยคุกคามในด้านต่างๆ รวมถึงความจำเป็นที่จะต้องมีมาตรการการป้องกันความปลอดภัยข้อมูลภายในองค์กร สำหรับเนื้อหาของบทนี้จะอธิบายถึงเรื่องความปลอดภัยของฐานข้อมูลเป็นสำคัญ ประกอบด้วยแนวคิดของการควบคุมและความปลอดภัยในแง่มุมต่างๆที่เกี่ยวกับนโยบายการรักษาความปลอดภัยสำหรับการจัดการฐานข้อมูล

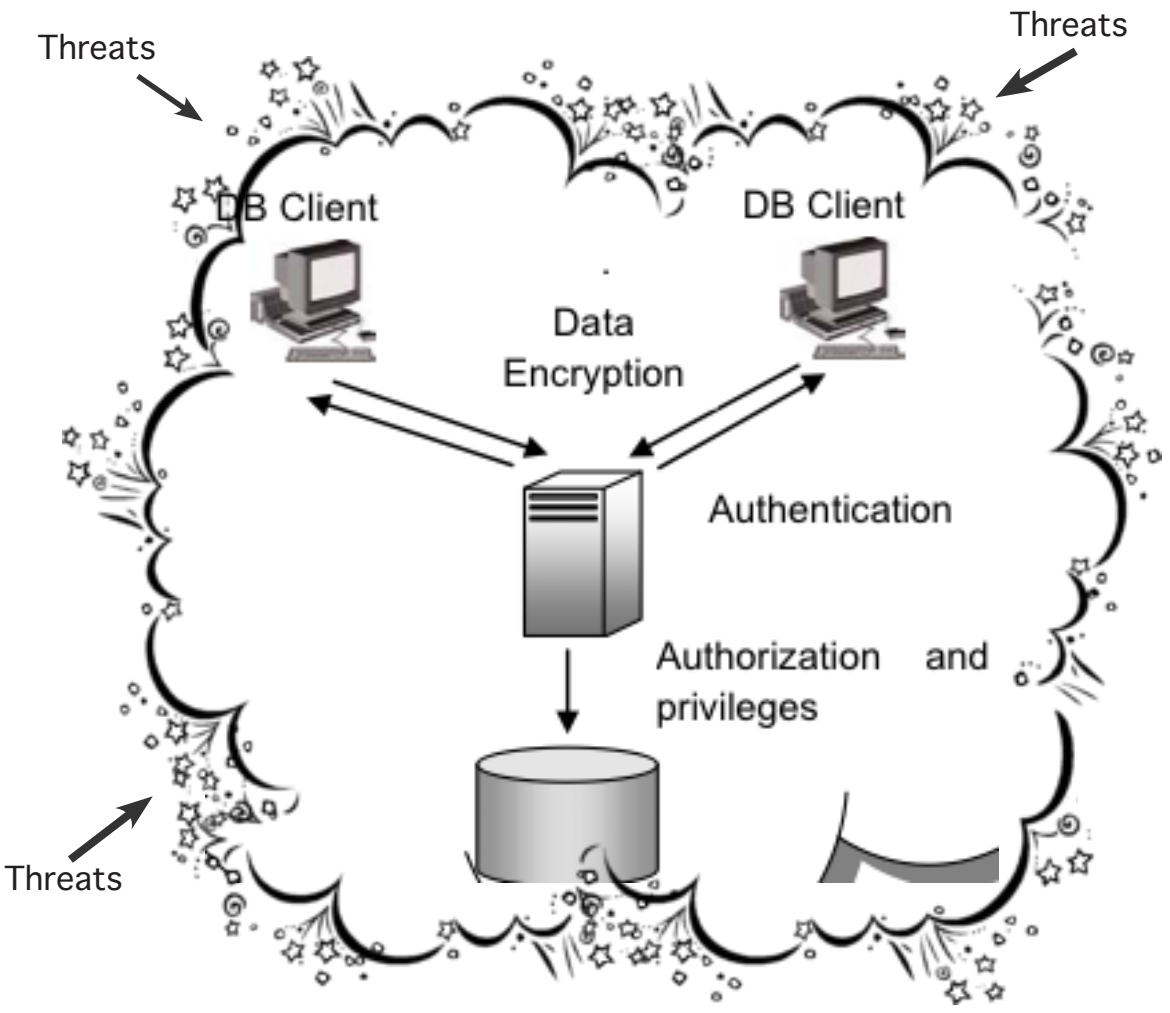
9.1 ภาพรวมของความปลอดภัยของฐานข้อมูล

โดยปกติ ปัญหาที่เกี่ยวข้องกับการรักษาความปลอดภัยของข้อมูลเป็นปัญหาที่ซับซ้อน และเกี่ยวข้องกับประเด็นของกฎหมาย และทางด้านสังคมหรือจริยธรรม ปัญหาที่เกี่ยวข้องกับนโยบายการใช้งาน หรือเกี่ยวข้องกับการควบคุมอุปกรณ์ทางกายภาพ ความปลอดภัยของฐานข้อมูลจะเกี่ยวข้องกับการป้องกันฐานข้อมูลจากภัยคุกคามที่เกิดขึ้นโดยความตั้งใจหรือไม่ตั้งใจ ซึ่งปัจจัยสำคัญของการรักษาความปลอดภัยอาจจะเกี่ยวข้องหรือไม่เกี่ยวข้องกับอุปกรณ์คอมพิวเตอร์

การวิเคราะห์การรักษาความปลอดภัยฐานข้อมูลจะไม่ได้ หมายถึงเฉพาะฟังก์ชันต่างๆที่มีมาในระบบฐานข้อมูลเชิงสัมพันธ์เท่านั้น แต่ขอบเขตของประเด็นปัญหาของความปลอดภัยและฐานข้อมูลนั้นจะครอบคลุมไปถึงสภาพแวดล้อมอื่นๆ อีกด้วย นอกจากนี้ ข้อควรพิจารณาด้านความปลอดภัยไม่ได้หมายถึงเพียงแค่ข้อมูลที่จัดเก็บอยู่ในฐานข้อมูลเท่านั้น แต่อาจจะเกิดจากช่องโหว่ของความปลอดภัย ในส่วนอื่นๆของระบบซึ่งอาจจะส่งผลกระทบต่อฐานข้อมูล

นอกจากนี้การที่เราจะเน้นไปที่การรักษาความปลอดภัยของฐานข้อมูลเพียงอย่างเดียวนั้น อาจจะไม่ได้ทำให้ข้อมูลมีความปลอดภัย แต่จำเป็นต้องพิจารณาถึงองค์ประกอบอื่นๆทั้งระบบเครือข่าย ระบบปฏิบัติการ และสถานที่ที่เราติดตั้งระบบฐานข้อมูล เช่น อาคาร รวมถึงบุคคลที่สามารถเข้าถึงระบบฐานข้อมูล ก็เป็นสิ่งจำเป็นในการคำนึงถึงการรักษาความปลอดภัย

รูป 9.1 แสดงให้เห็นถึงภาพรวมของความปลอดภัยของฐานข้อมูล



รูป 9.1 –ภาพรวมของการรักษาความปลอดภัยของฐานข้อมูล

การออกแบบ และการติดตั้งความปลอดภัยให้กับฐานข้อมูลเกี่ยวข้องกับวัตถุประสงค์ต่อไปนี้:

- ข้อมูลส่วนบุคคล (Privacy) หมายถึง ผู้ใช้ที่ไม่มีสิทธิ์ จะต้องไม่สามารถเห็นข้อมูล
- ความสมบูรณ์ (Integrity) หมายถึงการที่อนุญาตให้ เฉพาะผู้ใช้ที่มีสิทธิ์ สามารถเปลี่ยนแปลงข้อมูล
- การมีให้ (Availability) หมายถึงการที่ผู้ใช้ที่มีสิทธิ์จะต้องสามารถ ใช้สิทธิ์ที่เขามีได้

เพื่อให้บรรลุวัตถุประสงค์เหล่านี้ องค์กรจำเป็นต้องมีการประกาศเป็นนโยบายและมาตรการด้านความปลอดภัยของฐานข้อมูลที่ชัดเจน โดยเฉพาะการกำหนดกลุ่มผู้ใช้ที่สามารถเข้าถึงฐานข้อมูล และประเภทข้อมูลที่สามารถเข้าถึงได้ นอกจากนี้เราจำเป็นที่จะต้องระบุกระบวนการในการเรียกใช้ข้อมูลและการเข้าถึงข้อมูล

สำหรับการป้องกันความปลอดภัยของระบบฐานข้อมูลเชิงสัมพันธ์นั้น เราจำเป็นต้องพึ่งพากลไกของการรักษาความปลอดภัยของข้อมูลที่มีอยู่ในระบบฐานข้อมูลเชิงสัมพันธ์ (DBMS) หรือระบบปฏิบัติการ และสำหรับกลไกภายนอกที่เกี่ยวข้อง เช่น การรักษาความปลอดภัยในการเข้าถึงอาคารหรือห้องที่จัดเก็บระบบฐานข้อมูลในเชิงทางกายภาพนั้นจะไม่อยู่ในขอบเขตของหนังสือเล่มนี้

สำหรับบุคลากรสำคัญที่มีหน้าที่รับผิดชอบด้านการรักษาความปลอดภัยของฐานข้อมูลนั้นได้แก่ผู้ดูแลระบบฐานข้อมูล (Data Base Administrator: DBA) ซึ่งเป็นผู้ที่ต้องพิจารณาถึงปัจจัยความเสี่ยงต่างๆที่อาจจะก่อให้เกิดปัญหาเรื่องการคุกคามในระบบ นอกจากนี้ DBA ยังเป็นผู้ที่กำหนดว่าผู้ใช้งานคนไหนบ้างที่สามารถเข้า

ถึงฐานข้อมูล หรือข้อมูลส่วนไหนบ้างที่สามารถเข้าถึงและการจัดการกับฐานข้อมูลอะไรบ้างที่จะสามารถดำเนินการได้

9.1.1 ความจำเป็นสำหรับความปลอดภัยของฐานข้อมูล

ปัจจุบันความปลอดภัยของฐานข้อมูลเป็นประเด็นที่สำคัญเนื่องมาจากการเพิ่มขึ้นของปริมาณข้อมูล รวมถึงความสำคัญของข้อมูลที่มีการรวบรวมและจัดเก็บในระบบคอมพิวเตอร์ตามหน่วยงานต่างๆ ทำให้คนตระหนักถึงว่าข้อมูลจำเป็นต้องมีความพร้อมในการใช้งาน และในกรณีที่ข้อมูลมีการสูญหาย ก็จะส่งผลกระทบและก่อให้เกิดความเสียหายตามมา ดังนั้นเราจึงมองว่าฐานข้อมูลว่าเป็นทรัพยากรสำคัญที่ควรให้ความสำคัญในเรื่องของความปลอดภัย และมีระบบการควบคุมอย่างเหมาะสม

ภัยคุกคามที่เกิดแก่ความปลอดภัยของข้อมูลอาจเป็นสถานการณ์หรือเหตุการณ์ที่เกิดขึ้นโดยตั้งใจหรือไม่ตั้งใจ ซึ่งแน่นอนว่าจะส่งผลกระทบต่อองค์กรในที่สุด ซึ่งความเสียหายใดๆที่เกิดขึ้นอาจก่อให้เกิดผลกระทบโดยตรง คือทำให้สูญเสียข้อมูล หรือผลกระทบทางอ้อมคือการสูญเสียความน่าเชื่อถือขององค์กร ซึ่งทำให้ลูกค้ามีความเชื่อมั่นลดลง ดังนั้นภัยคุกคามต่างๆอาจถูกมองได้ว่าเป็นภัยที่จะส่งผลกระทบต่อระบบความปลอดภัย ซึ่งจะมีผลกระทบต่อการดำเนินงานขององค์กรโดยตรง ตัวอย่างของที่มาของภัยคุกคามที่อาจเกิดขึ้น มีดังต่อไปนี้

- การใช้งานข้อมูลโดยผู้ใช้ที่มีสื่อแบบดิสก์ เช่น ซีดีรอม
- การรวบรวมและการ ก็อปปี้ข้อมูลโดยผู้ที่ไม่ได้รับอนุญาต
- การแก้ไขและปรับปรุงโปรแกรม
- การเข้าถึงข้อมูลในลักษณะของการบุกรุกโดยแฮกเกอร์
- การโจรกรรมข้อมูล โปรแกรม หรืออุปกรณ์คอมพิวเตอร์
- การฝึกอบรมเจ้าหน้าที่ที่ไม่มีประสิทธิภาพ
- การที่ข้อมูลถูกเปิดเผยโดยไม่ได้รับอนุญาต
- การเกิดภัยพิบัติ เนื่องมาจากสาเหตุของไฟไหม้ น้ำท่วม หรือระเบิด
- การที่สายเคเบิลถูกตัดขาดหรือไม่สามารถเชื่อมต่อ
- การถูกโจมตีโดยคอมพิวเตอร์ไวรัส

ผลกระทบของความรุนแรงที่องค์กรได้รับผลจากภัยคุกคามนั้นอาจเกิดขึ้นได้หลายระดับขึ้นอยู่กับแนวทางการป้องกัน ความเข้มงวดของมาตรการการรักษาความปลอดภัยและการวางแผนล่วงหน้า ตัวอย่างเช่น กรณีที่พบปัญหาความผิดพลาดของระบบฮาร์ดแวร์ ซึ่งจำเป็นต้องใช้ข้อมูลจากที่เก็บข้อมูลสำรอง (secondary storage) ซึ่งจะส่งผลกระทบต่อการดำเนินการและกิจกรรมการประมวลผลอื่นๆทั้งหมดขององค์กร จนกว่าปัญหาจะได้รับการแก้ไข ระยะเวลาที่ระบบหยุดทำงาน จะขึ้นอยู่กับความเร็วในการกู้คืนฐานข้อมูล และองค์ประกอบอื่นๆทั้งฮาร์ดแวร์และซอฟต์แวร์ รวมทั้งขึ้นอยู่กับข้อมูลทั้งหมดมีการสำรอง (backup) ครั้งล่าสุดเมื่อไร ซึ่งปัจจัยดังกล่าวนี้เป็นตัวกำหนดเวลาทั้งหมดที่จะต้องใช้สำหรับการกู้คืนระบบ และโอกาสความเป็นไปได้ที่ต้องสูญเสียข้อมูลที่ไม่สามารถกู้คืนได้

องค์กรต่างๆควรมีการวิเคราะห์ถึงภัยคุกคามที่อาจจะเกิดขึ้น รวมถึงควรกำหนดแผนและมาตรการการป้องกัน รวมถึงแผนงบประมาณและค่าใช้จ่ายสำหรับการป้องกันภัยที่อาจจะเกิดขึ้น ดังนั้นการที่ข้อมูลจะมีความปลอดภัย ระบบคอมพิวเตอร์รวมถึงสภาพแวดล้อมโดยรวมจะต้องมีความปลอดภัยด้วย
ประเด็นที่ควรให้ความสนใจในแผนการรักษาความปลอดภัยมีดังต่อไปนี้

- การโจรกรรมและการฉ้อโกง การดำเนินการเหล่านี้เป็นความผิดพลาดที่เกิดขึ้นโดยมนุษย์โดยที่อาจจะเกี่ยวข้องหรือไม่เกี่ยวข้องกับข้อมูล กรณีนี้เรามุ่งเน้นให้ความสนใจเกี่ยวกับ สถานที่ที่เก็บข้อมูล หรือโปรแกรมประยุกต์ในเชิงกายภาพ โดยมุ่งเน้นในเรื่องของการเข้าถึง สถานที่ เช่น การอนุญาตให้บุคคลที่มีส่วนเกี่ยวข้องโดยตรงสามารถเข้าไปยังห้องที่ติดตั้งเครื่องคอมพิวเตอร์แม่ข่าย หรือแฟ้มข้อมูลคอมพิวเตอร์ การใช้อุปกรณ์ไฟล์วอลล์เพื่อป้องกันไม่ให้ผู้ที่ไม่มีสิทธิ์เข้าถึงส่วนต่างๆของฐานข้อมูล ผ่านทางการสื่อสารของระบบเครือข่ายจากหน่วยงานภายนอก ซึ่งจะเป็นกระบวนการที่จะป้องกันผู้ที่มีเจตนาในการขโมยหรือการฉ้อโกงได้

- การสูญเสียความลับของข้อมูลขององค์กร ข้อมูลส่วนบุคคลถือได้ว่าเป็นความลับที่ต้องมีมาตรการในการจัดเก็บรักษาความปลอดภัยและความเป็นส่วนตัวของข้อมูล ทั้งนี้รวมถึงข้อมูลส่วนบุคคลของบุคลากรภายในองค์กร การสูญหายของข้อมูลอาจนำไปสู่การสูญเสียความได้เปรียบของการแข่งขันในเชิงธุรกิจ และความล้มเหลวในการควบคุมความปลอดภัยของข้อมูลส่วนบุคคลจากการโจรกรรมรหัสผ่านผู้ใช้ ซึ่งอาจนำไปสู่การกรรโชก การติดสินบน และการ สูญเสียความน่าเชื่อถือขององค์กร ซึ่งปัญหาต่างๆเหล่านี้อาจเป็นหนทางที่จะนำองค์กรไปสู่ความยุ่งยากที่เกี่ยวข้องกับคดีความทางกฎหมาย

- การสูญเสียความสมบูรณ์ของข้อมูล ซึ่งอาจก่อให้เกิดปัญหาในเรื่องการขาดความสมบูรณ์ของข้อมูลจากความผิดพลาดหรือความเสียหาย กรณีนี้อาจทำให้องค์กรนำข้อมูลที่ผิดพลาดไปใช้ในการประกอบการตัดสินใจและทำให้ประสบปัญหาการขาดทุนจากการตัดสินใจที่ผิดพลาด
- การสูญเสียความพร้อมในการใช้งาน ซึ่งหมายถึงระบบฮาร์ดแวร์ ซอฟต์แวร์ หรือความพร้อมของข้อมูลที่ผู้ใช้ไม่สามารถใช้งานได้ตามปกติ ซึ่งบางครั้งอาจเกิดขึ้นจากการปรับปรุงแก้ไขข้อมูล หรือจากปัญหาที่รุนแรงมากขึ้นจากความล้มเหลวของการทำงานของอุปกรณ์ฮาร์ดแวร์ ระบบเครือข่าย หรือโปรแกรมประยุกต์ที่ถูกคอมพิวเตอร์ไวรัสโจมตี
- การสูญเสียข้อมูลจากอุบัติเหตุโดยไม่ตั้งใจ ทั้งนี้อาจเป็นผลของข้อผิดพลาดที่เกิดขึ้นจากตัวบุคคลหรือ ช่องโหว่ของซอฟต์แวร์และฮาร์ดแวร์ การหลีกเลี่ยงการสูญเสียข้อมูลโดยไม่ได้ตั้งใจ องค์กรควรสร้างกระบวนการที่ชัดเจนสำหรับการอนุญาตผู้ใช้ในการติดตั้งซอฟต์แวร์และบำรุงรักษาฮาร์ดแวร์ ซึ่งประโยชน์ของการกำหนดเป็นแนวนโยบายและกระบวนการที่ชัดเจนจะช่วยลดการสูญเสียให้เกิดขึ้นน้อยที่สุด

แนวทางการรักษาความปลอดภัยของฐานข้อมูลดังที่กล่าวมา มีจุดมุ่งหวังเพื่อให้ลดการสูญเสียขององค์กรที่อาจจะเกิดขึ้น ทั้งนี้สาเหตุมาจากปริมาณอาชญากรรมทางคอมพิวเตอร์ที่มีแนวโน้มที่สูงขึ้น ซึ่งอาชญากรรมเหล่านี้จะส่งผลกระทบต่อองค์ประกอบต่างๆของระบบโดยรวม ซึ่งองค์กรจำเป็นต้องให้ความสำคัญต่อมาตรการการรักษาความปลอดภัยเป็นสำคัญ มาตรการส่วนใหญ่ที่ใช้เพื่อการป้องกันและรักษาความปลอดภัยของข้อมูล ประกอบด้วยการสร้างนโยบายการควบคุมสิทธิของการเข้าถึง (access control) การใช้วิว (view) การเข้ารหัสข้อมูลที่เป็นความลับ (encryption) การควบคุมความถูกต้อง (integrity control) ทั้งนี้จะครอบคลุมถึงแนวนโยบาย (policy) กระบวนการ (procedure) และการป้องกันในเชิงกายภาพ

9.1.2 การควบคุมการเข้าถึง (Access Control)

การรักษาความปลอดภัยของข้อมูลในทางปฏิบัตินั้นมีอยู่ 2 วิธี ประกอบด้วย การควบคุมในเชิงคุณภาพของการรักษาความปลอดภัย (Discretionary control) และการควบคุมแบบบังคับที่จะต้องทำ (Mandatory control) ทั้งสองกรณีนี้การป้องกันข้อมูลอาจจะทำได้ตั้งแต่ระดับฐานข้อมูลจนถึงหน่วยข้อมูลระดับแถว โดยวิธีแรกนั้นผู้ใช้งานจะได้รับสิทธิ์ในการเข้าถึงออบเจ็กต์ (access rights หรือ privileges) ที่ต่างกัน ตัวอย่างเช่น การควบคุมสิทธิ์ของผู้ใช้ A สามารถเข้าถึงวัตถุ X ของฐานข้อมูล แต่ไม่สามารถเข้าถึงวัตถุ Y ในขณะที่ผู้ใช้ B สามารถเข้าถึงวัตถุ Y ได้แต่ไม่สามารถเข้าถึง X การควบคุมสิทธิ์แบบการควบคุมในเชิงคุณภาพนั้นมีความยืดหยุ่นมาก เราสามารถรวมสิทธิ์ในการเข้าถึงแบบต่างๆและกำหนดสิทธิ์นั้นให้แก่ผู้ใช้และออบเจ็กต์ได้ตามความต้องการ

ในกรณีของการควบคุมแบบบังคับ แต่ละออบเจ็กต์ข้อมูล เช่น ตาราง จะมีการแบ่งออกเป็นประเภทต่างๆ และแต่ละผู้ใช้จะถูกกำหนดระดับของสิทธิ์ที่สัมพันธ์กันกับประเภทของออบเจ็กต์ข้อมูล ดังนั้นออบเจ็กต์ข้อมูลจะสามารถเข้าถึงได้โดยผู้ใช้ที่มีสิทธิ์ที่เหมาะสมเท่านั้น ซึ่งการกำหนดสิทธิ์ของการควบคุมแบบบังคับนั้นจะมีโครงสร้างแบบลำดับชั้น (Hierarchy) ดังนั้นจึงมีความยืดหยุ่นน้อยกว่า การควบคุมในเชิงคุณภาพ สำหรับการออกแบบโครงสร้างของการรักษาความปลอดภัย ไม่ว่าจะเป็นประเด็นเรื่องของรูปแบบ การกำหนดสิทธิ์ต่างๆ จะขึ้นอยู่กับรูปแบบการดำเนินการทางธุรกิจเป็นสำคัญ ไม่ใช่ประเด็นทางเทคนิค การตัดสินใจที่จะกำหนดประเภทของความปลอดภัยและสิทธิ์ใดๆให้แก่ผู้ใช้ สิ่งสำคัญคือระบบจำเป็นต้องจดจำแหล่งที่มาของการร้องขอหรือจดจำผู้ใช้งานในแต่ละแอพพลิเคชัน รวมถึงการตรวจสอบสิทธิ์ของผู้ใช้

9.1.3 กรณีศึกษาเรื่องการรักษาความปลอดภัยของฐานข้อมูล

ในส่วนนี้เราจะทำการศึกษาว่า IBM DB2 ซึ่งเป็นหนึ่งในผลิตภัณฑ์ชั้นนำในเรื่องของแม่ข่ายฐานข้อมูลจะแก้ไขปัญหาด้วยกลไกการรักษาความปลอดภัยอย่างไร กลไกการรักษาความปลอดภัยของ DB2 ประกอบด้วยการตรวจสอบด้วยวิธีการยืนยันตัวตน (authentication) และการอนุญาต (authorization) ที่เป็นองค์ประกอบหลักซึ่งทำงานร่วมกันในรักษาความปลอดภัยในการเข้าถึงทรัพยากรต่างๆ ของ DB2

9.1.3.1 การยืนยันตัวตน (Authentication)

การยืนยันตัวตนเป็นการกระทำที่เกิดขึ้นเป็นสิ่งแรกเมื่อมีการเชื่อมต่อกับฐานข้อมูล DB2 โดยเป็นกระบวนการที่ผู้ใช้จะต้องพิสูจน์ตัวตนก่อนการเข้าใช้งานฐานข้อมูล การตรวจสอบข้อมูลเฉพาะตัวระดับผู้ใช้งานแต่ละคนหรือระดับกลุ่มนั้น DB2 ทำงานร่วมกับระบบการรักษาความปลอดภัยอื่นๆที่อยู่ภายนอกระบบฐานข้อมูล เช่น การทำงานร่วมกับระบบปฏิบัติการ หรือบางครั้งเป็นการใช้ฟังก์ชันการรักษาความปลอดภัยระบบอื่นๆเช่น Kerberos หรือ Lightweight Directory Access Protocol (LDAP) เพื่อรับรองความถูกต้อง ซึ่งผู้ใช้จำเป็นต้องมี รหัสผู้ใช้ และรหัสผ่านในการยืนยันตัวตน

รหัสผู้ใช้จะช่วยให้คอมโพเนนต์ที่เกี่ยวข้องกับความปลอดภัยในการระบุตัวตนของผู้ใช้ สามารถทำการ

ตรวจสอบข้อมูลเฉพาะตัวของผู้ใช้โดยการใช้รหัสผ่าน หรือ เพื่อให้เกิดความยืดหยุ่นมากขึ้นเราสามารถสร้างการตรวจสอบความปลอดภัยด้วยวิธีการยืนยันตัวตนขึ้นมาเองแล้วทำการปลั๊กอินโมดูลดังกล่าวให้ใช้งานร่วมกับ DB2 หลังจากผ่านขั้นตอนในการยืนยันตัวตนแล้ว รหัสผู้ใช้จะถูกใช้เป็น authorization ID โดย DB2 จะใช้ authorization ID ดังกล่าวในการตรวจสอบสิทธิ์ในการเชื่อมต่อ (connect privilege) กับฐานข้อมูล

9.1.3.2 การอนุญาต (Authorization)

หลังจากที่ผู้ใช้ผ่านขั้นตอนการยืนยันตัวตน แล้ว สิ่งที่ต้องทำต่อไปคือการตรวจสอบว่าผู้ใช้นั้นได้รับอนุญาตให้เข้าถึงข้อมูลหรือทรัพยากรใดได้บ้าง การอนุญาตเป็นการกระบวนการการให้สิทธิแก่ผู้ใช้ (grant privilege)ในการเข้าใช้งานระบบหรือเข้าถึงออบเจ็กต์ตามนโยบายที่กำหนดในระบบ คำจำกัดความของการอนุญาตจะประกอบด้วย ซับเจ็กต์ และ ออบเจ็กต์ โดยที่ ซับเจ็กต์ จะเกี่ยวข้องกับผู้ใช้ และ โปรแกรม ส่วนออบเจ็กต์ จะเกี่ยวข้องกับ ตาราง วิว แอพพลิเคชัน procedure หรือออบเจ็กต์อื่นๆที่มีอยู่ในระบบ

การควบคุมการอนุญาต สามารถควบคุมได้โดยซอฟต์แวร์ที่ถูกพัฒนาขึ้น และสามารถกำหนดสิทธิการเข้าถึงทั้งในส่วนของระบบ และออบเจ็กต์ โดยสามารถกำหนดได้ว่าผู้ใช้คนไหนบ้างมีสิทธิ์ในการเข้าถึง และผู้ใช้สามารถทำอะไรบ้างกับออบเจ็กต์ ด้วยเหตุนี้เราจึงเรียกการอนุญาต ว่าการควบคุมการเข้าถึง (access control) ได้อีกด้วย ตัวอย่างเช่นการอนุญาตให้ผู้ใช้สามารถอ่าน แถวในฐานข้อมูล แต่ไม่สามารถปรับเปลี่ยนแก้ไขหรือแทรก แถวใหม่ได้

กฎในการควบคุมการอนุญาตสามารถกำหนดได้ในระบบฐานข้อมูล (DBMS) เพื่อใช้สำหรับการควบคุมและจำกัดสิทธิ์สำหรับการเข้าถึงข้อมูล การเก็บข้อมูลสิทธิที่เรากำหนดขึ้นเหล่านี้จะถูกเก็บไว้ในตารางและแฟ้มการกำหนดค่า (configuration files) ใน DB2 ที่จะบันทึกสิทธิต่างๆที่เกี่ยวข้องในการใช้งานฐานข้อมูล

ทุกครั้งที่ผู้ใช้งานพยายามที่จะใช้งานระบบและเข้าถึงข้อมูล ระบบจะทำการตรวจสอบ authorization ID ของผู้ใช้ และสิทธิ์ที่มีของผู้ใช้ที่ถูกกำหนดขึ้นไม่ว่าในทางตรง หรือทางอ้อม (จากการกำหนดสิทธิ์ผ่านกลุ่ม หรือ Role) แล้วทำการเปรียบเทียบกับ สิทธิที่เก็บไว้ในฐานข้อมูล ซึ่งผลของการเปรียบเทียบทำให้ฐานข้อมูลทราบว่า จะอนุญาตหรือปฏิเสธการเข้าถึงดังกล่าว

โดยปกติแล้ว การกำหนดสิทธิให้ผู้ใช้งานสามารถทำงานกับฐานข้อมูลสามารถกำหนดสิทธิให้แก่ผู้ใช้นั้นๆได้โดยผ่านทางรหัสผู้ใช้ (authorization ID) โดยตรง หรือบางครั้งการกำหนดสิทธิสามารถกระทำผ่านกลุ่มผู้ใช้ (Group หรือ Role) ที่ผู้ใช้ดังกล่าวเป็นสมาชิกอยู่ นอกจากนี้เรายังสามารถให้สิทธิ์ผ่านกลุ่มผู้ใช้แบบสาธารณะ (Public Group) ได้ ส่วนสิทธิ์ที่เกี่ยวข้องกับ context-sensitive สามารถกำหนดได้ผ่าน trusted context role

DB2 สามารถให้สิทธิ์แก่ผู้ใช้ทั้งในรูปแบบที่เป็น implicit และ explicit โดยที่ explicit เป็นการให้สิทธิ์โดยตรงแก่ผู้ใช้ กลุ่มของผู้ใช้ หรือ role ผ่านคำสั่ง Grant ส่วน implicit เป็นสิทธิ์ที่ผู้ใช้ได้รับจากการเป็นสมาชิกของกลุ่ม

DB2 มีรูปแบบในการควบคุมสิทธิอยู่ 3 ลักษณะด้วยกัน คือสิทธิ์สำหรับผู้ดูแลระบบ สิทธิ (privilege) และข้อมูลประจำตัว Label-Based Access Control (LBAC) โดยที่ผู้ใช้ กลุ่มของผู้ใช้ หรือ role สามารถใช้วิธีในการควบคุมสิทธิ์ดังกล่าว

9.1.3.3 สิทธิระดับผู้ดูแลระบบ (Administrative Authority)

สิทธิระดับผู้ดูแลระบบหมายถึงการกำหนดให้บุคคลมีสิทธิ ในการควบคุมฐานข้อมูลและมีความรับผิดชอบในเรื่องของความถูกต้องของฐานข้อมูล โครงสร้างของสิทธิระดับผู้ดูแลระบบเป็นโครงสร้างแบบลำดับชั้น (hierarchy) ซึ่งผู้ที่มีสิทธิสูงสุดได้แก่ SYSADM และภายใต้ SYSADM จะประกอบด้วยสิทธิรองลงมาในระดับ instance และระดับฐานข้อมูล ในแต่ละระดับของสิทธิระดับผู้ดูแลระบบ จะเป็นการรวมกลุ่มสิทธิ์ (privilege) ไว้ด้วยกัน

สิทธิระดับ instance นั้นจะใช้ครอบคลุมสำหรับฐานข้อมูลทั้งหมดที่อยู่ภายใต้อินสแตนซ์นั้น และมีความสัมพันธ์กันกับกลุ่ม(group) ชื่อของกลุ่ม ที่มีสิทธิ์ระดับ instance จะเก็บอยู่ในแฟ้มของการกำหนดค่าในระดับ instance (database manager configuration file)

DB2 กำหนดสิทธิ์ในระดับ instance ไว้ 4 ระดับด้วยกัน ประกอบด้วย

- SYSADM (System administrator) เป็นสิทธิ์ระดับผู้ดูแลระบบที่ เป็นผู้ที่ควบคุมและดูแลรักษาทรัพยากรที่ถูกสร้างขึ้นโดย instance ผู้ใช้ที่มีสิทธิระดับ SYSADM สามารถทำงานต่อไปนี้ ทำการถ่ายโอนฐานข้อมูล (migrate database) แก้ไข ปรับเปลี่ยน พารามิเตอร์ในระดับ instance และระดับฐานข้อมูล การสำรองฐานข้อมูลและ transaction log รวมถึงการกู้คืนฐานข้อมูล และ ออบเจ็กต์ของฐานข้อมูล เช่นพื้นที่ที่ใช้จัดเก็บตาราง (Table Space) การกำหนดสิทธิ์ (Grant) และการถอนสิทธิ์ของพื้นที่ที่ใช้จัดเก็บตาราง และมีสิทธิทั้งหมดในการจัดการ instance และการทำการ audit ในระดับ instance
- SYSCTRL (System control) เป็นสิทธิ์ที่สามารถควบคุม โอเปอเรชันที่มีผลต่อทรัพยากรต่างๆ ของระบบ ผู้ใช้งานที่มีสิทธินี้สามารถทำการสร้างฐานข้อมูล เปลี่ยนแปลงโครงสร้างของฐานข้อมูล และยังสามารถ Start/Stop instance แต่ผู้มีสิทธินี้จะไม่สามารถเข้าถึงข้อมูลที่จัดเก็บในตารางได้

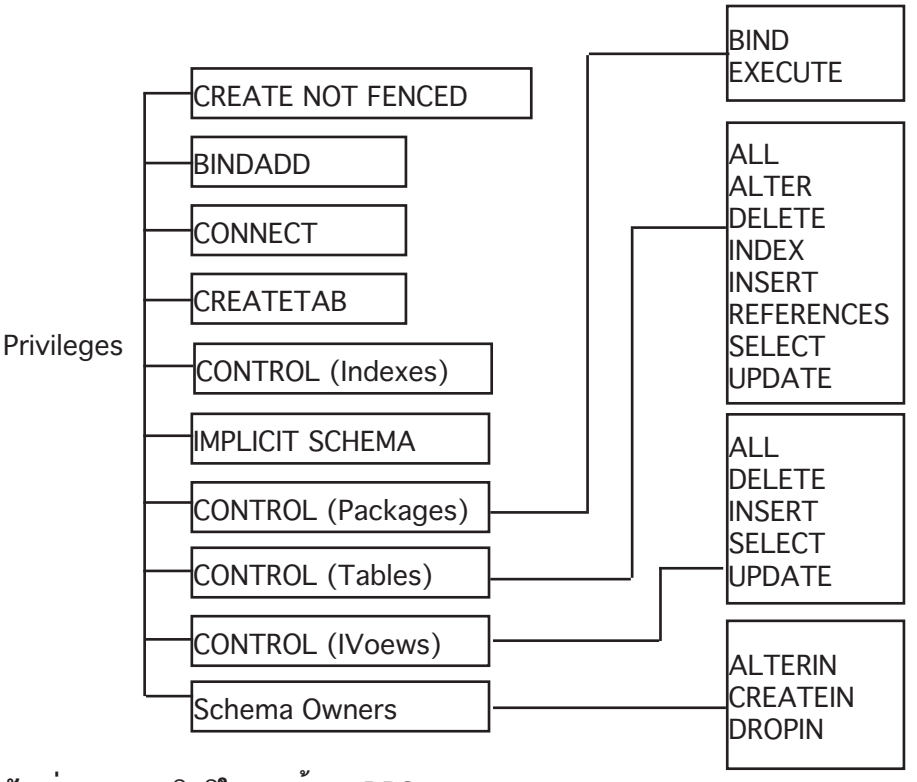
- SYSMAINT (System maintenance) เป็นสิทธิที่สามารถควบคุมโอเปอเรชันที่เกี่ยวการบำรุงรักษาฐานข้อมูลทุกฐานข้อมูลภายใต้ instance นั้น รวมถึงการกำหนดคาพารามิเตอร์ในระดับฐานข้อมูล การมอนิเตอร์หรือการเฝ้าระวังประสิทธิภาพการทำงานของระบบ การสำรองและกู้คืนฐานข้อมูล แต่ผู้มีสิทธินี้จะไม่สามารถเข้าถึงข้อมูลในฐานข้อมูลได้
- SYSMON (System monitor) เป็นสิทธิที่สามารถทำงานได้เฉพาะระดับอินสแตนซ์ ซึ่งสิทธินี้จะเป็นสิทธิที่เกี่ยวข้องกับการตรวจสอบประสิทธิภาพการทำงานของระบบเป็นส่วนใหญ่

สำหรับผู้ใช้งานที่มีสิทธิในระดับฐานข้อมูล (Database authority) ซึ่งจะเป็นสิทธิที่ DB2 กำหนดขึ้นมาเพื่อให้สามารถควบคุมได้เฉพาะฐานข้อมูลนั้นๆ ซึ่งสิทธิเหลานี้ประกอบดวย

- DBADM (Database administrator) เป็นสิทธิในระดับฐานข้อมูลโดยมีสิทธิสูงสุดเฉพาะฐานข้อมูลใดฐานข้อมูลหนึ่งเท่านั้น ผู้มีสิทธินี้ จะมีความสามารถในการ สร้างออบเจ็กต์ของฐานขอมูล สามารถเรียกใช้คำสั่งการจัดการฐานข้อมูล และสามารถเข้าถึงข้อมูลในฐานข้อมูล (เฉพาะ DB2 version ตำกวา 9.7 ใน DB2 9.7 DBADM จะไม่สามารถเข้าถึงข้อมูลได้ถ้าไม่ได้รับสิทธิ์ที่เป็น DATAACCESS) ผู้มีสิทธิ์นี้ยังสามารถสร้างไฟล์ transaction log สามารถสอบถามข้อมูลจากตารางที่เก็บรายละเอียดของฐานข้อมูล (system catalog) รวมถึงสามารถปรับปรุง log history files สามารถทำการ Reorganize ตาราง และดัชนี นอกจากนี้ยังสามารถรวบรวมข้อมูลสถิติต่างๆ ของตารางที่เก็บรายละเอียดของฐานข้อมูล (runstats)
- SECADM (Security administrator) เป็นสิทธิที่สามารถสร้าง ลบ ให้สิทธิหรือถอนสิทธิ โอนย้ายสิทธิสำหรับออบเจ็กต์ต่างๆในฐานข้อมูล (รวมถึง Role และ LBAC labels) แต่ผู้ใช้ที่มีสิทธินี้จะไม่สามารถเข้าถึงข้อมูลได
- CONNECT สิทธิที่ยอมให้ผู้ใช้เชื่อมต่อกับฐานข้อมูล
- BINDADD สิทธิที่ยอมให้ผู้ใช้สร้างแพคเกจใหม่ในฐานข้อมูล
- CREATETAB สิทธิที่ยอมให้ผู้ใช้สามารถสร้างตารางขึ้นใหม่ในฐานข้อมูล
- CREATE_EXTERNAL_ROUTINE เป็นสิทธิที่ยอมให้ผู้ใช้สร้าง procedure เพื่อใช้ในซอฟตแวร์แอพพลิเคชันและใช้งานโดยผู้ใช้คนอื่นๆในฐานข้อมูล
- CREATE_NOT_FENCED_ROUTINE เป็นสิทธิที่ยอมให้ผู้ใช้สร้างฟังก์ชัน (User-Defined Function: UDF) ของผู้ใช้ได้เอง รวมถึงสร้าง procedure ที่เป็น not fenced
- IMPLICIT_SCHEMA เป็นสิทธิที่ยอมให้ผู้ใช้สร้าง schemaโดยการสร้างเป็นออบเจ็กต์ผ่านคำสั่ง CREATE SCHEMA พร้อมทั้งระบุชื่อ schema ที่ยังไม่เคยถูกสร้างมากอน ในกรณีนี้ SYSIBM จะกลายมาเป็นเจ้าของออบเจ็กต์ และ PUBLIC จะได้รับสิทธิในการสร้างออบเจ็กต์ภายใต้ schema นี้

9.1.3.4 สิทธิ (Priviledges)

สิทธิ (Priviledges) เป็นการกำหนด ความสามารถให้แก่ผู้ใช้ กลุ่ม หรือRole ซึ่งเป็นการยอมให้ผู้ใช้งานฐานข้อมูลสามารถกระทำกิจกรรมต่างๆที่เกี่ยวข้องกับออบเจ็กต์ของระบบฐานข้อมูล จากรูป 9.2 แสดงให้เห็นถึง สิทธิต่างๆที่อยู่ใน DB2 [9.1]



รูป 9.2 ตัวอย่างรายการสิทธิในฐานข้อมูล DB2

สิทธิเป็นการกำหนด ความสามารถของผู้ใช้งานในการดำเนินการกับออบเจ็กต์ของฐานข้อมูล ซึ่งสามารถกำหนดให้เป็นรายบุคคล รายกลุ่ม ตามบทบาท (Role) หรือเป็นแบบสาธารณะ (PUBLIC) สำหรับ PUBLIC นั้นถือว่าเป็นกลุ่มที่มีความพิเศษ เราสามารถกำหนดสิทธิ์ หรือ ถอนสิทธิ์จาก PUBLIC ได้ ซึ่งหลังจากที่ฐานข้อมูลถูกสร้างขึ้นมา ฐานข้อมูลนั้นจะกำหนดสิทธิ์ต่อไปนี้ให้กับ Public โดย default คือ CONNECT, CREATETAB, BINDADD, IMPLICIT_SCHEMA, SELECT, UPDATE, EXECUTE, USE

และผู้ใช้หรือกลุ่มที่ได้รับสิทธิ CONTROL ยังสามารถกำหนดสิทธิ์ให้แก่อื่นผู้ใช้คนอื่นๆหรือกลุ่มได้อีกด้วย

9.1.3.5 การควบคุมสิทธิแบบ Label Based Access Control

Label Based Access Control (LBAC) เป็นการควบคุมสิทธิแบบยืดหยุ่นสำหรับสิทธิที่มีการบังคับ mandatory access control (MAC) สำหรับ LBAC นั้นควบคุมในระดับแถว (row) และคอลัมน์ (column) และสามารถทำงานรวมกันกับการควบคุมการเข้าถึงแบบ discretionary access control (DAC) เราสามารถใช้ LBAC เพื่อตอบสนองความต้องการด้านความปลอดภัยในแต่ละสภาพแวดล้อม การตั้งค่าคอนฟิกทั้งหมดนั้นจะสามารถกระทำได้โดยผู้ดูแลระบบความปลอดภัย (SECADM) โดยวิธีการสร้างเป็นนโยบาย ด้านความปลอดภัย (security policy) ซึ่งจะประกอบด้วยคำอธิบายและเงื่อนไขว่าจะยอมให้ผู้ใช้คนไหนสามารถเข้าถึงและใช้งานข้อมูลใดได้บ้าง

หลังจากที่มีการกำหนดเป็นนโยบายด้านความปลอดภัยแล้ว ผู้ดูแลระบบด้านความปลอดภัยจะสามารถสร้างออบเจ็กต์ที่เรียกว่าป้ายความปลอดภัย (security labels) ซึ่งเป็นส่วนหนึ่งของนโยบายด้านความปลอดภัยที่กำหนดขึ้นมา และใช้ป้ายความปลอดภัย ในการเชื่อมโยงกับแต่ละแถวและคอลัมน์ในตารางเพื่อเป็นการปกป้องข้อมูล โดยเราจะเรียกข้อมูลที่ถูกปกป้องนี้ว่า protected data ผู้ดูแลระบบด้านความปลอดภัยจะสามารถกำหนดให้ผู้ใช้เข้าถึงข้อมูลที่มีการปกป้องนี้โดยวิธีการ Grant ป้ายความปลอดภัยให้กับผู้ใช้นั้น และเมื่อผู้ใช้ต้องการเรียกใช้งานหรือเข้าถึงข้อมูลชุดนี้ ป้ายความปลอดภัย ของผู้ใช้จะถูกเปรียบเทียบกับป้ายความปลอดภัยในระดับแถว หรือระดับคอลัมน์ หรือทั้งสองอย่าง ซึ่งทำให้ DB2 ทราบว่าผู้ใช้ดังกล่าวสามารถเข้าถึงข้อมูล ในระดับแถว หรือระดับคอลัมน์ หรือทั้งสองอย่าง ได้หรือไม่ ซึ่ง ป้ายความปลอดภัย นี้จะเป็นข้อมูลประจำตัวของ LBAC และถูกเก็บไว้ใน ตารางที่เก็บรายละเอียดของฐานข้อมูล (system catalog) ของฐานข้อมูล

ประโยชน์หลักของการใช้ LBAC ในการป้องกันข้อมูลที่สำคัญคือผู้ใช้ที่มีสิทธิ SYSDBA, DBADM และ SECADM จะไม่สามารถเข้าถึงข้อมูลชุดนี้ได้

9.1.3.6 บทบาท (Roles)

ในสถานการณ์ที่เราอยากให้ผู้ใช้หลายคนมีสิทธิในระดับเดียวกัน และมีสิทธิในการทำงานที่เหมือนกัน ซึ่งการให้สิทธิ์กับผู้ใช้ทีละคนนั้นอาจจะเป็นเรื่องที่ยุ่งยาก ในระบบฐานข้อมูล เราสามารถจัดการกับกลุ่มผู้ใช้งานที่ต้องการให้มีสิทธิที่เหมือนกัน โดยการกำหนดเป็นบทบาท (Role)

บทบาท (Role) เป็นออบเจ็กต์ในฐานข้อมูล ซึ่งสามารถรวมสิทธิ (privilege) ต่างๆเข้าด้วยกัน และบทบาทจะถูกกำหนดให้แก่ผู้ใช้งานหนึ่งคน กลุ่มของผู้ใช้งาน หรือกลุ่มแบบสาธารณะ (public) โดยคำสั่ง GRANT ตัวอย่างเช่น เราสามารถกำหนดบทบาทนักพัฒนา (Developer role) และอนุญาตให้บทบาทนี้สามารถแทรก (insert) ปรับปรุง (update) และลบข้อมูลในตาราง เมื่อกำหนดบทบาทนี้ให้แก่ผู้ใช้งาน สิทธิต่างๆที่มีอยู่ในบทบาทนี้ จะถูกเพิ่ม ให้กับผู้ใช้งานนั้นโดยรวมกันกับสิทธิ์ที่เขามีอยู่แล้ว

บทบาท สามารถใช้ในการควบคุมสิทธิ์ ให้เหมาะสมกับโครงสร้างขององค์กรได้ เนื่องจากแต่ละองค์กรสามารถสร้างบทบาทที่ เทียบเคียงกันกับโครงสร้างภายในองค์กร ในกรณีนี้เราสามารถกำหนดสิทธิ์ให้ผู้ใช้ผ่านบทบาท โดยกำหนดสิทธิ์ตามหน้าที่รับผิดชอบของผู้ใช้ในองค์กร เมื่อผู้ใช้เปลี่ยนแปลงหน้าที่ความรับผิดชอบในองค์กร การจัดการกับสิทธิของผู้ใช้ผ่านทางบทบาทสามารถกระทำได้โดยง่ายโดยวิธีการเพิ่มสิทธิ หรือ เพิกถอนสิทธิโดยตรงไปที่บทบาทโดยที่ไม่ต้องจัดการสิทธิของผู้ใช้เป็นรายบุคคล บทบาท จะถูกบริหารจัดการภายในฐานข้อมูล

9.1.3.7 การเชื่อถือในบริบท (Trusted Contexts)

สำหรับสถาปัตยกรรมแอพพลิเคชันแบบ 3 ระดับ (three-tiered application model) ซึ่งประกอบด้วยเครื่องแม่ข่ายเว็บ เครื่องแม่ข่ายแอพพลิเคชัน และเครื่องแม่ข่ายฐานข้อมูล แอพพลิเคชันที่จัดอยู่ใน ระดับที่ 2 หรือระดับกลางนั้นจะรับผิดชอบในการตรวจสอบสิทธิ์ผู้ใช้งานที่เรียกใช้โปรแกรมประยุกต์จากไคลเอ็นต์ และทำการโต้ตอบกับเครื่องแม่ข่ายฐานข้อมูล ในการเชื่อมต่อระหว่างแอพพลิเคชันกับฐานข้อมูล เครื่องแม่ข่ายแอพพลิเคชันจะต้องระบุชื่อผู้ใช้ (application's server authorization ID) เพื่อใช้ในการยืนยันตัวตนกับฐานข้อมูล โดยผู้ใช้ดังกล่าวจำเป็นต้องมีสิทธิในการใช้งานฐานข้อมูล ซึ่งสิทธิ์ดังกล่าวจะต้องสัมพันธ์กันกับชื่อผู้ใช้ (end users) ที่ถูกระบุจากไคลเอ็นต์ เพื่อใช้ในการยืนยันตัวตนกับเครื่องแม่ข่ายแอพพลิเคชัน เพื่อให้สามารถดำเนินการใดๆกับระบบฐานข้อมูลได้ ในขณะที่สถาปัตยกรรมแอพพลิเคชันแบบ 3 ระดับนั้นมีประโยชน์มีการทำงานระหว่าง เครื่องแม่ข่ายแอพพลิเคชัน กับ เครื่องแม่ข่ายฐานข้อมูล โดยใช้ชื่อผู้ใช้ที่ระบุจากเครื่องแม่ข่ายแอพพลิเคชัน แต่อาจจะพบปัญหาในเรื่องของความปลอดภัย เช่น ไม่สามารถตรวจสอบว่าผู้ใช้ที่ถูกระบุจากไคลเอ็นต์ (end users) ใดที่สร้างการสอบถามข้อมูลที่มีความบกพร่อง เพราะว่า ฐานข้อมูลจะรู้จักเฉพาะผู้ใช้งานที่ระบุจากเครื่องแม่ข่ายแอพพลิเคชัน (application's server authorization ID) เท่านั้นแต่จะไม่ทราบว่าแท้ที่จริงแล้วผู้ใช้งานดังกล่าว คือ ผู้ใช้ที่ถูกระบุจากไคลเอ็นต์ (end users) คนใด

Trusted Contexts เป็นการสร้างความสัมพันธ์ที่เชื่อถือได้ (trusted relationship) ระหว่าง DB2 กับองค์ประกอบภายนอกเช่น เครื่องแม่ข่ายแอพพลิเคชัน หรือกับเครื่องแม่ข่ายฐานข้อมูล DB2 อื่นๆ โดยหลักการทำงานนั้นจะมีการสร้างความสัมพันธ์โดยใช้แอตทริบิวต์ดังต่อไปนี้ การให้สิทธิโดยระบบ (system authorization) หมายเลขไอพี (IP address) หรือชื่อโดเมนของเครื่องแม่ข่าย และการเข้ารหัสของข้อมูลที่ส่งระหว่างกัน (data stream encryption)

หลังจากที่มีการเชื่อมต่อกับฐานข้อมูล แอตทริบิวต์ที่ใช้ในการเชื่อมต่อนั้นจะถูกเปรียบเทียบกับแอตทริบิวต์ที่ถูกระบุไว้ใน Trusted Contexts ที่เราสร้างขึ้นในฐานข้อมูล ถ้าแอตทริบิวต์ที่เปรียบเทียบเหมือนกัน การเชื่อมต่อ ระหว่างฐานข้อมูลกับแอพพลิเคชันดังกล่าว จะถือว่าเป็นการเชื่อมต่อแบบมีความน่าเชื่อถือ (trusted connection)

เราสามารถเลือกรูปแบบของการเชื่อมต่อแบบมีความน่าเชื่อถือได้ 2 ลักษณะคือ explicit และ implicit สำหรับแบบ explicit นั้นจำเป็นต้องมีการร้องขอในกรณีที่มีการเชื่อมต่อ และจะอนุญาตให้ผู้ร้องขอสามารถทำการเปลี่ยนชื่อผู้ใช้ในปัจจุบัน (ชื่อผู้ใช้ที่ถูกระบุมาจากเครื่องแม่ข่ายแอพพลิเคชัน เพื่อใช้ยืนยันการมีตัวตนกับฐานข้อมูล) ที่ใช้ในการเชื่อมต่อไปเป็น ชื่อผู้ใช้อื่น โดยที่จะทำการยืนยันตัวตนใหม่ หรือไม่ก็ได้ รวมทั้งสามารถได้รับสิทธิ์ (privilege) เพิ่มเติมจาก สิทธิ์ที่เราระบุไว้ใน trusted context สำหรับการเชื่อมต่อแบบ implicit นั้นจะเป็นการเชื่อมต่อโดยที่ไม่ได้มีการร้องขอ ซึ่งจะอนุญาตให้ได้รับสิทธิ์ (privilege) เพิ่มเติมจาก สิทธิ์ที่เราระบุไว้ใน trusted context เท่านั้น

เมื่อต้องการกำหนดการเชื่อมต่อแบบเชื่อถือได้ต้องมีกำหนดแอตทริบิวต์ต่อไปนี้:

- รหัสการอนุญาตโดยระบบ (A system authorization ID) ซึ่งจะ เป็นชื่อผู้ใช้ ที่ถูกระบุในกรณีของการเชื่อมต่อ
- รายการของที่อยู่ IP (A list of IP addresses) คือเบอร์ไอพีของเครื่องที่จะทำการเชื่อมต่อเข้ามาเพื่อให้เกิดความน่าเชื่อถือ ซึ่งโดยทั่วไป คือ เครื่องแม่ข่ายแอพพลิเคชัน
- การเข้ารหัสของข้อมูล (A data stream encryption) แสดงให้เห็นถึงระดับของการเข้ารหัสที่ต้องใช้ในการเชื่อมต่อเข้ามา

9.1.4 วิว (Views)

วิวเป็นองค์ประกอบที่สำคัญของกลไกการรักษาความปลอดภัยโดยระบบฐานข้อมูล โดยที่วิวเป็นการสร้างมุมมองใหม่ของฐานข้อมูลที่อนุญาตให้ผู้ใช้เห็นข้อมูลในมุมมองต่างๆ ที่เราต้องการให้เห็น และเรายังสามารถซ่อนข้อมูลใดๆ ที่ไม่ต้องการให้ผู้ใช้ เข้าถึงผ่านวิวได้อีกด้วย

วิวเป็นการสร้างมุมมอง เพื่อให้ได้ผลลัพธ์เป็นแบบไดนามิก ที่ถูกสร้างขึ้นมาจากโอเปอเรชั่นต่างๆ เช่น union โดยเกี่ยวข้องกับตารางปกติ (base table) หนึ่งตารางหรือมากกว่า เพื่อให้เกิดเป็น ตารางใหม่ วิว จะแสดงผลลัพธ์ที่เป็นข้อมูลปัจจุบันที่ได้จากตารางปกติ ที่มันอ้างอิงถึง

ประโยชน์ของวิวในมุมมองด้านความปลอดภัยคือวิวถูกสร้างขึ้นมาเพื่อใช้สำหรับการนำเสนอข้อมูลในแก่ผู้ใช้ในมุมมองต่างๆ ตามที่เราต้องการ โดยที่วิวจะไม่แสดงข้อมูลที่เป็นความลับที่เราต้องการปกปิดไว้ เราสามารถกำหนดสิทธิให้แก่ผู้ใช้ในการทำงานกับ วิว โดยที่ไม่จำเป็นต้องให้สิทธิ์ในการเข้าถึงตารางปกติ ที่วิวอ้างอิงถึง

9.1.5 การควบคุมความสมบูรณ์ (Integrity Control)

วัตถุประสงค์ของการควบคุมความสมบูรณ์ในระบบฐานข้อมูลคือการปกป้องไม่ให้ข้อมูลถูกแก้ไขเปลี่ยนแปลงโดยผู้ที่ไม่มีสิทธิ์ โดยวิธีการบังคับว่าข้อมูลที่จะนำเข้าสามารถมีค่าที่เป็นไปได้อะไรบาง หรือ เราสามารถทำโอเปอเรชั่นใดกับข้อมูลได้บาง การควบคุมความสมบูรณ์ ยังสามารถกระทำได้ โดยการใช้ trigger ในการตรวจจับการทำงานของ Procedure เช่นการเก็บประวัติการเปลี่ยนแปลงข้อมูลไว้ใน log file การควบคุมความสมบูรณ์สามารถทำได้ในหลายรูปแบบ

รูปแบบแรกของการควบคุมความสมบูรณ์ที่เราจะกล่าวถึง ได้แก่โดเมน (domain) ซึ่งโดเมน สามารถมองคล้ายกับการสร้าง ชนิดของข้อมูลแบบกำหนดขึ้นเอง (User-defined data type) หลังจากสร้าง domain แล้ว เราสามารถกำหนดให้เป็นชนิดของข้อมูล สำหรับคอลัมน์ได้ ทำให้เรามั่นใจได้ว่าข้อมูลที่ถูกบันทึกในคอลัมน์นั้นเป็นไปตามชนิดข้อมูลที่เรากำหนด นอกจากนี้ในการสราง domain ยังสามารถใช้ constraint เช่น Check constraint เพื่อใช้ควบคุมค่าที่บันทึกนั้นเป็นไปตามเงื่อนไขที่เรากำหนด

การใช้ตัวควบคุม Assertions ซึ่งเป็น constraint รูปแบบหนึ่งที่ใช้ในการบังคับการทำงานของฐานข้อมูลให้เป็นไปตามเงื่อนไขที่ต้องการ ซึ่งจะทำการตรวจสอบแบบอัตโนมัติโดยฐานข้อมูล เมื่อมีการทำงานกับตาราง หรือ คอลัมน์ ที่มีกำหนด Assertion อยู่ ซึ่ง จะมีการสร้างข้อความที่แสดงข้อผิดพลาดในกรณีที่การบันทึกข้อมูลไม่เป็นไปตามเงื่อนไขที่ระบุใน Assertions

เพื่อการรักษาความปลอดภัย เรายังสามารถใช้ Trigger ในการควบคุมการทำงานของฐานข้อมูลให้เป็นไปตามเงื่อนไขที่กำหนด Trigger ประกอบด้วย code ที่เราพัฒนาขึ้นมา และจัดเก็บอยู่ในฐานข้อมูล ซึ่งการทำงานของ Trigger นั้นจะเป็นการทำงานเพื่อตอบสนองต่อคำสั่ง INSERT, UPDATE หรือ DELETE โดย Trigger จะประกอบด้วยเหตุการณ์ที่ทำให้ทริกเกอร์ทำงาน (event) เงื่อนไข (condition) และกิจกรรมที่ต้องการทำเมื่อมีเหตุการณ์เกิดขึ้น (action) ตัวอย่างเช่น การกำหนดให้ Trigger ทำงานเมื่อมีการบันทึกข้อมูล หรือการเปลี่ยนแปลงข้อมูลในตารางที่สำคัญ โดยกำหนดให้ทริกเกอร์ทำการบันทึกรายละเอียดของผู้ที่แก้ไขข้อมูล วันและเวลาที่แก้ไข รวมถึงรายละเอียดของธุรกรรมที่ทำกับข้อมูล

9.1.6 การเข้ารหัสข้อมูล

ข้อมูลส่วนบุคคลและข้อมูลที่มีความสำคัญที่ถูกจัดเก็บในตารางของฐานข้อมูล หรือ ข้อมูลสำคัญที่ถูกส่งผ่านทางเครือข่าย เช่น ชื่อผู้ใช้และรหัสผ่าน ซึ่งข้อมูลเหล่านี้เป็นข้อมูลที่มีความสำคัญและควรได้รับการป้องกัน การเข้ารหัสข้อมูลเป็นกระบวนการเข้ารหัสโดยเทคนิควิธีทางอัลกอริธึมเฉพาะ ซึ่งจะทำให้โปรแกรมอื่นๆ ไม่สามารถอ่านและเข้าใจได้หากไม่มีคีย์การถอดรหัสลับ โดยปกติการเข้ารหัสข้อมูลจะปกป้องข้อมูลที่ส่งผ่านทางเครือข่ายการสื่อสาร ซึ่งเทคนิคการเข้ารหัสข้อมูลนั้นมีอยู่หลายวิธีด้วยกัน ระบบที่มีฟังก์ชันในการเข้ารหัสนั้นจำเป็นต้องมีวิธีการในการถอดรหัสด้วยเทคนิควิธีทางอัลกอริธึมเฉพาะ ซึ่งวิธีการถอดรหัสดังกล่าวควรจะมีการรักษาความปลอดภัยที่เหมาะสมด้วยเช่นกัน

9.2 นโยบายและกระบวนการในการรักษาความปลอดภัย

การกำหนดนโยบายการบริหารและการสร้างกระบวนการสำหรับมาตรการด้านความปลอดภัยอย่างมีประสิทธิภาพนั้นเป็นเรื่องที่จำเป็นสำหรับทุกองค์กร ซึ่งนโยบายการรักษาความปลอดภัยสามารถแบ่งออกได้เป็น 2 ลักษณะคือ ขั้นตอนการควบคุมการปฏิบัติงานของบุคลากรและการควบคุมการเข้าถึงทางกายภาพ

9.2.1 การควบคุมด้านบุคลากร

ความเสี่ยงมากที่สุดในเรื่องของความปลอดภัยขององค์กรนั้นมักจะมาจากภายในองค์กรมากกว่า

ภายนอก ดังนั้นการจัดหามาตรการสำหรับการควบคุมด้านบุคลากรนั้นเป็นเรื่องที่สำคัญ ซึ่งนั่นหมายถึงกระบวนการอนุญาตและการบังคับใช้มาตรการการรักษาความปลอดภัย เริ่มต้นจากกระบวนการสรรหาบุคลากร การตรวจสอบประวัติการทำงาน และความรู้ความสามารถ พนักงานควรได้รับการอบรมเกี่ยวกับความเสี่ยงที่อาจจะเกิดขึ้นในด้านความปลอดภัยที่เกี่ยวข้องกับงานของตน เพื่อเป็นการกระตุ้นให้เห็นความสำคัญและตระหนักถึงผลกระทบที่อาจจะเกิดขึ้น และเพื่อให้ปฏิบัติตามมาตรการด้านความปลอดภัย และหากมีการเปลี่ยนแปลงในเรื่องของสถานะภาพของบุคคลากร เช่น การลาออก ก็จะต้องมีขั้นตอนในการถอดถอนสิทธิ และมีการแจ้งเกี่ยวกับการเปลี่ยนแปลงดังกล่าวให้แก่พนักงานคนอื่นๆได้รับทราบ

9.2.2 การควบคุมด้านกายภาพ

การควบคุมการเข้าถึงฐานข้อมูลทางด้านกายภาพเป็นเรื่องที่สำคัญ และเป็นเรื่องที่เกี่ยวข้องกับการจำกัดการเข้าถึงพื้นที่เฉพาะ ภายในอาคาร ตัวอย่างเช่นการใช้สมาร์ทการ์ดเพื่อเข้าถึงพื้นที่ที่ต้องการความปลอดภัยเป็นพิเศษ ในกรณีนี้ ทุกครั้งที่มีการเข้าใช้งาน ระบบจำเป็นต้องมีการบันทึกประวัติการใช้หรือการเข้าถึงในฐานข้อมูล หรือเมื่อมีแขกหรือบุคคลภายนอกที่เข้ามาในองค์กรก็ควรจะมีการแขวนป้ายเพื่อบ่งบอกถึงสถานะว่าเป็นบุคคลภายนอก และมีบุคคลากรในองค์กรร่วมทำงานด้วย

อุปกรณ์สำคัญๆ เช่นเครื่องแม่ข่าย สามารถควบคุมได้โดยการจัดวางในพื้นที่ที่มีความปลอดภัยสูง อุปกรณ์อื่นๆ สามารถถูกล็อกแบบตั้งโต๊ะหรือมีติดตั้งสัญญาณเตือนภัย สื่อที่ใช้ในการสำรองข้อมูล เช่น ม้วนเทปควรเก็บไว้ในตู้เซฟที่สามารถป้องกันไฟได้ หรือเก็บอยู่ในต่างสถานที่ที่มีความปลอดภัย จะต้องมีแผนและขั้นตอนพร้อมทั้งตารางเวลาสำหรับการใช้ เคลื่อนย้าย หรือ ยกเลิก สื่อที่ใช้ในการสำรองข้อมูล รวมถึงการทำฉลาก การทำดัชนีของ สื่อที่ใช้ในการสำรองข้อมูลที่เก็บไว้เพื่อให้ง่ายต่อการสืบค้น

บริษัทสมัยใหม่มักจะใช้สถานที่ในการทำงานในรูปแบบ Mobile office กล่าวคือ พนักงานจะไม่มีโต๊ะทำงานเป็นของตนเองโดยเฉพาะ ทุกอย่างจะใช้งานร่วมกันสำหรับพนักงานทุกคนในกรณีนี้ พนักงานส่วนใหญ่จะมีการใช้เครื่องแล็ปท็อป ซึ่งมีความเสี่ยงต่อการถูกโจรกรรมสูง ทำให้ข้อมูลที่จัดเก็บในเครื่องแล็ปท็อปมีความเสี่ยงสูงที่จะถูกโจรกรรม ดังนั้นจึงได้มีเทคโนโลยีและผลิตภัณฑ์ที่สามารถเข้ารหัสลับและเครื่องมืออีกหลายอย่างเพื่อปกป้องข้อมูล เช่นอุปกรณ์กันขโมยโดยการล็อคสายเคเบิล หรือเครื่องมือสำหรับติดตามเครื่องคอมพิวเตอร์ที่ถูกการโจรกรรม (GPS) เพื่อเป็นการป้องกันการขโมย

9.3 สรุป

บทนี้ครอบคลุมในประเด็นความปลอดภัยทั่วไปของฐานข้อมูล ซึ่งในช่วงเริ่มต้นของบทได้มีการอธิบายถึงความจำเป็นในการปกป้องข้อมูลและสภาพแวดล้อมและการคุกคามในรูปแบบต่างๆที่สามารถส่งผลกระทบต่อข้อมูลในฐานข้อมูลและองค์กรโดยรวม

การตรวจสอบถึงความปลอดภัยอันดับแรกคือการควบคุมการเข้าถึง (access control) ซึ่งประกอบไปด้วย การควบคุมในเชิงคุณภาพของการรักษาความปลอดภัย (Discretionary control) และการควบคุมแบบบังคับที่จะต้องทำ (Mandatory control) ซึ่งเรานำเสนอกรณีต่างๆ ของการควบคุมการเข้าถึงที่สามารถนำมาใช้ใน DB2 เช่นกลไกการ ยืนยันการมีตัวตน (authentication) และการอนุญาต (authorization) สิทธิพิเศษ (privilege) บทบาท (role) ป้ายชื่อที่ใช้ควบคุมการเข้าถึง (LBAC) และบริบทที่เชื่อถือได้ (trust context) ซึ่งวิธีง่ายและยืดหยุ่นคือวิธีการใช้มุมมอง (view) เพื่อซ่อนข้อมูลส่วนใหญ่ของฐานขอมูลจากผู้ใช้

การควบคุมความสมบูรณ์ในการป้องกันข้อมูลจากการเข้าถึงที่ไม่ได้รับอนุญาต โดยการจำกัดค่าที่เป็นไปได้ว่ามีอะไรบ้าง หรือ เราสามารถทำโอเปอเรชั่นใดกับข้อมูลได้บ้าง โดยการใช้โดเมน การตรวจสอบ Check constraint, Assertions และ Trigger

ข้อมูลสำคัญที่ถูกส่งผ่านเครือข่ายและข้อมูลส่วนบุคคล สามารถป้องกันได้ โดยการเข้ารหัสข้อมูล ซึ่งมาตรการที่กล่าวถึงในบทนี้ไม่สามารถหยุดการเข้าถึงที่ไม่ประสงค์ดี ได้ทั้งหมด เพื่อให้บรรลุวัตถุประสงค์องค์กรจำเป็นที่จะต้องกำหนดนโยบายบริหารและกระบวนการที่กับความปลอดภัย และบังคับใช้มาตรการเหล่านี้ ให้มีประสิทธิภาพ รวมถึงจะต้องมีนโยบายการรักษาความปลอดภัยที่ควบคุมบุคลากรและควบคุมการเข้าถึงทางกายภาพ

9.4 แบบฝึกหัด

ให้ทำการอ่านทบทวนบทที่ 10 “ความปลอดภัยของฐานข้อมูล” ในหนังสือ eBook ที่แจกฟรี Getting started with DB2 Express-C และ ให้ทำการสร้างผู้ใช้งาน 2 คนในระบบปฏิบัติการของคุณ และให้ตอบคำถามต่อไปนี้

- ผู้ใช้ทั้งสองคนสามารถสร้างฐานข้อมูลได้ หรือไม่? ทำไม
- ผู้ใช้ทั้งสองคนสามารถเชื่อมต่อไปยังฐานข้อมูลหรือไม่? ทำไม
- ผู้ใช้ทั้งสองคนสามารถเลือกวิว SYSCAT.TABLES ได้หรือไม่? ทำไม

9.5 คำถามท้ายบท

1. เหตุใดจึงจำเป็นต้องปกป้องข้อมูลในฐานข้อมูล
2. การป้องกันเฉพาะข้อมูลที่จัดเก็บในฐานข้อมูล ถือว่าเพียงพอหรือไม่
3. ภัยคุกคามใดบ้างที่ต้องให้ความสนใจและจัดทำเป็นแผนการรักษาความปลอดภัย
4. อธิบายคำจำกัดความของการควบคุมในเชิงคุณภาพของการรักษาความปลอดภัย (Discretionary control)
5. อธิบายลักษณะของ การอนุญาต (Authorization) ในฐานข้อมูล DB2
6. สิทธิ (Privileges) คืออะไร
7. บริบทที่เชื่อถือ (Trusted contexts) ใน DB2 มีหลักการทำงานอย่างไร
8. วิว คืออะไร และทำไมจึงถือว่าเป็นส่วนประกอบสำคัญของกลไกรักษาความปลอดภัย
9. อธิบาย การควบคุมความสมบูรณ์ (Integrity control) ว่าสามารถควบคุมความถูกต้องของข้อมูลได้อย่างไร
10. นโยบายการรักษาความปลอดภัยและกระบวนการที่ใช้มากที่สุดคืออะไร

10

บทที่ 10 - แนวโน้มเทคโนโลยีและฐานข้อมูล

จากการสำรวจของ ไอบีเอ็ม ในปี ค.ศ. 2010 เพื่อศึกษาถึงแนวโน้มทางด้านเทคโนโลยีจนถึงปี ค.ศ. 2015 พบว่าจะมีแนวโน้มของเทคโนโลยีที่หลากหลายเกิดขึ้น และได้จัดทำการสำรวจบุคลากรทางด้านไอทีมากกว่า 2,000 คน ทั่วโลก โดยมีความเชี่ยวชาญในด้านต่าง ๆ เช่น นักทดสอบระบบคอมพิวเตอร์ ผู้ดูแลระบบคอมพิวเตอร์ และผู้ดูแลระบบเครือข่าย นักวิเคราะห์ออกแบบสถาปัตยกรรมระบบคอมพิวเตอร์ และนักพัฒนาโปรแกรมระดับองค์กร และนักพัฒนาเว็บไซต์ ซึ่งได้พบสองเทคโนโลยีที่มีแนวโน้มว่าจะเป็นเทคโนโลยีในอนาคตจากการสำรวจครั้งนี้

- การประมวลผลแบบ Cloud Computing จะเข้ามาแทนที่การประมวลผลของคอมพิวเตอร์แบบดั้งเดิม และจะเป็นทางเลือกหลักสำหรับการจัดการทรัพยากรทางด้านไอทีขององค์กร
- การพัฒนาโปรแกรมบนโทรศัพท์เคลื่อนที่ เช่น iPhone และ Android, และ Tablet PCs เช่น iPad และ PlayBook, ซึ่งจะมีความก้าวล้ำกว่าการพัฒนาโปรแกรมในรูปแบบอื่น

ในบทนี้จะอธิบายเกี่ยวกับ Cloud Computing, การพัฒนาโปรแกรมบนโทรศัพท์เคลื่อนที่ และแนวโน้มเทคโนโลยีอื่น ๆ และยังอธิบายถึงบทบาทของฐานข้อมูลบนเทคโนโลยีเหล่านั้น นอกจากนี้ในบทนี้ยังแสดงให้เห็นถึงกรณีตัวอย่างในชีวิตจริงที่มีการใช้เทคโนโลยีเหล่านี้ โดยหลังจากที่สิ้นสุดบทนี้คุณจะมีความเข้าใจในเทคโนโลยีเหล่านี้ และนำไปใช้ประโยชน์ได้ในการพัฒนาของคุณในอนาคต

10.1 การประมวลผลแบบกลุ่มเมฆ (Cloud computing) คืออะไร?

เทคโนโลยีการประมวลผลแบบกลุ่มเมฆ (Cloud computing) ไม่ใช่เทคโนโลยีใหม่ แต่จะเป็นรูปแบบใหม่ในการส่งมอบทรัพยากรทางด้านไอทีที่ทำให้ผู้ใช้งานมีความรู้สึกว่าสามารถเข้าใช้งานทรัพยากรที่มีอยู่อย่างไม่จำกัด ด้วย Cloud คุณสามารถเช่าใช้การประมวลผลที่มีประสิทธิภาพโดยไม่จำเป็นที่จะต้องซื้อเครื่องแม่ข่าย เพียงแค่จ่ายเงินสำหรับการใช้งานเท่านั้น รูปแบบการทำงานแบบใหม่นี้มักจะถูกเทียบกับการที่ผู้คนใช้จ่ายสำหรับสาธารณูปโภคทั่วไปเช่นการจ่ายเงินสำหรับน้ำประปาหรือไฟฟ้าตามปริมาณการใช้งาน

Cloud computing ได้เปลี่ยนแปลงวิธีการเข้าถึงทรัพยากร อนุญาตให้เกือบทุก ๆ คน, วิสาหกิจขนาดเล็ก, วิสาหกิจขนาดใหญ่สามารถจัดการโครงการที่ไม่อาจสามารถทำได้มาก่อน

ตาราง 10.1 เป็นตารางเปรียบเทียบระหว่างรูปแบบการทำงานของไอทีแบบดั้งเดิม กับรูปแบบการทำงานของ Cloud computing

รูปแบบการทำงานของไอทีแบบดั้งเดิม	รูปแบบการทำงานของ Cloud computing
ต้องมีงบประมาณในการลงทุน	เป็นส่วนหนึ่งของค่าใช้จ่ายในการดำเนินงาน
มีการลงทุนขนาดใหญ่	เริ่มต้นที่ 2 เซ็นต์ต่อชั่วโมง
ต้องเตรียมแผนรองรับปริมาณการใช้งานสูงสุด	สามารถปรับเปลี่ยนตามการใช้งาน
ใช้ระยะเวลา 120 วันในการเตรียมทรัพยากรก่อนที่จะเริ่มโครงการได้	ใช้เวลาน้อยกว่า 2 ชั่วโมงในการเตรียมทรัพยากรเพื่อให้ระบบทำงานได้

ตารางที่ 10.1 - เปรียบเทียบการทำงานของไอทีแบบดั้งเดิม กับการทำงานของ Cloud computing

ในขณะที่รูปแบบการทำงานของไอทีแบบดั้งเดิมต้องมีการตั้งงบประมาณเพื่อซื้อฮาร์ดแวร์ และลงทุนเป็นเงินจำนวนมากตั้งแต่ต้น แต่กับ Cloud computing ค่าใช้จ่ายที่เกิดขึ้นจะเป็นค่าใช้จ่ายในการดำเนินงานเพื่อการทำธุรกิจเท่านั้น เป็นการจ่ายเพียงเล็กน้อยตอชั่วโมงตามการใช้งานของทรัพยากร

ในรูปแบบการทำงานของไอทีแบบเดิม ต้องการงบประมาณ ฮาร์ดแวร์ และซอฟต์แวร์ ติดตั้งในห้องปฏิบัติการ หรือศูนย์ข้อมูล และทำการกำหนดค่าให้ซอฟต์แวร์เป็นเวลานาน โดยเฉลี่ยจะใช้เวลา 120 วัน หรือมากกว่านั้นระบบถึงจะสามารถใช้งานได้ แต่สำหรับ Cloud computing สามารถกำหนดค่าของระบบ และทำงานได้ภายใน 2 ชั่วโมง

นอกจากนี้องค์กรต้องวางแผนรองรับปริมาณการใช้งานสูงสุด ตัวอย่างเช่น ต้องวางแผนรองรับปริมาณการใช้งานในอนาคตอาจต้องการเครื่องแม่ขาย 3 เครื่องสำหรับการประมวลผลของ 25 วันในหนึ่งเดือน แต่ต้องการเครื่องแม่ขายเพิ่มอีก 2 เครื่องเพื่อจัดการการประมวลผลสำหรับ 5 วันที่เหลือในหนึ่งเดือน ดังนั้นองค์กรต้องการเครื่องแม่ขาย 5 เครื่องไม่ใช่เพียงแค่ 3 เครื่อง แต่ใน Cloud computing ในองค์กรเดียวกันสามารถลงทุนซื้อเครื่องแม่ขาย 3 เครื่อง และเช่าใช้อีก 2 เครื่องสำหรับ 5 วันที่เหลือของเดือน

10.1.1 คุณลักษณะของ Cloud

Cloud computing จะมีคุณลักษณะพื้นฐาน 3 ประการ:

- ความเป็นมาตรฐาน (Standardization)

  ความเป็นมาตรฐานจะช่วยให้สามารถสร้างทรัพยากรทางด้านไอทีขนาดใหญ่ได้จากส่วนประกอบที่ราคาไม่แพง ซึ่งตรงข้ามกับการกำหนดเอง

- ทรัพยากรเสมือน (Virtualization)

  ทำให้สามารถแบ่งทรัพยากรด้านไอทีออกเป็นส่วนๆและแบ่งให้ใช้ตามความต้องการได้ และเมื่อหลังจากการใช้งานแล้วทรัพยากรจะถูกส่งกลับไปที่กองกลางเพื่อส่งต่อให้ผู้อื่นใช้งานต่อ

- การทำงานแบบอัตโนมัติ (Automation)

  อนุญาตให้ผู้ใช้งานบน Cloud ควบคุมทรัพยากรโดยไม่ต้องรอการบริการจากผู้ดูแลระบบ นี้คือสิ่งสำคัญในระบบ Cloud ที่มีขนาดใหญ่

ลองคิดดูว่าในอดีตเราเคยใช้บริการของ Cloud หรือไม่ Facebook, Yahoo, และ Gmail คือตัวอย่าง ทุกคนได้รับบริการเป็นมาตรฐานเหมือนกัน มีการแบ่งปันทรัพยากร และเป็นการทำงานแบบอัตโนมัติ โดยสามารถสร้างบัญชีรายชื่อ และเริ่มใช้บริการได้อย่างทันที

10.1.2 รูปแบบการบริการของ Cloud computing

รูปแบบการบริการของ Cloud computing มี 3 ประเภทคือ:

- การบริการแบบโครงสร้างพื้นฐาน (Infrastructure as a service: IaaS)

- การบริการแบบแพลตฟอร์ม (Platform as a service: PaaS)

- การบริการแบบซอฟต์แวร์ (Software as a service: SaaS)

การบริการแบบโครงสร้างพื้นฐาน คือการให้บริการทางด้านศูนย์ข้อมูล ฮาร์ดแวร์ ระบบปฏิบัติการ โดยที่ไม่ผู้ใช้งานไม่ต้องกังวลในประเด็นดังกล่าว
การบริการแบบแพลตฟอร์ม คือการบริการที่ดูแลโปรแกรม หรือมิดเดิลแวร์ ตัวอย่างเช่น ในกรณีมิดูเดิลแวร์ของ IBM PaaS จะให้บริการสำหรับเครื่องแม่ขายของ DB2, WebSphere Application Server เป็นต้น
การบริการแบบซอฟต์แวร์เป็นการบริการซอฟต์แวร์ที่ต้องการใช้งาน เหมือนเป็นร้านขายซอฟต์แวร์ ซึ่งสามารถทำการเช่าใช้งานซอฟต์แวร์เป็นรายชั่วโมง ตัวอย่างของ SaaS เช่น Salesforce.com

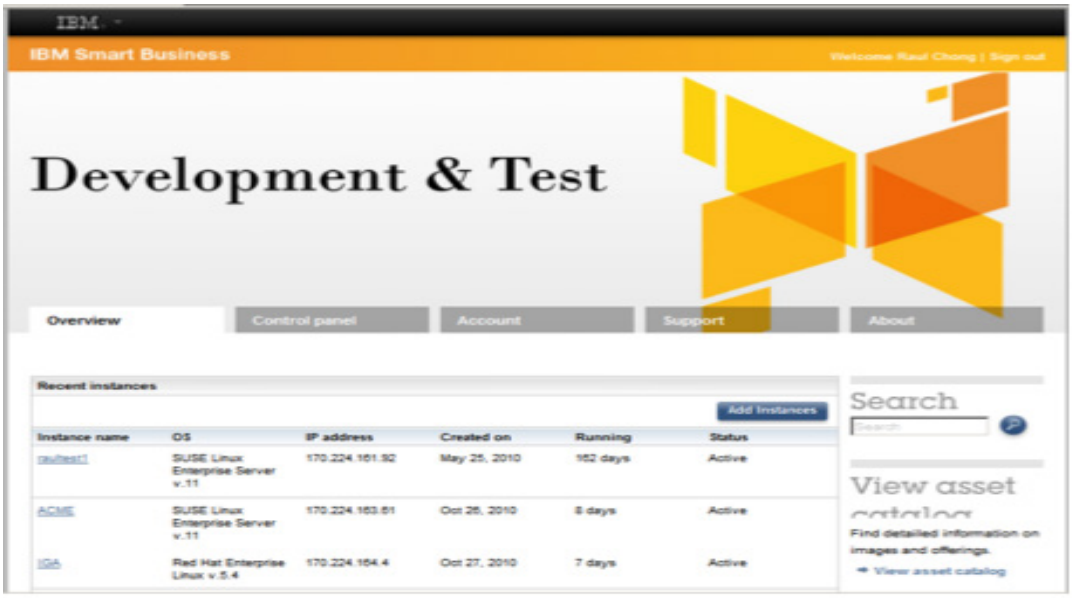
10.1.3 ผู้ให้บริการ Cloud

ในท้องตลาดมีผู้ให้บริการ Cloud อยู่ตอนนี้มากมายหลายเจ้า อย่างเช่น ไอบีเอ็ม จะมีบริการ “Smart Business Development and Test on the IBM Cloud”, Amazon ก็จะมีบริการ Amazon Web Services (AWS), Rackspace เป็นต้น

10.1.3.1 IBM Smart Business Development and Test on the IBM Cloud

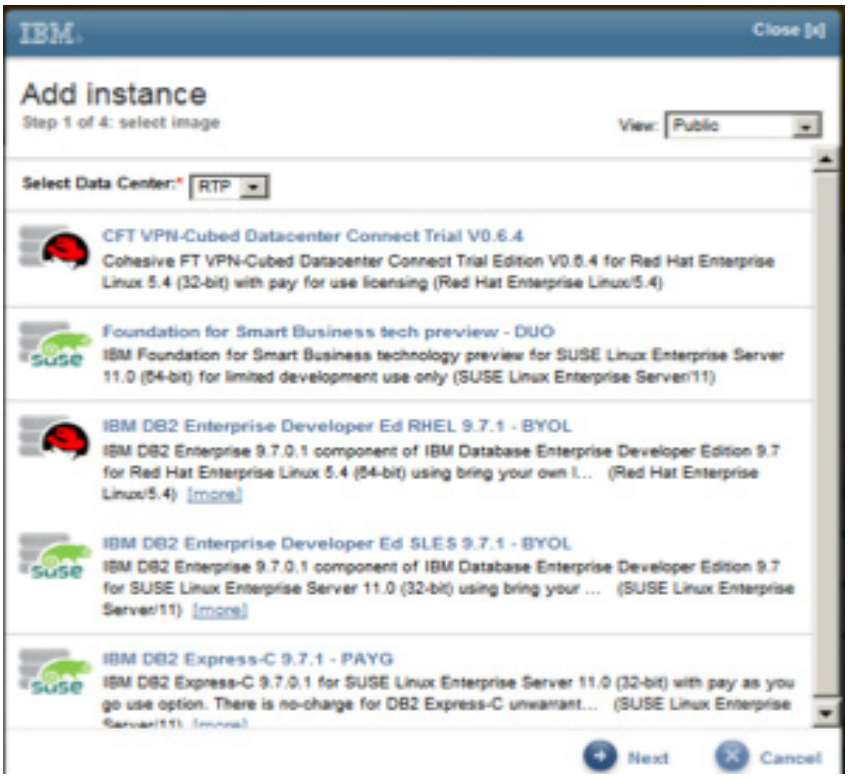
IBM Smart Business Development and Test on the IBM Cloud หรือ IBM developer cloud เป็นบริการ Cloud พิเศษที่ใช้ในการพัฒนา และทดสอบ การบริการมีศูนย์ข้อมูลในสหรัฐอเมริกา และยุโรป โดยอนุญาตให้คุณสามารถจัดเตรียมใช้งานเครื่องในสเปค Intel ทั้ง 32-บิต และ 64-บิต โดยใช้ระบบปฏิบัติการ Linux RedHat หรือ SUSE Linux หรือ Microsoft Windows โดยจะทำเป็นเครื่องแม่ขายเสมือน

IBM Developer Cloud สามารถใช้บริการได้ที่ https://www-147.ibm.com/cloud/enterprise รูปที่ 10.1 แสดงแดชบอร์ดของ IBM Developer Cloud โดยแสดงสามระบบงานที่ทำงานอยู่ แสดงด้วยประเภทระบบปฏิบัติการ และหมายเลขไอพีแอดเดส



รูปที่ 10.1 The IBM Developer Cloud dashboard

การสร้างระบบงานให้ทำงานได้ใช้เวลาเพียงไม่กี่นาที ทำได้ง่าย ๆ โดยการคลิกที่ปุ่ม “Add instances” จากนั้นก็ทำการเลือกใช้บริการได้ตามที่ปรากฏในรูปที่ 10.2



รูปที่ 10.2 เลือกการใช้บริการบน IBM Developer Cloud

ตัวอย่างเช่น ถ้าต้องการทำงานกับ IBM DB2 Express-C 9.7.1 – PAYG (Pay as you go) บนระบบปฏิบัติการ SUSE Linux สามารถทำได้ง่าย ๆ โดยการคลิกเลือกแล้วคลิกที่ปุ่ม Next จากนั้นจะปรากฏจอภาพแสดงขึ้นมาตามรูป 10.3

ในการกำหนดค่าสามารถเลือกประเภทของบริการได้ตามที่ต้องการ (32-บิต, 64-บิต) และจำนวนแกนของหน่วยประมวลผลโดยแบ่งประเภทตามชื่อ (copper, bronze, silver, gold) ตัวอย่างเช่น bronze ในระบบ 32-บิต จะให้บริการ 1 หน่วยประมวลผล มีความเร็วที่ 1.25 GHz, มีหน่วยความคำหลัก 2GB และ 175GB สำหรับหน่วยความจำสำรอง

รูปที่ 10.3 จะแสดงกรอบที่ต้องกำหนดค่าตัวอักษร ซึ่งจะใช้ในการสร้างคีย์ ที่ต้องทำการดาวน์โหลดมาเก็บไว้ที่คอมพิวเตอร์ส่วนตัวเพื่อที่ภายหลังจะสามารถ SSH ไปยังอินสแตนซ์ที่สร้างขึ้นโดยใช้ ssh ซอฟต์แวร์อย่างเช่นโปรแกรมที่ไม่เสียค่าใช้จ่ายมีชื่อว่า putty

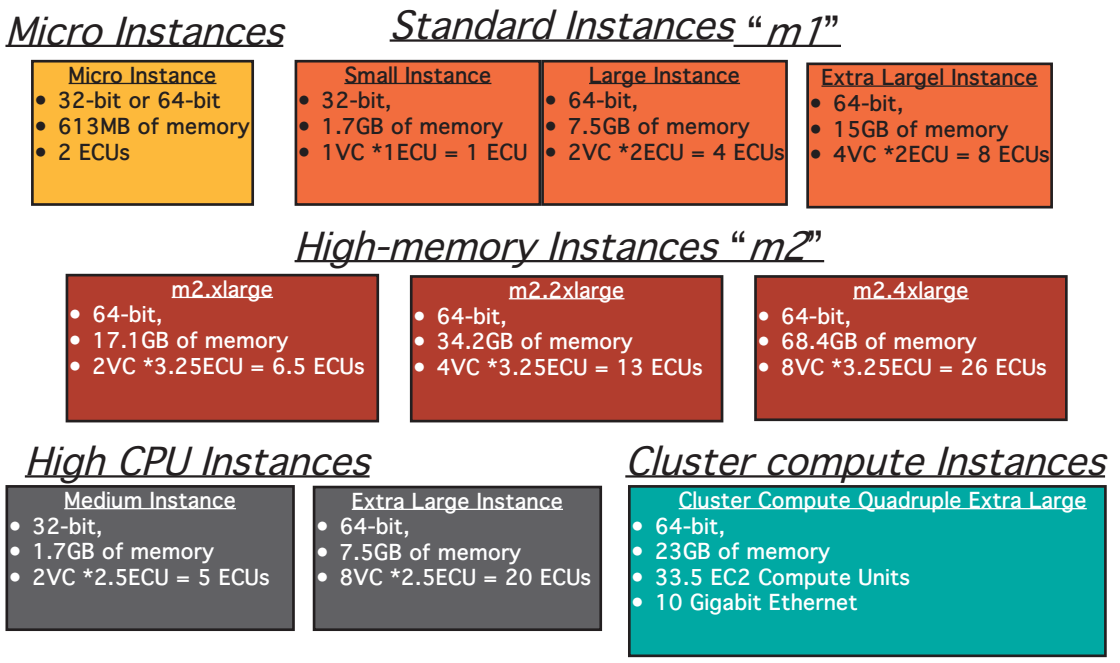
รูปที่ 10.3 กำหนดค่าเริ่มต้นสำหรับบริการ

หลังจากที่คลิก Next บริการก็จะถูกจัดเตรียมไว้ ซึ่งหมายความว่าทรัพยากรต่าง ๆ อย่างหน่วยประมวลผล และหน่วยความจำสำรองจะถูกจัดสรร ระบบปฏิบัติการจะถูกติดตั้ง และซอฟต์แวร์ (เช่น ระบบฐานข้อมูล DB2 Express-C ในตัวอย่างนี้) ก็จะถูกติดตั้ง กระบวนการทำงานที่กล่าวมาจะใช้เวลาเพียงไม่กี่นาที หลังจากนั้นก็สามารถเพิ่มตัวเก็บข้อมูลเข้าไปในอินสแตนซ์ของคุณได้

10.1.3.2 เว็บเซอร์วิสของบริษัท Amazon

Amazon Web Services หรือ AWS เป็นการบริการ Cloud ในรูปแบบโครงสร้างพื้นฐาน AWS จะมีศูนย์ข้อมูลอยู่ในสี่พื้นที่: ทางตะวันออกของสหรัฐอเมริกา ทางตะวันตกของสหรัฐอเมริกา ยุโรป และ เอเชียแปซิฟิก แต่ละพื้นที่จะรับผิดชอบหลายโซนเพื่อความต่อเนื่องในการทำงานทางธุรกิจ

ในลักษณะที่คล้ายกันกับ IBM developer cloud ด้วย AWS คุณสามารถเลือกเครื่องแม่ข่ายจำลองและเรียกใช้บริการ Elastic Cloud compute (EC2) instances ได้ โดยบริการเหล่านี้จะอยู่บนพื้นฐานหน่วยประมวลผลของ Intel (32 และ 64-บิต) และสามารถติดตั้งระบบปฏิบัติการได้ทั้ง Windows หรือ Linux สามารถเลือกหน่วยประมวลผล และหน่วยความจำสำรองได้หลากหลาย ดังในรูปที่ 10.4 เป็นการสรุปรูปแบบบริการต่าง ๆ บน AWS EC2 แต่ละบริการจะแสดงราคาการใช้งานต่อชั่วโมงที่ aws.amazon.com



VC = Virtual core
ECU = EC2 Compute Unit. 1 ECU ~ CPU capacity of 1.0 - 1.2 GHz 2007 or 2007 Xeon processsor

รูปที่ 10.4 แสดงประเภทของบริการบน AWS EC2

คุณสามารถเลือกใช้บริการโดยการเพิ่มหน่วยความจำสำรองใน AWS สามประเภทได้แก่

- Instance storage - เป็นบริการที่ไม่มีค่าใช้จ่ายพิเศษเพิ่ม อย่างไรก็ตามถ้ามีข้อขัดข้องของบริการ หรือคุณปิดบริการกะทันหันจะทำให้ข้อมูลของคุณหายไป
- Simple Storage Service (S3) - S3 ทำงานเหมือนกับการเก็บแฟ้มที่ใช้ในองค์กร คุณสามารถใช้งานผ่านโพโตคอล http ได้
- Elastic Block Storage (EBS) - EBS จะทำงานเหมือนทำงานอยู่บนฮาร์ดดิสก์บนเครื่องคอมพิวเตอร์ ซึ่งอนุญาตให้คุณจัดเก็บข้อมูลได้ และเหมาะสมกับการเก็บฐานข้อมูลแนวคิดนี้คล้ายๆ กับฐานข้อมูล

10.1.4 การจัดการความปลอดภัยบน Cloud

ความปลอดภัยมีความเสี่ยงสูงนั่นเป็นเหตุผลที่ทำไมองค์การต่าง ๆ ถึงอาจไม่อยากใช้บริการ Cloud แบบสาธารณะ แนวความคิดที่จะให้บุคคลที่สามมาจัดการข้อมูลที่เป็นความลับเป็นเรื่องที่มีความเสี่ยงสูง

ในขณะที่มีความกังวลอยู่ การประมวลผลแบบ Cloud ก็พัฒนาต่อไปเรื่อย ๆ การให้บริการแบบ Cloud ส่วนตัวเป็นทางเลือกที่จะประกันความเชื่อมั่นในความปลอดภัยของข้อมูลของลูกค้า ฮาร์ดแวร์ และซอฟต์แวร์ เช่น IBM Cloudburst™ และ IBM WebSphere Cloudburst Appliance ทำงานร่วมกันเพื่อให้องค์การพัฒนา Cloud ของตนเอง

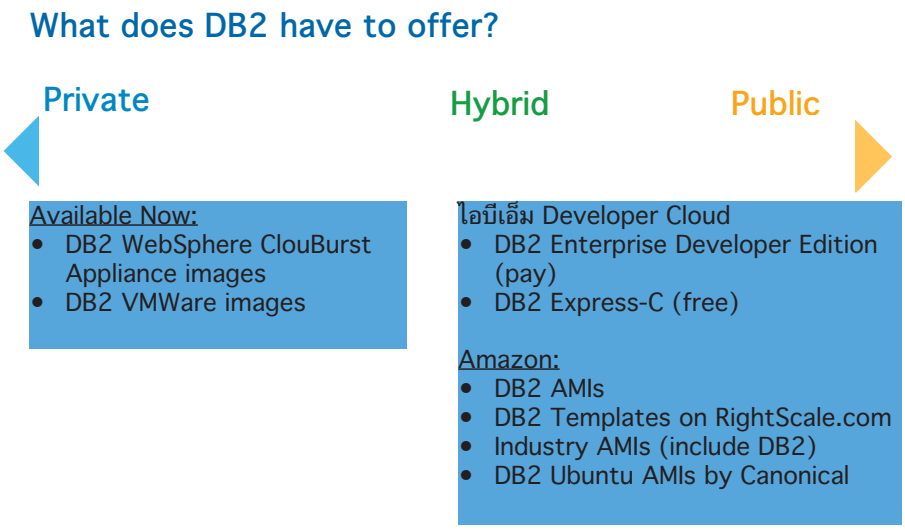
องค์กรอย่างเช่น Amazon และ ไอบีเอ็ม เสนอบริการทำ Cloud ส่วนตัวแบบจำลอง ที่ซึ่งเครื่องแม่ข่ายยังคงอยู่ที่ศูนย์ข้อมูลของผู้ให้บริการแต่ไม่สามารถเข้าถึงได้โดยการใช้ Internet ความปลอดภัยจะถูกกำหนดโดยองค์กรด้วยตนเอง

องค์กรสามารถใช้งาน hybrid clouds ซึ่งเก็บข้อมูลที่สำคัญไว้ใน cloud แบบส่วนตัว ในขณะที่ข้อมูลที่ใช้สำหรับการพัฒนา หรือทดสอบจะถูกเก็บไว้ใน cloud แบบสาธารณะ

10.1.5 ฐานข้อมูล และ Cloud

การประมวลผลแบบ Cloud จะเป็นวิธีการใหม่สำหรับทรัพยากรทางด้านไอทีรวมถึงฐานข้อมูล เครื่องแม่ข่ายข้อมูล IBM DB2 ได้ถูกเพิ่มความสามารถของ Cloud เป็นที่เรียบร้อยแล้ว

ผลิตภัณฑ์ต่าง ๆ ของ DB2 ทั้งที่ใช้ในการทำงานการพัฒนา และการทดสอบมีอยู่บน AWS และ IBM developer cloud นอกจากนั้น DB2 ยังมีใน cloud แบบส่วนตัวโดยใช้ VMware หรือ WebSphere Cloudburst appliance รูปที่ 10.5 สรุปการทำงานของ DB2 ที่สนับสนุนการทำงานแบบ ส่วนตัว (private cloud) แบบผสม (hybrid cloud) และสาธารณะ (public cloud)



รูปที่ 10.5 สรุป DB2 ที่มีบริการ Cloud แบบ Private, Hybrid, and Public

ในแง่ของการอนุญาตให้ใช้สิทธิ์ สามารถใช้วิธีการเหล่านี้ได้:

- นำลิขสิทธิ์การใช้งานของ DB2 อยู่แล้วของคุณ (BYOL) มาใช้ในCloud
- Pay as you go (PAYG) เสียค่าใช้จ่ายเฉพาะเท่าที่คุณใช้งาน

คุณสามารถใช้งาน DB2 Express-C บน Cloud โดยไม่เสียค่าใช้จ่าย แต่คุณยังคงต้องเสียค่าใช้จ่ายสำหรับการให้บริการ Cloud แบบพื้นฐานอยู่

ในความสามารถของ DB2 สามารถใช้งานร่วมกับความสามารถของ Cloud เช่น:

- Database Partitioning Feature (DPF)
- High Availability Disaster Recovery (HADR)
- Compression

DPF เป็นสถาปัตยกรรมที่อยู่บนพื้นฐานของการไม่มีการใช้งานร่วมกัน ซึ่งเข้ากันกับการทำงานในรูปแบบของ Cloud provisioning DPF เป็นแนวคิดที่เหมาะสมกับสภาพแวดล้อมแบบคลังข้อมูล ที่ซึ่งมีข้อมูลปริมาณมหาศาลที่ต้องการสอบถามข้อมูลด้วย Business Intelligence Reports DPFทำให้ การสอบถามข้อมูลเกิดขึ้นแบบคู่ขนานในกลุ่มของเครื่องแม่ข่ายโดยอัตโนมัติ การทำงานแบบคู่ขนานนี้เกิดขึ้นได้โดยไม่ต้องปรับเปลี่ยนโปรแกรมใช้งานแต่อย่างใด คุณสามารถเพิ่มเครื่องแม่ข่ายได้ตามต้องการบน Cloud ที่ซึ่งมีหน่วยประมวลผล และหน่วยความจำสำรองเป็นของตนเอง DB2 จะทำการปรับการทำงานซึ่งจะทำให้ประสิทธิภาพโดยรวมเพิ่มขึ้นเกือบเป็นเส้นตรง

HADR เป็นคุณสมบัติของ DB2 ในท้องตลาดมามากกว่า 10 ปี HADR ทำงานด้วยเครื่องแม่ข่ายสองเครื่อง เครื่องหลัก และเครื่องสำรอง เมื่อเครื่องแม่ข่ายทั้งสองถูกติดตั้งไว้ที่เดียวกันจะมีความพร้อมในการทำงานสูง แต่เมื่อเครื่องอยู่แยกกัน (ต่างมลรัฐ หรือต่างประเทศ) HADR จะถูกใช้สำหรับการกู้คืนความเสียหาย ซึ่งการกู้คืนความเสียหายเป็นสิ่งที่ต้องการ และเสียค่าใช้จ่ายมากสำหรับด้านไอที ที่เสียค่าใช้จ่ายมากก็เพราะองค์การ

จะต้องจ่ายสำหรับพื้นที่วางเครื่องแม่ข่ายอีกสถานที่หนึ่งซึ่งจะต้องมีทรัพยากรด้านไอทีต่าง ๆ (เครื่องแม่ข่าย หน่วยความจำสำรอง ระบบเครือข่าย) การประมวลผลแบบ Cloud ได้จัดสรรสิ่งเหล่านี้ไว้ให้เรียบร้อยแล้ว โดยใน Cloud คุณ “สามารถเช่า” ศูนย์ข้อมูลแยกออกมาได้โดยที่ไม่เสียค่าใช้จ่ายสำหรับพื้นที่ ไฟฟ้า ระบบทำความเย็น ระบบการรักษาความปลอดภัย เป็นต้น คุณสามารถมีทรัพยากรทั้งหมดโดยไม่ต้องเสียค่าใช้จ่ายที่นอกเหนืองบประมาณ และยังสามารถเข้าไปยัง “DR site” บน Cloud ได้จากที่ไหนก็ได้อย่างปลอดภัยโดยการใช้เว็บเบราว์เซอร์ และ ssh ด้วย HADR คุณสามารถให้เครื่องแม่ข่ายหลักทำงานอยู่ที่บริษัทของคุณ และเครื่องสำรองอยู่บน Cloud สำหรับการเพิ่มความปลอดภัยคุณสามารถนำเครื่องสำรองไปวางไว้บน Cloud แบบส่วนตัวได้ HADR อนุญาตให้ระบบสำรองของคุณทำงานหลังจากระบบหลักล้มเหลวลงภายใน 15 วินาที

ถ้าต้องการประหยัดงบประมาณ และค่าใช้จ่ายเราสามารถใช้งานคุณสมบัติในการบีบอัดของ DB2 ได้บน Cloud เพื่อประหยัดปริมาณทรัพยากรบน Cloud คุณสมบัติในการบีบอัดของ DB2 จะช่วยเพิ่มประสิทธิภาพในการทำงานในภาพรวมทั้งหมด

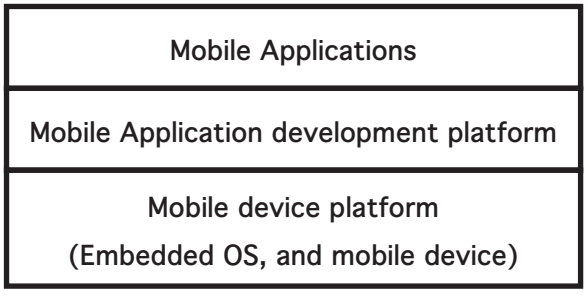
มีคุณสมบัติอื่น ๆ อีกมากมายของ DB2 ที่ไม่ได้กล่าวถึงในหนังสือเล่มนี้ แต่สามารถประยุกต์ใช้งานได้ตามหัวข้อที่กล่าวมา

10.2 การพัฒนาซอฟต์แวร์บนโทรศัพท์เคลื่อนที่

ซอฟต์แวร์บนโทรศัพท์เคลื่อนที่ เป็นการพัฒนาซอฟต์แวร์ให้สามารถทำงานได้บนสภาวะแวดล้อมของโทรศัพท์เคลื่อนที่ ซึ่งองค์กรขนาดใหญ่ได้พัฒนาซอฟต์แวร์บนโทรศัพท์เคลื่อนที่ให้สามารถเรียกใช้งานข้อมูลได้จากเครื่องคอมพิวเตอร์แม่ข่ายได้ทุกที่ทุกเวลา นอกจากนี้ซอฟต์แวร์บนโทรศัพท์เคลื่อนที่ได้ใช้ Cloud เป็นเบื้องหลังได้กลายเป็นที่นิยมมากขึ้นเรื่อยๆ

การออกแบบ และการจัดทำซอฟต์แวร์บนโทรศัพท์เคลื่อนที่ไม่เหมือนกับการพัฒนาซอฟต์แวร์บนเครื่องคอมพิวเตอร์ส่วนบุคคล สิ่งที่สำคัญสำหรับนักพัฒนาซอฟต์แวร์บนโทรศัพท์เคลื่อนที่คือตัดสินใจเกี่ยวกับสภาวะแวดล้อมที่ซอฟต์แวร์จะถูกใช้งาน จากมุมมองทางธุรกิจมันเป็นเรื่องยากสำหรับผู้จัดการโครงการในการประเมินความเสี่ยงของโครงการที่ถูกเปรียบเทียบกันในหลายแพลตฟอร์ม

ทุกวันนี้ แพลตฟอร์มเก่า ๆ อย่าง Symbian, Microsoft Windows Mobile, Linux และ BlackBerry OS กำลังถูกเข้ารวมในตลาดด้วย Apple’s OS X iPhone, Android และ Palm’s Web OS ซึ่งเป็นการเพิ่มความยุ่งยากให้แก่นักพัฒนา รูปที่ 10.6 ประเภทของสถาปัตยกรรมในการพัฒนาซอฟต์แวร์บนโทรศัพท์เคลื่อนที่



รูปที่ 10.6 ประเภทของสถาปัตยกรรมในการพัฒนาซอฟต์แวร์บนโทรศัพท์เคลื่อนที่

ในรูปที่ 10.6 แสดงถึงการพัฒนาซอฟต์แวร์บนโทรศัพท์เคลื่อนที่ที่อยู่บนสุดของอีกสองเลเยอร์

Mobile device platform อ้างอิงถึงฮาร์ดแวร์ของโทรศัพท์เคลื่อนที่ และระบบปฏิบัติการ และหรือซอฟต์แวร์ที่ติดพวงมาด้วย ตัวอย่างเช่น ระบบปฏิบัติการ Microsoft Windows Mobile 5.0 ที่ถูกติดตั้งมาบนเครื่อง Dell AXIM X51v

Mobile application development platform อ้างอิงถึงการรวมกันของภาษาในการพัฒนาโปรแกรม, APIs ในการพัฒนา และสภาวะแวดล้อมในการทำงานของภาษานั้น ๆ บนโทรศัพท์เคลื่อนที่ Mobile application development platform จะอยู่ตำแหน่งด้านบนของ Mobile device platform และทำการป้องกันซอฟต์แวร์จากการเสียหายที่เกิดจากซอฟต์แวร์ที่มุ่งร้าย ตัวอย่างเช่น การพัฒนาซอฟต์แวร์สำหรับ Windows Mobile 6.0 ต้องพัฒนาบน Microsoft Visual Studio 2005/2008 รวมกับ Windows Mobile 6.0 SDK

เนื่องจากมี Mobile device platforms และ mobile application development platforms ให้เลือกหลากหลาย จึงต้องถามตัวเองด้วยคำถามต่อไปนี้ก่อนที่จะพัฒนาซอฟต์แวร์: คุณกำลังพัฒนาซอฟต์แวร์

สำหรับอุปกรณ์เฉพาะ หรือแพลตฟอร์มเฉพาะหรือไม่?

10.2.1 การพัฒนาสำหรับอุปกรณ์เฉพาะ

เมื่อทำการพัฒนาซอฟต์แวร์สำหรับอุปกรณ์เฉพาะ (หรือชุดอุปกรณ์) หรือ ระบบปฏิบัติการของอุปกรณ์ ผู้จัดจำหน่ายจะมีแพลตฟอร์มของ SDK ตัวจำลอง และเครื่องมืออื่น ๆ ที่สามารถช่วยในการพัฒนาซอฟต์แวร์ของแต่ละแพลตฟอร์ม

ตารางที่ 10.2 จะเป็นการแสดงรายการของ SDKs, เครื่องมือ ของแต่ละแพลตฟอร์ม

Operating System (OS)	Language/OS options	SDKs/Tools	Developer Web Site
Symbian OS	C++, Java, Ruby, Python, Perl, OPL, Flash Lite, .NET	Carbide C++, IDE(Nokia) – C++, and Java SDKs, per device	developer.symbian.com
Windows Mobile	Java, C++	Visual Studio, Platform Builder, Embedded Visual C++ (eVC), Free Pascal, and Lazarus	Microsoft Windows Embedded Developer Center
iPhone OS	Objective-C	Xcode	www.apple.com/iphone
Android	Java	Eclipse, Android SDK, Android Development Tool Plugin, JDK 5	www.android.com
Palm OS	C, C++, Java	Palm OS SDK, Java development with IBM Websphere EveryPlace Micro Environment, Palm Windows Mobile SDK for Palm products on Windows Mobile platform	www.palm.com/us/developer

ตารางที่ 10.2 แสดงรายการของ SDKs เครื่องมือ ของแต่ละแพลตฟอร์ม

10.2.2 การพัฒนาสำหรับแพลตฟอร์มของแอพพลิเคชัน

ทุกคนมีความเชื่อในภาษา และสภาวะแวดล้อมในการพัฒนาซอฟต์แวร์ แต่ทำไมพวกเขาถึงมีความเชื่อเช่นนั้น?

การพัฒนาซอฟต์แวร์บนโทรศัพท์เคลื่อนที่ได้หยิบยกคำถามนี้ขึ้นมาครั้งแล้วครั้งเล่า และคุณอาจจะต้องทำการประเมินคำตอบของคุณอีกครั้งว่าซอฟต์แวร์บนโทรศัพท์เคลื่อนที่ของคุณที่พัฒนาด้วย Java, .NET (C#, VB), C, C++ และภาษาอื่น ๆ มีความพร้อมใช้งานในการพัฒนาซอฟต์แวร์บนโทรศัพท์เคลื่อนที่ ในบางทีอาจใช้วิธีการประเมินภาษาเหล่านี้เหมือนกับการพัฒนาซอฟต์แวร์บนเครื่องคอมพิวเตอร์ส่วนบุคคล หรือเครื่องแม่ข่ายมาประยุกต์ใช้บนการพัฒนาบนโทรศัพท์เคลื่อนที่ อย่างไรก็ตามการพัฒนาซอฟต์แวร์บนโทรศัพท์เคลื่อนที่นำมาสู่การตัดสินใจที่แตกต่างจากการพัฒนาซอฟต์แวร์บนคอมพิวเตอร์ส่วนบุคคล/เครื่องแม่ข่ายแบบเดิม ๆ

ตัวอย่างเช่น Java มีข้อได้เปรียบตรงที่สามารถใช้งานได้หลากหลายแพลตฟอร์ม หรือได้หลากหลายบนอุปกรณ์มือถือ อย่างไรก็ตาม Java ก็ไม่ได้สนับสนุนอุปกรณ์เหล่านั้นทั้งหมด ซึ่งการที่จะทำให้ Java นั้นสามารถทำงานได้หลากหลายแพลตฟอร์มนั้นต้องทำการกำหนดค่า และโปรไฟล์ที่แตกต่างกันตามอุปกรณ์ และความสามารถของอุปกรณ์นั้น ๆ ดังนั้นการเขียนไปยังแพลตฟอร์มเหล่านี้อาจจะทำให้คุณไม่สามารถเข้าถึงกลุ่มลูกค้าของคุณได

คุณจะดูความเหมาะสมของซอฟต์แวร์สำหรับอุปกรณ์มือถืออย่างไร? ในสองหัวข้อถัดไปจะแสดงให้เห็นอุปกรณ์มือถือที่มีความแพร่หลาย และตัวเลือกของแพลตฟอร์มของโทรศัพท์เคลื่อนที่ นำมาเปรียบเทียบกันระหว่างแพลตฟอร์มเหล่านั้น

10.2.3 แพลตฟอร์มของโทรศัพท์เคลื่อนที่

แพลตฟอร์มที่ถูกใช้บนโทรศัพท์เคลื่อนที่อย่างแพร่หลายจะถูกอธิบายไว้ดังต่อไปนี้

10.2.3.1 Windows Mobile

Windows Mobile มีที่มาจาก Windows CE (วางจำหน่ายครั้งแรกเมื่อปีค.ศ. 1996) เป็นส่วนย่อยของ Microsoft Windows APIs ซึ่งเป็นระบบปฏิบัติการที่มีขนาดกะทัดรัด และทำงานแบบเรียลไทม์ ซึ่งออกแบบมาสำหรับอุปกรณ์คอมพิวเตอร์มือถือ และเทอร์มินัลแบบไร้สาย หลังจากการเริ่มต้นที่ผิดพลาดของระบบ PDA แบบสแตนด์อโลน Windows Mobile ได้ประสบความสำเร็จบน Pda และโทรศัพท์เคลื่อนที่โดยเฉพาะ Motorola Q และอุปกรณ์อย่างเช่น Treo

Microsoft ได้ให้บริการเทคโนโลยี ActiveSync สำหรับการซิงโครไนส์ทั้งในส่วนของผู้ใช้ส่วนบุคคลและผู้ใช้ระดับองค์กร นอกจากนี้ในผลิตภัณฑ์รุ่นปัจจุบันของ Microsoft Windows Mobile จะเป็นแพลตฟอร์มเอนกประสงค์สำหรับนักพัฒนาที่อนุญาตให้พัฒนาซอฟต์แวร์ได้ด้วยภาษา C หรือ C++ ด้วยการสนับสนุนการจัดการคำสั่งด้วย .NET Compact Framework

10.2.3.2 Symbian OS

Symbian OS คือระบบปฏิบัติการที่ออกแบบมาสำหรับอุปกรณ์มือถือ ประกอบได้ด้วยไลบรารี, อินเทอร์เฟซของผู้ใช้งาน, เฟรมเวิร์คและเครื่องมืออื่นๆ ที่สนับสนุนพัฒนาขึ้นโดยบริษัท Symbian Ltd. โดยเริ่มต้นที่ EPOC Release 5 ที่เป็นซอฟต์แวร์ EPOC ที่พัฒนาท้ายสุดโดย Psion ที่ซึ่งได้เข้าร่วมกิจการSymbian มีแพลตฟอร์มที่มีประสิทธิภาพมากซึ่งถูกพัฒนาบนภาษา C++ โดยใช้ Symbian's frameworks สำหรับซอฟต์แวร์ และบริการ แพลตฟอร์ม Symbian จะทำงานด้วย Java ME ดังนั้นโทรศัพท์ที่ใช้ระบบปฏิบัติการ Symbian สามารถใช้งานซอฟต์แวร์ของ Symbian ได้ดีพอ ๆ กับซอฟต์แวร์ของ Java ME

10.2.3.3 iPhone

iPhone เป็น มัลติมีเดียสมาร์ทโฟนที่สามารถเชื่อมต่ออินเทอร์เน็ต ถูกออกแบบ และทำการตลาดโดยบริษัท Apple Inc. เป็นแพลตฟอร์มแบบปิด อนุญาตให้ติดตั้งซอฟต์แวร์ผ่านทางเว็บคิตเว็บเบราว์เซอร์ Safari เนื่องจากไม่มีฮาร์ดแวร์ที่เป็นแป้นพิมพ์ จึงต้องใช้แป้นพิมพ์จำลองที่รอรับระบบมัลติทัชสกีน

iPhone และ iPod Touch SDK ใช้ภาษา Objective C ซึ่งอยู่บนการเขียนโปรแกรมบนภาษา C เป็นภาษาที่ใช้ในการพัฒนา สำหรับเครื่องมือในการพัฒนาจะใช้โปรแกรม Xcode ซึ่งเป็นเครื่องมือสำหรับการพัฒนาซอฟต์แวร์บน Max OS x ของ Apple

10.2.3.4 Android

Android เป็นระบบเปิด และเป็นซอฟต์แวร์ที่ไม่เสียค่าใช้จ่ายซึ่งรวมถึง ระบบปฏิบัติการ, มิดเดิลแวร์,

และโปรแกรมประยุกต์ที่สำคัญ โดยภาษาที่ใช้ในการพัฒนา Android คือ ภาษา Java

การพัฒนาซอฟต์แวร์บน Android จะพัฒนาได้เฉพาะการใช้ภาษา Java ดังนั้นจึงต้องการ Java SDK ซึ่งรวมอยู่ในเครื่องมือสำหรับการพัฒนา ซึ่งเครื่องมือเหล่านั้นจะรวมไปถึง ตัวดีบัก, ไลบรารี, ตัวจำลอง (อยู่บนพื้นฐานของ QEMU), เอกสาร, โค๊ดตัวอย่าง และเอกสารช่วยสอน ในปัจจุบันจะรองรับการพัฒนาบนสถาปัตยกรรม x86 ที่ทำงานบนระบบปฏิบัติการ Linux (ทุก ๆ Linux ที่มีการแจกจ่าย), ระบบปฏิบัติการ Max OS X 10.4.8 หรือรุ่นใหม่กว่านี้, Windows XP หรือ Vista โดยระบบปฏิบัติการเหล่านั้นจะต้องติดตั้ง Java Development Kit, Apache Ant และ Python 2.2 หรือรุ่นใหม่กว่านี้ โดยเครื่องมือสำหรับการพัฒนาจะใช้ Eclipse (3.2 หรือรุ่นใหม่กว่านี้) โดยใช้ปลั๊กอิน Android Development Tools (ADT) และยังสามารถใช้เครื่องมือในการแก้ไขข้อความในการแก้ไข Java และ XML จากนั้นใช้คำสั่งจากเครื่องมือในคอมมานไลน์ในการสร้าง, ประมวลผล และดีบักโปรแกรมบน Android

10.2.4 การพัฒนาโมบายแอพพลิเคชันแพลตฟอร์ม

สำหรับในหัวข้อนี้จะเป็นการแนะนำแพลตฟอร์มในการพัฒนาซึ่งนำไปสู่โปรแกรมของอุปกรณ์มือถือ

10.2.4.1 Java Micro Edition (Java ME)

Java ME, หรือที่เรียกว่า Java 2 Micro Edition (J2ME) เป็นแพลตฟอร์มที่ไม่ใช้ทรัพยากรมากบนอุปกรณ์มือถือ และรวมถึงอุปกรณ์อื่น ๆ โดยการเขียนเพียงครั้งเดียวสามารถทำงานที่ไหนก็ได้ นักพัฒนา Java ME ยังคงต้องใช้ทักษะในการเคลื่อนย้ายโปรแกรมจากอุปกรณ์หนึ่งไปยังอุปกรณ์หนึ่ง เพราะว่าภาษาระดับภาษาเครื่อง เช่น C หรือ C++ บนแพลตฟอร์มอื่นจะต้องอาศัยความรู้พิเศษในด้าน API ของแต่ละแพลตฟอร์มเช่น Windows Mobile หรือ Symbian OS

10.2.4.2 .NET Compact Framework

แพลตฟอร์มนี้เป็นผลิตภัณฑ์ .NET Framework สำหรับแพลตฟอร์ม Windows CE ซึ่งเป็นส่วนย่อยของแพลตฟอร์ม .NET Framework แบบมาตรฐาน ซึ่งตัดออกมาเฉพาะในส่วนของคลาสที่ต้องใช้งานในอุปกรณ์มือถือ

ในปัจจุบัน .NET Compact Framework V1.0, V2.0 และ V3.5 ได้เตรียมไว้สำหรับพร้อมให้ผู้พัฒนาได้ใช้งาน แต่คุณคงสงสัยว่าจะต้องเลือกใช้งานผลิตภัณฑ์ในรุ่นไหน? “รุ่นล่าสุด” คือทางเลือกที่ดีที่สุด แต่ก็มีข้อขัดแย้งกันในบางประเด็น เช่นถ้าคุณพัฒนาโปรแกรมบนผลิตภัณฑ์รุ่น 1.0 คุณสามารถมั่นใจได้ว่าโปรแกรมของคุณสามารถทำงานได้ทุกอุปกรณ์ เพราะผลิตภัณฑ์รุ่น 2.0 และรุ่นใหม่กว่านี้ของ .NET Compact Framework จะรองรับการทำงานของผลิตภัณฑ์รุ่นเก่า ๆ แต่อย่างไรก็ตามถ้าคุณพัฒนาโปรแกรมโดยใช้งานผลิตภัณฑ์รุ่น 2.0 ผลิตภัณฑ์รุ่นนี้จะต้องทำการติดตั้ง .NET Compact Framework Runtime บนอุปกรณ์ที่จะทำงานด้วย

10.2.4.3 Native C++

โปรแกรมที่พัฒนาด้วยภาษา C++ จะทำงานแบบ native บนอุปกรณ์นั้น ซึ่งทำให้การโอนย้ายโปรแกรมเป็นไปได้ยาก แต่จะทำให้คุณสามารถเชื่อมต่อกับอุปกรณ์ต่าง ๆ ได้อย่างรวดเร็วมีประสิทธิภาพ และควบคุมอุปกรณ์ได้หลากหลายมากขึ้น

โดยทั่วไป Native C++ จะพัฒนาโดยใช้เครื่องมือ เช่น Microsoft Visual C++6/.NET, Eclipse ที่ติดตั้ง C++ SDK, หรือ Borland C++ Builder ส่วนการพัฒนาสำหรับระบบปฏิบัติการ Symbian จะใช้เครื่องมือที่ชื่อว่า Metrowerks CodeWarrior Studio ที่ต้องติดตั้ง Symbian C++ SDK ของ Nokia มาด้วย

10.2.4.4 Binary Runtime Environment สำหรับเครือข่ายไร้สาย

Binary Runtime Environment for Wireless (BREW) เป็นแพลตฟอร์มสำหรับการพัฒนาด้วย Qualcomm สำหรับโทรศัพท์บนพื้นฐาน CDMA โดย BREW จะบริการ SDK และตัวจำลองสำหรับการพัฒนาและทดสอบโปรแกรมที่พัฒนาโดยภาษา C หรือ C++

ในการพัฒนา BREW คุณต้องการเครื่องมือ เช่น BREW SDK, ARM ReslView Compilation Tools, Visual C++ ฯลฯ

10.2.5 แนวโน้มยุคต่อไปในการพัฒนาซอฟต์แวร์สำหรับโทรศัพท์เคลื่อนที่

การพัฒนาแอพพลิเคชันหลัก ๆ และบริการบนโทรศัพท์เคลื่อนที่ในยุคต่อไปจะมุ่นเน้นไปทาง PDA หรือสมาร์ทโฟนเป็นส่วนใหญ่ โดยผู้ให้บริการทางด้านมือถือจะพัฒนาความสามารถ และบริการเพื่อใช้ประโยชน์จากระบบ 3G และโครงข่ายข้อมูลแบบไร้สาย ส่วนโปรแกรมแบบเดิม ๆ เช่น เสียง และบริการข้อมูล เช่น การท่อง

อินเทอร์เน็ตจะไม่เพียงพอ ผู้บริโภคจะมองหาอุปกรณ์มือถือที่ใช้พลังงานต่ำ สามารถเชื่อมต่อกับโครงข่ายไร้สายเพื่อเข้าถึงบริการทางด้านบันเทิง และข้อมูลสำหรับการดำเนินธุรกิจ

ในโลกของโทรศัพท์เคลื่อนที่ได้มีการตลาดที่หลากหลายมากขึ้น เช่น ระบบ Telematics, m-Commerce และ m-Enterprise, บริการ Mobile Multimedia Streaming, Mobile Messaging, Location Based Mobile Computing และบริการอื่น ๆ

10.2.6 DB2 ในทุก ๆ ที่

ผลิตภัณฑ์ของ DB2 ที่ใช้สำหรับอุปกรณ์มือถือคือ DB2 Everyplace ซึ่ง DB2 Everyplace จะมีคุณสมบัติที่เป็นฐานข้อมูลเชิงสัมพันธ์ที่มีขนาดเล็ก และมีประสิทธิภาพสูงในการซิงโครไนส์ข้อมูล นั่นทำให้โปรแกรมสำหรับระดับองค์กร และข้อมูลต่าง ๆ สามารถใช้งานได้อย่างปลอดภัยในอุปกรณ์มือถือ

หมายเหตุ: ถ้าต้องการเรียนรู้เพิ่มเติมเกี่ยวกับการพัฒนาโปรแกรมบนมือถือสามารถอ้างอิงได้โดยไม่เสียค่าใช้จ่ายที่ Getting started with mobile application development, ซึ่งเป็นส่วนหนึ่งของชุดหนังสือของ DB2

10.3 ธุรกิจชาญฉลาด (Business intelligence and appliances)

เป็นธรรมดาที่ฐานข้อมูลถูกใช้สำหรับการทำธุรกรรมออนไลน์ เช่น การทำธุรกรรมทางการเงินกับธนาคาร แต่ฐานข้อมูลยังสามารถถูกใช้ใน business intelligence มาช่วยในการดูแนวโน้ม และรูปแบบที่สามารถช่วยในการตัดสินใจให้ดียิ่งขึ้นได้อีก โดยองค์กรทั้งหลายในปัจจุบันจะมีการเก็บรวบรวมข้อมูลเป็นปริมาณที่มากมายในแต่ละวัน ซึ่งข้อมูลเหล่านี้จะถูกเก็บไว้ในคลังข้อมูลเพื่อทำการประมวลผลเป็นรายงานด้วยซอฟต์แวร์อย่างเช่น IBM Cognos®

องค์กรต่าง ๆ ที่พยายามที่จะสร้างคลังข้อมูลขึ้นมาเพื่อใช้งานมีแนวคิดที่จะเสาะหาอุปกรณ์ในการประมวลผลคลังข้อมูล ซึ่งอุปกรณ์นั้นจะเป็นการรวมตัวของฮาร์ดแวร์ และซอฟต์แวร์ที่ประสานงานกันอย่างแน่นหนา และทำการทดสอบว่าสามารถทำงานตามที่ต้องการได้ โดย IBM Smart Analytics System เป็นหนึ่งในอุปกรณ์สำหรับประมวลผลข้อมูลในคลังข้อมูล และการทำ Business Intelligence

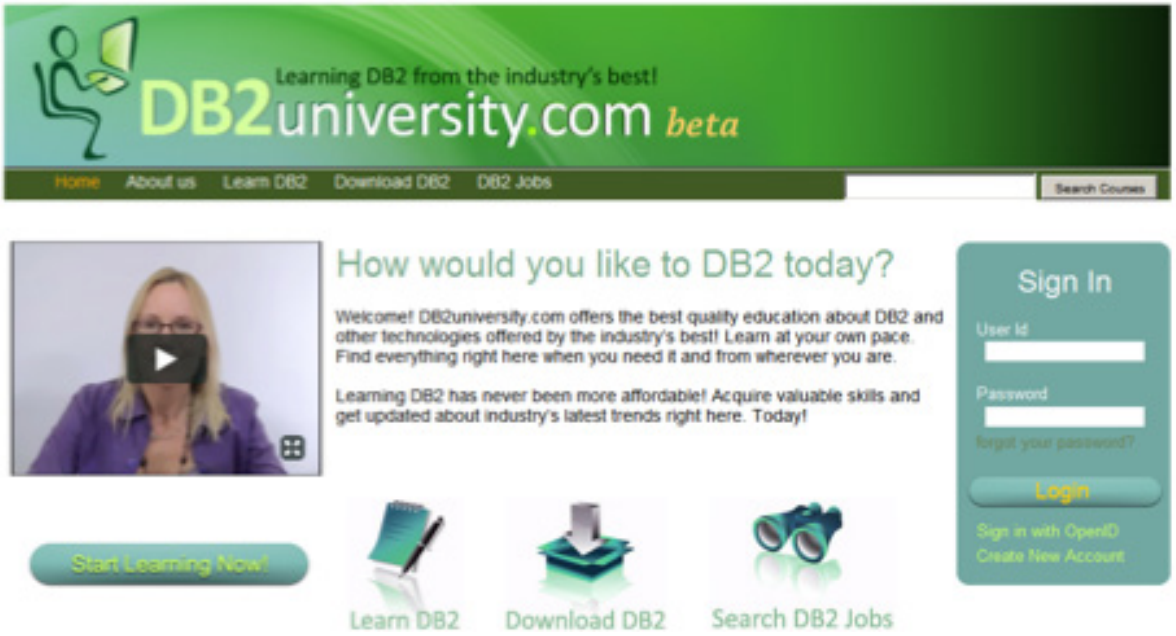
หมายเหตุ: ถ้าต้องการเรียนรู้เพิ่มเติมเกี่ยวกับคลังข้อมูล และ business intelligence สามารถอ้างอิงได้โดยไม่เสียค่าใช้จ่ายที่ Getting started with data warehousing, ซึ่งเป็นส่วนหนึ่งของชุดหนังสือของ DB2

10.4 db2university.com: การใช้งานโปรแกรมประยุกต์บน Cloud (กรณีศึกษา)

ต้องขอบคุณอินเทอร์เน็ตที่ทำให้เกิดความก้าวหน้าทางเทคโนโลยี โดยมีมหาวิทยาลัยหลาย ๆ แห่งพยายามที่จะทำการเรียนการสอนผ่านอินเทอร์เน็ต เช่น การลงทะเบียนสำหรับนักศึกษาที่ลงทะเบียนเรียนหลักสูตรออนไลน์ โดยที่วิชาเหล่านี้มักจะประกอบได้ด้วย การเรียนการสอนผ่านเว็บ, กระดานข่าวออนไลน์ ฯลฯ ซึ่งเป็นการเรียนการสอนแห่งอนาคตอันใกล้ที่ไม่จำเป็นต้องเข้าเรียนที่มหาวิทยาลัย

การศึกษาแบบออนไลน์จะช่วยให้คุณสามารถเรียนรู้ด้วยตนเองที่บ้าน หรือที่ไหนก็ตามแล้วแต่คุณจะสะดวก นอกจากนี้ต้องขอบคุณอุปกรณ์มือถือที่ทำให้คุณสามารถ “เรียนได้ทุกที่”

ในหัวข้อนี้จะนำเสนอ db2university.com เป็นเว็บไซต์การศึกษาที่มีการใช้เทคโนโลยีต่าง ๆ ที่ได้อธิบายไว้ในบทนี้ในการพัฒนา โดยรูปที่ 10.7 แสดงให้เห็นถึงการใช้งาน db2university



รูปที่ 10.7 เว็บไซต์ db2university.com

10.4.1 ระบบจัดการหลักสูตรแบบไม่เสียค่าใช้จ่ายด้วย Moodle

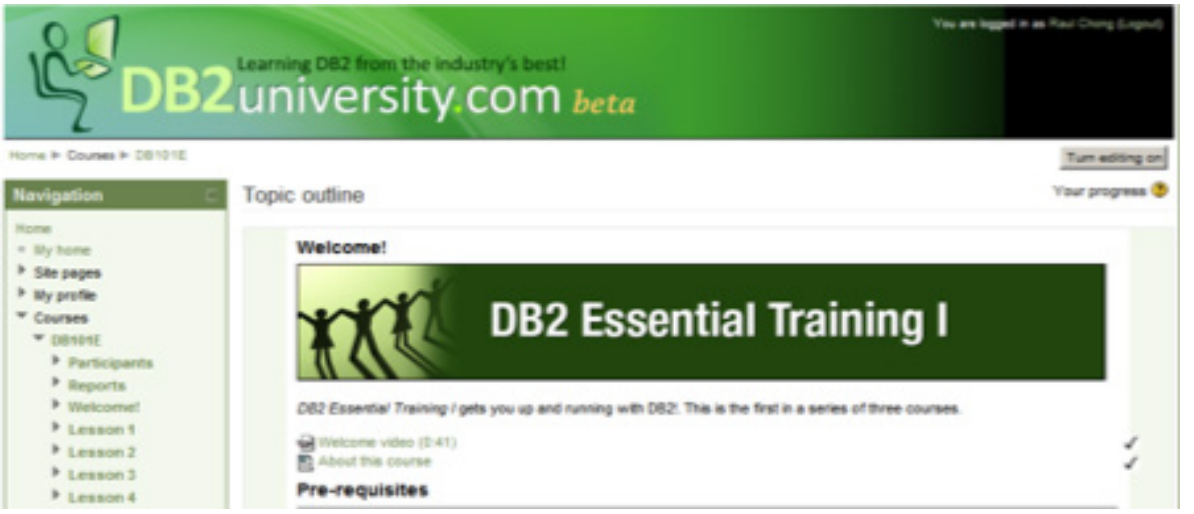
มีแหล่งข้อมูลความรู้มากมายที่ใช้ในการเรียนรู้ด้วยตนเอง เช่น การแพร่ภาพวิดีโอ และหนังสือออนไลน์ อย่างไรก็ตามพวกเขามักจะไม่จัดเรียงหมวดหมู่ให้ผู้เรียนได้ทำการเข้าถึงได้ง่าย ๆ โดยข้อมูลบางอย่างอาจจะล้าสมัยไปแล้ว แต่ระบบจัดการหลักสูตร จะอนุญาตให้ผู้สอนสามารถสร้าง และจัดเรียงเนื้อหาเพื่อให้ง่ายต่อการเข้าถึง ที่ db2university.com เราใช้ Moodle เป็นระบบจัดการหลักสูตรซึ่งไม่มีค่าใช้จ่ายใด ๆ (CMS)

Moodle พัฒนาด้วยภาษา PHP ซึ่งเป็น CMS ที่ไม่เสียค่าใช้จ่ายที่ได้รับความนิยมมากที่สุดระบบหนึ่งในท้องตลาด ซึ่งถูกพัฒนาขึ้นโดย Martin Dougiamas จากมหาวิทยาลัย Curtin University of Technology ในกรุง Perth ประเทศออสเตรเลีย ได้พัฒนาผลิตภัณฑ์รุ่นแรกเมื่อปีค.ศ. 2001 และในวันนี้มีผู้ลงทะเบียนใช้งานมากกว่า 52,000 ไซต์ และมากกว่า 950,000 คนลงทะเบียนใช้งาน โดยใช้งานอยู่มากกว่า 175 ประเทศทั่วโลก ข้อมูลจาก moodle.org

ส่วนในเว็บไซต์ของเราได้เริ่มใช้งาน Moodle ในปีค.ศ. 2009 ที่ผ่านมา โดยนักศึกษาจาก California State University, Long Beach (CSULB) ได้ถูกกำหนดให้เป็นส่วนหนึ่งในหลักสูตรของตน โดยใช้ผลิตภัณฑ์ Moodle รุ่น 1.9.6 ทำงานกับฐานข้อมูล DB2 Express-C สามารถเข้าไปศึกษาได้ที่ (รวมถึงดาวน์โหลด patch) http://moodle.org/mod/data/view.php?d=13&rid=3100&filter=1

ในปี ค.ศ. 2010 เราเริ่มใช้งาน Moodle 2.0 และใช้ DB2 Express-C โดย Moodle 2.0 เป็นผลิตภัณฑ์ที่ทำการเปลี่ยนแปลงจากผลิตภัณฑ์รุ่นก่อนหน้านี้ และเปลี่ยนวิธีในการเชื่อมต่อกับฐานข้อมูล โดยผลิตภัณฑ์รุ่น Moodle 2.0 Release Candidate 1 จะเป็นรุ่นที่สามารถใช้งานกับ DB2 Express-C และเป็นส่วนหนึ่งที่ทำงานบน db2univertsity.com เมื่อผลิตภัณฑ์รุ่น 2.0 ออกสู่ท้องตลาดเราจะทำการปรับปรุง db2university.com และเข้าไปมีส่วนร่วมในชุมชนของ Moodle

รูปที่ 10.8 แสดงถึงหลักสูตรต่าง ๆ ที่สามารถเรียนรู้ได้จาก db2university.com สามารถใช้งานหน้าเว็บไซต์นี้ได้โดยการคลิกที่หัวข้อ Learn ในหน้าโฮมเพจ โดยหัวข้อ Learn จะแสดงหัวข้อของเนื้อหาทั้งหมดที่มีอยู่ใน db2university

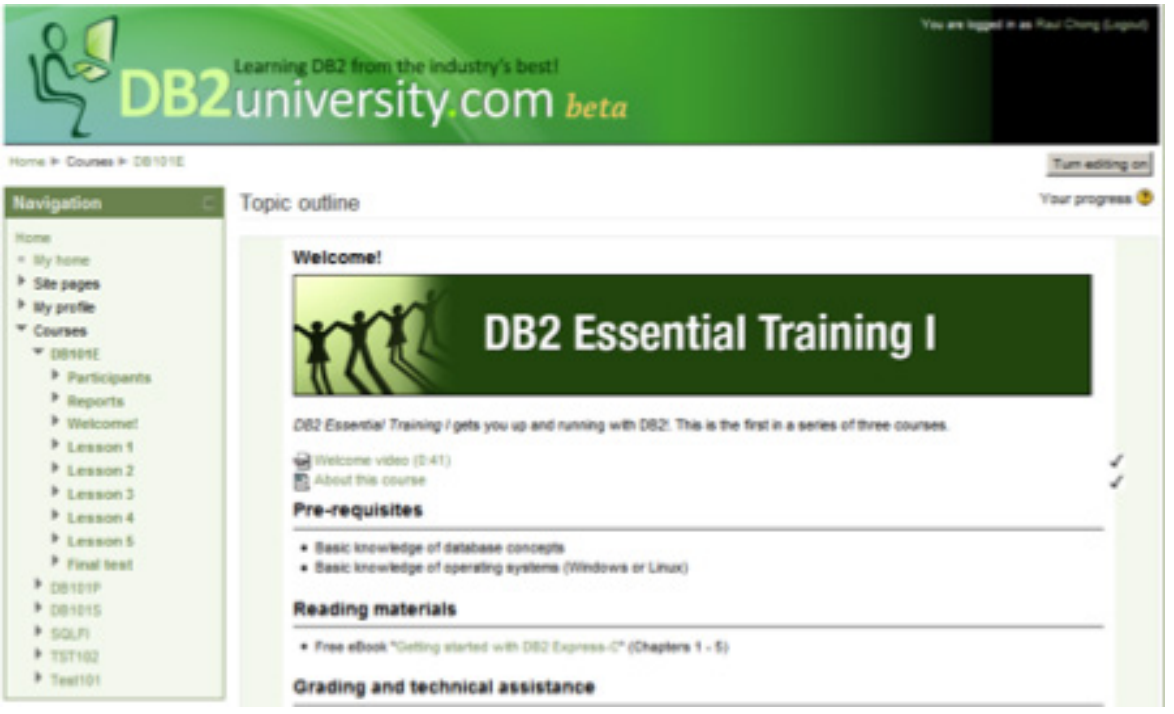


รูปที่ 10.8 แสดงถึงหลักสูตรต่าง ๆ ที่สามารถเรียนรู้ได้จาก db2university.com

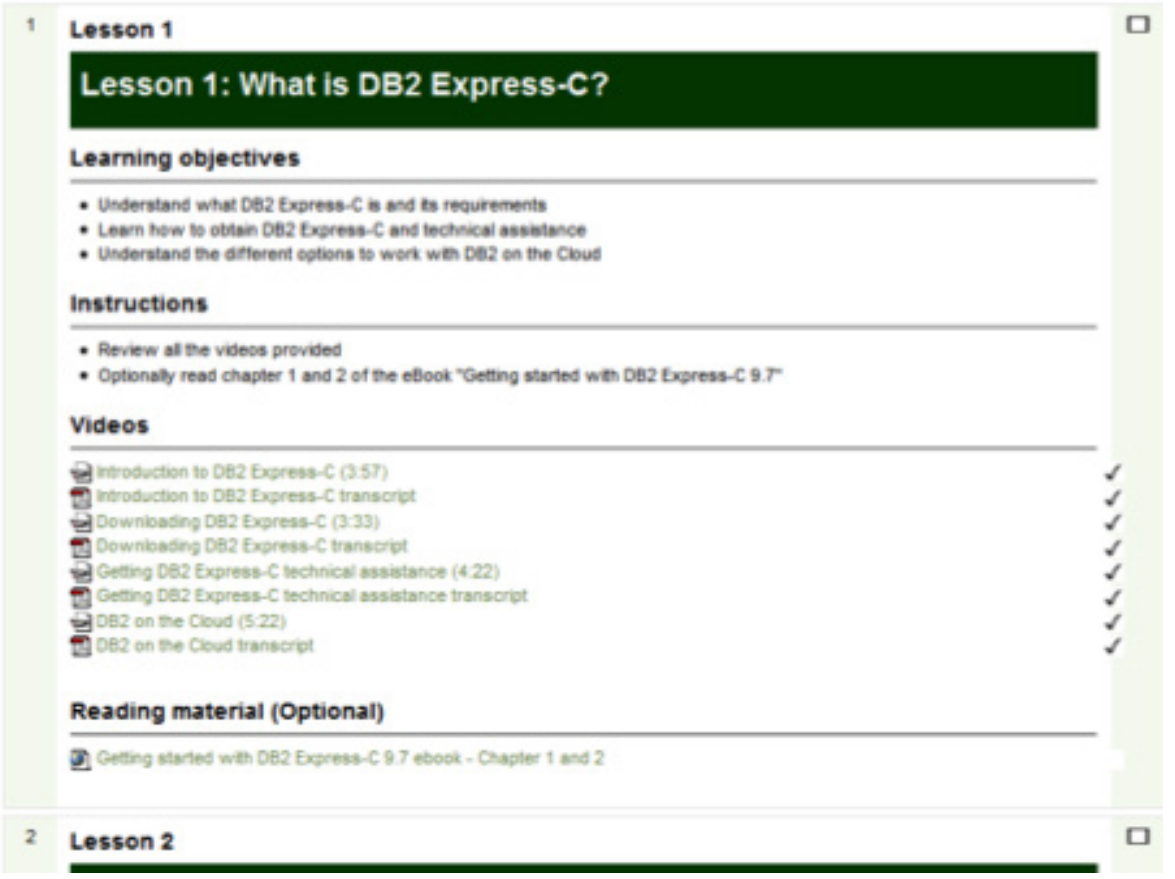
หลักสูตรใน db2university.com ไม่มีแค่เพียงเนื้อหาที่เกี่ยวกับ DB2 เท่านั้น แต่จะมีเนื้อหาที่เกี่ยวข้องกับเรื่องอื่น ๆ เช่น PHP หรือ Ruby on Rails เป็นต้น แต่ในเนื้อหาเหล่านั้นจะเป็นหลักสูตรที่มีการเชื่อมต่อกับฐานข้อมูลโดยใช้ DB2 ผู้สอนสำหรับหลักสูตรต่าง ๆ จะเป็นศาสตราจารย์ในมหาวิทยาลัย มืออาชีพ ลูกจ้างของบริษัท ไอบีเอ็ม และนักศึกษา

ในเว็บไซต์จะมีหลักสูตรอยู่สองประเภทได้แก่: หลักสูตรที่ไม่เสียค่าใช้จ่าย และหลักสูตรที่ต้องเสียค่าใช้จ่าย สำหรับหลักสูตรที่ต้องจ่ายเงินเราจะใช้ปลั๊กอินของ Paypal ที่มีอยู่ใน Moodle ในการรับชำระเงิน

รูปที่ 10.9 และ 10.10 เป็นรูปแสดงรายละเอียดของบริการของหลักสูตร “DB2 Essential Training I” ในหลักสูตรนี้จะเป็นตัวอย่างว่าเราสามารถทำอะไรได้บ้างใน Moodle หมายเหตุ: ในหลักสูตรนี้มีวีดิโอ, transcripts (ในรูปแบบ PDF ไฟล์) และลิงค์ที่เชื่อมต่อไปยังหนังสืออิเล็คโทรนิกส์ที่อ้างอิงถึงอยู่มากมาย ที่ไม่ได้แสดงอยู่ในรูปคือลิงค์ของฟอรั่ม หน้าเพจ และบททดสอบ



รูปที่ 10.9 หัวข้อต่าง ๆ ในหลักสูตร DB2 Essential Training I



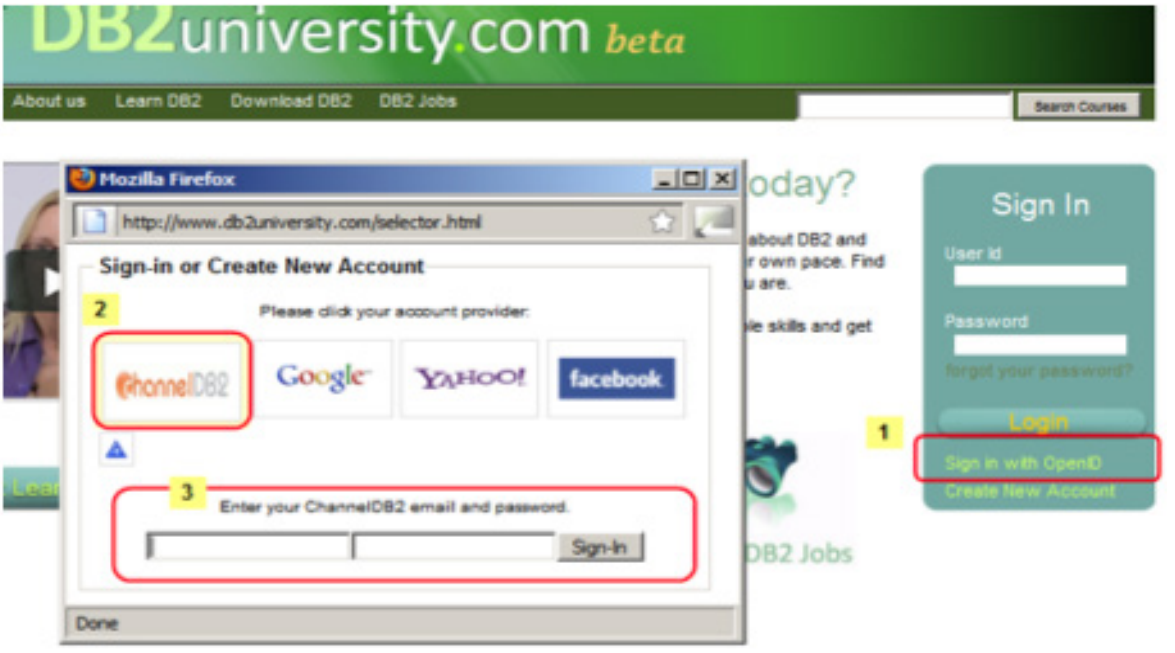
รูปที่ 10.10 หัวข้อต่าง ๆ ในหลักสูตร DB2 Essential Training I (ต่อ)

บทเรียนที่ปรากฏอยู่ใน IBM_developerWorks® จะสอนวิธีในการสร้างหลักสูตรใน db2university.com

หมายเหตุ:
ถ้าต้องการเรียนรู้เพิ่มเติมที่กับการพัฒนาซอฟต์แวร์แบบไม่เสียค่าใช้จ่าย สามารถอ้างอิงได้โดยไม่เสียค่าใช้จ่ายได้ที่ Getting started with open source development ซึ่งเป็นส่วนหนึ่งในชุดหนังสือของ DB2

10.4.2 การอนุญาตใช้งาน openID ในการเข้าสู่ระบบ

จากงานวิจัยแสดงว่าเว็บไซต์ที่จัดทำการลงทะเบียนไว้ยุ่งยาก และใช้ขั้นตอนมากมายทำให้ผู้ใช้ไม่สนใจที่จะสมัครใช้งาน ดังนั้นการสมัครเข้าใช้งาน db2university.com เราจะใช้งาน openID ในการเข้าสู่ระบบ และลงทะเบียนในการใช้งานเว็บไซต์นี้ ซึ่งบุคคลที่มี Facebook, Google, Yahoo, AOL, ChannelDB2 และบัญชีอื่น ๆ ไม่จำเป็นต้องกรอกรายละเอียดในการลงทะเบียน โดย db2university จะใช้ข้อมูลที่ผู้ใช้งานเปิดเผยร่วมกันจากบัญชีที่กล่าวมาข้างต้น รูป 10.11 ขั้นตอนการลงทะเบียน



รูปที่ 10.11 ขั้นตอนการเข้าสู่ระบบด้วย openID

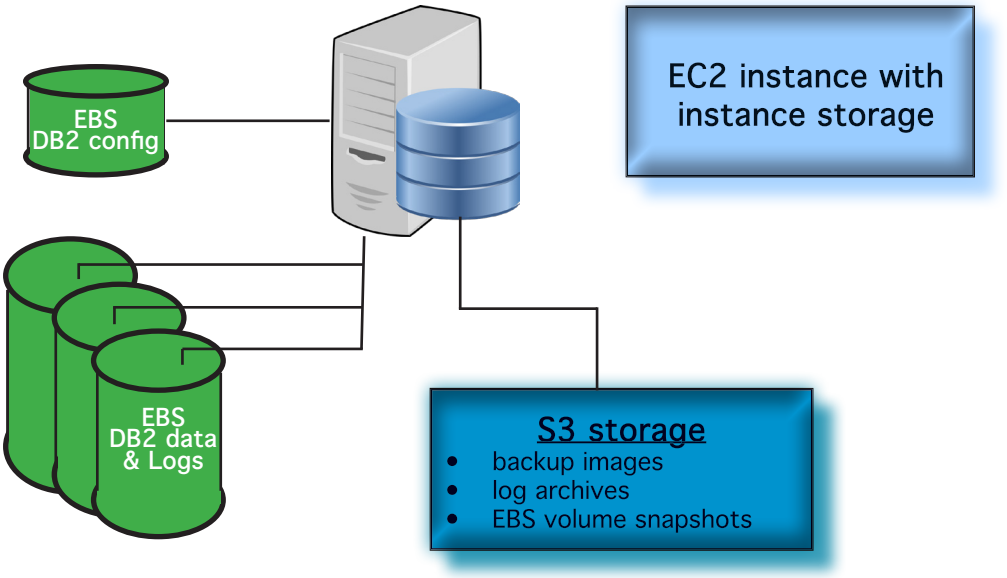
ขั้นแรกทำการคลิกที่ลิงค์ Sign in with OpenID จากนั้นจะปรากฏหน้าต่างให้เลือกว่าจะใช้บัญชีของผู้ให้บริการใด ให้คุณทำการเลือกบัญชีของผู้ให้บริการ ในตัวอย่างจะเป็นการเลือก ChannelDB2 จากนั้นทำการกรอกชื่อผู้ใช้งาน/อีเมล์ และรหัสผ่านให้ถูกต้อง หลังจากขั้นตอนนี้คุณจะถูกผู้ให้บริการที่คุณเลือกถามว่าคุณต้องการเปิดเผยข้อมูลส่วนตัวร่วมกับเว็บไซต์นี้หรือไม่ ถ้าคุณยอมรับระบบจะทำการลงทะเบียนให้อัตโนมัติ และข้อมูลส่วนตัวของคุณก็จะถูกใช้งานร่วมกับเว็บไซต์นี้ และทุกครั้งที่คุณต้องการเข้าสู่ระบบของ db2university.com คุณจะต้องทำตามขั้นตอนนี้ทุกครั้ง
และนี้เป็นบทความใน IBM developerWorks ที่อธิบายถึงการให้บริการของ openID

10.4.3 การทำงานบนระบบ Cloud ของ Amazon

db2university.com เป็นเครื่องแม่ข่ายที่อยู่บนระบบ Cloud ของ Amazon ซึ่งอนุญาตให้เราสามารถใช้ความสามารถของการประมวลผลแบบ Cloud ต่าง ๆ ได้ รวมถึงการติดตั้ง DB2 HADR สำหรับการสำรองระบบบน Cloud ซึ่งทรัพยากรที่เราใช้ใน AWS จะมีดังนี้:

- EC2 standard large instance (64-บิต, หน่วยประมวลผล 4 แกน, หน่วยความจำหลัก 7.5 GB, หน่วยความจำสำรอง 850 GB)
- EBS ขนาด 50 GB
- S3 buckets จำนวน 3 และมีขนาด 1 GB
- CloudFront

มีการใช้งาน EC2 ถึงสามเครื่องในภาคตะวันออกของสหรัฐอเมริกา หนึ่งในนั้นจะถูกใช้เป็น Moodle และเว็บเซิร์ฟเวอร์ และอีกหนึ่งจะถูกใช้งานเป็นเซิร์ฟเวอร์ของฐานข้อมูล DB2 (DB2 HADR เครื่องหลัก) โดย DB2 HADR เครื่องสำรองจะเป็นเครื่องที่สาม นี้คือทรัพยากรที่ใช้สำหรับภาคตะวันออกของสหรัฐอเมริกา
เงื่อนไขในการจัดเก็บข้อมูล เราใช้อินสแตนซ์เก็บ S3 และ EBS รูปที่ 10.12 แสดงรายละเอียด



รูปที่ 10.12 การจัดเก็บข้อมูลของ DB2 บนระบบ Cloud ของ Amazon

จากรูปภาพที่แสดงใน 10.3

- EBS ไดรฟ์จะถูกใช้ในการเก็บข้อมูล DB2 และบันทึกการกู้คืนของ DB2 เราใช้หน่วยความจำสำรอง 50 GB โดยการทำ RAID
- และ EBS ไดรฟ์ที่มีขนาด 2 GB เพื่อใช้เก็บข้อมูลโค๊ดของ DB2 และไฟล์ในการกำหนดค่าของ DB2
- หน่วยความจำสำรองของอินสแตนซ์จะถูกใช้เพื่อจัดเก็บอิมเมจสำรองของ DB2 ไว้ชั่วคราวก่อนที่จะย้ายไปเก็บไว้อย่างถาวรในหน่วยความจำสำรอง S3
- หน่วยความจำสำรอง S3 จะถูกใช้เก็บข้อมูลสำรอง และไฟล์ที่ใช้กำหนดค่า โดย EBS volume snapshots จะถูกโอนย้ายมาจัดเก็บไว้ใน S3 นี้

AWS CloudFront จะอนุญาตให้ผู้ใช้งานสามารถดาวน์โหลดวีดิโอของหลักสูตร และวัตถุดิบต่าง ๆ จากเครื่องเซิร์ฟเวอร์ของอเมซอนที่ชื่อว่า “edge” ซึ่งอยู่ใกล้กับบริเวณที่ผู้ในงานอาศัยอยู่ ไฟล์ที่ดาวน์โหลดจะถูกคัดลอกไปฝากข้อมูลไว้ยัง S3 bucket ในครั้งแรก และจะถูกคัดลอกไปยัง edge เซิร์ฟเวอร์ในภายหลัง ซึ่งในแงของซอฟต์แวร์ที่ใช้ในการติดตั้งระบบ

- ใช้ระบบปฏิบัติการ Ubuntu Linux 10.04.1 LTS
- DB2 Express 9.7.2 สำหรับ Linux 64-บิต
- Moodle 2.0 Release Candidate 1
- PHP 5.3.3

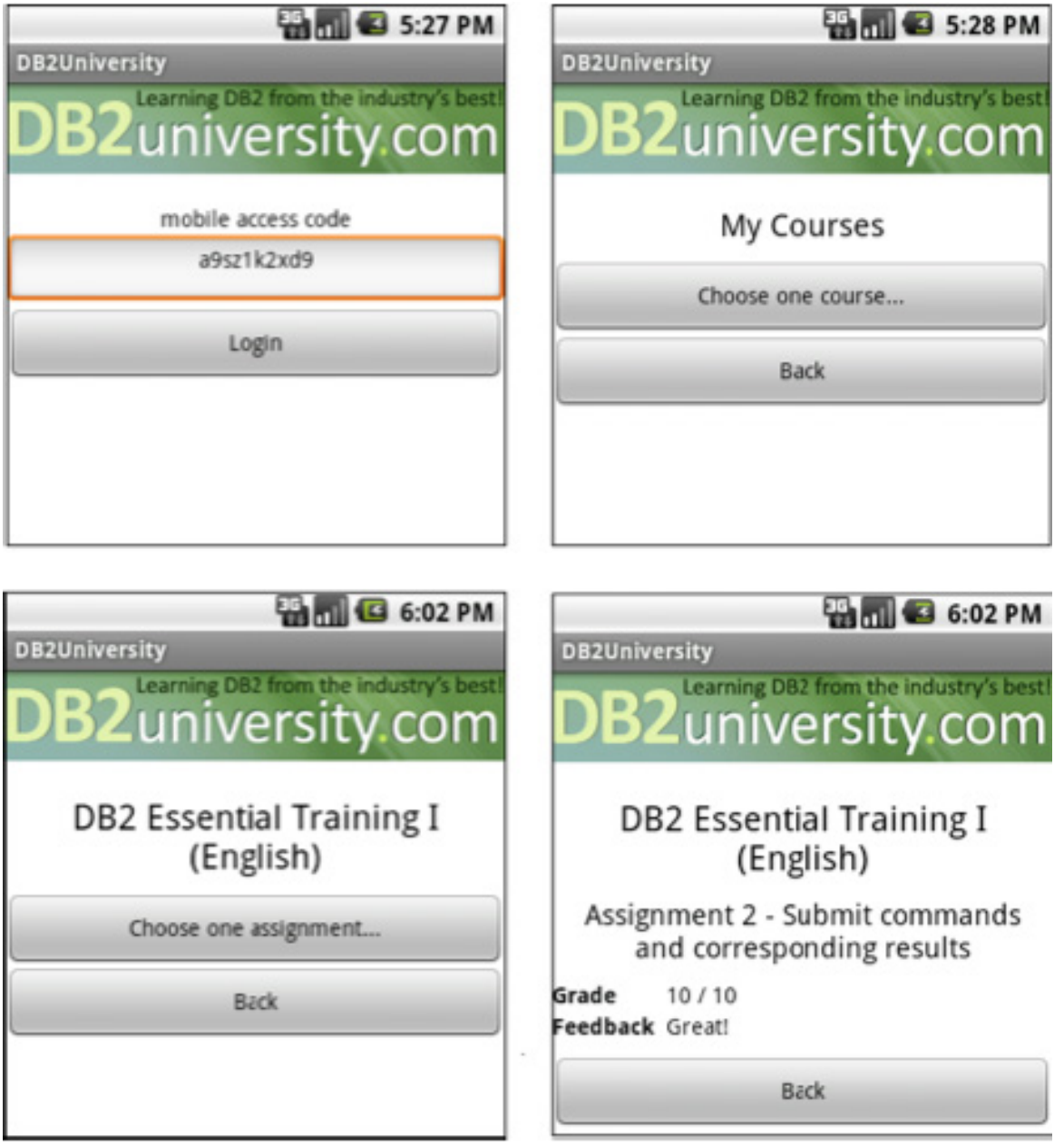
DB2 Express จะถูกใช้งานแทน DB2 Express-C เนื่องจากเราต้องการใช้งานคุณสมบัติ HADR ของ DB2 ซึ่งจะสามารถใช้งานได้ในผลิตภัณฑ์รุ่น DB2 Express
ใช้งาน RightScale ในการจัดการทรัพยากรของ AWS ซึ่ง RightScale เป็นพันธมิตรของ ไอบีเอ็ม และซอฟต์แวร์ทำให้สามารถในการสร้างสถาพแวดล้อมได้อย่างรวดเร็วโดยการใช้แม่แบบ และสคริปต์
ในบทความที่อยู่ใน IBM_developerWorks จะอธิบายรายละเอียดต่าง ๆ ของ AWS ที่ถูกใช้งานสำหรับ db2university

10.4.4 การใช้โทรศัพท์มือถือ Android เพื่อรับข้อมูลของหลักสูตร

Moodle อนุญาตให้นักเรียนสามารถมองเห็นงานที่ถูกมอบหมาย และทำบททดสอบได้บนเว็บไซต์ แต่อย่างไรก็ตามเราได้พัฒนาโปรแกรมบนมือถือสำหรับระบบปฏิบัติการ Android โดยใช้ App Inventor

เป็นโปรแกรมง่าย ๆ อย่างที่แสดงในรูป 10.13 โดยขั้นแรกจะต้องกรอกรหัสในการเข้าใช้งานบนมือถือที่ซึ่งเป็นรหัสที่ไม่ซ้ำกันโดยจะได้จากโปรไฟล์ของตนใน Moodle จากนั้นทำแล้วเลือกหลักสูตร และแสดงหลักสูตรที่ทำการเลือกทั้งหมด

ในการใช้งานโปรแกรมนี้ในการทำงานในฝั่งเครื่องลูกข่ายจะต้องใช้งาน App Inventor และฝั่งเครื่องแม่ข่าย (ใช้งาน Web Service ที่ถูกสร้างขึ้นโดย Ruby/Sinatra) สามารถเข้าไปดูรายละเอียดได้ที่ IBM_developerWorks



รูปที่ 10.13 การเข้าใช้งาน db2university ของผ่าน App Inventor สำหรับ Android

10.5 บทสรุป

ในบทนี้ได้กล่าวถึงแนวโน้มของเทคโนโลยีที่จะปรากฏขึ้นภายในปีค.ศ. 2015 และจะเป็นหัวข้อที่สำคัญในการใช้ฐานข้อมูลในเทคโนโลยีเหล่านี้ โดยการประมวลผลแบบ Cloud จะเป็นเรื่องที่สำคัญที่สุด และเป็นหัวข้อที่ถูกพูดถึงมากที่สุดในเรื่องของไอที โดยเป็นการทำวิธีจัดการทรัพยากรแบบใหม่ที่อนุญาตให้องค์กรต่าง ๆ และบุคคลต่าง ๆ สามารถเข้าถึงทรัพยากรเท่าไหร่ก็ได้ของระบบคอมพิวเตอร์ได้ทุกที่ตามความต้องการ และเสียค่าใช้จ่ายเท่าที่คุณใช้งานเท่านั้น โดยเนื้อหาส่วนหลังของบทนี้เราจะพูดถึงเรื่องที่เกี่ยวกับโปรแกรมบนอุปกรณ์มือถือที่กำลังมีการเจริญเติบโตเพิ่มมากขึ้น โดยบทนี้จะแนะนำคุณเกี่ยวกับแพลตฟอร์มต่าง ๆ ในการพัฒนาบนอุปกรณ์มือถือ

ในบทนี้จะสรุปสั้น ๆ เกี่ยวกับ business intelligence และโปรแกรมที่เกี่ยวข้อง โดยองค์กรต้องการเครื่องมือที่มีความฉลาดในการประมวลผลข้อมูลไว้สำหรับช่วยในการตัดสินใจในธุรกิจขององค์กร ในขณะเดียวกันพวกเขาก็ไม่ต้องการที่จะลงทุนในงบประมาณจำนวนมากเกี่ยวกับเรื่องนี้ ซึ่งคลังข้อมูล และ business intelligence อย่างเช่น IBM Smart Analytics สามารถแก้ปัญหานี้ได้

สุดท้ายบทนี้กล่าวถึง db2university.com ที่เป็นกรณีศึกษาของเทคโนโลยีทั้งหมดที่กล่าวมาในบทนี้

ภาคผนวก ก – เฉลยคำถามท้ายบท

เฉลยบทที่ 1

1. ฐานข้อมูลเป็นสิ่งที่ใช้จัดเก็บข้อมูล ออกแบบมาเพื่อสนับสนุนการจัดเก็บ การเรียกใช้ข้อมูลและการบำรุงรักษาข้อมูล ให้มีประสิทธิภาพ
2. ระบบจัดการฐานข้อมูลหรือเรียกง่ายๆว่า DBMS คือ กลุ่มซอฟต์แวร์ที่เป็นเครื่องมือที่ทำหน้าที่ในการควบคุมการเข้าถึง การจัดระเบียบ การจัดเก็บ การจัดการ การเรียกใช้ และรักษาข้อมูล ในฐานข้อมูล
3. รูปแบบข้อมูลจะมีลักษณะเป็นนามธรรมที่อยู่ในรูปแบบของเอนทิตี้โดยมีคุณสมบัติที่ประกอบด้วย ความสัมพันธ์และการดำเนินงานที่สามารถดำเนินการในตัวเอนทิตี้เอง ในทางกลับกันในแบบจำลองข้อมูลส่วนมากจะถูกกำหนดให้อยู่ในระดับที่เป็นรูปธรรมและประกอบด้วยรายละเอียดมากขึ้น รูปแบบข้อมูลเฉพาะเจาะจงมากกว่ารูปแบบข้อมูล และใช้เป็นเอกสารหลักของระบบฐานข้อมูล
4. ประโยชน์หลักคือความเป็นอิสระของข้อมูล (Data Inpendence) (ในที่นี้ไม่เจาะจงระดับการจัดเก็บข้อมูลในลักษณะทางกายภาพ)
5. ติดตั้งและทดสอบระบบการจัดการฐานข้อมูล (DBMS) เวอร์ชันใหม่ และกำหนดสิทธิ์ในการเข้าถึงข้อมูลของผู้ใช้งาน
6. ตอบข้อ A. แบบจำลอง pureXML
7. ตอบข้อ C. ประสิทธิภาพการทำงาน
8. ตอบข้อ E. ไม่มีคำตอบที่ถูกต้อง
9. ตอบข้อ D. การรวมกัน
10. ตอบข้อ B. แม้ว่า pureXML จะเป็นสิ่งที่มีความแตกต่างอย่างชัดเจนกับ DB2 เมื่อเทียบกับ RDBMS อื่นๆโดยทั่วไป สิ่งสำคัญที่ cloud ที่เป็นกุญแจของความแตกต่างคือคุณลักษณะ Database Partitioning Feature (DPF) ของ DB2 ซึ่งช่วยให้ขยายขีดความสามารถบน cloud ที่มีตัวอย่างมาตรฐานให้

เฉลยบทที่ 2

1. ข้อจำกัดความสมบูรณ์ของเอนทิตี้ สำหรับหมายเลขของตัวแทนจำหน่าย เราจะต้องกำหนดข้อจำกัดของค่าที่ไม่ซ้ำกันและข้อจำกัดไม่ให้มีค่าว่าง สำหรับชื่อตัวแทนจำหน่าย เราจะต้องกำหนด ข้อจำกัดไม่ให้มีค่าว่าง และ ส่วนลดของตัวแทนจำหน่าย เราจะต้องกำหนดข้อจำกัดค่าที่ต้องตรวจสอบข้อมูลว่าต้องอยู่ในช่วงที่ต้องการและข้อจำกัดไม่ให้มีค่าว่าง
2. Intersection: $R1 \cap R2 = R1-(R1-R2)$.

   Join: $R1_{join\_condition} \blacktriangleright\blacktriangleleft R2 = \sigma_{join\_condition} (R1 X R2)$

   Division for relation R1(A,B) and R2(A):

   $R1 \div R2 = \pi_B (R1) - \pi_B ((R2 X \pi_B (R1)) - R1)$
3. $\pi_{Name}(\sigma_{Address=}$'New York' AND Discount>0.05(R))
4. RANGE OF SUPPLIERS IS SUPPLIERS.Name WHERE (SUPPLIERS.Address ='New York' AND SUPPLIERS.Discount>0.05)
5. NameX WHERE ∃ (DiscountX >0.05 AND SUPPLIERS (Name:NameX, Discount:DiscountX, Address:'New York')
6. ตอบข้อ A. แบบจำลองฐานข้อมูลที่สร้างจากข้อมูลจริง

   ตอบข้อ C. คุณลักษณะของข้อมูล(data characteristic)
7. ตอบข้อ B. ค่าทูเปิลในรีเลชันต้องเป็นค่าที่ไม่ซ้ำ

   ตอบข้อ C. แอตทริบิวต์เป็นค่าที่แบ่งแยกไม่ได้
8. ตอบข้อ B. อินสแตนซ์ (instance)

   ตอบข้อ D. ดีกรี (degree)
9. ตอบข้อ A. คีย์หลักคือคีย์แคนดิเดตคีย์
10. ตอบข้อ D. Cascade

เฉลยบทที่ 3

1. ตอบข้อ D.
2. ตอบข้อ A.
3. ตอบข้อ C.
4. ตอบข้อ A.
5. ตอบข้อ B.
6. ตอบข้อ D.
7. ตอบข้อ B.
8. ตอบข้อ D.
9. ตอบข้อ C.
10. ตอบข้อ B.

เฉลยบทที่ 4

1. ดูตัวอย่างหัวข้อ 4.6.1 Lossless และ Lossy Decompositions
2. ตอบข้อ C.
3. ตอบข้อ B.
4. ตอบข้อ E.
5. ตอบข้อ B.
6. ตอบข้อ C.

เฉลยบทที่ 5

1. ตอบข้อ D.
2. ตอบข้อ D.
3. ตอบข้อ A, C.
4. ตอบข้อ D.
5. ตอบข้อ D.
6. ตอบข้อ A.
7. ตอบข้อ B.
8. ตอบข้อ D.
9. ตอบข้อ D.
10. ตอบข้อ B.

เฉลยบทที่ 6

1. ตอบข้อ C.
2. ตอบข้อ C.
3. ตอบข้อ E.
4. ตอบข้อ C.
5. ตอบข้อ C.
6. ตอบข้อ D.
7. ตอบข้อ D.
8. ตอบข้อ D.
9. ตอบข้อ A.
10. ตอบข้อ A.

เฉลยบทที่ 7

1. ตอบข้อ B.
2. ตอบข้อ A.
3. ตอบข้อ B.
4. ตอบข้อ A.
5. ตอบข้อ B.
6. ตอบข้อ B.
7. ตอบข้อ B.
8. ตอบข้อ A.
9. ตอบข้อ B.
10. ตอบข้อ B.

เฉลยบทที่ 8

1. ตอบข้อ D.
2. ตอบข้อ B.
3. ตอบข้อ B.
4. ตอบข้อ D.
5. ตอบข้อ A.
6. ตอบข้อ A.
7. ตอบข้อ C.
8. ตอบข้อ A.
9. ตอบข้อ A.

เฉลยบทที่ 9

1. การพัฒนาของเทคโนโลยีสารสนเทศที่นำไปสู่ข้อมูลจำนวนมากและขนาดใหญ่ที่เก็บรวบรวมโดยองค์กร ข้อมูลเหล่านี้ได้เก็บรวบรวมกิจกรรมทั้งหมดในองค์กรและสามารถใช้เป็นข้อมูลอ้างอิงที่ช่วยในการตัดสินใจที่สำคัญของการดำเนินธุรกิจ การรักษาความปลอดภัยของข้อมูลเพื่อให้เป็นความลับและสามารถใช้งานข้อมูลได้อย่างต่อเนื่องจึงเป็นวัตถุประสงค์ของมาตรการรักษาความปลอดภัย
2. ไม่ , การรักษาความปลอดภัยฐานข้อมูลเพียงอย่างเดียว อาจจะไม่ทำให้แน่ใจได้ว่าฐานข้อมูลมีความปลอดภัย แต่ละส่วนที่ประกอบขึ้นเป็นระบบของฐานข้อมูลทั้งหมดจะต้องมีความปลอดภัยทั้งในส่วนของ : ฐานขอมูล เครือขาย ระบบปฏิบัติการ อุปกรณ์ที่ฐานข้อมูลอยู่ในลักษณะทางกายภาพ และบุคคลใดๆ ที่มีโอกาสในการเข้าถึงระบบ
3. ในแผนการรักษาความปลอดภัย ควรให้ความสนใจเพิ่มเติมในการคุกคามเหล่านี้: การโจรกรรมและการฉ้อโกง การสูญหายของข้อมูลส่วนบุคคลหรือข้อมูลที่เป็นความลับ การสูญเสียความสมบูรณ์ของข้อมูล การสูญเสียความพร้อมใช้งานและการสูญเสียข้อมูลโดยไม่ตั้งใจ
4. การควบคุมการเข้าถึงแบบ Discretionary Control คือ วิธีการที่ช่วยให้ผู้ใช้เข้าถึง และ การดำเนินการต่าง ๆ บนข้อมูลโดยใช้สิทธิการเข้าถึงของตนหรือสิทธิพิเศษของผู้ใช้ในแต่ละคน
5. DB2 มีการให้สิทธิโดยการกำหนดได้ 3 รูปแบบ คือ: สิทธิหรืออำนาจในระดับการจัดการกับฐานข้อมูล สิทธิในการเข้าถึงข้อมูล และ Label-Based Access Control (LBAC) ของแต่ละบุคคล

6. สิทธิ เป็นอำนาจหน้าที่ที่กำหนดให้กับผู้ใช้แต่ละบุคคล หรือกลุ่มผู้ใช้ ซึ่งอนุญาตให้ผู้ใช้แต่ละบุคคลหรือกลุ่มผู้ใช้ทำกิจกรรมที่แตกต่างกันบนออบเจ็กต์ของฐานข้อมูล
7. Trusted Context สร้างความสัมพันธ์ที่น่าเชื่อถือระหว่าง DB2 และองค์ประกอบภายนอก อย่างเช่นเว็บเซิร์ฟเวอร์หรือแอพพลิเคชันเซิร์ฟเวอร์ที่เชื่อถือได้ ความสัมพันธ์นี้จะขึ้นอยู่กับการให้สิทธิ์ระบบ, ไอพี แอดเดรส และข้อมูลที่ได้ทำการเข้ารหัส ซึ่งกำหนดได้โดย Trusted Context ของฐานข้อมูล
8. สำหรับวิว เป็นตารางเสมือนซึ่งเป็นผลลัพธ์แบบไดนามิกของการดำเนินการความสัมพันธ์หนึ่งหรือมากกว่าหนึ่งความสัมพันธ์กับหนึ่งหรือมากกว่าหนึ่งตารางพื้นฐาน วิวสามารถสร้างขึ้นเพื่อนำเสนอข้อมูลเฉพาะที่ผู้ใช้มีสิทธิ์การเข้าถึงและสามารถป้องกันไม่ให้ดูข้อมูลอื่น ๆ ที่เป็นข้อมูลความลับ
9. การควบคุมความถูกต้องของข้อมูลเพื่อมุ่งหวังป้องกันการเข้าถึงข้อมูลจากผู้ที่ไม่ได้รับอนุญาตในใช้งานและปรับปรุงโดยการจำกัดสิทธิ์การดำเนินงานที่สามารถจัดการกับข้อมูลได้ โดยสามารถใช้การกำหนดขอบเขต, การยืนยันสิทธิ์ และทริกเกอร์
10. นโยบายการรักษาความปลอดภัยและขั้นตอนการดำเนินการส่วนใหญ่ คือ การควบคุมบุคลากรรายบุคคล และการควบคุมการเข้าถึงทางกายภาพ

ข

ภาคผนวก ข – การติดตั้งและการทำงานกับ DB2

รายละเอียดของเนื้อหาภายในภาคผนวกเป็นพื้นฐานสำหรับการเรียนรู้เกี่ยวกับ DB2 ภาคผนวกนี้จะช่วยให้คุณได้เรียนรู้การติดตั้งและการทำงานกับ DB2 อย่างรวดเร็ว และง่ายดาย

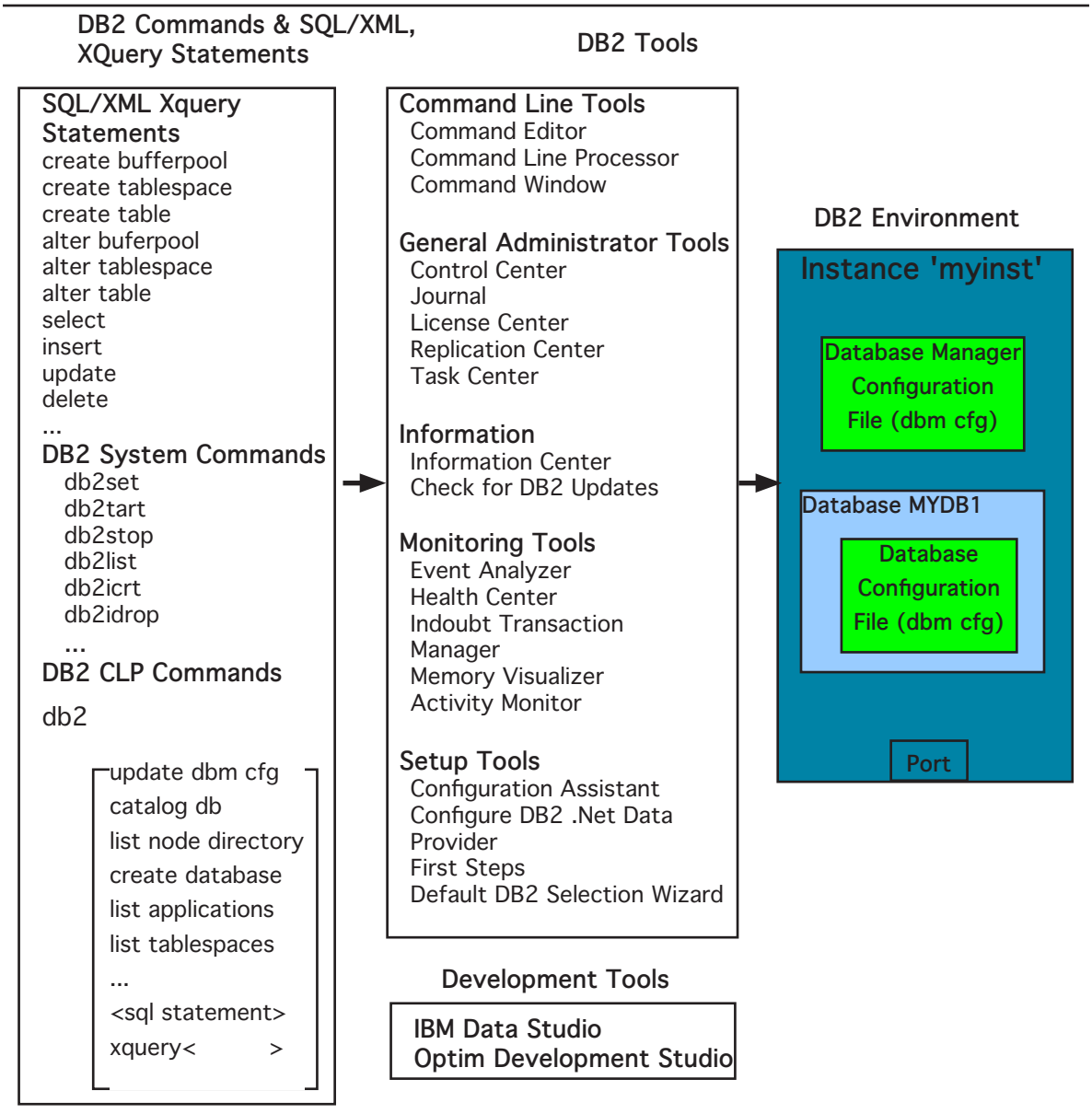
ในภาคผนวกนี้ คุณจะได้เรียนรู้เกี่ยวกับ:

- บรรจุภัณฑ์ของ DB2
- การติดตั้ง DB2
- เครื่องมือของ DB2
- สภาพแวดล้อมของ DB2
- การตั้งค่าคอนฟิกใน DB2
- การเชื่อมต่อกับฐานข้อมูล
- โปรแกรมตัวอย่างพื้นฐาน
- เอกสารประกอบ DB2

> หมายเหตุ:
>
> สำหรับข้อมูลเพิ่มเติมเกี่ยวกับ DB2 อ้างอิงจากหนังสืออิเล็กทรอนิกส์ฟรี (free e-book) ที่ชื่อว่า Getting Started with DB2 Express-C ที่เป็นส่วนหนึ่งของหนังสือชุดนี้

ข.1 ภาพรวมของ DB2

DB2 เป็นซอฟต์แวร์ระบบฐานข้อมูลที่ช่วยให้คุณสามารถจัดเก็บได้อย่างปลอดภัย และเรียกใช้ข้อมูล DB2 จะใช้คำสั่ง XQuery และคำสั่ง SQL เพื่อใช้ในการติดต่อกับเซิร์ฟเวอร์ DB2 ซึ่งอนุญาตให้ผู้ใช้สามารถสร้างอ็อบเจ็กต์ฐานข้อมูล และจัดการกับข้อมูลในสภาพแวดล้อมที่ปลอดภัยและสามารถใช้เครื่องมือต่าง ๆ ในการป้อนคำสั่ง ดังแสดงในรูป ข.1 รูปนี้แสดงภาพรวมของ DB2 ที่ได้ดึงออกมาจากหนังสือ Getting Started with DB2 Express-C e-book.



รูป ข.1 แสดงภาพรวมของ DB2

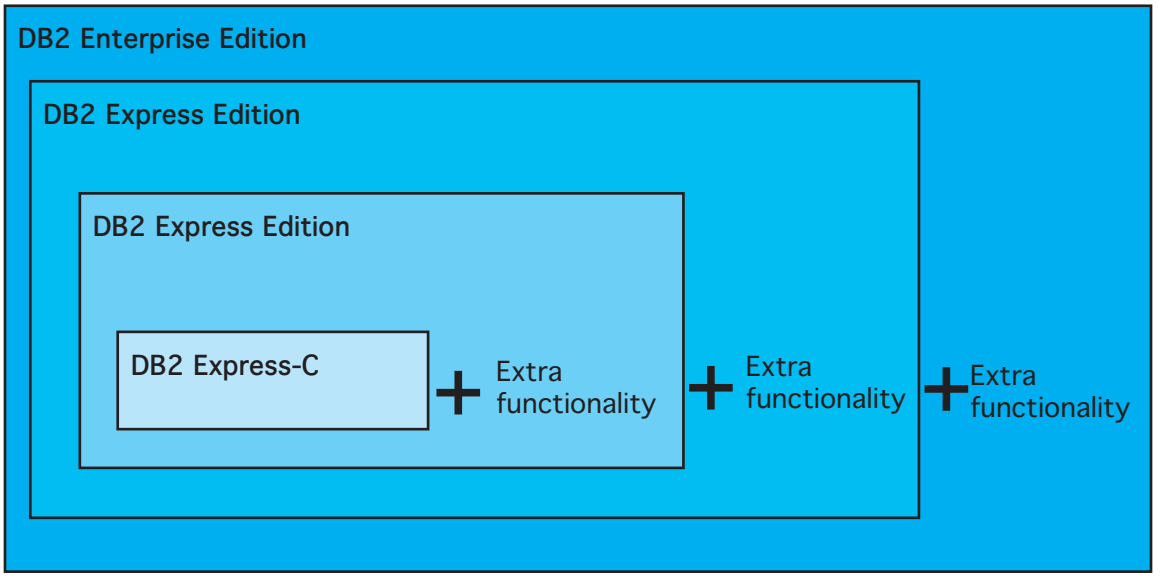
ในรูป ข.1 ด้านซ้ายมือ แสดงตัวอย่างของรายการคำสั่งของ DB2 และคำสั่ง SQL ที่ผู้ใช้สามารถใช้งานได้ ส่วนกลางของรูปแสดงรายการของเครื่องมือที่ใช้ในการป้อนคำสั่ง และทางด้านขวาของรูปแสดงสภาพแวดล้อมของฐานข้อมูล DB2 ที่เก็บไว้ในส่วนต่าง ๆ ซึ่งเราจะกล่าวรายละเอียดเพิ่มเติมในส่วนต่อไป

ข.2 บรรจุภัณฑ์ของ DB2

DB2 เซิร์ฟเวอร์, ไคลแอนต์ และไดรเวอร์ ถูกสร้างขึ้นเป็นองค์ประกอบหลักและบรรจุในบรรจุภัณฑ์ของ DB2 ในลักษณะที่ช่วยให้ผู้ใช้สามารถเลือกฟังก์ชันที่ต้องการในราคาที่เหมาะสม ต่อไปนี้จะอธิบายความแตกต่างของ DB2 แต่ละรุ่น หรือบรรจุภัณฑ์ของ DB2 ที่มีอยู่

ข.2.1 เซิร์ฟเวอร์ DB2

รูป ข.2 แสดงภาพรวมเซิร์ฟเวอร์ DB2 ในรุ่นต่าง ๆ



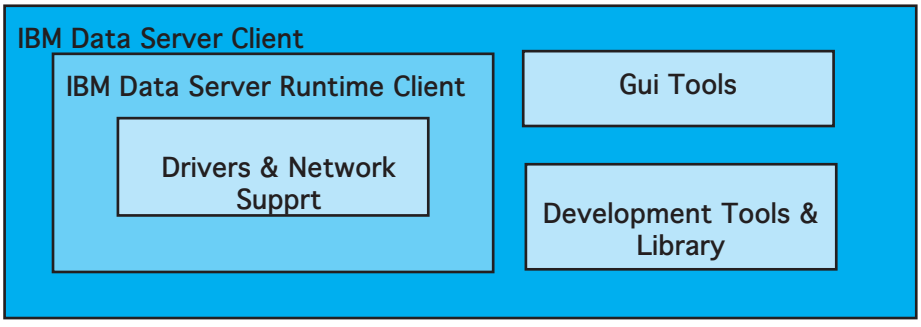
รูป ข.2 แสดงเซิร์ฟเวอร์ของ DB2 รุ่นต่าง ๆ

ในรูป ข.2 รุ่นเซิร์ฟเวอร์ DB2 ทั้งหมดถูกสร้างโดยมีองค์ประกอบที่เป็นแกนหลักที่เหมือนกัน DB2 Express-C เป็น DB2 รุ่นที่ให้ใช้ฟรีและมีองค์ประกอบหลักของผลิตภัณฑ์ DB2 เมื่อเพิ่มเติมฟังก์ชันพิเศษเข้าไปในไป DB2 Express-C จะเปลี่ยนเป็นรุ่น DB2 Express และเมื่อเพิ่มฟังก์ชันพิเศษเพิ่มลงไปใน DB2 Express จะเปลี่ยนเป็นรุ่น DB2 Workgroup และอื่น ๆ ดังรูป ข.2 แสดงให้เห็นว่าเป็นการง่ายที่จะปรับปรุงรุ่นของ DB2 จากรุ่น DB2 Express-C ไปเป็นเซิร์ฟเวอร์ DB2 รุ่นอื่น เซิร์ฟเวอร์ DB2 ทุกรุ่นถูกสร้างขึ้นมาจากพื้นฐานหลักของ DB2 Express-C.

ดังนั้น โปรแกรมประยุกต์ที่สร้างโดยใช้งาน DB2 Express-C จะสามารถใช้ได้กับเซิร์ฟเวอร์ DB2 รุ่นอื่น ๆ ได้เช่นกัน โดยไม่ต้องมีการแก้ไขโปรแกรม

ข.2.2 DB2 ไคลเอ็นต์ และไดรเวอร์

เมื่อได้ติดตั้งเซิร์ฟเวอร์ DB2 DB2 ไคลเอ็นต์คอมโพเนนต์จะถูกติดตั้งด้วย หากต้องการติดตั้ง DB2 ไคลเอ็นต์เพียงอย่างเดียว คุณสามารถติดตั้ง IBM Data Server Client หรือ IBM Data Server Runtime Client เพิ่มเติมได้ แสดงในรูป ข.3



รูป ข.3 DB2ไคลเอ็นต์

รูป ข.3 จะสังเกตเห็น IBM Data Server Runtime Client มีองค์ประกอบทั้งหมดที่ต้องการ (โปรแกรมไดรเวอร์และเครือข่าย) ในการเชื่อมต่อและทำงานกับเซิร์ฟเวอร์ DB2 ไคลเอ็นต์ IBM Data Server Client มีองค์ประกอบที่สนับสนุนการทำงานในรูปแบบเดียวกันกับ IBM Data Server Runtime Client และยังมีส่วนเพิ่มเติมเครื่องมือ GUI และไลบรารีสำหรับการนักพัฒนาโปรแกรมประยุกต์

นอกจากนั้น DB2 ไคลเอ็นต์มีส่วนเพิ่มเติมของไคลเอ็นต์ และ ไดรเวอร์ อื่น ดังต่อไปนี้:

- DB2 Runtime Client Merge Modules for Windows: ส่วนใหญ่ใช้การฝัง DB2 runtime client ให้เป็นส่วนหนึ่งกับการติดตั้งโปรแกรมประยุกต์ของ Windows
- IBM Data Server Driver for JDBC and SQLJ: อนุญาตให้แอพพลิเคชัน Java เชื่อมต่อกับเซิร์ฟเวอร์ DB2 โดยไม่ต้องติดตั้งไคลเอ็นต์
- IBM Data Server Driver for ODBC and CLI: อนุญาตให้โปรแกรมประยุกต์ ODBC และ CLI เชื่อมต่อไปยังเซิร์ฟเวอร์ DB2 โดยไม่ต้องติดตั้งไคลเอ็นต์
- IBM Data Server Driver Package: ประกอบด้วย ไดรเวอร์พิเศษบนวินโดว์ เพื่อสนับสนุน .NET เพิ่มเติมจาก ODBC, CLI และ open source ไดรเวอร์นี้รู้จักกันในชื่อ IBM Data Server Driver สำหรับ ODBC, CLI และ NET

ซึ่งสำหรับ IBM DB2 จะไม่มีค่าธรรมเนียมการใช้ DB2 ไคลเอ็นต์ หรือไดรเวอร์

ข.3 การติดตั้ง DB2

ในส่วนนี้ เราอธิบายวิธีการติดตั้ง DB2 โดยใช้ตัวช่วยในการติดตั้ง DB2

ข.3.1 การติดตั้งบน Windows

การติดตั้ง DB2 บน Windows ให้ทำตามขั้นตอนเบื้องต้นต่อไปนี้:

1. ตรวจสอบให้แน่ใจว่า ชื่อผู้ใช้ (user) ที่ใช้ในการติดตั้งเป็นชื่อผู้ใช้ที่อยู่ในกลุ่มผู้ดูแลระบบ (Administrator Group) บนเครื่องเซิร์ฟเวอร์ที่คุณกำลังติดตั้ง DB2
2. หลังจากดาวน์โหลด และ unzip DB2 Express-C สำหรับ Windows จาก ibm.com/db2/express ค้นหาแฟ้มชื่อ setup.exe และคลิกสองครั้งบนแฟ้มนั้น
3. ทำตามคำแนะนำจากตัวช่วยติดตั้ง และ เลือกค่าเริ่มต้นจากตัวช่วยติดตั้ง ถือว่าเหมาะสมระดับหนึ่ง
4. การเลือกใช้ค่าเริ่มต้นในระหว่างการติดตั้ง มีรายละเอียดดังนี้:
   - DB2 ติดตั้งอยู่ใน C:\Program Files\IBM\SQLLIB
   - บน Windows กลุ่มระบบปฏิบัติการ DB2USERS และ DB2ADMNS จะถูกสร้างขึ้น
   - อินสแตนซ์ DB2 จะถูกสร้างภายใต้ C:\Program Files\IBM\SQLLIB\DB2
   - DB2 Administration Server (DAS) จะถูกสร้างขึ้น
   - บันทึกรายละเอียดการติดตั้งจะถูกเก็บไว้ใน:
     - My Documents\DB2LOG\db2.log
     - My Documents\DB2LOG\db2wi.log

   และ Windows Services จะถูกสร้างขึ้น

ข.3.2 การติดตั้ง DB2 บน Linux

DB2 ติดตั้งบน Linux ให้ทำตามขั้นตอนเบื้องต้นต่อไปนี้:

1. เข้าสู่ระบบเป็นผู้ใช้ Root ในการติดตั้ง DB2
2. หลังจากดาวน์โหลด DB2 Express-C สำหรับ Linux จาก ibm.com/db2/express และทำการค้นหาแฟ้มชื่อ db2setup และทำการรัน :

   ./ db2setup
3. ทำตามคำแนะนำจากตัวช่วยติดตั้งและ เลือกค่าเริ่มต้นจากตัวช่วยติดตั้ง ถือว่าเหมาะสมระดับหนึ่ง
4. การเลือกใช้ค่าเริ่มต้นในระหว่างการติดตั้ง มีรายละเอียดดังนี้:

- DB2 ถูกติดตั้งใน /opt/ibm/db2/V9.7
- ชื่อผู้ใช้ทั้งสามชื่อถูกสร้าง ดังนี้:

db2inst1 (ชื่อผู้ใช้ที่เป็นเจ้าของอินสแตนซ์และชื่ออินสแตนซ์)

db2fenc1 (ชื่อผู้ใช้สำหรับ fenced ฟังก์ชัน)

dasusr1 (ชื่อผู้ใช้ที่ใช้ในการจัดการ DAS อินสแตนซ์)

- กลุ่มผู้ใช้สามกลุ่มถูกสร้างให้สอดคล้องกับชื่อผู้ใช้ด้านบน:

db2iadm1

db2fadm1

dasadm1

- สร้างอินสแตนซ์ชื่อ db2inst1 สำหรับจัดเก็บฐานข้อมูล
- สร้างอินสแตนซ์ชื่อ dasusr1
- บันทึกรายละเอียดการติดตั้งจะถูกเก็บไว้ใน:

/tmp/db2setup.his

/tmp/db2setup.log

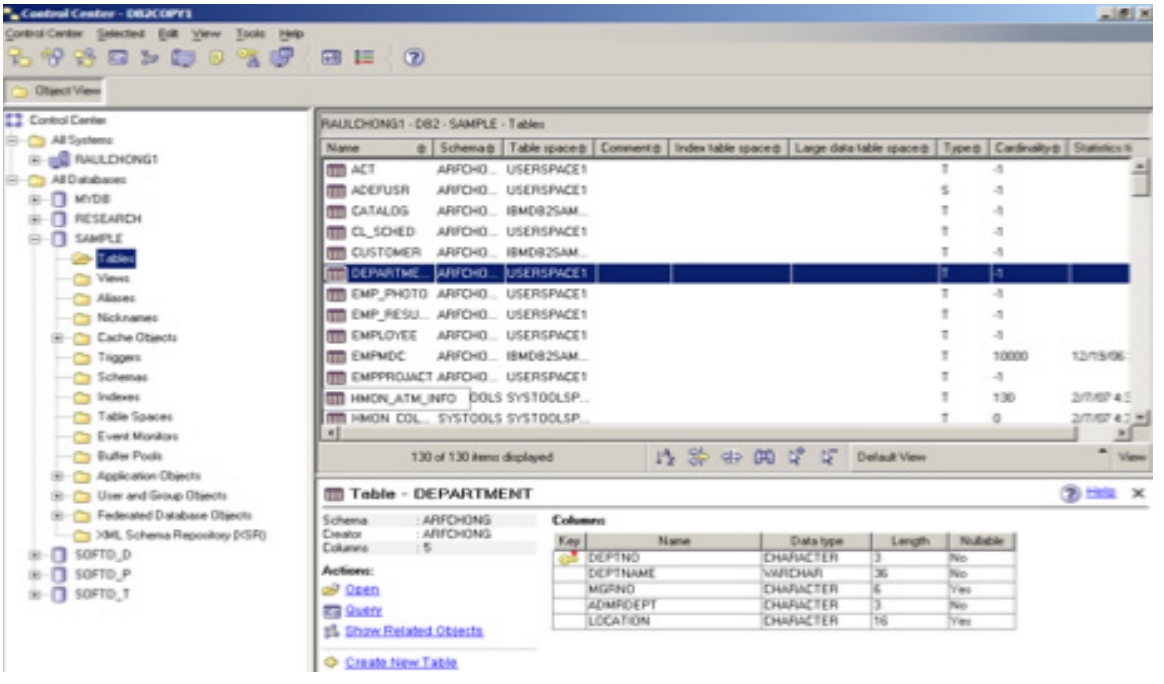
/tmp/db2setup.err

ข.4 เครื่องมือของ DB2

มีเครื่องมือหลาย ๆ อย่างที่มีอยู่ในเซิร์ฟเวอร์ของ DB2 เช่น DB2 Control Center, DB2 Command Editor และอื่น ๆ ตั้งแต่ DB2 เวอร์ชัน 9.7 เครื่องมือเหล่านี้จะไม่แนะนำให้ใช้งาน เพราะจะไม่มีการพัฒนาเพิ่มเติม แต่จะสนับสนุนให้ใช้เครื่องมือใน IBM Data Studio โดยที่ IBM Data Studio ถูกจัดเป็นผลิตภัณฑ์ที่แยกต่างหากไม่รวมอยู่กับ DB2 สามารถดูรายละเอียดเพิ่มเติมจาก Ebook Getting started with IBM Data Studio for DB2

ข.4.1 Control Center

ก่อน DB2 9.7 เครื่องมือหลักของ DB2 สำหรับการจัดการฐานข้อมูลคือ Control Center ตามที่แสดงในรูป ข.4 แม้เครื่องมือนี้จะไม่แนะนำให้ใช้งานแต่ยังคงเป็นเครื่องมือที่รวมไว้กับ DB2 เซิร์ฟเวอร์



รูป ข.4 DB2 Control Center

ในการเริ่มการทำงาน Control Center บน Windows เริ่มจาก start - > Programs - > IBM DB2 - > DB2COPY1(Default) - > General Administration Tools - > Control Center หรืออีกวิธีหนึ่งคือ พิมพ์คำสั่ง db2cc ใน Windows Command Prompt or Linux shell Control Center เป็นเครื่องมือการจัดการฐานข้อมูลแบบศูนย์รวมที่ช่วยให้สามารถ:

- ดูชื่อเครื่อง อินสแตนซ์ ฐานข้อมูล และ อ็อบเจ็กต์ของฐานข้อมูล
- สราง แก้ไข และจัดการฐานข้อมูลและอ็อบเจ็กต์ของฐานข้อมูล
- เปิดใช้เครื่องมืออื่น ๆ ของ DB2

หน้าต่างด้านซ้ายแสดงลำดับชั้นของอ็อบเจ็กต์ของฐานข้อมูลในระบบ ซึ่งประกอบไปด้วยโฟลเดอร์ของตาราง วิวและอื่น ๆ เมื่อดับเบิ้ลคลิกที่โฟลเดอร์ (ตัวอย่างเช่น โฟลเดอร์ของตาราง ดังที่แสดงในรูป ข.5) หน้าต่างด้านขวาด้านบนจะมีรายการทั้งหมดที่เกี่ยวของกับอ็อบเจ็กต์นั้น เช่นในที่นี้จะแสดงชื่อ ตารางทั้งหมดในฐานข้อมูลชื่อ SAMPLE เมื่อทำการเลือกตารางในหน้าต่างด้านขวาด้านบน หน้าต่างด้านขวาด้านล่างจะแสดงขอมูลเพิ่มเติมที่เกี่ยวกับตารางนั้น

การคลิกขวาบนอ็อบเจ็กต์ในลำดับชั้นหรือโฟลเดอร์ต่าง ๆ จะแสดงเมนูที่สามารถใช้ได้กับโฟลเดอร์หรืออ็อบเจ็กต์ที่เลือก ตัวอย่าง คลิกขวาบนอินสแตนซ์ และการเลือกการกำหนดคาพารามิเตอร์จะให้คุณสามารถดู และปรับปรุงค่าพารามิเตอร์ในระดับอินสแตนซ์ ในทำนองเดียวกัน ถาคลิกขวาบนฐานขอมูล และเลือกการกำหนดคาพารามิเตอร์ จะทำให้สามารถดูและปรับปรุงค่าพารามิเตอร์ในระดับฐานข้อมูล

ข.4.2 เครื่องมือ Command Line

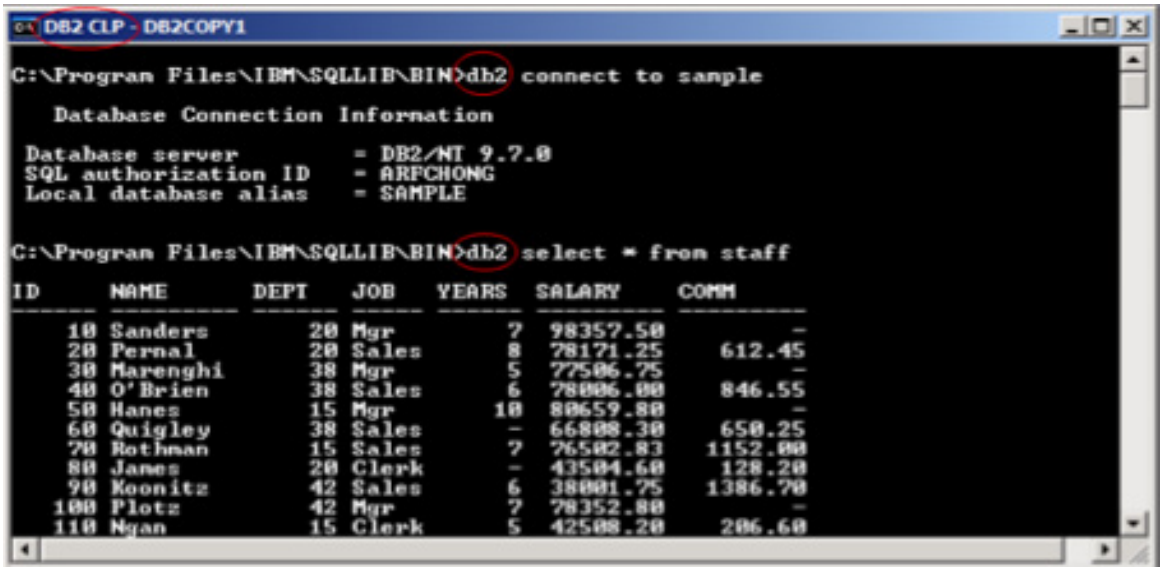
เครื่องมือ Command Line มีอยู่สามชนิด:

- DB2 Command Window (สำหรับ Windows)
- DB2 Command Line Processor (DB2 CLP)
- DB2 Command Editor (GUI-based, ไม่สนับสนุนให้ใช้)

เครื่องมือเหล่านี้ได้รับการอธิบายในรายละเอียดเพิ่มเติมในส่วนถัดไป

ข.4.2.1 DB2 Command Window

ใน DB2 Command Window สามารถใช้เฉพาะบนระบบปฏิบัติการ Windows นอกจากนี้มักจะสับสนกับ Windows Command Prompt ถึงแม้ว่าจะดูไม่แตกต่างกัน การเริ่มต้นทำงานกับ DB2 Command Window เริ่มจาก Start -> Programs -> IBM DB2 -> DB2COPY1 (Default) -> Command Line Tools -> Command Window หรืออีกวิธีหนึ่งคือ พิมพ์คำสั่ง db2cmd จาก Windows Command Prompt เพื่อสั่งการทำงานให้เริ่มในอีกหน้าต่างหนึ่ง รูป ข.5 แสดงในหน้าตาง DB2 Command Window



รูปที่ ข.5 หน้าต่างคำสั่ง DB2

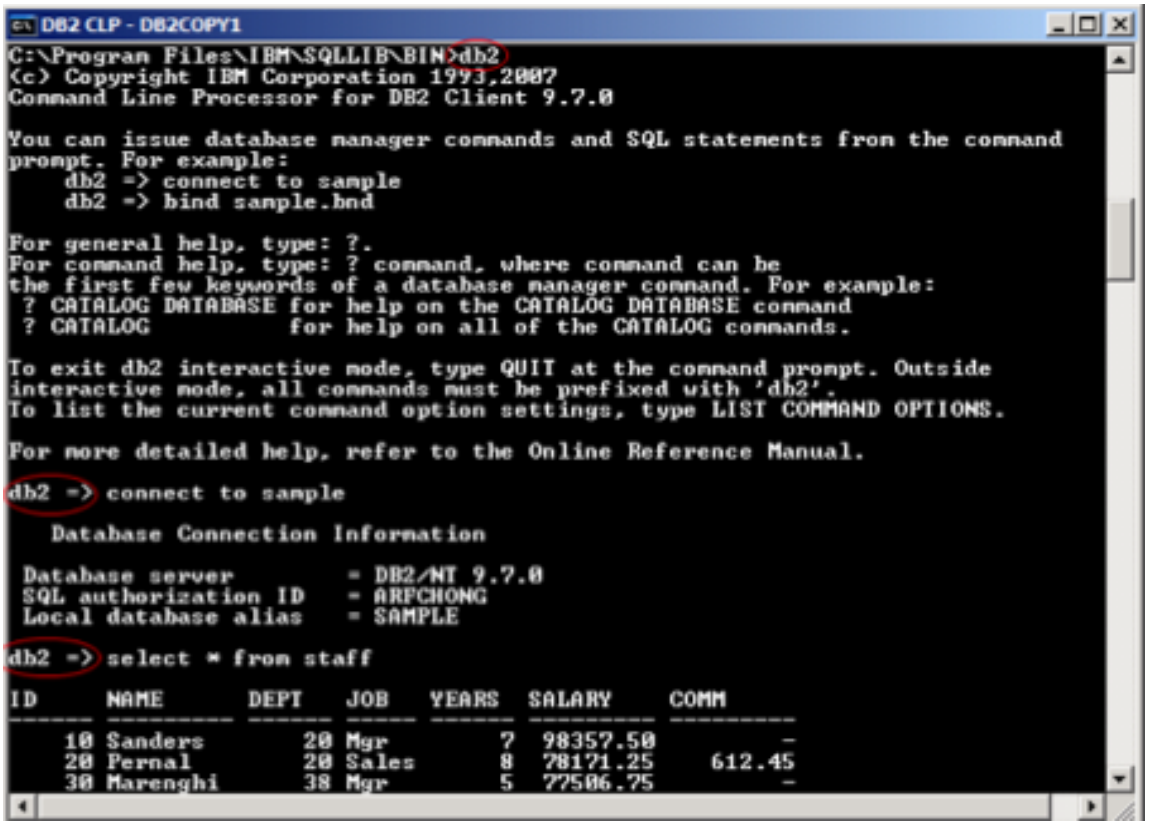
คุณสามารถระบุว่าคุณกำลังทำงานใน DB2 Command Window ด้วยการดูที่หัวเรื่องด้านบนของหน้าต่างซึ่งประกอบด้วยคำว่าDB2 CLP จาก DB2 Command Window คำสั่งแต่ละคำสั่งต้องนำหน้าด้วย db2 ตัวอย่างเช่น ในรูปข้างบน แสดงคำสั่งทั้งสอง ดังนี้:

```
db2 connect to sample
db2 select * from staff
```

สำหรับบน Linux DB2 Command Window เป็นเพียงแค่การ Linux shell (หรือเทอร์มินัล) ซึ่งสภาพแวดล้อมของ DB2 ได้ถูกตั้งค่าด้วยการดำเนินการทำงานจากแฟ้ม db2profile แฟ้มนี้ถูกสร้างขึ้นให้โดยปริยาย และถูกเพิ่มเข้าไปในแฟ้ม .login ของผู้ใช้ที่เป็นเจ้าของอินสแตนซ์ DB2 โดยค่าเริ่มต้นผู้ใช้ที่เป็นเจ้าของอินสแตนซ์ คือ db2inst1

ข.4.2.2 DB2 Command Line Processor

DB2 Command Line Processor (CLP) จะมีลักษณะเหมือนกับ DB2 Command Window มีข้อแตกต่างอย่างหนึ่งที่พร้อมท์จะแสดง db2 = > แทนที่พร้อมท์จะแสดงพร้อมท์ของระบบปฏิบัติการ เมื่อต้องการเริ่มต้น DB2 Command Line Processor บน Windows จาก Start -> Programs -> IBM DB2 -> DB2COPY1 (Default) -> Command Line Tools -> Command Line Processor หรือบน DB2 Command Window หรือ Linux shell พิมพ์ db2 และกด Enter พรอมท์จะเปลี่ยนเป็น db2 = > ดังแสดงในรูป ข.6

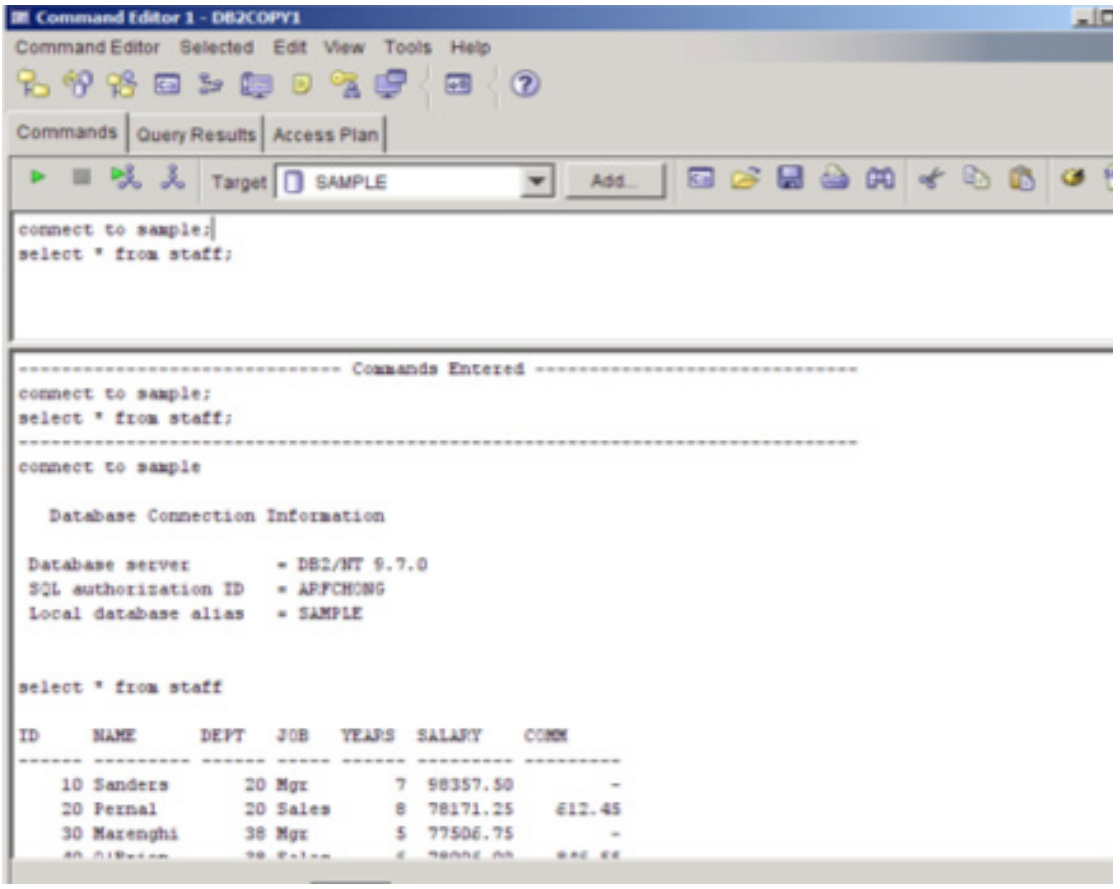


รูปที่ ข.6 DB2 Command Line Processor (CLP)

หมายเหตุ จะสังเกตเห็นว่ารูป ข.6 แสดงให้เห็นว่า เมื่อทำงานใน CLP ไม่จำเป็นต้องใส่คำนำหน้าให้กับคำสั่ง DB2 เมื่อต้องการออกจาก CLP ให้พิมพ์คำสั่ง quit

ข.4.2.3 DB2 Command Editor

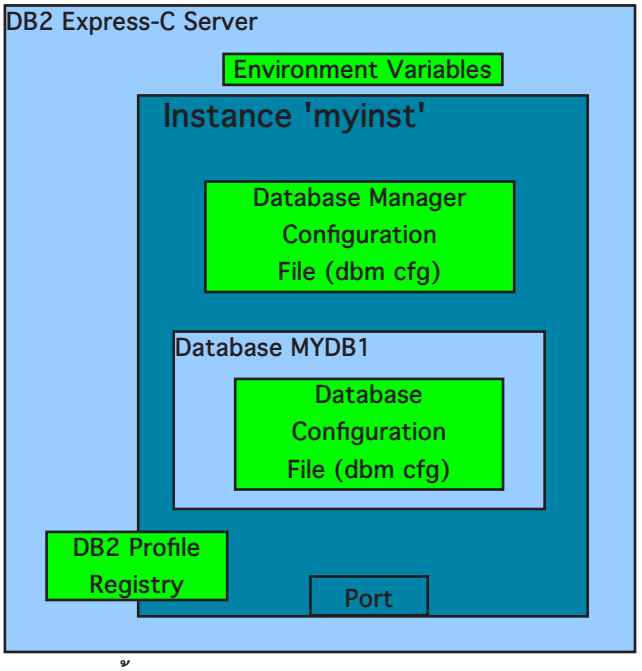
DB2 Command Editor เป็นรูปแบบ GUI ของ DB2 Command Window หรือ DB2 Command Line Processor ดังแสดงในรูป ข.7 เครื่องมือนี้ไม่สนับสนุนให้ใช้ DB2 เวอร์ชัน 9.7



รูปที่ ข.7 DB2 Command Editor

ข.5 สภาพแวดล้อมของ DB2

ในรูป ข.8 แสดงภาพรวมสภาพแวดล้อมของ DB2



รูปที่ ข.8 แสดงสภาพแวดล้อมของ DB2

ในรูปที่ ข.8 แสดงเซิร์ฟเวอร์ DB2 Express-C ได้ถูกติดตั้ง ในกล่องขนาดเล็กในสีเขียวอ่อน เช่น ตัวแปรสภาพแวดล้อม ตัวแปรระดับระบบจัดการฐานข้อมูล ตัวแปรระดับฐานข้อมูล ตัวแปรรีจิสทรี (Environment Variables, Database Manager Configuration File, Database Configuration File, DB2 Profile Registry) เป็นส่วนต่าง ๆ ที่สามารถกำหนดค่าตัวแปรหรือพารามิเตอร์ของเซิร์ฟเวอร์ DB2 ซึ่งจะอธิบายในรายละเอียดเพิ่มเติมในส่วนถัดไป กล่องสีเขียวเข้มที่มีขนาดใหญ่แทนอินสแตนซ์ซึ่งในตัวอย่างนี้มีชื่อว่า myinst

โดยที่อินสแตนซ์ คือ สภาพแวดล้อมที่สามารถสร้างอ็อบเจ็กต์ของฐานข้อมูล บนเซิร์ฟเวอร์เดียวกัน สามารถสร้างอินสแตนซ์ได้หลายอินสแตนซ์ ซึ่งทำงานอย่างอิสระต่อกัน ตัวอย่างเช่น คุณสามารถใช้อินสแตนซ์หนึ่งสำหรับระบบที่กำลังทำการพัฒนา อีกอินสแตนซ์หนึ่งสำหรับระบบการทดสอบ และอีกอินสแตนซ์หนึ่งสำหรับระบบงานจริง ในตาราง ข.1 แสดงบางคำสั่งที่สามารถใช้ได้ในระดับของอินสแตนซ์ จะสังเกตเห็นได้ว่าคำสั่งที่แสดงในส่วนนี้ยังสามารถใช้ได้ในเครื่องมือ GUI ของ DB2 ได้อีกด้วย

คำสั่ง	คำอธิบาย
db2start	เริ่มการทำงานของอินสแตนซ์ปัจจุบัน
db2stop	หยุดการทำงานของอินสแตนซ์ปัจจุบัน
db2icrt <instance_name>	สร้างอินสแตนซ์ใหม่
db2idrop <instance_name>	ลบอินสแตนซ์
db2ilist	แสดงรายการ ชื่ออินสแตนซ์ ที่มีอยู่บนระบบ
db2 get instance	แสดงชื่ออินสแตนซ์ปัจจุบันที่ทำงาน

ตาราง ข.1 คำสั่ง DB2 ในระดับอินสแตนซ์

คุณสามารถสร้างฐานข้อมูลหลายฐานข้อมูลภายในอินสแตนซ์เดียวกัน ฐานข้อมูลคือชุดของอ็อบเจ็กต์ต่างๆ เช่น ตาราง วิว ดัชนี และอื่น ๆ ตัวอย่างเช่น ในรูป ข.8 ฐานข้อมูล MYDB1 ถูกสร้างขึ้นภายใต้อินสแตนซ์ myinst ในตาราง ข.2 แสดงบางคำสั่งที่คุณสามารถใช้ในระดับฐานขอมูล

คำสั่งคำ สั่ง/SQL	คำอธิบาย
create database <database_name>	สร้างฐานข้อมูลใหม่
drop database <database_name>	ลบฐานข้อมูล
connect to <database_name>	เชื่อมต่อกับฐานข้อมูล
create table/create view/create index	คำสั่ง SQL เพื่อสร้างตาราง สร้างวิว และสร้างดัชนีตามลำดับ

ตาราง ข.2 รูปแบบคำสั่งและคำสั่ง SQL ที่ระดับฐานข้อมูล

ข.6 การตั้งค่าใน DB2

พารามิเตอร์ของ DB2 สามารถกำหนดค่าได้โดยใช้เครื่องมือการตั้งค่า Configuration Advisor GUI tool ที่สามารถกำหนดค่าซึ่งสามารถเข้าถึงผ่าน Control Center โดยการคลิกขวาที่ชื่อฐานข้อมูล และทำการเลือก Configuration Advisor จะให้เราตอบคำถามเกี่ยวกับทรัพยากรของระบบและปริมาณงานที่ต้องทำบนระบบ แล้ว Configuration Advisor จะช่วยแนะนำการกำหนดค่าพารามิเตอร์ต่างๆ ที่เหมาะสมใน DB2 หากต้องการรายละเอียดเพิ่มเติมเกี่ยวกับการตั้งค่าพารามิเตอร์ของ DB2 สามารถหาอ่านเพิ่มเติมได้ แต่การใช้ Configuration Advisor ก็ทำให้เราพร้อมที่จะทำงานกับ DB2 แล้ว

DB2 เซิร์ฟเวอร์สามารถกำหนดค่าพารามิเตอร์ในระดับที่แตกต่างกัน 4 ระดับ ตามที่แสดงในรูป ข.8:

- ตัวแปรสภาพแวดล้อม (Environment variables) เป็นตัวแปรที่ตั้งค่าในระดับระบบปฏิบัติการ โดยที่มีตัวแปรสภาพแวดล้อมที่สำคัญตัวหนึ่ง คือ DB2INSTANCE ตัวแปรนี้บ่งชี้ถึง อินสแตนซ์ปัจจุบันที่กำลังทำงานกับ DB2
- ค่าพารามิเตอร์ระดับระบบจัดการฐานข้อมูล (Database Manager Configuration File หรือ

dbm cfg) ประกอบด้วยพารามิเตอร์ที่ส่งผลต่อ อินสแตนซ์และฐานข้อมูลทั้งหมดในอินสแตนซ์ ตาราง ข.3 แสดงตัวอย่างคำสั่งที่ใช้ในการกำหนดค่า dbm cfg

คำสั่ง	คำอธิบาย
get dbm cfg	คำสั่งในการดูรายละเอียดค่า dbm cfg
update dbm cfg using <parameter_ name> <value>	การปรับปรุงค่าของพารามิเตอร์ dbm cfg

ตาราง ข.3 คำสั่งที่ใช้ในการจัดการ dbm cfg

- ค่าพารามิเตอร์ระดับฐานข้อมูล (Database Configuration File หรือ db cfg) ประกอบด้วยพารามิเตอร์ที่ส่งผลต่อฐานขอมูลใดฐานข้อมูลหนึ่ง ตาราง ข.4 แสดงตัวอย่างคำสั่งที่ใช้ในการกำหนดค่า db cfg

คำสั่ง	คำอธิบาย
get db cfg for <database_name>	คำสั่งในการดูรายละเอียดค่า db cfg ของฐานข้อมูล
update db cfg for <database_ name> using <parameter_name> <value>	การปรับปรุงค่าของพารามิเตอร์ db cfg ของฐานข้อมูล

ตาราง ข.4 คำสั่งที่ใช้ในการจัดการ db cfg

ตัวแปร DB2 รีจิสทรี (DB2 Profile Registry variables) มีพารามิเตอร์ที่กำหนดค่าสำหรับแต่ละแพลตฟอร์ม และใช้กำหนดค่าโดยรวมทั้งหมด (ผลกระทบกับอินสแตนซ์ทั้งหมด) หรือ ที่ระดับอินสแตนซ์ (ส่งผลกระทบต่ออินสแตนซ์ใด ๆ) ตาราง ข.5 แสดงคำสั่งที่ใช้ในการจัดการตัวแปร DB2 รีจิสทรี

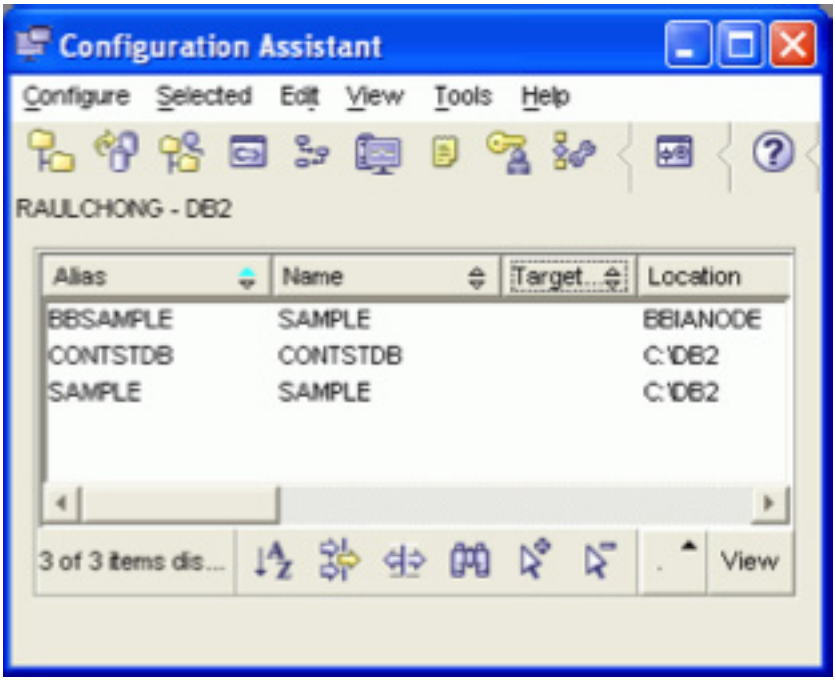
คำสั่ง	คำอธิบาย
db2set -all	แสดงรายการค่าพารามิเตอร์ของ DB2 profile registry ทั้งหมดที่มีการกำหนดไว้
db2set <parameter>=<value>	การปรับปรุงค่าของตัวแปร DB2 รีจิสทรี

ตาราง ข.5 แสดงคำสั่งที่ใช้ในการจัดการตัวแปร DB2 รีจิสทรี

ข.7 การเชื่อมต่อกับฐานข้อมูล

หากฐานข้อมูลของคุณอยู่ภายใต้ระบบเดียวกัน (เครื่องคอมพิวเตอร์เดียวกัน) กับโปรแกรมประยุกต์ที่กำลังดำเนินการอยู่ การตั้งค่าการเชื่อมต่อ (Catalog DB)จะถูกดำเนินการโดยอัตโนมัติเมื่อมีสร้างฐานข้อมูล ถ้าคุณต้องการเชื่อมต่อกับฐานข้อมูล เพียงแค่ใช้คำสั่งในการเชื่อมต่อ connect to database_name ในกรณีการเชื่อมต่อฐานข้อมูล ที่ไม่ได้อยู่ในเครื่องคอมพิวเตอร์เดียวกัน วิธีง่ายที่สุดในการตั้งค่าการเชื่อมต่อฐานข้อมูล โดยใช้เครื่องมือการตั้งค่า Configuration Assistant GUI tool ตามขั้นตอนดังนี้:

1. เปิดโปรแกรม Configuration Assistant โดยใช้คำสั่ง db2ca จาก Windows command prompt หรือ Linux shell รูป ข.9 แสดง Configuration Assistant GUI tool



รูปที่ ข.9 Configuration Assistant GUI tool

2. บนหน้าต่าง Configuration Assistant การตั้งค่าให้คลิกเลือกเมนู Selected --> Add database เพื่อใช้ตัวช่วยการตั้งค่าการเชื่อมต่อฐานข้อมูล

3. จากนั้นจะแสดงหน้าต่างการตั้งค่าการเชื่อมต่อฐานข้อมูล โดยสามารถใช้วิธีการค้นหาฐานข้อมูลจากเครือข่ายถ้าเครือข่ายมีขนาดเล็ก หรือถ้าทราบชื่อของเซิร์ฟเวอร์ที่มี DB2 อยู่ให้เลือกที่ Known systems และเลือกฐานข้อมูลที่ต้องการเชื่อมต่อ โดยตัวช่วยจะใช้ค่าเริ่มต้นของโปรแกรม ถ้าคุณไม่ทราบชื่อของเซิร์ฟเวอร์ที่มี DB2 อยู่ให้เลือก Other systems (Search the network) วิธีการนี้อาจใช้เวลานานหากเครือข่ายมีขนาดใหญ่

4. กรณีค้นหาจากเครือข่ายไม่พบ ให้กลับไปยังหน้าต่างการตั้งค่าการเชื่อมต่อฐานข้อมูล จากนั้นทำการเลือก Manually configure a connection to a database ให้ทำการเลือก TCP/IP และคลิก ถัดไป จากนั้นทำการกรอกไอพี แอดเดรส (IP Address) หรือชื่อโฮสต์ที่อยู่เซิร์ฟเวอร์ DB2 และป้อนข้อมูล service name หรือหมายเลขพอร์ต.

5. ดำเนินต่อไปตามตัวช่วย โดยใช้ค่าเริ่มต้นต่าง ๆ ที่กำหนดมาให้

6. หลังจากที่เสร็จสิ้นการตั้งค่า หน้าต่างจะปรากฏขึ้นว่า ต้องการทดสอบการเชื่อมต่อฐานข้อมูลหรือไม่ คุณสามารถทดสอบการเชื่อมต่อหลังจากการตั้งค่าเสร็จสิ้นแล้ว โดยการคลิกขวาบนฐานข้อมูลเลือก Test Connection

ข.8 โปรแกรมตัวอย่าง

ทั้งนี้ขึ้นอยู่กับภาษาที่ใช้ในการเขียนโปรแกรม ซึ่งจะใช้ไวยากรณ์ที่แตกต่างกันในเชื่อมต่อกับฐานข้อมูล DB2 รายละเอียดส่วนล่างนี้เป็นโปรแกรมตัวอย่างของการเชื่อมต่อกับฐานข้อมูล เราขอแนะนำให้คุณสามารถดาวน์โหลดโปรแกรมตัวอย่างทั้งหมดจากเว็บไซต์

ftp://ftp.software.ibm.com/software/data/db2/udb/db2express/samples.zip

โปรแกรม CLI

http://www.ibm.com/developerworks/db2/library/techarticle/dm-0401chong/index.html#scenario1

โปรแกรม ODBC

http://www.ibm.com/developerworks/db2/library/techarticle/dm-0401chong/index.html#scenario2

โปรแกรมภาษา C ที่ฝัง SQL
http://www.ibm.com/developerworks/db2/library/techarticle/dm-0401chong/index.html#scenario3

โปรแกรม JDBC ที่ใช้ Type 2 Universal (JCC)ไดรเวอร์
http://www.ibm.com/developerworks/db2/library/techarticle/dm-0401chong/index.html#scenario6

โปรแกรม JDBC ที่ใช้ Type 4 Universal (JCC)ไดรเวอร์
http://www.ibm.com/developerworks/db2/library/techarticle/dm-0401chong/index.html#scenario8

Visual Basic และ โปรแกรม c ++ ADO - ใช้ IBM OLE DB provider สำหรับ DB2 (IBMDADB2)
http://www.ibm.com/developerworks/db2/library/techarticle/dm-0402chong2/index.html#scenario1

Visual Basic และ โปรแกรม c ++ ADO - ใช้ Microsoft OLE DB Provider สำหรับ ODBC (MSDASQL)
http://www.ibm.com/developerworks/db2/library/techarticle/dm-0402chong2/index.html#scenario2

Visual Basic และ C# ADO .NET ใช้ IBM DB2 .NET Data Provider
http://www.ibm.com/developerworks/db2/library/techarticle/dm-0402chong2/index.html#scenario3

Visual Basic และ C# ADO .NET ใช้ Microsoft OLE DB .NET Data Provider
http://www.ibm.com/developerworks/db2/library/techarticle/dm-0402chong2/index.html#scenario4

Visual Basic และ C# ADO .NET ใช้ Microsoft ODBC .NET Data Provider
http://www.ibm.com/developerworks/db2/library/techarticle/dm-0402chong2/index.html#scenario5

ข.9 เอกสารประกอบ DB2

DB2 Information Center แสดงเอกสารประกอบ DB2 ที่เป็นออนไลน์บนเว็บไซต์ คุณสามารถเข้าไปยังเว็บไซต์ DB2 Information Center จาก http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp หรือดาวน์โหลดเอกสารและติดตั้ง DB2 Information Center บนเครื่องของคุณ จาก http://www.ibm.com/software/data/db2/9/download.htmlS_TACT=download & S_CMP = expcsite

แหล่งข้อมูลเพิ่มเติม

เว็บไซต์

1. DB2 Express-C โฮมเพจ
   ibm.com/db2/express
   เว็บไซต์นี้เป็นหน้าแรกของ DB2 Express-C. คุณสามารถค้นหาเพื่อดาวน์โหลด DB2 Express-C ฟรีจากหน้านี้
2. IBM Data Studio โฮมเพจ
   http://www-01.ibm.com/software/data/optim/data-studio/
   เว็บไซต์นี้เป็นโฮมเพจของ ฟรี IBM Data Studio, เครื่องมือ Eclipse-based tool ที่สามารถใช้กับ DB2
3. DB2 Express-C และ IBM Data Studio ดาวน์โหลดฟรี
   http://www.ibm.com/db2/express/download.htmlS_CMP = ECDDWW01 & S_TACT = DOCBOOK01
4. InfoSphere Data Architect โฮมเพจ

http://www-01.ibm.com/software/data/optim/data-architect/

หนังสือ

Free ebook: Getting started with DB2 Express-C (3rd Edition)
Raul F. Chong et all - June 2009
http://www.db2university.com
Free ebook: Getting started with IBM Data Studio for DB2
Debra Eaton et all - Dec 2009
http://www.db2university.com
DB2 9 pureXML® Guide
Whei-Jen Chen, Art Sammartino, Dobromir Goutev, Felicity Hendricks, Ippei Komi, Ming-Pang Wei, Rav Ahuja
August 2007 - SG24-7315-01
http://www.redbooks.ibm.com/abstracts/sg247315.html
Free Redbook®: DB2 Security and Compliance Solutions for Linux, UNIX, and Windows, Whei-Jen Chen, Ivo Rytir, Paul Read, Rafat Odeh, March 2008, SG 24-7555-00
http://www.redbooks.ibm.com/abstracts/sg247555.html?Open

เอกสารอ้างอิง

[1.1] CODD, E.F. A relational model of data for large shared data banks, CACM 13, NO 6, 1970
[2.1] DATE, C.J. An introduction to database systems, Addison-Wesley Publishing Company, 1986
[2.2] MITEA, A.C. Relational and object-oriented databases, “Lucian Blaga” University Publishing Company, 2002
[2.3] CODD, E.F. Relational completeness on data base sublanguage, Data Base Systems, Courant Computer Science Symposia Series, Vol.6 Englewood Cliffs, N.J, Prentice-Hall, 1972
[2.4] KUHNS, J.L. Answering questions by computer: A logical study, Report RM-5428-PR, Rand Corporation, Santa Monica, California, 1967
[2.5] CODD, E.F. A data base sublanguage founded on the relational calculus, Proceedings ACM SIGFIDET Workshop on Data Description, Access and Control, 1971
[2.6] LACROIX, M., PIROTTE, A. Domain oriented relational languages, Proceedings 3rd International Conference on Very Large Data Bases, 1977
[2.7]LACROIX, M., PIROTTE, A. Architecture and models in data base management systems, G.M. Nijssen Publishing company, North-Holland, 1977
[3.1] IBM Rational Data Architect Evaluation Guide
[3.2] Connolly, T., Begg, C., Strachan, A. – Database Systems – A Practical Approach to Design, Implementation and Management, Addison Wesley Longman Limited 1995, 1998
[3.3] IBM InfoSphere Data Architect – Information Center
[3.4] http://www.ibm.com/developerworks/data/bestpractices/
[3.5] 03_dev475_ex_workbook_main.pdf, IBM Rational Software, Section 1: Course Registration Requirements, Copyright IBM Corp. 2004
[4.1] Codd, E. F. The Relational Model for Database Management
[4.2] Codd, E.F. “Further Normalization of the Data Base Relational Model.”
[4.3] Date, C. J. “What First Normal Form Really Means”
[4.4] Silberschatz, Korth, Sudershan - Database System Concepts
[4.5] William Kent - A Simple Guide to Five Normal Forms in Relational Database Theory
[4.6] Raghu Ramakrishnan, Johannes Gehrke - Database management systems
[4.7] Vincent, M.W. and B. Srinivasan. “A Note on Relation Schemes Which Are in 3NF But Not in BCNF.”
[4.8] C. J Date : An Introduction to Database Systems 8th Edition

[4.9] William Kent: A simple guide to five normal forms in relational database theory
http://www.bkent.net/Doc/simple5.htm
[4.10] Ronald Fagin, C J Date: Simple conditions for guaranteeing higher normal forms in relational databases
http://portal.acm.org/citation.cfm?id=132274
[4.11] Ronald Fagin: A Normal Form for Relational Databases That Is Based on Domains and Keys
http://www.almaden.ibm.com/cs/people/fagin/tods81.pdf
[4.12] C. J. Date, Hugh Darwen, Nikos A. Lorentzos: Temporal data and the relational model p172
[4.13] C J Date: Logic and databases, Appendix –C
[5.1] Differences between SQL procedures and External procedures
http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.db29.doc.apsg/db2z_differencesqlprocexternalproc.htm
[5.2] SQL Reference Guide
http://www.ibm.com/developerworks/data/library/techarticle/0206sqlref/0206sqlref.html
[6.1] http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.db29.doc.apsg/db2z_differencesqlprocexternalproc.htm

ข้อมูลติดต่อ

อีเมลผู้ติดตอ: กล่องจดหมาย DB2 Campus Program : db2univ@ca.ibm.com

การเริ่มต้นศึกษาด้วย ฐานข้อมูลเบื้องต้นเล่มนี้ ท่านจะได้รับประโยชน์ดังต่อไปนี้

- ได้รู้ว่าฐานข้อมูลเกี่ยวของกับอะไรบ้าง
- มีความเข้าใจแบบจำลองข้อมูลเชิงสัมพันธ์ รูปแบบข้อมูล และแบบจำลองแนวคิด
- เรียนรู้วิธีการออกแบบฐานข้อมูล
- การเริ่มต้นวิธีเขียนคำสั่ง SQL ฟังก์ชันของฐานข้อมูล และ Procedure
- รู้จัก DB2 pureXML และการใช้ XML รวมกับข้อมูลเชิงสัมพันธ์
- ความเข้าใจเกี่ยวกับความปลอดภัยของฐานข้อมูล
- ฝึกปฏิบัติโดยใช้ภาคปฏิบัติจากแบบฝึกหัด

ข้อมูลเป็นสินทรัพย์ที่สำคัญมากอย่างหนึ่งของธุรกิจ ซึ่งจะมีทั้งการจัดเก็บและเรียกออกมาใช้งาน เช่นธุรกิจเกี่ยวกับบัตรเครดิตจะมีการรวบรวมข้อมูลของลูกค้าเพื่อศึกษาแผนการใช้จ่ายเงินของลูกค้า ธุรกิจเกี่ยวกับอวกาศจะมีการรวบรวมข้อมูลของดาวเคราะห์ต่าง ๆ ในอวกาศ ระบบฐานข้อมูลสนับสนุนการจัดการข้อมูลเป็นที่นิยมใช้อย่างแพร่หลายซึ่งจะมีผู้ใช้นับล้านเข้ามาใช้งาน

หนังสือเล่มนี้ช่วยให้คุณเริ่มเข้าสู่โลกของฐานข้อมูล และการเรียนพื้นฐานของระบบการจัดการฐานข้อมูล โดยการอ้างอิงระบบการจัดการฐานข้อมูล IBM DB2 โดยใช้ DB2 Express-C ซึ่งเป็น DB2 รุ่นที่แจกจ่ายฟรีที่คุณสามารถเรียนรู้องค์ประกอบต่าง ๆ ที่ประกอบเป็นระบบการจัดการฐานข้อมูล รวมถึงเรียนรู้ภาษา SQL และ XML

นอกจากนั้นยังมีตัวอย่างและแบบฝึกหัดที่จะให้คุณฝึกฝนให้เกิดประสบการณ์ รวมถึงแนวคิดเกี่ยวกับการใช้ฐานข้อมูลในงานจริง
เมื่อต้องการเรียนรู้เพิ่มเติมเกี่ยวกับฐานข้อมูลเบื้องต้นและหัวข้อการจัดการข้อมูล สามารถเยี่ยมชมได้ที่:
ibm.com/developerworks/data/
เมื่อต้องการเรียนรู้เพิ่มเติม หรือดาวน์โหลด DB2 Express-C สามารถเยี่ยมชมได้ที่:
ibm.com/db2/express
การค้นหาและดูวิดีโอที่เกี่ยวข้อง สามารถเยี่ยมชมได้ที่:
channelDB2.com
หนังสือเล่มนี้เป็นส่วนหนึ่งของ DB2 on Campus ฟรี ebook สำหรับชุมชน เรียนรู้เพิ่มเติมที่ db2university.com

ประวัติผู้แปล

ชื่อ นายรัฐสิทธิ์ สุขะหุต
Rattasit Sukhahuta
ตำแหน่งปัจจุบัน ผู้ช่วยศาสตราจารย์ ระดับ 8
ภาควิชาวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ มหาวิทยาลัยเชียงใหม่
rattasit.s@cmu.ac.th

ประวัติการทำงาน

- ผู้ช่วยศาสตราจารย์ ระดับ 8 ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยเชียงใหม่ (ปีที่บรรจุ 2539 - ปัจจุบัน)
- รองผู้อำนวยการสำนักบริการเทคโนโลยีสารสนเทศ มหาวิทยาลัยเชียงใหม่ (กันยายน 2548 -ปัจจุบัน)

ประวัติการศึกษา

- ระดับปริญญาตรี สาขาวิทยาการคอมพิวเตอร์
  Bachelor Degree in Computer Science
  University of Hawaii at Hilo, Hawaii, USA, 1995
- ระดับปริญญาโท สาขาวิชาเทคโนโลยีสารสนเทศ
  Master Degree in Information Systems
  Hawaii Pacific University, Hawaii, USA, 1996
- ระดับปริญญาเอก สาขาวิทยาการคอมพิวเตอร์
  Doctor of Philosophy
  University of East Anglia, Norwich, United Kingdom, 2001