



การพัฒนาภาษาริเลขณนตคยใช่วิธีการเชิงวัตถุ

The Development of Object - Base relational Language



วัน เดือน ปี.....	15.ค.ค.2541
เลขทะเบียน.....	038970
เลขเรียกหนังสือ.....	T.2011 ๗ 262 ก

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มี

038970

การพัฒนาภาษาริเลชันแนลโดยใช้วิธีการเชิงวัตถุ
The Development of Object - Base relational Language



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิศวกรรมคอมพิวเตอร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2540

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ปีการศึกษา 2540

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาภาษาริเลชั่นแนลโดยใช้วิธีการเชิงวัตถุ

ผู้จัดทำ

1. นายศิริพงศ์ วิเศษมงคลชัย

2. นายสิริโรจน์ โรจน์วัฒนกุล

— อ.ว. จิตระ อาจารย์ที่ปรึกษา
(อ.อ. อ.อ. จิตระ)



การพัฒนาภาษารีเลชันแนลโดยใช้วิธีการเชิงวัตถุ

นาย ศิริพงศ์ วิเศษมงคลชัย

นาย สิริโรจน์ โรจน์วัฒนกุล

อ. สุภูมิตร จิตตะยโสธร

ปีการศึกษา 2540

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของโครงการวิจัยเพื่อศึกษาการนำหลักการเชิงวัตถุ (Object - Oriented Concept) มาใช้กับงานด้านฐานข้อมูลแบบรีเลชันแนล (Relational Database) รวมทั้งวิเคราะห์และเปรียบเทียบระหว่างการเขียนโปรแกรมในรูปแบบโปรซีเคออรอล (Procedural Programming) กับ การเขียนโปรแกรมโดยใช้หลักการเชิงวัตถุ

โครงการวิจัยนี้ได้นำภาษาสมอลทอล์ค (Smalltalk Language) มาใช้ในการทดลองสร้างโปรแกรมที่นำหลักการมาจากการทำงานของโปรแกรมวิซวลเอสคิวแอล (Program Visual SQL) และทำการค้นหาและแสดงผลข้อมูลโดยได้ออกแบบสร้างภาษาที่นำหลักการมาจากการทำงานของเอสคิวแอล (SQL)

The Development of Object - Base relational Language

Siripong Visasmongkolchai

Siroj Rojwathanakul

Suphamit Chittayasotorn

1997

Abstract

This thesis deals with the study of the Object - Oriented Concept to apply to the Relational database works. This includes the analysis and making the comparison between Procedural Programming and Object - Oriented Programming.

This study uses the Object - Oriented Language called Smalltalk to write the program that have a concept of the Visual SQL program to search and give results by using the concept of SQL language.

สารบัญ

หน้า

บทคัดย่อภาษาไทย.....	I
บทคัดย่อภาษาอังกฤษ.....	II
สารบัญ.....	III
สารบัญตาราง.....	VI
สารบัญภาพ.....	VII
บทที่ 1 บทนำ.....	1
1.1 ความสำคัญและที่มา.....	1
1.2 วัตถุประสงค์ของงานวิจัย.....	1
1.3 ขอบเขตของงานวิจัย.....	2
1.4 วิธีการดำเนินงาน.....	3
บทที่ 2 ทฤษฎีที่เกี่ยวข้อง.....	4
2.1 หลักการเชิงวัตถุ.....	4
2.1.1 ออบเจกต์.....	4
2.1.2 คลาส.....	4
2.1.3 เมธอด.....	4
2.1.4 เมสเสจ.....	4
2.1.5 ลำดับชั้นของคลาส (Class Hierarchy).....	4
2.2 คุณสมบัติของหลักการเชิงวัตถุ.....	5
2.2.1 Encapsulation.....	5
2.2.2 Abstraction.....	6
2.2.3 Polymorphism.....	6
2.2.4 Inheritance.....	7
2.3 การเขียนโปรแกรมโดยใช้หลักการเชิงวัตถุ.....	7
2.3.1 การเกิดแนวความคิดในหลักการเชิงวัตถุ.....	7
2.3.2 หลักการของการเขียน โปรแกรมเชิงวัตถุ.....	8
2.3.3 เป้าหมายหลักในการพัฒนาซอฟต์แวร์แบบ OOP.....	8
2.3.4 คุณสมบัติที่สำคัญของ OOP.....	8
2.3.5 ความแตกต่างของการแก้ปัญหาโดย OOP กับแบบ โพรซีเดรอล.....	9
2.3.6 บทสรุปการเขียนโปรแกรมเชิงวัตถุ.....	10

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4	ฐานข้อมูลแบบรีเลชันแนล.....	10
2.4.1	โครงสร้างฐานข้อมูลแบบรีเลชันแนล.....	11
2.4.2	ลักษณะการทำงานของตัวปฏิบัติการต่างๆ.....	14
2.5	เอสคิวแอล (SQL)	15
2.5.1	ทฤษฎีพื้นฐานของภาษาเอสคิวแอล.....	15
2.6	วิซวลเอสคิวแอล (Visual SQL)	17
2.6.1	การนำรูปแบบวิซวล (Visual Approach) มาใช้ในการเข้าถึงระบบฐานข้อมูล.....	17
บทที่ 3	โปรแกรมภาษาสมอลทอล์ค (Smalltalk)	19
3.1	ประวัติความเป็นมาของภาษาสมอลทอล์ค.....	19
3.2	หลักการพื้นฐานของสมอลทอล์ค.....	19
3.2.1	ออบเจค	19
3.2.2	ตัวแปร (Variable)	20
3.2.3	คลาส.....	20
3.2.4	เมธอด.....	22
3.2.5	เมสเสจ.....	23
3.2.6	Object and message Are Safe.....	25
3.3	รูปแบบของคำสั่งในภาษาสมอลทอล์ค.....	26
3.3.1	การกำหนดค่า.....	26
3.3.2	การเรียกใช้ฟังก์ชัน.....	26
3.3.3	คำสั่งเงื่อนไข.....	27
3.3.4	คำสั่งทำซ้ำ.....	28
3.4	บล็อกอาร์กิวเมนต์.....	29
3.4.1	เมสเสจ do:	29
3.4.2	เมสเสจ select:	30
3.4.3	เมสเสจ reject:	30
3.4.4	เมสเสจ collect:	30
3.5	การส่งค่ากลับของฟังก์ชัน.....	30
3.6	การใช้สมอลทอล์คไฟว์ (smalltalk/V)	31
3.7	เมนู.....	31
3.8	การบำรุงรักษาระบบ.....	32
3.8.1	การบำรุงรักษา.....	32
3.8.2	การบันทึกการเปลี่ยนแปลงแบบอัตโนมัติ.....	32
3.8.3	ความสำคัญของการเก็บอิมเมจ.....	32
3.8.4	การลดขนาดไฟล์เซนจ์ล็อก (Compressing Change Log).....	32

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น เมื่ออนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.8.5 การลดขนาดของไฟล์ซอร์ส (Vompressing Source)	33
3.8.6 การกู้ระบบ (Surviving a System Crash)	33
3.8.7 การกู้ข้อมูล (Recovery From a System Crash)	34
บทที่ 4 การออกแบบสร้างโปรแกรม.....	35
4.1 แนวคิดของโปรแกรม.....	35
4.2 การกำหนดคลาสและตัวแปรที่ใช้ภายในคลาส.....	35
4.2.1 ศึกษาคลาสต่างๆ ที่ใช้ในการเก็บข้อมูลในภาษาสมอลทอลล์.....	35
4.2.2 ทำการเลือกคลาสที่เหมาะสม.....	38
4.3 ข้อกำหนดของระบบงาน.....	42
4.4 การกำหนดเมธอดในแต่ละคลาส.....	42
4.4.1 เมธอดในคลาส RDB.....	42
4.4.2 เมธอดในคลาส VisualSQL.....	43
4.4.3 เมธอดในคลาส Data.....	48
4.4.4 องค์ประกอบโดยรวมของแต่ละคลาส.....	66
4.5 หลักการทำงานของโปรแกรม.....	66
4.5.1 ส่วนโปรแกรมที่นำหลักการทำงานมาจากโปรแกรมวิซวลเอสคิวแอล.....	66
4.5.2 ส่วนโปรแกรมที่นำหลักการทำงานมาจากภาษาเอสคิวแอล.....	69
บทที่ 5 บทวิจารณ์และสรุป.....	70
5.1 วิจารณ์การเขียนโปรแกรมเชิงวัตถุ.....	70
5.1.1 การศึกษาทฤษฎี.....	70
5.1.2 การพัฒนาระบบงานด้วยโปรแกรมสมอลทอลล์.....	70
5.1.3 ภาษาที่เหมาะสมจะใช้กับฐานข้อมูลสำหรับงานทางด้านสารสนเทศ.....	70
5.1.4 ปัญหาที่เกิดจากการเขียน โปรแกรมสมอลทอลล์.....	71
5.1.5 ปัญหาที่เกิดขึ้นจากการ Inheritance.....	73
5.1.6 Reuse of Software.....	73
5.2 สรุปผลการวิจัย.....	74
5.3 ข้อเสนอแนะ.....	74
ภาคผนวก ก คู่มือการใช้งาน โปรแกรม.....	75
กิตติกรรมประกาศ.....	84
บรรณานุกรม.....	85

สารบัญตาราง

ตารางที่

หน้า

1 - 1 การเปรียบเทียบการทำงานของภาษาเชิงวัตถุ.....	2
4 - 1 ตาราง President ที่ใช้ทดสอบโปรแกรม.....	38
4 - 2 ตาราง PresMarriage ที่ใช้ทดสอบโปรแกรม.....	39
4 - 3 ข้อมูลอินสแตนซ์ภายในคลาส RDB	40
4 - 4 ตัวอย่าง Physical Database ในคลาส RDB.....	40
4 - 5 ข้อมูลอินสแตนซ์ภายในคลาส VisualSQL.....	41
4 - 6 ตัวอย่างส่วนที่ติดต่อกับผู้ใช้งานในคลาส VisualSQL.....	41



สารบัญภาพ

รูปที่

หน้า

2 - 1 แสดงตัวอย่างของ Encapsulation ของแคปซูล.....	6
2 - 2 แสดงคุณสมบัติ Inheritance.....	7
2 - 3 แสดงตารางความสัมพันธ์ข้อมูลลูกค้า.....	11
2 - 4 แสดงแถว , คอลัมน์ , และฟิลด์ ในตาราง.....	16
2 - 5 แสดงตาราง TEACHERS.....	17
2 - 6 แสดงตารางการรับข้อมูลจากผู้ใช้.....	18
4 - 1 แสดงหลักการทำงานของโปรแกรมที่นำหลักการมาจากโปรแกรมวิซวลเบสิก.....	35
4 - 2 แสดง hierarchy ของคลาส Collection.....	36
4 - 3 ข้อมูลอินสแตนซ์ภายในคลาส Data ที่เป็นซับคลาสของคลาส OrderedCollection.....	41
4 - 4 ตัวอย่างฐานข้อมูลที่น่าไปค้นหาจริงๆ อยู่ในคลาส Data.....	42
4 - 5 แสดง Flow chart ของเมธอด makeConditionTable.....	51
4 - 6 แสดง Flow chart ของเมธอด getConditionTable.....	51
4 - 7 แสดง Flow chart ของเมธอด putname:.....	52
4 - 8 แสดง Flow chart ของเมธอด getname.....	52
4 - 9 แสดง Flow chart ของเมธอด runVisualSQL.....	53
4 - 10 แสดง Flow chart ของเมธอด run.....	53
4 - 11 แสดง Flow chart ของเมธอด scan.....	54
4 - 12 แสดง Flow chart ของเมธอด chkOrGroupSelect:.....	54
4 - 13 แสดง Flow chart ของเมธอด chkHaving:.....	55
4 - 14 แสดง Flow chart ของเมธอด chkOrHaving:.....	55
4 - 15 แสดง Flow chart ของเมธอด chkWhere:.....	56
4 - 16 แสดง Flow chart ของเมธอด chkOrWhere:.....	56
4 - 17 แสดง Flow chart ของเมธอด createSQL.....	57
4 - 18 แสดง Flow chart ของเมธอด sjoin:.....	57
4 - 19 แสดง Flow chart ของเมธอด swhere:.....	58
4 - 20 แสดง Flow chart ของเมธอด sgroupBy:.....	58
4 - 21 แสดง Flow chart ของเมธอด shaving:.....	59
4 - 22 แสดง Flow chart ของเมธอด sselect:.....	59
4 - 23 แสดง Flow chart ของเมธอด sorder:.....	60

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น เมื่ออนุญาตให้ไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4 - 24 แสดง Flow chart ของเมธอด select:	60
4 - 25 แสดง Flow chart ของเมธอด where:	61
4 - 26 แสดง Flow chart ของเมธอด groupBy:	61
4 - 27 แสดง Flow chart ของเมธอด having:	62
4 - 28 แสดง Flow chart ของเมธอด orderAsc:	62
4 - 29 แสดง Flow chart ของเมธอด orderDec:	63
4 - 30 แสดง Flow chart ของเมธอด join:	63
4 - 31 แสดง Flow chart ของเมธอด sum:	64
4 - 32 แสดง Flow chart ของเมธอด min:	64
4 - 33 แสดง Flow chart ของเมธอด max:	65
4 - 34 แสดง Flow chart ของเมธอด avg:	65
4 - 35 แสดงองค์ประกอบของคลาส RDB , VisualSQL , Data.....	66
4 - 36 แสดงขั้นตอนการทำงานของโปรแกรมที่นำหลักการทำงานมาจากโปรแกรมเอสคิวแอล.....	68
4 - 37 แสดงขั้นตอนการทำงานของโปรแกรมที่นำหลักการทำงานมาจากภาษาเอสคิวแอล.....	69

บทที่ 1

บทนำ

1.1 ความสำคัญและที่มา

ในอดีตที่ผ่านมาการเขียนโปรแกรมแบบโพรซีเดรอล (Procedural Programming) ได้เป็นที่นิยมอย่างมาก เนื่องจากการเขียนโปรแกรมตามลักษณะแนวคิดของมนุษย์ ทำให้เขียนโปรแกรมได้ง่ายและรวดเร็ว และใช้วิธีการเก็บรวบรวมข้อมูลต่างๆ เป็นแฟ้มข้อมูล (File) หรือ ฐานข้อมูล (Database) แยกกันกับโปรแกรมในระบบงาน แต่เมื่อได้ทำการใช้งานไปนานๆ จะพบว่า การเขียนโปรแกรมในลักษณะนี้มีข้อจำกัดอยู่มาก เช่น ในเรื่องของ การนำโปรแกรมมาพัฒนาให้ทันสมัยนั้นทำได้ยาก เพราะต้องทำการศึกษาโปรแกรมเก่าให้เข้าใจอย่างแท้จริงก่อนลงมือแก้ไข อีกทั้งยังไม่มีความปลอดภัยของข้อมูล เพราะทุกโพรเซส (Process) สามารถนำข้อมูลนั้นไปใช้งานได้ตามต้องการ และมีปัญหาที่พบอีกมากมาย ดังจะ ได้อธิบายไว้ในบทที่ 5

ทำให้เกิดแนวความคิดการเขียนโปรแกรมเชิงวัตถุ (Object - Oriented Programming) ขึ้นมา โดยเป็นวิธีใหม่ในการพัฒนาซอฟต์แวร์ (Software) เพื่อเพิ่มประสิทธิภาพในการผลิตซอฟต์แวร์ วิธีนี้ทำให้การออกแบบและการเขียนซอฟต์แวร์มีความใกล้ชิดกันมากขึ้น และจะกำหนดสิ่งต่างๆ ที่อยู่ภายในขอบเขตของระบบงานเป็นออบเจกต์ (Object) ซึ่งในออบเจกต์จะประกอบด้วยข้อมูล (Data) และเมธอด (Method) รวมกันอยู่ภายในออบเจกต์ (Encapsulation) เมธอดเปรียบเทียบกับขั้นตอนการทำงานเดียวกันกับโพรซีเจอร์ (Procedure) ในโปรแกรมระบบงานแบบเดิมนั้นเอง การเรียกใช้เมธอดหรือการสั่งให้เกิดการกระทำตามที่กำหนดไว้ในเมธอดคือการส่งเมสเสจ (Message) ไปยังออบเจกต์นั้นๆ ดังนั้นหากมีการเปลี่ยนแปลงวิธีการกระทำ (Function) ใดๆ ของออบเจกต์ จะไม่มีผลกระทบกับเมสเสจที่ส่งไปยังออบเจกต์นั้นและจะไม่มีผลกระทบกับออบเจกต์อื่นๆ ด้วย ซึ่งนับว่าเป็นผลดีอย่างมากกับระบบงานเพราะโดยปกติแล้วพบว่าระบบงานมักจะมีการเปลี่ยนแปลงวิธีการปฏิบัติอยู่เสมอๆ เมธอดสามารถใช้อธิบายถึงพฤติกรรมของออบเจกต์แต่ละออบเจกต์ได้ ออบเจกต์ที่มีพฤติกรรมแบบเดียวกันสามารถจัดรวมกลุ่มเป็นกลุ่มเดียวกันได้เรียกว่า คลาส (Class) ในระหว่างคลาสสามารถที่จะถ่ายทอดคุณสมบัติ (Inheritance) จากคลาสหนึ่งไปยังอีกคลาสหนึ่งได้ โดยเรียกคลาสที่ถ่ายทอดคุณสมบัติว่า ซุปเปอร์คลาส (Superclass) และเรียกคลาสที่ได้รับการถ่ายทอดคุณสมบัติว่า ซับคลาส (Subclass) ซึ่งทำให้การบำรุงรักษาระบบงานทำได้ง่าย และสามารถขจัดปัญหาต่างๆ ในการเขียนโปรแกรมแบบโพรซีเดรอลได้อีกด้วย

1.2 วัตถุประสงค์ของงานวิจัย

1.2.1 ศึกษาความเป็นไปได้ในการนำหลักการเชิงวัตถุ (Object - Oriented Concept) มาใช้ในงานด้านฐานข้อมูลแบบรีเลชันแนล (Relational Database)

1.2.2 ศึกษาการใช้ข้อดีของการเขียนโปรแกรมเชิงวัตถุ มาใช้ในการเขียนโปรแกรมที่นำหลักการมาจากการทำงานของโปรแกรมวิซวลเอสคิวแอล (Visual SQL) และทำการค้นหาและแสดงผลข้อมูล โดยได้ออกแบบสร้างภาษาที่นำหลักการทำงานมาจากภาษาเอสคิวแอล (SQL Language)

1.2.3 วิเคราะห์และเปรียบเทียบข้อดีข้อเสียของการเขียนโปรแกรมเชิงวัตถุ กับ การเขียนโปรแกรมแบบโพรซีเดรอล

1.3 ขอบเขตของงานวิจัย

งานวิจัยนี้ได้ทำการทดลองสร้างโปรแกรมที่นำหลักการมาจากการทำงานของโปรแกรมวิซวลเอสคิวแอลและทำการค้นหาและแสดงผลข้อมูล โดยได้ออกแบบสร้างภาษาที่นำหลักการมาจากการทำงานของภาษาเอสคิวแอลและมีภาษาเชิงวัตถุให้ทำการเลือกมาใช้ในการพัฒนาระบบงานดังนี้

Feature vs Language	Abstract Data Type	Single Inheritance	Multiple Inheritance	Dynamic Binding	All are Objects	Library Support
Simula	y	y	n	y	n	n
CLU	y	n	n	y	n	n
Ada	y	n	n	n	n	y
Smalltalk	y	y	n	y	y	y
Objective-C	y	y	n	y	n	y
C++	y	y	y	y	n	y
Object Pascal	y	y	n	y	n	n
Beta	y	y	n	y	n	n

ตารางที่ 1-1 การเปรียบเทียบของภาษาเชิงวัตถุ

โดยพัฒนาระบบงานบนภาษาสมอลทอลล์ เพราะภาษาสมอลทอลล์เป็นต้นแบบของซอฟต์แวร์เชิงวัตถุ ในภาษาสมอลทอลล์สามารถนำแนวความคิดที่ว่าทุกอย่างเป็นออบเจกต์มาใช้ได้จริง โดยเมื่อมองความสัมพันธ์ระหว่างคลาส คลาสด้านบนสุดจะมีชื่อว่า “ ออบเจค “ และคลาสอื่นๆ ทุกคลาสจะมีคลาส “ ออบเจค ” เป็นซูเปอร์คลาส ส่วนคุณสมบัติที่สำคัญอื่นๆ ของสมอลทอลล์นั้น จะได้อธิบายในบทที่ 3 ต่อไป

ในงานวิจัยนี้จะไม่สนใจในแง่ที่มาของแบบจำลองข้อมูลระดับแนวคิด โดยจะสมมติว่าแบบจำลองข้อมูลถูกจัดเก็บในฐานข้อมูลเรียบร้อยแล้ว งานวิจัยนี้จะนำข้อมูลนั้นมาใช้ได้เลย โดยไม่สนใจว่าจะได้ข้อมูลเหล่านั้นมาได้อย่างไร และนั่นก็หมายความว่า การออกแบบสร้างโปรแกรมนั้นจะออกแบบทั้ง

ระบบ แต่ในการพัฒนาระบบนั้น เราจะพัฒนาเฉพาะส่วนที่ติดต่อกับผู้ใช้นั้น จะไม่สนใจส่วนที่ติดต่อกับผู้ออกแบบแบบจำลองระดับแนวคิด

นอกจากนั้นโครงการนี้ยังถือเป็นโครงการที่ทดลองสร้าง เพื่อศึกษาความเป็นไปได้ที่การเขียนโปรแกรมเชิงวัตถุจะสามารถนำไปใช้ได้จริง และเป็นที่ยอมรับของนักเขียนโปรแกรมรุ่นต่อไป

1.4 วิธีการดำเนินงาน

งานวิจัยในโครงการนี้จะเริ่มด้วยการศึกษาทฤษฎีพื้นฐานต่างๆ ที่เกี่ยวข้องกับงานวิจัย ซึ่งก็มีเรื่องหลักๆ อยู่ 3 เรื่อง คือ ทฤษฎีพื้นฐานของออบเจกต์ หลักการใช้งานโปรแกรมวิซวลเอสคิวแอล พร้อมทั้งหลักการที่เกี่ยวข้องทั้งหมด และภาษาสมอลทอล์ค ซึ่งเป็นภาษาเชิงวัตถุที่ใช้ในการทำงานวิจัยนี้ มีรายละเอียดดังในบทที่ 2 และ 3

จากนั้นจะนำเอาความรู้ที่ได้ศึกษาทั้งหมดมาออกแบบสร้างโปรแกรมนำหลักการมาจากการทำงานของโปรแกรมวิซวลเอสคิวแอล ซึ่งมีรายละเอียดในบทที่ 4 เมื่อได้ทำการสร้างโปรแกรมนำหลักการมาจากเรียบร้อยแล้วจะนำเอาความรู้ที่ได้จากการเขียนโปรแกรมเองนั้นมาวิเคราะห์ถึงข้อดี ข้อเสีย และความเหมาะสมในการนำภาษาเชิงวัตถุมาประยุกต์ใช้กับงานฐานข้อมูลแบบรีเลชันแนล ซึ่งจะมีรายละเอียดในบทที่ 5

บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

2.1 หลักการเชิงวัตถุ มีคำศัพท์ที่เกี่ยวข้องกับหลักการเชิงวัตถุอยู่ 5 คำ คือ

2.1.1 ออบเจกต์

ออบเจกต์ในหลักการเชิงวัตถุจะตรงกับคำว่าเอนติตี้ (Entity) ในไนแอม (NIAM) ออบเจกต์จะอยู่ภายใต้คลาส ซึ่งเหมือนกับเอนติตี้ที่อยู่ภายใต้ชนิดของเอนติตี้ และทุกออบเจกต์ในระบบจะต้องไม่ซ้ำกัน (Unique)

ออบเจกต์จะง่ายหรือซับซ้อนขึ้นอยู่กับระบบ สิ่งแรกจะต้องเตรียมสร้างคลาสพื้นฐาน เช่น Integer, string, character ฯลฯ ซึ่งผู้ใช้สามารถสร้างออบเจกต์ของตัวเองขึ้นมาได้จากคลาสพื้นฐานได้ หรือออบเจกต์ที่ซับซ้อนสามารถสร้างจากการรวมกันของออบเจกต์ที่มีอยู่แล้ว

ออบเจกต์ ประกอบด้วย 2 ส่วน คือ

- ข้อมูล เป็นสถานะ (state) ของออบเจกต์นั้น
- เมธอด เป็นวิธีการเข้าถึงข้อมูลและจัดการ ปรับปรุง เปลี่ยนแปลงสถานะของออบเจกต์นั้น

2.1.2 คลาส

คลาสในออบเจกต์ ตรงกับคำว่าชนิดของเอนติตี้ หรือคลาสของชนิดเอนติตี้ (Entity type class) ในไนแอม ใช้จัดกลุ่มของออบเจกต์ตามคุณสมบัติ และผู้ใช้สามารถติดต่อกับคลาสโดยผ่านทางเมธอดภายในคลาส หรือซูเปอร์คลาส เท่านั้น (superclass คือ parent class.)

2.1.3 เมธอด

เป็นการทำงาน (Operation) ที่สำคัญหรือเป็นวิธีการกระทำที่สามารถทำงานกับออบเจกต์ได้ แต่ละคลาสจะมีเมธอดของตัวเอง กลุ่มของเมธอดที่นำไปใช้กับคลาสที่กำหนด จะต้องคำนึงถึงข้อจำกัด (definition) ที่เก็บในคลาสนั้น

2.1.4 เมสเสจ

ออบเจกต์จะติดต่อกับออบเจกต์อื่น โดยผ่านทางเมสเสจ การส่งเมสเสจ คือ การไปเรียกเมธอดมาทำงาน แล้วส่งผลลัพธ์ไปยังผู้ส่งเมสเสจ (Sender)

2.1.5 ลำดับชั้นของคลาส (Class Hierarchy)

ลำดับชั้นของคลาสในหลักการเชิงวัตถุตรงกับคำว่า Subtype ในไนแอม และเหมือนกับ โครงสร้างลำดับชั้น (Hierarchy Concept) เช่น ถ้าคลาส Integer เป็นซับคลาสของคลาส Number แล้ว ทุก ๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

อินสแตนซ์ (instance) ของคลาส Integer จะเป็นอินสแตนซ์ของคลาส Number โดยอัตโนมัติ และคลาส Integer จะได้รับการถ่ายทอดคุณสมบัติจากคลาส Integer ด้วย

การถ่ายทอดคุณสมบัติมี 2 ชนิด คือ

- Structure Inheritance เช่น B ได้รับการถ่ายทอดคุณสมบัติจากตัวแปรที่ใช้ภายในคลาส (Instance variable) จาก A
- Behavioral Inheritance เช่น B ได้รับการถ่ายทอดเมธอดจาก A

2.2 คุณสมบัติของหลักการเชิงวัตถุ คุณสมบัติที่สำคัญของออบเจกต์ มี 4 ประการ คือ

2.2.1 ENCAPSULATION

คือ การป้องกันของโครงสร้างข้อมูล (data structure) กับวิธีการทำงานที่เกี่ยวข้อง หรือ ความสามารถในการซ่อนข้อมูลจากระบบภายนอกได้

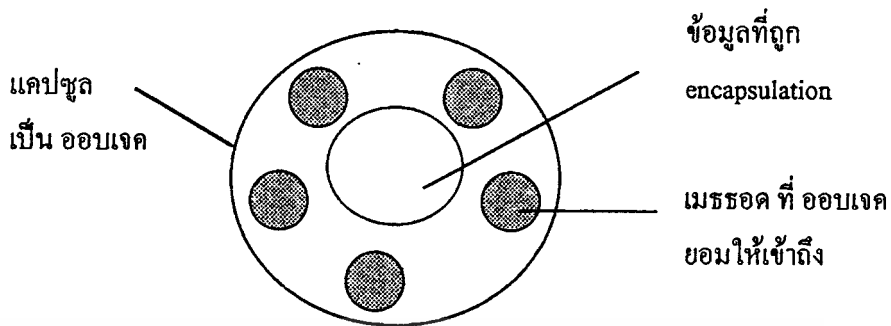
ความหมายของ encapsulation คือ แต่ละอินสแตนซ์ในคลาส (ออบเจกต์) คือ 1 encapsulation ออบเจกต์ (encapsulation) มีลักษณะดังนี้

- 1) **Shared Data** องค์ประกอบส่วนนี้จะมีค่าเหมือนกันสำหรับออบเจกต์ทั้งหมดที่เป็นอินสแตนซ์ในคลาส อินสแตนซ์ทั้งหมดในคลาสสามารถเข้าถึง shared data นี้ได้
- 2) **Private Data** แต่ละออบเจกต์ จะกำหนดส่วนประกอบของแต่ละอินสแตนซ์ในคลาส ซึ่งออบเจกต์ทั้งหมดที่เป็นอินสแตนซ์ในคลาสจะมีองค์ประกอบส่วนนี้ แต่จะมีค่าต่างกัน ออบเจกต์หนึ่ง ๆ จะสามารถเข้าถึง private data ของตัวมันเองได้เท่านั้น
- 3) **Message** กลุ่มของเมสเสจ คือ สิ่งที่ออบเจกต์ตอบสนองต่อออบเจกต์ทั้งหมด ที่อยู่ในคลาสเดียวกัน จะตอบสนองต่อกลุ่มของเมสเสจเดียวกัน
- 4) **Global -Shared Data** เป็นข้อมูลที่สามารถถูกใช้ร่วมกันโดยอินสแตนซ์ต่างคลาสได้ Global -Shared data จะมีค่าเหมือนกันสำหรับทุกออบเจกต์ที่ใช้ร่วมกันอยู่

ข้อดีของ encapsulation คือ

- 1) กำหนดขอบเขต (scope) ของซอฟต์แวร์ภายในตัวมันอย่างชัดเจน
- 2) มีการเชื่อมต่อที่ดีคือ ระบุได้ว่าองค์ประกอบของซอฟต์แวร์แต่ละตัวกระทำกับตัวอื่นอย่างไร
- 3) มีการป้องกัน internal implementation โดยให้รายละเอียดของ functionality ซึ่งจัดหาโดยองค์ประกอบของซอฟต์แวร์

ตัวอย่างของ encapsulation เช่น ขาเม็ดยาแคปซูล ซึ่งมองไม่เห็นตัวยาภายใน รู้แต่ว่ายานี้รักษาโรคอะไร ดังรูปที่ 2 - 1



รูปที่ 2 - 1 แสดงตัวอย่างของ encapsulation ของ แคปซูล

2.2.2 ABSTRACTION

คือ การแทนความคิดที่ยู่ยากซับซ้อน ให้สั้น กระทัดรัด รัดกุมขึ้น โดยใช้ออบเจกต์ ซึ่งในออบเจกต์จะรวมทั้ง ข้อมูล และ การทำงาน ไว้ในตัวเลย หรืออาจกล่าวได้ว่า ออบเจกต์ เป็น encapsulation of abstractions

การกำหนด abstraction ขึ้นมาใหม่ในวิธีการแก้ปัญหาแบบเชิงวัตถุ จะต้องการออบเจกต์ใหม่ และสำหรับแต่ละออบเจกต์ใหม่ ต้องกำหนดคุณสมบัติ และเมสเสจที่จะติดต่อจัดการเกี่ยวกับออบเจกต์นั้นได้

วิธีการแก้ปัญหานั้น อาจประกอบด้วยหลายออบเจกต์ บางออบเจกต์ก็มีคุณสมบัติคล้ายกัน ซึ่งเรียกกลุ่มออบเจกต์ที่มีคุณสมบัติคล้ายกันนี้ว่า คลาสคุณสมบัติพื้นฐานของออบเจกต์ ในคลาสหนึ่ง ๆ จะถูกกำหนดไว้ในคลาส แต่ละออบเจกต์เป็นอินสแตนซ์ของคลาสและการกระทำของออบเจกต์ ถูกกำหนดโดยกลุ่มของคุณสมบัติ

2.2.3 POLYMORPHISM

คือ การที่เราส่งเมสเสจที่เหมือนกัน ไปในออบเจกต์ที่ต่างกัน แต่ละออบเจกต์จะตอบสนองออกมาไม่เหมือนกัน ตามแต่วิธีและหน้าที่ของออบเจกต์

ความสามารถที่ใช้เมสเสจเหมือนกัน สำหรับการกระทำที่เหมือนกัน ไปยังออบเจกต์ต่างชนิดกัน มีลักษณะเหมือนกับการที่มนุษย์คิดในการแก้ปัญหานั้น ๆ

ตัวอย่างของ polymorphism

- | | | |
|-----------|------------|---------------|
| ● AB + CD | ได้ผลลัพธ์ | ABCD (concat) |
| ● 5 + 3 | ได้ผลลัพธ์ | 8 (add) |

ทั้ง 2 แบบ เป็นการส่งเมสเสจ '+' เข้าไปยังออบเจกต์ ภายในคลาส String และ Integer ตามลำดับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.4 INHERITANCE

คือ การสร้างคลาสใหม่ขึ้นมา โดยมีการสืบทอดคุณสมบัติพื้นฐานที่มาจากคลาสเดิม แต่จะมีข้อมูล หรือ เมธอด ที่พิเศษเป็นของตัวเองเพิ่มขึ้นมาจากคลาสเดิม

ถ้า ชั้นคลาส 1 เป็นชั้นคลาสของคลาส A ออบเจกต์ที่เป็นอินสแตนซ์ของชั้นคลาส 1 จะมีคุณสมบัติพิเศษกว่าอินสแตนซ์ของคลาส A แต่อย่างน้อยที่สุด จะต้องมียุทธศาสตร์เหมือนอินสแตนซ์ของคลาส A เรียกว่า อินสแตนซ์ของชั้นคลาส 1 ซึ่งจะถ่ายทอดคุณสมบัติมาจากคลาส A

การถ่ายทอดคุณสมบัติ นี้จะรวมทั้ง private data , shared data และ message คลาส A จะถูกเรียกว่าเป็น ซุปเปอร์คลาส ของคลาส 1

Inheritance เป็นคุณสมบัติของออบเจกต์ที่เป็นอินสแตนซ์ของคลาสจะถ่ายทอดคุณสมบัติจาก ซุปเปอร์คลาสของมัน ชั้นคลาสมีหลายระดับ คลาสที่อยู่ระดับต่ำสุด ก็จะมีคุณสมบัติของซุปเปอร์คลาสหลายคลาส

ชั้นคลาสสามารถเข้าถึง private data , shared data , message ของซุปเปอร์คลาสของมันได้ทุกคลาส

ตัวอย่างของ inheritance เช่น

คลาสของทหาร แต่ละประเภทของทหารจะถูกแยกออกเป็นชั้น ๆ ที่ยอดบนสุด เป็นคลาสพื้นฐาน คือ ทหาร และชั้นถัดมาเป็นทหารบก , ทหารเรือ และ ทหารอากาศ ซึ่งได้รับการถ่ายทอดคุณสมบัติของ ซุปเปอร์คลาส ข้างบนมา และมีคุณสมบัติต่าง ๆ เพิ่มเติมสำหรับทหารแต่ละประเภท ดังรูปที่ 2 - 2



รูปที่ 2 - 2 แสดงคุณสมบัติ Inheritance

2.3 การเขียนโปรแกรมโดยใช้หลักการเชิงวัตถุ

2.3.1 การเกิดแนวความคิดในหลักการเชิงวัตถุ

เดิมลักษณะการเขียนโปรแกรม เป็นแบบโพรซีเจอร์อล คือ การสั่งให้คอมพิวเตอร์ทำงานอย่างหนึ่ง โดยความหมายของแต่ละคำสั่ง (instruction) ซึ่งเป็นลำดับในโปรแกรม ถ้าโปรแกรมเล็ก ๆ จะไม่มีปัญหาหรือผลกระทบที่เห็นได้เด่นชัด ถ้าโปรแกรมมีขนาดใหญ่มากจะแก้ไขหาข้อผิดพลาดได้ยาก แต่มีข้อดีที่สามารถแบ่งโปรแกรมเป็นโปรแกรมย่อยได้ แต่ถ้าโปรแกรมมีความซับซ้อนมากเกินไปจะทำให้เกิดปัญหาขึ้นได้อีก

ถ้าเป็นการเขียนโปรแกรมเชิงวัตถุ จะรวมส่วนที่เป็นข้อมูลและการทำงาน ที่เรียกใช้ข้อมูลนั้นไว้ด้วยกัน เรียกว่า ออบเจกต์ เมื่อต้องการเข้าถึงข้อมูลในออบเจกต์ เมธอดจะเป็นตัวเข้าไปจัดการข้อมูล หลังจากนั้นจะทำการส่งข้อมูลที่ต้องการกลับมาตามที่ต้องการ เพื่อป้องกันการเข้าถึงข้อมูลของการกระทำอื่นๆ ซึ่งการเขียนโปรแกรมแบบนี้จะช่วยลดความยุ่งยากสำหรับโปรแกรมที่ซับซ้อน และยังช่วยให้ข้อมูลมีความปลอดภัยเพิ่มขึ้นด้วย

2.3.2 หลักการของการเขียนโปรแกรมเชิงวัตถุ (OOP)

การเขียนโปรแกรมเชิงวัตถุ เป็นวิธีการใหม่ในการพัฒนาซอฟต์แวร์ โดยมุ่งที่จะเพิ่มประสิทธิภาพในการผลิตซอฟต์แวร์ วิธีนี้ทำให้การออกแบบและการเขียนซอฟต์แวร์มีความใกล้ชิดกันมากขึ้น

หลักการสำคัญของการเขียนโปรแกรมเชิงวัตถุคือ abstract data type, คลาส, การจัดลำดับของชั้นคลาส, inheritance และ polymorphism

abstract data type เป็นแก่นของการเขียนโปรแกรมเชิงวัตถุ abstract data type จะรวมชนิดของข้อมูล และชุดของการทำงานเข้าไว้ด้วยกัน ซึ่งการทำงานจะกำหนดคุณสมบัติของชนิดของข้อมูลนั้น

การนิยามคลาส (class definition) ซึ่งจะกำหนดการทำงานของ abstract data type สามารถทำได้ด้วยการนิยามว่า จะเรียกใช้การทำงานของชนิดของข้อมูลนั้นอย่างไร การนิยามคลาสยังกำหนดโครงสร้างข้อมูลของคลาสด้วย

การทำงานของชนิดของข้อมูล จะแบ่งออกเป็น 2 ชนิด คือ

- 1) การทำงานแบบ Public คือ สามารถเรียกใช้ได้จากภายนอกคลาส
- 2) การทำงานแบบ Private คือ สามารถเรียกใช้ได้เฉพาะภายในคลาส

2.3.3 เป้าหมายหลักในการพัฒนาซอฟต์แวร์แบบ OOP

- 1) ทำให้การพัฒนาซอฟต์แวร์ใช้เวลาสั้นลง ต้นทุนต่ำ โดยการใช้คลาสที่สามารถนำกลับมาใช้ได้ และสร้างชั้นคลาสขึ้นมาเพื่อช่วยในการแก้ปัญหา
- 2) ทำให้ต้นทุนในการบำรุงรักษาซอฟต์แวร์ต่ำลง เพราะสามารถหาสิ่งที่ต้องการเปลี่ยนแปลงในซอฟต์แวร์ได้ง่าย และการเปลี่ยนแปลงไม่มีผลกระทบต่อภายนอกคลาสได้อย่างแน่นอน

2.3.4 คุณสมบัติที่สำคัญของ OOP

- 1) ข้อมูลที่เก็บในออบเจกต์ จะถูกคืนพื้นที่ในหน่วยความจำโดยอัตโนมัติ ถ้าไม่มีการใช้ข้อมูลนั้น
- 2) การกระทำ จะใช้การส่งเมสเสจไปที่ออบเจกต์
- 3) คุณสมบัติของออบเจกต์ จะระบุไว้ในคลาสเท่านั้น

2.3.5 ความแตกต่างระหว่างการแก้ไข้ปัญหาโดย OOP กับการเขียนโปรแกรมในแบบโปรแกรมเชิงวัตถุ

1) วิธีการเข้าถึงข้อมูล

OOP	จะส่งเมสเสจ (action) ไปยังออบเจก (ข้อมูล) และ ออบเจกจะตอบสนองต่อส่วนที่เป็นผลลัพธ์ไปยังผู้ส่ง
Structure	ข้อมูล (data) ถูกส่งไปที่โปรแกรมย่อย (action) โปรแกรมย่อยจะกระทำกับข้อมูลนั้น

2) Data Abstraction

OOP	data abstraction ถูกแทนที่โดยคลาสของออบเจก
Structure	data abstraction ถูกแทนที่โดยชนิดของข้อมูลของแต่ละภาษา

3) Functional Abstraction

OOP	Function abstraction ถูกแทนที่โดยเมสเสจ
Structure	Function abstraction ถูกแทนที่โดยโปรแกรมย่อยและการกระทำ

4) Encapsulation

OOP	หน่วยของ Encapsulation คือ ออบเจก
Structure	การ Encapsulation อยู่ในรูป Library Module ซึ่งส่วนประกอบหนึ่งอาจบรรจุมากกว่าหนึ่ง data abstraction

5) Inheritance and Polymorphism

OOP	สนับสนุนทั้ง 2 อย่าง ทำให้การแสดงความสัมพันธ์ระหว่าง ชั้นคลาส ไม่ขึ้นต่อกันและยังช่วยลดความซ้ำซ้อน
Structure	ไม่สนับสนุนทั้ง 2 อย่าง โดย Software Component และ Data Abstractions แทนด้วย Hierarchical Level ที่เท่ากัน และยังยุ่งยากในการตั้งชื่อ Procedure

6) Extensibility

OOP	คือ คุณสมบัติของภาษาทางคอมพิวเตอร์ ที่อนุญาตให้ ผู้ใช้ กำหนดโครงสร้างใหม่ได้
OOP	ออบเจกทั้งหมดที่มีสถานะที่เท่ากันทั้ง ออบเจกที่ถูกผู้สร้างสร้างขึ้นและออบเจกที่เป็นส่วนของ System Kernel จะมีความสำคัญและประสิทธิภาพเท่ากัน
Structure	การสร้างขึ้นมาใหม่ จะมีความสำคัญและประสิทธิภาพน้อยกว่าการที่สร้างจากระบบ

7) Iteration

OOP	จัดหาโครงสร้างที่เหมาะสมสำหรับการทำ Loop
Structure	มี Loop มากเกินไป

2.3.6 บทสรุปของการเขียนโปรแกรมเชิงวัตถุ

- 1) การเขียนโปรแกรมเชิงวัตถุ ประกอบด้วย การส่งเมสเสจไปยังออบเจกต์
- 2) ผลที่ได้จากการส่งเมสเสจไปยังออบเจกต์ คือ ออบเจกต์
- 3) วิธีการแก้ไขปัญหาประกอบด้วย
 - กำหนดออบเจกต์ใน Problem Statement
 - กำหนดเมสเสจที่เกี่ยวข้องกับแต่ละออบเจกต์
 - พัฒนาลำดับของเมสเสจที่มี ไปยังออบเจกต์เพื่อแก้ไขปัญหา
- 4) ภาษาที่ใช้หลักการเขียนโปรแกรมเชิงวัตถุ จะต้องมีความสมบัติ Encapsulation , Abstraction , Inheritance , Polymorphism
- 5) หน่วยหนึ่งของ Encapsulation คือออบเจกต์ซึ่งประกอบด้วย Private data , Shared data และ message description
- 6) Abstraction ถูกสนับสนุนโดยลำดับชั้นของคลาส ซึ่งใช้แทนแต่ละออบเจกต์ที่ต่างชนิดกัน ; Class description protocol กำหนดคุณสมบัติของแต่ละ Abstraction ; New Abstraction จะถูกเพิ่มโดยการเพิ่มชั้นคลาสขึ้นมาใหม่
- 7) คุณสมบัติของชั้นคลาสที่ถ่ายทอดมาจากซูเปอร์คลาส ประกอบด้วย Private data , shared data และ message
- 8) Inherited message เป็น polymorphism โดยชั้นคลาส
- 9) Polymorphism หมายถึง การส่งเมสเสจที่เหมือนกัน ไปยังคลาสต่างกันได้
- 10) Data คือ ข้อมูลที่ถูกเก็บในออบเจกต์และ Automatic Reallocation
- 11) สนับสนุนการเพิ่มขึ้นของการแก้ไข้ปัญหา
- 12) ไม่ว่าออบเจกต์ที่เกิดจากระบบ หรือที่ผู้ใช้สร้างขึ้น มีสถานะเท่ากัน
- 13) สนับสนุนในเรื่องต่าง ๆ ได้เต็มที่
- 14) การใช้ Iterative Structure และ Multilevel Index ถูกทำให้สั้นและเฉพาะส่วน โดยการแทนรายละเอียดใน Simple message

2.4 ฐานข้อมูลแบบรีเลชันแนล

ระบบฐานข้อมูลที่ใช้กันอยู่ในปัจจุบันสามารถแบ่งประเภทตามโมเดล (Model) ได้ 3 ประเภท คือ เน็ตเวิร์ก (Network) ไลราคี (Hierarchy) และ รีเลชันแนล ซึ่งรีเลชันแนลเป็น โมเดลในการเก็บข้อมูล ที่นิยม และแพร่หลายมากที่สุดซึ่งอาจกล่าวได้ว่าระบบฐานข้อมูลส่วนใหญ่ในโลกนี้เป็นฐานข้อมูลแบบรีเลชันแนล ซึ่งดูได้จากซอฟต์แวร์ที่เกี่ยวกับระบบจัดการฐานข้อมูลที่มีขายอยู่ทั่วไป ล้วนใช้โมเดลแบบรีเลชันแนลทั้งสิ้น

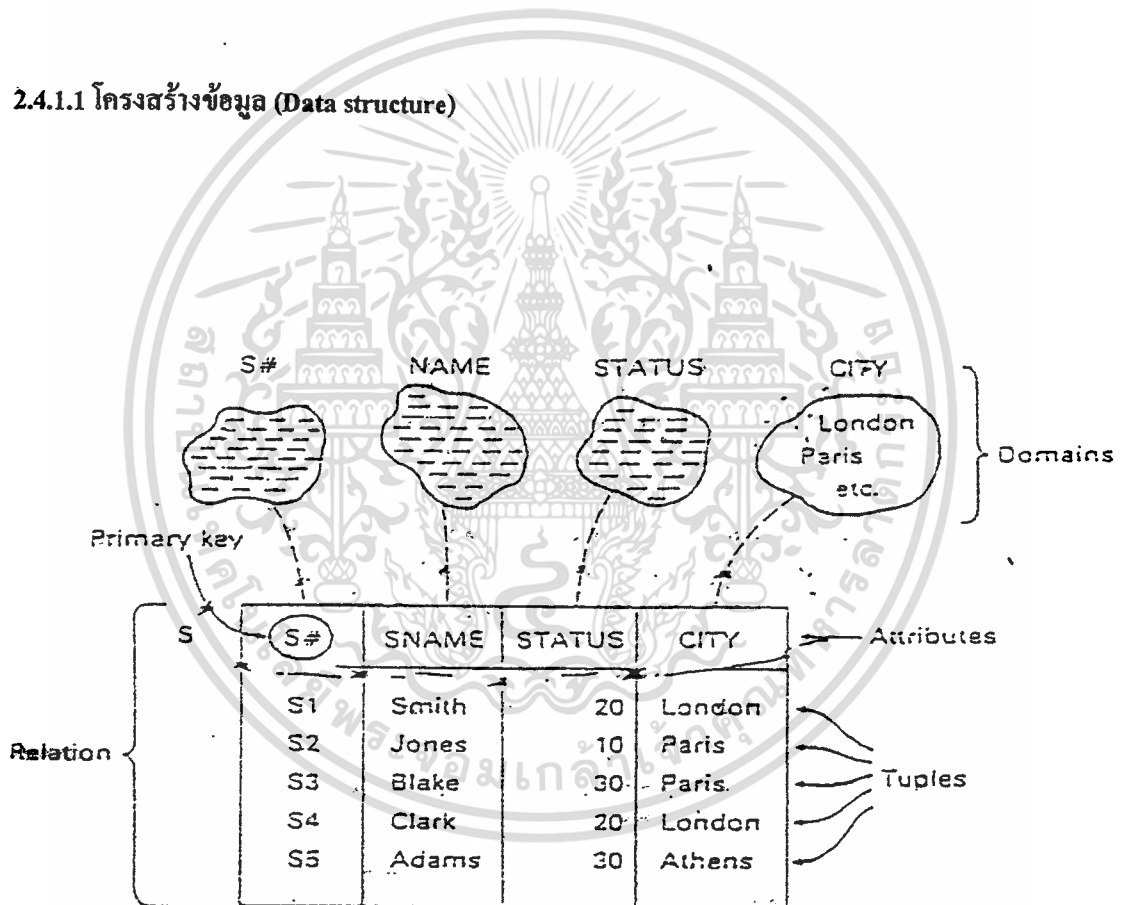
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4.1 โครงสร้างฐานข้อมูลแบบรีเลชันแนล

ฐานข้อมูลแบบรีเลชันแนลจะแสดงการเก็บข้อมูลในรูปของตาราง ซึ่งง่ายต่อความเข้าใจและสามารถตอบสนองต่อคำถามที่ซับซ้อนได้ดีกว่าโมเดลชนิดอื่น องค์ประกอบที่สำคัญของระบบจัดการฐานข้อมูลแบบรีเลชันแนลแบ่งเป็น 4 ส่วนคือ

- โครงสร้างข้อมูล
- คุณสมบัติของฐานข้อมูลรีเลชันแนล
- ความถูกต้องของข้อมูล
- การจัดการข้อมูล

2.4.1.1 โครงสร้างข้อมูล (Data structure)



รูปที่ 2 - 3 แสดงตารางความสัมพันธ์ข้อมูลลูกค้า

จากรูปที่ 2 - 3 มีคำเฉพาะที่ใช้ในการเรียกส่วนต่างๆ ของตารางข้อมูลดังนี้

แอททริบิวต์ (Attributes) หมายถึง ชื่อคอลัมน์ของตารางความสัมพันธ์ ซึ่งจะเป็นตัวที่บอกถึงคุณลักษณะต่างๆที่มีความสัมพันธ์กันแล้วประกอบขึ้นเป็นตารางฐานข้อมูล เช่น แอททริบิวต์ SNAME หมายถึง ชื่อพนักงาน แอททริบิวต์ CITY หมายถึง ชื่อเมือง มีความสัมพันธ์กันคือพนักงานชื่อ (SNAME) อาศัยอยู่ที่เมือง (CITY) เป็นต้น (ลักษณะของความสัมพันธ์ระหว่างแอททริบิวต์จะไม่สามารถมองเห็นได้จากตารางฐานข้อมูลแต่จะมองเห็นได้จากแบบจำลองฐานข้อมูล เช่น FD Diagram, แผนภาพ ในแอม หรือแบบจำลอง ER เป็นต้น)

โดเมน (Domains) หมายถึง ขอบเขตของข้อมูลในแต่ละแอททริบิวต์ เช่น ข้อมูลที่เก็บลงบนแอททริบิวต์ CITY จะต้องเป็นชื่อของเมืองต่างๆในทวีปยุโรปเท่านั้น หรือ ข้อมูลที่เก็บลงบนแอททริบิวต์ SNAME จะต้องเป็นชื่อคนเท่านั้นจะเป็นอย่างอื่นไม่ได้ เป็นต้น

ทูปเปิล (Tuples) หมายถึง แถวของตารางฐานข้อมูล

ไพรมารีคีย์ (Primary key) หมายถึง แอททริบิวต์หรือกลุ่มของแอททริบิวต์ที่ทำหน้าที่เป็นตัวแยกความแตกต่างของข้อมูลในแต่ละทูปเปิลในตารางฐานข้อมูล หรือกล่าวอีกนัยหนึ่งได้ว่าไพรมารีคีย์คือ แอททริบิวต์หรือกลุ่มของแอททริบิวต์ที่จะเป็นตัวที่กำหนดเพื่อให้ได้ข้อมูล 1 ทูปเปิลที่ไม่ซ้ำกันจากตารางฐานข้อมูล

รีเลชัน (Relation) หมายถึง ผลคูณคาร์ทีเซียนของโดเมนที่สนใจในแอททริบิวต์ ประกอบขึ้นด้วยองค์ประกอบ 2 ส่วนคือ เฮดดิ้ง (heading) กับ บอดี้ (body) กำหนดให้ $D_1, D_2, \dots, D_n$ แทนโดเมนทั้งหลายในระบบงาน

เฮดดิ้ง (Heading) ประกอบขึ้นมาจากเซตที่มีขนาดคงที่ของแอททริบิวต์ $A_1, A_2, \dots, A_n$ โดยที่ A_i คือ แอททริบิวต์ที่แทนข้อมูลในโดเมน D_i ($i = 1, 2, \dots, n$)

บอดี้ (Body) ประกอบขึ้นจากเซตของทูปเปิลที่มีขนาดแปรผันตามเวลารีเลชัน โดยที่แต่ละทูปเปิลประกอบด้วยค่าข้อมูลของแอททริบิวต์ต่างๆซึ่งแสดงได้ดังนี้ $(A_i: v_i)$ ($i = 1, 2, 3, \dots, n$) คือแต่ละค่าแอททริบิวต์ A_i ในเฮดดิ้งจะกำหนดค่า $(A_i: v_i)$ โดยที่ v_i เป็นค่าต่างๆที่ไม่ซ้ำกันจากโดเมน D_i ที่มีความสัมพันธ์กับแอททริบิวต์ A_i

ในหนึ่งทูปเปิลจะประกอบขึ้นด้วยค่าข้อมูลของแอททริบิวต์ที่อ้างอิงโดเมนต่างๆในตารางจากภาพที่ 1 แสดง ได้ดังนี้

(S#	:	'S1')
(SNAME	:	'SMITH')
(STATUS	:	20)
(CITY	:	'LONDON')

ตารางความสัมพันธ์ที่มีจำนวนแอททริบิวต์ n แอททริบิวต์ จะเรียกว่ามีดีกรีขนาด n (n degree) เรียกกรีเลชันที่มีแอททริบิวต์เดียวว่า ยูนารี (unary) รีเลชันที่มีสองแอททริบิวต์ว่า ไบนารี (binary) และกรีเลชันที่มีสามแอททริบิวต์ว่า เทอร์นารี (ternary) และหากมีดีกรีขนาด n ก็เรียก เอนนารี (n -ary)

2.4.1.2 คุณสมบัติของฐานข้อมูลรีเลชันแนล พิจารณาจากข้อกำหนด 4 ประการ คือ

- ต้องไม่มีทับเปิดที่ซ้ำกันภายในรีเลชัน

เนื่องจากบอดีของรีเลชันซึ่งประกอบขึ้นจากทับเปิดหลายทับเปิด มีลักษณะเป็นเซตทางคณิตศาสตร์ในรูปผลคูณคาร์ทีเซียน การซ้ำกันเองของทับเปิดจึงเหมือนกับการซ้ำกันของสมาชิกภายในเซต ซึ่งผิดนิยามทางคณิตศาสตร์ ดังนั้นจึงกำหนดคุณสมบัติข้อนี้ เพื่อให้โครงสร้างข้อมูลแบบสัมพันธ์เป็นไปตามกฎทางคณิตศาสตร์ และเนื่องจากความเป็นหนึ่ง (ไม่ซ้ำกัน) ของแต่ละทับเปิดในรีเลชันเอง ทำให้เราสามารถกล่าวได้ว่า “ทุกๆรีเลชันจะต้องมีไพรมารีคีย์เกิดขึ้นเสมอ” เพราะอย่างน้อยที่สุดกลุ่มของแอททริบิวต์ที่ค่าข้อมูลมีความเป็นหนึ่ง ก็คือเซตของแอททริบิวต์ทั้งหมดในรีเลชันเอง

- ลำดับของทับเปิด ไม่มีความสำคัญในการเก็บ

จากนิยามทางคณิตศาสตร์ที่ถือว่าสมาชิกภายในเซต จะอยู่กระจ่ายไม่มีลำดับ บอดีของรีเลชันซึ่งเป็นเซตที่มีสมาชิกคือ ทับเปิด จึงไม่ให้ความสำคัญกับลำดับของทับเปิด

- ลำดับของแอททริบิวต์ไม่มีความสำคัญในการเก็บ

ในนิยามเดียวกันกับคุณสมบัติข้อสอง ลำดับของแอททริบิวต์จึงไม่มีความสำคัญต่อการเก็บข้อมูลเช่นเดียวกัน

- ข้อมูลของแต่ละแอททริบิวต์ต้องเป็นข้อมูลเดี่ยวที่ไม่สามารถแบ่งแยกได้ หรือเรียกว่าอะตอมมิกแวลู (atomic value)

หมายความว่าเมื่อมีการกำหนดชื่อรีเลชัน ชื่อแอททริบิวต์ และทับเปิดที่ต้องการแล้ว จะต้องได้ค่าข้อมูลแอททริบิวต์ออกมาเพียงค่าหนึ่งเท่านั้น หรือสามารถกล่าวได้อีกอย่างหนึ่งว่า “รีเลชันใดๆจะต้องไม่มีกลุ่มซ้ำ (repeating group) อยู่ภายใน”

2.4.1.3 ความถูกต้องของข้อมูล (Data integrity)

ความถูกต้องของข้อมูลในรีเลชัน ขึ้นอยู่กับพฤติกรรมของข้อมูลที่มีต่อกฎความถูกต้อง (Integrity rules) ซึ่งมีอยู่ 2 กฎ ได้แก่

- กฎความถูกต้องของเอนติตี้ (entity integrity)

เอนติตี้ข้อมูลซึ่งหมายถึงค่าของข้อมูลในแอททริบิวต์ ซึ่งเป็นแอททริบิวต์ไพรมารีคีย์ของรีเลชันใดๆ ต้องไม่เป็นนัล (null) (นัล หมายถึง ไม่บันทึกค่า)

- กฎความถูกต้องในการอ้างอิง (relational integrity)

ถ้ามีความสัมพันธ์ R2 ซึ่งมีฟอร์เรนคีย์ (foreign key) FK ที่เกี่ยวเนื่องกับไพรมารีคีย์ PK ของความสัมพันธ์ R1 แล้ว ทุกค่าที่อยู่ใน FK ของความสัมพันธ์ R2 จะต้องปรากฏค่าอยู่ใน PK ของความสัมพันธ์ R1 เสมอ ยกเว้น ในกรณีที่ค่าของ FK เป็นนัล

2.4.1.4 การจัดการข้อมูล (Data manipulation)

ข้อมูลภายในฐานข้อมูลจะถูกจัดการด้วยภาษาจัดการฐานข้อมูลภายในฐานข้อมูล ภาษาจัดการฐานข้อมูลนี้ต้องเป็นไปตามนิยามของรีเลชันแนลที่สมบูรณ์ (relational complete) กล่าวคือต้องสนับสนุนกลุ่มปฏิบัติการพื้นฐานที่นิยามในภาษา พีชคณิตสัมพันธ์ (Relational Algebra) หรือ แคลคูลัสสัมพันธ์ (Relational Calculus) ซึ่งประกอบด้วยปฏิบัติการ 8 อย่าง โดยแบ่งออกเป็น 2 กลุ่มดังนี้

- กลุ่มปฏิบัติการพื้นฐาน (Traditional Set Operations) ได้แก่ ผลคูณคาร์ทีเซียน, ยูเนียน, อินเตอร์เซกชัน และดิฟเฟอเรนซ์
- กลุ่มปฏิบัติการพิเศษ (Special Relation Operations) ได้แก่ ซีเล็ค, โปรเจ็คท์, จอยน์ และ ดีไวค์

2.4.2 ลักษณะการทำงานของปฏิบัติการต่างๆ

SELECT คือ การสร้างความสัมพันธ์จากการเลือกเอาข้อมูลในทปเปิด ที่มีคุณสมบัติตามที่ระบุออกมาจากความสัมพันธ์ที่มีอยู่

PROJECT คือ การสร้างความสัมพันธ์โดยการเลือกเอาข้อมูลจากแอททริบิวต์บางตัวตามที่ได้กำหนด ออกมาจากความสัมพันธ์ที่มีอยู่

PRODUCT คือ การสร้างความสัมพันธ์ขึ้นมาใหม่จากความสัมพันธ์ 2 ความสัมพันธ์ ซึ่งความสัมพันธ์ที่สร้างขึ้น จะประกอบด้วยข้อมูลที่เป็นไปได้ทั้งหมด ที่เกิดจากการนำข้อมูลในทปเปิดของความสัมพันธ์มาจับคู่กัน

UNION คือ การสร้างความสัมพันธ์ขึ้นมาใหม่จากความสัมพันธ์ 2 ความสัมพันธ์ โดยความสัมพันธ์ที่สร้างขึ้นนี้ จะมีข้อมูลที่ประกอบไปด้วยข้อมูลของทั้ง 2 ความสัมพันธ์

INTERSECT คือ การสร้างความสัมพันธ์จากความสัมพันธ์ 2 ความสัมพันธ์ที่กำหนดให้ โดยความสัมพันธ์ที่ได้ จะประกอบด้วยข้อมูลที่อยู่ในทปเปิดที่เหมือนกันของทั้ง 2 ความสัมพันธ์ที่กำหนด

DIFFERENCE คือ การสร้างความสัมพันธ์จากความสัมพันธ์ 2 ความสัมพันธ์ที่กำหนดให้ซึ่งความสัมพันธ์ที่เกิดขึ้นใหม่จะประกอบด้วย ข้อมูลในทปเปิดทั้งหมดที่มีอยู่ในความสัมพันธ์แรกที่กำหนดให้ แต่ไม่อยู่ในความสัมพันธ์ที่สอง

JOIN คือ การสร้างความสัมพันธ์จากความสัมพันธ์ 2 ความสัมพันธ์ โดยความสัมพันธ์ที่สร้างขึ้นใหม่ จะประกอบด้วยทปเปิดที่เกิดจากการจับคู่ระหว่างทปเปิดของทั้งสองความสัมพันธ์ซึ่งการจับคู่ของทปเปิดนี้ จะต้องอยู่ภายใต้เงื่อนไขอย่างหนึ่ง

DIVIDE คือ การสร้างความสัมพันธ์จากความสัมพันธ์ 2 แบบคือ ความสัมพันธ์แบบไบนารีและความสัมพันธ์แบบยูนารี ซึ่งความสัมพันธ์ที่สร้างขึ้นจะประกอบด้วยข้อมูลทั้งหมดของแอททริบิวต์แรกของความสัมพันธ์แบบไบนารี ที่มีข้อมูลในแอททริบิวต์ที่สองเหมือนกับข้อมูลทั้งหมดในความสัมพันธ์แบบยูนารี

2.5 เอสคิวแอล (SQL)

ประโยชน์ประการหนึ่งสำหรับคอมพิวเตอร์คือ การที่จะจัดเก็บและดึงข้อมูลข่าวสารขึ้นมา กลุ่มของข้อมูลข่าวสารจะเรียกว่าระบบฐานข้อมูล และซอฟต์แวร์ที่ใช้ในงานด้านนี้ก็จะเรียกว่าระบบจัดการฐานข้อมูล (database management system : DBMS) ผู้ใช้สามารถที่จะทำการเก็บข้อมูล, แก้ไขเปลี่ยนแปลง, และเข้าถึงข้อมูล ได้อย่างมีประสิทธิภาพ

โดยปกติผู้ที่จะใช้ DBMS จะเป็นโปรแกรมเมอร์การเข้าถึงข้อมูลทำได้โดยการเขียนโปรแกรมด้วยภาษาที่ใช้ เช่น COBOL ในขณะที่โปรแกรมเหล่านี้ส่วนใหญ่จะนำเสนอในรูปแบบที่เข้าใจง่ายแก่ผู้ที่ไม่มีความรู้ทางด้านนี้ การเข้าถึงข้อมูลก็ต้องการความรู้ความเข้าใจของโปรแกรมเมอร์ที่เขียนโปรแกรมนั้นๆ ด้วย

คอมพิวเตอร์ถูกสร้างมาเพื่อให้เกิดประโยชน์แก่ผู้ใช้ ถึงแม้ว่าพวกเขาจะไม่ต้องการก็ตาม ใน การที่จะเข้าถึงข้อมูลที่สนใจ พวกเขาต้องการ โปรแกรมเมอร์ที่มีความสามารถที่จะเขียนโปรแกรมได้ ตัวอย่างเช่น ถ้าผู้จัดการฝ่ายขายต้องการที่จะทราบยอดรวมการขายของพนักงานแต่ละคนของเดือนที่แล้ว มีทางเลือกอยู่ 2 ทาง คือ โปรแกรมอนุญาตให้มีการเข้าถึงข้อมูลได้โดยตรง หรือไม่ก็ทำการจัดหา โปรแกรมเมอร์เพื่อที่จะเขียนโปรแกรมให้ ซึ่งกรณีนี้จะมีประสิทธิภาพสูงสุดแต่ก็จะเป็นวิธีที่เสียค่าใช้จ่ายในการทำงานอย่างใดอย่างหนึ่ง ซึ่งปัญหานี้ก็เริ่มจะมากขึ้น

ในการเข้าถึงระบบฐานข้อมูลจะเรียกว่า query และภาษาที่ใช้ก็จะเรียกว่า query language มี query language มากมายที่พัฒนาขึ้นมาสำหรับ DBMS หนึ่งในจำนวนนั้นพัฒนาโดยบริษัท IBM ซึ่งในปัจจุบันเริ่มที่จะเป็นที่ยอมรับมาก query language นั่นก็คือ Structured Query Language แต่ส่วนใหญ่จะเรียกเป็นตัวย่อ คือ SQL ในปัจจุบัน เอสคิวแอล เป็น query language ที่ได้รับการสนับสนุนจากหลายๆ DBMS

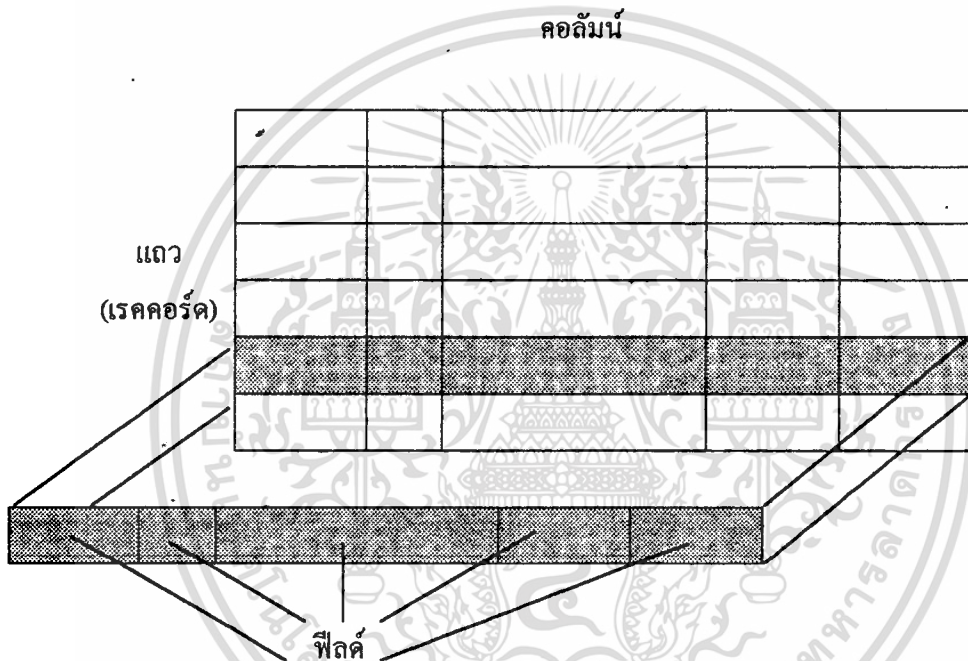
2.5.1 ทฤษฎีพื้นฐานของภาษาเอสคิวแอล

โมเดลแบบรีเลชันแนลได้ถูกคิดค้นโดย E.F. Codd นักวิจัยในห้องทดลองของ IBM วิธีการแบบรีเลชันแนลเริ่มมีความสำคัญมากขึ้น ซึ่งการเข้าถึงข้อมูลก็ง่ายแต่มีประสิทธิภาพ เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์โดย IBM การนำเอกสารนี้ไปใช้โดยไม่ได้รับอนุญาตจาก IBM ถือว่าผิดกฎหมาย การนำเอกสารนี้ไปใช้โดยไม่ได้รับอนุญาตจาก IBM ถือว่าผิดกฎหมาย

2.5.1.1 ตาราง (Table), แถว (Row), และ คอลัมน์ (Column)

ฐานข้อมูลแบบรีเลชันแนลจะประกอบไปด้วยกลุ่มของตารางข้อมูลทั้งหมดจะถูกเก็บอยู่ในตารางใดตารางหนึ่ง แต่ละตารางจะมีโครงสร้างเป็น 2 มิติ ซึ่งจะประกอบไปด้วยแถวของข้อมูลจำนวนมาก (บางครั้งอาจจะเรียกว่า เรคคอร์ด (Record)) และคอลัมน์แต่ละคอลัมน์ในตารางจะถูกกำหนดให้ไม่ซ้ำกันและมีข้อมูลชนิดต่างๆ เช่น ตัวอักษร หรือ ตัวเลข แต่ละเรคคอร์ดจะประกอบด้วยข้อมูลสำหรับแต่ละคอลัมน์ของตารางซึ่งจุดรวมระหว่างคอลัมน์และแถวจะเรียกว่าฟิลด์ (Field)

ข้อมูลที่เก็บอยู่ในแต่ละฟิลด์เป็นข้อมูลที่ใช้งานจริง โดยที่ลำดับของเรคคอร์ดในตารางนั้นไม่สำคัญ ไม่ว่าข้อมูลนั้นจะอยู่ก่อนหรือหลังเรคคอร์ดใดเรคคอร์ดหนึ่ง ซึ่งรูปแบบของตารางจะแสดงดังรูปที่ 2 - 4



รูปที่ 2 - 4 แสดง แถว , คอลัมน์ , และ ฟิลด์ ในตาราง

ระบบฐานข้อมูลที่ง่ายที่สุดจะมีเพียงตารางเดียว แต่ระบบฐานข้อมูลส่วนใหญ่จะมีตารางมากกว่านั้น ตัวอย่างเช่น ข้อมูลเกี่ยวกับที่ได้กล่าวมาแล้วนั้น จะเก็บอยู่ภายในตารางเดียว แต่จะมีอีกตารางหนึ่งเก็บข้อมูลเกี่ยวกับข้อมูลส่วนตัวของพนักงานขาย และยังมีตารางอื่นๆ ที่เก็บข้อมูลต่างๆ ที่เกี่ยวข้อง การเข้าถึงระบบฐานข้อมูลนั้น บางครั้งจะต้องกระทำกับตารางหลายๆ ตารางแต่สิ่งนี้ไม่เป็นปัญหา เพราะคำสั่งภาษาเอสคิวแอลเพียงคำสั่งเดียว สามารถที่จะทำการดึงข้อมูลจากตารางเดียวหรือหลายๆ ตารางได้ภายในระบบฐานข้อมูลเดียวกัน

2.5.1.2 คีย์ (Key)

เราจะทำการกำหนดคอลลัมน์ขึ้นมา 1 คอลลัมน์หรือมากกว่า ในแต่ละตารางซึ่งจะเรียกว่าเป็น คีย์ ของตารางข้อมูลที่อยู่ภายในคีย์ หรือฟิลด์อื่นๆ นั้นจะต้องไม่ซ้ำกันสำหรับแต่ละเรคคอร์ด ยกตัวอย่างเช่น ในระบบฐานข้อมูลที่กำลังกล่าวมาข้างต้น อาจจะกำหนดให้ชื่อของพนักงานเป็น คีย์ แต่ว่าอาจจะเป็นไปได้ที่จะมีพนักงานที่มีชื่อซ้ำกัน อย่างไรก็ตามก็อาจจะทำการเก็บข้อมูลอื่นๆ เพิ่มเข้าไป ข้อมูลที่ไม่เกิดการซ้ำกันแน่ๆ ทางเลือกที่เป็นไปได้ก็คือ หมายเลขประกันสังคมของพนักงานแต่ละคน

ในบางตารางอาจจะไม่มีข้อมูลใดที่สามารถนำมากำหนดให้เป็นคีย์ได้ ก็อาจจะสร้างคอลลัมน์ขึ้น โดยค่าภายในนั้นต้องไม่ซ้ำแน่ๆ (ส่วนใหญ่จะเป็นตัวเลข) มาเป็นคีย์ หรืออาจจะใช้วิธีการนำคอลลัมน์ 2 คอลลัมน์หรือมากกว่านั้นมารวมกันเพื่อทำให้ไม่เกิดการซ้ำกันเกิดขึ้น

การที่จะเลือกคีย์อย่างถูกต้องสำหรับแต่ละตารางนั้นไม่ใช่สิ่งง่าย อันที่จริงแล้ว การเลือกคีย์ นั้นเป็นปัญหาที่ใหญ่มากในการที่จะออกแบบฐานข้อมูลแบบรีเลชันแนล ผู้ที่ออกแบบ ซึ่งในบางครั้งอาจจะเป็น database administrator (DBA) ที่ทำหน้าที่รับผิดชอบ จะทำการวิเคราะห์ก่อนว่าโครงสร้างของระบบควรจะเป็นอย่างไร โครงสร้างที่ดีที่สุดจะขึ้นอยู่กับว่าข้อมูลจะถูกเข้าถึงอย่างไรบ้าง DBA (หรือผู้ที่ เป็นผู้ออกแบบ) จะต้องพิจารณาถึงจำนวนของคอลลัมน์ในแต่ละ table

การเข้าถึงข้อมูลผ่านภาษาเอสคิวแอลนั้น ผู้ที่ใช้ไม่จำเป็นต้องมีความรู้เป็นพิเศษสำหรับระบบ ฐานข้อมูลแบบรีเลชันแนล ซึ่งนับว่าเป็นสิ่งที่ช่วยให้่ายยิ่งขึ้น

2.6 วิซวลเอสคิวแอล (VisualSQL)

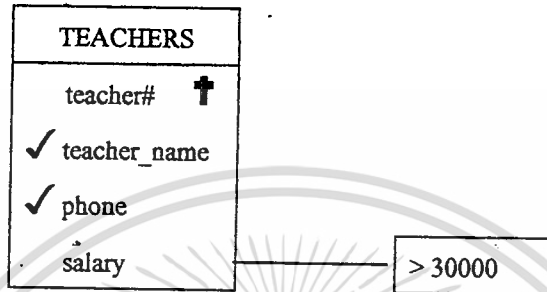
2.6.1 การนำรูปแบบวิซวล (Visual approach) มาใช้ในการเข้าถึงระบบฐานข้อมูล

ในขณะที่เอสคิวแอลเป็นภาษาที่มีประสิทธิภาพมาก มันก็เป็นภาษาที่มีความซับซ้อนในการใช้ ด้วย เพื่อที่จะขจัดความซับซ้อนนี้ จะใช้วิธีการในรูปแบบวิซวล ซึ่งถ้าเกิดความคิดในการที่จะ การเข้าถึง ระบบฐานข้อมูลในรูปแบบวิซวลก่อน เชื่อว่าจะสามารถทำให้สร้างคำสั่งภาษาเอสคิวแอลที่ถูกต้องได้ตรง ตามความต้องการสำหรับระบบฐานข้อมูลแบบรีเลชันแนลจริงๆ ได้

TEACHERS	
teacher#	†
teacher_name	
phone	
salary	

รูปที่ 2 - 5 แสดงตาราง TEACHERS

ในรูปแบบวิวทวิชนี แต่ละตารางจะในลักษณะของกล่อง ซึ่งภายในจะประกอบไปด้วยชื่อคอลัมน์ อยู่ภายใน รูปที่ 2 - 5 จะแสดงถึงตาราง TEACHERS โดยอาศัยรูปแบบวิวทวิชนี (รูปถูกแ่งที่อยู่ข้างๆกับ คอลัมน์ teacher# จะแสดงถึงคีย์ ของ TEACHERS) ตัวอย่างการที่จะ การเข้าถึงระบบฐานข้อมูลในรูปแบบวิวทวิชนี นั้น จะแสดงดังในรูปที่ 2 - 6 ซึ่งเป็นตารางที่แสดงถึงคำถาม “ จงแสดงชื่อและหมายเลข โทรศัพท์ของอาจารย์ทุกคนที่ได้เงินเดือนมากกว่า \$30,000 ต่อปี ”



รูปที่ 2 - 6 แสดงตารางการรับข้อมูลจากผู้ใ้

ส่วนในรูปแบบของภาษาเอสคิวแอล จะเป็นดังนี้

```
SELECT teacher_name, phone
FROM TEACHERS
WHERE salary > 30000;
```

ในการเข้าถึงระบบฐานข้อมูล บางครั้งอาจจะต้องใช้ข้อมูลจากตาราง 2 ตารางหรือมากกว่านั้น ซึ่งการที่จะใช้ภาษาเอสคิวแอลโดยตรงนั้นอาจจะทำได้ยาก จึงใช้การสร้างตารางที่ใช้รูปแบบวิวทวิชนี ก่อน แล้วจึงแสดงภาษาเอสคิวแอลที่ใช้ได้จริง



โปรแกรมภาษาสมอลทอล์ค (Smalltalk)

3.1 ประวัติความเป็นมาของภาษาสมอลทอล์ค

สมอลทอล์ค เกิดขึ้นในราวปี 1970 เป็นภาษาพื้นฐานของการเขียนโปรแกรมเชิงวัตถุ ซึ่ง สมอลทอล์ค นับว่าเป็น ภาษาที่ใช้หลักการเชิงวัตถุที่สมบูรณ์แบบที่สุด จนถึงกับกล่าวได้ว่าเป็น "The purest of pure Object - Oriented language"

เนื่องจากสมอลทอล์ค เป็นภาษาที่ไม่มีนิยามของคำว่าข้อมูล มีแต่อบเจกต์จะกระทำการใดๆกับอบเจกต์ในโปรแกรมได้โดยผ่านเมสเสจเท่านั้น ทุกอย่างของสมอลทอล์ค ไม่ว่าจะเป็นคำสั่งของภาษา, บล็อกของโปรแกรม หรือแม้แต่ editor ในคอมไพเลอร์ (Compiler) ก็ล้วนแต่เป็นอบเจกต์ทั้งสิ้น

เมื่อเรียกโปรแกรมสมอลทอล์คขึ้นมาทำงาน ก็จะพบกับหลักการเชิงวัตถุในทุก ๆ ส่วนของโปรแกรม ตั้งแต่การติดต่อกับผู้ใช้แบบกราฟิกที่เรียกว่า การติดต่อแบบเชิงวัตถุ , หน้าต่างต่างๆที่เป็นอบเจกต์ในระบบ , class library ซึ่งเป็นที่เก็บของคลาสต่างๆ ของระบบรวม , editor ที่ใช้ในการเขียนโปรแกรม ก็เป็นอบเจกต์หนึ่งของคลาส Editor ในระบบ หรือแม้กระทั่งคอมไพเลอร์ ที่ใช้คอมไพล์โปรแกรมภาษา สมอลทอล์ค ก็เป็นอบเจกต์หนึ่งในคลาส Compiler ของสมอลทอล์คนั่นเอง

คลาสในภาษาสมอลทอล์ค เทียบได้กับอบเจกต์ในโปรแกรม เราสามารถส่งเมสเสจไปยังคลาสได้เช่นเดียวกับที่ส่ง ไปยังอบเจกต์ ทั้งนี้เนื่องจากในความเป็นจริงแล้ว คลาสในสมอลทอล์คก็เป็นอบเจกต์ของคลาส ที่ชื่อว่า metaclass อีกต่อหนึ่ง

สมอลทอล์ค เป็นภาษาที่ถูกกำหนดให้ผูกติดกับ GUI มาตั้งแต่เกิด เพราะความเชื่อที่ว่าภาษาเชิงวัตถุในอุดมคตินั้น ต้องมีการติดต่อกับผู้ใช้ในเชิงวัตถุด้วย สมอลทอล์คจึงเป็นภาษาแรกๆ ที่ใช้ GUI เป็นหลัก และด้วยเหตุนี้เองทำให้ภาษาสมอลทอล์คสามารถแสดงลักษณะของ class library ได้อย่างชัดเจน โปรแกรมเมอร์สามารถดัดแปลง แก้ไข หรือเพิ่ม / ลด class library ได้ โดยเพียงแค่เปลี่ยนการทำงานมายังหน้าต่างของ class library เท่านั้น เมื่อโปรแกรมเมอร์ต้องการ text editor ก็สามารถดึงมาจากคลาส Editor ใน library ซึ่งจะได้อีเตอร์ที่มีความสามารถเหมือนกับที่ใช้งานอยู่

สมอลทอล์ค เป็นภาษาที่ง่ายสำหรับการเรียนรู้และการใช้ เพราะมี syntax และมี semantic ที่เข้าใจง่าย มีหลักการเพียงเล็กน้อย

3.2 หลักการพื้นฐานของสมอลทอล์ค ประกอบด้วย

3.2.1 อบเจกต์

อบเจกต์ เป็นตัวอธิบายโครงสร้างข้อมูล ทุกๆ อบเจกต์เป็นอินสแตนซ์ของคลาส คลาสของอบเจกต์ถูกกำหนดโดยการส่ง เมสเสจ "class" ซึ่ง เมสเสจ นี้เป็นที่เข้าใจของทุกๆ อบเจกต์

อบเจกต์ เป็นบล็อกพื้นฐานของภาษาสมอลทอล์คที่แสดงโครงสร้างข้อมูล ทุกอบเจกต์เป็นอินสแตนซ์ของคลาส อบเจกต์จะมีการป้องกันโครงสร้างข้อมูล โดยข้อมูลจะถูกเก็บอยู่ในอบเจกต์ซึ่งจะถูกเข้าถึงได้ โดยผ่านทางเมสเสจเท่านั้น

เช่น 336 เป็นอินสแตนซ์ของคลาส Integer
 \$C เป็นอินสแตนซ์ของคลาส Character

ออบเจกต์ เป็นโครงสร้างข้อมูลที่ถูกป้องกันโดยข้อมูลที่ถูกเก็บในออบเจกต์จะถูกเข้าถึง โดยผ่านทางเมธอดเท่านั้น

3.2.2 Variable

Variable เป็นที่เก็บของออบเจกต์โดยจะเก็บ pointer ของออบเจกต์เดียว variable หนึ่ง ๆ อาจเก็บ pointer ที่ต่างกัน ณ เวลาต่างๆ กัน

variable เป็นได้ทั้ง private และ shared โดย private variable จะ ถูก เข้าถึง ได้เพียงออบเจกต์เดียว ส่วน share variable ถูก เข้าถึง ได้โดยหลายๆ ออบเจกต์

private variable จะขึ้นต้นด้วยอักษรตัวเล็ก ส่วน share variable จะขึ้นต้นด้วยอักษรตัวใหญ่

Variable แบ่งเป็น 3 ชนิด

3.2.2.1 Instance Variable

คือ Variable ที่อ้างถึงข้อมูลทั้งหมดภายในออบเจกต์โดยจะขึ้นต้นด้วยตัวเล็ก แบ่งเป็น

- Named Instance Variable คือ Variable ที่อ้างถึงข้อมูลจริง ๆ ภายใน ออบเจกต์
- Index Instance Variable คือ Variable ที่อ้างถึงข้อมูลที่ใช้อ้างอิงตำแหน่งของ Named Instance Variable

3.2.2.2 Temporary Variable

คือ Variable ที่ใช้ชั่วคราว เมื่อใช้เสร็จจะมีการ reallocate โดยอัตโนมัติ โดย Variable จะอยู่ภายในเครื่องหมาย vertical bar

3.2.2.3 Shared Variable

คือ Variable ที่สามารถใช้ร่วมกันได้ ในหลาย ๆ ชั้นคลาส พร้อม ๆ กัน โดยจะขึ้นต้นด้วยตัวใหญ่ แบ่งเป็น

- Global variable คือ Variable ที่ใช้ร่วมกันได้ในทุก ๆ ชั้นคลาส
- Class Variable คือ Variable ที่ใช้ร่วมกันได้เฉพาะคลาสนั้น และ ชั้นคลาสของคลาสนั้น เท่านั้น

3.2.3 คลาส

คลาส เป็นตัวอธิบายโครงสร้างข้อมูล (ออบเจกต์) , อัลกอริทึม (เมธอด) และ external interface (message protocol)

ทุกๆ ออบเจกต์เป็น อินสแตนซ์ ของคลาส ออบเจกต์ที่เป็น อินสแตนซ์ ของคลาสเดียวกันจะคล้ายกันเพราะมันมีโครงสร้างเหมือนกัน (เช่น มีตัวแปรที่ใช้ภายในคลาสเหมือนกัน) , มี เมสเสจ ซึ่งมัน ตอบสนองเหมือนกันและมี เมธอด ที่ใช้เหมือนกัน

เราถือว่าคลาสเป็นออบเจกต์ซึ่งอยู่ใน global variables ดังนั้นชื่อของคลาส ขึ้นต้นด้วยอักษรตัวใหญ่เสมอ

3.2.3.1 ลำดับชั้นของคลาส

ประกอบด้วย root จะเป็นคลาสที่ชื่อ Object พร้อมกับซับคลาสอีกมากมาย แต่ละคลาสจะได้รับการถ่ายทอดคุณสมบัติและหน้าที่ทั้งหมดของซูเปอร์คลาสของมันในลำดับชั้นของคลาส สำหรับทุกๆ ออบเจกต์ แต่ละซับคลาสจะสร้างอยู่บนซูเปอร์คลาสของมัน โดยการเพิ่มเมธอด และ ตัวแปรที่ใช้ภายในคลาสของคลาสของมันเอง โดยสมอลทอลล์ได้จัดเตรียมคลาสต่างๆ ไว้อย่างมากมาย

3.2.3.2 Inheritance

คลาส หนึ่งๆ จะได้รับการถ่ายทอดคุณสมบัติทุกอย่างจากซูเปอร์คลาส ของมัน เช่น ตัวแปรที่ใช้ภายในคลาส , class variable และ method inheritance ของ variable ทำให้เมธอดของคลาส อ้างถึง class variable ที่ได้สร้างไว้ในซูเปอร์คลาสของมันได้

3.2.3.3 Class message

เมสเสจที่ใช้สำหรับคลาสใหม่ โดยทั่วไปนิยมใช้ new , new: แต่บางคลาส อาจสร้างเมสเสจ ใช้สำหรับสร้างอินสแตนซ์เองก็ได้

เช่นเดียวกันกับออบเจกต์ คลาสก็รู้ว่าเมสเสจใดบ้างที่สามารถที่จะตอบสนองได้ โดย metaclass จะเป็นตัวกำหนดว่า เมสเสจใดบ้างที่มันตอบสนองได้

3.2.3.4 การสร้างคลาส มี option ดังนี้

3.2.3.4.1 Adding Classes มี 4 ประเภท

- | | |
|------------------------|---|
| ● subclass | ข้อมูลภายในออบเจกต์สามารถมี Named Instance Variable เท่านั้น |
| ● variableSubclass | ข้อมูลภายในออบเจกต์สามารถมี Named Instance Variable และ Index Instance Variable |
| ● variableWordSubclass | คล้าย variableSubclass แต่ละ variable มีขนาดได้ 2 Byte |
| ● variableByteSubclass | คล้าย variableSubclass แต่ละ variable มีขนาดได้ 1 Byte |

การสร้าง เมธอด มี option ดังนี้

1) สร้างเมธอดขึ้นมาใหม่ มี 2 ประเภท

- class method เป็น เมธอด ที่ส่งไปยังคลาสโดยตรง
 - instance method เป็น เมธอด ที่ส่งไปยัง ตัวแปรที่ใช้ภายในคลาส ภายในคลาสนั้น
- ทั้ง 2 รูปแบบ มีการกำหนดรูปแบบไว้ โดย

messagePattern

“ comment “

| temporary variables |

statement

- 2) remove a method เมธอดนั้นจะถูกลบออกไปโดยอัตโนมัติ ไม่สามารถกู้คืนกลับมาได้
- 3) sender cause จะเป็นการแสดงรายชื่อเมธอดที่เรียกใช้เมธอดนี้
- 4) implement cause จะเป็นการแสดงรายชื่อเมธอดที่ถูกเมธอดนี้เรียกใช้

3.2.5 เมสเซจ

คือ สิ่งที่ส่งไปยังออบเจกต์ เพื่อเรียกใช้เมธอดในออบเจกต์ ให้ทำงานได้ตามต้องการ เมสเซจ คล้ายกับการ call function ในภาษาแบบโปรแกรมมิ่ง

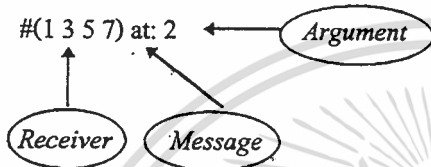
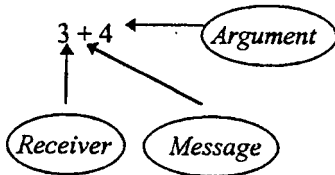
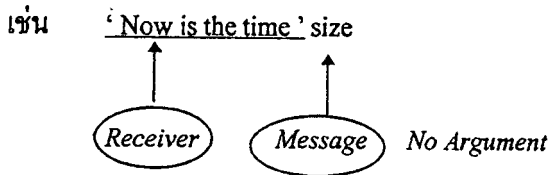
ทุกๆ ขั้นตอนในสมอลทอล์ค จะเกี่ยวข้องกับการส่งเมสเซจไปให้กับออบเจกต์ เมสเซจเป็นภาษาที่ใช้เพื่อแสดงความต้องการที่จะทำอะไรกับออบเจกต์ เมสเซจจะ request service จากออบเจกต์ โดยอยู่ในรูปของการติดต่อจากภายนอก

การส่งเมสเซจเกี่ยวข้องกับ

- 1) การกำหนดออบเจกต์ที่จะส่งเมสเซจ (เรียกรหัสไฟเวอร์ของเมสเซจ)
- 2) การกำหนดออบเจกต์เพิ่มเติมที่จะรวมอยู่ในเมสเซจ (เรียกว่า อาร์กิวเมนต์)
- 3) การระบุการทำงานที่ต้องการ (เรียกว่า ซีเลคเตอร์ (selector))
- 4) การยอมรับออบเจกต์เดียวที่จะถูก ส่งค่ากลับในฐานะที่เป็นผลลัพธ์ของเมสเซจ

เมสเซจ ประกอบด้วย

- 1) Receiver object
- 2) Message selector
- 3) Argument (จะมีหรือไม่มีก็ได้)



'20' factorial

จะเกิดความคิดพลาด เพราะ '20' ไม่รับ เมสเสจ factorial

Object always know the message that are appropriate for them

โดย เมสเสจ จะแบ่งเป็น

3.2.5.1 Unary Message คือ เมสเสจที่ไม่มีอาร์กิวเมนต์

เช่น	Message	Result
	'Hello' reversed	'olleH'
	#('a','of','string') size	3
	\$A asciiValue	65
	'Now is the time' asUppcase	'NOW IS THE TIME'

3.2.5.2 Keyword Message คือ เมสเสจที่มี 1 หรือมากกว่า 1 อาร์กิวเมนต์

เช่น	Message	Result
	'Hello' includes: \$e	True
	'Now is the string' at: 6	\$s
	#(1 0 4 5) at: 2 put: #(2 3)	(1 (2 3) 4 5)

3.2.5.3 Arithmetic Message คือ เมสเซจที่เกี่ยวข้องกับการคำนวณ

เช่น	Message	Result
	$5 * 7$	35
	$5 // 2$	2 (DIV)
	$5 \setminus 2$	1 (MOD)
	$2 / 6$	1/3 (Rational division)
	$3 + 4 * 2$	14
	$3 + (4 * 2)$	11

สมอลทอลล์ จะทำคำสั่งจากซ้ายไปขวา ถ้าไม่มีวงเล็บ แต่ถ้ามีให้ทำในวงเล็บก่อน

3.2.5.4. Binary Message คือ เมสเซจที่มี 1 อาร์กิวเมนต์ และมีอักขระพิเศษ

เช่น	Message	Result
	'Hello', 'there'	'Hellothere'
	#(1 2 3), #(4 5 6)	(1 2 3 4 5 6)

ในที่นี้อักขระพิเศษ คือ , (Comma) ใช้ทำการเชื่อมต่อระหว่างอาร์กิวเมนต์ กับ รีซีฟเวอร์

3.2.5.5 Message inside of message คือ การใช้เมสเซจที่ให้ผลลัพธ์เป็นออบเจกแทน ออบเจก

เช่น	Message	Result
	'Hello' at: #(5 3 1) at: 2)	1
	'Hello' size + 4	9

3.2.6 Object and Message Are Safe

ออบเจกมีสถานะซึ่งออบเจกสามารถจดจำสิ่งต่าง ๆ ได้ เมสเซจสามารถเปลี่ยนสถานะของออบเจกได้

เช่น จากตัวอย่าง ออบเจก Turtle สามารถจดจำตำแหน่ง heading และ สีของมันได้

- เมสเซจ black เปลี่ยนสี
- เมสเซจ go: และ home: เปลี่ยนตำแหน่ง
- เมสเซจ turn: เปลี่ยน heading

'Hello' at :1 put:23

เกิดความผิดพลาด เพราะ 23 ไม่ใช่ character เราสามารถเปลี่ยนสถานะของ string ได้โดยการส่งเมสเซจ และ string ก็สามารถใช้ได้ว่าอาร์กิวเมนต์ถูกต้องหรือเปล่า

- สิ่งนี้ทำให้ภาษาสมอลทอลล์เป็นภาษาที่มีความปลอดภัยสูงมาก

3.3 รูปแบบของคำสั่งในภาษาสมอลทอล์ค

3.3.1 การกำหนดค่า โดย

- 1) ใช้เครื่องหมาย := (Assignment Operator) กำหนดค่าให้กับตัวแปร
- 2) ใช้เครื่องหมาย . (Period) แบ่งคำสั่ง (statement) แต่ละคำสั่ง ยกเว้นคำสั่งสุดท้ายในชุดคำสั่ง ตัวอย่างเช่น

```
| temp index factorial |
factorial := #(3 4 5 6).
index := 1.
temp := factorial at: index.
index := index + 1
```

สองบรรทัดแรกกำหนดขอบเขตให้กับตัวแปรแบบชั่วคราว สองบรรทัดต่อมากำหนดผลลัพธ์ของเมสเสจ ซึ่งผลลัพธ์ของเมสเสจมักจะเป็นขอบเขตเดียวเสมอ โดยปกติมักจะกำหนดขอบเขตให้กับตัวแปรแบบชั่วคราว

3.3.2 การเรียกใช้ฟังก์ชัน

- 1) แบบมีอาร์กิวเมนต์เดียว

การใช้ฟังก์ชัน size ในภาษาปาสคาล (Pascal) จะเขียนเป็น

```
a := size(array);
```

แต่การเรียกใช้ฟังก์ชันในภาษาสมอลทอล์คคือการส่งเมสเสจ ตัวอย่างเช่น

```
a := array size
```

คือการส่งเมสเสจ size ไปยังค่าของตัวแปร array ซึ่งเป็นอาร์กิวเมนต์

- 2) แบบสองอาร์กิวเมนต์

ในสมอลทอล์ค อาร์กิวเมนต์จะนำหน้าและตามหลังเมสเสจ ตัวอย่างเช่น

Smalltalk

Pascal

```
x := x1 max: x2.
```

```
x := max(x1,x2);
```

```
y := p + q.
```

```
y := p + q;
```

เมสเสจ max: ถูกส่งไปยังค่าของ x1 (อาร์กิวเมนต์แรก) กับค่าของ x2 (อาร์กิวเมนต์ที่สอง) แล้วกำหนดผลลัพธ์ที่ได้กับ x

- 3) แบบอาร์กิวเมนต์ตั้งแต่สามขึ้นไป

ตัวอย่างเช่น

```
a := x between: x1 and: x2
```

เปรียบเทียบกับปาสคาลจะเขียนเป็น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$a := \text{between}(x, x1, x2);$

ในสมอลทอล์คชื่อของเมสเซจจะแตกออกเป็นส่วนๆ และชื่อของเมสเซจแต่ละส่วนจะนำหน้าแต่ละอาร์กิวเมนต์ที่ต่อมาจากอาร์กิวเมนต์แรก ในตัวอย่างชื่อเมสเซจคือ `between:and:` และอาร์กิวเมนต์คือ x , $x1$ และ $x2$

3.3.2.1 การใช้ซับสคริปต์ (Subscript) ในสมอลทอล์คใช้เมสเซจ `at:` และ `at:put:` ตัวอย่างเช่น

Smalltalk	Pascal
$x := a \text{ at: } i.$	$x := a[i];$
$a \text{ at: } i \text{ put: } y.$	$a[i] := y;$
$a \text{ at: } i+1 \text{ put: } (a \text{ at: } i)$	$a[i+1] := a[i];$

จากตัวอย่างในคำสั่งบรรทัดแรกค่าของ i เป็น index ของอาร์เรย์เพื่อกำหนดค่าของ a และส่งผลลัพธ์ไปเก็บที่ x คำสั่งในบรรทัดที่สองคือแทนค่าของ a ที่ตำแหน่ง i ด้วยค่าของ y คำสั่งในบรรทัดที่สามเป็นการแทนค่าในทำนองเดียวกัน เครื่องหมายวงเล็บในสมอลทอล์คเป็นการกำหนดของการกระทำ และในสมอลทอล์กด้านซ้ายมือของเครื่องหมาย $:=$ ต้องเป็นตัวแปรแบบสเกลาร์ (Scalar) เท่านั้น

3.3.2.2 การเปรียบเทียบออบเจกต์

ในสมอลทอล์คเปรียบเทียบออบเจกต์โดยการส่งเมสเซจ `<`, `<=`, `=`, `>`, `>=` และ `==` ซึ่งพัฒนาเป็นไบนารีเมสเซจ ดังนั้นเครื่องหมายวงเล็บมักจะต้องใช้ถ้ามีไบนารีเมสเซจอื่น เช่น

```
3 < 4
#(1 2 3 4) = #(1 2 3 4)
5 = (2 + 3)
```

การเปรียบเทียบมักจะส่งค่ากลับเป็นจริงหรือเท็จ

3.3.2.3 การทดสอบออบเจกต์

ออบเจกต์หลายออบเจกต์เข้าใจเมสเซจซึ่งในบางครั้งทดสอบสเตทของออบเจกต์หรือเงื่อนไข เช่น

```
$a isUpperCase
('Hello' at: 1) isVowel
```

3.3.3 คำสั่งเงื่อนไข

ภาษาส่วนมากใช้คำสั่ง `if` สำหรับประโยคเงื่อนไข สมอลทอล์คใช้บล็อกของโค้ด (Code) และเมสเซจสำหรับประโยคเงื่อนไข เช่น

```
| max a b |
```

```
a := 5 squared.
```

```
b := 4 factorial.
```

```
a < b
```

```
ifTrue: [max := b]
```

```
ifFalse: [max := a].
```

```
^max
```

เมสเซนเจอร์เปรียบเทียบ $a < b$ ส่งค่ากลับเป็นจริงหรือเท็จซึ่งเป็นรีซีฟเวอร์ของเมสเซนเจอร์ ifTrue:ifFalse: การเปรียบเทียบมักจะไม่ใช่วิธีแบบเดี่ยวแต่มักจะเป็นแบบผสม (Compound) โดยใช้เมสเซนเจอร์ and: และ or: ตัวอย่างเช่น ($c < \$0$ or: [$c > \$9$])

รีซีฟเวอร์ของเมสเซนเจอร์ or: คือผลลัพธ์ของการเปรียบเทียบแรกซึ่งเป็นจริงหรือเท็จ ส่วนอาร์กิวเมนต์คือบล็อกของโค้ดซึ่งเป็นการเปรียบเทียบสุดท้ายมีผลลัพธ์เป็นจริงหรือเท็จเช่นกัน

3.3.4 คำสั่งทำซ้ำ (Iterative Statement)

คำสั่งวนรูปแบบธรรมดาของสโมลทอล์คใช้เมสเซนเจอร์ timesRepeat: กำหนดจำนวนครั้งในการทำซ้ำ เช่น

```
| string index c |
```

```
string := 'Now is the time'.
```

```
index := 1.
```

```
String size timesRepeat: [
```

```
    c := string at: index.
```

```
    string at: index put:
```

```
        (c isVowel
```

```
            ifTrue: [c UpperCase]
```

```
            ifFalse: [c asLowerCase]).
```

```
    index := index + 1.
```

```
^string
```

เมสเซนเจอร์วนรูปแบบอื่น เช่น

```
“copy a disk file”
```

```
| input output |
```

```
input := File pathName: 'go'.
```

```
output := File pathName: 'junk'.
```

```
[input atEnd]
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

whileFalse: [output nextPut: input next].

input close.

output close

ในภาษาสมอลทอล์คข้อความในเครื่องหมายคำพูดคือคำอธิบาย (Comment) ภายในบล็อก input atEnd จะส่งค่ากลับเป็นจริงหรือเท็จ โดยไฟล์ input จะถูกอ่านค่าถัดไปด้วยเมสเสจ next ไฟล์ output จะเขียนค่าอาร์กิวเมนต์ (คือค่าที่ได้จากการอ่านค่าในไฟล์ input) ด้วยเมสเสจ nextPut:

ตัวอย่างของ whileTrue: เช่น

...

[i<10]

whileTrue: [sum := sum + (a at: i).

i := i + 1]

เมสเสจทำซ้ำ to:do: มีอาร์กิวเมนต์ 2 อาร์กิวเมนต์ คือ side และจำนวนรอบ

“draw several polygons”

3 to: 6 do: [:side| side timesRepeat: [Turtle go: 6; turn: 360//side]]

จากตัวอย่างข้างบน 3 คือวิธีห้เวอร์รอบแรก 6 คืออาร์กิวเมนต์ตัวแรกซึ่งเป็นค่าสูงสุดของการทำซ้ำ และอาร์กิวเมนต์ตัวที่สอง คือบล็อกของโค้ดซึ่งมี side เป็นบล็อกอาร์กิวเมนต์ (จะได้กล่าวถึงรายละเอียดในหัวข้อถัดไป) จะเพิ่มค่าที่ละ 1 ในแต่ละรอบของการทำซ้ำ แต่ถ้าต้องการจะกำหนดค่าที่จะเพิ่มขึ้นในแต่ละรอบของการทำซ้ำเป็นค่าอื่น สามารถใช้เมสเสจ to:by:do: ได้ ตัวอย่างเช่น

| sum |

sum := 0.

1/2 to: 1 by: 1/8 do: [:i| sum := sum + i].

^sum

3.4 บล็อกอาร์กิวเมนต์

บล็อกอาร์กิวเมนต์คือตัวแปรชั่วคราวแบบหนึ่ง แต่ไม่ต้องกำหนดไว้ที่บรรทัดแรกของชุดคำสั่ง บล็อกอาร์กิวเมนต์เป็นตัวแปรที่ใช้ภายในบล็อกโดยกำหนดเป็นส่วนแรกของบล็อก ซึ่งหน้าตัวแปรมีเครื่องหมาย : (colon) และตัวแปรเขียนแยกกับคำสั่งภายในบล็อกด้วยเครื่องหมาย | (vertical bar)

บล็อกและอาร์กิวเมนต์ทำให้สมอลทอล์คมีเมสเสจการทำซ้ำหลายเมสเสจ เช่นเมสเสจ do:, select:, reject: และ collect:

3.4.1 เมสเสจ do:

ตัวอย่างเช่น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

| vowel |

vowel := 0.

'Now is the time' do: [:char| char is Vowel

ifTrue: [vowel := vowel + 1]].

^vowel

เมสเซนเจอร์การทำซ้ำ do: มีผลทำให้สตริงส่งตัวอักษรทีละตัวเข้าไปยังบล็อก

3.4.2 เมสเซนเจอร์ select:

ตัวอย่างเช่น

('Now is the time' select: [:c| c is Vowel]) size

เมสเซนเจอร์การทำซ้ำ select: มีผลทำให้รีซีฟเวอร์ส่งค่ากลับเป็นสมาชิกทั้งหมดเข้าไปยังบล็อก ผลลัพธ์คืออักขรสระทั้งหมด และเมสเซนเจอร์ size จะให้ค่าจำนวนสมาชิกทั้งหมดที่ถูกเลือก

3.4.3 เมสเซนเจอร์ reject:

ตัวอย่างเช่น

#(1 2 3 4 5 6 7 8 9) reject: [:i| i factorial >= (i * i * i * i)]

การทำงานของเมสเซนเจอร์ reject: จะทำนองเดียวกับ select: แต่จะส่งค่ากลับเฉพาะสมาชิกทั้งหมดของรีซีฟเวอร์ที่ได้ผลลัพธ์จากบล็อกเป็นเท็จ ในขณะที่เมสเซนเจอร์ select: ผลลัพธ์จากบล็อกเป็นจริง

3.4.4 เมสเซนเจอร์ collect:

ตัวอย่างเช่น

#(1 13 7 10) collect: [:i| i * i]

การทำงานของเมสเซนเจอร์ collect: มีผลทำให้บล็อกของโค้ดทำงานสำหรับแต่ละสมาชิกของรีซีฟเวอร์ และให้ผลลัพธ์เป็นกลุ่มของผลลัพธ์ทั้งหมดที่ส่งกลับโดยบล็อก

3.5 การส่งค่ากลับของฟังก์ชัน

ในสมอลทอล์คใช้เครื่องหมาย ^ (caret) นำหน้านิพจน์ซึ่งเป็นผลลัพธ์ของเมธอด เมธอดจะหยุดการทำงานและส่งค่าผลลัพธ์ เช่น

^answer

เปรียบเทียบกับภาษาปาสคาลผลลัพธ์ของฟังก์ชันจะเขียนเป็น

functionName := answer;

return

สว่านบนข้อความที่เลือก แต่ถ้าใช้คีย์บอร์ดให้กดปุ่ม '+' บนคีย์บอร์ดด้านตัวเลขที่จุดเริ่มต้นของข้อความ และกดปุ่ม '-' บนคีย์บอร์ดด้านตัวเลขที่ท้ายสุดของข้อความ

3.8 การบำรุงรักษาระบบ

3.8.1 การบำรุงรักษา

สิ่งที่จะต้องทำในเรื่องของการบำรุงรักษา คือ

- 1) จะเก็บรักษาไฟล์ของระบบอย่างไรถ้าขนาดของไฟล์ใหญ่ขึ้น
- 2) ขั้นตอนปกติที่เราสามารถป้องกันไม่ให้งานสูญหายไป ถ้าหากระบบเกิดมีปัญหากำหนดไม่ได้
- 3) จะทำอย่างไรถ้าระบบเกิดปัญหากำหนดไม่ได้

3.8.2 การบันทึกการเปลี่ยนแปลงแบบอัตโนมัติ

สมอลทอลล์จะบันทึกการเปลี่ยนแปลงไว้ในไฟล์เชนจ์ล็อก (Change Log) ส่วนโคดที่ไม่มีการเปลี่ยนแปลงจะเก็บไว้ในไฟล์ซอร์ส (Source) และเก็บพอยน์เตอร์ของไฟล์เชนจ์ล็อก ไฟล์ซอร์สไว้ในไฟล์อิมเมจ (Image) ดังนั้นเราจะต้องเก็บรักษาไฟล์ทั้งสามไว้ด้วยกัน จากเหตุผลดังกล่าวเราจึงไม่ควรที่จะปรับปรุงแก้ไขไฟล์เชนจ์ล็อก เราสามารถเรียกมาดูได้และสามารถที่จะนำเมธอดและการกำหนดคลาสสลับเข้ามาในระบบอีกในกรณีที่มีการกู้ข้อมูล (Recovery) อันเนื่องมาจากระบบเกิดปัญหากำหนดไม่ได้ เหตุการณ์ที่สำคัญอื่นๆที่ทำให้สมอลทอลล์มีการจดบันทึกแบบอัตโนมัติ เช่น ทุกครั้งที่สั่งให้มีการทำงาน (ใช้คำสั่ง do it หรือ show it) จะมีการจดบันทึกรวมทั้งเมื่อมีการสั่งให้ลบเมธอดออกจากคลาสก็มีการจดบันทึกเช่นกัน

3.8.3 ความสำคัญของการเก็บอิมเมจ

ไฟล์อิมเมจ คือ กลุ่มของออบเจกต์ทุกออบเจกต์ในสมอลทอลล์ซึ่งถูกสร้างขึ้นมา เมื่อเราสั่งสำเนาข้อมูลในไฟล์อิมเมจ คำอธิบายของออบเจกต์ทั้งหมดจะบันทึกลงในไฟล์อิมเมจ เมื่อเราเริ่มต้นระบบงานใหม่ในภายหลังทุกๆออบเจกต์ในไฟล์อิมเมจจะถูกสร้างขึ้นม่อีกครั้งหนึ่ง เฉพาะตรงจุดที่ไฟล์อิมเมจถูกสั่งให้ทำสำเนาไว้ก่อนที่จะออกจากระบบงานครั้งล่าสุด ดังนั้นระบบจะเริ่มต้นใหม่เหมือนกับที่เราได้เคยใช้งานระบบก่อนหน้านั้น เมื่อเราเปลี่ยนแปลงโคด (Code) หรือสร้างออบเจกต์ใหม่และเปลี่ยนแปลงรายละเอียดของออบเจกต์เดิมเราจะต้องสั่งทำสำเนาไฟล์อิมเมจเพื่อให้การเปลี่ยนแปลงนั้นถาวร เราสามารถสั่งสำเนาไฟล์อิมเมจด้วยการเลือกใช้ฟังก์ชันเซฟอิมเมจในเมนูระบบ หรือเลือกฟังก์ชันเซฟอิมเมจเมื่อตอนที่ออกจากโปรแกรมสมอลทอลล์ ถ้าเราไม่ได้สั่งให้ทำสำเนาไฟล์อิมเมจไว้เราจะได้การเปลี่ยนแปลงต่างๆ เหล่านั้นเราเริ่มต้นระบบใหม่ในคราวหน้า

3.8.4 การลดขนาดไฟล์เชนจ์ล็อก (Compressing Change Log)

การลดขนาดของไฟล์เชนจ์ล็อกนั้นจะมีการสร้างสำเนาของเมธอดล่าสุดของแต่ละเมธอดที่เป็นเมธอดใหม่หรือมีการเปลี่ยนแปลง ไฟล์อิมเมจใหม่จะถูกเขียนขึ้นโดยอัตโนมัติ เราควรลดขนาดของเอกสารเป็นเอกสารที่ส่งงานไว้ส่วหนึ่งเพื่อป้องกันการเกิดปัญหา เมื่อคุณได้เห็นการเปลี่ยนแปลงเอกสารแล้ว ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้คัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ไฟล์เซนต์ล็อกเมื่อมันเริ่มใหญ่จะขึ้น ซึ่งในขณะที่จะทำการลดขนาดของไฟล์เซนต์ล็อกนี้เนื้อที่ในดิสก์จะต้องมีพอสำหรับไฟล์เซนต์ล็อกเก่าและใหม่ในเวลาเดียวกันด้วย ดังนั้นเราต้องมั่นใจว่าเราได้ลดขนาดของไฟล์เซนต์ล็อกก่อนที่ดิสก์จะเต็ม คำสั่งที่ใช้คือ `Smalltalk compressChanges`

3.8.5 การลดขนาดของไฟล์ซอร์ส (Compressing source)

การลดขนาดไฟล์ซอร์สจะมีการสร้างซอร์สไฟล์ใหม่สำหรับทุกๆ เมธอดในระบบในขณะที่นั้น โดยที่จะคัดลอกสำเนาล่าสุดของซอร์สโคดสำหรับแต่ละเมธอดจากไฟล์ซอร์สเก่าหรือจากไฟล์เซนต์ล็อก หลังจากการทำให้ขนาดของไฟล์ซอร์สลดลงแล้วจะได้ไฟล์อิมเมจและไฟล์เซนต์ล็อกใหม่เกิดขึ้นโดยอัตโนมัติด้วย ไฟล์เซนต์ล็อกใหม่ที่ได้จะเป็นไฟล์ที่ว่างเปล่า ไฟล์ซอร์สจะถูกเขียนในรูปแบบบีบเพื่อประหยัดเนื้อที่ในดิสก์ ดังนั้นเราไม่สามารถแก้ไขไฟล์ซอร์สได้นอกจากโปรแกรมสมอลทอล์คเอง แต่เราสามารถทำสำเนาไฟล์ซอร์สได้ หลังจากที่เราบีบซอร์สเราจะได้ไฟล์อิมเมจใหม่ ไฟล์ซอร์สใหม่ และไฟล์เซนต์ล็อกใหม่ แต่ไฟล์เซนต์ล็อกจะว่างเปล่า ดังนั้นเราจึงควรที่จะทำสำเนาของไฟล์ทั้งสามไว้ก่อนที่จะทำการลดขนาดของไฟล์ซอร์สเพื่อว่าอาจจะมีข้อผิดพลาดอะไรเกิดขึ้นจะได้สามารถกู้ข้อมูลคืนได้ คำสั่งที่ใช้คือ `Smalltalk compressSources`

3.8.6 การกู้ระบบ (Surviving a System Crash)

สมอลทอล์คมีความยืดหยุ่นกับข้อผิดพลาดมาก ซึ่งอาจจะโชคไม่ดีนักที่ความเสียหายสามารถที่จะเกิดขึ้นได้โดยเฉพาะถ้าหากว่าเรามีการปรับปรุงแก้ไขในสมอลทอล์ค แต่ว่าคุณสมบัติการบันทึกสำเนาแบบอัตโนมัติของสมอลทอล์คทำให้การกู้ข้อมูลสามารถทำได้ ความเสียหายไม่ได้เกิดขึ้นเพราะว่าในโปรแกรมสมอลทอล์คมีบั๊ก (bug) แต่เพราะว่ามีการเปลี่ยนแปลงสถานะภายในของโปรแกรมสมอลทอล์คอันเนื่องมาจากโปรแกรมสมอลทอล์คยอมให้เราเปลี่ยนแปลงอะไรก็ได้ ทุกอย่างแม้แต่การทำลายสถานะของระบบ แต่สิ่งเหล่านี้ก็ไม่ได้เป็นปัญหาอะไร เพราะว่าเราสามารถกู้ข้อมูลกลับคืนมาได้เกือบทั้งหมด ในความเป็นจริงการทดลองทำกับระบบคือวิธีการเรียนรู้ที่ดีที่สุด แต่ว่าเราควรจะมีใจว่าเราได้ทำสิ่งต่อไปนี้ เป็นการป้องกันไว้ก่อนล่วงหน้าแล้ว คือ

- 1) ต้องทำสำเนาของไฟล์ซอร์สไว้เสมอ
- 2) ต้องทำสำเนาที่เชื่อถือได้ของไฟล์อิมเมจและเซนต์ล็อก จะต้องจำไว้ว่าไฟล์อิมเมจและไฟล์เซนต์ล็อกใช้ในการทำงานร่วมกัน ซึ่งถ้าหากว่าเราใช้ไฟล์อิมเมจที่เป็นเวอร์ชันเก่ากว่าเวอร์ชันของไฟล์เซนต์ล็อก เราก็จะไม่สามารถใช้ซอร์สโคดสำหรับบางเมธอดได้ เพราะว่าหลังจากที่เราลดขนาดของไฟล์เซนต์ล็อกแล้วจะทำให้ไฟล์อิมเมจและไฟล์เซนต์ล็อกถูกเปลี่ยนไปด้วย เราจึงไม่สามารถใช้ไฟล์อิมเมจเก่ากับไฟล์เซนต์ล็อกเวอร์ชันล่าสุดได้ ถ้าหากว่าไฟล์เซนต์ล็อกนั้นยังไม่ได้ถูกลดขนาดลงในช่วงระยะเวลาที่ไฟล์อิมเมจกำลังใช้ไฟล์เซนต์ล็อก สรุปก็คือจะต้องทำสำเนาไฟล์อิมเมจและไฟล์เซนต์ล็อกร่วมกันแล้วเราจะไม่มีปัญหา
- 3) เราควรจะทำสำเนาไฟล์อิมเมจเก็บไว้ในดิสก์ก่อนที่จะพยายามทำบางสิ่งบางอย่างที่อาจจะทำ

ให้ระบบเกิดปัญหาจนไม่สามารถทำงานได้ ซึ่งเราสามารถที่จะใช้คำสั่งทำสำเนาไฟล์อิมเมจได้จากเมนูระบบ การทำเช่นนี้จะช่วยให้เราสามารถรักษางานทั้งหมดของเราไว้ได้ถ้าหากว่าระบบเกิดปัญหาจนทำงานไม่ได้ขึ้นมา เราจะสามารถเริ่มต้นใหม่อีกครั้งได้โดยการใช้สำเนาไฟล์อิมเมจนี้ แต่ถ้าหากว่าเราทำสำเนาไฟล์อิมเมจยังไม่สำเร็จและระบบเกิดมีปัญหามาไม่ทำงานขึ้นมา เราก็ยังสามารถที่จะใช้ไฟล์เซนต์ล๊อคช่วยได้

- 4) ถ้าเราทดลองกับ โปรแกรมสมอลทอลล์และเราเกิดไปทำระบบเสียหรือว่าเราไม่ต้องการให้ การเปลี่ยนแปลงนั้นถาวรเราก็ไม่ต้องทำไฟล์สำเนาอิมเมจ โดยการที่เราออกจากโปรแกรมสมอลทอลล์ด้วยการใช้ฟังก์ชันฟอร์เกทอิมเมจ (forget image) แต่ว่าการเปลี่ยนแปลงทั้งหมดเหล่านั้นถูกบันทึกไว้ในไฟล์เซนต์ล๊อคแล้วเราก็สามารถที่จะเรียกกลับเข้ามาในระบบอีกครั้งได้
- 5) ถ้าระบบเกิดมีปัญหามาไม่ทำงาน อย่าตกใจ ให้ทำสำเนาของไฟล์อิมเมจและไฟล์เซนต์ล๊อค โดยปกติเราสามารถที่จะกู้ข้อมูลกลับคืนมาได้ เพราะถ้าเราทำสำเนาเอาไว้เราก็สามารถที่จะค่อยๆอ่านไปที่ละส่วนเพื่อหว่านสารที่ต้องการจะกู้ข้อมูลคืนมา

3.8.7 การกู้ข้อมูล (Recovering From a System Crash)

ถ้าระบบ ไม่ทำงานเราจะทำสำเนาของไฟล์อิมเมจและไฟล์เซนต์ล๊อคไว้ก่อนแล้ว การกู้ข้อมูลมีลำดับขั้นตอนดังนี้

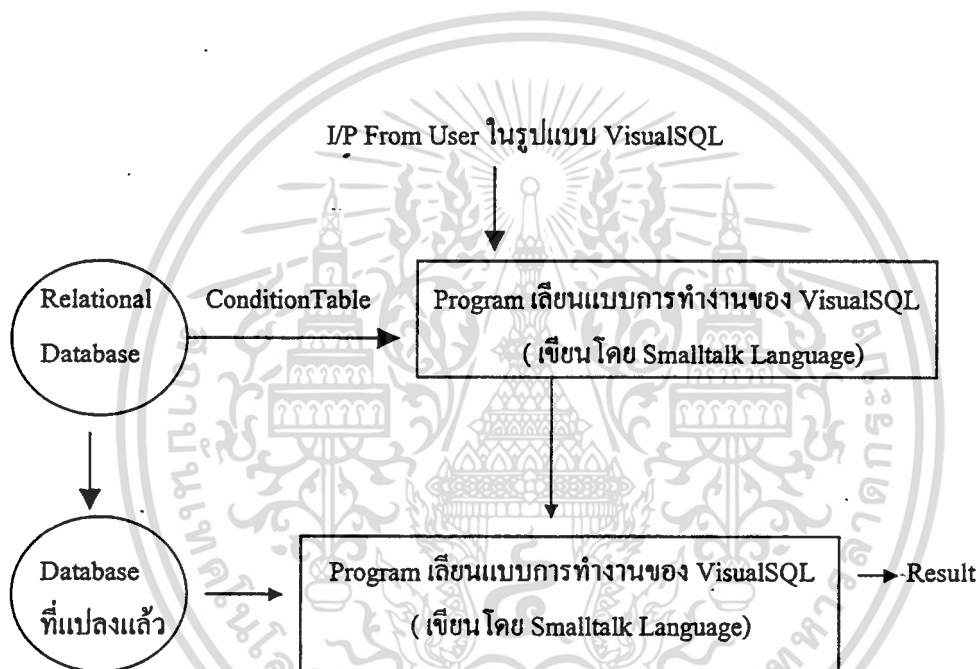
- 1) ตรวจสอบถ้าหากว่าเรามีไฟล์อิมเมจที่ดีแล้ว พยายามที่จะเริ่มต้น โปรแกรมสมอลทอลล์แบบปกติโดยการใช้ไฟล์อิมเมจนี้ แต่ถ้ามันเริ่มต้นไม่ได้เราควรจะทำสำเนาของไฟล์อิมเมจและไฟล์เซนต์ล๊อคที่ใหม่ที่สุดแล้วเริ่มต้น โปรแกรมสมอลทอลล์ด้วยเวอร์ชันนั้น
- 2) ในขณะที่โปรแกรมสมอลทอลล์กำลังทำงาน ให้มองหาในไฟล์เซนต์ล๊อคที่กำลังใช้ในขณะที่นั้นเมื่อระบบเกิดไม่ทำงาน ในไฟล์เซนต์ล๊อคจะมีการเปลี่ยนแปลงทั้งหมดที่เราทำและการทำงานทั้งหมดที่เราสั่ง
- 3) มองหาจุดที่เราทำสำเนาไฟล์อิมเมจล่าสุดในไฟล์เซนต์ล๊อคเพราะว่าทุกครั้งที่เราทำสำเนาไฟล์อิมเมจ โปรแกรมสมอลทอลล์จะเขียนจะเขียนข้อความบอกวันเวลาที่ทำในไฟล์เซนต์ล๊อค ไล่หาในไฟล์เซนต์ล๊อคจากข้างท้ายขึ้นมาถ้าต้องการสำเนาอิมเมจล่าสุด
- 4) เมื่อได้จุดที่ทำสำเนาอิมเมจซึ่งงานของเราได้หายไปจนจบไฟล์เซนต์ล๊อค ใช้ฟังก์ชันอินสทอล (install) จากเมนูเพน (pane menu) เราต้องระวังที่จะไม่นำเอาส่วนที่มีข้อผิดพลาดกลับเข้ามาในระบบอีก

บทที่ 4

การออกแบบสร้างโปรแกรม

4.1 แนวคิดของโปรแกรม

ผู้ใช้งาน (User) จะเป็นผู้ป้อนข้อมูลที่ต้องการเข้าไปในโปรแกรมที่นำหลักการทำงานมาจากโปรแกรมวิซวลเอสคิวแอล จากนั้นโปรแกรมจะทำการสร้างข้อความ (text) ในรูปแบบของภาษาที่นำหลักการทำงานมาจากโปรแกรมภาษาเอสคิวแอล ต่อจากนั้นจะทำการค้นหาจากฐานข้อมูลแบบรีเลชันแนล ที่ถูกออกแบบมาให้เป็นออบเจก และแสดงผลตามที่ผู้ใช้งานต้องการ ตามรูปที่ 4-1



รูปที่ 4 - 1 แสดงหลักการทำงานของโปรแกรมที่นำหลักการทำงานมาจากโปรแกรมวิซวลเอสคิวแอล

4.2 การกำหนดคลาสและตัวแปรที่ใช้ภายในคลาส

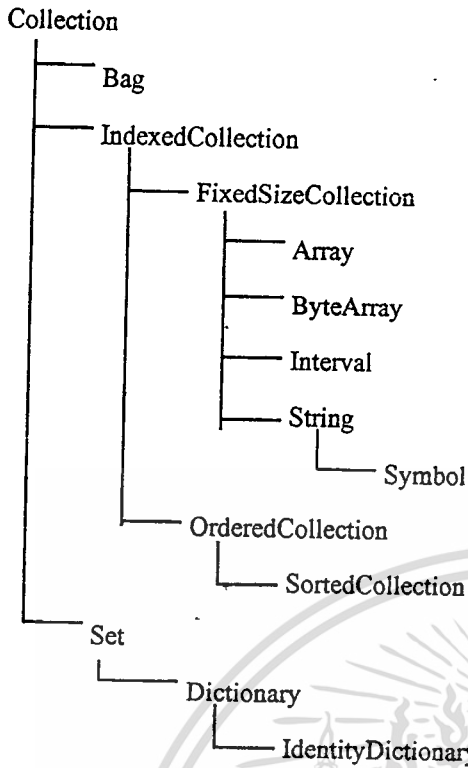
4.2.1 ศึกษาคลาสต่างๆ ที่ใช้ในการเก็บข้อมูลในภาษาสมอลทอล์ก โดยการเก็บข้อมูลในรูปแบบต่างๆ นั้นจะอยู่ในคลาส Collection ซึ่งภายในจะมีคลาสอื่นๆ อยู่ เช่น Array , String ดังรูปที่ 4 - 2

4.2.1.1 คลาส Dictionaries

Dictionaries จะทำการเก็บและนำออบเจกกลับมาโดยใช้คีย์ (key) ตัวอย่างเช่น สมุดโทรศัพท์ จะต้องสร้าง global variable ชื่อ PhoneBook โดยใช้ เมสเสจ

PhoneBook := Dictionary new.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4 - 2 แสดง hierarchy ของคลาส Collection

การเพิ่มหมายเลขโทรศัพท์ที่สามารถทำได้โดยใช้เมสเสจ “at.put:”

PhoneBook

at: 'Marisa' put: '645-1082';
 at: 'Francesca' put: '555-1222';
 at: 'Jackie' put: '392-481-5000';
 at: 'Rakesh' put: '645-1083';
 at: 'Vijay' put: '645-1083'

โดยที่ string 'Marisa' และ 'Francesca' จะเป็นคีย์ และ '645-1082' และ '555-1222' จะเป็น value ที่สอดคล้องกับคีย์ข้างต้น ตามลำดับ

การนำออกเจตออกมาจาก dictionary จะใช้ เมสเสจ “at:” ตามหลังด้วยคีย์เป็นอาร์กิวเมนต์ เช่น

PhoneBook at: 'Marisa'

จะได้ผลลัพธ์เป็น string '645-1082'

4.2.1.2 คลาส Sets

Sets จะมีลักษณะคล้าย ๆ กับเซตในทางคณิตศาสตร์ คือ จะเก็บกลุ่มของออบเจกต์ซึ่งแต่ละออบเจกต์จะปรากฏเพียงครั้งเดียว การเก็บ element เข้าไปใน set สามารถทำได้โดยใช้ เมสเสจ “add:”

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวอย่างเช่น

```
| input words word |
input := File pathName: 'in.fil'.
words := Set new.
[( word := input nextWord) isNil ]
    whileFalse: [ words add: word asLowerCase ].
^words asSortedCollection
```

จะเป็นการเรียงลำดับคำทั้งหมดที่อยู่ในแฟ้มข้อมูล

4.2.1.3 คลาส Bags

Bags จะมีลักษณะคล้ายกับ Set แต่ว่า แต่ละออบเจกต์จะสามารถปรากฏได้หลายครั้ง การเก็บ element เข้าไปใน Bag ทำได้โดยใช้ เมสเสจ “add:” และยังสามารถนับจำนวนครั้งที่ปรากฏ ของแต่ละ element ได้ โดยใช้เมสเสจ “occurrencesOf:” ตัวอย่างเช่น

```
| input frequency output word |
input := File pathName: 'in.fil'.
output := File pathName: 'out.fil'.
frequency := Bag new.
[( word := input nextWord) isNil ]
    whileFalse: [ frequency add: word asLowerCase ].
frequency asSet asSortedCollection do: [ :word |
    output
        nextPutAll: word;
        tab;
        nextPutAll: ( frequency occurrencesOf: word ) printString;
        cr ].
output close.
```

จะเป็นการแสดงจำนวนครั้งที่ปรากฏของแต่ละคำภายในแฟ้มข้อมูล

4.2.1.4 คลาส OrderedCollections

OrderedCollections จะเก็บออบเจกต์เป็นลำดับ ซึ่งจะมีลักษณะคล้ายกันกับเป็น dynamic array คือจะมีขนาดของ array เปลี่ยนแปลงได้โดยไม่ต้องกำหนดขนาดตายตัว

เมสเสจ “add:” จะถ่ายทอศ คุณสมบัติมาจาก คลาส Collection ซึ่งจะทำให้การเก็บ element เข้าไปตามลำดับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.2 ทำการเลือกคลาสที่เหมาะสม โดยพิจารณาการนำไปใช้งานในโปรแกรมที่นำหลักการทำงานมาจากโปรแกรมวิซวลเบสิกและภาษาที่นำหลักการทำงานมาจากภาษาเบสิกแอตที่ได้ทำการออกแบบไว้

4.2.2.1 คลาส RDB

PRES_NAME	BIRTH_YR	YRS_SERV	DEATH_AGE	PARTY	STATE_BORN
Washington G	1732	7	67	Federalist	Virginia
Adams J	1735	4	90	Federalist	Massachusetts
Jefferson T	1743	8	83	Demo-Rep	Virginia
Madison J	1751	8	85	Demo-Rep	Virginia
Monroe J	1758	8	73	Demo-Rep	Virginia
Adams J Q	1767	4	80	Demo-Rep	Massachusetts
Jackson A	1767	8	78	Democratic	South Carolina
Van Buren M	1782	4	79	Democratic	New York
Harrison W H	1773	0	68	Whig	Virginia
Tyler J	1790	3	71	Whig	Virginia
Polk J K	1795	4	53	Democratic	North Carolina
Taylor Z	1784	1	65	Whig	Virginia
Fillmore M	1800	2	74	Whig	New York
Pierce F	1804	4	64	Democratic	New Hampshire
Buchanan J	1791	4	77	Democratic	Pennsylvania
Lincoln A	1809	4	56	Republican	Kentucky
Johnson A	1808	3	66	Democratic	North Carolina
Grant U S	1822	8	63	Republican	Ohio
Hayes R B	1822	4	70	Republican	Ohio
Garfield J A	1831	0	49	Republican	Ohio
Arthur C A	1830	3	56	Republican	Vermont
Cleveland G	1837	8	71	Democratic	New Jersey
Harrison B	1833	4	67	Republican	Ohio
McKinley W	1843	4	58	Republican	Ohio
Roosevelt T	1858	7	60	Republican	New York
Taft W H	1857	4	72	Republican	Ohio
Wilson W	1856	8	67	Democratic	Virginia
Harding W G	1865	2	57	Republican	Ohio
Coolidge C	1872	5	60	Republican	Vermont
Hoover H C	1874	4	90	Republican	Iowa
Roosevelt F D	1882	12	63	Democratic	New York
Truman H S	1884	7	88	Democratic	Missouri
Eisenhower D D	1890	8	79	Republican	Texas
Kennedy J F	1917	2	46	Democratic	Massachusetts
Johnson L B	1908	5	65	Democratic	Texas
Nixon R M	1913	5	-	Republican	California
Ford G R	1913	2	-	Republican	Nebraska
Carter J E	1924	4	-	Democratic	Georgia
Reagan R	1911	3	-	Republican	Illinois

ตารางที่ 4-1 ตาราง President ที่ใช้ทดสอบโปรแกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

PRES_NAME	SPOUSE_NAME	PR_AGE	SP_AGE	NR-CHILDREN	MAR_YEAR
Washington G	Custis M D	26	27	0	1759
Adams J	Smith A	28	19	5	1764
Jefferson T	Skebon M W	28	23	6	1772
Madison J	Todd D D P	43	26	0	1794
Monroe J	Kortright E	27	17	3	1786
Adams J Q	Johnson L C	30	22	4	1797
Jackson A	Robards R D	26	26	0	1794
Van Buren M	Hoes H	24	23	4	1807
Harrison W H	Symmes A T	22	20	10	1795
Tyler J	Christian L	23	22	8	1813
Tyler J	Gardiner J	54	24	7	1844
Polk J K	Childress S	28	20	0	1824
Taylor Z	Smith M M	25	21	6	1810
Fillmore M	Power A	26	27	2	1826
Fillmore M	Melntosh C C	58	44	0	1858
Pierce F	Appieton J M	29	28	3	1834
Lincoln A	Todd M	33	33	4	1842
Johnson A	McCardle E	18	16	5	1827
Grant U S	Dent J B	26	22	4	1848
Hayes R B	Webb L W	30	21	8	1852
Garfield J A	Rudolph L	26	26	7	1858
Arthur C A	Herndon E L	29	22	3	1859
Cleveland G	Folson F	49	21	5	1886
Harrison B	Scott C L	20	21	2	1853
Harrison B	Dimmick M S L	62	37	1	1896
McKinley W	Saxton I	27	23	2	1871
Rooseveit T	Lee A H	22	19	1	1880
Rooseveit T	Carow E K	28	25	5	1886
Taft W H	Herron H	28	25	3	1886
Wilson W	Axson E L	28	25	3	1885
Wilson W	Gah E B	58	43	0	1915
Harding W G	De Wolfe F K	25	30	0	1891
Coolidge C	Kgoodhue G A	33	26	2	1905
Hoover H C	Henry L	24	23	2	1899
Roosevelt F D	Roosevelt A E	23	20	6	1905
Truman H S	Wallace E V	35	34	1	1919
Eisenhower D D	Doud G	25	19	2	1916
Kennedy J F	Bouvier J L	36	24	3	1953
Johnson L B	Taylor C A	26	21	2	1934
Nixon R M	Ryan T C	27	28	2	1940
Ford G R	Warren E B	35	30	4	1948
Carter J E	Smith R	21	18	4	1946
Reagan R	Wyman J	28	25	2	1940
Reagan R	Davis N	41	28	2	1952

ตารางที่ 4 - 2 ตาราง PresMarriage ที่ใช้ทดสอบโปรแกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่ควรนำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในระดับ Logical Database การเก็บข้อมูลจะเป็นในรูปลักษณะของระบบฐานข้อมูลแบบรีเลชันแนล ซึ่งจะมีรูปแบบการจัดเก็บข้อมูลต่าง ๆ ในรูปแบบของตาราง ดังตารางที่ 4 - 1 และ 4 - 2 โดยจะเป็นตารางที่ใช้สำหรับการทดสอบ โปรแกรมที่นำหลักการทำงานมาจากโปรแกรมวิซวลเอสคิวแอลและภาษาที่นำหลักการทำงานมาจากภาษาเอสคิวแอล

ส่วนการเก็บข้อมูลใน Physical Database ถูกออกแบบโดยใช้อินสแตนซ์ของคลาส Dictionary เนื่องจากมีความเหมาะสมในการนำไปใช้ในส่วนของโปรแกรมที่นำหลักการทำงานมาจากโปรแกรมวิซวลเอสคิวแอลและภาษาที่นำหลักการทำงานมาจากภาษาเอสคิวแอล โดยจะมีคีย์เป็นแอททริบิวต์ของแต่ละคอลัมน์ และจะมี value เป็นอินสแตนซ์ของคลาส OrderedCollection โดยที่ภายในจะมีข้อมูลของแต่ละคอลัมน์ ที่กำหนดให้เป็นเช่นนี้ เนื่องจากจะทำให้ไม่เกิดความซ้ำซ้อนในการเก็บชื่อของแอททริบิวต์ไว้ในทุก ๆ แถว และยังสามารถขยายเนื้อหาในการเก็บข้อมูลได้

โดยกำหนดคลาสขึ้นมาใหม่ ชื่อว่าคลาส RDB โดยกำหนดให้เป็นซับคลาสของคลาส Dictionary เพื่อที่จะได้รับการถ่ายทอดคุณสมบัติต่าง ๆ จากคลาส Dictionary โดยมีคีย์เป็นชื่อของแอททริบิวต์ และมีอินสแตนซ์เป็นคลาสที่สร้างขึ้นใหม่อีกเช่นกัน ชื่อว่าคลาส Data โดยกำหนดให้เป็นซับคลาสของคลาส OrderedCollection เพื่อที่จะได้รับการถ่ายทอดคุณสมบัติต่าง ๆ จากคลาส OrderedCollection ดังตารางที่ 4 - 3 และ 4 - 4

ได้กำหนดตัวแปรที่ใช้ภายในคลาส ภายในคลาส RDB ดังนี้

- name (ชื่อของตาราง)
- conditionTable (ตารางเงื่อนไข)

key	ข้อมูลภายในเป็นคลาส Data ที่เป็น subclass ของ คลาส Orderedcollection
key	ข้อมูลภายในเป็นคลาส Data ที่เป็น subclass ของ คลาส Orderedcollection
key	ข้อมูลภายในเป็นคลาส Data ที่เป็น subclass ของ คลาส Orderedcollection

ตารางที่ 4 - 3 ข้อมูลอินสแตนซ์ภายในคลาส RDB ที่เป็นซับคลาสของคลาส Dictionary

PRES_NAME	Washington G, Adams J, Jefferson T, Madison J, Monroe J, Adams J Q
BIRTH_YR	1732, 1735, 1743, 1751, 1758, 1767, 1767
YRS_SERV	7, 4, 8, 8, 8, 4, 8
DEATH_AGE	67, 90, 83, 85, 73, 80, 78
PARTY	Federalist, Federalist, Demo-Rep, Demo-Rep, Demo-Rep, Demo-Rep, Democratic.
STATE_BORN	Virginia, Massachusetts, Virginia, Virginia, Virginia, Massachusetts, South Carolina

ตารางที่ 4 - 4 ตัวอย่าง Physical Database ในคลาส RDB

4.2.2.2 คลาส VisualSQL

เป็นคลาสที่ออกแบบไว้รับข้อมูลจากผู้ใช้งานเข้ามา ข้อมูลเหล่านั้นจะถูกเก็บไว้ในอินสแตนซ์ของคลาส VisualSQL โดยกำหนดให้เป็นจับคลาสของคลาส Dictionary เพื่อที่จะได้รับการถ่ายทอดคุณสมบัติต่าง ๆ จากคลาส Dictionary โดยมีคีย์เป็นชื่อของแอททริบิวต์ และมีอินสแตนซ์เป็นคลาสของ Array ดังตารางที่ 4 - 5 และ 4 - 6

key	ข้อมูลภายในเป็นคลาส array
key	ข้อมูลภายในเป็นคลาส array
key	ข้อมูลภายในเป็นคลาส array

ตารางที่ 4 - 5 ข้อมูลอินสแตนซ์ภายในคลาส VisualSQL ที่เป็นจับคลาสของคลาส Dictionary

PRES_NAME	โดยส่วนนี้จะรองรับจากผู้ใช้งานในรูปแบบของ array
BIRTH_YR	โดยส่วนนี้จะรองรับจากผู้ใช้งานในรูปแบบของ array
YRS_SERV	โดยส่วนนี้จะรองรับจากผู้ใช้งานในรูปแบบของ array
DEATH_AGE	โดยส่วนนี้จะรองรับจากผู้ใช้งานในรูปแบบของ array
PARTY	โดยส่วนนี้จะรองรับจากผู้ใช้งานในรูปแบบของ array
STATE_BORN	โดยส่วนนี้จะรองรับจากผู้ใช้งานในรูปแบบของ array

ตารางที่ 4 - 6 ตัวอย่างส่วนที่ใช้ติดต่อกับผู้ใช้งาน อยู่ในคลาส VisualSQL

4.2.2.3 คลาส Data

เป็นคลาสที่ออกแบบไว้สำหรับการแปลงข้อมูลจากคลาส RDB เพื่อให้สามารถค้นหาข้อมูลได้ง่ายขึ้น เหตุผลที่ไม่เก็บฐานข้อมูลไว้ในคลาสนี้โดยตรง เนื่องจากจะเป็นการเปลืองเนื้อที่ของหน่วยความจำ เพราะจะมีความซ้ำซ้อนในการเก็บแอททริบิวต์ของตาราง โดยกำหนดให้เป็นจับคลาสของคลาส OrderedCollection เพื่อที่จะได้รับการถ่ายทอดคุณสมบัติต่าง ๆ จากคลาส OrderedCollection โดยมีอินสแตนซ์เป็นคลาสของ Dictionary โดยมีคีย์เป็นชื่อของแอททริบิวต์ และมีอินสแตนซ์เป็นคลาสของ String ดังรูปที่ 4 - 3 และ 4 - 4

key	คลาส string		key	คลาส string
key	คลาส string	,	key	คลาส string
key	คลาส string		key	คลาส string

รูปที่ 4 - 3 ข้อมูลอินสแตนซ์ภายในคลาส Data ที่เป็นจับคลาสของคลาส OrderedCollection

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

PRES_NAME	Washington G
BIRTH_YR	1732
YRS_SERV	7
DEATH_AGE	67
PARTY	Federalist
STATE_BORN	Virginia

PRES_NAME	Adams J
BIRTH_YR	1735
YRS_SERV	4
DEATH_AGE	90
PARTY	Federalist
STATE_BORN	Massachusetts

รูปที่ 4 - 4 ตัวอย่างฐานข้อมูลที่น่าไปค้นหาจริงๆ อยู่ในคลาส Data

4.3 ข้อจำกัดของระบบงาน

- ออบเจกต์แต่ละตัวในระบบต้องไม่ซ้ำกัน
- ชื่อของตาราง แต่ละตาราง ต้องต่อเนื่องกัน ไม่มีช่องว่างระหว่างคำ
- ชื่อของตาราง แต่ละตาราง ต้องต่อเนื่องกัน ไม่มีช่องว่างระหว่างคำ
- ชื่อของคอลัมน์ แต่ละคอลัมน์ ต้องต่อเนื่องกัน ไม่มีช่องว่างระหว่างคำ
- ข้อมูลที่ป้อนให้กับระบบ แต่ละคำ ต้องต่อเนื่องกัน ไม่มีช่องว่างระหว่างคำ

4.4 การกำหนดเมธอดในแต่ละคลาส

การออกแบบเมธอดนั้นเพื่อนำไปใช้ในโปรแกรมที่นำหลักการทำงานมาจากโปรแกรมวิวอล เอสคิวแอล และภาษาที่นำหลักการทำงานมาจากภาษาเอสคิวแอล

4.4.1 เมธอดในคลาส RDB

4.4.1.1 เมธอด makeConditionTable

เป็นเมธอดที่ใช้สำหรับการสร้างตาราง condition ขึ้นมา ไว้ในตัวแปรที่ใช้ภายในคลาสที่มีชื่อว่า conditionTable โดยกำหนดให้เป็นอินสแตนซ์ภายในคลาส VisualSQL โดยมีคีย์เป็นแอททริบิวต์ และมีอินสแตนซ์ เป็น nil ไว้ตอนเริ่มสร้าง เพื่อเตรียมรับข้อมูลจากผู้ใช้งาน

เช่น ถ้าต้องการสร้างตาราง condition ให้กับออบเจกต์ President ที่อยู่ในคลาส RDB จะเขียนว่า
President makeConditionTable

4.4.1.2 เมธอด getConditionTable

เป็นเมธอดที่ใช้สำหรับการนำข้อมูลจาก ตัวแปรที่ใช้ภายในคลาส ที่มีชื่อว่า condition ออกมาใช้งาน

เช่น ถ้าต้องการนำข้อมูลตาราง condition ของออบเจกต์ President ที่อยู่ในคลาส RDB ออกมาใช้ จะเขียนว่า

A := President getConditionTable

ดังนั้น ออบเจกต์ เอ จะมีค่าตาราง condition ของออบเจกต์ President

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.4.1.3 เมธอด putname: aString

เป็นเมธอดที่ใช้สำหรับกำหนดชื่อของตารางให้กับอินสแตนซ์ภายในคลาส RDB และชื่อมันจะถูกเก็บไว้ใน ตัวแปรที่ใช้ภายในคลาส ที่มีชื่อว่า name

เช่น ถ้าต้องการกำหนดชื่อของตารางให้มีชื่อว่า President ให้กับออบเจกต์ President ที่อยู่ในคลาส RDB จะเขียนว่า

President putname: 'President'

4.4.1.4 เมธอด getname

เป็นเมธอดที่ใช้สำหรับทำการนำข้อมูลจากตัวแปรที่ใช้ภายในคลาส ที่มีชื่อว่า name. ออกมาใช้งาน

เช่น ถ้าต้องการนำชื่อของตารางจากออบเจกต์ President ที่อยู่ในคลาส RDB ออกมาใช้ จะเขียนว่า

A := President getname

ดังนั้น ออบเจกต์ เอ จะมีค่าของชื่อตาราง ของออบเจกต์ President

4.4.1.5 เมธอด runVisualSQL

เป็นเมธอดที่ใช้สำหรับนำเอาข้อมูลที่อยู่ในตัวแปรที่ใช้ภายในคลาส ที่มีชื่อว่า conditionTable ซึ่งตอนแรกเราสร้างให้มีคีย์เป็นแอททริบิวต์ และมีอินสแตนซ์เป็น nil นั้นมารอให้ผู้ใช้งานป้อนข้อมูลที่ต้องการจะค้นหาและให้แสดงผลต่อไป

เช่น ถ้าต้องการค้นหาข้อมูลในตาราง President ซึ่งเป็นออบเจกต์ในคลาส RDB นั้น จะเขียนว่า

President runVisualSQL

4.4.2 เมธอดในคลาส VisualSQL

4.4.2.1 เมธอด run

เมื่อผู้ใช้งานโปรแกรมที่นำหลักการทำงานมาจากโปรแกรมวิวอลเอสคิวแอล ทำการป้อนข้อมูลให้กับตาราง Condition ซึ่งอยู่ภายในคลาส VisualSQL เรียบร้อยแล้ว จะทำการเรียกใช้เมธอดนี้ เพื่อที่จะนำข้อมูลที่ป้อนนั้นไปประมวลผลออกมาเป็นข้อความในรูปแบบของภาษาที่นำหลักการทำงานมาจากภาษาเอสคิวแอล ต่อไป

เช่น A เป็นออบเจกต์ภายในคลาส VisualSQL ซึ่งใช้สำหรับเก็บตาราง condition เมื่อป้อนข้อมูลไปในตาราง condition เรียบร้อยแล้ว ต้องการสั่งให้โปรแกรมที่นำหลักการทำงานมาจากโปรแกรมวิวอลเอสคิวแอลเริ่มทำงาน จะเขียนว่า

A run

เมื่อใช้เมธอด run โปรแกรมจะทำการประมวลผลได้ผลลัพธ์ออกมาเป็นข้อความในรูปแบบของภาษาที่นำหลักการทำงานมาจากภาษาเอสคิวแอล

4.4.2.2 เมธอด scan

เป็นเมธอดที่ใช้สำหรับ scan (ตรวจสอบ) ข้อมูลภายในคลาส VisualSQL โดยจะได้ผลลัพธ์เป็นออบเจกต์ที่อยู่ในคลาส VisualSQL อีกเช่นกัน โดยมีคีย์ เป็น select , where , orwhere , having , orhaving , groupby , order , join และมีอินสแตนซ์เป็นออบเจกต์ภายในคลาส OrderedCollection โดยการ scan ข้อมูลที่รับเข้ามานั้นจะทำการดูว่าข้อมูลนั้นเป็นเงื่อนไขอะไรตามคีย์ที่ได้ตั้งไว้ แล้วนำไปใส่ไว้ในคีย์นั้นๆ

เช่น ถ้าต้องการ scan ออบเจกต์ A ที่เป็นออบเจกต์ของคลาส VisualSQL ที่ใช้สำหรับเก็บตาราง condition ซึ่งมีข้อมูลที่ได้ถูกป้อนโดยผู้ใส่แล้ว จะเขียนว่า

A scan

4.4.2.3 เมธอด chkOrGroupSelect: aVisualSQL

เป็นเมธอดที่ใช้สำหรับตรวจสอบเงื่อนไขเกี่ยวกับการ select , groupby , order ของออบเจกต์ภายในคลาส VisualSQL โดยมีการส่งผ่านอาร์กิวเมนต์ เป็นออบเจกต์ภายในคลาส VisualSQL อีกเช่นกัน ไว้สำหรับเก็บข้อมูลที่ได้ทำการตรวจสอบแล้ว โดยมีคีย์เป็น select , where , orwhere , having , orhaving , groupby , order , join และมีอินสแตนซ์เป็นออบเจกต์ภายในคลาส OrderedCollection

เช่น ถ้าต้องการตรวจสอบเงื่อนไขเกี่ยวกับการ select , groupby , order ของออบเจกต์ A ที่เป็นออบเจกต์ของคลาส VisualSQL ที่ใช้สำหรับเก็บตาราง condition ซึ่งมีข้อมูลที่ได้ถูกป้อนโดยผู้ใส่ และต้องผ่านอาร์กิวเมนต์ที่อยู่ในคลาส VisualSQL ในที่นี้ให้เป็นออบเจกต์ B โดยมีคีย์เป็น select , where , orwhere , having , orhaving , groupby , order , join และมีอินสแตนซ์เป็นออบเจกต์ภายในคลาส OrderedCollection จะเขียนว่า

B := A chkOrGroupSelect: B

ผลลัพธ์ที่ได้จะเป็นออบเจกต์ B ที่มีข้อมูลในคีย์ที่ชื่อ select , groupby , order เพิ่มเติมเข้ามา

4.4.2.4 เมธอด chkHaving: aVisualSQL

เป็นเมธอดที่ใช้สำหรับตรวจสอบเงื่อนไขเกี่ยวกับการ having แบบ and ของออบเจกต์ภายในคลาส VisualSQL โดยมีการส่งผ่านอาร์กิวเมนต์เป็นออบเจกต์ภายในคลาส VisualSQL อีกเช่นกัน ไว้สำหรับเก็บข้อมูลที่ได้ทำการตรวจสอบแล้ว โดยมีคีย์เป็น select , where , orwhere , having , orhaving , groupby , order , join และมีอินสแตนซ์เป็นออบเจกต์ภายในคลาส OrderedCollection

เช่น ถ้าต้องการตรวจสอบเงื่อนไขเกี่ยวกับการ having แบบ and ของออบเจกต์ A ที่เป็นออบเจกต์ของคลาส VisualSQL ที่ใช้สำหรับเก็บตาราง condition ซึ่งมีข้อมูลที่ได้ถูกป้อนโดยผู้ใส่ และต้องผ่านอาร์กิวเมนต์ที่อยู่ในคลาส VisualSQL ในที่นี้ให้เป็นออบเจกต์ B โดยมีคีย์เป็น select , where , orwhere , having , orhaving , groupby , order , join และมีอินสแตนซ์เป็นออบเจกต์ภายในคลาส OrderedCollection จะเขียนว่า

B := A chkOrGroupSelect: B

ผลลัพธ์ที่ได้จะเป็นออบเจกต์ B ที่มีข้อมูลในคีย์ที่ชื่อ having เพิ่มเติมเข้ามา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น มิได้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.4.2.5 เมธอด chkOrHaving: aVisualSQL

เป็นเมธอดที่ใช้สำหรับตรวจสอบเงื่อนไขเกี่ยวกับการ having แบบ or ของออบเจกต์ภายในคลาส VisualSQL โดยมีการส่งผ่านอาร์กิวเมนต์เป็นออบเจกต์ภายในคลาส VisualSQL อีกเช่นกัน ไว้สำหรับเก็บข้อมูลที่ได้ทำการตรวจสอบแล้ว โดยมีคีย์เป็น select , where , orwhere , having , orhaving , groupby , order , join และมีอินสแตนซ์เป็นออบเจกต์ภายในคลาส OrderedCollection

เช่น ถ้าต้องการตรวจสอบเงื่อนไขเกี่ยวกับการ having แบบ or ของออบเจกต์ A ที่เป็นออบเจกต์ของคลาส VisualSQL ที่ใช้สำหรับเก็บตาราง condition ซึ่งมีข้อมูลที่ได้ถูกป้อนโดยผู้ใช้ และต้องผ่านอาร์กิวเมนต์ที่อยู่ในคลาส VisualSQL ในที่นี้ให้เป็นออบเจกต์ B โดยมีคีย์เป็น select , where , orwhere , having , orhaving , groupby , order , join และมีอินสแตนซ์เป็นออบเจกต์ภายในคลาส OrderedCollection จะเขียนว่า

B := A chkOrGroupSelect: B

ผลลัพธ์ที่ได้จะเป็นออบเจกต์ B ที่มีข้อมูลในคีย์ที่ชื่อ orhaving เพิ่มเติมเข้ามา

4.4.2.6 เมธอด chkWhere: aVisualSQL

เป็นเมธอดที่ใช้สำหรับตรวจสอบเงื่อนไขเกี่ยวกับการ where แบบ and ของออบเจกต์ภายในคลาส VisualSQL โดยมีการส่งผ่านอาร์กิวเมนต์เป็นออบเจกต์ภายในคลาส VisualSQL อีกเช่นกัน ไว้สำหรับเก็บข้อมูลที่ได้ทำการตรวจสอบแล้ว โดยมีคีย์เป็น select , where , orwhere , having , orhaving , groupby , order , join และมีอินสแตนซ์เป็นออบเจกต์ภายในคลาส OrderedCollection

เช่น ถ้าต้องการตรวจสอบเงื่อนไขเกี่ยวกับการ where แบบ and ของออบเจกต์ A ที่เป็นออบเจกต์ของคลาส VisualSQL ที่ใช้สำหรับเก็บตาราง condition ซึ่งมีข้อมูลที่ได้ถูกป้อนโดยผู้ใช้ และต้องผ่านอาร์กิวเมนต์ที่อยู่ในคลาส VisualSQL ในที่นี้ให้เป็นออบเจกต์ B โดยมีคีย์เป็น select , where , orwhere , having , orhaving , groupby , order , join และมีอินสแตนซ์เป็นออบเจกต์ภายในคลาส OrderedCollection จะเขียนว่า

B := A chkOrGroupSelect: B

ผลลัพธ์ที่ได้จะเป็นออบเจกต์ B ที่มีข้อมูลในคีย์ที่ชื่อ where เพิ่มเติมเข้ามา

4.4.2.7 เมธอด chkOrWhere: aVisualSQL

เป็นเมธอดที่ใช้สำหรับตรวจสอบเงื่อนไขเกี่ยวกับการ where แบบ or ของออบเจกต์ภายในคลาส VisualSQL โดยมีการส่งผ่านอาร์กิวเมนต์เป็นออบเจกต์ภายในคลาส VisualSQL อีกเช่นกัน ไว้สำหรับเก็บข้อมูลที่ได้ทำการตรวจสอบแล้ว โดยมีคีย์เป็น select , where , orwhere , having , orhaving , groupby , order , join และมีอินสแตนซ์เป็นออบเจกต์ภายในคลาส OrderedCollection

เช่น ถ้าต้องการตรวจสอบเงื่อนไขเกี่ยวกับการ where แบบ or ของออบเจกต์ A ที่เป็นออบเจกต์ของคลาส VisualSQL ที่ใช้สำหรับเก็บตาราง condition ซึ่งมีข้อมูลที่ได้ถูกป้อนโดยผู้ใช้ และต้องผ่านอาร์กิวเมนต์ที่อยู่ในคลาส VisualSQL ในที่นี้ให้เป็นออบเจกต์ B โดยมีคีย์เป็น select , where , orwhere , having ,

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น เมื่ออนุญาตให้นำไปเผยแพร่โดยไม่มีการดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

orhaving , groupby , order , join และมีอินสแตนซ์เป็นออบเจกต์ภายในคลาส OrderedCollection จะเขียนว่า

`B := A chkOrGroupSelect: B`

ผลลัพธ์ที่ได้จะเป็นออบเจกต์ B ที่มีข้อมูลในคีย์ที่ชื่อ orwhere เพิ่มเติมเข้ามา

4.4.2.8 เมธอด createSQL

เป็นเมธอดที่ใช้สำหรับสร้าง string ทั้งหมดที่อยู่ในรูปแบบของภาษาที่นำหลักการทำงานมาจากภาษาเอสคิวแอล โดยออบเจกต์ที่ใช้จะต้องมีคีย์เป็น select , where , orwhere , having , orhaving , groupby , order , join และมีอินสแตนซ์เป็นออบเจกต์ภายในคลาส OrderedCollection

เช่น ถ้าต้องการสร้าง string ทั้งหมดที่อยู่ในรูปแบบของภาษาที่นำหลักการทำงานมาจากภาษาเอสคิวแอล ให้กับออบเจกต์ A ที่อยู่ในคลาส VisualSQL จะเขียนว่า

`A createSQL`

4.4.2.9 เมธอด sjoin: aString

เป็นเมธอดที่ใช้สำหรับเพิ่ม string ในส่วนของการ join ที่อยู่ในรูปแบบของภาษาที่นำหลักการทำงานมาจากภาษาเอสคิวแอล เข้าไปในส่วนของอาร์กิวเมนต์ที่รับเข้ามาโดยออบเจกต์ที่ใช้จะต้องมีคีย์ที่มีชื่อว่า join และมีอินสแตนซ์เป็นออบเจกต์ภายในคลาส OrderedCollection

เช่น ถ้าต้องการเพิ่ม string เข้าไปในออบเจกต์ B ซึ่งอยู่ในคลาส String ในส่วนของการ join ที่อยู่ในรูปแบบของภาษาที่นำหลักการทำงานมาจากภาษาเอสคิวแอล ให้กับออบเจกต์ A ที่อยู่ในคลาส VisualSQL จะเขียนว่า

`A sjoin: B`

4.4.2.10 เมธอด swhere: aString

เป็นเมธอดที่ใช้สำหรับเพิ่ม string ในส่วนของการ where ทั้งในแบบ and และ or ที่อยู่ในรูปแบบของภาษาที่นำหลักการทำงานมาจากภาษาเอสคิวแอล เข้าไปในส่วนของอาร์กิวเมนต์ที่รับเข้ามา โดยออบเจกต์ที่ใช้จะต้องมีคีย์ที่มีชื่อว่า where , orwhere และมีอินสแตนซ์เป็นออบเจกต์ภายในคลาส OrderedCollection

เช่น ถ้าต้องการเพิ่ม string เข้าไปในออบเจกต์ B ซึ่งอยู่ในคลาส String ในส่วนของการ where ทั้งในแบบ and และ or ที่อยู่ในรูปแบบของภาษาที่นำหลักการทำงานมาจากภาษาเอสคิวแอล ให้กับออบเจกต์ A ที่อยู่ในคลาส VisualSQL จะเขียนว่า

`A swhere: B`

4.4.2.11 เมธอด sgroupBy: aString

เป็นเมธอดที่ใช้สำหรับเพิ่ม string ในส่วนของการ groupby ที่อยู่ในรูปแบบของภาษาที่นำหลักการทำงานมาจากภาษาเอสคิวแอล เข้าไปในส่วนของอาร์กิวเมนต์ที่รับเข้ามา โดยออบเจกต์ที่ใช้จะต้องมีคีย์ที่มีชื่อว่า groupBy และมีอินสแตนซ์เป็นออบเจกต์ภายในคลาส OrderedCollection

เช่น ถ้าต้องการเพิ่ม string เข้าไปในออบเจกต์ B ซึ่งอยู่ในคลาส String ในส่วนของการ groupby ที่อยู่ในรูปแบบของภาษาที่นำหลักการทำงานมาจากภาษาเอสคิวแอล ให้กับออบเจกต์ A ที่อยู่ภายในคลาส VisualSQL จะเขียนว่า

A sgroupBy: B

4.4.2.12 เมธอด shaving: aString

เป็นเมธอดที่ใช้สำหรับเพิ่ม string ในส่วนของการ having ทั้งในแบบ and และ or ที่อยู่ในรูปแบบของภาษาที่นำหลักการทำงานมาจากภาษาเอสคิวแอล เข้าไปในส่วนของอาร์กิวเมนต์ที่รับเข้ามา โดยออบเจกต์ที่ใช้จะต้องมีคีย์ที่มีชื่อว่า having , orhaving และมีอินสแตนซ์เป็นออบเจกต์ภายในคลาส OrderedCollection

เช่น ถ้าต้องการเพิ่ม string เข้าไปในออบเจกต์ B ซึ่งอยู่ในคลาส String ในส่วนของการ having ทั้งในแบบ and และ or ที่อยู่ในรูปแบบของภาษาที่นำหลักการทำงานมาจากภาษาเอสคิวแอล ให้กับออบเจกต์ A ที่อยู่ภายในคลาส VisualSQL จะเขียนว่า

A shaving: B

4.4.2.13 เมธอด sselect: aString

เป็นเมธอดที่ใช้สำหรับเพิ่ม string ในส่วนของการ select ที่อยู่ในรูปแบบของภาษาที่นำหลักการทำงานมาจากภาษาเอสคิวแอล เข้าไปในส่วนของอาร์กิวเมนต์ที่รับเข้ามา โดยออบเจกต์ที่ใช้จะต้องมีคีย์ที่มีชื่อว่า select และมีอินสแตนซ์เป็นออบเจกต์ภายในคลาส OrderedCollection

เช่น ถ้าต้องการเพิ่ม string เข้าไปในออบเจกต์ B ซึ่งอยู่ในคลาส String ในส่วนของการ select ที่อยู่ในรูปแบบของภาษาที่นำหลักการทำงานมาจากภาษาเอสคิวแอล ให้กับออบเจกต์ A ที่อยู่ภายในคลาส VisualSQL จะเขียนว่า

A sselect: B

4.4.2.14 เมธอด sorder: aString

เป็นเมธอดที่ใช้สำหรับเพิ่ม string ในส่วนของการ order ที่อยู่ในรูปแบบของภาษาที่นำหลักการทำงานมาจากภาษาเอสคิวแอล เข้าไปในส่วนของอาร์กิวเมนต์ที่รับเข้ามา โดยออบเจกต์ที่ใช้จะต้องมีคีย์ที่มีชื่อว่า order และมีอินสแตนซ์เป็นออบเจกต์ภายในคลาส OrderedCollection

เช่น ถ้าต้องการเพิ่ม string เข้าไปในออบเจก B ซึ่งอยู่ในคลาส String ในส่วนของการ order ที่อยู่ในรูปแบบของภาษาที่นำหลักการทำงานมาจากภาษาเอสคิวแอล ให้กับออบเจก A ที่อยู่ในคลาส VisualSQL จะเขียนว่า

A sorder: B

4.4.3 เมธอดในคลาส Data

เมธอดในคลาส Data ใช้สำหรับค้นหาและแสดงผล จากข้อมูลในตาราง ที่เป็นออบเจกในคลาส Data ที่ถูกแปลงมาจากคลาส RDB

4.4.3.1 เมธอด select: aBlock

เป็นเมธอดที่ใช้สำหรับเลือกเฉพาะ คอลัมน์ ที่ต้องการจากออบเจกในคลาส Data

เช่น ถ้าต้องการเลือกคอลัมน์ที่มีชื่อว่า PRES_NAME ในออบเจก A ที่อยู่ในคลาส Data จะเขียนว่า

A select: [:c | (c choose: #'PRES_NAME'))]

โดยผลลัพธ์ที่ได้จะเป็นข้อมูลของคอลัมน์ที่เลือกไว้เท่านั้น

4.4.3.2 เมธอด where: aBlock

เป็นเมธอดที่ใช้สำหรับเลือกแถว ตามเงื่อนไขที่ต้องการจากออบเจกในคลาส Data

เช่น ถ้าต้องการเลือกแถว ตามเงื่อนไข NR_CHILDREN มากกว่า 3 ในออบเจก A ที่อยู่ในคลาส Data จะเขียนว่า

A where: [:c | (c at: 'NR_CHILDREN') > 3)]

โดยผลลัพธ์ที่ได้จะเป็นข้อมูลของแถว ตามเงื่อนไขที่ต้องการเท่านั้น

4.4.3.3 เมธอด groupBy: aColumn

เป็นเมธอดที่ใช้สำหรับการจัดกลุ่มของข้อมูลที่มีค่าเดียวกันภายในคอลัมน์ที่กำหนด ภายในคลาส Data

เช่น ถ้าต้องการ groupby คอลัมน์ PRES_NAME ในออบเจก A ที่อยู่ในคลาส Data จะเขียนว่า

A groupBy: 'PRES_NAME'

ผลลัพธ์ที่ได้จะทำการเรียงแถวที่มีค่าเดียวกัน ไว้ติดกันตามคอลัมน์ที่กำหนดไว้

4.4.3.4 เมธอด having: aBlock

เป็นเมธอดที่ใช้สำหรับเลือกแถว ตามเงื่อนไขที่ต้องการที่ทำกับ built-in function จากออบเจกในคลาส Data

เช่น ถ้าต้องการเลือกแถว ตามเงื่อนไขผลรวมของ NR_CHILDREN มากกว่า 3 ในออบเจก A ที่อยู่ในคลาส Data จะเขียนว่า

A where: [:c | (c sum: 'NR_CHILDREN') > 3)]

โดยผลลัพธ์ที่ได้จะเป็นข้อมูลของแถว ตามเงื่อนไขที่ต้องการเท่านั้น

4.4.3.5 เมธอด orderAsc: anInteger

เป็นเมธอดที่ใช้สำหรับการเรียงลำดับข้อมูลโดยพิจารณาจากลำดับที่ระบุ จากน้อยไปมากภายในคลาส Data

เช่น ถ้าต้องการเรียงลำดับข้อมูลจากน้อยไปมาก ของคอลัมน์ที่ 2 ที่ปรากฏเป็นผลลัพธ์ ในออบเจก A ที่อยู่ในคลาส Data จะเขียนว่า

A orderAsc: 2

ผลลัพธ์ที่ได้จะทำการเรียงลำดับข้อมูลจากน้อยไปมาก ของคอลัมน์ที่ 2 ที่ปรากฏเป็นผลลัพธ์

4.4.3.6 เมธอด orderDec: anInteger

เป็นเมธอดที่ใช้สำหรับการเรียงลำดับข้อมูลโดยพิจารณาจากลำดับที่ระบุ จากมากไปน้อยภายในคลาส Data

เช่น ถ้าต้องการเรียงลำดับข้อมูลจากมากไปน้อย ของคอลัมน์ที่ 2 ที่ปรากฏเป็นผลลัพธ์ ในออบเจก A ที่อยู่ในคลาส Data จะเขียนว่า

A orderDec: 2

ผลลัพธ์ที่ได้จะทำการเรียงลำดับข้อมูลจากมากไปน้อย ของคอลัมน์ที่ 2 ที่ปรากฏเป็นผลลัพธ์

4.4.3.7 เมธอด join: aData

เป็นเมธอดที่ใช้สำหรับทำ natural join ระหว่าง 2 ออบเจกที่อยู่ในคลาส Data

เช่น ถ้าต้องการทำ natural join ระหว่างออบเจก A และ B ที่อยู่ในคลาส Data จะเขียนได้ว่า

A join: B

4.4.3.8 เมธอด sum: aColumn

เป็นเมธอดที่ใช้สำหรับทำการหาผลรวมของค่าทั้งหมด ที่อยู่ในคอลัมน์นั้น

เช่น ถ้าต้องการทำการหาผลรวมของค่าทั้งหมด ที่อยู่ในคอลัมน์ NR_CHILDREN ของออบเจก A ภายในคลาส Data จะเขียนได้ว่า

A sum: 'NR_CHILDREN'

ผลลัพธ์ที่ได้จะเป็นตัวเลขผลรวมที่ได้ทั้งหมด ที่อยู่ในคอลัมน์ที่กำหนด

4.4.3.9 เมธอด min: aColumn

เป็นเมธอดที่ใช้สำหรับทำการหาค่าที่น้อยที่สุดที่อยู่ภายในคอลัมน์นั้น

เช่น ถ้าต้องการทำการหาค่าที่น้อยที่สุดที่อยู่ภายในคอลัมน์ NR_CHILDREN ของออบเจกต์ A ภายในคลาส Data จะเขียนได้ว่า

A min: 'NR_CHILDREN'

ผลลัพธ์ที่ได้จะเป็นตัวเลขของค่าที่น้อยที่สุดที่อยู่ภายในคอลัมน์ที่กำหนด

4.4.3.10 เมธอด max: aColumn

เป็นเมธอดที่ใช้สำหรับทำการหาค่าที่มากที่สุดที่อยู่ภายในคอลัมน์นั้น

เช่น ถ้าต้องการทำการหาค่าที่มากที่สุดที่อยู่ภายในคอลัมน์ NR_CHILDREN ของออบเจกต์ A ภายในคลาส Data จะเขียนได้ว่า

A max: 'NR_CHILDREN'

ผลลัพธ์ที่ได้จะเป็นตัวเลขของค่าที่มากที่สุดที่อยู่ภายในคอลัมน์ที่กำหนด

4.4.3.11 เมธอด avg: aColumn

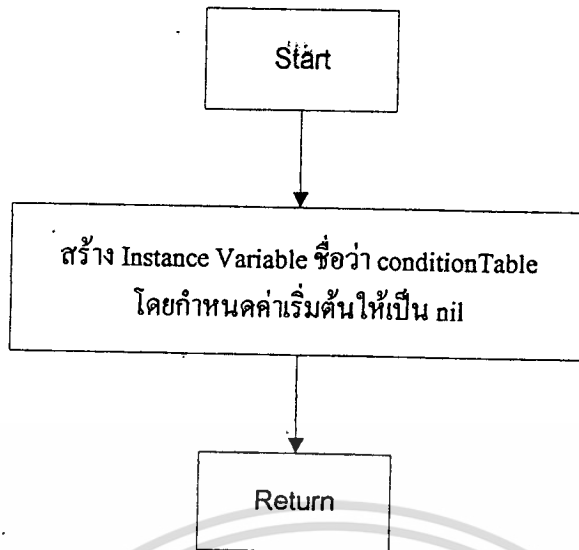
เป็นเมธอดที่ใช้สำหรับทำการหาค่าเฉลี่ยที่อยู่ภายในคอลัมน์นั้น

เช่น ถ้าต้องการทำการหาค่าเฉลี่ยที่อยู่ภายในคอลัมน์ NR_CHILDREN ของออบเจกต์ A ภายในคลาส Data จะเขียนได้ว่า

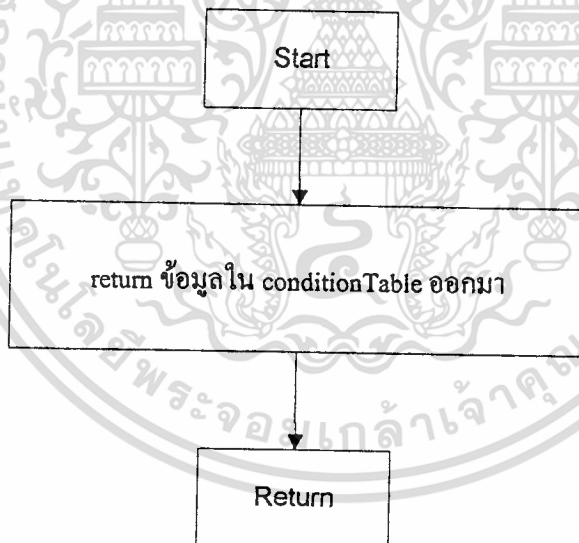
A avg: 'NR_CHILDREN'

ผลลัพธ์ที่ได้จะเป็นตัวเลขของค่าเฉลี่ยที่อยู่ภายในคอลัมน์ที่กำหนด

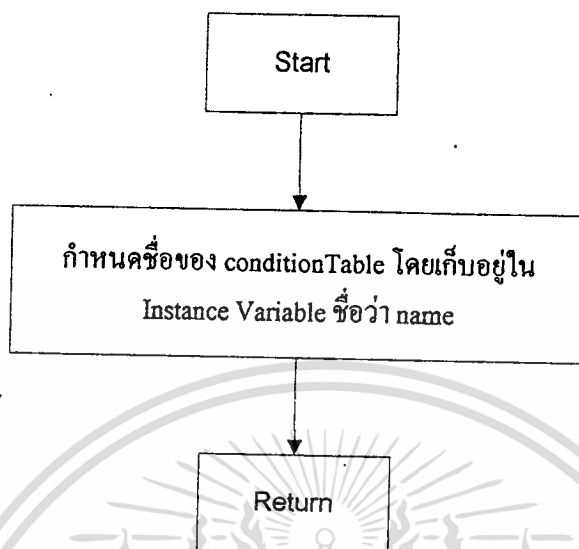
จากเมธอดของคลาส Data , VisualSQL และ RDB สามารถนำมาเขียนเป็น Flow Chart ได้ดังรูปที่ 4 - 5 ถึงรูปที่ 4 - 34



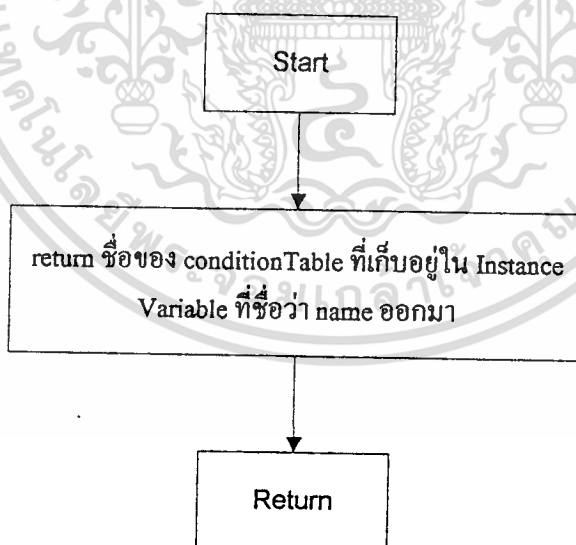
รูปที่ 4 - 5 แสดง Flow chart ของเมธอด `makeConditionTable`



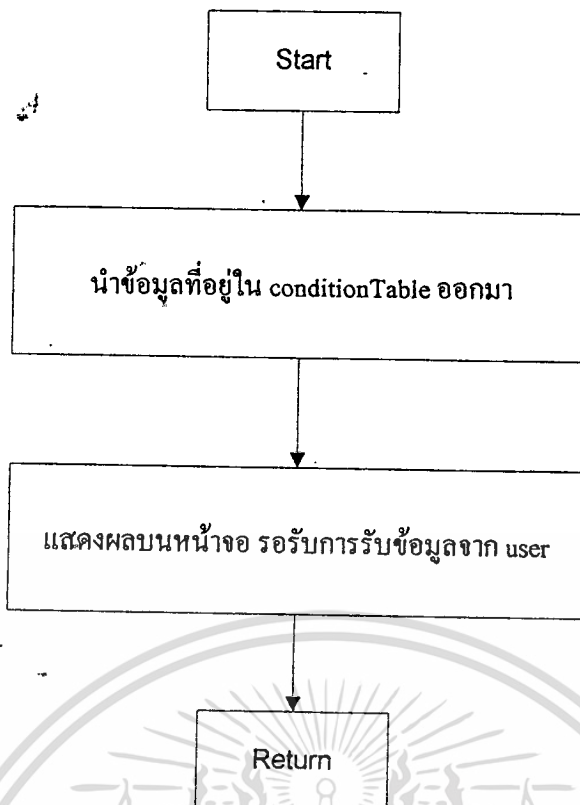
รูปที่ 4 - 6 แสดง Flow chart ของเมธอด `getConditionTable`



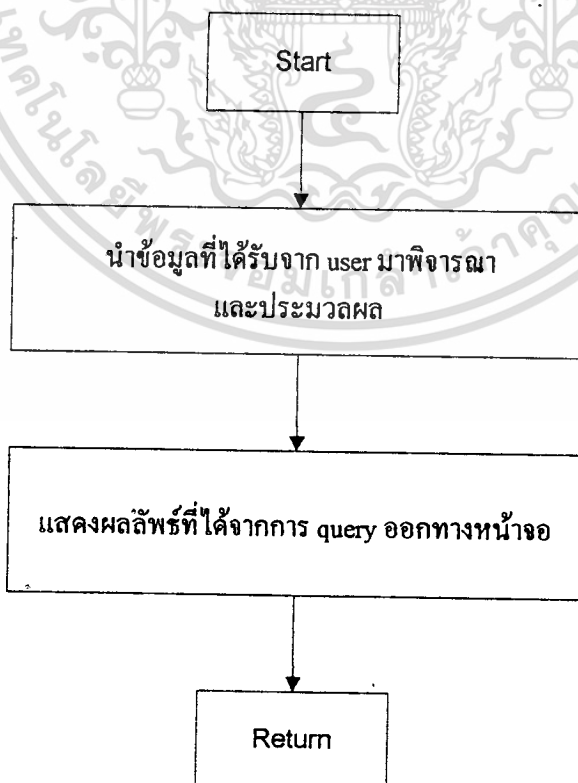
รูปที่ 4 - 7 แสดง Flow chart ของเมธอด `putname`:

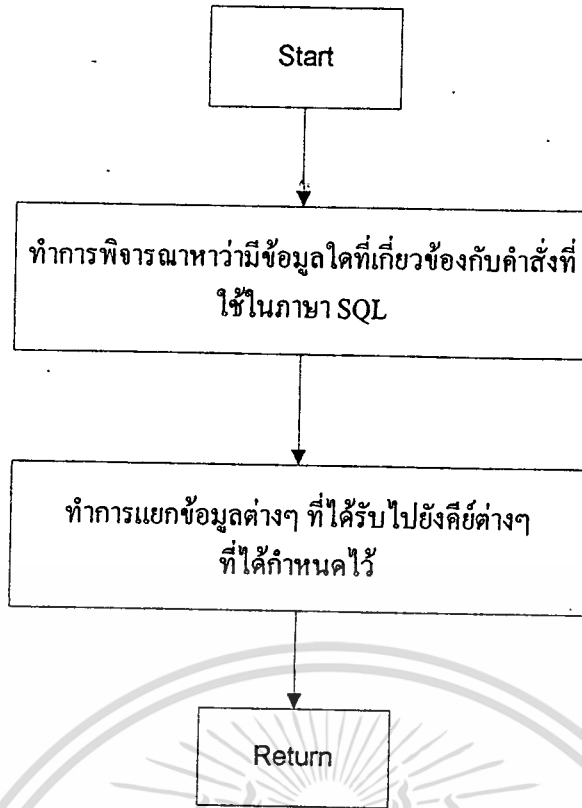


รูปที่ 4 - 8 แสดง Flow chart ของเมธอด `getname`

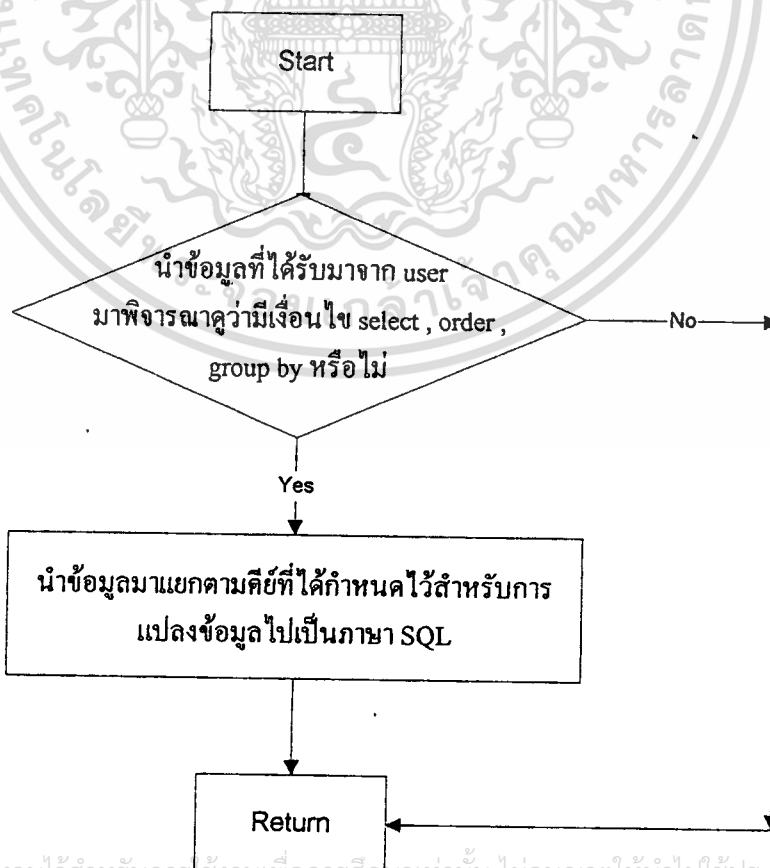


รูปที่ 4 - 9 แสดง Flow chart ของเมธอด runVisualSQL

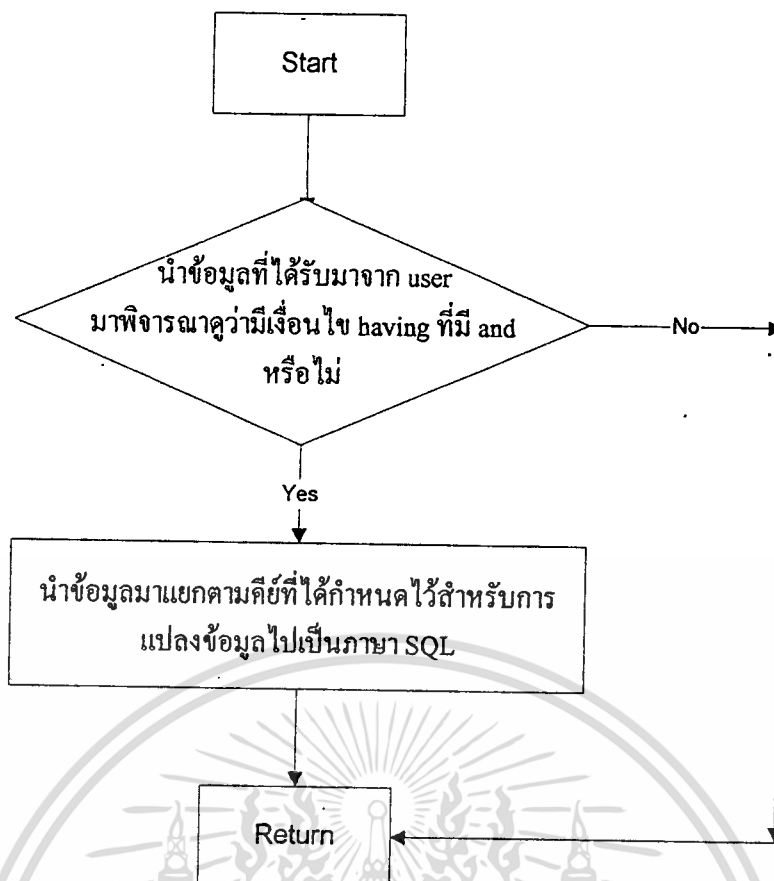




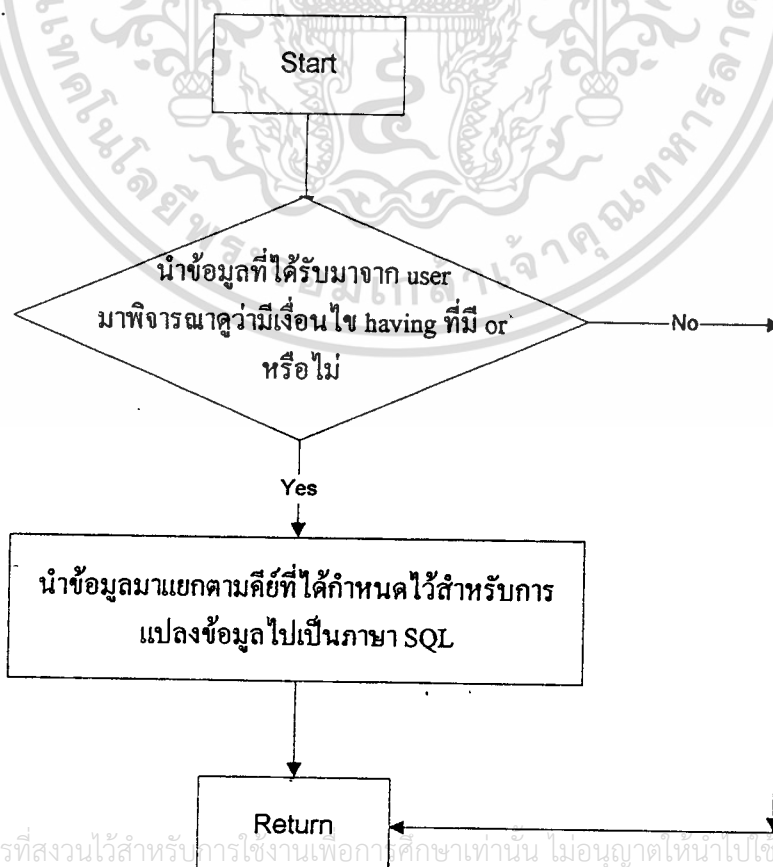
รูปที่ 4 - 11 แสดง Flow chart ของเมธอด scan



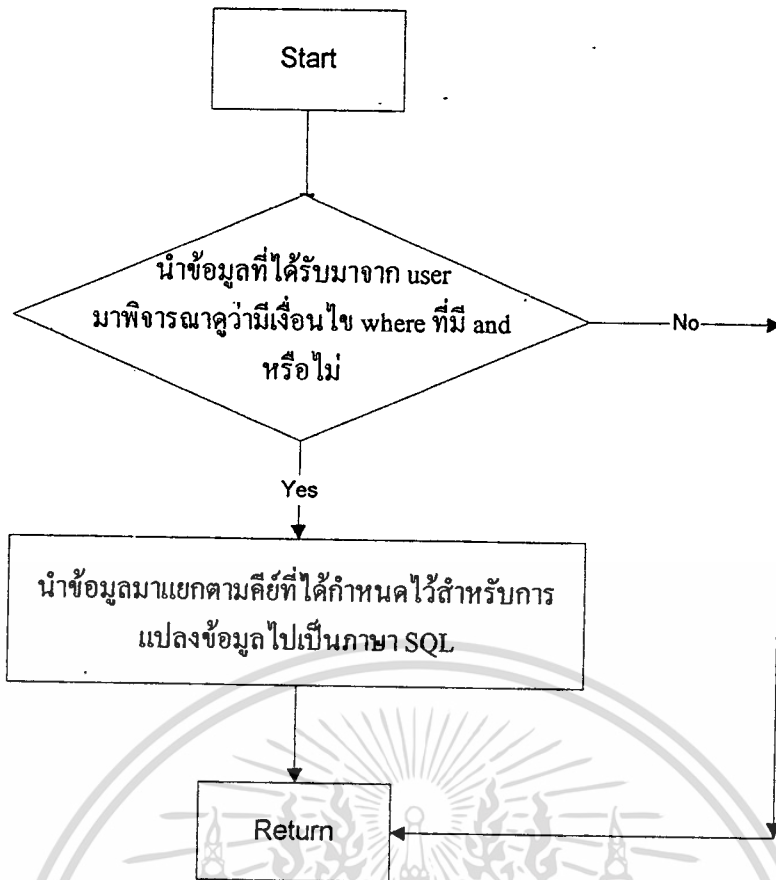
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น **รูปที่ 4 - 12 แสดง Flow chart ของเมธอด chkOrGroupSelect:** สารทุกครั้งที่มีการนำไปใช้



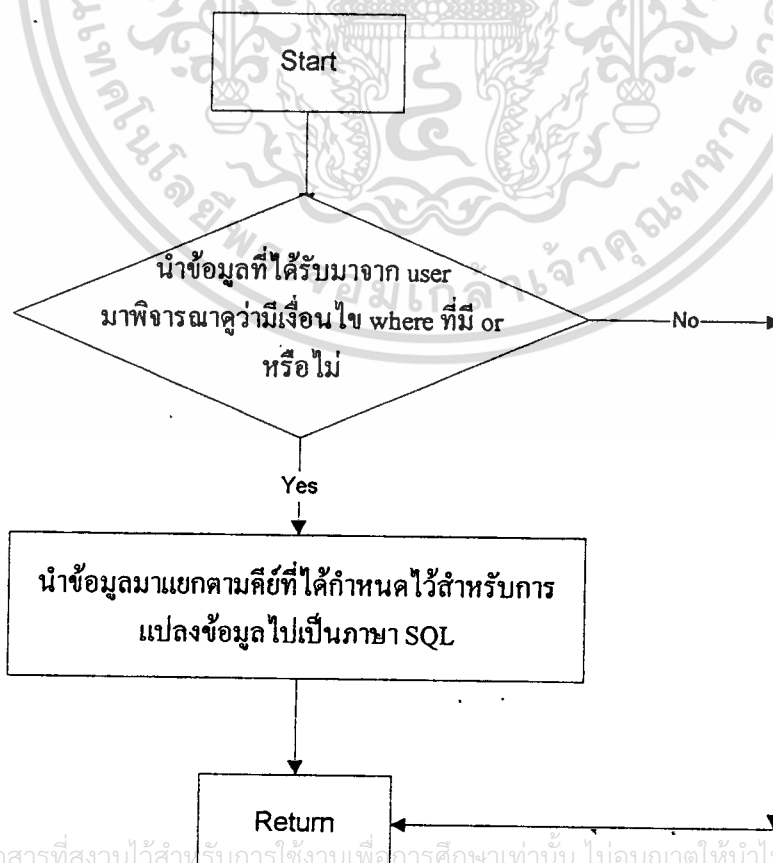
รูปที่ 4 - 13 แสดง Flow chart ของเมธอด chkHaving:



รูปที่ 4 - 14 แสดง Flow chart ของเมธอด chkOrHaving:



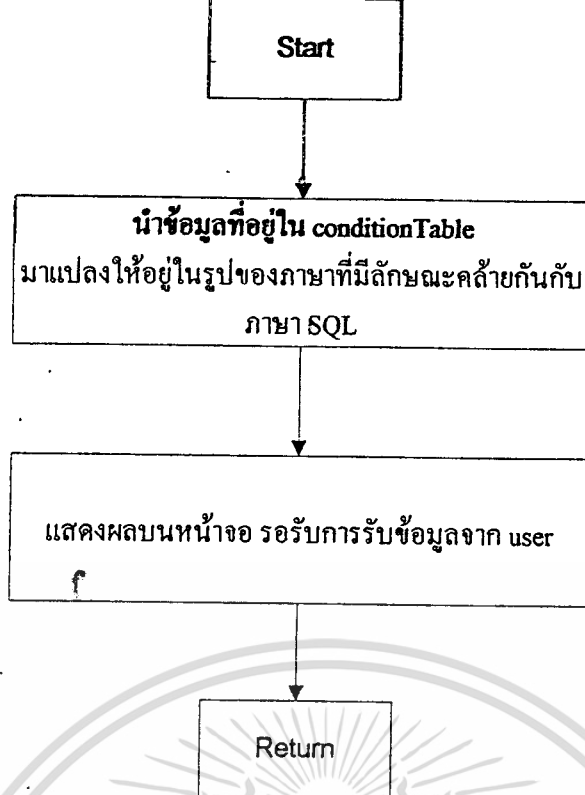
รูปที่ 4 - 15 แสดง Flow chart ของเมธอด *chkWhere*:



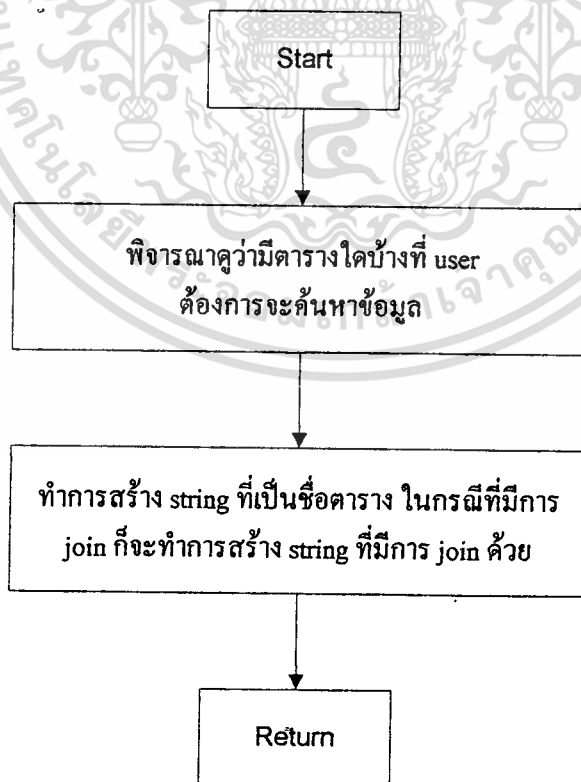
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

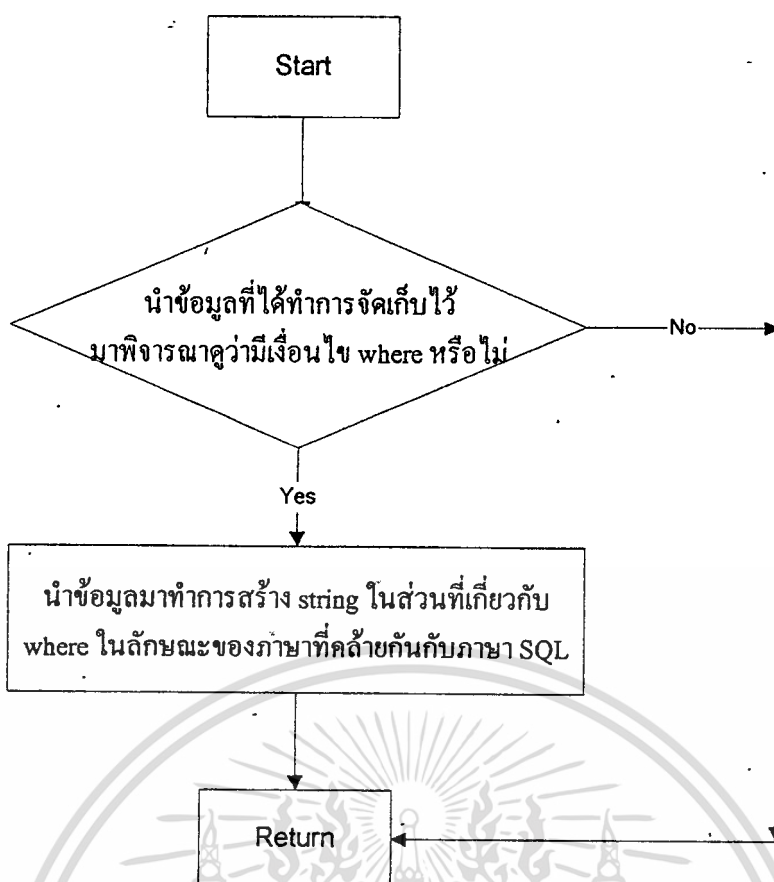
ไม่ว่ากรณีใดๆ ทั้งนี้ ลิขสิทธิ์ทั้งหมดจะยังคงอยู่กับเรา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 4 - 16 แสดง Flow chart ของเมธอด *chkOrWhere*:

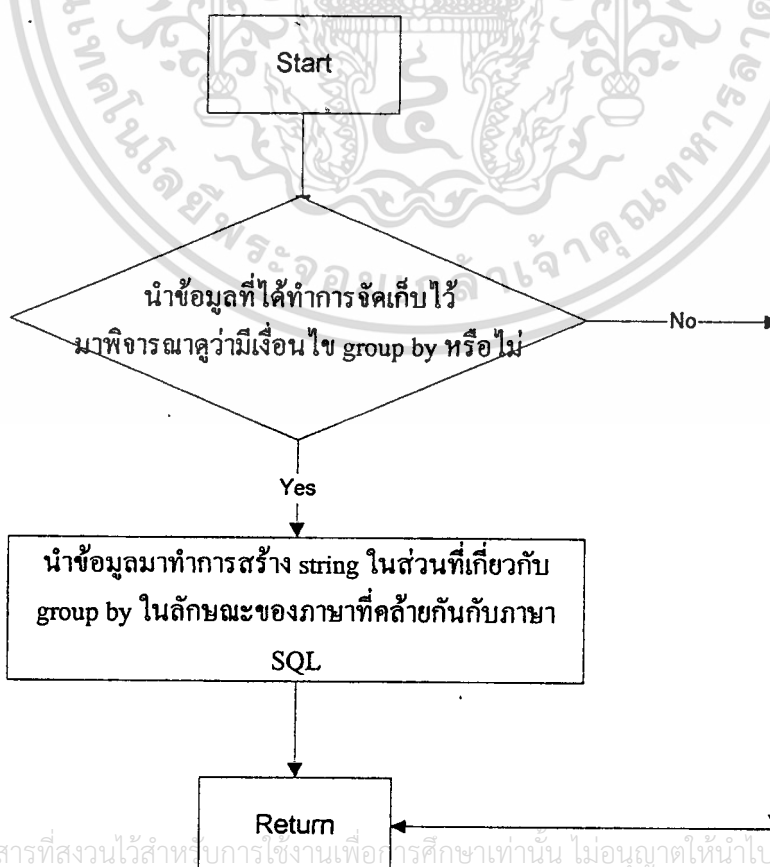


รูปที่ 4 - 17 แสดง Flow chart ของเมธอด *createSQL*





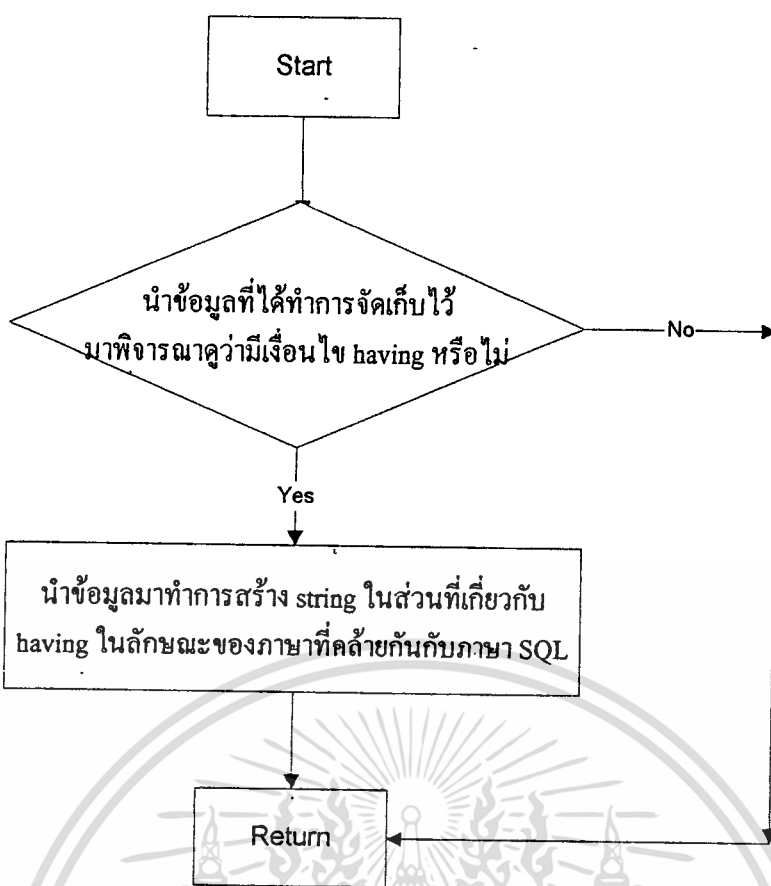
รูปที่ 4 - 19 แสดง Flow chart ของเมธอด *swhere*:



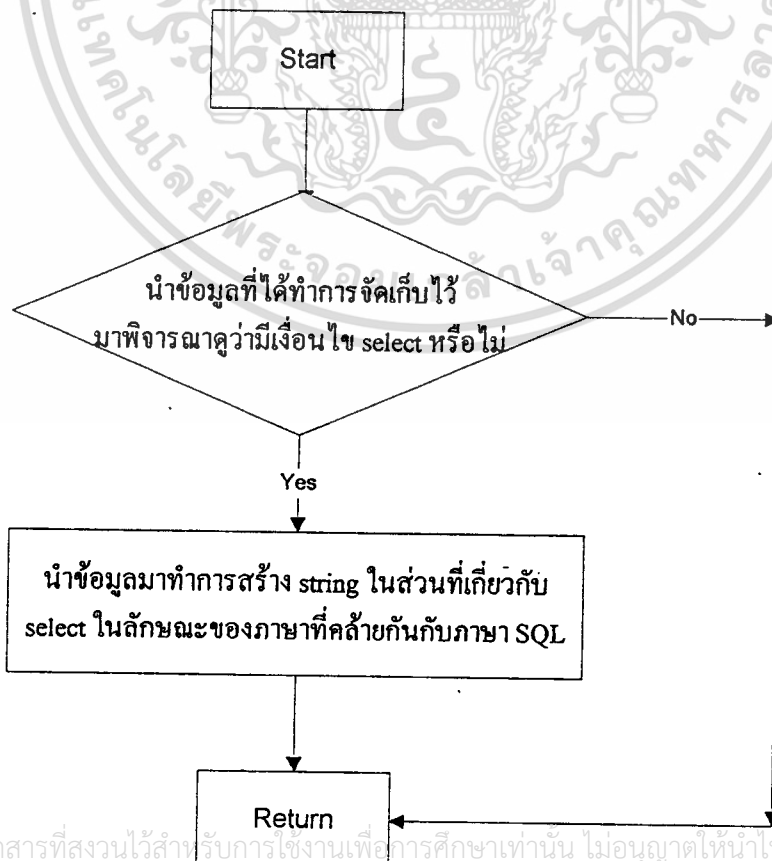
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 4 - 20 แสดง Flow chart ของเมธอด *sgroupBy*:



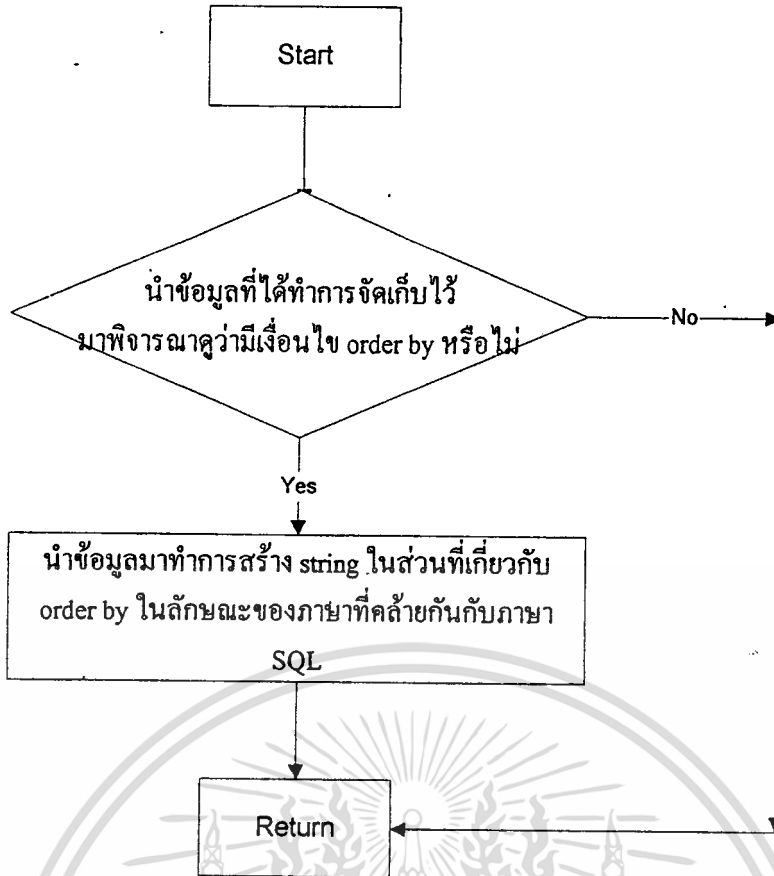
รูปที่ 4 - 21 แสดง Flow chart ของเมธอด shaving:



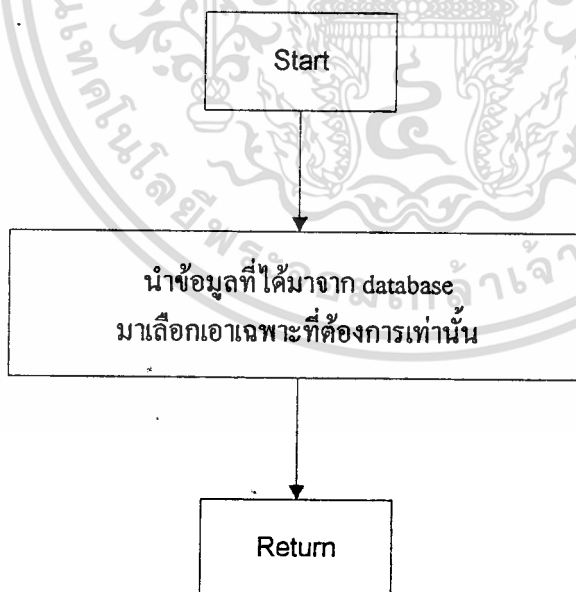
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น ถ้าหากพบให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

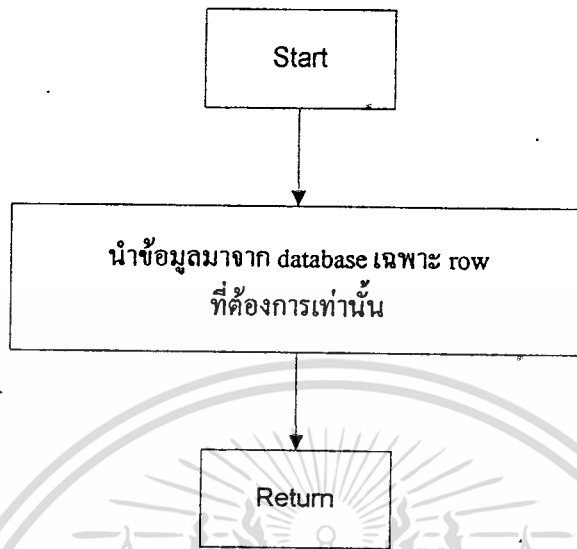
รูปที่ 4 - 22 แสดง Flow chart ของเมธอด sselect:



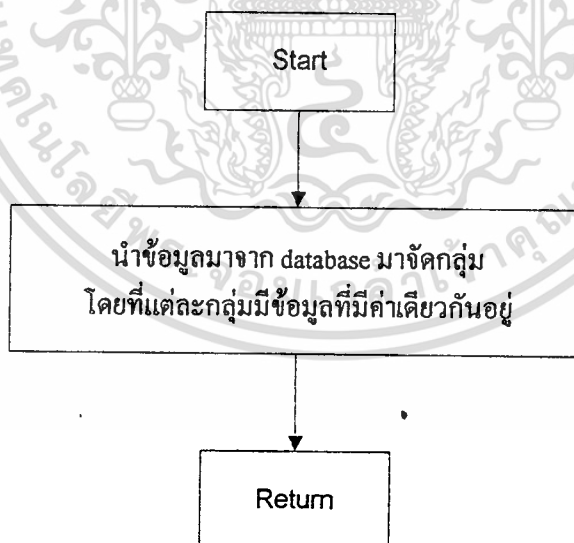
รูปที่ 4 - 23 แสดง Flow chart ของเมธอด *sorder*:



รูปที่ 4 - 24 แสดง Flow chart ของเมธอด *select*:

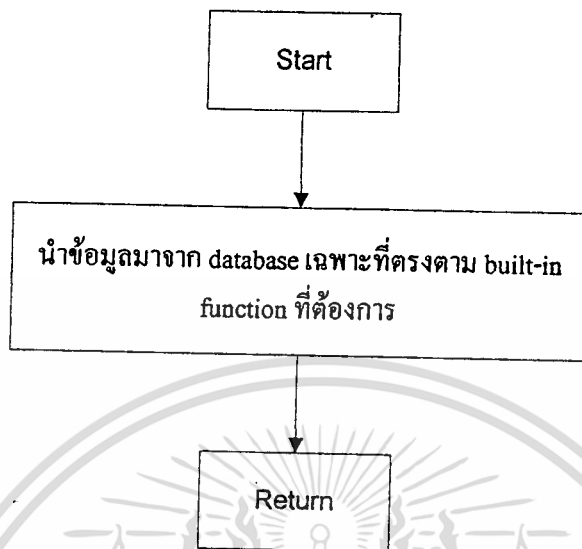


รูปที่ 4 - 25 แสดง Flow chart ของเมธอด where:

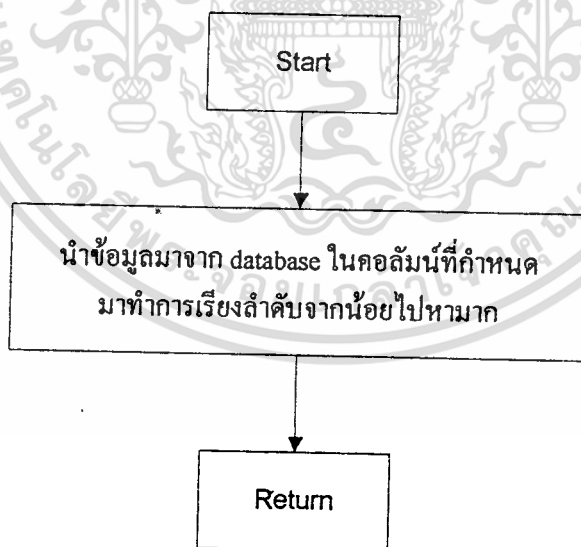


รูปที่ 4 - 26 แสดง Flow chart ของเมธอด groupBy:

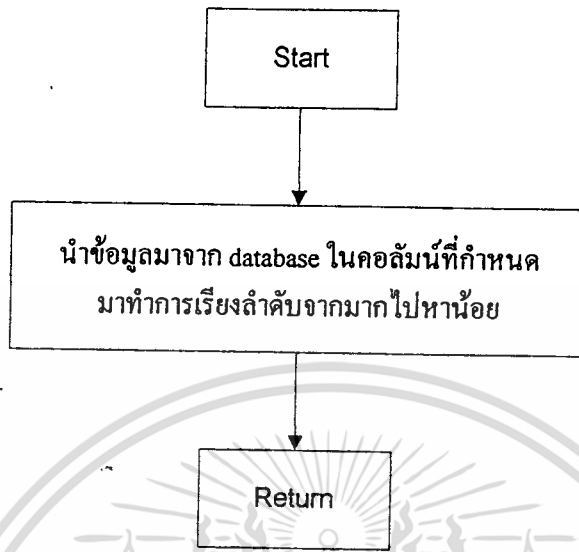
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



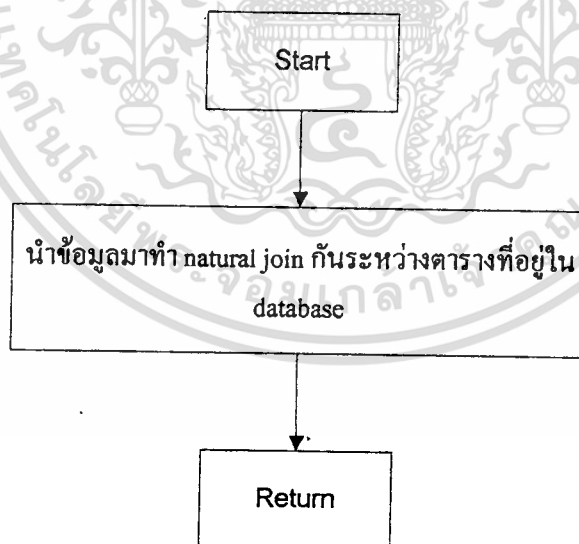
รูปที่ 4 - 27 แสดง Flow chart ของเมธอด having:



รูปที่ 4 - 28 แสดง Flow chart ของเมธอด orderAsc:

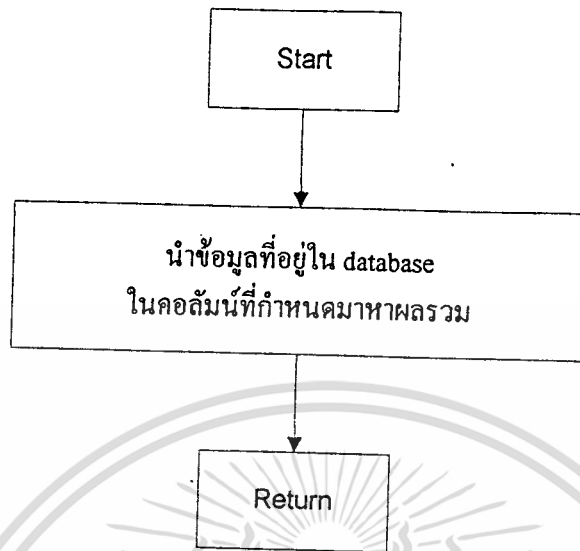


รูปที่ 4 - 29 แสดง Flow chart ของเมธอด *orderDec*:

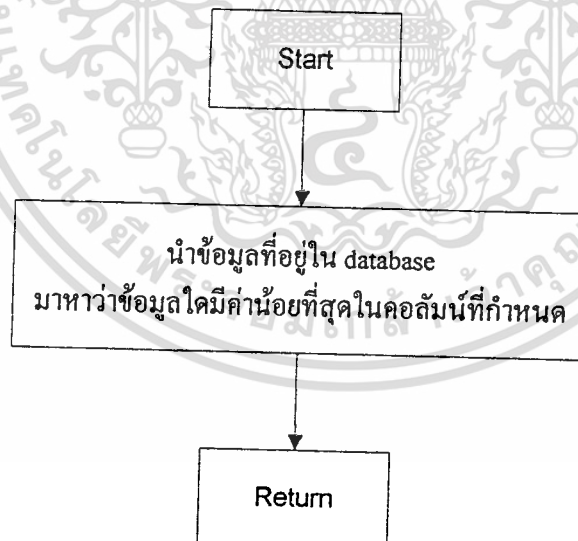


รูปที่ 4 - 30 แสดง Flow chart ของเมธอด *join*:

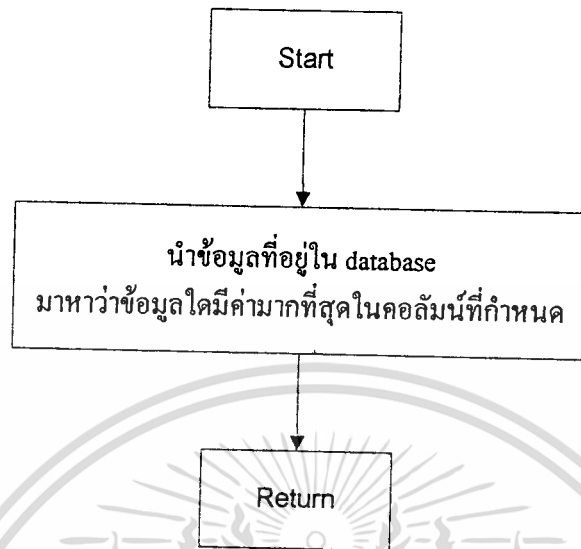
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



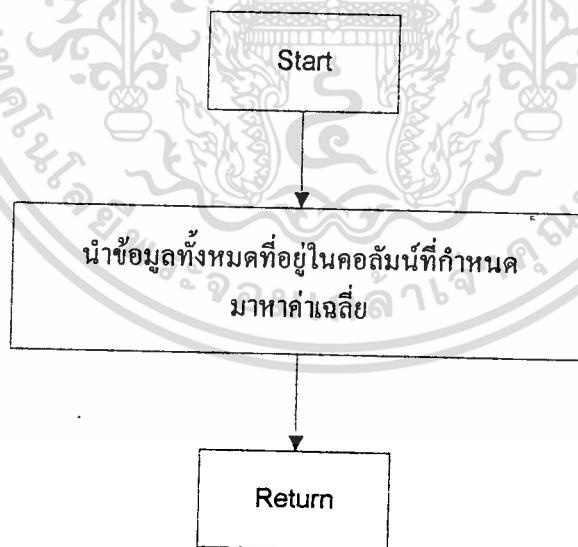
รูปที่ 4 - 31 แสดง Flow chart ของเมธอด sum:



รูปที่ 4 - 32 แสดง Flow chart ของเมธอด min:



รูปที่ 4 - 33 แสดง Flow chart ของเมธอด max:



รูปที่ 4 - 34 แสดง Flow chart ของเมธอด avg:

4.4.4 องค์ประกอบโดยรวมของแต่ละคลาส

RDB	VisualSQL	Data
name	run	select: aBlock
conditionTable	scan	where: aBlock
makeconditionTable	chkOrGroupSelect: aVisualSQL	groupBy: aColumn
getconditionTable	chkhaving: aVisualSQL	having: aBlock
putname: aString	chkOrhaving: aVisualSQL	orderAsc: aninteger
getname	chkwhere: aVisualSQL	orderDec: aninteger
runVisualSQL	chkOrwhere: aVisualSQL	join: aData
	createSQL	sum: aColumn
	sjoin: aString	min: aColumn
	swhere: aString	max: aColumn
	sgroupBy: aString	avg: aColumn
	shaving: aString	
	sselect: aString	
	sorder: aString	

รูปที่ 4 - 35 แสดงองค์ประกอบของคลาส RDB, VisualSQL และ Data

4.5 หลักการทำงานของโปรแกรม

4.5.1 ส่วนโปรแกรมที่นำหลักการทำงานมาจากโปรแกรมวิวอลเอสคิวแอล

จากรูปที่ 4 - 1 ส่วนโปรแกรมนำหลักการทำงานมาจากโปรแกรมวิวอลเอสคิวแอล จะรับ input เป็นภาษาในรูปแบบของวิวอลเอสคิวแอลจากผู้ใช้งาน และทำการเปลี่ยนเป็นข้อความของภาษาที่นำหลักการทำงานมาจากภาษาเอสคิวแอล โดยจะมีหลักการทำงานดังนี้

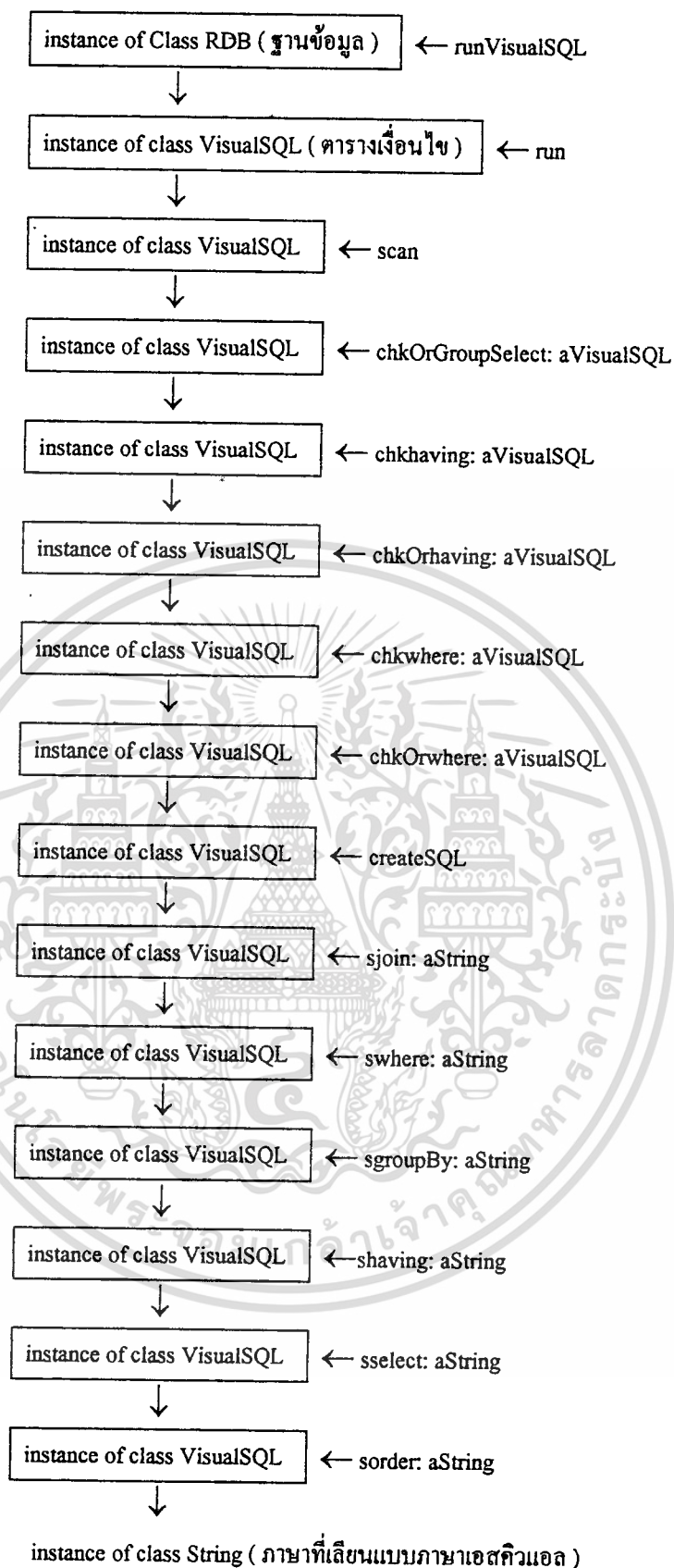
เมื่อผู้ใช้งานส่งเมสเซจ runVisualSQL เข้าไปออบเจกต์ของฐานข้อมูล โปรแกรมจะทำการสร้าง conditionTable ซึ่งเป็นออบเจกต์ภายในคลาส VisualSQL เพื่อให้ผู้ใช้งานป้อนข้อมูลเข้ามาในโปรแกรมในรูปแบบของภาษาในรูปแบบวิวอลเอสคิวแอล จากนั้นเมื่อผู้ใช้งานป้อนข้อมูลครบตามที่ต้องการ แล้วส่งเมสเซจ run เข้าไปในออบเจกต์ของ conditionTable โปรแกรมจะทำการนำภาษาในรูปแบบวิวอลเอสคิวแอลที่ได้มาทำการเปลี่ยนแปลงเป็นภาษาที่นำหลักการทำงานมาจากภาษาเอสคิวแอล โดยใช้เมธอด scan และ createSQL ตามลำดับ ดังรูปที่ 4 - 36

เมธอด scan จะทำการเปลี่ยน conditionTable ที่มีคีย์เป็นคอลัมน์ไปเป็นออบเจกต์ของคลาส VisualSQL ซึ่งมี คีย์ เป็น select , where , having , groupby , order โดยจะไปเรียกใช้เมธอดต่อไปนี้

1. chkOrGroupSelect: เป็นการนำตารางเงื่อนไขที่ได้มาดูเงื่อนไขที่เกี่ยวกับ order , groupby และ select มาใส่ไว้ในออบเจกต์ใหม่ที่มีคีย์เป็น order , groupby และ select ตามลำดับ
2. chkhaving: เป็นการนำตารางเงื่อนไขที่ได้มาดูเงื่อนไขที่เกี่ยวกับ having ที่เกี่ยวกับการ and มาใส่ไว้ในออบเจกต์ใหม่ที่มีคีย์เป็น having
3. chkOrhaving: เป็นการนำตารางเงื่อนไขที่ได้มาดูเงื่อนไขที่เกี่ยวกับ having ที่เกี่ยวกับการ or มาใส่ไว้ในออบเจกต์ใหม่ที่มีคีย์เป็น orhaving
4. chkwhere: เป็นการนำตารางเงื่อนไขที่ได้มาดูเงื่อนไขที่เกี่ยวกับ where ที่เกี่ยวกับการ and มาใส่ไว้ในออบเจกต์ใหม่ที่มีคีย์เป็น where
5. chkOrwhere: เป็นการนำตารางเงื่อนไขที่ได้มา ดูเงื่อนไขที่เกี่ยวกับ where ที่เกี่ยวกับการ or มาใส่ไว้ในออบเจกต์ใหม่ที่มีคีย์เป็น orwhere

ส่วนเมธอด createSQL จะทำการนำออบเจกต์ของคลาส VisualSQL ที่ได้มาทำการเปลี่ยนให้อยู่ในภาษาที่นำหลักการมาจากภาษาเอสคิวแอลอีกทีหนึ่ง โดยไปเรียกใช้เมธอดต่อไปนี้

1. sjoin: นำข้อมูลที่อยู่ในคีย์ join มาสร้างเป็นข้อความที่เกี่ยวกับการ join
2. swhere: นำข้อมูลที่อยู่ในคีย์ where orwhere มาสร้างเป็นข้อความที่เกี่ยวกับการ where
3. sgroupBy: นำข้อมูลที่อยู่ในคีย์ groupby มาสร้างเป็นข้อความที่เกี่ยวกับการ groupby
4. shaving: นำข้อมูลที่อยู่ในคีย์ having และ orhaving มาสร้าง เป็นข้อความที่เกี่ยวกับการ having
5. sselect: นำข้อมูลที่อยู่ในคีย์ select มาสร้างเป็นข้อความที่เกี่ยวกับการ select
6. sorder: นำข้อมูลที่อยู่ในคีย์ order มาสร้างเป็นข้อความที่เกี่ยวกับการ order



รูปที่ 4 - 36 แสดงขั้นตอนการทำงานของโปรแกรมที่นำหลักการการทำงานของโปรแกรมวิชาเอสคิวแอล

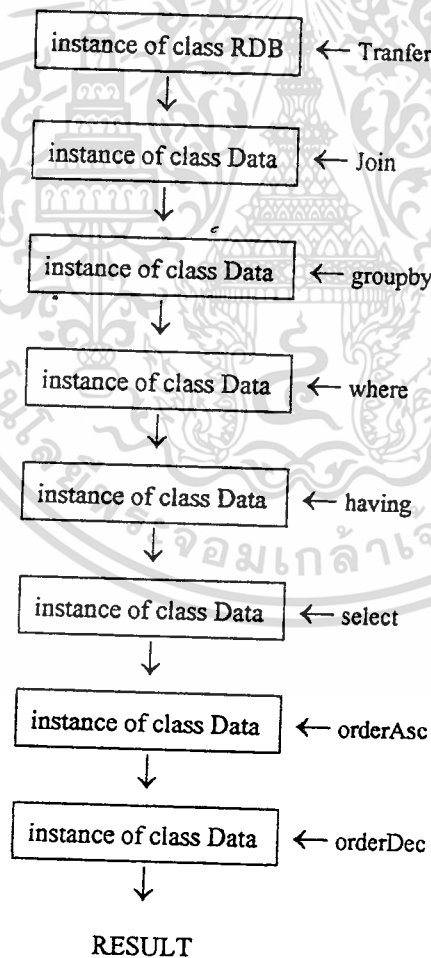
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.5.2 ส่วนโปรแกรมที่นำหลักการทำงานมาจากภาษาเอสคิวแอล

จากรูปที่ 4 - 1 ส่วน โปรแกรมที่นำหลักการทำงานมาจากภาษาเอสคิวแอล จะรับ input เป็นภาษาที่นำหลักการทำงานมาจากภาษาเอสคิวแอล และทำการค้นหาจากฐานข้อมูลแบบรีเลชันแนลที่ถูกออกแบบมาให้เป็นออบเจกต์ในคลาส RDB จะมีขั้นตอนการค้นหา โดยการเรียกใช้เมธอดต่อไปนี้ตามลำดับ

1. Tranfer
2. join
3. groupby
4. where
5. having
6. select
7. orderAsc
8. orderDec

จากนั้นจะทำการแสดงผลตามที่ผู้ใช้งานต้องการ ดังรูปที่ 4 - 37



รูปที่ 4 - 37 แสดงขั้นตอนการทำงานของโปรแกรมที่นำหลักการทำงานมาจากภาษาเอสคิวแอล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5

บทวิจารณ์และสรุป

5.1 วิจารณ์การเขียนโปรแกรมเชิงวัตถุ

5.1.1 การศึกษาทฤษฎี

การออกแบบระบบโดยใช้หลักการเชิงวัตถุแตกต่างไปจากการออกแบบระบบงานเดิม ซึ่งใช้วิธีการเก็บรวบรวมข้อมูลต่างๆ เป็นแฟ้มข้อมูล หรือฐานข้อมูล แยกกันกับโปรแกรมระบบงาน ในขณะที่หลักการเชิงวัตถุจะกำหนดสิ่งต่างๆ ที่อยู่ภายในขอบเขตของระบบงานเป็นออบเจกต์ และในออบเจกต์จะประกอบด้วยข้อมูลและเมธอด รวมกันอยู่ภายในออบเจกต์ การเรียกใช้เมธอดหรือการสั่งให้เกิดการกระทำตามที่กำหนดไว้ในเมธอด คือ การส่งเมสเสจไปยังออบเจกต์นั้นๆ ดังนั้นหากมีการเปลี่ยนแปลงวิธีการกระทำใดๆ ของออบเจกต์จะไม่มีผลกระทบต่อข้อมูลที่ส่งไปยังออบเจกต์นั้นและจะไม่มีผลกระทบต่อออบเจกต์อื่นๆ ด้วย นั่นคือ โปรแกรมของระบบไม่มีการเปลี่ยนแปลง ซึ่งนับว่าเป็นผลดีอย่างมากกับระบบงานเพราะโดยปกติแล้วพบว่า ระบบงานมักจะมีการเปลี่ยนแปลงวิธีการปฏิบัติอยู่เสมอๆ เมธอดสามารถใช้อธิบายถึงพฤติกรรมของออบเจกต์แต่ละออบเจกต์ได้ ออบเจกต์ที่มีพฤติกรรมแบบเดียวกันสามารถจัดรวมกลุ่มเป็นกลุ่มเดียวกันได้เรียกว่าคลาส ในระหว่างคลาสสามารถที่จะถ่ายทอดคุณสมบัติจากคลาสหนึ่งไปยังอีกคลาสหนึ่งได้ โดยเรียกคลาสที่ถ่ายทอดคุณสมบัติว่าซูเปอร์คลาส และเรียกคลาสที่ได้รับการถ่ายทอดคุณสมบัติว่าซับคลาสซึ่งทำให้การพัฒนากระบวนการทำงานทำได้ง่าย

5.1.2 การพัฒนาระบบงานด้วยโปรแกรมสมอลทอลล์

จากผลของการทำวิจัยเริ่มตั้งแต่การออกแบบระบบงานไปจนถึงการพัฒนาระบบงานด้วยโปรแกรมสมอลทอลล์ พบว่าสามารถที่จะวิเคราะห์และออกแบบระบบงานได้ดีเช่นเดียวกับวิธีการแบบเดิมหรืออาจจะดีกว่าเมื่อออกแบบสำหรับระบบงานบางประเภทเสียด้วยซ้ำ เช่นระบบงานที่มีข้อมูลซับซ้อนมากๆ หรือระบบงานที่มีการเปลี่ยนแปลงวิธีการทำบ่อยๆ ระบบงานที่ได้จากการวิจัยสามารถทำงานได้ดีเช่นเดียวกับวิธีการแบบเดิม ในคำของผู้ใช้ก็ไม่ได้รู้สึกว่าอะไรเปลี่ยนแปลง ตรงกันข้ามกับวิธีการแบบเดิมที่ส่วนมากแล้วเมื่อมีการแก้ไข ผู้ใช้ต้องมีการอบรมใหม่ ในระยะแรกๆ ของการทำอาจจะรู้สึกติดขัดบ้างเพราะว่ายังไม่คุ้นเคยกับแนวความคิดแบบใหม่ แต่เมื่อเริ่มคุ้นเคยจะพบว่าสามารถทำได้ง่ายและสะดวกกว่าวิธีการแบบเดิม การสร้างความคุ้นเคยกับแนวความคิดแบบใหม่นี้ก็ใช้เวลาไม่มากนักเพราะหลักการเข้ากับชีวิตประจำวันของเราที่มักจะมองสิ่งต่างๆ รอบตัวเราเป็นออบเจกต์ๆ หนึ่ง

5.1.3 ภาษาที่เหมาะสมจะใช้กับฐานข้อมูลสำหรับงานทางด้านสารสนเทศ

ภาษาที่ใช้กับฐานข้อมูล สำหรับงานทางด้านสารสนเทศในปัจจุบันนี้ ส่วนหนึ่งได้รับอิทธิพลมาจากโปรแกรมสมอลทอลล์ (เช่น ภาษา Opal ของ Gemstone OODBMS ที่มีชื่อเสียง) และอีกส่วนหนึ่งได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รับอิทธิพลมาจากการต่อเติมภาษา C (เช่น ภาษา COP และ TDL ของ Vbase OODBMS) ภาษาทั้งสองนั้นเป็น record-at-a-time คือ การอ่าน (read) 1 ครั้ง จะได้ข้อมูล 1 เรคคอร์ด และการเขียน (write) 1 ครั้ง จะได้ 1 เรคคอร์ด ดังนั้นเมื่อมาประยุกต์กับออบเจกต์ การปฏิบัติการก็จะยังปฏิบัติตามเซตของออบเจกต์ ไม่ได้หรือไม่ดีพอ

ส่วนภาษาที่ใช้กับฐานข้อมูลรีเลชันแนล เช่น เอสคิวแอล มีการปฏิบัติการเป็นเซต นอกจากจะให้ประสิทธิภาพที่สูงกว่าแล้ว ยังเหมาะสมสำหรับการทำงานในระบบฐานข้อมูลแบบกระจาย (Distributed Database System) อีกด้วย เนื่องจากไม่ต้องส่งคำสั่ง อ่าน และ เขียน จำนวนมากไปมาในโครงข่ายสื่อสารนั่นเอง

สำหรับหลักการของฐานข้อมูลสำหรับงานทางด้านสารสนเทศนั้น ฐานข้อมูลมีไว้เพื่อรองรับงานประมวลผลตามความต้องการของผู้ใช้งาน จึงจำเป็นที่จะต้องมีความเป็นเอกประสงค์ในระดับหนึ่ง การนำเมธอดหรือโปรแกรมย่อย ซึ่งทำงานเฉพาะกิจกับออบเจกต์ฝังไว้ร่วมกับโครงสร้างข้อมูล เท่ากับเป็นการนำโปรแกรมไปผูกไว้กับตัวข้อมูลโดยตรง ทำให้ฐานข้อมูลซึ่งปกติมีแต่ข้อมูลก็ใหญ่และซับซ้อนมากอยู่แล้วยิ่งซับซ้อนขึ้นไปอีก นอกจากจะทำให้มีข้อผิดพลาดเพิ่มขึ้นแล้ว มุมมองที่ผู้ใช้งานออบเจกต์จากงานที่ต่างกันก็อาจแตกต่างกัน ทำให้มีเมธอดเป็นจำนวนมาก ซึ่งอาจไม่เกี่ยวข้องกับงานของเราถูกรวมไว้กับออบเจกต์ เป็นการเพิ่มภาระให้แก่ฐานข้อมูลโดยไม่จำเป็น จะเห็นได้ชัดว่าหลักการดังกล่าวนี้ตรงข้ามกับวัตถุประสงค์ของการเขียนโปรแกรมเชิงวัตถุ เพราะ โปรแกรมถูกเขียนขึ้นมาเพื่อวัตถุประสงค์เฉพาะงาน

5.1.4 ปัญหาที่เกิดจากการเขียนโปรแกรมสมอลทอล์ค

5.1.4.1 ปัญหาที่เกิดจากการกำหนดตัวแปร

A := OrderedCollection new.

A add: 1;

add: 2;

add: 3.

B := A.

เมื่อเราใช้งาน โปรแกรมข้างบนนี้ ตัวแปร A จะชี้ไปยังข้อมูล (1 2 3) และเมื่อใช้คำสั่ง B := A. จะเป็นการกำหนดให้ ตัวแปร B ชี้ไปที่เดียวกันกับตัวแปร A คือ (1 2 3) และเมื่อใช้คำสั่ง A add: 4. ไปอีก ผลลัพธ์ที่ได้ เป็นดังนี้

A = (1 2 3 4)

B = (1 2 3 4)

ซึ่งค่าของตัวแปร B ที่เปลี่ยนไปเนื่องจากตัวแปร B ชี้ไปที่ตำแหน่งของข้อมูลที่เดียวกับตัวแปร

A

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การแก้ไขทำได้โดยการเปลี่ยนมาทำการสร้างตัวแปร B ขึ้นมาใหม่แล้วทำการใส่ค่าของตัวแปร A ที่ละค่าให้กับตัวแปร B ดังโปรแกรมข้างล่างนี้

```
A := OrderdeCollection new.
```

```
B := OrderedCollection new.
```

```
A add: 1;
```

```
add: 2;
```

```
add: 3.
```

```
A do: [:c | B add: c. ].
```

จากนั้นเมื่อทำคำสั่ง A add: 4. ผลลัพธ์ที่ได้จะเป็นดังนี้

```
A = ( 1 2 3 4 )
```

```
B = ( 1 2 3 )
```

5.1.4.2 การแสดงผล string ในคลาส Pane

คลาส Pane เป็นคลาสที่ใช้สำหรับการแสดงผลออกมาที่หน้าจอคอมพิวเตอร์ ถ้าเรามีข้อมูลที่มีเครื่องหมาย apostrophe อยู่ภายใน string อีกทีหนึ่ง

เช่น abcd'efg

สามารถทำได้โดยใช้คำสั่ง A := 'abcd''efg'.

ค่าของ ตัวแปร A คือ abcd'efg

แต่เมื่อทำการแสดงผลที่หน้าจอคอมพิวเตอร์ในคลาส Pane จะได้ข้อมูล abcd''efg ปรากฏที่หน้าจอ แทนที่จะเป็น abcd'efg

5.1.4.3 ปัญหาที่เกิดจากการไม่รู้ว่ามีวิธีฟเวอร์อยู่ภายในคลาสอะไร และมีเมธอดอะไรบ้าง

การแก้ไขทำได้โดยใช้คำสั่ง inspect ดูข้อมูลและคลาสของออบเจกต์ที่เราสนใจ ส่วนเมธอดนั้นสามารถดูได้จากใช้คำสั่ง Class Browser

5.1.4.4 ปัญหาที่เกิดจากการแก้ไขข้อผิดพลาดของโปรแกรม

โปรแกรมสมอลทอล์คนั้นเป็นโปรแกรมที่เกิดขึ้นมานานแล้ว การแก้ไขข้อผิดพลาดที่เกิดขึ้นจากการเขียนโปรแกรมนั้นสามารถทำได้ยาก เนื่องจากการบอกจุดที่ผิดพลาดเป็นเมธอด แต่ไม่สามารถทราบได้ว่าโปรแกรมตรงส่วนไหนในเมธอดนั้นๆ เกิดการผิดพลาดขึ้น ดังนั้นจึงต้องอาศัยประสบการณ์ของผู้เขียนโปรแกรมเอง

5.1.5 ปัญหาที่เกิดขึ้นกับ Inheritance

Inheritance เป็นคุณสมบัติที่มีความสำคัญและมีประโยชน์อย่างมาก แต่ว่าก็อาจจะทำให้มีความซับซ้อนเพิ่มขึ้นในความสัมพันธ์ระหว่างคลาส ในการใช้คุณสมบัติ Inheritance ชั้นคลาสสามารถเข้าถึงและอ้างถึงแอตทริบิวต์ของซูเปอร์คลาสของมัน และการทำงานของชั้นคลาสสามารถที่จะเข้าไปเรียกใช้การทำงานในซูเปอร์คลาสของมันได้

ปัญหาที่เกิดขึ้น คือ แทนที่จะรวมแอตทริบิวต์ และ การทำงาน ไว้ด้วยกัน class hierarchy จะแยกออกจากกัน นอกจากนั้น Inheritance ยังจะทำให้เกิดความสัมพันธ์กันระหว่างซูเปอร์คลาส และชั้นคลาส ความเปลี่ยนแปลงที่เกิดขึ้นกับซูเปอร์คลาสจะจัดการได้ยาก เพราะว่า การเปลี่ยนแปลงภายในซูเปอร์คลาสจะมีผลกระทบไปถึงทุกชั้นคลาสซึ่งอยู่ภายใต้ซูเปอร์คลาสเดียวกัน มีคนเคยกล่าวไว้ว่า “Inheritance นั้นสามารถซ่อนรายละเอียดต่างๆ แม้กระทั่งความผิดพลาดที่เกิดขึ้นก็ค้นพบได้ยากด้วย ”

Inheritance บางครั้งอาจจะไม่มีประโยชน์ ถ้าหากว่าซูเปอร์คลาสนั้นถูกสร้างมาเพียงแต่เป็นส่วนประกอบ ไม่ได้มีจุดประสงค์เฉพาะเจาะจงไป ตัวอย่างเช่น คลาส Course อาจจะสร้างเพื่อให้ครอบคลุมคลาส Student และคลาส Lecturer แต่ว่าจริงๆ แล้ว Student และ Lecturer ไม่ได้มีลักษณะเฉพาะเจาะจงลงไปจาก Course ความผิดพลาดที่เกิดขึ้นนี้อาจจะมีสาเหตุมาจากความเข้าใจผิดในความสัมพันธ์ “has - a “ แทนที่จะเป็นความสัมพันธ์ “is - a “ ความสัมพันธ์ “has - a “ จะเกิดขึ้นจากการที่เอนิตีหนึ่งมีส่วนประกอบเป็นอีกเอนิตีหนึ่ง ตัวอย่างเช่น เครื่องบินมีปีก แต่ถ้าเป็นความสัมพันธ์ “is - a “ จะเกิดขึ้นจากการที่เอนิตีหนึ่งเป็นลักษณะที่พิเศษลงไปกว่าอีกเอนิตีหนึ่ง เช่น พนักงานเป็นบุคคลคนหนึ่ง

5.1.6 Reuse Of Software

การนำกลับมาใช้ใหม่เป็นการนำส่วนประกอบที่มีอยู่แล้วในระบบมาใช้ การนำกลับมาใช้ใหม่นี้เป็นส่วนสำคัญที่ในวิศวกรรมซอฟต์แวร์ และสามารถเพิ่มคุณภาพของซอฟต์แวร์ และช่วยลดค่าใช้จ่ายในการพัฒนาระบบ ผลที่ได้ก็คือ การนำกลับมาใช้ใหม่จะช่วยเพิ่มผลผลิต การที่เราเสียเวลาส่วนหนึ่งในการค้นหาส่วนประกอบที่สามารถนำมาใช้ใหม่ได้นั้น บางครั้งจะคุ้มค่ากว่าการที่จะเสียเวลาสร้างและทดสอบส่วนประกอบนั้นขึ้นมาใหม่

ที่ผ่านมา การนำกลับมาใช้ใหม่นี้จะเป็นการ copy source code หรือโปรแกรมย่อย มาจาก Library ซึ่งเป็นการนำมาใช้ใหม่ในขั้นตอนการนำไปใช้ แต่การนำมาใช้ในขั้นตอนการนำไปใช้นี้ อาจจะหายไป ในขั้นตอนเริ่มต้นของการพัฒนาระบบ ผู้ที่ออกแบบจะสนใจในส่วนประกอบที่มีอยู่แล้ว และทำการแยกแยะระดับและส่วนประกอบเหล่านั้น ซึ่งจะแยกตามลักษณะการทำงานของแต่ละส่วนประกอบ

แต่ก็มีเหตุผลหลายประการที่ทำให้การออกแบบโดยใช้วิธีการนำมาใช้ใหม่นี้ทำได้ยาก ปัญหาส่วนใหญ่ก็คือ การที่จะแบ่งแยกส่วนประกอบต่างๆ ของซอฟต์แวร์ และการค้นหาส่วนประกอบที่สามารถนำมาใช้ใหม่ได้นั้น ทำได้ยาก และบางครั้งในการที่จะพิจารณาค้นหาส่วนประกอบต่างๆ เหล่านี้ การที่จะทำความเข้าใจกับส่วนประกอบต่างๆ เหล่านี้ จะใช้เวลาในการพิจารณาอีก เหตุผลหนึ่งก็คือ ในระบบซอฟต์แวร์บางครั้งจะไม่มีส่วนประกอบที่สามารถจะนำมาใช้ใหม่ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การที่จะออกแบบให้ส่วนประกอบที่สร้างขึ้นใหม่นั้นสามารถนำไปใช้ได้ด้วยในอนาคตนั้น อาจจะทำให้เกิดความยุ่งยากในการออกแบบ ซึ่งจะต้องใช้ทั้งเวลาและความสามารถในการที่จะพัฒนาให้ระบบนั้น มีความคงทนและมีความสามารถที่จะนำมาใช้ใหม่ได้ในอนาคต

5.2 สรุปผลการวิจัย

การวิจัยตามโครงการนี้ได้ทำการวิจัยเป็นที่น่าพอใจ ตรงตามเป้าหมายที่ได้วางไว้

5.3 ข้อเสนอแนะ

งานวิจัยนี้มุ่งที่จะทดลองใช้หลักการเชิงวัตถุมาออกแบบระบบงานแทนวิธีการแบบเดิมๆ เพื่อจะศึกษาหาข้อสรุปผลการทำงาน ดังนั้นจึงเลือกใช้โปรแกรมสมอลทอล์ค ซึ่งเป็นโปรแกรมเชิงวัตถุอย่างแท้จริง แต่ว่าในปัจจุบันมีซอฟต์แวร์เชิงวัตถุใหม่เกิดขึ้นมากมาย ซึ่งใช้งานได้ง่ายและสะดวกกว่า จึงขอเสนอแนะให้ผู้สนใจได้ทดลองใช้โปรแกรมเชิงวัตถุใหม่ๆ เหล่านั้น



ภาคผนวก ก

คู่มือการใช้งานโปรแกรม

1. การติดตั้งโปรแกรม

1.1 การติดตั้งโปรแกรม smalltalk

ทำการ copy ข้อมูลจาก แผ่นดิสก์ที่ได้เตรียมไว้ มายัง C:\SMALTALK

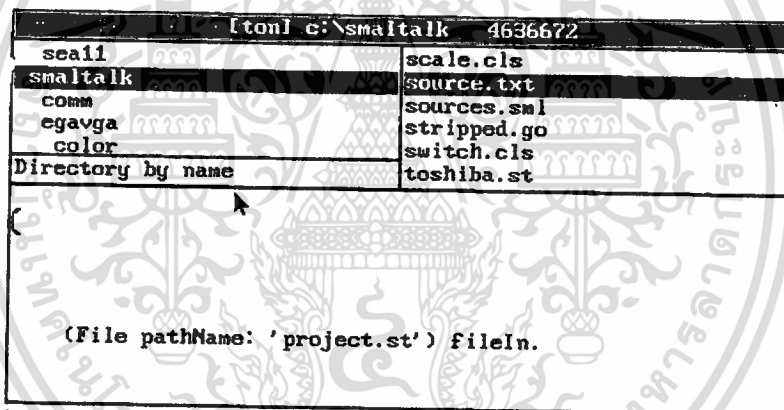
จากนั้นให้ใช้คำสั่ง C:\SMALTALK\I.EXE จะเป็นการเรียกใช้งานโปรแกรมภาษา smalltalk

1.2 การติดตั้งโปรแกรมที่เลียนแบบการทำงานของโปรแกรม VisualSQL

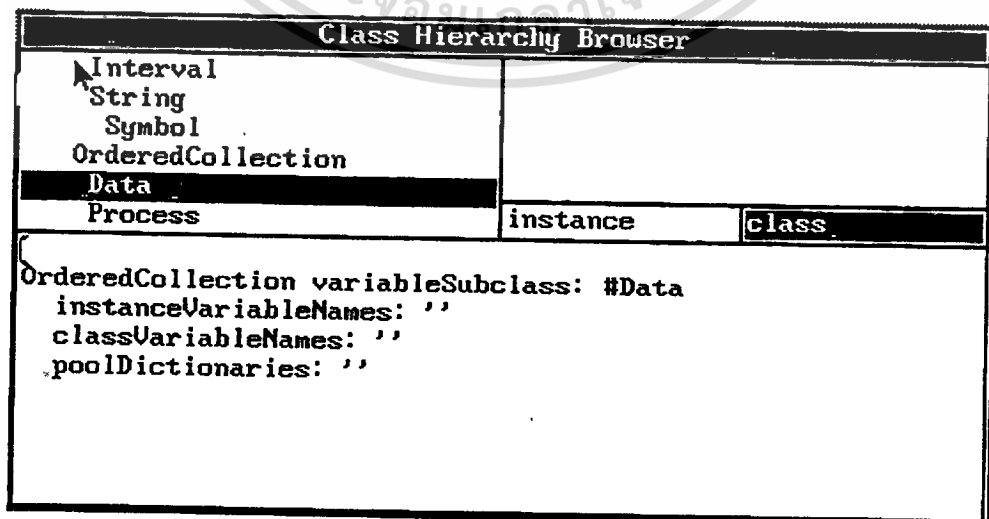
เมื่อเข้ามายังโปรแกรมภาษา smalltalk แล้ว จะทำการติดตั้งส่วนโปรแกรมที่เลียนแบบการทำงานของโปรแกรม VisualSQL โดยใช้คำสั่ง

(File pathName: 'project.st') fileIn. ดังรูปที่ 1

ซึ่งจะไปสร้างคลาสใหม่ให้กับระบบ คือ คลาส Data, RDB และ VisualSQL ดังรูปที่ 2 และ 3



รูปที่ 1 แสดงส่วนการติดตั้งโปรแกรมเลียนแบบการทำงานของโปรแกรม VisualSQL



รูปที่ 2 แสดงคลาส Data ที่ได้สร้างขึ้นใหม่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Class Hierarchy Browser		
Dictionary		
IdentityDictionary		
MethodDictionary		
RDB		
VisualSQL		
SystemDictionary	instance	class
RDB variableSubclass: #VisualSQL instanceVariableNames: '' classVariableNames: '' poolDictionaries: ''		

รูปที่ 3 แสดงคลาส RDB และ VisualSQL ที่ได้สร้างขึ้นมาใหม่

2. การใช้งานโปรแกรมเขียนแบบการทำงานของโปรแกรม VisualSQL

เมื่อได้ทำการติดตั้งโปรแกรมเรียบร้อยแล้ว จะทำการใช้งานโปรแกรมต่อไปดังนี้

2.1 เมื่อต้องการ join หลายตารางเข้าด้วยกัน

เช่น President join: PresMarriage

ผู้ใช้งานทำการสร้าง key ชื่อว่า join ให้กับตาราง President โดยมี instance ที่มีข้อมูลดังรูปที่ 4 และสามารถใส่ข้อมูลที่เป็นเงื่อนไขได้ทั้งในตาราง President และ PresMarriage ดังรูปที่ 5

TABLE NAME : President	
'BirthYr'	PresMarriage
'DeathAge'	
'join'	
'Party'	
'PresName'	
'StateBorn'	
'YrsServ'	

รูปที่ 4 แสดงส่วนการ join ระหว่างตาราง President กับ PresMarriage

TABLE NAME : PresMarriage	
'MarYear'	
'NrChildren'	
'PrAge'	
'PresName'	
'SpAge'	
'SpouseName'	

รูปที่ 5 แสดงส่วนการรับข้อมูลจากผู้ใช้งาน สำหรับตาราง PresMarriage

2.2 เมื่อต้องการ groupby

เช่น groupBy: 'PresName'

ทำการใส่ข้อมูล groupby ลงไปใน key ที่มีชื่อว่า PresName ดังรูปที่ 6

TABLE NAME : President	
'BirthYr'	#('groupby')
'DeathAge'	
'join'	
'Party'	
'PresName'	
'StateBorn'	
'YrsServ'	

รูปที่ 6 แสดงการ groupby column PresName

2.3 เมื่อต้องการ where

เช่น where: [:c| (((c at: 'BirthYr') > 1800) or: [((c at: 'DeathAge') > 60))])
and: [((c at: 'YrsServ') < 8)]]

จะทำการใส่เงื่อนไขทั้ง 3 อย่างลงไปใน column ที่ต้องการ จากนั้นจะทำการสร้าง key ที่มีชื่อว่า orwhere ขึ้นมาเพื่อใส่ชื่อ column ที่ต้องการทำการ or กัน ดังรูปที่ 7

TABLE NAME : President	
'BirthYr'	# (' > 1800 ')
'DeathAge'	# (' > 60 ')
'Party'	
'PresName'	
'StateBorn'	
'YrsServ'	# (' < 8 ')
'orwhere'	# ('BirthYr' 'DeathAge')

รูปที่ 7 แสดงการป้อนข้อมูลที่เกี่ยวข้องกับการ where

2.4 เมื่อต้องการ having

เช่น having: [:c| (((c max: 'BirthYr') > 1800) or: [((c min: 'DeathAge') > 60))])
 and: [((c sum: 'YrsServ') < 8)]]

จะทำการใส่เงื่อนไขทั้ง 3 อย่างลงไปใน key ที่สร้างขึ้นใหม่ ชื่อว่า max min sum ตามลำดับ จากนั้นจะทำการสร้าง key ที่มีชื่อว่า orwhere ขึ้นมาเพื่อใส่ชื่อ column ที่ต้องการทำการ or กัน ดังรูปที่ 8

TABLE NAME : President	
'BirthYr'	
'DeathAge'	
'Party'	
'PresName'	
'StateBorn'	
'YrsServ'	
'max'	# ('BirthYr' ' > 1800')
'min'	# ('DeathAge' ' > 60')
'sum'	# ('YrsServ' ' < 8')
'orhaving'	# ('max' 'min')

รูปที่ 8 แสดงการป้อนข้อมูลที่เกี่ยวข้องกับการ having

2.5 เมื่อต้องการ select

เช่น select: [:c | (c choose: #('PresName' 'StateBorn')), (c min: 'BIRTH_YR'),
(c sum: 'YrsServ')]

ทำการใส่ข้อมูล 'p' ลงไปใน key ที่ต้องการ select ดังรูปที่ 9

TABLE NAME : President	
'BirthYr'	
'DeathAge'	
'Party'	
'PresName'	# ('p')
'StateBorn'	# ('p')
'YrsServ'	
'min'	# ('p' 'BirthYr')
'sum'	# ('p' 'YrsServ')

รูปที่ 9 แสดงการป้อนข้อมูลเกี่ยวกับการ select

2.6 เมื่อต้องการ order by

ถ้าต้องการจะ Ascending order column BirthYr ก่อนอื่นเราจะต้องทำการ select column ที่ต้องการก่อน แล้วทำการใส่ข้อมูล 'inc' ลงไปใน column ที่ต้องการ ดังรูปที่ 10

ถ้าต้องการจะ Descending order column BirthYr ก่อนอื่นเราจะต้องทำการ select column ที่ต้องการก่อน แล้วทำการใส่ข้อมูล 'dec' ลงไปใน column ที่ต้องการ ดังรูปที่ 11

TABLE NAME : President	
'BirthYr'	# ('p' ' inc')
'DeathAge'	
'Party'	
'PresName'	
'StateBorn'	
'YrsServ'	

รูปที่ 10 แสดงการป้อนข้อมูลเกี่ยวกับการ Ascending order

TABLE NAME : President	
'BirthYr'	#('p' 'dec')
'DeathAge'	
'Party'	
'PresName'	
'StateBorn'	
'YrsServ'	

รูปที่ 11 แสดงการป้อนข้อมูลเกี่ยวกับการ *Desending Order*

3. การสร้างฐานข้อมูลแบบรีเลชันนอลโดยใช้ออบเจก

PRES_NAME	BIRTH_YR	YRS_SERV	DEATH_AGE	PARTY	STATE_BORN
Washington G	1732	7	67	Federalist	Virginia
Adams J	1735	4	90	Federalist	Massachusetts
Jefferson T	1743	8	83	Demo-Rep	Virginia
Madison J	1751	8	85	Demo-Rep	Virginia

รูปที่ 12 ฐานข้อมูลแบบรีเลชันแนล

จากรูปที่ 12 เป็นตัวอย่างฐานข้อมูลที่จะนำมาใช้ในการอธิบายการสร้างในรูปแบบของออบเจก และมีการขั้นตอนการสร้างดังนี้

3.1 มีการกำหนด GlobalVariable ของ String ต่างๆ ดังนี้

ในส่วนของ column PresName := 'PresName'.

BirthYr := 'BirthYr'.

YrsServ := 'YrsServ'.

DeathAge := 'DeathAge'.

Party := 'Party'.

StateBorn:= 'StateBorn'.

ในส่วนของ Party

Federalist := 'Federalist'.

DemoRep := 'Demo-Rep'.

ในส่วนของ StateBorn

Virginia := 'Virginia'.

Massachusetts := 'Massachusetts'.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2 มีการกำหนด GlobalVariable อื่นๆ ที่จำเป็นเนื่องจาก ข้อมูลที่ได้มีช่องว่างระหว่างคำ ทำให้โปรแกรม smalltalk มองเห็นเป็น 2 ออบเจกต์ ดังนี้

```
WashingtonG := 'Washington G'.
```

```
'AdamsJ := 'Adams J'.
```

```
JeffersonT := 'Jefferson T'.
```

```
MadisonJ := 'Madison J'.
```

3.3 ทำการสร้างตาราง President ขึ้นมา โดยให้เป็นออบเจกต์ของคลาส Table และภายในมีข้อมูลของคลาส Data ดังนี้

```
President := Table new.
```

```
President putname: 'PRESIDENT'.
```

```
President makeconditionTable.
```

```
A := Data new.
```

```
A add: 'Washington G';
```

```
add: 'Adams J';
```

```
add: 'Jefferson T';
```

```
add: 'Madison J'.
```

```
President at: 'PresName' put: A.
```

```
A := Data new.
```

```
A add: 1732;
```

```
add: 1735;
```

```
add: 1743;
```

```
add: 1751.
```

```
President at: 'BirthYr' put: A.
```

```
A := Data new.
```

```
A add: 7;
```

```
add: 4;
```

```
add: 8;
```

```
add: 8.
```

```
President at: 'YrsServ' put: A.
```

A := Data new.

A add: 67;

add: 90;

add: 83;

add: 85.

President at: 'DeathAge' put: A.

A := Data new.

A add: 'Federalist';

add: 'Federalist';

add: 'Demo-Rep';

add: 'Demo-Rep'.

President at: 'Party' put: A.

A := Data new.

A add: 'Virginia';

add: 'Massachusetts';

add: 'Virginia';

add: 'Virginia'.

President at: 'StateBorn' put: A.

3. ตัวอย่างการใช้โปรแกรมเขียนแบบการทำงานของโปรแกรม VisualSQL

Question 10.08

ในหนังสือ INTRODUCTION TO IBM SQL เขียนโดย G.M NYSSSEN , E . D FALKENBERG

For each president with more than three children , list their name, their birth year and the number of children from all marriages. Order by number of children in descending order

จาก ตัวอย่างที่ 10.08 สามารถใช้งาน โปรแกรม ตามตัวอย่างรูปที่ 13 และ 14

TABLE NAME : President	
'BirthYr'	
'DeathAge'	
'join'	PresMarriage
'min'	#('p' 'BirthYr')
'Party'	
'PresName'	#('p' 'groupby')
'StateBorn'	
'YrsServ'	

รูปที่ 13 แสดงส่วนการรับข้อมูลตามตัวอย่าง 10.08 ของตาราง President

TABLE NAME : PresMarriage	
'MarYear'	
'NrChildren'	
'PrAge'	
'PresName'	
'SpAge'	
'SpouseName'	
'sum'	#('p' 'NrChildren' '>3' 'dec')

รูปที่ 14 แสดงส่วนการรับข้อมูลตามตัวอย่าง 10.08 ของตาราง PresMarriage

เมื่อใช้เมธอด run จะทำการสร้างข้อความในรูปของภาษาที่เลียนแบบการทำงานของภาษาเอสคิวแอล ดังนี้

```
(((( President join: PresMarriage )
```

```
groupBy: PRES_NAME )
```

```
having: [ :c | ((c sum: NR_CHILDREN ) > 3 ) ] )
```

```
select: [ :c | ( c choose: # ( PRES_NAME ) ), ( c min: BIRTH_YR ),
          ( c sum: NR_CHILDREN ) ] )
```

```
orderDec: 3
```

กิตติกรรมประกาศ

ผู้จัดทำขอขอบคุณ รศ.ดร. ศุภมิตร จิตตะขุโสธร เป็นอย่างสูง ที่ท่านได้ให้ความรู้พื้นฐานทางด้าน
ฐานข้อมูล ตลอดจนถึงคำปรึกษา และ คำแนะนำต่างๆ ในการแก้ไขปัญหที่เกิดขึ้นตลอดการทำงาน

ขอขอบคุณ พี่พิทักษ์ พี่บัณฑิต พี่เนวิน ที่ได้ให้ความช่วยเหลือในด้านต่างๆ โดยเฉพาะอย่างยิ่ง
ช่วยให้ผู้จัดทำสามารถติดต่อกับอาจารย์ได้สะดวก

ขอขอบคุณ พี่นพพร ที่เอื้อเฟื้อโปรแกรมภาษาสมอลทอล์ค สำหรับใช้ในการศึกษาและทำการ
วิจัยโครงการนี้

ขอขอบคุณ คุณออม คุณเล็ก คุณป้อม คุณแท่ง คุณเอื้อ คุณอ้วน คุณเคื้อ คุณปริชญ์ เพื่อนๆ ชาวห้อง
8 และ ห้อง 9 ทุกคน และเรอคนนั้นที่ให้อำลัังใจพวกเราในการทำโครงการวิจัยนี้

สุดท้ายนี้ บุคคลที่ผู้จัดทำจะลืมเสียมิได้เลย คือ คุณพ่อ คุณแม่ ที่ท่านได้ถูกเป็นอย่างดี



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



- [1] G.M. Nyssen and E.D. Falkenberg (1984) : “Introduction to IBM SQL : covering SQL/DS release 2” , Chapel , Hill/Kenmore , 1984
- [2] Digitalk Inc. (1986) : “Smalltalk/V : Tutorial and Programming Handbook” , Digitalk Inc. , 1986
- [3] Margaret M. Burnett , Adele Goldberg and Ted G. Lewis : “Visual Object - Oriented Programming Concepts and Environments”
- [4] Luiz Fernando Capretz and Miriam A M Capretz : “Object - Oriented Software : Design and Maintenance”
- [5] Dr. Suphamit Chittayasothorn (1994) : “Seminar : Relational Database Design” , Institute of Thai Information Technology (ITIT) , 1994

