บทที่ 2
แนวคิดและทฤษฎีที่เกี่ยวข้อง

ปัจจุบันสารสนเทศได้เข้ามามีบทบาทสำคัญในการดำเนินกิจกรรม และระบบงานของแต่ละองค์กร ซึ่งการจะทำให้ระบบสารสนเทศเหล่านี้มีประสิทธิภาพเพียงพอที่จะสนับสนุนระบบงานได้ จำเป็นต้องมีการจัดการข้อมูลอย่างมีประสิทธิภาพ ด้วยเหตุผลดังกล่าว จึงเป็นที่มาของการพัฒนาเทคโนโลยีฐานข้อมูลสำหรับการจัดการและบริหารสารสนเทศอย่างมีโครงสร้างสำหรับข้อมูลและสารสนเทศในองค์กร

ฐานข้อมูล คือ การร่วมใช้กลุ่มของข้อมูลที่มีความสัมพันธ์กันซึ่งได้ออกแบบมาเพื่อให้กลุ่มข้อมูลเหล่านี้สามารถให้สารสนเทศที่เพียงพอต่อความต้องการใช้งานสำหรับหน่วยงานต่าง ๆ ในองค์กร (อัจฉราและคณะ, 2544:1-6)

ฐานข้อมูล หมายถึงการเก็บรวบรวมข้อมูลที่มีความสัมพันธ์กันไว้ในที่ที่เดียวกัน (สมจิตรและงานนิจ อาจอินทร์, 2541:12)

ฐานข้อมูล ประกอบด้วยรายละเอียดของข้อมูลที่เกี่ยวข้องกัน ซึ่งถูกนำมาใช้ในงานด้านต่าง ๆ เช่นด้านธนาคาร จะมีฐานข้อมูลที่เกี่ยวข้องกับข้อมูลเงินฝาก ข้อมูลการให้สินเชื่อ หรือด้านโรงพยาบาลจะมีฐานข้อมูลที่เกี่ยวข้องกับประวัติคนไข้ ข้อมูลแพทย์เชี่ยวชาญเฉพาะโรค เป็นต้น ข้อมูลเหล่านี้จะถูกจัดเก็บไว้อย่างมีระบบ เพื่อประโยชน์ในการจัดการและเรียกใช้ข้อมูลได้อย่างมีประสิทธิภาพ (ศิริลักษณ์ โรจนกิจอำนวย, 2542:11)

สรุปได้ว่าความหมายของฐานข้อมูล คือ การเก็บรวบรวมข้อมูลที่มีความสัมพันธ์กัน ซึ่งได้ออกแบบมาใช้ในงานด้านต่าง ๆ ข้อมูลเหล่านี้จะถูกเก็บไว้อย่างมีระบบ เพื่อประโยชน์ในการจัดการและเรียกใช้ข้อมูลได้อย่างมีประสิทธิภาพ

2.1 สถาปัตยกรรมฐานข้อมูล

สถาปัตยกรรมฐานข้อมูล (Database Architecture) ถูกพัฒนาขึ้นในปีค.ศ.1971 โดยกลุ่มทำงานที่เรียกว่า กลุ่มทำงานฐานข้อมูล(Database Task Group:DBTG) ภายใต้ข้อตกลงร่วม

กันของการประชุมกำหนดรูปแบบของระบบฐานข้อมูลและภาษาที่ใช้ที่เรียกว่า CODASYL (the Conference On Data Systems and Language) โดยจัดความต้องการหรือมุมมองของระบบออกเป็นสองระดับคือ เค้าร่าง(Schema) และมุมมองของผู้ใช้ที่เป็นเค้าร่างย่อย (Subschemas)

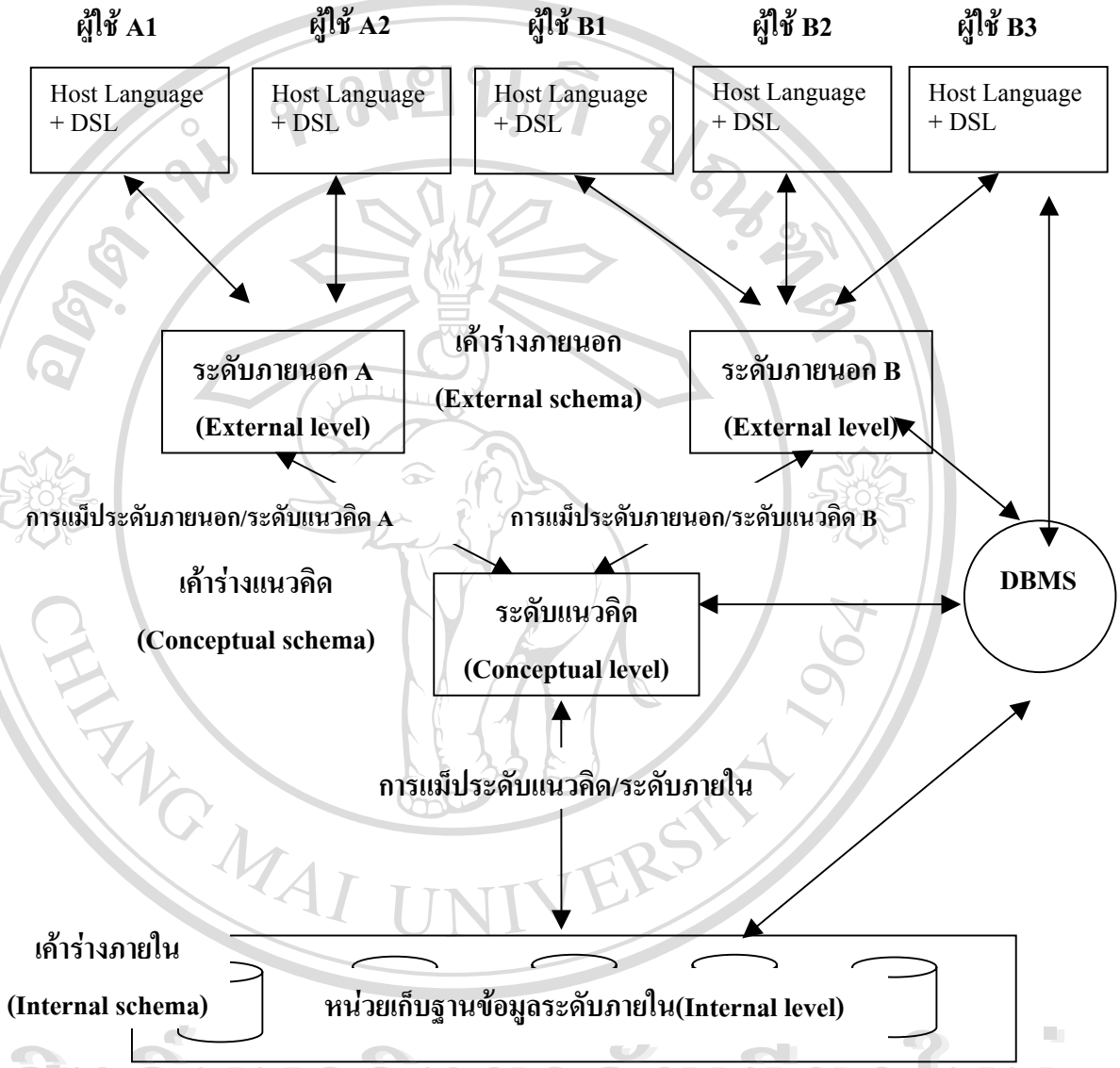
ในปีค.ศ.1975 สถาบันมาตรฐานแห่งชาติอเมริกัน(The American National Standards Institute:ANSI) และคณะกรรมการวางแผนมาตรฐานและคำร้องขอ(Standards Planning and Requirements Committee:SPARC) ได้จัดความต้องการใช้ โดยแบ่งสถาปัตยกรรมฐานข้อมูลออกเป็น 3 ระดับดังนี้ (อัจฉราและคณะ, 2544:1-44)

2.1.1 สถาปัตยกรรมฐานข้อมูล 3 ระดับ

(1) ระดับแนวคิด (Conceptual level) เป็นระดับของการออกแบบฐานข้อมูล ซึ่งจะเริ่มตั้งแต่การวิเคราะห์ความต้องการสารสนเทศของผู้ใช้ ว่าควรจะเก็บข้อมูลอะไรบ้างและความสัมพันธ์ระหว่างข้อมูลเป็นอย่างไร เพื่อจะนำข้อมูลที่ได้มาออกแบบฐานข้อมูล ผลลัพธ์ที่ได้จากการออกแบบฐานข้อมูล จะทำให้เกิดสิ่งที่เรียกว่าเค้าร่าง ในระดับนี้จะเรียกว่าเค้าร่างแนวคิด (Conceptual schema) ซึ่งเป็นสิ่งที่ใช้อธิบายว่าฐานข้อมูลที่สร้างขึ้นประกอบ

(2) ระดับภายนอก (External level) เป็นระดับการมองหรือวิว (View) ของข้อมูลภายในฐานข้อมูลสำหรับผู้ใช้งานแต่ละคน ผู้ใช้ระดับภายนอกนี้สามารถเป็นได้ตั้งแต่นักเขียนโปรแกรมประยุกต์หรืออาจเป็นผู้ปฏิบัติการทั่วไป ซึ่งระดับนี้จะเป็นระดับที่มีการนำข้อมูลจากฐานข้อมูลไปใช้งาน โดยผู้ใช้แต่ละคนสามารถเลือกอ่านข้อมูลเฉพาะที่ตนเองสนใจหรือต้องการใช้เท่านั้น ดังนั้นผู้ใช้แต่ละคนจะมีการมองของข้อมูลในฐานข้อมูลที่แตกต่างกันได้ ซึ่งการมองของข้อมูลนี้จะถูกดึงมาจากเค้าร่างแนวคิด และสิ่งที่จะใช้อธิบายการมองข้อมูลที่ถูกดึงข้อมูลมาจากฐานข้อมูลที่อยู่ในระดับแนวคิดเรียกว่าเค้าร่างย่อย ซึ่งในระดับภายนอกนี้จะสามารถมีได้หลายเค้าร่างย่อยตามจำนวนผู้ใช้ที่มีการสร้างวิวของตนเองขึ้นมา (สมจิตรและงานนิจ อาจอินทร์, 2541:47)

(3) ระดับภายใน (Internal level) เป็นระดับของการจัดเก็บฐานข้อมูลรวมไปถึงโครงสร้างข้อมูลและการจัดการแฟ้มในแหล่งเก็บข้อมูลทางกายภาพของฐานข้อมูล โดยมีระบบจัดการฐานข้อมูลทำงานร่วมกับระบบปฏิบัติการ เพื่อจัดเก็บข้อมูลลงในอุปกรณ์จัดเก็บ รวมถึงทำการสร้างดัชนี (index) และกำหนดตัวชี้ (pointer) ที่ใช้ในการดึงข้อมูลจากฐานข้อมูล (อัจฉราและคณะ, 2544:1-46)



รูป 2.1 แสดงสถาปัตยกรรม 3 ระดับตามมาตราฐาน ANSI/SPARC

2.1.2 ประโยชน์ของสถาปัตยกรรม 3 ระดับ

จากโครงสร้างสถาปัตยกรรมฐานข้อมูลทั้ง 3 ระดับนี้ แต่ละระดับจะมีระบบจัดการฐานข้อมูลทำหน้าที่แม็ประดับข้อมูลระดับหนึ่งเป็นอีกระดับหนึ่ง ซึ่งได้แก่ระหว่างระดับภายนอกกับ

ระดับแนวคิดและระหว่างระดับแนวคิดกับระดับภายใน การแม็ปนี้จะทำให้เกิดประโยชน์ในด้านต่าง ๆ คือ

(1) มุมมองของผู้ใช้งาน

ระดับแนวคิดและระดับภายนอก การแม็ปข้อมูลระหว่างระดับแนวคิดและระดับภายนอก จะทำให้ผู้ใช้ฐานข้อมูลสามารถมีมุมมองของข้อมูลที่แตกต่างกันได้

ระดับแนวคิดและระดับภายใน การแม็ปข้อมูลระหว่างระดับแนวคิดและระดับภายใน จะทำให้ผู้ใช้ฐานข้อมูลไม่ว่าระดับแนวคิดหรือระดับภายนอก ไม่จำเป็นต้องทราบว่าข้อมูลที่ตนเองต้องการใช้นั้นถูกจัดเก็บอยู่อย่างไร เมื่อต้องการใช้ข้อมูลไม่ว่าจะเป็นตารางหรือเขตข้อมูล จะสามารถอ้างถึงชื่อตารางหรือชื่อเขตข้อมูลนั้นได้โดยตรง ซึ่งเป็นหน้าที่ของระบบจัดการฐานข้อมูลที่จะดูว่าข้อมูลที่ผู้ใช้ต้องการนั้นถูกเก็บอยู่ ณ ตำแหน่งใดในดิสก์ แล้วทำการดึงข้อมูลนั้นจากดิสก์มาให้แก่ผู้ใช้ได้

(2) ความเป็นอิสระของข้อมูล

ระดับแนวคิดและระดับภายนอก การเปลี่ยนแปลงฐานข้อมูลในระดับแนวคิดไม่ว่าจะเป็นการเพิ่มตาราง การเพิ่มเขตข้อมูลหรือการเปลี่ยนแปลงขนาดของข้อมูล จะไม่มีผลกระทบกับโปรแกรมประยุกต์ที่ผู้ใช้เขียนขึ้นในระดับภายนอก ตัวอย่างเช่น ถ้ามีการเปลี่ยนแปลงขนาดของข้อมูล รหัสนักศึกษาจากเดิม คือ 9 เปลี่ยนเป็น 10 ในตารางการลงทะเบียนของฐานข้อมูลในระดับแนวคิดก็สามารถทำได้โดยไม่ต้องไปเปลี่ยนแปลงโปรแกรมประยุกต์ตามไปด้วย ยกเว้นว่าถ้ามีการลบตารางหรือเขตข้อมูลใดจากฐานข้อมูลในระดับแนวคิด แล้วโปรแกรมประยุกต์มีการเรียกใช้งานตารางหรือเขตข้อมูลนั้นอยู่ก็อาจจะต้องไปแก้ไขโปรแกรมประยุกต์นั้นด้วย แต่โดยปกติแล้วการเปลี่ยนแปลงโครงสร้างของฐานข้อมูลเป็นสิ่งที่ไม่ควรทำ หรือถ้าทำก็ไม่ควรจะทำบ่อยนัก

ระดับแนวคิดและระดับภายใน การแม็ปข้อมูลระดับแนวคิดและระดับภายในทำให้เกิดความเป็นอิสระกันของข้อมูล เนื่องจากถ้าระดับภายในมีการเปลี่ยนแปลงวิธีการจัดเก็บข้อมูลลงในดิสก์ เช่น จากเดิมใช้วิธีการจัดเก็บแบบลำดับเชิงดัชนี (Index sequential) เปลี่ยนเป็นแบบสุ่ม (Direct access) ในระดับแนวคิดเมื่อมีการสร้างตารางใดจะไม่มีผลกระทบหรือไม่ต้องรับรู้ถึงการเปลี่ยนแปลงดังกล่าว หรือโปรแกรมประยุกต์ที่เขียนในระดับภายนอกไม่จำเป็นต้องแก้ไข โปรแกรมตามวิธีการจัดเก็บที่เปลี่ยนแปลงไป (สมจิตรและงานนิจ อาจอินทร์, 2541:51-52)

2.2 แบบจำลองฐานข้อมูล

แบบจำลองฐานข้อมูล (Database Model) คือ การจัดกลุ่มของโครงสร้างทางแนวคิดที่ใช้เป็นตัวแทนโครงสร้างข้อมูลและความสัมพันธ์ข้อมูลในฐานข้อมูล สามารถแบ่งได้ออกเป็น 2 กลุ่มดังนี้

1. แบบจำลองในลักษณะแนวคิด (Conceptual Models) เน้นการแสดงข้อมูลในแนวคิดที่เป็นธรรมชาติ ซึ่งจะรวมถึงแบบจำลองความสัมพันธ์เอนทิตีและแบบจำลองเชิงวัตถุ (Object-oriented Model)

2. แบบจำลองในลักษณะใช้งาน (Implementation Models) เน้นการแสดงข้อมูลในลักษณะของการประยุกต์ใช้งาน ซึ่งจะแบ่งออกเป็น 3 ประเภทคือ แบบจำลองฐานข้อมูลเชิงชั้น (Hierarchical Database Model) แบบจำลองฐานข้อมูลข่ายงาน (Network Database Model) และแบบจำลองฐานข้อมูลเชิงสัมพันธ์ (Relational Database Model) (อัจฉราและคณะ, 2544:1-50)

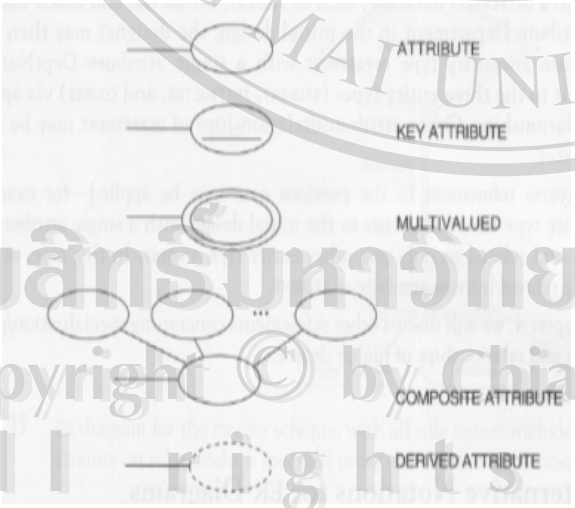
สำหรับงานวิจัยนี้เกี่ยวข้องกับแบบจำลอง 3 แบบจำลองด้วยกันคือ แบบจำลองความสัมพันธ์เอนทิตี แบบจำลองเชิงวัตถุและแบบจำลองฐานข้อมูลเชิงสัมพันธ์ ดังนั้นในหัวข้อต่อไปจึงขอกล่าวถึงเฉพาะแบบจำลองทั้งสามนี้เท่านั้น โดยแบบจำลองความสัมพันธ์เอนทิตีจะกล่าวในหัวข้อ 2.3 แบบจำลองฐานข้อมูลเชิงสัมพันธ์จะกล่าวในหัวข้อ 2.4 และแบบจำลองเชิงวัตถุจะกล่าวในหัวข้อ 2.6

2.3 แบบจำลองความสัมพันธ์เอนทิตี

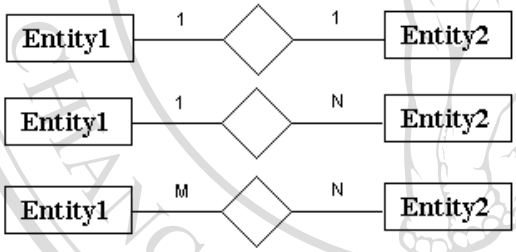
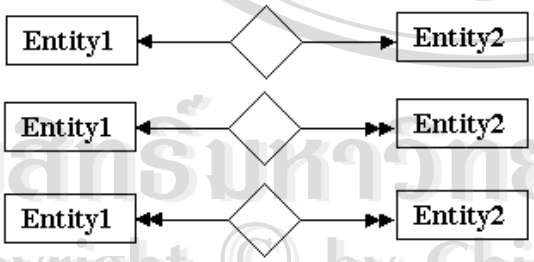
แบบจำลองความสัมพันธ์เอนทิตีหรือเรียกสั้น ๆ ว่าอี-อาร์ นำเสนอโดยปีเตอร์ เชน (Peter Chen) ในปีค.ศ.1976 และได้ถูกกำหนดมาตราฐานโดย ANSI ในปีค.ศ.1988 แบบจำลองความสัมพันธ์เอนทิตีใช้แผนภาพความสัมพันธ์เอนทิตี (Entity Relationship Diagram) ซึ่งเป็นแผนภาพที่แสดงความสัมพันธ์ของสิ่งต่าง ๆ เพื่อใช้เป็นเครื่องมือในการสื่อสารระหว่างผู้ออกแบบฐานข้อมูลและผู้ใช้ฐานข้อมูล โดยใช้ในขั้นตอนการวิเคราะห์ข้อมูลเพื่ออธิบายโครงสร้างของฐานข้อมูล โดยแสดงเอนทิตีทั้งหมดในฐานข้อมูลและความสัมพันธ์ระหว่างเอนทิตีเหล่านั้น (อัจฉราและคณะ, 2544:3-2)

สัญลักษณ์ที่ใช้แสดงส่วนประกอบต่าง ๆ ของแผนภาพความสัมพันธ์เอนทิตี มีดังนี้

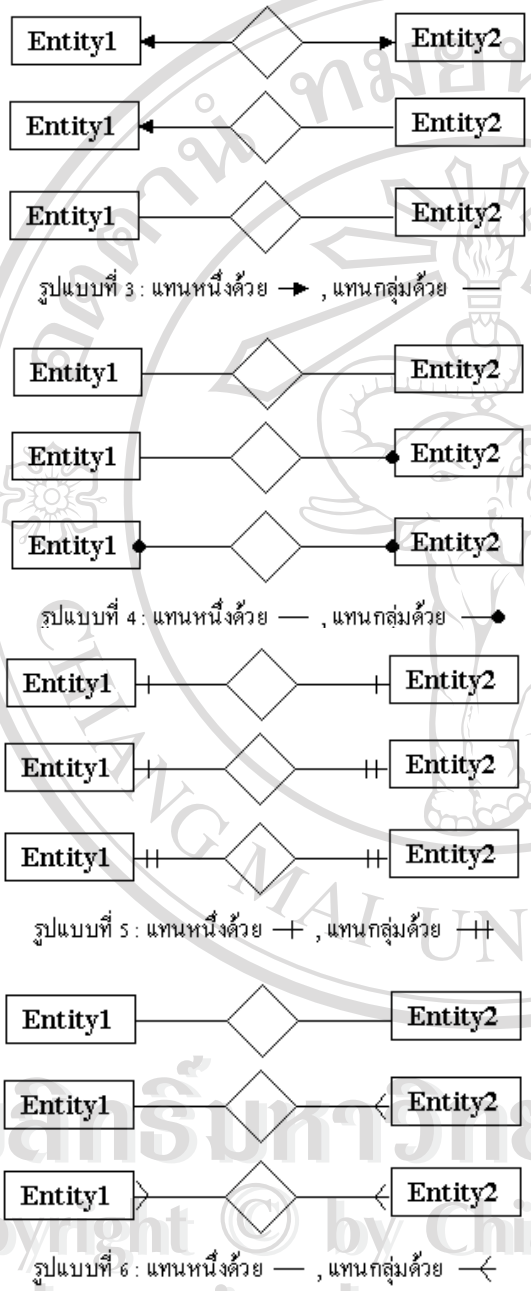
ตาราง 2.1 แสดงสัญลักษณ์ของแผนภาพความสัมพันธ์เอนทิตี

Entity	**เอนทิตีแบบธรรมดา** (Regular entity type หรือ Strong entity type) คือ กลุ่มของเอนทิตีที่มีคุณสมบัติเหมือนกันและมีคีย์หลักในการกำหนดความแตกต่างของสมาชิกแต่ละตัว โดยการคงอยู่ของเอนทิตีนี้ไม่ขึ้นอยู่กับเอนทิตีอื่น
Weak Entity	**เอนทิตีแบบอ่อน** (Weak entity type) คือ กลุ่มของเอนทิตีที่การคงอยู่ของสมาชิกขึ้นอยู่กับเอนทิตีอื่นและไม่มีคีย์หลักเป็นของตัวเอง สมาชิกของเอนทิตีแบบอ่อนจะสามารถแยกความแตกต่างได้ โดยอาศัยคีย์หลักของเอนทิตีแบบธรรมดามาประกอบกับแอททริบิวต์ใด ๆ (partial key) ในเอนทิตีแบบอ่อน
 (อ้างอิงจาก Elmasri, 2000 : 63)	**แอททริบิวต์** (Attribute) คือ คุณสมบัติหรือลักษณะของเอนทิตี โดยแบ่งเป็น 4 ประเภท 1. แอททริบิวต์ค่าเดียว (Single-valued attribute) คือ แอททริบิวต์ที่มีค่าของข้อมูลในแต่ละสมาชิกของเอนทิตีได้เพียงค่าเดียว 2. แอททริบิวต์หลายค่า (Multi-values attribute) คือ แอททริบิวต์ที่มีค่าของข้อมูลในแต่ละสมาชิกของเอนทิตีได้หลายค่า 3. แอททริบิวต์ผสม (Composite attribute) คือ แอททริบิวต์ที่ค่าภายในแอททริบิวต์นั้นสามารถแยกเป็นแอททริบิวต์ย่อยได้อีก

ตาราง 2.2 แสดงสัญลักษณ์ของแผนภาพความสัมพันธ์เอนทิตี (ต่อ)

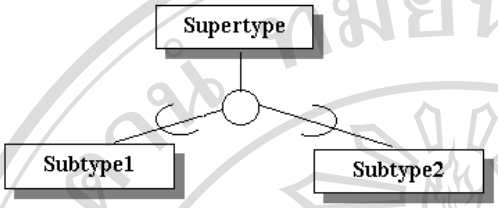
	4. แอททริบิวต์ที่แปลค่ามา (Derived attribute) คือ แอททริบิวต์ที่ค่าของข้อมูลในแต่ละสมาชิกของเอนทิตีได้มาจากการนำค่าของข้อมูลในแอททริบิวต์อื่นมาทำการคำนวณ
	**ความสัมพันธ์ระหว่างเอนทิตี** (Relationship) คือ ความสัมพันธ์ระหว่างเอนทิตีต่าง ๆ ภายในแผนภาพความสัมพันธ์เอนทิตี
	**Identifying relationship** คือ ความสัมพันธ์ระหว่างเอนทิตีธรรมดาและเอนทิตีแบบอ่อน
 รูปแบบที่ 1 : แทนหนึ่งด้วย 1, แทนกลุ่มด้วย M,N  รูปแบบที่ 2 : แทนหนึ่งด้วย → , แทนกลุ่มด้วย →→	**สัดส่วนความสัมพันธ์ระหว่างเอนทิตี** (Cardinality ratio) แบ่งได้เป็น 3 ประเภท 1. ความสัมพันธ์แบบหนึ่งต่อหนึ่ง (One-to-One relationship) เป็นความสัมพันธ์ที่แต่ละสมาชิกในเอนทิตีหนึ่งมีความสัมพันธ์กับสมาชิกในอีกหนึ่งเอนทิตี เพียงสมาชิกเดียว 2. ความสัมพันธ์แบบหนึ่งต่อกลุ่ม (One-to-Many relationship) เป็นความสัมพันธ์ที่แต่ละสมาชิกในเอนทิตีที่หนึ่งมีความสัมพันธ์กับหลายสมาชิกของเอนทิตีที่สอง แต่ละสมาชิกในเอนทิตีที่สองจะมีความสัมพันธ์กับเอนทิตีหนึ่งได้เพียงหนึ่งสมาชิก

ตาราง 2.3 แสดงสัญลักษณ์ของแผนภาพความสัมพันธ์เอนทิตี (ต่อ)

 (อ้างอิงจาก อัจฉราและคณะ, 2544 : 3-14 - 3-15)	3. ความสัมพันธ์แบบกลุ่มต่อกลุ่ม (Many-to-Many relationship) เป็นความสัมพันธ์ที่หลายสมาชิกในเอนทิตีหนึ่งมีความสัมพันธ์กับหลายสมาชิกของอีกเอนทิตีหนึ่ง แผนภาพความสัมพันธ์เอนทิตี มีการใช้สัญลักษณ์เพื่อแสดงสัดส่วนความสัมพันธ์ 3 ลักษณะนี้แตกต่างกันออกไป ซึ่งได้รวบรวมมาแสดงไว้ 6 รูปแบบ[1]

[1] สำหรับงานวิจัยนี้ จะใช้แบบจำลองความสัมพันธ์เอนทิตีในรูปแบบที่ 1

ตาราง 2.4 แสดงสัญลักษณ์ของแผนภาพความสัมพันธ์เอนทิตี (ต่อ)

 (อ้างอิงจาก Elmasri, 2000:77)	**เอนทิตีหลัก** (Supertype) คือ เอนทิตีที่สามารถจำแนกออกเป็นเอนทิตีย่อย ๆ ได้อีก **เอนทิตีย่อย** (Subtype) คือ เอนทิตีย่อยที่จำแนกออกมาจากเอนทิตีหลักและประกอบด้วยทุกแอททริบิวต์ที่มีในเอนทิตีหลัก โดยเอนทิตีย่อยสามารถมีแอททริบิวต์เพิ่มเติมจากเอนทิตีหลักได้ และเอน ทิตีย่อยจะได้รับคุณสมบัติของเอนทิตีหลักอย่างอัตโนมัติ ซึ่งเรียกว่าการรับคุณสมบัติถ่ายทอด

2.4 แบบจำลองฐานข้อมูลเชิงสัมพันธ์

แบบจำลองฐานข้อมูลเชิงสัมพันธ์ เป็นแบบจำลองฐานข้อมูลที่อยู่บนพื้นฐานความคิดทางคณิตศาสตร์เรื่องความสัมพันธ์ของเซ็ต เป็นงานวิจัยซึ่งถูกนำเสนอขึ้นประมาณ ค.ศ.1970 โดยดร. คอดด์ (Dr.Codd) (อัจฉราและคณะ, 2544:2-2)

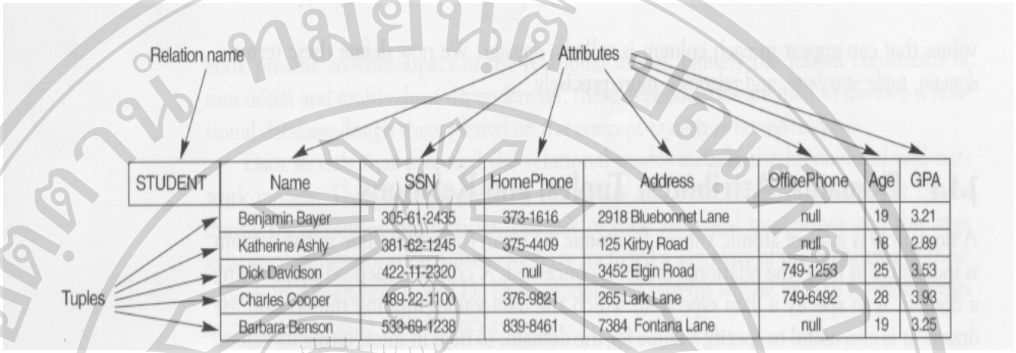
2.4.1 รีเลชัน (Relation)

แบบจำลองฐานข้อมูลเชิงสัมพันธ์จะมีการเก็บข้อมูลในรูปแบบของตารางสองมิติ ซึ่งสามารถเรียกได้อีกอย่างว่า รีเลชัน

รีเลชันจะประกอบไปด้วยแถวในแนวนอนซึ่งเรียกว่าทูเพิล (Tuple) และคอลัมน์ในแนวตั้งซึ่งเรียกว่าแอททริบิวต์ ดังรูป 2.2 ค่าและขอบเขตของข้อมูลที่เป็นไปได้ของแต่ละแอททริบิวต์เรียกว่าโดเมน (Domain) โดยแต่ละรีเลชันจะมีคุณสมบัติดังนี้คือ

1. ช่องแต่ละช่องของตารางจะเก็บข้อมูลเพียงค่าเดียว
2. ข้อมูลที่อยู่ในคอลัมน์เดียวกันจะต้องมีชนิดข้อมูลเป็นแบบเดียวกัน เช่นคอลัมน์รหัสคนงานจะต้องมีข้อมูลที่เป็นตัวเลขที่เป็นรหัสคนงานเท่านั้น
3. แต่ละคอลัมน์จะต้องมีชื่อคอลัมน์ที่แตกต่างกันและการเรียงลำดับของคอลัมน์ก่อนและหลังไม่ถือว่าสำคัญ

4. ข้อมูลแต่ละแถวของตารางจะต้องแตกต่างกันและการเรียงลำดับของแถวไม่ถือว่าสำคัญ (สมจิตรและงานนิจ อาจอินทร์, 2541:66)



STUDENT	Name	SSN	HomePhone	Address	OfficePhone	Age	GPA
	Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	null	19	3.21
	Katherine Ashly	381-62-1245	375-4409	125 Kirby Road	null	18	2.89
	Dick Davidson	422-11-2320	null	3452 Elgin Road	749-1253	25	3.53
	Charles Cooper	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
	Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	null	19	3.25

(อ้างอิงจาก Elmasri, 2000 : 198)

รูป 2.2 แสดงส่วนประกอบของรีเลชัน STUDENT

2.4.2 คีย์ (Key)

คุณสมบัติหนึ่งที่สำคัญของรีเลชันคือ ความเป็นเอกลักษณ์ (Uniqueness property) สิ่งที่ใช้กำหนดความเป็นเอกลักษณ์ของทูเพิลในรีเลชัน เรียกว่า คีย์

ฐานข้อมูลหนึ่ง ๆ จะมีข้อมูลอยู่มากมาย ยิ่งฐานข้อมูลมีขนาดใหญ่แสดงว่ายิ่งมีจำนวนข้อมูลมาก ข้อมูลเหล่านี้อาจมีค่าแตกต่างกัน คล้ายกันหรือแม้กระทั่งเหมือนกัน ทำให้การแยกแยะโดยอาศัยเพียงตัวข้อมูลอย่างเดียวทำได้อย่างยากลำบาก ดังนั้นจึงมีการกำหนดค่าคีย์ ประจำข้อมูลเพื่อทำให้การแยกแยะข้อมูลในฐานข้อมูลเป็นไปอย่างถูกต้อง

(1) คีย์หลัก (Primary Key) เป็นคีย์เดี่ยวหรือคีย์ผสม (Single or Composite key) โดยแอทริบิวต์ที่มีคุณสมบัติของข้อมูลที่เป็นค่าเอกลักษณ์หรือมีค่าที่ไม่ซ้ำซ้อน คุณสมบัติดังกล่าวจะสามารถระบุว่าข้อมูลนั้นเป็นข้อมูลของทูเพิลใด การเลือกคีย์หลักสามารถเลือกได้จากทูเพิลใด ๆ ก็ได้ที่ไม่มีโอกาสซ้ำซ้อนกันในฐานข้อมูลนั้น

(2) คีย์นอก (Foreign key) เป็นคีย์เดี่ยวหรือคีย์ผสม ซึ่งปรากฏเป็นคีย์ทั่วไปของรีเลชันหนึ่ง แต่ไปปรากฏเป็นคีย์หลักในอีกรีเลชันหนึ่ง คีย์นอกเป็นอีกคีย์หนึ่งที่มีความสำคัญในฐานข้อมูลเชิงสัมพันธ์ เนื่องจากเป็นตัวที่ใช้สร้างความสัมพันธ์ระหว่างรีเลชัน การเปลี่ยนแปลงค่าของคีย์นอกจะต้องอาศัยความระมัดระวังเป็นอย่างมากเนื่องจากจะมีผลกระทบโดยตรงต่อข้อมูลในรีเลชันอื่น

ที่มีการอ้างอิงถึงคีย์นอกตัวนี้ จึงมีกฎและเงื่อนไขที่บังคับใช้เพื่อทำให้ข้อมูลมีความถูกต้องอยู่เสมอ

การกำหนดค่าให้กับคีย์นอกของรีเลชันที่อ้างอิงถึง จะต้องกำหนดค่าของคีย์ให้อยู่ในโดเมนเดียวกันกับรีเลชันที่คีย์นอกนั้นเป็นคีย์หลัก แต่คีย์นอกนั้นไม่จำเป็นจะต้องเป็นส่วนหนึ่งในคีย์หลักของรีเลชันอื่น (ศิรินุช เทียนรุ่งโรจน์, 2543:43-46)

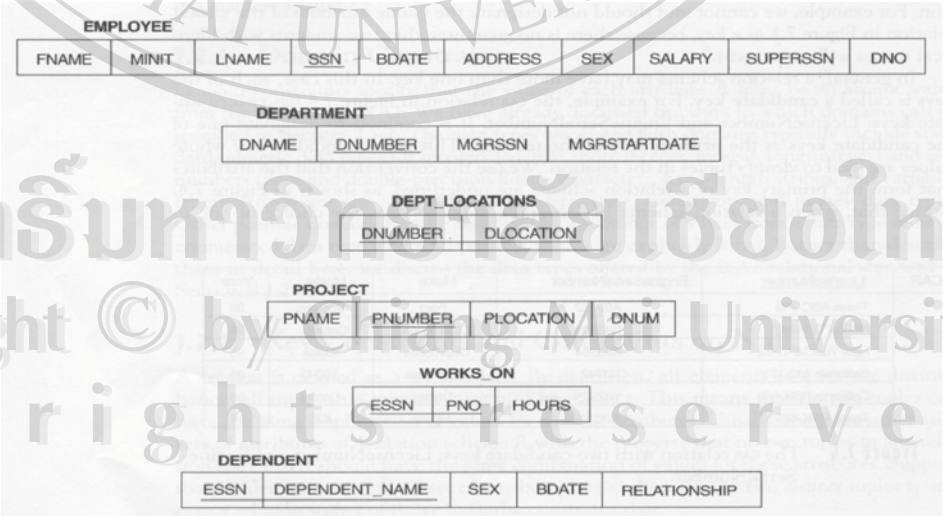
(3) คีย์คู่แข่ง (Candidate key) เป็นแอททริบิวต์หรือกลุ่มแอททริบิวต์ที่สามารถจะเป็นคีย์หลักได้ คีย์คู่แข่งที่ไม่ได้รับเลือกเป็นคีย์หลักจะเรียกว่าคีย์สำรอง(Alternate key) ในรีเลชัน (อัจฉราและคณะ, 2544:2-8)

(4) นันคีย์แอททริบิวต์ (Nonkey Attribute) เป็นแอททริบิวต์อื่น ๆ ในรีเลชันที่ไม่ใช่ส่วนใดส่วนหนึ่งของคีย์หลักในรีเลชันนั้น (อัจฉราและคณะ, 2544:2-9)

2.4.3 เค้าร่างฐานข้อมูลเชิงสัมพันธ์ (Relational database schema)

เค้าร่างของฐานข้อมูล (Database schema) คือ รายละเอียดของโครงสร้างฐานข้อมูลที่กำหนดตารางต่าง ๆ ในฐานข้อมูลและแต่ละตารางประกอบด้วยฟิลด์อะไร รวมถึงการกำหนดความสัมพันธ์ของข้อมูลระหว่างตาราง โดยทั่วไปเค้าร่างฐานข้อมูลมักจะไม่เปลี่ยนแปลงบ่อยนัก

เค้าร่างฐานข้อมูลเชิงสัมพันธ์ ประกอบด้วยเค้าร่างรีเลชันหลาย ๆ รีเลชัน เค้าร่างรีเลชันจะเขียนโดยใช้ชื่อความสัมพันธ์ตามด้วยวงเล็บของชื่อหลักและขีดเส้นใต้คีย์หลัก ดังรูป 2.3



(อ้างอิงจาก Elmasri, 2000 : 204)

รูป 2.3 แสดงตัวอย่างของเค้าร่างของฐานข้อมูลเชิงสัมพันธ์

2.5 แนวคิดเชิงวัตถุ (Object-Oriented concept)

แนวคิดเชิงวัตถุมุ่งเน้นสิ่งต่าง ๆ ที่ใกล้เคียงกับโลกแห่งความจริงในลักษณะรูปธรรม โดยจะมองระบบเป็นกลุ่มของออบเจ็กต์ที่มีปฏิกิริยาต่อกัน มีการนำข้อมูลและฟังก์ชันการทำงานรวมเข้าด้วยกันในออบเจ็กต์ ทำให้ข้อมูลที่เป็นออบเจ็กต์นั้นสามารถอธิบายคุณสมบัติรวมทั้งมีฟังก์ชันการทำงานในตัวเองได้

แนวคิดเชิงวัตถุได้เข้าไปปรากฏในภาษาโปรแกรมปลายทศวรรษ 1960 เพื่อให้สามารถทำงานแบบเชิงวัตถุอันประกอบไปด้วยความสามารถหลักที่เกี่ยวกับการห่อหุ้ม(Encapsulation) การรับทอดคุณสมบัติ (Inheritance) และการเปลี่ยนรูป (Polymorphism) แนวคิดในการสร้างโปรแกรมเชิงวัตถุที่เกิดขึ้นนี้ ใช้เวลาค่อนข้างมาก กว่าจะเป็นที่ยอมรับโดยทั่วไป เพราะเป็นการปฏิวัติแนวทางการพัฒนาโปรแกรมในหลายด้าน ส่วนสำคัญที่ทำให้วิธีการเขียนโปรแกรมเชิงวัตถุเข้ามามีบทบาทมากขึ้นอย่างเห็นได้ชัด คือ การพัฒนาภาษาซีพลัสพลัส ในช่วงต้นทศวรรษที่ 1980 ทำให้มีภาษาโปรแกรมเชิงวัตถุที่ทำงานได้ดีและสะดวกขึ้น อีกทั้งทำให้ผู้เขียนภาษาซีที่มีอยู่แล้วจำนวนมาก สามารถเรียนรู้แนวทางใหม่ของการโปรแกรมเชิงวัตถุได้ง่าย

ระบบคอมพิวเตอร์ใหม่ ๆ ที่เกิดขึ้นในช่วงทศวรรษที่ 1980 เป็นเครื่องแสดงให้เห็นถึงความสำคัญของแนวคิดเชิงวัตถุ ซึ่งจะทำให้ผู้เขียนโปรแกรมสามารถสร้างโปรแกรมโดยการใช้ชิ้นส่วนสำเร็จรูปมาประกอบกันเป็นโปรแกรม อันเป็นวิธีที่ทำให้การสร้างโปรแกรมขนาดใหญ่เป็นไปได้ง่ายขึ้น เชื่อถือได้มากขึ้น แก้ไขเปลี่ยนแปลงได้ง่ายขึ้นและสร้างได้รวดเร็วขึ้นกว่าวิธีแบบเดิมที่ใช้กันมา แม้ว่าวิธีการเชิงวัตถุจะไม่ใช่แนวทางที่จะแก้ปัญหาทางซอฟต์แวร์ได้ทุกอย่าง แต่ได้แสดงให้เห็นความสามารถอย่างสำคัญในการช่วยลดภาระ ลดปัญหา และลดค่าใช้จ่ายในการสร้างซอฟต์แวร์ขนาดใหญ่ได้เป็นอย่างดี (รอม หิรัญพฤกษ์, 2536:191-192)

2.5.1 ออบเจ็กต์

เป็นสิ่งที่มีทั้งข้อมูลและพฤติกรรม ออบเจ็กต์เป็นสิ่งที่ใช้แทนสิ่งที่มีอยู่ในโลกความจริง (อภิเนตร อูนากูล, 2543:21) เช่น สุนัข ระบบควบคุมการไหลของอากาศ เซนเซอร์หรือเครื่องยนต์หรืออาจจะใช้แทนสิ่งที่เป็นนามธรรม เช่น ความเป็นเจ้าของ เที่ยวบิน การวิ่ง หรือออบเจ็กต์

สามารถเป็นสิ่งที่มองเห็นได้เช่น จักรยาน รถ องค์กร ใบรายการสินค้า เป็นต้น โดยออบเจ็กต์ทั้งหมดนี้จะเป็นสิ่งที่ผู้พัฒนาสนใจ ซึ่งออบเจ็กต์แบบใดก็ตามล้วนมีลักษณะ ดังนี้

(1) แอททริบิวต์ คือ คุณสมบัติของออบเจ็กต์ที่เก็บไว้เพื่อที่จะให้ผู้อื่นเรียกไปใช้งาน เก็บไว้ใช้ในการประมวลผล เพื่อตอบสนองกับเหตุการณ์ภายนอกหรืออาจจะเก็บไว้เพื่อเป็นสถานะของตัวออบเจ็กต์เอง โดยทั่วไปแอททริบิวต์ของออบเจ็กต์จะถูกซ่อนไว้จากผู้ที่จะใช้งานออบเจ็กต์ ทั้งนี้เนื่องจากการเข้าใช้งานแอททริบิวต์โดยตรงเปรียบเสมือนเป็นการเข้าถึงโครงสร้างของวัตถุ (อภิเนตร อูนากูล, 2543:26) ตัวอย่างออบเจ็กต์ที่พบเห็นอยู่บ่อย ๆ เช่น ออบเจ็กต์รถยนต์มีแอททริบิวต์ เช่น ความกว้าง ความยาว สี น้ำหนัก เป็นต้น

(2) เมธอด (Method) คือ พฤติกรรม (Behavior) หรือบริการ ที่ออบเจ็กต์สามารถกระทำให้ได้ เช่น ออบเจ็กต์รถยนต์ อาจจะมีเมธอด สตาร์ทเครื่องยนต์ ออกวิ่ง เบรก ดับเครื่องยนต์ เปิดไฟยกเลี้ยว เป็นต้น ทั้งนี้จุดประสงค์หลักของเมธอดก็เพื่อใช้ในการจัดการกับแอททริบิวต์นั่นเอง (ชาลีและเทพฤทธิ์, 2544:16)

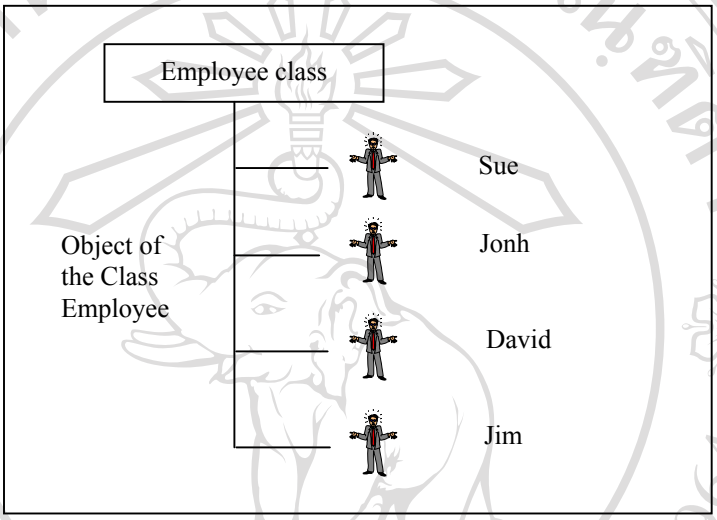
(3) สถานะ(State) คือ สถานะของออบเจ็กต์ ณ เวลาหนึ่ง โดยที่สถานะของออบเจ็กต์สามารถเปลี่ยนแปลงได้จากเดิม เมื่อมีเงื่อนไขบางประการ เช่น ออบเจ็กต์รถยนต์ มีสถานะของรถยนต์ เช่น กำลังออกตัว กำลังลดความเร็ว เป็นต้น

(4) เอกลักษณ์เฉพาะตัวของออบเจ็กต์หรือโอไอดี (Object Identity:OID) คือ คุณลักษณะบางอย่างที่ทำให้ออบเจ็กต์แต่ละออบเจ็กต์แตกต่างกัน โดยระบบจะกำหนดโอไอดีให้กับออบเจ็กต์แต่ละตัว โอไอดีไม่ขึ้นอยู่กับค่าแอททริบิวต์ใด ๆ ในออบเจ็กต์ ณ เวลาที่ออบเจ็กต์ถูกสร้างขึ้นมาและไม่สามารถเปลี่ยนแปลงได้ จะถูกลบได้ก็ต่อเมื่อออบเจ็กต์นั้นถูกล้างออกไปเท่านั้นและไม่มีการนำโอไอดีนั้นกลับมาใช้อีก มีลักษณะแตกต่างจากคีย์หลักที่ใช้ในแบบจำลองเชิงสัมพันธ์ตรงที่ คีย์หลักขึ้นอยู่กับการกำหนดค่าจากผู้ใช้ที่จะเลือกแอททริบิวต์ใดเป็นคีย์ก็ได้ และสามารถเปลี่ยนแปลงได้ตลอดเวลา (อัจฉราและคณะ, 2544:1-58)

(5) ข้อความ (Message) คือ สารที่ส่งจากออบเจ็กต์หนึ่งไปยังอีกออบเจ็กต์หนึ่ง เพื่อให้ออบเจ็กต์อื่นทำงานตามที่ต้องการหรือการเรียกใช้งานเมธอดของออบเจ็กต์นั่นเอง

2.5.2 คลาส

กลุ่มของออบเจ็กต์ที่มีโครงสร้างพื้นฐานคุณสมบัติและพฤติกรรมเดียวกัน ดังนั้น ออบเจ็กต์ที่มีคุณสมบัติลักษณะเดียวกันนี้จะรวมกลุ่มอยู่ในคลาสเดียวกัน หรืออาจกล่าวได้ว่าคลาสคือต้นแบบข้อมูลที่มีไว้เพื่อสร้างออบเจ็กต์ ดังรูป 2.4



รูป 2.4 แสดงตัวอย่างออบเจ็กต์ที่อยู่ในคลาส Employee

2.5.3 ความสัมพันธ์ระหว่างคลาส

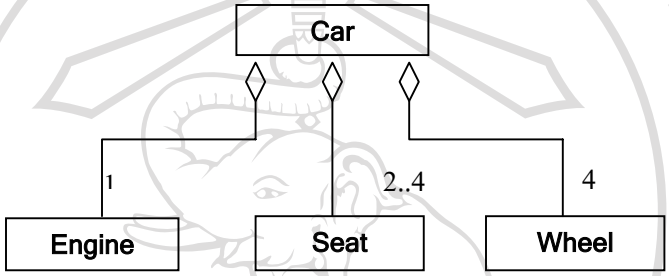
ความสัมพันธ์ระหว่างคลาส สามารถแบ่งได้ออกเป็น 3 ลักษณะด้วยกันคือ

(1) แอสโซซิเอชัน (Association) คือ ความสัมพันธ์ระหว่างคลาสหนึ่งกับอีกคลาสหนึ่ง แต่ไม่ได้มีความสัมพันธ์แบบเป็นส่วนหนึ่งส่วนใด เช่นความสัมพันธ์ระหว่างนักบินกับเครื่องบิน (โอภาส เอี่ยมสิริวงศ์, 2544:174) ดังตัวอย่างรูป 2.5

รูป 2.5 แสดงตัวอย่างความสัมพันธ์แบบแอสโซซิเอชัน

(2) อะกรีเกชัน(Aggregation) คือ ความสัมพันธ์ในลักษณะที่ออบเจ็กต์หนึ่งเป็นส่วนหนึ่งของอีกออบเจ็กต์หนึ่ง (สุนทริน วงศ์ศิริกุล, 2545:13) แบ่งออกเป็น 2 รูปแบบคือ

แชร์ อะกรีเกชัน(Shared Aggregation) คือ ความสัมพันธ์แบบเป็นส่วนหนึ่งของ (a part of) โดยจะมีคลาสที่ใหญ่ที่สุดเป็นออบเจ็กต์หลัก และมีคลาสหรือออบเจ็กต์ส่วนอื่น ๆ เป็นส่วนหนึ่งของออบเจ็กต์หลัก เช่น รถยนต์ มีส่วนประกอบคือ เครื่องยนต์ ที่นั่ง และล้อ เป็นต้น ดังตัวอย่างรูป 2.6

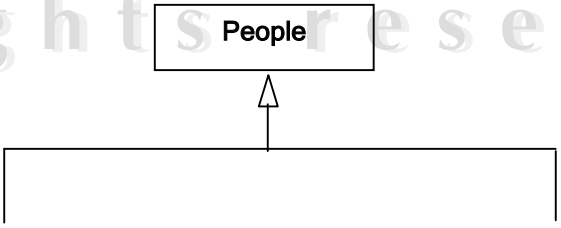


รูป 2.6 แสดงตัวอย่างความสัมพันธ์แบบอะกรีเกชัน

คอมโพสิต อะกรีเกชัน(Composite Aggregation) คือ ความสัมพันธ์ที่มีความข้องเกี่ยวกันเสมอ เช่น คลาสปุ่มไม่สามารถมีได้ หากไม่มีคลาสวินโดว์ เป็นต้น ดังตัวอย่างรูป 2.7

รูป 2.7 แสดงตัวอย่างความสัมพันธ์แบบคอมโพสิต อะกรีเกชัน

(3) เจเนอรัลไลเซชัน(Generalization) คือ ความสัมพันธ์ระหว่างคลาสในลักษณะของการรับทอดคุณสมบัติ จากโครงสร้างคลาสหนึ่งไปยังโครงสร้างอีกคลาสหนึ่ง (โอภาส เอี่ยมสิริวงค์, 2544:175-176) ดังตัวอย่างรูป 2.8



รูป 2.8 แสดงความสัมพันธ์แบบเจเนอรัลไลเซชัน

2.5.4 คุณลักษณะของวิธีการเชิงวัตถุ

(1) การกำหนดสาระสำคัญ (Abstraction) คือ การตีกรอบแนวความคิดเพื่อให้มองเห็นสิ่งที่เกี่ยวข้องในระบบที่สนใจ และลดความซ้ำซ้อนของสิ่งที่จะนำมาใช้ระบบ เช่น เมื่อพูดถึงคอมพิวเตอร์ บางคนอาจจะคิดถึงเครื่องคอมพิวเตอร์ บางคนอาจจะคิดถึงเครื่องโน้ตบุ๊ค เป็นต้น

(2) การห่อหุ้ม คือ การรวบรวมเมธอดและแอททริบิวต์เข้าเป็นหน่วยเดียวกันเพื่อที่แอททริบิวต์สามารถถูกเปลี่ยนแปลงได้อย่างเหมาะสมโดยผ่านเมธอด และเราจะเรียกผลที่เกิดจากการใช้งานการห่อหุ้มว่า การซ่อนข้อมูล (Information Hiding)

(3) การรับทอดคุณสมบัติ คือ วิธีการในการสร้างคลาสใหม่จากคลาสเดิมที่มีอยู่ ทั้งนี้คลาสที่สร้างขึ้นใหม่จะมีวัตถุประสงค์ในการทำงานที่เฉพาะเจาะจงมากยิ่งขึ้น ซึ่งคลาสที่ถ่ายทอดคุณสมบัติจะเรียกว่าซูเปอร์คลาสหรือคลาสแม่ และเรียกคลาสที่ได้รับการสืบทอดคุณสมบัติว่าซับคลาสหรือคลาสลูก เช่น เมื่อพูดถึงสัตว์ อาจแบ่งเป็น 2 ประเภทคือ แมลงและสัตว์เลี้ยงลูกด้วยนม ในส่วนของแมลงก็อาจจำแนกได้เป็น ผึ้ง ผีเสื้อ ตั๊กแตน ในส่วนของสัตว์เลี้ยงลูกด้วยนมก็อาจจำแนกเป็น กระต่าย ช้าง แมว เป็นต้น

(4) การเปลี่ยนรูป คือ การที่เมธอดเดียวกันมีพฤติกรรมที่แตกต่างกันเมื่อถูกใช้กับออบเจ็กต์ที่เกิดจากคนละคลาสกัน กล่าวอีกนัยหนึ่งคือออบเจ็กต์ที่เกิดจากต่างคลาสกันสามารถที่จะมีปฏิกิริยาตอบสนองต่อเมธอดชื่อเดียวกันได้อย่างแตกต่างกัน เช่น เมธอดที่ออกแบบไว้สำหรับการรับค่าพารามิเตอร์จำนวน 3 ค่า แต่สามารถเรียกใช้งานได้แม้วาจำนวนพารามิเตอร์จะไม่ครบก็ตาม นอกจากนั้นแล้วการโปรแกรมเชิงวัตถุยังยอมให้ผู้ใช้สามารถกำหนดใช้เมธอดที่มีชื่อเดียวกัน แต่มีจำนวนพารามิเตอร์ที่แตกต่างกันได้ (Edward Yourdon ,1994 : 6-8)

2.5.5 ข้อดีของการประยุกต์ใช้แนวคิดเชิงวัตถุ

(1) สนับสนุนการพัฒนาระบบที่ซับซ้อน ในการพัฒนาโปรแกรมรูปแบบเดิม นักพัฒนาโปรแกรมจะทำงานกันที่ระดับฟังก์ชัน กล่าวคือตั้งแต่เริ่มการพัฒนาจนจบ จะเป็นการสร้าง ปรับปรุง แก้ไข รายละเอียดของฟังก์ชันต่าง ๆ ภายในตัวโปรแกรม แต่สำหรับแนวคิดเชิงวัตถุ นัก

พัฒนาจะทำการวิเคราะห์ออกแบบระบบงานกันในระดับของออบเจ็กต์ซึ่งประกอบด้วยทั้งข้อมูลและเมธอดภายในแต่ละออบเจ็กต์ ดังนั้นจึงกล่าวได้ว่าการอาศัยแนวคิดเชิงวัตถุจะช่วยให้นักพัฒนาสามารถสร้างโปรแกรมได้ง่ายขึ้นมาก

(2) สนับสนุนการนำกลับมาใช้งานซ้ำอีกครั้ง เนื่องจากแต่ละคลาสหรือออบเจ็กต์ที่กำหนดขึ้นนั้นจะมีความสมบูรณ์อยู่ในตัวมันเองบนพื้นฐานแนวคิดของแต่ละออบเจ็กต์เอง รวมทั้งยังเป็นอิสระจากสภาพแวดล้อมอื่น ดังนั้นแต่ละคลาสจึงง่ายต่อการนำกลับมาใช้งานใหม่ การนำกลับมาใช้งานอาจอยู่ในรูปแบบของการสืบทอดคุณสมบัติระหว่างออบเจ็กต์ หรือการใช้งานซอฟต์แวร์คอมโพเนนต์ก็เช่นกัน

(3) ปรับปรุงแก้ไขและบำรุงรักษาง่าย เนื่องจากข้อมูลและเมธอดการทำงานที่เกี่ยวข้องกับออบเจ็กต์หนึ่ง ๆ จะถูกรวบรวมอยู่ที่เดียวกัน การทำงานภายในของแต่ละออบเจ็กต์จะไม่เกี่ยวข้องพึ่งพิงกับโค้ดที่อยู่ภายนอกออบเจ็กต์ ดังนั้นนักพัฒนาสามารถทำการแก้ไขปรับปรุงรายละเอียดภายในของแต่ละคลาสได้โดยไม่กระทบต่อส่วนที่เรียกใช้งานภายนอกแต่อย่างใด นอกจากนี้ในการขยายระบบก็สามารถทำได้ง่าย โดยการสร้างออบเจ็กต์หรือคลาสเพิ่มเติมลงไปในตัวโปรแกรมนั่นเอง (ชาลีและเทพฤทธิ์,2544:18)

2.5.6 ข้อจำกัดของการประยุกต์ใช้แนวคิดเชิงวัตถุ

เนื่องจากแนวคิดเชิงวัตถุเป็นการปฏิวัติแนวทางการพัฒนาโปรแกรมในหลายด้าน ไม่ว่าจะเป็นในด้านแนวความคิด การวิเคราะห์และการออกแบบ ขั้นตอนการทำงาน สิ่งเหล่านี้เป็นเรื่องใหม่สำหรับนักพัฒนาที่เคยชินกับการพัฒนาระบบด้วยวิธีการแบบดั้งเดิม ทำให้ต้องใช้เวลาในการเรียนรู้พอสมควร

2.6 แบบจำลองเชิงวัตถุด้วยยูเอ็มแอล

เทคโนโลยีในการพัฒนาระบบด้วยวิธีการเชิงวัตถุเป็นหลักการทางวิศวกรรม ซึ่งบรรจุไว้ด้วยองค์ประกอบต่าง ๆ ที่เรียกว่าแบบจำลองเชิงวัตถุ แบบจำลองเชิงวัตถุจะอธิบายถึงออบเจ็กต์ซึ่งเป็นที่รวมของข้อมูล ความสัมพันธ์ระหว่างออบเจ็กต์และการประมวลผลในระบบที่สนใจ

สำหรับงานวิจัยนี้เลือกใช้คลาสไดอะแกรมของยูเอ็มแอล สำหรับใช้ในการแสดงผลลัพธ์ของเค้าร่างฐานข้อมูลเชิงวัตถุ

2.6.1 ยูเอ็มแอล (Unified Modeling Language:UML)

(1) แบบจำลองในวลี “ภาษาแบบจำลอง” (Modeling Language) คือ ความพยายามในการที่จะอธิบายปัญหาของซอฟต์แวร์ที่จะดำเนินการพัฒนาขึ้นมา ตัวแบบจำลองจะแสดงให้เห็นถึงออบเจ็กต์ต่าง ๆ ที่เกี่ยวข้อง และความสัมพันธ์ระหว่างออบเจ็กต์เหล่านั้น นอกจากนั้นแบบจำลองยังแสดงให้เห็นถึงวิธีการที่จะแก้ไขปัญหา เราอาจจะใช้ไดอะแกรม เนื้อความ (Text) หรือรูปแบบอื่น ๆ ซึ่งเป็นที่ยอมรับกันระหว่างผู้พัฒนาและผู้ใช้ระบบในการนำเสนอแบบจำลอง ๆ หนึ่ง ดังนั้นเมื่อกล่าวถึงภาษาแบบจำลอง จะหมายถึงภาษาที่เอาไว้อธิบายแบบจำลองนั่นเอง ภาษาแบบจำลองทั้งหลายมักจะใช้ไดอะแกรมหรือเนื้อความในการอธิบายถึงออบเจ็กต์และความสัมพันธ์ระหว่างออบเจ็กต์เหล่านั้น (บรรจงและญาณวรรณ, 1999:185)

(2) ยูเอ็มแอล เป็นภาษาลักษณ์รูปภาพมาตรฐานสำหรับใช้ในการสร้างแบบจำลองเชิงวัตถุ และจัดการเอกสารต่าง ๆ ในการผลิตระบบซอฟต์แวร์ โดยทำให้การออกแบบซอฟต์แวร์หรือการสร้างพิมพ์เขียวของซอฟต์แวร์ทำได้ง่าย ผู้คิดค้นยูเอ็มแอลเกิดจากการพัฒนาของทีมบุคลากรแห่งบริษัท เรชันนอลซอฟต์แวร์และความร่วมมือจากบริษัทซอฟต์แวร์และคอมพิวเตอร์ต่าง ๆ เช่น ไอบีเอ็ม ไมโครซอฟต์ ออราเคิล ฮิวเล็ตแพ็กการ์ด ออบเจ็กต์ไทม์และยูนีซีส

ยูเอ็มแอล เป็นภาษาแบบจำลองภาษาหนึ่ง ซึ่งสามารถใช้ในการแก้ไขปัญหาในการดำเนินงานโครงการซอฟต์แวร์ ในการแก้ปัญหาหนึ่ง ๆ ยูเอ็มแอลจะใช้แบบจำลองที่มีรูปแบบแตกต่างกัน แบบจำลองที่ยูเอ็มแอลใช้จะมีลักษณะต่อเนื่องกันไป คือแบบจำลองหนึ่งจะอาศัยแบบจำลองที่สร้างขึ้นมาก่อนหน้านี้เพื่อทำการสร้างแบบจำลองต่อไป โดยแต่ละแบบจำลองจะมีมุมมองของปัญหาในแง่ที่แตกต่างกันออกไป แต่เมื่อเอาแบบจำลองเหล่านั้นมาประกอบกันเข้า ก็จะสามารถดำเนินการวิเคราะห์ ออกแบบ และพัฒนาซอฟต์แวร์ได้อย่างมีประสิทธิภาพ

2.6.2 ประเภทแผนภาพของยูเอ็มแอล

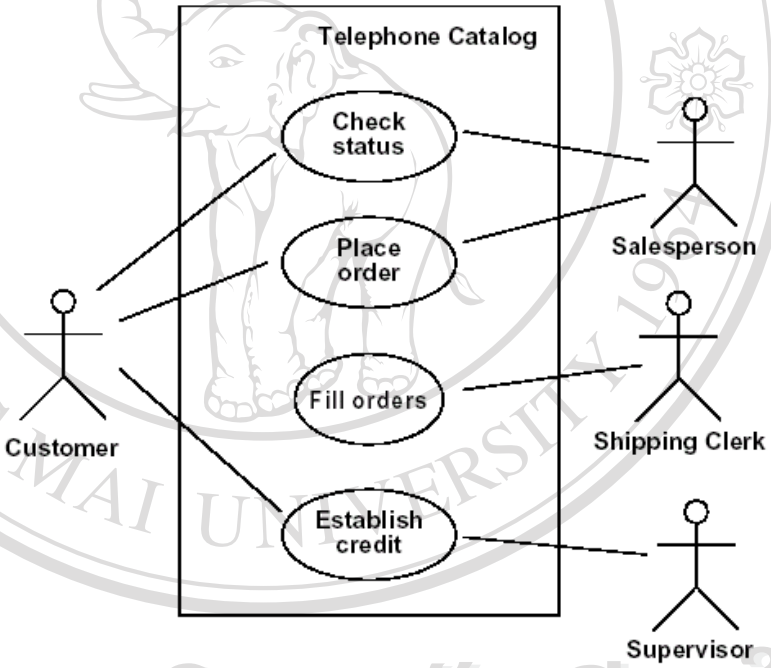
ยูเอ็มแอล ประกอบด้วยแผนภาพทั้งหมด 9 แผนภาพ เพื่อใช้ในการจำลองระบบงานเปรียบได้กับการมองในแง่มุมต่าง ๆ (แผนภาพแต่ละชนิด) เพื่อให้สามารถเข้าใจระบบงานให้มากที่สุด ผู้ใช้จำลองแบบไม่จำเป็นต้องใช้ทุกแผนภาพ สามารถเลือกใช้แผนภาพที่เหมาะสม

ยูเอ็มแอล ประกอบด้วยแผนภาพต่าง ๆ[2] ดังนี้

[2] เนื่องจากงานวิจัยนี้ใช้เฉพาะคลาสไดอะแกรม ส่วนแผนภาพอื่นๆ ผู้อ่านท่านใดสนใจสามารถหารายละเอียดเพิ่มเติมได้ http://www.omg.org/technology/documents/formal/uml_2.htm

(1) ยูสเคสไดอะแกรม (Use-case Diagram) ใช้มองภาพรวมของระบบและความต้องการต่าง ๆ ซึ่งคล้ายกับแผนภาพการไหลของข้อมูล ซึ่งสามารถบอกได้ว่า ใครเกี่ยวข้องกับงานใด ระบบอะไรและมีงานหลัก ๆ อะไรบ้าง และใช้สำหรับกำหนดความต้องการของระบบผ่านมุมมองของผู้ใช้ โดยเน้นไปที่ความสัมพันธ์ที่เกิดขึ้นในระบบ

ยูสเคสไดอะแกรม มีองค์ประกอบ 2 ส่วนคือ แอ็กเตอร์ (actor) และยูสเคส (Use-case) แอ็กเตอร์ คือ สิ่งที่อยู่นอกระบบแต่เป็นผู้ให้อะไรบางอย่างแก่ระบบ ส่วนยูสเคสจะแสดงถึงขอบเขตของระบบที่เรากำลังสนใจ ดังรูป 2.9

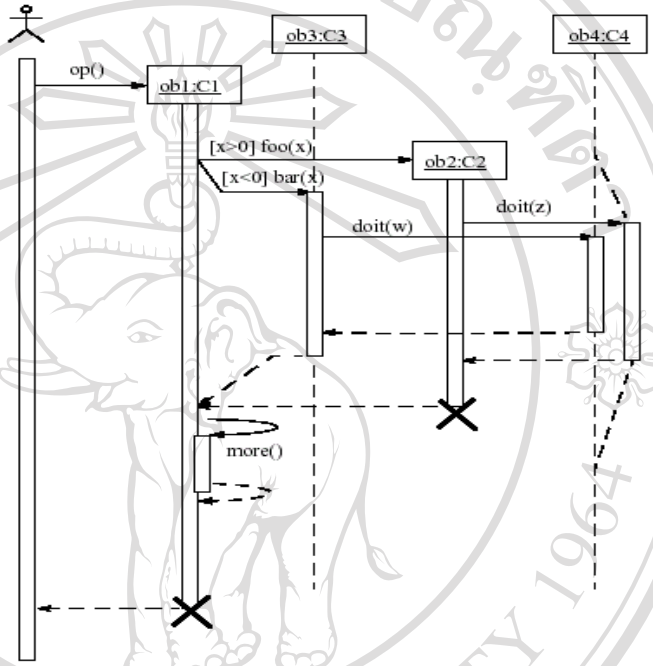


(อ้างอิงจาก http://www.omg.org/technology/documents/formal/uml_2.htm)

รูป 2.9 แสดงตัวอย่างยูสเคสไดอะแกรม

(2) ซีเควนไดอะแกรม (Sequence Diagram) เป็นแผนภาพที่แสดงขั้นตอนการทำงานของแต่ละยูสเคสระหว่างออบเจ็กต์ต่าง ๆ ที่ส่งข้อความถึงกันและกัน โดยแผนภาพนี้จะช่วยให้โปรแกรมเมอร์เห็นภาพรวม ทำให้ง่ายต่อความเข้าใจในการเขียนและควบคุมโปรแกรมตามที่ออก

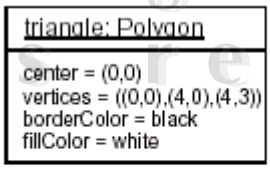
แบบไว้ อย่างไรก็ตามซีเควนไดอะแกรม จะไม่ได้แสดงความสัมพันธ์ระหว่างออบเจ็กต์ ตัวอย่างซีเควนไดอะแกรมแสดงดังรูป 2.10

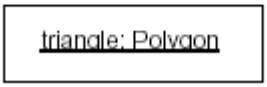


(อ้างอิงจาก http://www.omg.org/technology/documents/formal/uml_2.htm)

รูป 2.10 แสดงตัวอย่างซีเควนไดอะแกรม

(3) ออบเจ็กต์ไดอะแกรม (Object Diagram) เป็นแผนภาพที่แสดงความสัมพันธ์ระหว่างออบเจ็กต์ที่เชื่อมโยงกันในช่วงเวลาหนึ่งเท่านั้น โดยสัญลักษณ์ของออบเจ็กต์ไดอะแกรมจะมีลักษณะเดียวกับคลาสไดอะแกรม ต่างกันที่ชื่อของออบเจ็กต์ไดอะแกรมจะมีการขีดเส้นใต้เอาไว้ด้วย ดังรูป 2.11

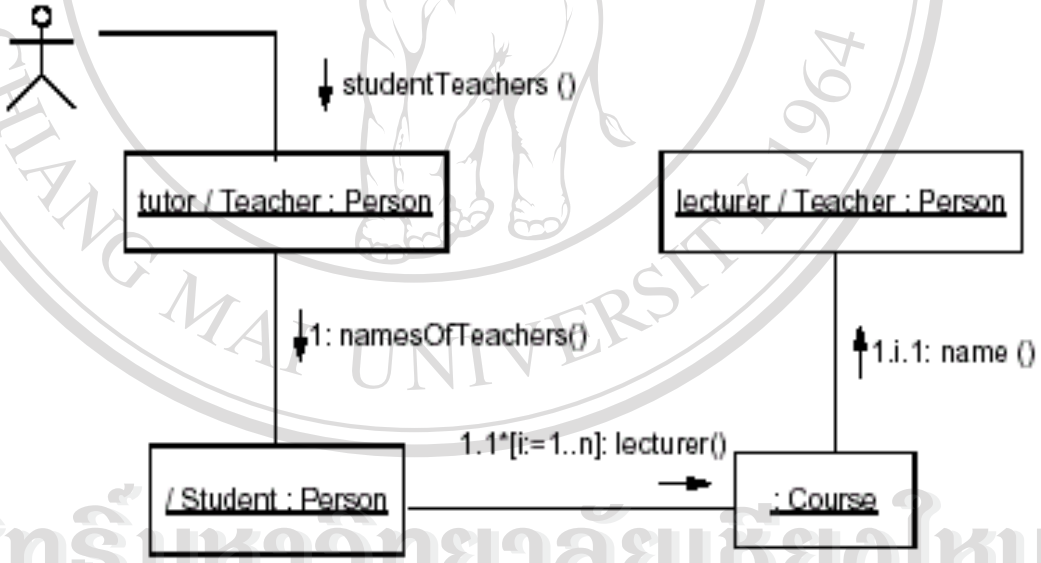




(อ้างอิงจาก http://www.omg.org/technology/documents/formal/uml_2.htm)

รูป 2.11 แสดงตัวอย่างออบเจ็กต์ไดอะแกรม

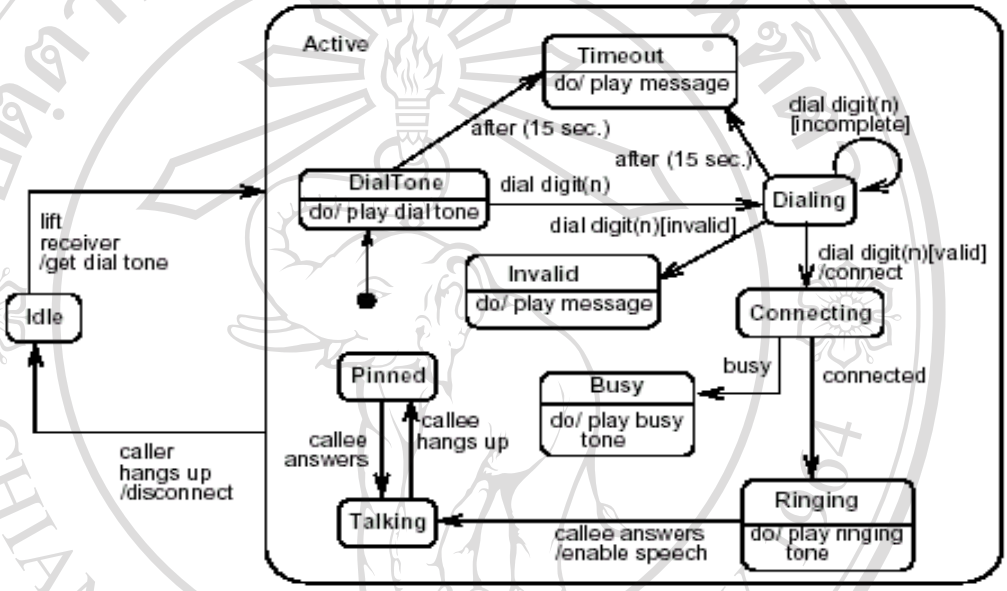
(4) คอลแลบอเรชันไดอะแกรม (Collaboration Diagram) เป็นแผนภาพชนิดเดียวกันกับซีเควทไดอะแกรม โดยซีเควนไดอะแกรมเป็นแผนภาพที่แสดงถึงการแลกเปลี่ยนข่าวสาร แต่คอลแลบอเรชันไดอะแกนใจะแสดงความสัมพันธ์ระหว่างออบเจ็กต์และการปฏิสัมพันธ์ โดยจะแสดงลำดับการทำงานก่อนและหลัง วิธีการเลือกใช้ คือ ถ้าเป็นการกำหนดช่วงเวลาที่แน่นอนและใช้เวลาเป็นสิ่งสำคัญให้เลือกใช้ซีเควนไดอะแกรม แต่ถ้าเป็นการให้ความสำคัญภายในออบเจ็กต์ให้เลือกใช้คอลแลบอเรชันไดอะแกรม ดังรูป 2.12



(อ้างอิงจาก http://www.omg.org/technology/documents/formal/uml_2.htm)

รูป 2.12 แสดงตัวอย่างใช้คอลแลบอเรชันไดอะแกรม

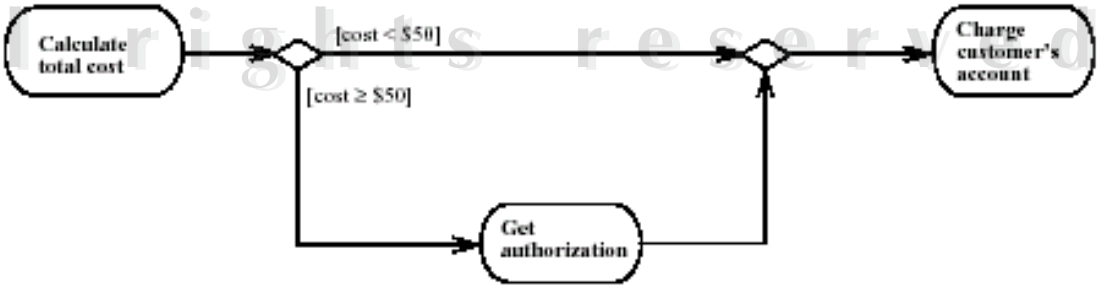
(5) สเตทชาร์ทไดอะแกรม (Statechart Diagram) เป็นแผนภาพที่แสดงถึงวงจรชีวิตของออบเจ็กต์โดยแสดง เหตุการณ์ต่าง ๆ (event) ที่มีผลกระทบต่อสถานะของออบเจ็กต์ ซึ่งอาจจะมีจุดเริ่มต้นและสิ้นสุดได้หลาย ๆ จุด ดังรูป 2.13



(อ้างอิงจาก http://www.omg.org/technology/documents/formal/uml_2.htm)

รูป 2.13 แสดงตัวอย่างสเตทชาร์ทไดอะแกรม

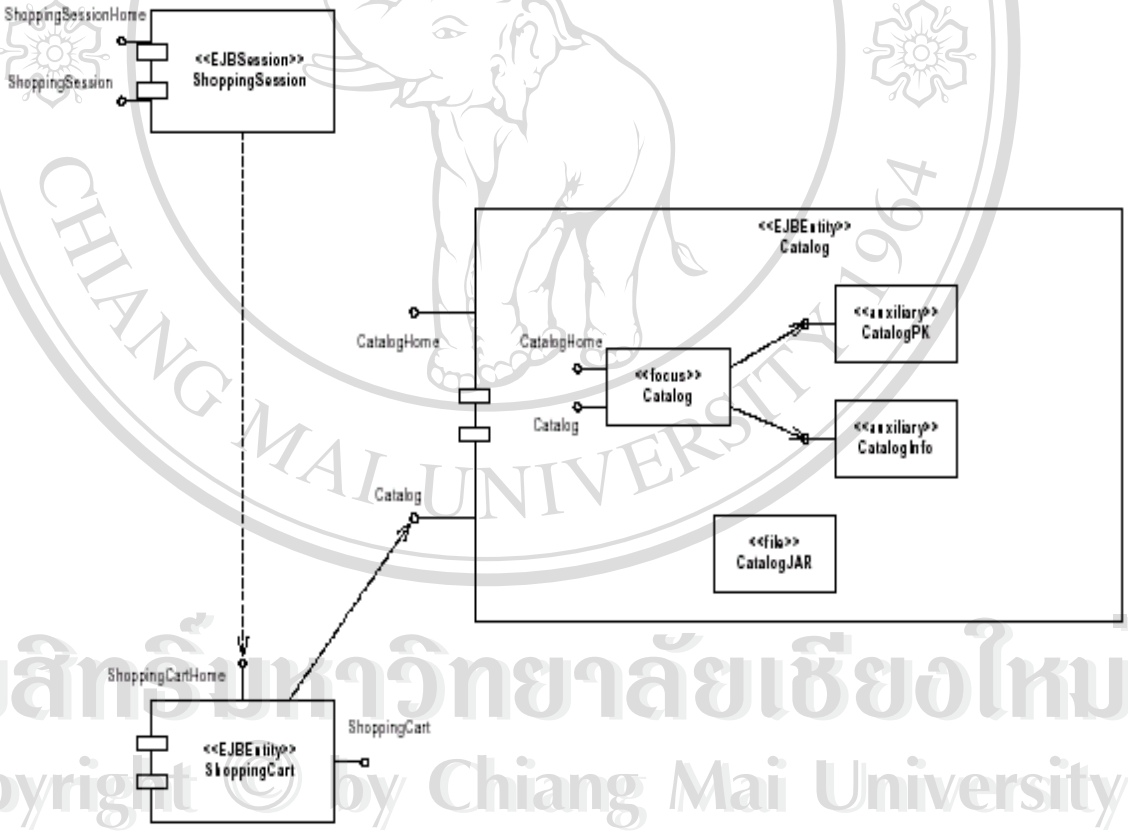
(6) แอ็กทิวิตีไดอะแกรม (Activity Diagram) เป็นแผนภาพที่แสดงขั้นตอนของการปฏิบัติงานและจุดที่ต้องมีการตัดสินใจที่เกิดภายในออบเจ็กต์หรือภายในกระบวนการทำงาน โดยแต่ละขั้นตอน (Activity) จะแสดงอยู่ภายในวงรี และจุดที่มีการตัดสินใจจะแทนด้วยรูปสี่เหลี่ยมขนมเปียกปูน ดังรูป 2.14 ส่วนมากจะถูกใช้สำหรับกระแสงาน (Workflow) หรือกระบวนการทางธุรกิจ (Business process) และกระบวนการภายใน (Internal operation)



(อ้างอิงจาก http://www.omg.org/technology/documents/formal/uml_2.htm)

รูป 2.14 แสดงตัวอย่างแอ็กทิวิตีไดอะแกรม

(7) คอมโพเนนต์ไดอะแกรม (Component Diagram) เป็นแผนภาพแสดงโครงสร้างทางกายภาพ ในส่วนของซอฟต์แวร์คอมโพเนนต์ เช่น ตัวโปรแกรม โปรแกรมที่ถูกคอมไพล์แล้วและส่วนติดต่อผู้ใช้งาน องค์ประกอบต่าง ๆ ของระบบที่เชื่องโยงกันจะเป็นความสัมพันธ์ในลักษณะขึ้นต่อกัน โดยจะแสดงในรูปของเส้นประที่มีหัวลูกศรชี้จากคอมโพเนนต์ลูกไปยังคอมโพเนนต์หลัก ดังรูป 2.15

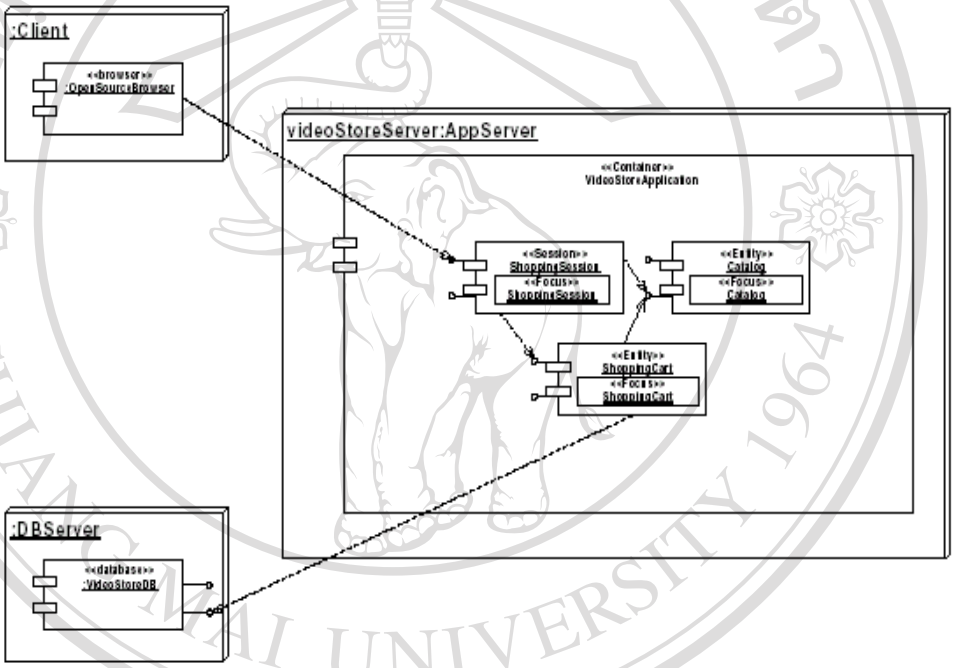


(อ้างอิงจาก http://www.omg.org/technology/documents/formal/uml_2.htm)

รูป 2.15 แสดงตัวอย่างคอมโพเนนต์ไดอะแกรม

(8) ดีพลอยเมนต์ไดอะแกรม (Deployment Diagram) เป็นแผนภาพสำหรับแสดงการเชื่อมต่อของอุปกรณ์ฮาร์ดแวร์ในระบบและมักใช้ร่วมกับคอมโพเนนต์ไดอะแกรม โดยข้างในฮาร์ดแวร์อาจประกอบด้วยซอฟต์แวร์คอมโพเนนต์ ดังรูป 2.16

ดีพลอยเมนต์ไดอะแกรม จะแสดงอยู่ในรูปออบเจ็กต์และแสดงในช่วงเวลาของการรันหรือระหว่างการเอ็กซิคิวต์ ดังนั้นไฟล์คอมโพเนนต์ของระบบที่ไม่ได้ใช้สำหรับรัน เช่น ตัวโปรแกรม จะไม่ปรากฏในแผนภาพประเภทนี้



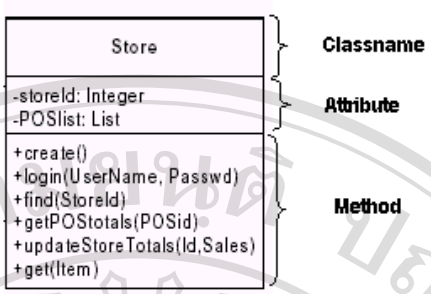
(อ้างอิงจาก http://www.omg.org/technology/documents/formal/uml_2.htm)

รูป 2.16 แสดงตัวอย่างดีพลอยเมนต์ไดอะแกรม

2.6.3 คลาสไดอะแกรม

เป็นแผนภาพที่ประกอบไปด้วยคลาสต่าง ๆ ที่มาประกอบกันกลายเป็นระบบหรือซอฟต์แวร์ตัวหนึ่ง โดยคลาสต่าง ๆ นี้มีที่มาจากซีเควนไดอะแกรมหรือคอลแลบอเรชันไดอะแกรม แต่ละคลาสจะประกอบไปด้วย 3 ส่วน คือ ชื่อคลาส แอททริบิวต์และเมธอด

สัญลักษณ์ที่ใช้แทนคลาสคือรูปสี่เหลี่ยมผืนผ้าโดยแบ่งเป็น 3 ส่วน คือ ชื่อคลาส แอททริบิวต์และเมธอด ดังรูป 2.17



(อ้างอิงจาก http://www.omg.org/technology/documents/formal/uml_2.htm)

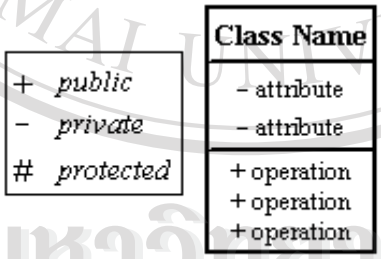
รูป 2.17 แสดงสัญลักษณ์ของคลาส

ในส่วนของสัญลักษณ์ที่อยู่ด้านหน้าแอททริบิวต์หรือเมธอด จะเป็นส่วนที่ใช้แสดงความสามารถเข้าถึงข้อมูลในคลาสของแอททริบิวต์หรือเมธอดนั้น ได้แก่

public จะใช้สัญลักษณ์เป็นเครื่องหมายบวก(+) ซึ่งจะอนุญาตให้ทุก ๆ คลาส สามารถเข้าถึงแอททริบิวต์หรือเมธอดของคลาสนั้น ๆ ได้

private จะใช้สัญลักษณ์เป็นเครื่องหมายลบ(-) ซึ่งจะไม่อนุญาตให้เข้าถึงแอททริบิวต์หรือเมธอดของคลาสนั้น ๆ ได้

protected จะใช้สัญลักษณ์เป็นเครื่องหมายชาร์ป(#) ซึ่งจะอนุญาตเฉพาะลูกของคลาสนั้น ๆ เท่านั้น



(อ้างอิงจาก http://www.smartdraw.com/drawing/software/indexUML.asp)

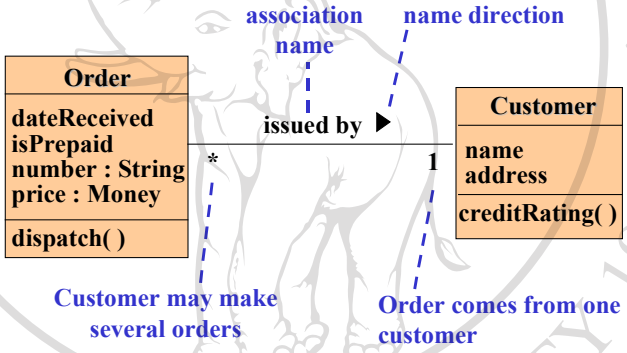
รูป 2.18 แสดงสัญลักษณ์ความสามารถในการเข้าถึงข้อมูลภายในคลาส

คลาสไดอะแกรมนอกจากจะประกอบไปด้วยคลาสต่าง ๆ แล้ว แต่ละคลาสที่เกิดขึ้นในระบบจะมีความสัมพันธ์กันระหว่างคลาส ซึ่งความสัมพันธ์เหล่านี้สามารถแบ่งได้ดังนี้

(1) แอสโซซิเอชัน แสดงความสัมพันธ์ระหว่างคลาสในลักษณะที่ใช้บริการของอีกคลาสหนึ่ง แต่ไม่ได้มีความสัมพันธ์แบบเป็นส่วนหนึ่ง มีความสัมพันธ์ได้ทั้งทางเดียวและสองทาง

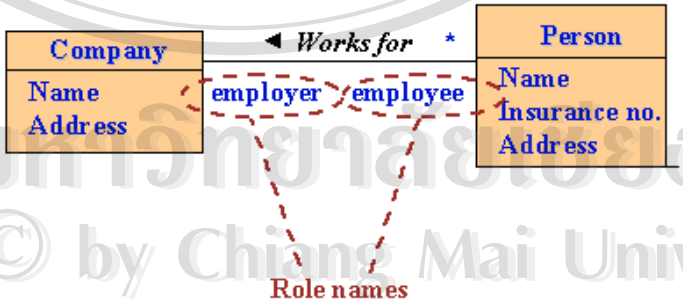
การกำหนดชื่อและทิศทางของความสัมพันธ์ ในคลาสไดอะแกรมสามารถแทนความสัมพันธ์นี้ได้ด้วยเส้นตรงระหว่างคลาสและมีชื่อความสัมพันธ์กำกับอยู่ ในการกำหนดทิศทางของชื่อความสัมพันธ์ทำได้โดยวาดสามเหลี่ยมทึบไว้ด้านซ้ายหรือด้านขวาของชื่อ ทั้งนี้ขึ้นอยู่กับทิศทางของความสัมพันธ์ซึ่งลูกศรจะช่วยในการอ่านความสัมพันธ์ให้เป็นไปได้อย่างถูกต้อง ดังรูป 2.19

โรล (Role) คือ ชื่อที่กำหนดให้กับด้านปลายของแต่ละคลาสที่เกิดความสัมพันธ์ซึ่งระบุวิธีการที่คลาสมีส่วนร่วมในความสัมพันธ์ ดังรูป 2.20



(อ้างอิงจาก http:// staff.buu.ac.th/~seree/310414/ooaClassDiagram.ppt)

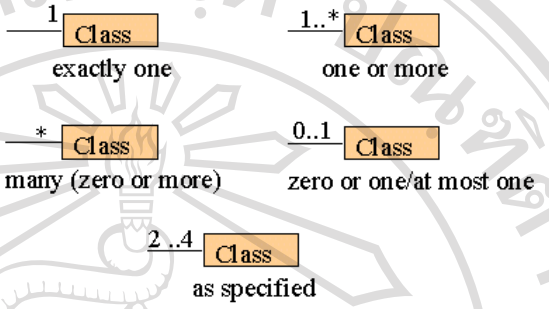
รูป 2.19 แสดงความสัมพันธ์แบบแอสโซซิเอชัน



(อ้างอิงจาก http:// staff.buu.ac.th/~seree/310414/ooaClassDiagram.ppt)

รูป 2.20 แสดงการกำหนดโรล

มัลติพลิซิตี (Multiplicity) คือ การพิจารณาจำนวนออบเจ็กต์ของคลาสหนึ่ง ที่สามารถเชื่อมโยงกับออบเจ็กต์ของคลาสที่เกี่ยวข้อง สามารถทำการกำหนดได้หลายรูปแบบ ดังรูป 2.21

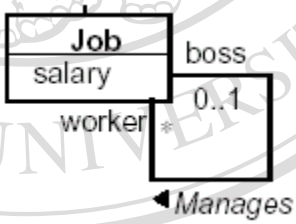


(อ้างอิงจาก http:// staff.buu.ac.th/~seree/310414/ooaClassDiagram.ppt)

รูป 2.21 แสดงสัญลักษณ์ การกำหนดมัลติพลิซิตี

สำหรับความสัมพันธ์แบบแอสโซซิเอชัน สามารถแบ่งได้เป็นดังนี้

- ยูนารี แอสโซซิเอชัน (Unary Association) เป็นความสัมพันธ์แบบที่ประกอบด้วยหนึ่งคลาสเท่านั้นหรือเกิดความสัมพันธ์ภายในคลาสเดียวกัน ดังรูป 2.22

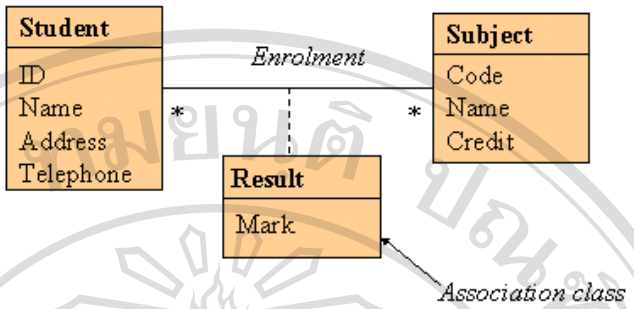


(อ้างอิงจาก http://www.omg.org/technology/documents/formal/uml_2.htm)

รูป 2.22 แสดงความสัมพันธ์แบบยูนารี แอสโซซิเอชัน

- ไบนารี แอสโซซิเอชัน(Binary Association) เป็นความสัมพันธ์ที่มีคลาสที่เกิดความสัมพันธ์ 2 คลาสเท่านั้น ดังรูป 2.20

- แอสโซซิเอชัน คลาส (Association Class) เป็นความสัมพันธ์ที่มีคลาสที่เกิดความสัมพันธ์มากกว่า 1 คลาสขึ้นไป และความสัมพันธ์ที่เกิดขึ้นมีแอททริบิวต์ สามารถแทนความสัมพันธ์นี้ได้ด้วยเส้นประระหว่างคลาส ดังรูป 2.23

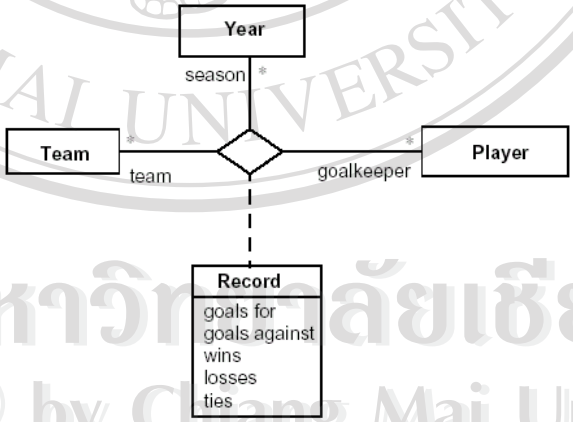


(อ้างอิงจาก http:// staff.buu.ac.th/~seree/310414/ooaClassDiagram.ppt)

รูป 2.23 แสดงความสัมพันธ์แบบแอสโซซิเอชัน คลาส

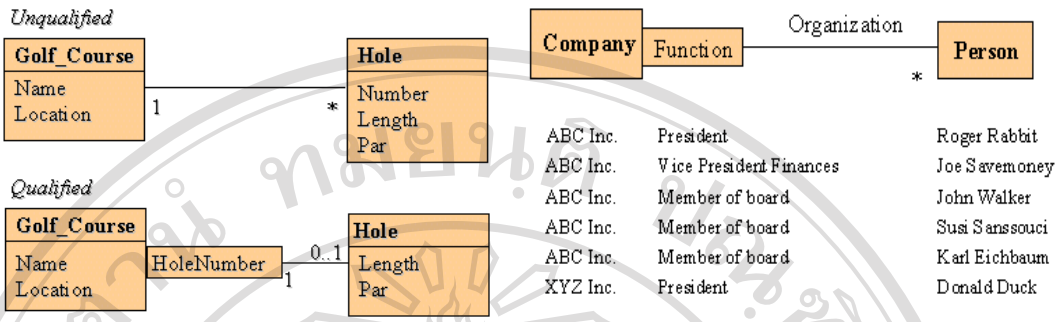
- เอ็นนารี แอสโซซิเอชัน (N-ary Association) เป็นความสัมพันธ์ที่มีคลาสเกิดความสัมพันธ์มากกว่า 2 คลาสขึ้นไป ดังรูป 2.24

- ควอลิไฟ แอสโซซิเอชัน (Qualified Association) เป็นความสัมพันธ์ที่ถูกทำให้ชัดเจนโดยแอททริบิวต์ที่เรียกว่า ควอลิไฟเออร์ (qualifier) พิจารณาควอลิไฟ แอสโซซิเอชันเช่นเดียวกับแนวคิดของเอนทิตีแบบอ่อนในแบบจำลองความสัมพันธ์เอนทิตี ดังรูป 2.25



(อ้างอิงจาก http://www.omg.org/technology/documents/formal/uml_2.htm)

รูป 2.24 แสดงความสัมพันธ์แบบเอ็นนารี แอสโซซิเอชัน

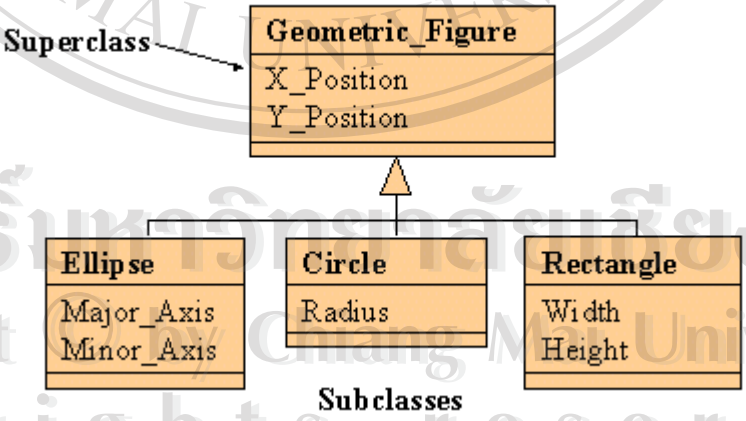


(อ้างอิงจาก http:// staff.buu.ac.th/~seree/310414/ooaClassDiagram.ppt)

รูป 2.25 แสดงความสัมพันธ์แบบควอลิไฟ แอสโซซิเอชัน

(2) เจเนอรัลไลเซชัน เป็นความสัมพันธ์ระหว่างซูเปอร์คลาสและซับคลาส โดยซับคลาสรับถ่ายทอดคุณสมบัติ ได้แก่ แอททริบิวต์และเมธอดมาจากซูเปอร์คลาสของตัวเอง ความสัมพันธ์แบบนี้ถูกเรียกว่า “Is-a” นอกจากนี้ถ้าแอททริบิวต์หรือเมธอดของซูเปอร์คลาสถูกกำหนดให้เป็นค่าใหม่ในซับคลาสจะเป็นการทับคำนิยามที่กำหนดไว้ในซูเปอร์คลาส นำสู่แนวคิดการเปลี่ยนรูป

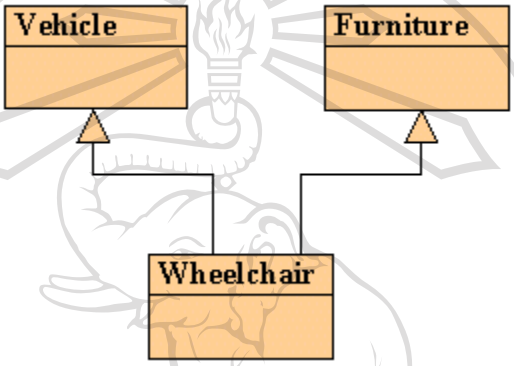
ความสัมพันธ์ระหว่างซูเปอร์คลาสและซับคลาสสามารถแทนได้ด้วยสัญลักษณ์รูปสามเหลี่ยม ดังรูป 2.26



(อ้างอิงจาก http:// staff.buu.ac.th/~seree/310414/ooaClassDiagram.ppt)

รูป 2.26 แสดงความสัมพันธ์แบบเจเนอรัลไลเซชัน

- ซิงเกิล เจเนอรัลไลเซชัน (Single Generalization) คือ โครงสร้างลำดับชั้นที่แต่ละซับคลาส มีเพียงหนึ่งซุปเปอร์คลาส
- มัลติเพิล เจเนอรัลไลเซชัน (Multiple Generalization) คือ โครงสร้างลำดับชั้นที่แต่ละซับคลาส มีมากกว่าหนึ่งซุปเปอร์คลาส ดังรูป 2.27

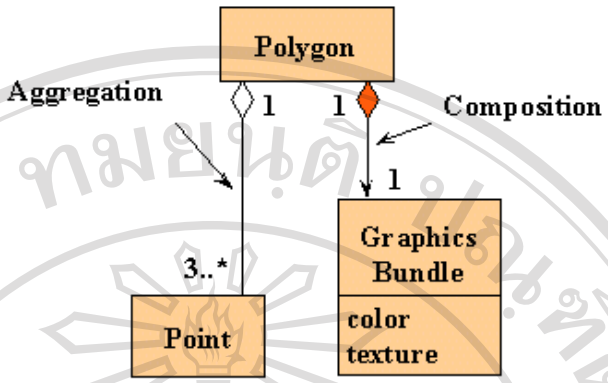


(อ้างอิงจาก http:// staff.buu.ac.th/~seree/310414/ooaClassDiagram.ppt)

รูป 2.27 แสดงความสัมพันธ์แบบมัลติเพิล เจเนอรัลไลเซชัน

(3) อะกรีเกชัน เป็นความสัมพันธ์ระหว่างคลาสหรือออบเจ็กต์ในแง่ของการรวมกัน ความสัมพันธ์ระหว่างคลาสที่เกิดอะกรีเกชันสามารถแทนด้วยสี่เหลี่ยมข้าวหลามตัดติดอยู่ระหว่างปลายเส้นความสัมพันธ์กับคลาสที่หมายถึงสิ่งที่ใหญ่กว่า ดังรูป 2.28

(4) คอมโพสิตชัน เป็นความสัมพันธ์การประกอบกันเป็นความสัมพันธ์ระหว่าง Whole และ part บางครั้งถูกเรียกว่า “ Has a” คือ Whole จะประกอบไปด้วย parts ต่าง ๆ ดังนั้น การคงอยู่ของ parts จะต้องขึ้นอยู่กับ Whole สามารถแทนด้วยสี่เหลี่ยมข้าวหลามตัดทึบติดอยู่ระหว่างปลายเส้นความสัมพันธ์กับคลาสที่หมายถึงสิ่งที่ใหญ่กว่า



(อ้างอิงจาก http:// staff.buu.ac.th/~seree/310414/ooaClassDiagram.ppt)

รูป 2.28 แสดงความสัมพันธ์แบบอะกรีเกชันและคอมโพสิตชัน

2.7 ภาษาเชิงวัตถุ

การเขียนโปรแกรมเชิงวัตถุได้นำเอาส่วนที่ดีที่สุดของการโปรแกรมแบบโปรซีเยอร์มาพัฒนาร่วมกับแนวความคิดใหม่ ๆ ที่ประยุกต์มาจากสิ่งที่เกิดขึ้นในชีวิตประจำวันของมนุษย์ ซึ่งผลที่ได้จะนำไปสู่แนวทางการเขียนโปรแกรมรูปแบบใหม่ที่มีประสิทธิภาพดีกว่าแบบเดิม โดยปกติแล้วการเขียนโปรแกรมเชิงวัตถุจะทำการแบ่งขอบเขตของงานที่ต้องการออกเป็นส่วนย่อย ๆ ที่มีความสัมพันธ์กันรวมไว้ด้วยกันเป็นกลุ่มจากนั้นกลุ่มดังกล่าวจะถูกนำมาเรียงลำดับในรูปของโครงสร้างที่มีลักษณะเป็นลำดับชั้น เพื่อเพิ่มประสิทธิภาพในการเรียกใช้งานนั่นเอง โดยภาษาที่จะเป็นภาษาในการเขียนโปรแกรมเชิงวัตถุได้นั้นต้องมีคุณสมบัติ 3 ข้อดังนี้

1. จะต้องเป็นภาษาที่ใช้ออบเจ็กต์ในการแก้ปัญหาเป็นหลัก เช่น ถ้าต้องการเขียนโปรแกรมติดต่อกับโมเด็ม หากเป็นการเขียนโปรแกรมแบบเก่าจะแบ่งฟังก์ชันการทำงานออกเป็น ส่งรับหรือตรวจสอบข้อมูล แต่ถ้าเป็นภาษาเชิงวัตถุแล้วผู้เขียนต้องมองโมเด็มให้เป็นออบเจ็กต์ตัวหนึ่ง โดยในออบเจ็กต์โมเด็มก็อาจประกอบไปด้วยออบเจ็กต์ย่อย ๆ และออบเจ็กต์ย่อยเหล่านี้ก็จะมีการสื่อสารซึ่งกันและกัน
2. ออบเจ็กต์ที่ปรากฏในการเขียนโปรแกรมต้องเป็นอินสแตนซ์ของคลาส
3. คลาสสามารถรับทอดคุณสมบัติการใช้งานไปยังคลาสอื่นได้

ถ้าภาษาใดที่ขาดคุณสมบัติในสามข้อนี้ไปก็จะไม่เรียกภาษานั้นว่าเป็นภาษาในการเขียนโปรแกรมเชิงวัตถุ เช่นถ้าภาษาใดสนับสนุนการสร้างคลาสและออบเจ็กต์ แต่ไม่สนับสนุนการรับ

ทอดคุณสมบัติของคลาสแล้ว จะเรียกภาษานั้นว่า ภาษาที่ทำงานบนชนิดข้อมูลนามธรรม (Abstract Data Type) หรืออาจเรียกว่าภาษาในการเขียนโปรแกรมบนพื้นฐานของวัตถุ ไม่ใช่ภาษาในการเขียนโปรแกรมเชิงวัตถุ (อภิเนตร อูนากูล, 2543:18-19)

ในการแสดงให้เห็นถึงแนวทางเชิงวัตถุ หลักการต่าง ๆ จะถูกนำไปใช้ในการเขียนโปรแกรมซึ่งสามารถทำได้วิธีใดวิธีหนึ่ง ดังนี้

1. นำวิธีการเชิงวัตถุไปใช้เป็นเครื่องมือในการออกแบบเพื่อดำเนินการในขั้นต่อไป เช่น การออกแบบฐานข้อมูลเชิงสัมพันธ์ เริ่มจากการออกแบบแบบจำลองความสัมพันธ์เอนทิตี จากนั้นก็แปลงไปเป็นรีเลชัน

2. วิธีการเชิงวัตถุถูกนำไปรวมในภาษาใดภาษาหนึ่ง เพื่อใช้ในการจัดการฐานข้อมูล ด้วยวิธีการนี้มีอยู่หลายภาษาที่สามารถรวมวิธีการเชิงวัตถุได้ ดังนี้

- เพิ่มความสามารถเชิงวัตถุเข้าไปในภาษาเอสคิวแอล(SQL) เพื่อให้สามารถจัดการกับชนิดข้อมูลที่ซับซ้อนได้ เรียกว่าระบบฐานข้อมูลเชิงวัตถุสัมพันธ์(Object-relational database system)

- นำภาษาโปรแกรมที่มีลักษณะเชิงวัตถุอยู่แล้วมาเพิ่มความสามารถทางด้านฐานข้อมูลเข้าไป เรียกว่า ภาษาในการเขียนโปรแกรมแบบถาวร (persistent programming languages)

2.8 ภาษาในการเขียนโปรแกรมแบบถาวร

ภาษาในการเขียนโปรแกรมแบบถาวร คือโปรแกรมภาษาที่เพิ่มส่วนสำหรับจัดการข้อมูลให้คงอยู่แม้ว่าโปรแกรมที่สร้างข้อมูลนั้นขึ้นมาได้ถูกปิดไปแล้วก็ตาม ซึ่งแตกต่างจากภาษาที่รวมเอาภาษาเอสคิวแอลเข้าไว้ในตัวภาษาอย่างน้อย 2 อย่างคือ

1. ภาษาที่รวมเอาภาษาเอสคิวแอลเข้าไว้ในตัวภาษา ชนิดข้อมูลของภาษาเจ้าบ้าน (host language) จะแตกต่างชนิดข้อมูลของภาษาจัดการข้อมูล ดังนั้นจึงเป็นหน้าที่ของนักเขียนโปรแกรมที่จะทำการแปลงชนิดข้อมูลระหว่างภาษาเจ้าบ้านและภาษาจัดการข้อมูล ซึ่งมีอุปสรรคหลายอย่างคือ

- โปรแกรมที่ใช้ในการแปลงระหว่างออบเจ็กต์และทูเพิล ไม่ได้มีการดำเนินการในเชิงวัตถุ ซึ่งอาจจะก่อให้เกิดความผิดพลาดขึ้นได้

- การแปลงข้อมูลระหว่างข้อมูลที่อยู่ในรูปแบบเชิงวัตถุกับข้อมูลที่อยู่ในรูปแบบเชิงสัมพันธ์ ต้องเขียนโปรแกรมเป็นจำนวนมาก

ในทางกลับกันภาษาในการเขียนโปรแกรมแบบถาวรและภาษาเอสคิวแอลถูกรวมเข้าด้วยกันกับภาษาเจ้าบ้านและทั้งสองภาษาต่างก็มีการใช้งานชนิดข้อมูลชนิดเดียวกัน ดังนั้นออบเจ็กต์สามารถถูกสร้างและถูกจัดเก็บในฐานข้อมูล โดยไม่ต้องมีการเปลี่ยนแปลงรูปแบบใด ๆ ทั้งสิ้น

2. นักเขียนโปรแกรมที่ใช้งานภาษาที่รวมเอาภาษาเอสคิวแอลเข้าไว้ในตัวภาษา เพื่อไปดึงข้อมูลจากฐานข้อมูลเข้าสู่หน่วยความจำ ถ้ามีการปรับปรุงข้อมูลก็จะเขียนคำสั่งเพื่อสั่งให้นำข้อมูลบันทึกกลับลงไปสู่ฐานข้อมูล ส่วนในภาษาในการเขียนโปรแกรมแบบถาวรนั้น นักเขียนโปรแกรมสามารถจัดการกับข้อมูลได้โดยไม่ต้องมีการเขียนโปรแกรมเพื่อดึงข้อมูลหรือบันทึกข้อมูลกลับลงไปในฐานข้อมูล

2.8.1 ออบเจ็กต์ถาวร (Persistent Object)

ปกติเมื่อโปรแกรมจบการทำงาน ข้อมูลของออบเจ็กต์ที่อยู่ในหน่วยความจำจะถูกลบทิ้งไป การจะให้ข้อมูลของออบเจ็กต์คงอยู่ทำได้โดยเก็บข้อมูลเหล่านั้นไว้ในหน่วยเก็บข้อมูลสำรอง เช่น เก็บลงไฟล์ ในเทคโนโลยีเชิงวัตถุมีแนวคิดของการทำออบเจ็กต์ถาวร คือ ออบเจ็กต์ที่สามารถคงอยู่ได้หลังจากโปรแกรมจบการทำงาน สถานะของออบเจ็กต์ถาวรสามารถที่จะถูกเก็บลงในหน่วยเก็บข้อมูลถาวรได้ ดังนั้นเมื่อโปรแกรมจบการทำงานข้อมูลเหล่านั้นก็ยังคงอยู่ และเมื่อรันโปรแกรมใหม่อีกครั้ง ออบเจ็กต์นั้นก็สามารถที่จะโหลดสถานะเดิมของตนขึ้นมาใหม่และทำงานต่อเนื่องไปได้

การทำให้ออบเจ็กต์สามารถคงสถานะได้มีอยู่หลายวิธีคือ

(1) Persistence by class เป็นวิธีที่ง่ายที่สุด แต่ไม่เหมาะที่จะนำมาใช้งาน วิธีการนี้เป็นการประกาศว่าคลาสที่สร้างขึ้นเป็นแบบถาวร นั่นคือออบเจ็กต์ทุกออบเจ็กต์ของคลาสจะเป็นออบเจ็กต์ถาวรไปโดยปริยาย ซึ่งวิธีการนี้ไม่มีความยืดหยุ่น เนื่องจากออบเจ็กต์มักถูกใช้ทั้งแบบชั่วคราวและแบบถาวรในคลาส

(2) Persistence by creation วิธีการนี้เป็นการกำหนดรูปแบบในการสร้าง p ออบเจ็กต์ถาวร โดยการเพิ่มรูปแบบสำหรับสร้างออบเจ็กต์ชั่วคราว ดังนั้นออบเจ็กต์ที่ถูกสร้างขึ้นจะเป็นแบบถาวรหรือแบบชั่วคราวขึ้นอยู่กับตอนที่สร้างออบเจ็กต์ว่าจะสร้างอย่างไร

(3) Persistence by marking วิธีนี้จะทำการกำหนดให้ออบเจ็กต์เป็นแบบถาวรหลังจากที่ได้ถูกสร้างขึ้นมา ทุก ๆ ออบเจ็กต์ถูกสร้างขึ้นมาแบบชั่วคราว แต่ถ้าออบเจ็กต์ไหนต้องการที่จะให้คงอยู่หลังจากจบโปรแกรม ก็จะทำการกำหนดก่อนที่จะจบโปรแกรม

(4) Persistence by reference ออบเจ็กต์หนึ่งหรือหลาย ๆ ออบเจ็กต์จะถูกกำหนดให้ถาวร ในลักษณะเป็นรากของออบเจ็กต์ ดังนั้นออบเจ็กต์ทุกตัวที่ถูกอ้างถึงจากรากก็จะเป็นแบบถาวร

2.8.2 การกำหนดโอไอดีและตัวชี้

เมื่อออบเจ็กต์ถาวรถูกสร้าง ออบเจ็กต์จะถูกกำหนดโอไอดีแบบถาวร แนวคิดโอไอดีมีลักษณะคล้ายกับตัวชี้ สำหรับการสร้างโอไอดีสามารถทำได้ คือ ใช้ตัวชี้ชี้ไปที่ตำแหน่งทางกายภาพของแหล่งเก็บข้อมูล โดยเฉพาะภาษาเชิงวัตถุอย่างซีพลัสพลัส มีการสร้างความสัมพันธ์ระหว่างออบเจ็กต์กับตำแหน่งทางกายภาพในแหล่งเก็บข้อมูล อาจมีการเปลี่ยนแปลงได้ตลอดเวลา อย่างไรก็ตามได้มีการแบ่งระดับความความถาวรไว้ดังนี้

(1) Intraprocedure: Identity จะคงอยู่ในขณะที่กำลังทำคำสั่งของโพซีเดอร์ เช่น ตัวแปรชนิดโลคอล ภายในโพซีเดอร์

(2) Intraprogram: Identity จะคงอยู่ในขณะที่กำลังทำคำสั่งของหนึ่งโปรแกรมหรือการสืบค้น เช่น ตัวแปรชนิดโกลบอล

(3) Interprogram: Identity จะคงอยู่จากโปรแกรมหนึ่งไปสู่อีกโปรแกรมหนึ่ง เช่น ตัวชี้ชี้ไปยังข้อมูลในแฟ้มบนดิสก์ แต่สามารถเปลี่ยนแปลงได้ขึ้นอยู่กับวิธีการจัดเก็บข้อมูลในแฟ้มถ้ามีการเปลี่ยนแปลงระบบแฟ้ม

(4) Persistent: Identity จะคงอยู่ไม่เฉพาะตอนที่โปรแกรมกำลังทำงานเท่านั้น แต่จะเกี่ยวข้องกับการปรับเปลี่ยนโครงสร้างของข้อมูลด้วย ซึ่งการคงอยู่ในลักษณะนี้จะถูกใช้ในระบบเชิงวัตถุ

2.8.3 การจัดเก็บและการเข้าถึงออบเจ็กต์แบบถาวร

ข้อมูลของแต่ละออบเจ็กต์จะถูกจัดเก็บแยกเฉพาะแต่ละออบเจ็กต์ในฐานข้อมูล และส่วนของโปรแกรมที่ใช้เขียนเมธอดควรถูกจัดเก็บไว้ในส่วนของเค้าร่างฐานข้อมูล รวมไปถึงการกำหนดชนิดข้อมูลต่าง ๆ ของคลาส อย่างไรก็ตามโดยส่วนมากจะมีการจัดเก็บไว้ในไฟล์แยกออกมาจากฐานข้อมูล การค้นหาออบเจ็กต์ ในฐานข้อมูลทำได้หลายวิธีคือ

(1) บอกชื่อของออบเจ็กต์ เหมาะกับจำนวนออบเจ็กต์น้อย ๆ

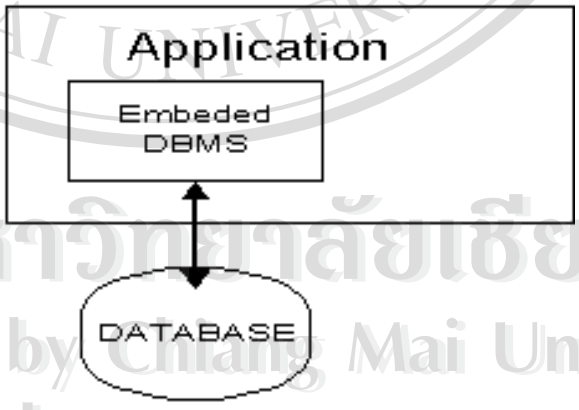
(2) ใช้โอไอดีหรือตัวชี้แบบถาวรชี้ไปที่ออบเจ็กต์ ซึ่งสามารถถูกจัดเก็บไว้ภายนอกฐานข้อมูล ตัวชี้เหล่านี้สามารถที่จะชี้ไปยังตำแหน่งที่เก็บข้อมูลในฐานข้อมูลได้ทันที

(3) เก็บคอลเลกชันของออบเจ็กต์และยอมให้โปรแกรมทำการค้นหาออบเจ็กต์ที่ต้องการในคอลเลกชันได้ และมีคลาสพิเศษที่เรียกว่า คลาสแบบขยาย (class extent) ซึ่งเป็นคอลเลกชันของออบเจ็กต์ทุกตัวของคลาส ซึ่งเมื่อไรก็ตามที่ออบเจ็กต์ถูกสร้าง ออบเจ็กต์นั้นก็จะถูกเพิ่มเข้าไปในคลาสแบบขยายทันที และเมื่อไรที่ออบเจ็กต์ถูกลบ ออบเจ็กต์นั้นก็จะถูกลบออกจากคลาสแบบขยาย เช่นกัน (วิทวัส พันธุมจินดา, 2543:397-400)

2.9 ฐานข้อมูลเชิงวัตถุ

จากแนวคิดของการทำออบเจ็กต์ถาวรได้นำไปสู่การสร้างฐานข้อมูลเชิงวัตถุที่สามารถจัดเก็บออบเจ็กต์และความสัมพันธ์ระหว่างออบเจ็กต์ที่ซับซ้อนได้ และมีความสามารถของระบบจัดการฐานข้อมูลเหมือนกับฐานข้อมูลเชิงสัมพันธ์ เช่น สามารถค้นหา ออบเจ็กต์ จนถึงการสืบค้น ภายในฐานข้อมูล, การใช้ข้อมูลออบเจคต์ร่วมกัน เป็นต้น

ระบบจัดการฐานข้อมูลเชิงวัตถุ ส่วนหนึ่งจะอยู่ในรูปแบบของคลาสไลบรารีที่ใช้สำหรับเขียนโปรแกรมการจัดการฐานข้อมูลเชิงวัตถุ ดังนั้นปกติแล้วฐานข้อมูลเชิงวัตถุจะสร้างมาเฉพาะสำหรับภาษาที่ใช้ เช่น ฐานข้อมูลเชิงวัตถุสำหรับภาษาซีพลัสพลัสก็สามารถใช้งานกับภาษาซีพลัสพลัสเท่านั้น ดังนั้นฐานข้อมูลประเภทนี้ส่วนมากจึงอยู่ในรูปของ ระบบจัดการฐานข้อมูลแบบฝังตัว (Embedded DBMS) ดังแสดงในรูป 2.29



รูป 2.29 แสดงระบบจัดการฐานข้อมูลแบบฝังตัว

ปกติระบบจัดการฐานข้อมูลเชิงวัตถุนั้นตัวจัดการฐานข้อมูลจะถูกรวมเข้าเป็นส่วนหนึ่งภายในแอปพลิเคชันเลย แต่ในฐานข้อมูลเชิงวัตถุบางตัวที่สามารถสนับสนุนการทำงานแบบไคล์เอนต์/เซิร์ฟเวอร์ ก็จะมีส่วนที่แยกออกมาเพื่อทำหน้าที่จัดการข้อมูลของออบเจ็กต์ภายในเซิร์ฟเวอร์ แล้วส่งข้อมูลผ่านทางระบบเครือข่าย

ถึงแม้ว่าฐานข้อมูลเชิงวัตถุนี้จะยังไม่มีมาตรฐานอย่างเป็นทางการ แต่ในทางปฏิบัติแล้วบริษัทผู้พัฒนาซอฟต์แวร์ระบบฐานข้อมูลเชิงวัตถุได้อ้างอิงเอามาตรฐานที่ Morgan Kaufmann Publisher ได้ตีพิมพ์ไว้ ซึ่งการจัดทำมาตรฐานนี้ได้รับการสนับสนุนจาก Object Database Management Group (ODMG) โดยที่เนื้อหาของมาตรฐานจะเน้นหนักไปในเรื่องของความสัมพันธ์ระหว่างวัตถุ ที่จะสามารถสนับสนุนการเขียนโปรแกรมด้วยภาษาเชิงวัตถุได้ รวมถึงการจัดเก็บวัตถุต่าง ๆ ในฐานข้อมูล

2.9.1 โครงสร้างข้อมูล

การใช้งานข้อมูลในฐานข้อมูลเชิงวัตถุจะอยู่ในรูปแบบของคลาส ซึ่งเป็นแนวความคิดพื้นฐานของวิธีการเชิงวัตถุ โดยที่ในคลาสจะประกอบด้วยแอททริบิวต์และเมธอด รวมไปถึงกฎความคงสภาพของคลาสนั้น ๆ ซึ่งจะมีการกำหนดโอไอดีเอาไว้ เพื่อแบ่งแยกความแตกต่างระหว่างออบเจ็กต์ นอกจากนี้ยังสนับสนุนการทำงานแบบการห่อหุ้มและการรับทอดคุณสมบัติด้วย สำหรับชนิดของข้อมูลนั้นสามารถรองรับชนิดข้อมูลนามธรรมได้ จากมาตรฐานของ ODMG ได้มีการจำแนกประเภทของออบเจ็กต์เป็น 2 ชนิดคือออบเจ็กต์แบบกำหนดโอไอดีและออบเจ็กต์แบบไม่กำหนดโอไอดี รวมไปถึงการแบ่งแยกประเภทคุณลักษณะของออบเจ็กต์เอาไว้เป็น 2 ชนิดคือ แอททริบิวต์และเมธอด

เนื่องจากฐานข้อมูลเชิงวัตถุนี้เกิดขึ้นจากการรวมเอาพื้นฐานแนวความคิดเชิงวัตถุกับหลักการด้านภาษาสำหรับการโปรแกรมเชิงวัตถุ และความสามารถด้านฐานข้อมูลเข้าด้วยกัน ผลลัพธ์ที่ได้คือทำให้เกิดความเข้ากันได้เป็นอย่างดีระหว่างโครงสร้างข้อมูลภายในฐานข้อมูล กับโครงสร้างข้อมูลที่ใช้ในการพัฒนาโปรแกรม ซึ่งอำนวยความสะดวกในการใช้ภาษาสำหรับการโปรแกรมเชิงวัตถุต่าง ๆ เพื่อพัฒนาโปรแกรมที่ใช้งานร่วมกับฐานข้อมูลเชิงวัตถุ ทำให้สามารถลดจำนวนโค้ดที่ใช้ในการเขียนโปรแกรมต่าง ๆ ลงได้

2.9.2 ภาษาที่ใช้ในการจัดการฐานข้อมูล

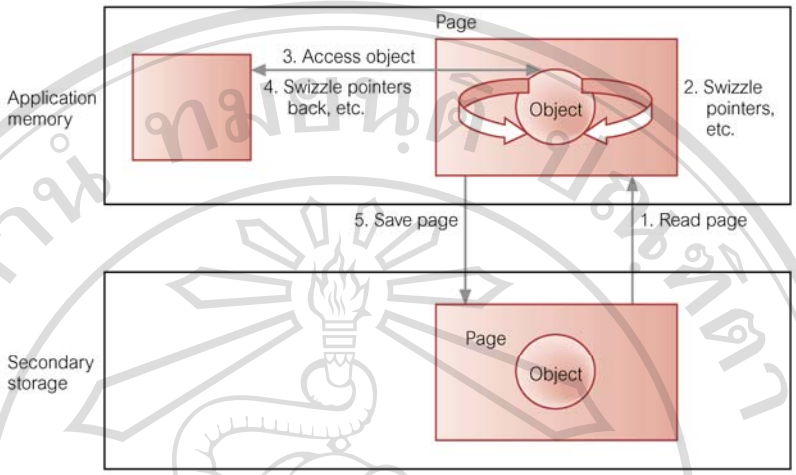
ระบบฐานข้อมูลเชิงวัตถุจะใช้ภาษาสำหรับการโปรแกรมเชิงวัตถุ ทั้งในการจัดการฐานข้อมูลและการพัฒนาโปรแกรม เช่น ซีพลัสพลัส จาวา เป็นต้น เนื่องจากแอปพลิเคชันที่ถูกพัฒนาด้วยโปรแกรมเชิงวัตถุกับการเก็บข้อมูลในระบบฐานข้อมูลชนิดนี้มีความสัมพันธ์กันโดยตรง ดังนั้น

การทำนิยาม การจัดการและการสืบค้นข้อมูล จึงสามารถทำได้ด้วยภาษาสำหรับการโปรแกรมเชิงวัตถุดังกล่าว

นอกจากนั้นแล้ว ตามมาตรฐาน ODMG ได้มีการกำหนดภาษาที่ใช้นิยามโครงสร้างออบเจ็กต์และความสัมพันธ์ที่เกิดขึ้นระหว่างออบเจ็กต์ คือ ภาษานิยามโครงสร้างออบเจ็กต์ และยังได้กำหนดภาษาสำหรับจัดการข้อมูลขึ้นมาอีกภาษาหนึ่งคือภาษาสืบค้นออบเจ็กต์ (Object Query Language) ซึ่งจะมีความแตกต่างกันออกไปจากภาษาเอสคิวแอลตามมาตรฐานเอสคิวแอลเวอร์ชันสอง เพราะในมาตรฐานเอสคิวแอลเวอร์ชันสองไม่มีความสามารถในการเข้าถึงข้อมูลแบบวัตถุได้ อย่างไรก็ดีในมาตรฐานใหม่ของเอสคิวแอลเวอร์ชันสาม จะได้มีบางส่วนที่จะได้เพิ่มความสามารถเหล่านี้ลงไป ซึ่งผลจากการพัฒนาภาษาสืบค้นออบเจ็กต์ จะสามารถสร้างความหลากหลายของผลลัพธ์ที่เกิดขึ้นจากการสืบค้น ได้แก่ ผลลัพธ์ที่อยู่ในรูปของ อะตอม โครงสร้าง ออบเจ็กต์แบบกำหนดโอไอดีและออบเจ็กต์แบบไม่กำหนดโอไอดีหรือแม้แต่เซตของออบเจ็กต์

2.9.3 กลไกที่ใช้ในการเข้าถึงข้อมูล

การสร้างและแก้ไขข้อมูลในระบบฐานข้อมูลเชิงวัตถุ จะใช้การเข้าถึงโดยตรงจากภาษาสำหรับการโปรแกรมเชิงวัตถุ สำหรับออบเจ็กต์ที่ถูกสร้างขึ้นในระบบฐานข้อมูลเชิงวัตถุนี้จะถูกกำหนดโอไอดีให้อัตโนมัติ โดยโอไอดีนี้จะกำหนดเอกลักษณ์เฉพาะตัวและถูกไปตลอดอายุการใช้งานออบเจ็กต์นั้น ๆ นอกจากนี้ออบเจ็กต์แต่ละออบเจ็กต์ยังสามารถจัดเก็บโอไอดีของออบเจ็กต์ตัวอื่น เพื่อสร้างเป็นการอ้างอิงทางตรรกะ (Logical Reference) ซึ่งการอ้างอิงนี้จะเป็นประโยชน์ในการสร้างความสัมพันธ์ระหว่างออบเจ็กต์ที่เกิดขึ้นตามโลกแห่งความเป็นจริง ในระบบฐานข้อมูลเชิงวัตถุบางระบบจะใช้โอไอดีในลักษณะทางกายภาพด้วย กล่าวคือ จะทำการแปลงรูปจากการอ้างอิงทางตรรกะมาเป็นตัวชี้ เพื่ออ้างอิงในการเข้าถึงข้อมูลที่เก็บไว้ในหน่วยความจำแคช การแปลงดังกล่าวจะเรียกว่า Pointer Swizzling ซึ่งจะทำให้เวลาในการเข้าถึงข้อมูลน้อยลง เพราะเป็นการเข้าถึงข้อมูลในหน่วยความจำแคชจึงมีความเร็วในการเข้าถึงข้อมูลในระดับของหน่วยความจำ ดังรูป 2.30



(อ้างอิงจาก Connolly และ Begg, 1998:764)

รูป 2.30 กลไกการเข้าถึงข้อมูลด้วย ODBMS

2.9.4 ข้อดีของฐานข้อมูลเชิงวัตถุ

(1) ง่ายในการออกแบบแอปพลิเคชัน เพราะทั้งในโปรแกรมและตัวฐานข้อมูลจะออกแบบแบบจำลองข้อมูลชุดเดียวกันคือ ออกแบบเป็นเชิงวัตถุ

(2) ได้ประสิทธิภาพที่ดีกว่าในการทำงานกับโครงสร้างข้อมูลที่ซับซ้อนเมื่อเทียบกับการใช้ฐานข้อมูลเชิงสัมพันธ์ เพราะไม่ต้องแปลงข้อมูลที่เป็นออบเจ็กต์ให้เข้าไปอยู่ในตารางหรือจากตารางกลับออกมาสร้างเป็นออบเจ็กต์ ทำให้เวลาที่ใช้ในการเขียนโปรแกรมสั้นลง และจำนวนบรรทัดของโค้ดก็น้อยลง

(3) การเก็บออบเจ็กต์ที่ซับซ้อนทำได้ง่ายขึ้น เพราะสามารถเก็บลงไปเป็นออบเจ็กต์ได้โดยตรง

(4) ประยุกต์ใช้กับงานประเภท Rule-based Expert System ได้โดยตรง

2.9.5 ข้อจำกัดของฐานข้อมูลเชิงวัตถุ

(1) ลักษณะของฐานข้อมูลเชิงวัตถุจะขึ้นกับแอปพลิเคชันค่อนข้างมาก เนื่องจากส่วนของโค้ดกับส่วนของข้อมูลจะไม่ได้แยกออกจากกันชัดเจนเหมือนฐานข้อมูลเชิงสัมพันธ์ เวลาที่ต้องการย้ายหรือเปลี่ยนไปใช้ฐานข้อมูลตัวอื่นจะทำได้ลำบาก

(2) เครื่องมือช่วยในการวิเคราะห์ข้อมูลยังมีไม่มากต่างจากฐานข้อมูลเชิงสัมพันธ์ที่มีเครื่องมือวิเคราะห์ข้อมูลให้เลือกใช้มากมาย

(3) ขาดมาตรฐานที่ยอมรับ

(4) ระบบจัดการฐานข้อมูลเชิงวัตถุแต่ละยี่ห้อแตกต่างกันมากในการทำงาน เทียบกับฐานข้อมูลเชิงสัมพันธ์ที่มีฟังก์ชันหลักเหมือนกันคือ ใช้ภาษาเอสคิวแอล

สำหรับอนาคตของฐานข้อมูลเชิงวัตถุอาจจะไม่ได้เข้ามาแทนที่ฐานข้อมูลเชิงสัมพันธ์ซึ่งมีข้อดีอยู่หลายประการรวมทั้งการมีแอปพลิเคชันปัจจุบันจำนวนมาก แต่สำหรับการพัฒนาแอปพลิเคชันใหม่ ๆ ที่มีความซับซ้อนและต้องการประสิทธิภาพสูง ฐานข้อมูลเชิงวัตถุก็น่าจะได้รับความนิยมมากกว่า และก็เป็นไปได้อีกเช่นกันที่จะใช้ฐานข้อมูลเชิงสัมพันธ์ร่วมกับฐานข้อมูลเชิงวัตถุ โดยส่วนที่ทำงานกับข้อมูลที่มีความซับซ้อนสูงและต้องการประสิทธิภาพมากก็ใช้ฐานข้อมูลเชิงวัตถุ ส่วนอื่น ๆ ที่เหลือก็ใช้ฐานข้อมูลเชิงสัมพันธ์ซึ่งมีเครื่องมือสนับสนุนต่าง ๆ ให้ใช้มากกว่า