การเชื่อมต่อระบบไคล์เอนท์เซิร์ฟเวอร์

CLIENT/SERVER INTERFACING

โดย

นายกอร์ปสินธุ์ จรูญเจตจำนง
นายกิตติพงษ์ ธีระเรืองไชยศรี

วัน เดือน ปี 30 ก.ย. 2540
เลขทะเบียน 038288
เลขเรียกหนังสือ T 59304 กชมc

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตร์บัณฑิต
สาขาวิศวกรรมคอมพิวเตอร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2539

038288

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปีการศึกษา 2539

การเชื่อมต่อระบบไคล์เอนท์เซิร์ฟเวอร์

(Client/Server Interfacing)

โดย

นายกอร์ปสินธุ์ จรูญเจตจำนง

นายกิตติพงษ์ ธีระเรืองไชยศรี

อาจารย์ที่ปรึกษา

อาจารย์บรรจง ปิยธำรง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ปีการศึกษา 2539

ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การเชื่อมต่อระบบไคล์เอนท์เซิร์ฟเวอร์
(Client/Server Interfacing)

ผู้จัดทำ
1. นายกอร์ปสินธุ์ จรูญเจตจำนง
2. นายกิตติพงษ์ ธีระเรืองไชยศรี

..... อาจารย์ที่ปรึกษา
(อาจารย์บรรจง ปิยธำรง)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การเชื่อมต่อระบบไคล์เอนท์เซิร์ฟเวอร์

นายกอร์ปสินธุ์ จรูญเจตจำนง
นายกิตติพงษ์ ธีระเรืองไชยศรี

อาจารย์บรรจง ปิยธำรง อาจารย์ที่ปรึกษา
ปีการศึกษา 2539

บทคัดย่อ

ในปริญญานิพนธ์ฉบับนี้เรียบเรียงขึ้นจากการศึกษาการเชื่อมต่อระบบไคล์เอนท์เซิร์ฟเวอร์ โดยมีวัตถุประสงค์หลักคือ ต้องการศึกษาถึงวิธีการใช้โปรแกรมภาษา 3 ภาษา ได้แก่ ภาษาวิชวลเบสิค (Visual Basic) ภาษาเดลไฟล์ (Delphi) และภาษาเพาเวอร์บิวเดอร์ (Power Builder) สำหรับใช้ในการพัฒนาแอพพลิเคชัน (Application) ที่ทำงานกับข้อมูลบนระบบไคล์เอนท์เซิร์ฟเวอร์ ซึ่งในโปรแกรมภาษาแต่ละภาษามีวิธีในการเชื่อมต่อ และการทำงานกับข้อมูลในฐานข้อมูล ทั้งเหมือนกันและแตกต่างกันออกไป คือ โปรแกรมภาษาทั้ง 3 ภาษาสามารถใช้โอดีบีซี (ODBC) เป็นมิดเดิ้ลแวร์ (Middleware) ในการเชื่อมต่อกับฐานข้อมูลได้ แต่หลักการในการทำงานกับข้อมูลในเดลไฟล์ และวิชวลเบสิคจะแตกต่างจากเพาเวอร์บิวเดอร์ คือ มีคอนโทรลที่ทำงานกับข้อมูลมากกว่าในเพาเวอร์บิวเดอร์ ที่มีเพียงคอนโทรลเดียว ซึ่งอาจทำให้ทำการพัฒนาแอพพลิเคชันยากกว่า แต่ก็มีเหตุการณ์สำหรับตอบสนองการทำงานของผู้ใช้มากกว่าในเพาเวอร์บิวเดอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Client/Server Interfacing

Mr.Korbsin Jaroonjetjumnong

Mr.Kittipong Theeraruangchaisree

Mr.Banjong Piyathumrong Advisor

1996

Abstract

The main purpose of this project is to study the using of 3 programming languages (Visual Basic, Delphi and Power Builder) for developing an application program working with database in the Client/Server environment. Each programming language has the way to connect and to work with data in database both similar and difference. All of the 3 programming languages can use ODBC (OpenDatabase Connectivity) as middleware for connect to the database, but the way to work with data by Delphi and Visual Basic is differ from Power Builder. Visual Basic and Delphi have many control for work with data more than in Power Builder which has only on control. So it may be hard to implement the application program but it has many event to response from users.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบาญ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบาญภาพ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบาญตาราง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

ในปัจจุบันมีการนำเอาระบบคอมพิวเตอร์ มาใช้งานในองค์กรทั่วไปอย่างแพร่หลาย โดยระบบหนึ่งที่นิยมใช้กันคือ ระบบงานในรูปแบบระบบไคล์เอนท์เซิร์ฟเวอร์ (Client/Server) ซึ่งภายในระบบระบบนี้ไคล์เอนท์เซิร์ฟเวอร์นี้ แบ่งส่วนประกอบออกเป็น 2 ส่วน ได้แก่ ส่วนแบคเอนด์ (BackEnd) และ ส่วนฟรอนท์เอนด์ (FrontEnd) โดยในส่วนของฟรอนท์เอนด์ จะทำติดต่อกับผู้ใช้งานเพื่อรับคำสั่ง และในส่วนแบคเอนด์จะทำหน้าที่จัดการกับข้อมูลของระบบที่อยู่ในส่วนเซิร์ฟเวอร์ตามคำสั่งที่ได้รับจากส่วนฟรอนท์เอนด์

โดยทั่วไปในส่วนแบคเอนด์ของระบบมักจะไม่มีการเปลี่ยนแปลง หรือถ้ามีการเปลี่ยนแปลงก็จะเปลี่ยนแปลงน้อยมากเมื่อเทียบกับในส่วนฟรอนท์เอนด์ โดยในส่วนฟรอนท์เอนด์ จะมีเครื่องมือต่างๆ (ในที่นี้หมายถึง ภาษาที่ใช้สำหรับพัฒนาโปรแกรมทำงานกับฐานข้อมูลบนระบบไคล์เอนท์เซิร์ฟเวอร์) ให้เลือกเพื่อใช้ทำงานติดต่อกับส่วนแบคเอนด์หลายชนิด ซึ่งในภาษาที่ใช้พัฒนาโปรแกรมแต่ละชนิดจะมีวิธีการในการติดต่อกับส่วนแบคเอนด์แตกต่างกันไป

วิธีการติดต่อระหว่างส่วนฟรอนท์เอนด์ และส่วนแบคเอนด์ของระบบมีวิธีในการติดต่อโดยอาศัยซอฟท์แวร์ (Software) ที่เรียกว่ามิดเดิ้ลแวร์ (Middleware) โดยมิดเดิ้ลแวร์ที่นิยมใช้กันอย่างแพร่หลายในปัจจุบันนี้คือ โอเพนดาต้าเบสคอนเนคติวิตี้ : โอดีบีซี (Open Database Connectivity : ODBC)

ด้วยเหตุนี้เองจึงเป็นเหตุจูงใจให้ทำการศึกษาวิธีการสร้างและพัฒนาระบบงานที่ต้องมีการติดต่อกับส่วนแบคเอนด์ ของระบบไคล์เอนท์เซิร์ฟเวอร์ที่ทำงานกับข้อมูล โดยใช้ภาษาต่างๆเช่น วิชวลเบสิค เดลไฟล์ และเพาเวอร์บิวเดอร์ และผ่านการเชื่อมต่อโดยใช้โอดีบีซี เป็นมิดเดิ้ลแวร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
ทฤษฎีและหลักการ

ในระบบการเชื่อมต่อแบบไคล์เอนท์เซิร์ฟเวอร์ แบ่งส่วนประกอบใหญ่ออกได้ดังนี้

1. ส่วนการติดต่อกับผู้ใช้งาน (Client)
2. ส่วนการประมวลผลข้อมูล (Server)
3. ส่วนการเชื่อมต่อ

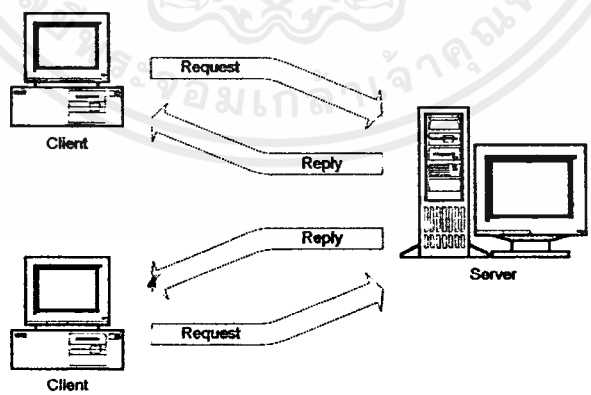
การทำงานกับข้อมูล ในโครงงานนี้อาศัยระบบจัดการฐานข้อมูล ไมโครซอฟท์เอสคิวแอลเซิร์ฟเวอร์ (Microsoft SQL Server) ทำหน้าที่ประมวลผลข้อมูล โดยอาศัยภาษาวิชวลเบสิค (Visual Basic) ภาษาเดลไฟล์ (Delphi) และภาษาเพาเวอร์บิวเดอร์ (PowerBuilder) สำหรับสร้างแอพพลิเคชันเพื่อทำงานกับข้อมูลผ่านโอดีบีซี (Open Database Connectivity:ODBC)

2.1 ความรู้เกี่ยวกับระบบไคล์เอนท์เซิร์ฟเวอร์ (Client/Server System)

โดยทั่วไปความหมายของไคล์เอนท์เซิร์ฟเวอร์ หมายถึง ความสัมพันธ์ระหว่าง 2 ระบบ หรือกระบวนการ โดยไคล์เอนท์ หมายถึง ระบบที่ทำการร้องขอการบริการจากส่วนเซิร์ฟเวอร์ โดยเปรียบไคล์เอนท์เป็นผู้ร้องขอ (Requester) ทำการขอการบริการ (Service) จากส่วนเซิร์ฟเวอร์

เซิร์ฟเวอร์เป็นผู้ให้บริการตามที่ไคล์เอนท์ร้องขอ โดยในเซิร์ฟเวอร์ 1 เซิร์ฟเวอร์สามารถให้บริการกับไคลเอนท์ ได้มากกว่า 1 การบริการ

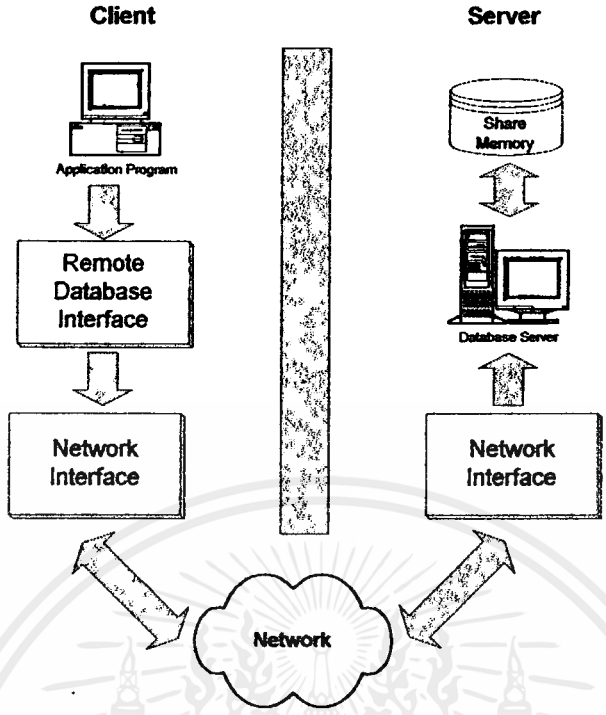
ลักษณะความสัมพันธ์ระหว่างไคล์เอนท์ และเซิร์ฟเวอร์ แสดงดังรูปที่ 2-1



รูปที่ 2-1 แสดงความสัมพันธ์ระหว่างไคล์เอนท์กับเซิร์ฟเวอร์

ในการทำโครงงานการเชื่อมต่อระบบไคล์เอนท์เซิร์ฟเวอร์ มุ่งเน้นการเชื่อมต่อในรูปแบบของเซิร์ฟเวอร์ฐานข้อมูล (Database Server) ซึ่งมีรูปแบบลักษณะการเชื่อมต่อแสดงดังรูปที่ 2-2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2-2 แสดงสถาปัตยกรรมของเซิร์ฟเวอร์ฐานข้อมูล

จากรูปแสดงความสัมพันธ์ของระบบไคล์เอนท์เซิร์ฟเวอร์ในลักษณะเซิร์ฟเวอร์ฐานข้อมูล และแสดงถึงการเคลื่อนที่ของข้อมูล และการควบคุม ในระบบไคล์เอนท์เซิร์ฟเวอร์แบบฐานข้อมูล (Database Client/Server)

ในส่วนแอพพลิเคชันโปรแกรม ประกอบด้วย การร้องขอไปยังฐานข้อมูล (Database Access Request) เช่น คำสั่งภาษาเอสคิวแอล (SQL Statement) ในระหว่างการคอมไพล์ (Compile) และการเชื่อมโยง (Link) ส่วนประกอบต่างๆของโปรแกรมนั้นรีโมทดาต้าแอคเซสสตับ (Remote Database Access Stub) จากไลบรารีของฐานข้อมูล (Database Application Library) จะทำการเชื่อมต่อกันเป็นโปรแกรมที่สมบูรณ์ จากนั้นรีโมทดาต้าเบสอินเตอร์เฟส (Remote Database Interface) จะทำการจัดรูปแบบของการร้องขอทำงานกับฐานข้อมูล (Database Access Request) เพื่อส่งผ่านไปตามเครือข่าย (Network) โดยผ่านในส่วนของการเชื่อมต่อระบบเครือข่าย (Network Interface) ไปยังส่วนเซิร์ฟเวอร์

ในส่วนเซิร์ฟเวอร์นั้น ส่วนของการเชื่อมต่อระบบเครือข่าย จะทำหน้าที่รับข่าวสาร (Message) ที่ส่งมาจากส่วนไคล์เอนท์ และนำส่งต่อไปให้ยังเซิร์ฟเวอร์ฐานข้อมูล เพื่อทำการประมวลผลตามการร้องขอที่ได้รับ และส่งผลลัพธ์กลับคืนสู่แอพพลิเคชันโปรแกรมที่ร้องขอในส่วนไคล์เอนท์ โดยผ่านระบบเครือข่าย

ดังนั้นจากข้อมูลข้างต้น เราสามารถแบ่งส่วนประกอบต่างๆภายในระบบไคล์เอนท์เซิร์ฟเวอร์ ได้เป็น 3 ส่วนประกอบหลัก ได้แก่

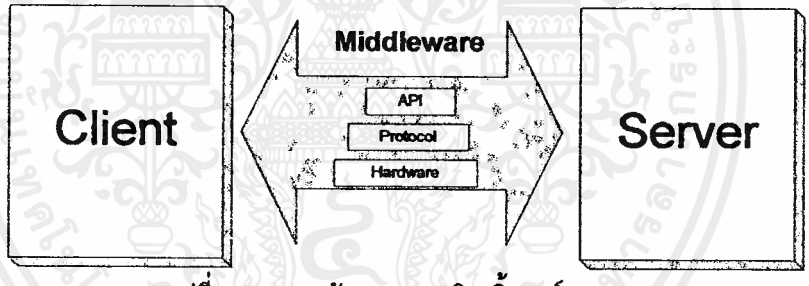
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1. ส่วนไคล์เอนท์ ทำงานบนอุปกรณ์ไคลเอนท์ (Client Hardware) เช่น แสดงรายงาน และ ทำการคำนวณ เบื้องต้น เป็นต้น โดยจะทำการส่งงานที่มีความซับซ้อนไปยังเซิร์ฟเวอร์ เช่น ไคล์เอนท์ทำการเรียกข้อมูลจากเซิร์ฟเวอร์ แล้วทำการเก็บข้อมูลที่เรียกไว้ให้กับผู้ใช้ และอนุญาตให้ผู้ใช้ทำงานกับข้อมูลเหล่านั้นได้ เช่น Search, Graph, Import และอื่นๆ ในการแก้ไขข้อมูลที่เรียกมานั้น ในบางครั้งเป็นการแก้ไขแบบชั่วคราว หรือบางครั้งจะส่งข้อมูลที่แก้ไขแล้วกลับไปที่เซิร์ฟเวอร์ ซึ่งจะเป็นการแก้ไขข้อมูลอย่างถาวร

ไคล์เอนท์โปรแกรม (Client Program) จะมีส่วนที่ทำการติดต่อกับมิดเดิ้ลแวร์ โดยปกติ คือ เอพีไอ (Application Program Interface:API) ซึ่งจะเป็นส่วนที่อนุญาตให้โปรแกรมสามารถทำการติดต่อสื่อสารกับมิดเดิ้ลแวร์ได้ โดยเอพีไอที่นิยมใช้ได้แก่ โอดีบีซี(Open Database Connectivity:ODBC)

2. ส่วนมิดเดิ้ลแวร์ ทำงานอยู่ระหว่างไคล์เอนท์ และเซิร์ฟเวอร์ ซึ่งนับได้ว่าเป็นส่วนที่ซับซ้อนมากที่สุด โดยทั่วไปมิดเดิ้ลแวร์ หมายถึง ซอฟท์แวร์ซึ่งทำให้ไคลเอนท์ และเซิร์ฟเวอร์สามารถสื่อสารกันได้ (เรียกว่า Glue) มิดเดิ้ลแวร์โดยปกติมีหลายชั้น เช่น ถ้าในวิชวลเบสิค ไคล์เอนท์ที่ต้องการเชื่อมต่อ กับเอสคิวแอลเซิร์ฟเวอร์ จะต้องอาศัยมิดเดิ้ลแวร์ที่ประกอบด้วย Winsock, TCP/IP, ODBC และ ODBC Driver

โดยปกติแล้ว ผู้ใช้จะไม่สามารถมองเห็นมิดเดิ้ลแวร์ ซึ่งในมิดเดิ้งแวร์จะประกอบด้วยกลุ่มของไดร์ฟเวอร์ และโปรแกรม



รูปที่ 2-3 แสดงลักษณะของมิดเดิ้ลแวร์

API เช่น Messaging API, ODBC API

Protocol เช่น TCP/IP, Winsock

Hardware เช่น Ethernet, Token Ring

เอพีไอจะถูกเรียกโดยไคล์เอนท์ . และจะทำการติดต่อกับเซิร์ฟเวอร์โดยอาศัยโปรโตคอล (Protocol) ผ่านฮาร์ดแวร์ (Hardware)

3. ส่วนเซิร์ฟเวอร์ จะทำงานที่ซับซ้อน และทำการคำนวณเป็นหลัก โดยทั่วไปแล้วเซิร์ฟเวอร์ หมายถึงฮาร์ดแวร์ที่ทำการทำงานโปรแกรมเซิร์ฟเวอร์ (Run Server Program) ซึ่งตอบสนองคำร้องขอผ่านระบบเครือข่าย เช่น เซิร์ฟเวอร์ฐานข้อมูลรับคำขอข้อมูลจากไคล์เอนท์ และทำการส่งผลลัพธ์ที่ได้จากคำขอมายังไคล์เอนท์ เป็นต้น

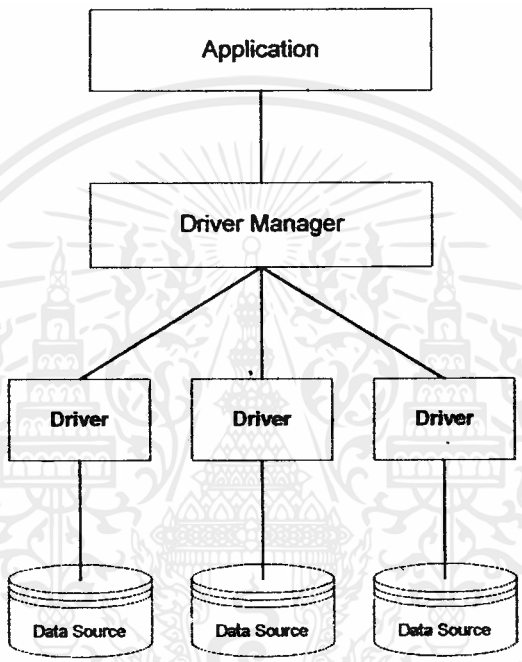
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในส่วนเซิร์ฟเวอร์ มักจะใช้งานยาก รูปแบบที่แสดงไม่สื่อความหมาย เนื่องจากไม่มีการติดต่อกับผู้ใช้โดยตรง แต่มีเพียงการติดต่อผ่านมิดเดิ้ลแวร์ และระบบเครือข่ายเท่านั้น

2.2 ความรู้เกี่ยวกับโอดีบีซี (OpenDatabase Connectivity:ODBC)

โอดีบีซีเป็นวิธีในการเชื่อมต่อระหว่างแอพพลิเคชันโปรแกรมต่างๆกับแหล่งข้อมูล (Data Source) ซึ่งกระทำภายใต้เงื่อนไขการติดต่อ และข้อตกลงเดียวกัน (Set of API)

โอดีบีซีประกอบด้วยส่วนประกอบต่างๆ ดังนี้



รูปที่ 2-4 แสดงส่วนประกอบต่างๆในโอดีบีซี (ODBC Component)

1. Data Source (DB) ในที่นี้หมายถึง ข้อมูล (Data), ระบบจัดการฐานข้อมูล (DBMS) และระบบปฏิบัติการ (PlatForm (O.S))

2. Driver ในที่นี้หมายถึง .DLL ต่างๆ ซึ่งทำหน้าที่รับคำสั่ง ภาษาเอสคิวแอล จากแอพพลิเคชันโปรแกรม และส่งต่อไปยังดาต้าซอสด้วยชุดคำสั่งที่ดาต้าซอสสามารถเข้าใจได้ และทำหน้าที่รับ Result ที่ได้จากดาต้าซอส ส่งกลับมาให้แอพพลิเคชันโปรแกรม

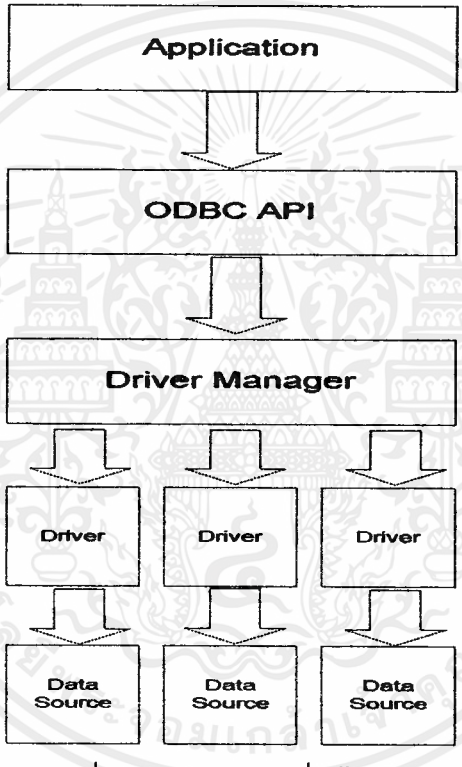
3. Driver Manager เป็นส่วนที่จัดการเกี่ยวกับการเรียกใช้ .DLL ต่างๆ ตามแอพพลิเคชันโปรแกรม

4. Application หมายถึง โปรแกรมที่ทำการใช้งานข้อมูลจากดาต้าซอส ซึ่งเขียนขึ้นโดยภาษาการเขียนโปรแกรมทั่วๆไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

<u>หมายเหตุ</u> DLL (Dynamiccally Linked Libraries) เป็นไดรฟ์เวอร์ที่ทำหน้าที่แปลง ชุดคำสั่งโอดีบีซีเอพีไอไปเป็นฟังก์ชันที่ดาต้าซอสสามารถเรียกใช้งานได้ โดยมากมักจะแปลงจากรูปแบบของเอสคิวแอลในโอดีบีซี ไปเป็นรูปแบบที่ดาต้าซอสเข้าใจ

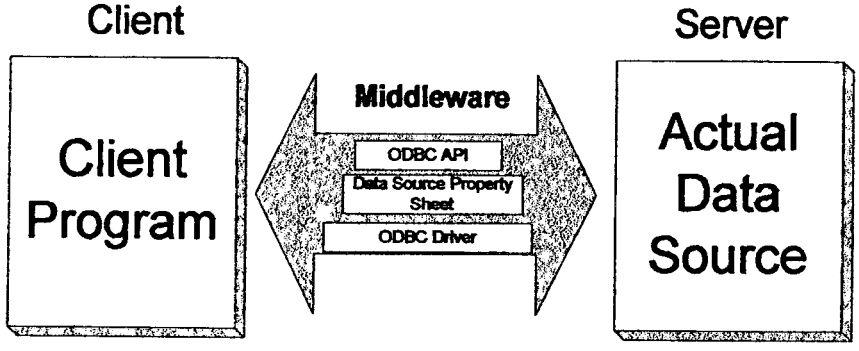
โอดีบีซีเป็นมาตรฐานสำหรับการสร้างดาต้าเบสไดร์ฟเวอร์ (Database Driver) ตามรูปที่ 2-5 โดยแอพพลิเคชันโปรแกรม จะใช้งานโอดีบีซีผ่านคำสั่งมาตรฐานคือ โอดีบีซีเอพีไอ (Open Database Connectivity Application Program Interface:ODBCAPI) โอดีบีซีไดร์ฟเวอร์ใช้ตามดาต้าซอสที่จะใช้แอพพลิเคชันโปรแกรม โดยใช้คำสั่งโอดีบีซี และสามารถใช้งานดาต้าซอสอื่นๆที่มีโอดีบีซีไดร์ฟเวอร์ได้ โดยไม่จำเป็นต้องเปลี่ยนแปลงโปรแกรม



รูปที่ 2-5 แสดงลักษณะทั่วๆไปของโอดีบีซี

มาตรฐานของโอดีบีซี สร้างโดยไมโครซอฟท์ (Microsoft) โดยโอดีบีซีจะทำงานเป็นมิดเดิ้ลแวร์ระหว่างแอพพลิเคชัน และดาต้าซอส โดยผ่านโอดีบีซีไดร์ฟเวอร์ ดังรูปที่ 2-6

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2-6 แสดงแผนภาพโอดีบีซีไดร์ฟเวอร์ (ODBC Driver Diagram)

โอดีบีซีจะมีเอพีไอมาตรฐานเพื่อช่วยในการเข้าถึงดาต้าซอสหลายๆชนิดได้พร้อมๆกัน โดยโอดีบีซีมีความสามารถที่เรียกว่า System Table Transparency (System Table ถูกแก้ไขโดยอัตโนมัติโดย Driver) Schrollable Cursors, Transactions, Asynchronous Calling, Array Fetch and Update, Dynamic Driver Information และ Stored Procedure สำหรับ SQL Database

โอดีบีซีเป็นมาตรฐานเอสคิวแอลเบส (SQL-Based) ซึ่งยอมรับการคิวรี (Query) ทั้งหมดผ่านคำสั่งภาษาเอสคิวแอล ซึ่งใช้ได้กับ Scalar Function, Outer Joins, Stored Procedures, Complex Data Type (เช่น Date, Time, TimeStamp และ Binary) และ SQL Pass-Through เพื่อใช้งาน System Function

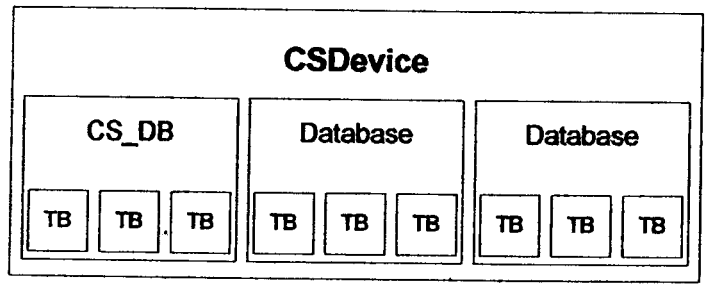
2.3 ความรู้เกี่ยวกับเอสคิวแอลเซิร์ฟเวอร์ (SQL Server)

เอสคิวแอลเซิร์ฟเวอร์-ดาต้าเบส (SQL Server Database) ทำการเข้าถึงข้อมูลโดยผ่านโอดีบีซีไดร์ฟเวอร์ เนื่องจากนิยมใช้โอดีบีซีกันอย่างแพร่หลาย โครงสร้างของเอสคิวแอลเซิร์ฟเวอร์ มีลักษณะแตกต่างจากมัลติยูสเซอร์-เดสทอปดาต้าเบส (MultiUser Desktop Database) ตัวอื่นๆ เนื่องจากภายในไม่มีเครื่องมือในการสร้างฟรอนท์เอนด์ (Front End) มากนัก และโครงสร้างของการเก็บข้อมูลต่างกัน โดยเอสคิวแอลเซิร์ฟเวอร์ใช้โครงสร้างการเก็บข้อมูลได้หลายแบบ ทำให้สามารถรองรับความต้องการของผู้ใช้ และเป็นที่นิยมใช้กันอย่างแพร่หลาย

2.3.1 โครงสร้างข้อมูลในเอสคิวแอลเซิร์ฟเวอร์

เอสคิวแอลเซิร์ฟเวอร์ใช้วิธีกำหนดพื้นที่ดิสก์ (Allocate disk) ตามแบบฉบับของตนเองเพื่อให้มีประสิทธิภาพสูงสุด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป 2-7 แสดงโครงสร้างข้อมูลของเอสคิวแอลเซิร์ฟเวอร์

ดีไวซ์ (Device) เป็นส่วนของการเก็บข้อมูลที่ต่อเนื่องกันบนฮาร์ดดิสค์ของเซิร์ฟเวอร์ ใช้สำหรับเก็บดาต้าเบส (Database) และทรานสแอคชั่น-ล็อก (TransAction Log) ซึ่งดาต้าเบสจะไม่สามารถสร้างขึ้นได้จนกว่าจะมีการสร้างและกำหนดดีไวซ์ก่อนเพื่อสำหรับเก็บดาต้าเบส ดาต้าเบสสามารถเก็บแยกให้อยู่ในดีไวซ์ที่ต่างกันได้

การกำหนดขนาดของดีไวซ์เป็นหน้าที่ของแอดมินิสเทรเตอร์ (Administrator) การทำให้ระบบมีประสิทธิภาพสูงสุด สามารถทำได้โดยการเพิ่มหรือลดขนาดของดาต้าเบสที่เก็บอยู่ในไฟล์พาร์ติชัน (File partition) โดยไม่อ้างอิงกับระบบไฟล์ของโอเอส (O.S) เมื่อมีการสร้างดีไวซ์จะมีการระบุชนิดของดีไวซ์ และมีการทำดาต้าเบสแบคอัพ (Database backup)

ในฐานข้อมูลทำการเก็บตารางต่างๆ โดยใช้เอสคิวแอลออบเจคต์เมเนเจอร์ (SQL Object Manager) หรือ อาจใช้จากคำสั่งเอสคิวแอลโดยตรง ฐานข้อมูลจะถูกทำการกำหนดลงในดีไวซ์ ซึ่งใน 1 ดีไวซ์สามารถเก็บได้หลายตาราง การเชื่อมต่อโดยใช้โอดีบีซีจะกำหนดให้ทำการแอคเซสฐานข้อมูลได้ครั้งละ 1 ฐานข้อมูลเท่านั้น ดังนั้นตารางที่มีความสัมพันธ์กันก็ควรที่จะเก็บในฐานข้อมูลเดียวกัน

การสร้างทรานสแอคชั่นล็อก จะทำการสร้างในดีไวซ์เดียวกันกับฐานข้อมูล ทำหน้าที่เป็นบัฟเฟอร์ (Buffer) เก็บทรานสแอคชั่นต่างๆที่ทำงานกับฐานข้อมูล เพื่อประโยชน์ในการแก้ไขภายหลัง โดยมีหลักการคือ ก่อนที่จะมีการเปลี่ยนแปลงข้อมูล จะต้องทำการบันทึกทรานสแอคชั่นล็อกก่อน และเปลี่ยนแปลงแก้ไขข้อมูลได้ ก็ต่อเมื่อเก็บทรานสแอคชั่นเรียบร้อย

2.3.2 ซิสเต็ม-แอดมินิสเตเตอร์ : เอสเอ (System Administrator : SA) และดาต้าเบสโอนเนอร์ : ดีบีโอ (Databas Owner : DBO)

ทั้งเอสเอและดีบีโอ เป็นบุคคลากรที่มีอำนาจในการเปลี่ยนแปลงโครงสร้างของฐานข้อมูลได้มากกว่าผู้ใช้อื่นๆทั่วไป โดยเอสเอทำหน้าที่ สร้างฐานข้อมูลใหม่ กำหนดและให้สิทธิ์ต่างๆ และควบคุมการทำงานในระบบทั้งหมด ส่วนดีบีโอ หมายถึงผู้เป็นเจ้าของฐานข้อมูล มีสิทธิ์ในการให้สิทธิ์การเข้าถึงข้อมูลแก่ผู้ใช้อื่นๆ

ซิสเต็มเทเบิ้ล (System Table)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอสคิวแอลเซิร์ฟเวอร์ทำการสร้างตาราง เพื่อจัดระเบียบของผู้ใช้ที่ทำการสร้างตารางและคอยควบคุมจัดการ ซึ่งเรียกว่าซิสเต็มคาตาล็อก (System Catalog) เก็บอยู่ในมาสเตอร์-ดาต้าเบส (Master Database) มีทั้งหมด 13 ตาราง

SysCharSet	ชุดตัวอักษรและการจัดลำดับของอักษรที่ใช้ใน SQL Server
SysConfigures	ค่าของ User Parameter ซึ่งจะส่งผลเมื่อกำหนดคำสั่งใน Reconfigure
SysCurConfig	ค่าของ User Parameter ขณะ Sesion หรือ Query จนกระทั่งปัจจุบัน
SysDatabase	ข้อมูลของ Database ต่างๆบนเซิร์ฟเวอร์
SysDevice	ชุดของดีไวซ์ต่างๆในระบบ
SysLanguage	เรคอร์ดของภาษาต่างๆที่ SQL Server สามารถเข้าใจได้ ซึ่งค่า Default คือ US English
SysLock	เรคอร์ดที่ทำการเก็บค่าของสิทธิ์ในการเข้าถึงข้อมูล เช่น Exclusive, Share, Exclusive Intent เป็นต้น
SysLogin	User Account และ Password ทั้งหมด
SysMessage	เรคอร์ดที่ทำการเก็บ Error Message ที่ Return จาก SQL Server
SysProcess	ข้อมูลเกี่ยวกับ Server Process
SysMoteLogins	เรคอร์ดของ Remote User ที่ทำการเรียก Stored Procedure
SysServed	เรคอร์ดสำหรับแต่ละ Remote SQL Server สามารถทำการร้องขอและเรียกใช้ Stored Procedure ได้
SysUsage	ข้อมูลเกี่ยวกับ Disk Allocation

ตารางที่ 2-1 แสดงการเก็บข้อมูลของซิสเต็มเทเบิ้ลต่างๆ

นอกจากนี้ในแต่ละฐานข้อมูลยังเก็บตาราง อีก 13 ตาราง เรียก ดาต้าเบสคาตาล็อก (Database Catalog) หรือดาต้าดิคชันนารี (Data Dictionary)

SysAlternative	ใช้สำหรับ Alternate user ที่ไม่ได้เก็บใน SysUser Table
SysColumn	เรคอร์ดสำหรับคอลัมน์ต่างๆในทุกๆ Table และ View
SysConnect	Comment ต่างๆของ View, Rules, Trigger, Default และ Stored Procedure
SysDepend	Cross-Reference Data
SysIndex	เรคอร์ดสำหรับ Cluster Index และ UnCluster Index
SysKey	เรคอร์ดสำหรับ Key ต่างๆในฐานข้อมูลทั้ง Primary key, Foreign key และ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

	Common key
SysLog	TransAction Log ของ Database
SysObject	เรคอร์ดสำหรับเก็บ Database Object แต่ละชนิด
SysProcedure	เรคอร์ดสำหรับแต่ละ View, Default, Rules, Trigger และ Stored Procedure
SysProtect	User Permission Information รวมถึง Grant และ Revoke
SysSegment	เรคอร์ดของแต่ละ Segment
SysType	ตารางสำหรับ Data type ต่างๆ
SysUser	เรคอร์ดสำหรับ User ที่อนุญาตให้ Access Database

ตารางที่ 2-2 แสดงการเก็บข้อมูลของดาต้าดิคชันนารี

2.3.3 ชนิดข้อมูล (Data Type)

เอสคิวแอลเซิร์ฟเวอร์ให้ใช้งานข้อมูลที่ตรงกับชนิดข้อมูลที่มีในเอสคิวแอลเซิร์ฟเวอร์ และชนิดข้อมูลที่ผู้ใช้กำหนดขึ้นเอง การเลือกใช้ชนิดข้อมูลให้ตรงกับฟิลด์ หรือคอลัมน์ เป็นสิ่งสำคัญ

ชื่อ (Name)	ชนิดข้อมูล (Data Type)	ขนาด (Size)
char(n)	Character	n (nondynamic)
varchar(n)	Character	Size of data,Maximum specified by n
int	Interger	4 bytes
smallint	Interger	2 bytes
tinyint	integer	1 byte
float	Floating-point	8 bytes
real	Floating-point	4 bytes
binary(n)	Binary	n (nondynamic)
varbinary(n)	Binary	Size of data,Maximum specified by n
bit	Other	1 bit
money	Money	8 bytes
smallmoney	Money	4 bytes
datetime	Date and Time	8 bytes .. 4 bytes for

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

		date and 4 bytes for time
smalldatetime	Date and Time	4 bytes .. 2 bytes for date and 2 bytes for time
text	Character	
image	Other	
timestamp	Other	
sysname	Other	NOT NULL

ตารางที่ 2-3 แสดงชนิดข้อมูลในเอสคิวแอลเซิร์ฟเวอร์

2.3.4 วิว (View)

ชนิดของตารางแบบเสมือน คือไม่มีตารางนั้นอยู่จริงในฐานข้อมูล แต่สามารถทำงานแบบตารางได้ เช่น คิวรี (Query) อัพเดท (Update) โดยวิวในเอสคิวแอล จะมีลักษณะเหมือนกับเรคอร์ดเซตในภาษาวิชวลเบสิค ซึ่งมีพื้นฐานจากตารางในฐานข้อมูล วิวมีประโยชน์ในการสร้างคิวรีมาก เพราะในแอพพลิเคชันจะได้ไม่จำเป็นที่จะต้องทำการคิวรี หรือจอยน์ (Join) ทุกครั้งที่มีการเรียกใช้ เพราะได้ทำไว้ก่อนแล้ว

2.3.5 เอสคิวแอลอินเดกซ์ (SQL Index)

ในเอสคิวแอลเซิร์ฟเวอร์ ยอมให้มีการสร้างอินเดกซ์ในฟิลด์ของข้อมูล โดยใช้โครงสร้างแบบบี-ทรี (B-Tree) ในการเก็บอินเดกซ์ และใน 1 ตารางสามารถมีได้มากกว่า 1 อินเดกซ์ และในแต่ละอินเดกซ์อาจเป็นแบบมัลติเทียร์ (Multitier) เอสคิวแอลเซิร์ฟเวอร์จะใช้อินเดกซ์โดยอัตโนมัติเมื่อทำการคิวรี และเพื่อเป็นการหลีกเลี่ยงปัญหาการสร้างอินเดกซ์มากเกินไป ซึ่งจะทำให้ประสิทธิภาพของฐานข้อมูลลดลงได้ ดังนั้นทุกครั้งที่มีการอินเสิร์ท (Insert) หรืออัพเดท (Update) เรคอร์ดก็ต้องมีการอัพเดทอินเดกซ์ด้วย

2.3.6 ทรานสแอคชั่น-เอสคิวแอล (Transaction SQL) สตอร์โพรซีเดอร์ (Store Procedure) และทริกเกอร์ (Trigger)

เอสคิวแอลเซิร์ฟเวอร์ยอมให้มีการเก็บโค้ดในส่วนเซิร์ฟเวอร์ได้ ซึ่งโค้ดเหล่านี้เขียนอยู่ในรูปทรานสแอคชั่น-เอสคิวแอล และเก็บในรูปแบบที่ยังไม่ทำการคอมไพล์ในดาต้าดิคชันนารีของเซิร์ฟเวอร์

ทริกเกอร์เป็นสตอร์โพรซีเดอร์ที่ประมวลผลโดยอัตโนมัติเมื่อตรงกับเงื่อนไขที่กำหนด โดยทำการตรวจสอบเงื่อนไขเหล่านั้นทุกครั้งที่ทำการอินเสิร์ท อัพเดท และดีลีท เรคอร์ด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
ภาษาที่ใช้ในการพัฒนาโปรแกรม

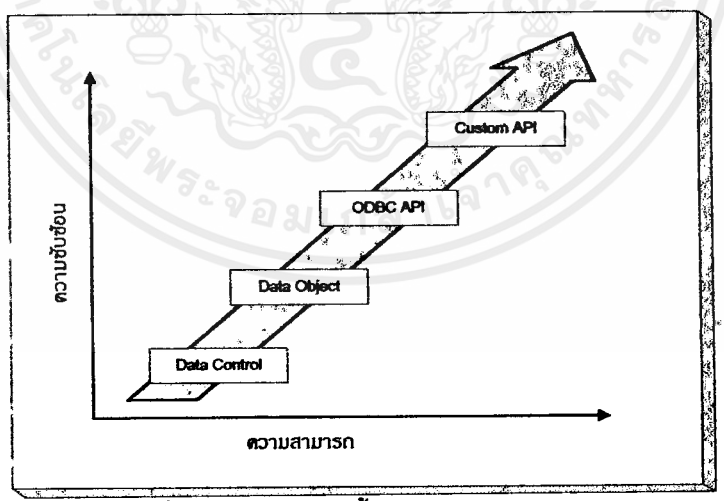
ภาษาที่ใช้ในการพัฒนาโปรแกรมที่ทำงานกับฐานข้อมูล และพัฒนาอยู่บนระบบไคล์เอนท์เซิร์ฟเวอร์ มีอยู่หลายภาษา ดังนี้

3.1 ลักษณะสำคัญและวิธีการใช้วิชวลเบสิคในการพัฒนาโปรแกรม

วิชวลเบสิคมีช่องทางการเข้าถึงข้อมูลได้หลายทาง ซึ่งสามารถใช้งานได้ตั้งแต่ขั้นงานทดลอง (Prototype) และสามารถพัฒนาต่อได้จนเต็มความสามารถ โดยมี 5 ช่องทางในการเข้าถึงข้อมูล ได้แก่

- ดาต้าคอนโทรล (Data Control)
- ดาต้าแอคเซสออบเจคต์ (Data Access Object : DAO)
- รีโมทดาต้าออบเจคต์ (Remote Data Object : RDO)
- โอดีบีซีเอพีไอ (Open Database Connectivity Application Program Interface : ODBC API)
- ไดร์ฟเวอร์เฉพาะกับงาน (Custom Driver)

โดยทั้ง 5 ช่องทางในการเข้าถึงข้อมูลนี้ จะมีความซับซ้อนและประสิทธิภาพและความยากง่ายในการเข้าถึงข้อมูลตามลำดับ โดยแสดงได้ดังรูปที่ 3-1



รูปที่ 3-1 แสดงลำดับขั้นของวิธีการเข้าถึงข้อมูล

การสร้างแอพพลิเคชันโปรแกรม จะเริ่มจากขั้นแรกๆ ซึ่งสามารถใช้งานและแก้ไขได้ง่ายกว่าขั้นอื่นๆ โดยสามารถพัฒนาเพิ่มเติมต่อไปได้อีก

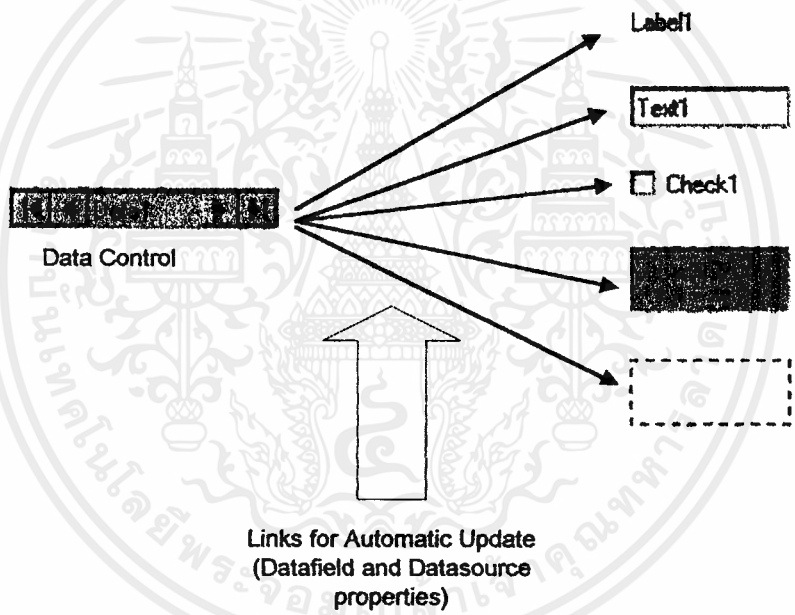
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.1.1 ช่องทางในการทำงานกับฐานข้อมูล

3.1.1.1 ดาต้าคอนโทรล (Data Control)

ดาต้าคอนโทรล เป็นเครื่องมือที่ง่ายที่สุดที่จะสามารถใช้งานได้ ซึ่งสามารถใช้งานได้โดยนำ (Drag) ดาต้าคอนโทรล มาไว้ในฟอร์ม และทำการกำหนดการเชื่อมต่อกับตารางในฐานข้อมูล ได้โดยเพียงแค่กำหนดค่าต่างๆลงในพร็อพเพอร์ตี้ของดาต้าคอนโทรลเท่านั้น ไม่จำเป็นต้องเขียนโค้ด (Code) ก็สามารถใช้งานข้อมูลได้อย่างอัตโนมัติโดยอาศัยคอนโทรลตัวอื่นๆเพิ่มเติม

คอนโทรลที่สามารถใช้งานกับดาต้าคอนโทรล จะเรียกว่า บาวด์คอนโทรล (Bound Control) ซึ่งสามารถทำงานกับข้อมูลในตารางได้อย่างอัตโนมัติ โดยทำการส่งสิ่งที่เปลี่ยนแปลงไปแก้ไขโดยอัตโนมัติ ตัวอย่างบาวด์คอนโทรล เช่น Text Box, Image Control, PictureBox, Label, CheckBox, Masked Edit, ListBox, DBListBox, DBComboBox และ DBGrid



รูปที่ 3-2 แสดงความสัมพันธ์ระหว่างดาต้าคอนโทรล และบาวด์คอนโทรล

ดาต้าคอนโทรลเป็นช่องทางพื้นฐานที่จะใช้งานฐานข้อมูล ใช้งานได้เร็ว สะดวก และมีความสามารถพอสมควร สามารถใช้คอนโทรลในการแก้ไขข้อมูลโดยอัตโนมัติ ดาต้าคอนโทรลสามารถทำงานกับฐานข้อมูลได้โดยใช้ดาต้าออบเจคต์ นอกจากนี้ยังมีอินเตอร์เฟส (Interface) สำหรับการเรียกใช้ข้อมูลในฐานข้อมูล เช่น Next Record, Previous Record, First Record, Last Record เป็นต้น ดาต้าคอนโทรลใช้คิวรี (Query) ในการแสดงข้อมูลเฉพาะที่กำหนดในคำสั่งคิวรี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ดาต้าคอนโทรลมีพร็อพเพอร์ตี้หลักๆที่ใช้ในการเข้าถึงฐานข้อมูล เช่น Connect, Database Name, RecordSource, RecordSet Type, BOF Action, EOF Action และ Read Only Property โดยพร็อพเพอร์ตี้ Connect, Database Name และ RecordSet ใช้เพื่อระบุ ดาต้าซอส (Data Source) ที่จะใช้ ส่วนพร็อพเพอร์ตี้อื่นๆ ใช้บอกลักษณะของการใช้งาน

ตารางต่อไปนี้แสดงถึงพร็อพเพอร์ตี้การเชื่อมต่อ (Connect) ใช้สำหรับบอกชนิดของฐานข้อมูลที่จะทำการเชื่อมต่อด้วย ดังตารางที่ 3-1

Database Format	Database Property	
Connect Property		
Microsoft Access	drive:\path\filename.MDB	Access ;
dBase III	drive:\path	dBase III;
dBase IV	drive:\path	dBase IV;
Paradox 3	drive:\path	Paradox 3.x;
Paradox 4	drive:\path	Paradox 4.x;
Btrieve	drive:\path\filename.DDF	Btrieve;
FoxPro 2.0	drive:\path	FoxPro 2.0;
FoxPro 2.5	drive:\path	FoxPro 2.5;
FoxPro 2.6	drive:\path	Foxpro 2.6;
Excel 3.0	drive:\path\file.XLS	Excel 3.0;
Excel 4.0	drive:\path\file.XLS	Excel 4.0;
Excel 5.0	drive:\path\file.XLS	Excel 5.0;
Text	drive:\path\file	Text;
ODBC	Data source name or an empty string ("")	*
*ODBC;DSN=server;DATABASE=defaultdatabase;UID=user;PWD=password;LoginTimeout=seconds		

ตารางที่ 3-1 แสดงพร็อพเพอร์ตี้การเชื่อมต่อที่เหมาะสมสำหรับดาต้าซอสต่างๆ

ในการติดต่อกับโอดีบีซี ในพร็อพเพอร์ตี้การเชื่อมต่อ (Connect) จะมีพารามิเตอร์เพื่อใช้สำหรับระบบที่จะทำงานกับฐานข้อมูลภายนอก เช่น ODBC;DSN=DataSourceName;UID=UserId; PWD=Password เป็นต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Database Name พร็อพเพอร์ตี้ ประกอบด้วยชื่อของไฟล์ข้อมูล (Data File) หรือดาต้าซอส ที่ต้องการจะใช้งาน ถ้าใช้โอดีบีซีแล้วไม่ได้ทำการกำหนดค่านี้ โอดีบีซีไดอะลอกซ์บอกซ์ (ODBC Dialog Box) จะปรากฏให้เลือกดาต้าซอส

Record Source พร็อพเพอร์ตี้จะใช้ในการระบุตารางหรือคิวรีที่ดาต้าคอนโทรล และดาต้าออบเจคต์จะนำเรคอร์ดมาใช้งาน ซึ่งสามารถเลือกได้โดยจะปรากฏเมนูขึ้นมาให้เลือกตามชื่อของตารางที่มีอยู่ในฐานข้อมูลขณะนั้น

การรวมข้อมูล (Join) สามารถทำได้โดยทำผ่านพร็อพเพอร์ตี้เรคอร์ดเซต ซึ่งสามารถทำรวมข้อมูลจากหลายตารางได้ ได้โดยไม่ต้องมีการเขียนโค้ด หรือคำสั่งภาษาเอสคิวแอล

สามารถเรียกใช้ รีเฟรชเมธอด (Refresh Method) เพื่อทำการเปิดฐานข้อมูลอีกครั้งถ้าค่าของพร็อพเพอร์ตี้เปลี่ยนไป หรือต้องการเขียนข้อมูลจากบัฟเฟอร์ (Buffer) ลงฐานข้อมูล

ขั้นตอนและวิธีการใช้งานดาต้าคอนโทรลจัดการกับฐานข้อมูล

1. ทำการกำหนดดาต้าคอนโทรลที่ต้องการใช้ขึ้นในฟอร์ม

2. ทำการกำหนดพร็อพเพอร์ตี้ของดาต้าคอนโทรลนั้น ซึ่งมีพร็อพเพอร์ตี้สำคัญที่เกี่ยวกับใช้งานฐานข้อมูล ดังนี้

   2.1 Database Name ใช้เพื่อบอกว่าต้องการใช้ดาต้าคอนโทรลทำงานกับฐานข้อมูลใด โดยระบุชื่อฐานข้อมูล

   2.2 Record Source ค่าที่กำหนดให้กับดาต้าคอนโทรล อาจเป็นชื่อของตารางหรือคิวรีเดฟ (QueryDef) ก็ได้

   2.3 Exclusive กำหนดเป็นจริง เมื่อต้องการใช้งานเพียงคนเดียว และกำหนดเป็นเท็จ เมื่อต้องการให้ใช้ร่วมกับผู้อื่น

   2.4 Connect สำหรับระบุชนิดของฐานข้อมูลที่ต้องการทำงานด้วย เช่น ACCESS, LOTUS หรือ ODBC เป็นต้น

   2.5 ReadOnly กำหนดค่าเป็นจริง เมื่อต้องการอ่านอย่างเดียว และกำหนดค่าเป็นเท็จ เมื่อต้องการเปลี่ยนแปลงค่าได้

   2.6 Option กำหนดตัวเลือกของไดนาเซต

ในกรณีการ Unload ฟอร์ม วิชวลเบสิคจะทำการปิดไดนาเซต และฐานข้อมูลให้เองโดยอัตโนมัติ ซึ่งเป็นการง่ายกว่าการเรียกใช้ฟังก์ชัน OpenDatabase และ CreateDynaset โดยตรง

3. กำหนดคอนโทรลอื่นๆขึ้นบนฟอร์ม เพื่อใช้ในการแสดงข้อมูลจากฐานข้อมูล หรือรับข้อมูลไปทำงาน โดยเรียกคอนโทรลอื่นๆที่ทำงานร่วมกับดาต้าคอนโทรลว่า บาวด์คอนโทรล (Bound Control)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เหตุการณ์เฉพาะของดาต้าคอนโทรล

1. Validate เหตุการณ์ที่เกิดขึ้นเมื่อมีการกระทำใดๆกับฐานข้อมูล ได้แก่ Update, Addnew, Delete, Move, Find, เปลี่ยนค่าBookmark หรือเมื่อปิดดาต้าคอนโทรล หรือเมื่อปิดฟอร์มที่มีดาต้าคอนโทรลอยู่ก็ได้ โดยอาจไม่มีการเปลี่ยนแปลงกับฐานข้อมูลแต่อย่างใด

2. Reposition เหตุการณ์ที่เกิดกับดาต้าคอนโทรลภายหลังจากการเปลี่ยนเรคอรด์ปัจจุบันแล้ว

ฟังก์ชัน (Method) ของดาต้าคอนโทรล

1. Refresh ใช้ในการอ่านค่าเรคอร์ดเซตตามที่กำหนดในพร็อพเพอร์ตี้ DatabaseName และ RecordSource เพื่อให้ได้ข้อมูลล่าสุด

2. UpdateControl ใช้เรียกเอาข้อมูลของเรคอร์ดปัจจุบันมาแสดงผลใหม่อีกครั้ง เพื่อให้มั่นใจได้ว่าไม่มีการเปลี่ยนแปลงข้อมูล กละอาจใช้สำหรับเพื่อยกเลิกการแก้ไข แต่ต้องทำการส่งก่อนเลื่อนข้อมูลไปเรคอร์ดอื่น

3. UpdateRecord ใช้สั่งให้วิชวลเบสิคนำเอาค่าคอนโทรลที่ผูกไว้กับฟิลด์ข้อมูล แก้ไขกลับเขาไปในฐานข้อมูลของดาต้าคอนโทรลทันที โดยไม่ตองทำการเลื่อนเรคอร์ดอื่นๆ

3.1.1.2 ดาต้าแอคเซสออบเจคต์ (Data Access Object)

ดาต้าแอคเซสออบเจคต์มีความสามารถมากกว่าดาต้าคอนโทรล แต่ต้องการการเขียนโปรแกรม เข้ามาเพิ่มเติม ดาต้าแอคเซสออบเจคต์สามารถทำงานทุกอย่างตามที่ดาต้าคอนโทรลทำได้ แต่จะไม่เป็นกราฟฟิคอยู่บนฟอร์มเหมือนคอนโทรล โดยดาต้าแอคเซสออบเจคต์สามารถรับคำสั่งทุกอย่างได้เหมือนการใช้คอนโทรล ดังนั้นจึงสามารถใช้โค้ดซึ่งใช้กับคอนโทรลมาใช้กับดาต้าแอคเซสออบเจคต์ได้โดยง่าย

ดาต้าแอคเซสออบเจคต์มีข้อดีเหนือกว่าดาต้าคอนโทรล เพราะสามารถทำการบำรุงรักษา (Maintain) ได้ง่ายกว่า. เพิ่มเติมส่วนประกอบอื่นๆได้ง่าย และทำการอ้างถึงได้ง่ายในการเขียนโปรแกรม นอกจากความน่าเชื่อถือ และความยืดหยุ่นในการใช้งานแล้ว คอนโทรลบางอย่างไม่สามารถใช้กับดาต้าคอนโทรล ได้โดยตรง จำเป็นต้องใช้การเขียนโปรแกรมเข้าช่วย

การแก้ไขข้อมูล โดยไม่ต้องใช้บาวด์คอนโทรลช่วย สามารถทำได้โดย

1. สร้างตาราง, เรคอร์ดเซต (RecordSet) หรือ สแน็พชอต (SnapShot) ออบเจคต์ของข้อมูลที่จะใช้งาน
2. เลือกเรคอร์ดที่ต้องการจะใช้งาน (โดยใช้ Search หรือ Query)
3. เรียกใช้อีดิทเมธอด (Edit Method) เพื่อเรียกเรคอร์ดเข้าสู่หน่วยความจำและตั้งค่าการแก้ไข
4. ใส่ข้อมูลลงในชุดของตัวแปร หรือ บัฟเฟอร์อาร์เรย์ (Buffer Array)
5. ทำการถ่ายข้อมูลจากบัฟเฟอร์มายังคอนโทรลตัวอื่นๆบนหน้าจอ
6. ผู้ใช้สามารถจัดการกับข้อมูลบนฟอร์มได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

7. เมื่อผู้ใช้เลือก OK. ก็จะทำการเก็บค่าจากหน้าจอลงในบัฟเฟอร์

8. ทำการแก้ไขเรคอร์ด โดยใช้ค่าในบัฟเฟอร์

9. เรียกใช้ อัพเดทเมธอด(Update Method) ซึ่งจะเป็นการเขียนข้อมูลลงสู่ตาราง

ตาราง, เรคอร์ดเซตและสแน็พชอต จะมีทางเลือกหลายทางเพื่อการเข้าถึงข้อมูล ตามตารางที่ 3-2

Access	Subsets of Data?	Read Only?	Join?	Optimized for small table?	Optimized for Query?
Table	No	Optional	No	No	No
Recordset	Yes	Optional	Yes	No	Yes
Snapshot	Yes	Yes	Yes	Yes	Yes

ตารางที่ 3-2 แสดงการเข้าถึงตาราง เรคอร์ดเซต และสแน็พชอต

3.1.1.3 รีโมทดาต้าออบเจคต์ (Remote Data Object)

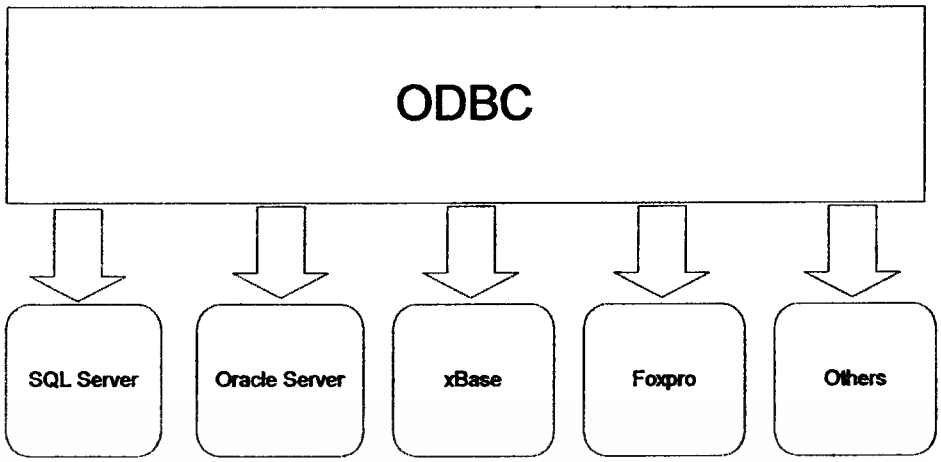
รีโมทดาต้าออบเจคต์ทำหน้าที่คล้ายดาต้าแอคเซสออบเจคต์ แต่สามารถทำงานกับโอดีบีซีดาต้าซอสได้ดีกว่า (รีโมทดาต้าออบเจคต์ มีเฉพาะในวิชวลเบสิคเอนเตอร์ไพรส์เอดิชันเท่านั้น) โดยรีโมทดาต้าออบเจคต์ทำหน้าที่ติดต่อกับโอดีบีซีดาต้าซอส ซึ่งสามารถทำ สตอร์โพร์ซีเดอร์ (Stored Procedure), เคอร์เซอร์ (Cursor) และ เรคอร์ดเซตได้ และในทำนองเดียวกันลักษณะการเขียนโปรแกรมของดาต้าแอคเซสออบเจคต์สามารถนำมาใช้กับรีโมทดาต้าออบเจคต์ได้ นอกจากนี้ยังมี รีโมทดาต้าคอนโทรล (Remote Data Control : RDC) ซึ่งมีการทำงานเหมือนดาต้าคอนโทรลอีกด้วย

สาเหตุที่รีโมทดาต้าออบเจคต์ (และรีโททดาต้าคอนโทรล) ทำงานได้ดีกว่าดาต้าแอคเซสออบเจคต์ เพราะดาต้าแอคเซสออบเจคต์ และดาต้าคอนโทรล ใช้เทคโนโลยีที่เรียกว่า เจทเอนจิ้น (Jet Engine) ซึ่งเป็นกลไกที่ใช้ในการติดต่อกับฐานข้อมูล (Database Interface Engine) โดยมีคำสั่งพื้นฐานของวิชวลเบสิคในการจัดการกับดาต้าซอส แต่เมื่อเจทเอนจิ้นทำงานกับฐานข้อมูลชนิดอื่น (เช่น โอดีบีซี) ที่ไม่ใช่ฐานข้อมูลชนิดแอคเซส (Access) จะทำงานได้ช้า ดังนั้นถ้าใช้รีโมทดาต้าออบเจคต์ จะทำงานได้เร็วกว่าดาต้าแอคเซสออบเจคต์ เพราะรีโมทดาต้าออบเจคต์ไม่ได้ใช้เจทเอนจิ้น

3.1.1.4 โอดีบีซีเอพีไอ (ODBC API)

โอดีบีซีเอพีไอเป็นมิดเดิ้ลแวร์ (Middleware) ซึ่งทำให้วิชวลเบสิคมีความสามารถในการทำงานกับข้อมูลในฐานข้อมูลมากยิ่งขึ้น โอดีบีซีเอพีไอเป็นมาตรฐานซึ่งใช้สำหรับเข้าถึงฐานข้อมูลใดๆที่มีโอดีบีซีไดร์ฟเวอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3-3 แสดงโอดีบีซีไดร์ฟเวอร์

โอดีบีซีเอพีไอเป็นช่องทางที่มีความซับซ้อนมากที่สุด เพราะต้องทำการเรียกใช้เอพีไอ (API) โดยตรง เพื่อทำการติดต่อ, ส่งคำสั่ง และ งานอื่นๆ

3.1.1.5 ไดร์ฟเวอร์เฉพาะกับงาน (Custom Driver)

ไดร์ฟเวอร์สำหรับทำงานกับฐานข้อมูลตัวใดตัวหนึ่งโดยเฉพาะ เช่น OCX, VBX, DLL เป็นต้น ข้อดีของไดร์ฟเวอร์เฉพาะกับงาน คือเป็นช่องทางที่เร็วและมีความเหมาะสมมากที่สุดกับฐานข้อมูลแต่ละตัว แต่มีข้อเสียอย่างค่อนข้างมากคือ โปรแกรมที่ใช้งานจะเป็นโปรแกรม เฉพาะการทำงานของระบบจัดการฐานข้อมูล (DBMS) แต่ละตัว ไม่สามารถนำไปใช้กับระบบจัดการฐานข้อมูลตัวอื่นๆได้ ซึ่งต่างกับโอดีบีซีที่สามารถเปลี่ยนดาต้าซอลโดยไม่ส่งผลกระทบต่อโปรแกรม

3.1.2 การจัดการกับข้อมูลโดยใช้โปรแกรม (Manipulating data with a program)

ดาต้าแอคเซสแอพพลิเคชั่น (Data access application) สามารถทำการพัฒนาได้โดยชุดคำสั่งโปรแกรม (Program Commmand) หรือโดยใช้ดาต้าคอนโทรลและบาวด์คอนโทรล หรืออาจใช้ทั้งสองแบบประกอบกัน ซึ่งการใช้ดาต้าคอนโทรลและบาวด์คอนโทรลใช้ในการเข้าถึงข้อมูลได้โดยง่ายเพียงแค่ทำการกำหนดค่าพร็อพเพอร์ตี้ต่างๆในคอนโทรลเหล่านั้น เมื่อมีการเรียกใช้ชุดคำสั่งโปรแกรมจะมีการทำงานกับดาต้าแอคเซสออบเจ็คต์ของวิชวลเบสิค ซึ่งการใช้งานจะค่อนข้างซับซ้อนกว่าดาต้าคอนโทรลและบาวด์คอนโทรล แต่จะมีความยืดหยุ่นที่ดีกว่าในบางแอพพลิเคชั่น นอกจากนี้ผู้เขียนโปรแกรมยังสามารถรู้ถึงการกระทำต่างๆในออบเจ็คต์นั้นด้วย ทำให้การเรียกใช้งานมีประสิทธิภาพ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.1.3 เรคอร์ดเซตออบเจคต์ (RecordSet Object)

เรคอร์ดเซตออบเจ็คต์ เป็นออบเจ็คต์ใหม่ที่มีในวิชวลเบสิค 4.0 ซึ่งมีรูปแบบเดิมมาจาก ตาราง (Table) ไดนาเซต (Dynaset) และ สแน็พชอต (Snapshot)ในวิชวลเบสิค 3.0 โดยแทนที่จะใช้คำสั่ง OpenTable หรือ OpenDynaset ก็ใช้คำสั่ง OpenRecordSet แทน

การเริ่มต้นการเขียนดาต้าแอคเซสโปรแกรมคือการกำหนดการเชื่อมต่อกับฐานข้อมูล (Set up link with database) ที่ต้องการทำงานด้วย เมื่อได้ทำการสร้างฐานข้อมูล (Create Database) แล้วฐานข้อมูลนั้นก็สามารถนำไปใช้งานในโปรแกรมได้ เมื่อจบโปรแกรมแล้วก็ทำการปิดฐานข้อมูล (Close Database) และถ้ามีฐานข้อมูลเก็บอยู่ก่อนแล้วจะต้องทำการเปิดฐานข้อมูลก่อนที่เรียกใช้งานด้วย โดยใช้คำสั่ง OpenDatabase แสดงดังตัวอย่าง

```
Dim  MyDB as Database
Dim  MyWS as Workspace
Set  MyWS = DBEngine.WorkSpaces(0)
Set  MyDB = MyOpenDatabase( "ชื่อฐานข้อมูล" ,Fals,False,"ODBC")
```

หลังจากที่ทำการเปิดฐานข้อมูลแล้วต้องทำการกำหนดเรคอร์ดเซต เพื่อที่จะใช้ในการเข้าถึงข้อมูลในฐานข้อมูล (Access Data in Database) เสมือนว่าการเปิดฐานข้อมูล (OpenDatabase) เป็นการบอกกับโปรแกรมว่าข้อมูลที่ต้องการใช้งานด้วยอยู่ที่ไหน และการเปิดเรคอรด์เซต (OpenRecordSet) เป็นการบอกกับโปรแกรมว่ามีข้อมูลอะไรบ้างที่จะทำงานด้วย

ชนิดของเรคอร์ดเซตที่มีในวิชวลเบสิค 4..0

- ตาราง (Table) คือ รูปแบบโครงสร้างโดยปกตของฐานข้อมูลที่ทำการเก็บข้อมูลในตาราง
- ไดนาเซต (Dynaset) คือ เซตของพอยน์เตอร์(Set of pointer) ที่บอกและชี้ไปยังฟิลด์(Field) และ เรคอร์ด(Record)ของข้อมูลจากตารางหนึ่งตารางหรือมากกว่านั้น
- สแน็พชอต (Snapshot) คือ ชุดสำรองของข้อมูล (Copy of Data) แบบอ่านได้อย่างเดียว(Read only) ที่ตรงกับเงื่อนไข ซึ่งเก็บในหน่วยความจำ (Memory)

3.1.3.1 การใช้งานตารางในเรคอร์ดเซต

ตารางเป็นตัวแทนของข้อมูลแบบกายภาพ (Physical Representation of data) เพราะข้อมูลที่เก็บในฐานข้อมูลเก็บอยู่ในรูปแบบของตาราง ดังนั้นการเข้าถึงข้อมูลจากตารางเป็นวิธีที่ตรงที่สุด ซึ่งตารางเป็นเรคอร์ดเซตรูปแบบเดียวที่สนับสนุนการใช้อินเด็กซ์ (Index) จึงทำให้การค้นหาข้อมูลในฐานข้อมูลทำได้เร็วกว่าโดยใช้เรคอร์ดเซตแบบไดนาเซต และเรคอร์ดเซตแบบสแน็พชอต

ในการใช้ตาราง ข้อมูลจะถูกเปลี่ยนแปลงในเวลาใดเวลาหนึ่งแค่หนึ่งเรคอร์ดจากหนึ่งตาราง จึงเป็นการใช้งานข้อมูลได้อย่างมีประสิทธิภาพ แต่ไม่ยินยอมให้มีการเปลี่ยนแปลงเรคอร์ดจากหลายๆตารางโดยคำสั่งเพียงคำสั่งเดียว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้
038288

ข้อดีของการใช้ตาราง

1) ใช้อินเด็กซ์ในการเปลี่ยนแปลงลำดับของข้อมูลในตาราง ขณะทำการประมวลผลโปรแกรม (Execute Program) ได้

2) ทำการค้นหาเรคอร์ด (Find Method) โดยใช้อินเด็กซ์ได้รวดเร็วกว่าเรคอร์ดเซตแบบอื่นๆ

3) การเปลี่ยนแปลงข้อมูลในตารางมีผลทันที ไม่จำเป็นต้องทำการรีเฟรช (Refresh)

ข้อเสียของการใช้ตาราง

1) ทำการกำหนดเงื่อนไข เพื่อทำการกรอง (Filter) เรคอร์ดหรือฟิลด์ที่ไม่ต้องการออกไม่ได้

2) ไม่สามารถเรียกใช้คำสั่งค้นหา (Find) ให้ตรงกับเงื่อนไขที่กำหนดได้

วิธีการเปิดตารางสำหรับใช้งาน มีดังนี้

```
Dim  MyTbl as Recordset
Set  MyTbl = MyDB.OpenRecordset( "ชื่อตาราง" ,dbOpenTable)
```

ตัวเลือก(Option) ที่ใช้สำหรับการเปลี่ยนแปลงวิธีการเข้าถึงข้อมูลในตาราง

1) dbDenyWrite สำหรับป้องกันผู้ใช้อื่นในระบบทำการเขียนข้อมูลขณะเปิดใช้งาน

2) dbDenyRead สำหรับป้องกันผู้ใช้อื่นในระบบทำการอ่านข้อมูลขณะเปิดใช้งาน

3) dbReadOnly สำหรับป้องกันการเปลี่ยนแปลงข้อมูลในตาราง

3.1.3.2 การใช้งานไดนาเซตในเรคอร์ดเซต (Dynaset)

ไดนาเซต เป็นกลุ่มของข้อมูลจากตาราง หรืออาจมากกว่าหนึ่งตาราง ซึ่งตรงกับเงื่อนไขที่กำหนด และทำการระบุตำแหน่งของเรคอร์ดจากตารางตั้งแต่เริ่มสร้างไดนาเซต โดยไดนาเซตเป็นเรคอร์ดเซตที่สามารถแก้ไขได้ (Updatable) ดังนั้นการเปลี่ยนแปลงที่เกิดขึ้นจากผู้ใช้จะถูกเก็บลงในฐานข้อมูล แต่ไดนาเซตจะไม่ได้รับผลกระทบจากการเพิ่ม หรือลบเรคอร์ดจากผู้ใช้อื่นในระบบ

ข้อดีของไดนาเซต

1) ไดนาเซตสามารถให้มีการรวม (Join) ข้อมูลจากหลายๆตารางได้

2) สามารถใช้ Find Method ในการกำหนดตำแหน่ง หรือทำงานกับแต่ละเรคอรด์ที่ตรงกับเงื่อนไข

3) สามารถใช้พร็อพเพอร์ตี้(Property)ของการกรอง (Filter) หรือการจัดลำดับ (Sort order) ในการเปลี่ยนแปลงมุมมองของข้อมูล(View Of Data) ได้

ข้อเสียของไดนาเซต

1) ไม่สามารถทำการสร้างอินเดกซ์ในไดนาเซตได้ ทั้งนี้เป็นการป้องกันการเปลี่ยนแปลงลำดับของข้อมูลในไดนาเซตจากการเปลี่ยนแปลงอินเดกซ์

2) ไม่ได้รับผลกระทบจากการเพิ่มหรือลบเรคอร์ดจากผู้ใช้อื่นๆ ดังนั้นไดนาเซตจึงจำเป็นต้องมีการรีเฟรช หรือ ทำการสร้างซ้ำ (Re-Create) เพื่อแสดงการเปลี่ยนแปลงของข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คำสั่งการกำหนดและติดตั้งไดนาเซต ใช้คำสั่งภาษาเอสคิวแอลเป็นการกรอง เพื่อนำฟิลด์ที่ต้องการใช้งานมาเท่านั้น โดยมีวิธีดังนี้

```
Dim  MyDN as Recordset
Set  MyDN = MyDB.OpenRecordset( "คำสั่งเอสคิวแอล",dbOpenDynaset)
```

ตัวเลือกที่ใช้ในการเปลี่ยนแปลงวิธีการเข้าถึงข้อมูลในไดนาเซต

1) dbDenyWrite สำหรับป้องกันผู้ใช้อื่นในระบบทำการเขียนลงในไดนาเซต ขณะเปิดใช้งาน

2) dbReadOnly สำหรับป้องกันการเปลี่ยนแปลงในไดนาเซต

3) dbAppendOnly กำหนดให้ยอมสำหรับสร้างเรคอร์ดใหม่ แต่ไม่ให้มีการแก้ไขเรคอร์ดเดิมที่มีอยู่ในเรคอร์ดเซต

4) dbSqlPassThrough สำหรับใช้ผ่านค่า SQL ที่ใช้สำหรับสร้างไดนาเซตใน โอดีบีซี ดาต้าเบสเซิร์ฟเวอร์ (ODBC Database Server) เพื่อใช้งาน

โอดีบีซีเซิร์ฟเวอร์(ODBC Server) เป็นกลไกฐานข้อมูล (DBEngine) เช่น ไมโครซอฟท์เอสคิวแอลเซิร์ฟเวอร์ (Microsoft SQL Server) ออราเคิล (Oracle) ที่ใช้โอดีบีซีเป็นมาตรฐาน โดยจุดประสงค์หลักของเซิร์ฟเวอร์ประเภทนี้ คือ รองรับการทำคิวรี (Query)ที่เซิร์ฟเวอร์ และทำการส่งเฉพาะผลลัพธ์ที่คิวรีได้กลับมาที่ไคลเอนท์ โดยมากผู้ผลิตกลไกฐานข้อมูลจะเป็นผู้สร้างโอดีบีซีไดร์ฟเวอร์ (ODBC Driver) เพื่อรองรับการเชื่อมต่อระหว่างวิชวลเบสิคกับดาต้าเบสเซิร์ฟเวอร์ โดยไม่ต้องทราบถึงการทำงานภายใน

เราสามารถทำการสร้างไดนาเซตได้จากไดนาเซตอื่นๆที่มีอยู่ก่อน หรือจากคิวรีเดฟ (QueryDef) โดยเหตุผลของการสร้างไดนาเซตจากไดนาเซตอื่น คือเพื่อทำการกรองหรือจัดลำดับข้อมูลในไดนาเซตต้นแบบให้มีขนาดของขอบเขตเล็กลง เพื่อให้มีการทำงานกับเรคอร์ดที่ต้องการได้เร็วยิ่งขึ้น

3.1.3.3 การใช้งานสแน็พชอตในเรคอร์ดเซต (Snapshot)

สแน็พชอตเป็นชุดสำรองข้อมูลในเรคอร์ดเซตที่ชี้อยู่ ณ เวลาใดเวลาหนึ่ง โดยสแน็พชอตมีลักษณะเหมือนกับไดนาเซตที่สามารถสร้างได้จาก คำสั่งเอสคิวแอล หรือคิวรีเดฟ หรือสแน็พชอตที่มีอยู่ได้ แต่ต่างกันตรงที่ไม่สามารถทำการแก้ไขได้ ซึ่งโดยมากจะใช้ในการสร้างรายงาน และการแสดงผลข้อมูลบนหน้าจอที่ใช้ข้อมูลแบบคงที่

ข้อดีของการใช้สแน็พชอต

1) สามารถทำการรวมข้อมูลจากหลายตารางได้

2) สามารถใช้การค้นหา (Find Method) เพื่อทำการกำหนดตำแหน่งของเรคอรด์ได้

3) การทำเรคอร์ดเนวิเกชั่น (Record Navigation) และ การสร้างเรคอร์ดเซต สำหรับใช้งานทำได้เร็วกว่าไดนาเซตแบบอ่านได้อย่างเดียว (Read Only Dynaset) เพราะเป็นชุดสำรองของข้อมูลไม่ใช่พอยน์เตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อเสียของการใช้สแน็พชอต

1) สแน็พชอตเป็นเรคอร์ดเซตแบบไม่สามารถทำการแก้ไขได้

2) ไม่สามารถทำการสร้างอินเดกซ์ในสแน็พชอตเพื่อใช้ในการกำหนดตำแหน่งและลำดับของข้อมูลได้

วิธีกำหนดและติดตั้งสแนพชอต มีวิธีคล้ายกับการกำหนดในไดนาเซต แสดงดังนี้

```
Dim  MySN as Recordset
Set  MySN = MyDB.OpenRecordset( "คำสั่งเอสคิวแอล" ,dbOpenSnapshot)
```

ตัวเลือกที่ใช้ในการเปลี่ยนแปลงวิธีการเข้าถึงข้อมูลของสแน็พชอต

1) dbDenyWrite สำหรับป้องกันผู้ใช้อื่นในระบบทำการเขียนลงในสแน็พชอต ขณะเปิดใช้งาน

2) dbForwardOnly กำหนดยอมให้ผู้ใช้เลื่อนตัวชี้เรคอรด์ (Record Pointer) ไปข้างหน้าได้เพียงทิศทางเดียวเท่านั้น

3) dbSqlPassThrough สำหรับใช้ในการผ่านค่าเอสคิวแอล ที่ใช้สร้างสแน็พชอตไปยัง โอดีบีซีเซิร์ฟเวอร์

3.1.4 การกำหนดตำแหน่งของตัวชี้เรคอรด์ (Record pointer)

ในภาษาวิชวลเบสิค 4.0 มีกลไกฐานข้อมูลในการเปลี่ยนตำแหน่งของตัวชี้เรคอรด์จากเรคอร์ดหนึ่งไปยังอีกเรคอร์ดหนึ่งในเรคอร์ดเซต ดังนี้

1) Move Method เป็นวิธีที่ทำการเปลี่ยนตำแหน่งของตัวชี้เรคอร์ดในเรคอร์ดเซต จากเรคอร์ดหนึ่งไปยังอักเรคอร์ดหนึ่ง

2) Find Method เป็นวิธีที่ทำการกำหนดตำแหน่งของเรคอร์ดที่ตรกับเงื่อนไขของการค้นหา ซึ่งทำงานกับไดนาเซตและสแน็พชอต

3) Seek Method เป็นวิธีในการค้นหาเรคอร์ดแรกในตารางที่ตรงกับเงื่อนไข

4) Bookmark Method เป็นวิธีในการกำหนดตำแหน่งของเรคอร์ดแบบเฉพาะเจาะจง

3.1.4.1 Move Method

ในภาษาวิชวลเบสิค 4.0 เราสามารถเรียกใช้ Move Method ในเรคอร์ดเซตได้ดังนี้

1) Move First เป็นการเปลี่ยนตำแหน่งของตัวชี้เรคอร์ดให้ชี้ไปยังเรคอร์ดแรกของเรคอร์ดเซต

2) Move Next เป็นการเปลี่ยนตำแหน่งของตัวชี้เรคอร์ดปัจจุบัน ไปยังเรคอร์ดถัดไปในเรคอร์เซต ถ้าเป็นเรคอร์ดสุดท้ายจะไม่มีเรคอร์ดถัดไปจะทำการกำหนดแฟลก (EOF Flag)

3) Move Previous เป็นการเปลี่ยนตำแหน่งของตัวชี้เรคอร์ดปัจจุบัน ไปยังเรคอร์ดก่อนหน้านี้ในเรคอร์เซต ถ้าเป็นเรคอร์ดแรกจะไม่มีเรคอร์ดก่อนหน้านี้จะทำการกำหนดแฟลก (BOF Flag)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4) Move Last เป็นการเปลี่ยนตำแหน่งของตัวชี้เรคอร์ดให้ชี้ไปยังเรคอร์ดสุดท้ายของเรคอร์ดเซต

5) Move n เป็นการเปลี่ยนตำแหน่งของตัวชี้เรคอร์ดแบบสัมพัทธ์ตามค่า n ที่กำหนดไว้ เทียบกับตำแหน่งปัจจุบัน ถ้าหากมีการเคลื่อนย้ายตัวชี้เรคอร์ดไปเกินกว่าตำแหน่งเริ่มต้น และสุดท้ายของเรคอร์ดเซตจะทำให้เกิดความผิดพลาดได้

3.1.4.2 Find Method

วิธีสำหรับการกำหนดตำแหน่งของเรคอร์ดที่ตรงกับเงื่อนไขที่กำหนด สามารถใช้ได้เฉพาะกับเรคอร์ดเซตแบบไดนาเซต และสแน็พชอต เราไม่สามารถใช้กับเรคอร์ดเซตแบบตารางได้ ลักษณะการเขียนเงื่อนไข ก็คล้ายกับคำสั่งเอสคิวแอลทั่วไป มี 4 แบบ คือ

1) Find First เริ่มค้นหาจากจุดเริ่มต้นของฐานข้อมูล และค้นหาเรคอร์ดแรกในเรคอร์ดเซตที่ตรงกับเงื่อนไขที่กำหนด

2) Find Next เริ่มค้นหาจากตำแหน่งปัจจุบัน และทำการค้นหาตำแหน่งเรคอร์ดถัดไปที่ตรงกับเงื่อนไข

3) Find Previous เริ่มค้นหาจากตำแหน่งปัจจุบัน และทำการค้นหาตำแหน่งเรคอร์ดก่อนหน้านี้ที่ตรงกับเงื่อนไข

4) Find Last เริ่มค้นหาจากเรคอร์ดสุดท้ายในเรคอร์ดเซต และค้นหาเรคอร์ดสุดท้ายในฐานข้อมูลที่ตรงกับเงื่อนไขที่กำหนด

การทำ Find Method จะทำการตรวจเช็คทุกๆเรคอร์ด เพื่อทำการหาเรคอร์ดที่ตรงกับเงื่อนไข ซึ่งขึ้นอยู่กับขนาดของเรคอร์ดเซต และเงื่อนไขในการค้นหา โดยอินเดกซ์จะเป็นตัวช่วยในการทำการค้นหาได้เร็วยิ่งขึ้น เมื่อทำการค้นหาเรคอร์ดพบ จะทำการเปลี่ยนตำแหน่งตัวชี้เรคอร์ดไปยังตำแหน่งใหม่ที่พบ แต่ถ้าไม่พบจะทำการกำหนดพร็อพเพอร์ตี้ NoMatch และตัวชี้เรคอร์ดยังคงอยู่ ณ ตำแหน่งเดิม

3.1.4.3 Seek Method

วิธี Seek Method นี้เป็นวิธีที่รวดเร็วที่สุดในการกำหนดตำแหน่งเรคอร์ดในตาราง แต่ยังมีข้อจำกัดดังนี้

1) การ Seek สามารถทำได้บนตารางเท่านั้น ไม่สามารถใช้กับไดนาเซต และสแน็พชอต

2) การ Seek สามารถใช้กับอินเดกซ์ที่ใช้งานอยู่เท่านั้น ซึ่งค่าพารามิเตอร์ (Parameter) ในการ Seek ต้องตรงกับฟิลด์ของอินเดกซ์ที่กำลังใช้งานอยู่

3) การ Seek จะทำการค้นหาเฉพาะเรคอร์ดแรกที่ตรงกับเงื่อนไข และค่าในอินเดกซ์เท่านั้น

การ Seek จะทำเริ่มต้นจากเรคอร์ดแรกสำหรับอินเดกซ์ปัจจุบัน และตรวจสอบไปตลอดทั้งอินเดกซ์เพื่อทำการหาเรคอร์ดที่ตรงกับเงื่อนไข สำหรับการเปรียบเทียบโดยใช้ตัวดำเนินการเปรียบเทียบ (Comparison Operator) = , >= , > ,< > ทำการตรวจสอบแบบไปข้างหน้า แต่ถ้าเป็นการเปรียบเทียบโดย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ใช้ตัวดำเนินการ < ,<= จะทำการตรวจสอบแบบย้อนกลับ และถ้าในกรณีที่มีอินเดกซ์มากกว่า 1 ตัวในคีย์ฟิลด์ (Key Field) เดียวกัน การ Seek จะขึ้นอยู่กับตัวดำเนินการเปรียบเทียบ และลำดับของข้อมูลในอินเดกซ์

เมื่อทำการ Seek สำเร็จจะทำให้ตัวชี้เรคอรร์ดเปลี่ยนตำแหน่งไปชี้ยังตำแหน่งของเรคอร์ดใหม่ แต่ถ้าไม่สำเร็จจะทำการกำหนดพร็อพเพอร์ตี้ NoMatch และตำแหน่งของตัวชี้เรคอร์ดไม่เปลี่ยนแปลง

3.1.4.4 BookMark Property

โดยทั่วไปมักจะทำการหาเรคอร์ดเฉพาะ (specific record) ภายหลังจากตัวชี้เรคอร์ดได้เปลี่ยนตำแหน่งไปแล้ว เราสามารถทำการหาเรคอร์ดเฉพาะเหล่านั้นได้โดยการกำหนดพร็อพเพอร์ตี้ bookmark ของเรคอร์ดเซต โดย bookmark เป็นตัวแปรที่กำหนดค่าได้ในระบบซึ่งมีความเกี่ยวข้องกับเรคอร์ดและในแต่ละเรคอร์ดในเรคอร์ดเซตมีเพียงค่าเดียวที่เป็นค่าเฉพาะ

3.1.5 การใช้ฟิลเตอร์ อินเดกซ์ และการจัดลำดับ

ฟิลเตอร์ การจัดลำดับ และอินเดกซ์ เป็นพร็อบเพอร์ตี้ ของเรคอร์ดเซตออบเจกต์ ใช้สำหรับให้ผู้เขียนโปรแกรมควบคุมขอบเขตของเรคอร์ดที่จะใช้ทำงาน และลำดับของเรคอร์ดที่จะเข้าทำงานด้วย โดยฟิลเตอร์จะทำการกำหนดถึงขอบเขตของเรคอร์ดเพื่อเลือกเอาเฉพาะที่ตรงกับเงื่อนไข และอินเดกซ์ (สำหรับตาราง) กับการจัดลำดับ (สำหรับไดนาเซตและสแน็พชอต) ทำการระบุลำดับของเรคอร์ดเซต

3.1.5.1 การกำหนดฟิลเตอร์พร็อพเพอร์ตี้

ฟิลเตอร์พร็อพเพอร์ตี้เหมาะสมกับเฉพาะไดนาเซตและสแน็พชอต เมื่อมีการฟิลเตอร์เกิดขึ้นจะไม่มีผลต่อไดนาเซต หรือสแน็พชอตปัจจุบัน แต่จะทำการคัดลอกไปยังสแน็พชอตหรือไดนาเซตที่สร้างขึ้นในครั้งถัดไป การระบุฟิลเตอร์พร็อพเพอร์ตี้ระบุเหมือน where clause ในภาษาเอสคิวแอล แต่ไม่มีคีย์เวอร์ด where

เงื่อนไขในการฟิลเตอร์จะไม่ส่งผลกระทบต่อไดนาเซตปัจจุบัน แต่จะมีผลกระทบกับไดนาเซตที่สร้างขึ้นใหม่จากไดนาเซตปัจจุบันหลังจากการฟิลเตอร์แล้วเท่านั้น วิธีเดียวที่จะทำการฟิลเตอร์เรคอร์ดเซตปัจจุบันได้ คือ ใช้ move ตลอดทั้งเรคอร์ดเซตและใช้ find เพื่อหาเรคอร์ที่ตรงกับเงื่อนไข ถ้าใช้งานเฉพาะฟิลเตอร์ไดนาเซตเราสามารถนำเงื่อนไขในการฟิลเตอร์กำหนดลงในคำสั่งเอสคิวแอล ขณะ open เรคอร์ดเซตจะทำให้ประสิทธิภาพดีกว่า

3.1.5.2 การกำหนดพร็อพเพอร์ตี้การจัดลำดับ

ลักษณะการกำหนดเช่นเดียวกับการกำหนดฟิลเตอร์พร็อพเพอร์ตี้ที่สามารถใช้ได้เฉพาะไดนาเซ็ตและสแน็พชอต ซึ่งพร็อพเพอร์ตี้การจัดลำดับระบุโดยชื่อฟิลด์และลักษณะการจัดลำดับข้อมูล (ascending or descending) ในฟิลด์นั้นๆ เมื่อต้องการทำการจัดลำดับที่ขึ้นอยู่กับหลายๆฟิลด์แล้วลำดับของฟิลด์ย่อมมีความสำคัญ และเมื่อใช้พร็อพเพอร์ตี้การจัดลำดับจะไม่ส่งผลกระทบต่อไดนาเซตหรือสแน็พช็อตปัจจุบัน แต่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะส่งผลต่อไดนาเซ็ตหรือสแน็พช็อตที่สร้างขึ้นจากไดนาเซ็ตปัจจุบัน เราสามารถใช้ order by ในภาษาเอสคิวแอลแทนพร็อพเพอร์ตี้การจัดลำดับได้ ซึ่งจะให้ผลเหมือนกัน

3.1.5.3 การกำหนดอินเดกซ์ปัจจุบันในตาราง

อินเดกซ์ใช้งานกับตารางเพื่อกำหนดลำดับให้แก่เรคอร์ด หรือใช้ทำงานกับ Seek Method เพื่อทำการหาเรคอร์ดได้อย่างรวดเร็ว โดยการใช้อินเดกซ์พร็อพเพอร์ตี้ของตารางจำเป็นต้องกำหนดชื่อให้ตรงกับอินเดกซ์ที่มีอยู่จริงในตารางด้วย ถ้าต้องการใช้งานอินเดกซ์ที่ยังไม่มีอยู่จะต้องทำการสร้างอินเดกซ์นั้นขึ้นมาก่อน

3.1.6 การพิจารณาโปรแกรมที่เปลี่ยนแปลงหลายๆเรคอร์ด

การทำงานกับฐานข้อมูลมักจะเป็นการหาข้อมูลจากฐานข้อมูลนั้น ซึ่งโดยทั่วไปแล้วการทำงานของโปรแกรมหรือฟังก์ชั่นในแอพพลิเคชั่นจะทำงานกับเรคอร์ดมากกว่าหนึ่งเรคอร์ดจากฐานข้อมูลลักษณะเช่นนี้มีวิธีในการนำเรคอร์ดจากตารางได้ดังนี้

1. โปรแกรมลูป คือ การใช้คำสั่ง Do...While หรือ Do until ทำการวนลูปจนกว่าจะตรงเงื่อนไขการออกจากลูป นอกจากนี้แล้วยังมีในลักษณะของอิมไพล์ลูป (Impiled loop) โดยในฟอร์มจะปรากฏปุ่มคำสั่งที่ใช้สำหรับเปลี่ยนตำแหน่งเรคอร์ดเป็นเรคอร์ดถัดไป หรืออาจจะเป็นเรคอร์ดก่อนหน้านี้ จุดที่พิจารณาสำคัญในการทำงานกับเรคอร์ด คือ ถ้าต้องการเคลื่อนที่ไปข้างหลังจากเรคอร์ดแรกหรือ เคลื่อนที่ไปข้างหน้าจากเรคอร์ดสุดท้ายหรือ พยายามเคลื่อนที่ไปยังตำแหน่งที่เป็นเรคอร์ดเซตว่างเปล่าจะทำให้เกิดความผิดพลาด ดังนั้น เจทเอนจิ้น จึงมีเรคอร์ดเซตพร็อพเพอร์ตี้บางอย่างเพื่อใช้บอกและป้องกันเหตุการณ์เหล่านี้ได้

BOF เป็นแฟลกแสดงจุดเริ่มต้นของไฟล์ มีค่าเป็นจริงเมื่อตัวชี้เรคอร์ดชี้อยู่ที่เรคอร์ดแรก และมีค่าเป็นเท็จเมื่อไม่ใช่

EOF เป็นแฟลกแสดงจุดสิ้นสุดของไฟล์ มีค่าเป็นจริงเมื่อตัวชี้เรคอร์ดชี้อยู่ที่เรคอร์ดสุดท้ายและมีค่าเป็นเท็จเมื่อไม่ใช่

Record Count เป็นค่าแสดงจำนวนเรคอร์ดที่สามารถเข้าถึงได้ โดยจะได้จำนวนเรคอร์ดทั้งหมดเมื่อเข้าถึงเรคอร์ดสุดท้ายในเรคอร์ดเซต

Nomatch เป็นแฟลกแสดงถึงความสำเร็จในการใช้ find method หรือ seek method ค่าแฟลกจะเป็นจริงเมื่อไม่สำเร็จ นอกนั้นจะมีค่าเป็นเท็จ

2. การใช้ชุดคำสั่งภาษาเอสคิวแอล

ในการทำงานกับเรคอร์ดโดยใช้โปรแกรมลูปสามารถใช้ชุดคำสั่งภาษาเอสคิวแอลเพื่อรองรับจำนวนฟังก์ชั่นที่ทำงานกับเรคอร์ดหลายๆเรคอร์ดได้โดยลักษณะของฟังก์ชั่นมีสองแบบหลักๆก็คือ

2.1 การคิวรีแบบคำนวณ เป็นการคำนวณของกลุ่มเรคอร์ดที่ร้องขอ เช่น การคำนวณหาค่าผลรวม ค่าสูงสุด ค่าต่ำสุด ค่าเฉลี่ย และจำนวนเรคอร์ด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2 การคิวรีแบบทำงาน (Action Cuery) เป็นการทำงานกับเรคอร์ดในเรคอร์ดเซตโดยตรง เช่น เพิ่ม (Insert) ลบ (Delete) และ เปลี่ยนแปลง (Modify) กลุ่มของเรคอร์ดที่ตรงกับเงื่อนไข

3.1.7 การทำความเข้าใจกับชุดคำสั่งโปรแกรมอื่นๆ

ในโปรแกรมส่วนใหญ่มักจะทำงานกับเรคอร์ด ได้แก่ เพิ่ม ลบ และเปลี่ยนแปลงเรคอร์ด ซึ่งคำสั่งเหล่านี้ใช้กับเฉพาะตารางและไดนาเซ็ตเท่านั้น ไม่ใช้กับสแน็พช็อต เพราะว่าสแน็พช็อตทำการแก้ไขข้อมูลไม่ได้

3.1.7.1 การใส่เรคอร์ดใหม่ (Insert)

ตัวอย่างวิธีการใส่เรคอร์ดใหม่

```
MyDN.AddNew
MyDN!No. = 1
***** ทำการกำหนดข้อมูลในฟิลด์ต่างๆของเรคอร์ดใหม่จนครบ ****
MyDN.Update
```

Addnew ยังไม่ได้ทำการเพิ่มเรคอร์ดในเรคอร์ดเซตโดยทันทีเป็นเพียงแค่ทำการกำหนดพื้นที่เพื่อรับข้อมูลของเรคอร์ดใหม่ที่จะทำการใส่เรคอรด์ลงไปเท่านั้น โดยจะเพิ่มลงในเรคอร์ดเซตก็ต่อเมื่อใช้ update

3.1.7.2 การแก้ไขเรคอร์ด (Edit)

ใช้สำหรับทำการเปลี่ยนแปลงค่าในเรคอร์ดและเหมือนกับ Addnew คือ จะมีผลเมื่อทำการ update

```
MyDN.Edit
MyDN!Name = Test
***** ทำการระบุค่าในฟิลด์ที่ต้องการแก้ไขข้อมูล *****
MyDN.Update
```

Update มักใช้ร่วมกับ Addnew และ Edit ในระบบมัลติยูสเซอร์ โดยอัพเดทเมธอด (Update Method) จะทำการปลดล็อคของเรคอร์ดที่เกี่ยวข้องเมื่อทำการอัพเดทเสร็จเรียบร้อย ในกรณีที่ใช้ดาต้าคอนโทรลที่ทำงานกับเรคอร์ดเซตก็ไม่จำเป็นต้องอัพเดทเรคอร์ด เพราะจะทำการอัพเดทเองโดยอัตโนมัติ เมื่อมีการประมวลผลดาต้าคอนโทรล

3.1.7.3 การลบเรคอร์ด (Delete)

การลบเรคอร์ดออกจากเรคอร์ดเซต และทำการกำหนดค่าตัวชี้เรคอร์ดให้เป็นค่า null

```
MyDN.Delete
```

เราสามารถทำการเรียกข้อมูลที่ถูกลบไปแล้วกลับคืนมาได้ในกรณีที่ใช้ begin trans ก่อนทำการลบ และใช้คำสั่ง roll back เมื่อต้องการเรียกข้อมูลเรียกข้อมูลกลับคืน นอกจากนี้ยังมีอีกวิธีหนึ่งคือ ทำการใส่เรคอร์ดนั้นซ้ำลงไปอีกทีหนึ่ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.1.8 การประมวลผลทรานสแอคชัน (Transaction Processing)

การประมวลผลทรานสแอคชัน เป็นการกำหนดกลุ่มการทำงาน เช่น เพิ่ม ลบ หรือ เปลี่ยนแปลงข้อมูลเป็นชุดๆ และสามารถแน่ใจได้ว่าการทำงานเสร็จสมบูรณ์ทั้งหมด แต่ถ้าไม่สามารถทำให้เสร็จสมบูรณ์ได้ก็เสมือนกับว่าไม่ได้ทำอะไรมาตั้งแต่ต้นเพื่อรักษาความถูกต้องของข้อมูล ในภาษาวิชวลเบสิค 4.0 มีคำสั่งที่เกี่ยวกับการทำทรานสแอคชัน ได้แก่

1. BeginTrans เป็นการกำหนดการเริ่มต้นของทรานสแอคชัน ใช้เพื่อระบุให้ระบบจัดการฐานข้อมูลได้ทราบว่าทรานสแอคชันได้เริ่มต้นขึ้นแล้ว

2. RollBack เป็นการกำหนดจุดสิ้นสุดของทรานสแอคชัน โดยกลับสู่จุดเริ่มต้นก่อนการทำทรานสแอคชัน และยกเลิกการเปลี่ยนแปลงทั้งหมดในทรานสแอคชัน ใช้ในกรณีที่เกิดความผิดพลาดในระหว่างการทำทรานสแอคชัน

3. CommitTrans เป็นการกำหนดจุดสิ้นสุดของทรานสแอคชัน โดยถือว่าการเปลี่ยนแปลงค่าในฐานข้อมูลในทรานสแอคชันถูกต้อง ให้นำค่าที่เปลี่ยนแปลงบันทึกลงในฐานข้อมูลได้เลย เมื่อทำการ commit tran แล้วจะไม่สามารถทำการยกเลิกการเปลี่ยนแปลงนั้นได้

3.1.9 ระดับการล็อคในฐานข้อมูล (Database Lock Level)

ในวิชวลเบสิคมีความสามารถจัดการเกี่ยวกับความสามารถในการเข้าถึงข้อมูล โดยกำหนดระดับการเข้าถึงข้อมูลไว้ 3 ระดับ คือ

3.1.9.1 การล็อคในระดับฐานข้อมูล (Database Locking)

การล็อคแบบนี้ทำเมื่อต้องการใช้ฐานข้อมูลนั้นแต่เพียงผู้เดียว ซึ่งเมื่อมีการล็อคฐานข้อมูลนั้นแล้ว ผู้อื่นไม่สามารถใช้ข้อมูลใดๆภายใต้ฐานข้อมูลนั้นได้เลยจนกว่าจะมีการปลดล็อคนั้น โดยการสั่งล็อคฐานข้อมูลได้โดยกำหนดในขั้นตอนการ Opendatabase โดยกำหนดค่าตัวเลือก Exclusive ของคำสั่ง Opendatabase เป็น True แต่ถ้าหากไม่กำหนดตัวเลือก Exclusive จะถือว่าเป็นการเปิดฐานข้อมูลเพื่อใช้งานร่วมกับผู้อื่น

3.1.9.2 การล็อคในระดับตาราง (Table Locking)

เมื่อเรายอมให้มีการใช้ฐานข้อมูลร่วมกันได้แล้ว หากต้องการล็อคตารางใดตารางหนึ่งที่อยู่ในฐานข้อมูลเพื่อใช้งานเพียงผู้เดียว เราจะต้องกำหนดผ่านคำสั่งที่ใช้ในการสร้างไดนาเซตออบเจคต์เท่านั้น จะไม่ใช้กับการสร้างสแน็พชอตออบเจคต์ก็เพราะเราไม่สามารถจะแก้ไขข้อมูลผ่านทางสแน็พชอตออบเจคต์ได้

ตารางที่ล็อค ต้องเป็นตารางที่เป็นฐานของไดนาเซตออบเจคต์นั้นๆ ซึ่งอาจเป็นเพียงหนึ่งตารางหรือมากกว่าก็ได้ ขึ้นอยู่กับว่าไดนาเซตนั้นถูกกำหนดขึ้นจากตารางใดบ้าง วิธีการกำหนดล็อคบนตารางนั้นทำโดยกำหนดตัวเลือกว่าไม่ต้องการจะใช้ข้อมูลร่วมกับผู้อื่นในฟังก์ชันการทำงาน CreateDynaset

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ถ้าหากเราสร้างไดนาเซตออบเจคต์ โดยไม่ได้กำหนดค่าตัวเลือกเพื่อกันไม่ให้ผู้อื่นใช้ตารางร่วมได้ ก็จะไม่มีการล็อกในระดับตารางที่ประกอบขึ้นเป็นไดนาเซตนั้น

3.1.9.3 การล็อคในระดับเพจ (Page Locking)

การล็อคในระดับเพจนั้น เจทเอนจิ้น (Jet Engine) จะทำให้เองโดยอัตโนมัติ ถึงแม้ว่าเราจะไม่ได้กำหนดไว้ก็ตาม ซึ่งเจทเอนจิ้นจะทำการล็อคข้อมูลเฉพาะส่วนที่อยู่ในเพจเดียวกันกับข้อมูล หรือเรคอร์ดที่เรากำลังจะแก้ไข

การสั่งล็อคในระดับนี้มี 2 แบบ คือ

1. Pessimistic Locking เป็นการสั่งล็อคข้อมูลในเพจทันทีที่เราสั่งว่าต้องการแก้ไขข้อมูล คือหลังจากคำสั่ง Edit

2.Optimistic Locking เป็นการสั่งล็อคในจังหวะสุดท้ายของการใช้คำสั่งแก้ไขข้อมูล โดยจะไม่ล็อคทันทีที่สั่งแก้ไขข้อมูล ลักษณะนี้เป็นการล็อคในระดับต่ำที่สุด คือยอมให้มีการใช้ข้อมูลร่วมกันมากที่สุดแล้ว

การกำหนดแบบการล็อคระดับเพจ ให้กำหนดที่พร็อพเพอร์ตี้ LockEdit ของตัวแปรไดนาเซตออบเจคต์นั้น ซึ่งหากมีค่าเป็น True ก็จะหมายความว่าต้องการให้ล็อคแบบ Pessimistic และหากเป็น False ก็จะหมายความว่าต้องการให้เป็นการให้ล็อกแบบ Optimistic โดยขอบเขตของการล็อคทั้ง 2 แบบนี้จะสิ้นสุดหลังจากที่สั่งฟังก์ชันการทำงานอัพเดทเสร็จแล้ว

ข้อควรพิจารณาในการเลือกใช้ล็อค

1. ระดับการล็อคจะมากน้อยเพียงใด ซึ่งยิ่งล็อกในระดับที่สูงมากก็จะยิ่งทำให้การใช้ข้อมูลร่วมกับผู้อื่นได้น้อยลง คือมีโอกาสสูงที่ผู้ใช้หรือโปรแกรมอื่นๆ ต้องหยุดรอในระหว่างที่โปรแกรมเราล็อคข้อมูลไว้

2. ขอบเขตของการล็อคจะยาวนานเพียงใด ยิ่งล็อคนานมากขึ้น โอกาสที่จะให้ผู้อื่นใช้งานร่วมด้วยก็น้อยลงเช่นกัน หากเราใช้ทรานสแอคชันแล้ว ขอบเขตของการล็อคก็จะเท่ากับขอบเขตของทรานสแอคชัน ยิ่งในทรานสแอคชันมีการแก้ไขข้อมูลมากขึ้น ก็จะทำให้จำนวนเพจที่ถูกล็อคก็จะมากยิ่งขึ้นด้วย ก็จะยิ่งทำให้ผู้อื่นใช้ข้อมูลร่วมกับเราน้อยลง

3. จุดประสงค์ในการใช้ล็อคเพื่อให้ข้อมูลมีความถูกต้องเหมาะกับการใช้งาน พร้อมๆกับยอมให้มีการใช้ข้อมูลร่วมกันได้มากขึ้น เราควรกำหนดการล็อคให้อยู่ในระดับที่เหมาะสมกับการใช้งานข้อมูลในโปรแกรม เพื่อเปิดโอกาสให้ผู้อื่นได้ใช้ข้อมูลร่วมกันได้มากขึ้น ทำให้งานโดยรวมเสร็จเร็วขึ้น

3.1.10 การทำงานกับฐานข้อมูลโดยใช้วิชวลเบสิค 4.0

ในวิชวลเบสิคสามารถทำการติดต่อกับฐานข้อมูลได้หลายทาง เช่น อาศัยดาต้าคอนโทรล (Data Control) อาศัยดาต้าออบเจคต์ (Data Object) อาศัยรีโมทดาต้าออบเจคต์ (Remote Data Object) หรือใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ประกอบกัน การทำงานกับฐานข้อมูลหลักๆ ได้แก่ การอินเสิร์ท (Insert) การอัพเดท (Update) และการดีลีท (Delete) นอกจากนี้ยังมี การสร้างตาราง การสร้างอินเดกซ์ และอื่นๆอีก

3.10.1 การแทรกเรคอร์ดข้อมูล (Insert)

วิธีการในการแทรกเรคอร์ดข้อมูลลงในตาราง โดยใช้วิชวลเบสิคสามารถทำได้ 2 ทางคือ ใช้คำสั่งภาษาเอสคิวแอลโดยตรง และทำการอินเสิร์ทผ่านเรคอร์ดเซต โดยใช้แอดนิวเมธอด (AddNew Method)ตามด้วยการกำหนดข้อมูลในฟิลด์ต่างๆที่จะทำการอินเสิร์ท และใช้อัพเดทเมธอด (UpdateMethod)

3.10.2 การแก้ไขเรคอร์ดข้อมูล (Update)

วิธีการในการเปลี่ยนแปลงแก้ไขเรคอร์ดข้อมูลในตารางทำได้ 2 ทางคือ ใช้คำสั่งภาษาเอสคิวแอลโดยตรง และทำการอัพเดทผ่านเรคอร์ดเซต โดยใช้อีดิทเมธอด (Edit Method) และทำการกำหนดค่าฟิลด์ต่างๆที่ต้องการเปลี่ยนแปลงในเรคอร์ดและใช้อัพเดทเมธอด (Update Method)

3.10.3 การลบเรคอร์ดข้อมูล (Delete)

วิธีการในการลบเรคอร์ดข้อมูลในตาราง สามารถทำได้โดยใช้คำสั่งภาษาเอสคิวแอลโดยตรง หรืออาจทำได้โดยใช้ดีลีทเมธอด (Delete Method) ของเรคอร์ดเซต

นอกเหนือจากนี้ก็สามารถใช้คำสั่งภาษาเอสคิวแอลได้โดยตรง แล้วตามด้วยเอกซ์ซิคิวท์เมธอด (Execute Method)

3.2 ลักษณะสำคัญและวิธีการใช้เดลไฟล์ในการพัฒนาโปรแกรม

ภาษาเดลไฟล์ เป็นอีกภาษาหนึ่งที่นิยมใช้เขียนแอพลิเคชันโปรแกรมในปัจจุบัน เนื่องจากใช้งานได้สะดวก ศึกษาและเข้าใจได้ง่าย มีเครื่องมือต่างๆคอยสนับสนุน ในที่นี้จึงนำมาใช้ในการพัฒนาโปรแกรมทำงานกับข้อมูล มีลักษณะสำคัญดังนี้

3.2.1 โครงสร้างเดลไฟล์ในการติดต่อกับมิดเดิ้ลแวร์

3.2.1.1 บอร์แลนด์ดาต้าเบสเอนจิ้น (Borland Database Engine:BDE)

บีดีอี(BDE) เป็นส่วนมิดเดิลแวร์ ส่วนแรกของเดลไฟล์ (Delphi) ซึ่งทำหน้าที่เป็นกลไกฐานข้อมูล (DBEngine)และทำหน้าที่เชื่อมต่อไปยังส่วนเซิร์ฟเวอร์ โดยบีดีอีจะมีเอพีไอ (API) พื้นฐาน เพื่อให้ผู้พัฒนาไม่จำเป็นต้องสร้างแอพพลิเคชั่นสำหรับดาต้าซอส (DataSource) แต่ละตัว ผู้พัฒนาเพียงแต่เรียกใช้เอพีไอให้ถูกต้องเพื่อทำการติดต่อกับดาต้าซอสเท่านั้น เมื่อเปลี่ยนดาต้าซอสก็ไม่จำเป็นต้องแก้ไขส่วนแอพพลิเคชั่นมากนัก ในเดลไฟล์สามารถเรียกเอพีไอเหล่านี้เพื่อทำงานได้แต่มักไม่จำเป็นเพราะส่วนประกอบต่างๆของเดลไฟล์ทำงานโดยผ่านบีดีอีเอพีไออยู่แล้ว ดังนั้นการเรียกเอพีไอจะทำในกรณีที่เป็นการทำงานเฉพาะ ซึ่งส่วนประกอบในเดลไฟล์ไม่สามารถทำได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

IDAPI32.DLL	ส่วนประกอบหลักของบีดีอี
IDQBE32.DLL	ตัวประมวลผลคิวรีแบบคิวบีอี
IDSQL32.DLL	ตัวประมวลผลคิวรีแบบเอสคิวแอล
IDODBC32.DLL	ส่วนเชื่อมต่อกับโอดีบีซีไดร์ฟเวอร์ (ODBC SOCKET DRIVER)
IDR20009.DLL	ข้อมูลของข้อความแสดงความผิดพลาด
BDECFG32.EXE	ชุดคำสั่งที่ใช้จัดการตัวเลือกต่างๆของบีดีอี
BDECFG32.HLP	ข้อมูลความช่วยเหลือ
IDAPI32.CFG	เก็บตัวเลือกต่างๆของบีดีอี

ตารางที่ 3-3 แสดงแฟ้มข้อมูลสำคัญของบีดีอี

การใช้งานเอพีไอโดยตรงในเดลไฟล์สามารถทำได้โดยกำหนดประโยค "USES UNIT" ชื่อว่า BDE ในส่วนของโปรแกรมหลังจากนั้นก็จะสามารถเรียกใช้เอพีไอได้โดยตรง

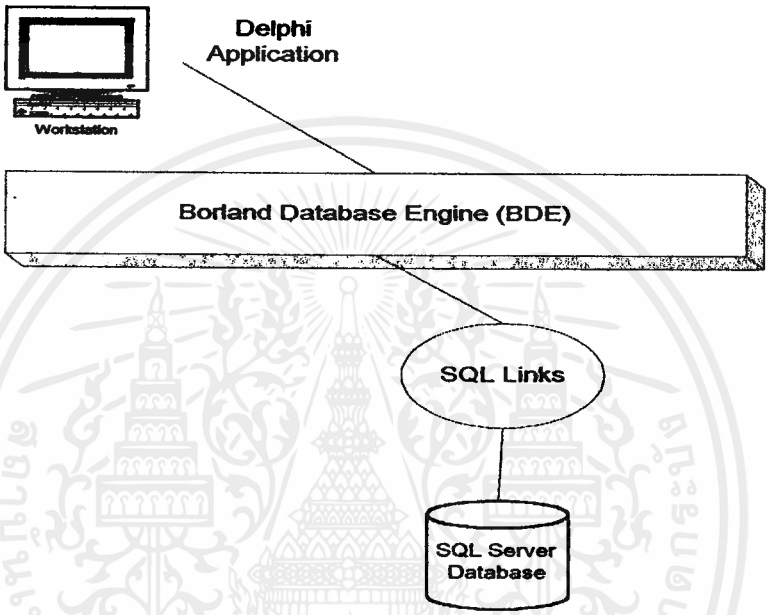
DbiCloseDatabase	ปิดฐานข้อมูลที่กำลังติดต่ออยู่
DbiOpenDatabase	เปิดฐานข้อมูล
DbiGetDatabaseDesc	อ่านข้อมูลตัวเลือกของฐานข้อมูลที่กำหนดจากตัวเลือก
DbiOpenDatabaseList	เรียกดูรายชื่อของฐานข้อมูลที่เปิดอยู่
DbiOpenTableList	เรียกดูรายชื่อของตารางที่เปิดอยู่
DbiCreateTable	สร้างตาราง
DbiDeleteTable	ลบตาราง
DbiOpenFieldList	เรียกดูรายชื่อของฟิลด์
DbiQExecDirect	สั่งให้คิวรีทำงาน
DbiAppendRecord	เพิ่มแถวข้อมูลว่างต่อท้ายตาราง
DbiDeleteRecord	ลบแถวข้อมูลในตาราง
DbiGetField	อ่านข้อมูลจากฟิลด์

ตารางที่ 3-4 แสดงบีดีอีเอพีไอบางส่วน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.1.2 เอสคิวแอลลิงค์ (SQL LINK)

ในการติดต่อกับเอสคิวแอลเซิร์ฟเวอร์ นอกจากใช้บีดีอีแล้ว ยังคงต้องใช้อีกส่วนหนึ่งคือ เอสคิวแอลลิงค์(SQL Link) ซึ่งทำหน้าที่ติดต่อกับเซิร์ฟเวอร์ โดยในเดลไฟล์ไคล์เอนท์เซิร์ฟเวอร์เอดิชั่น (Client/Server Edition) มีเอสคิวแอลลิงค์ของเซิร์ฟเวอร์หลายตัว เช่น ออราเคิล ไมโครซอฟท์เอสคิวแอลเซิร์ฟเวอร์ ไซเบสเอสคิวแอล (Sybase SQL) นอกจากนี้ยังมีเอสคิวแอลลิงค์ที่เป็นโอดีบีซีซ็อคเก็ต (ODBC Socket) ซึ่งหมายถึงเราสามารถนำโอดีบีซีไดร์ฟเวอร์มาเป็นเอสคิวแอลลิงค์ได้



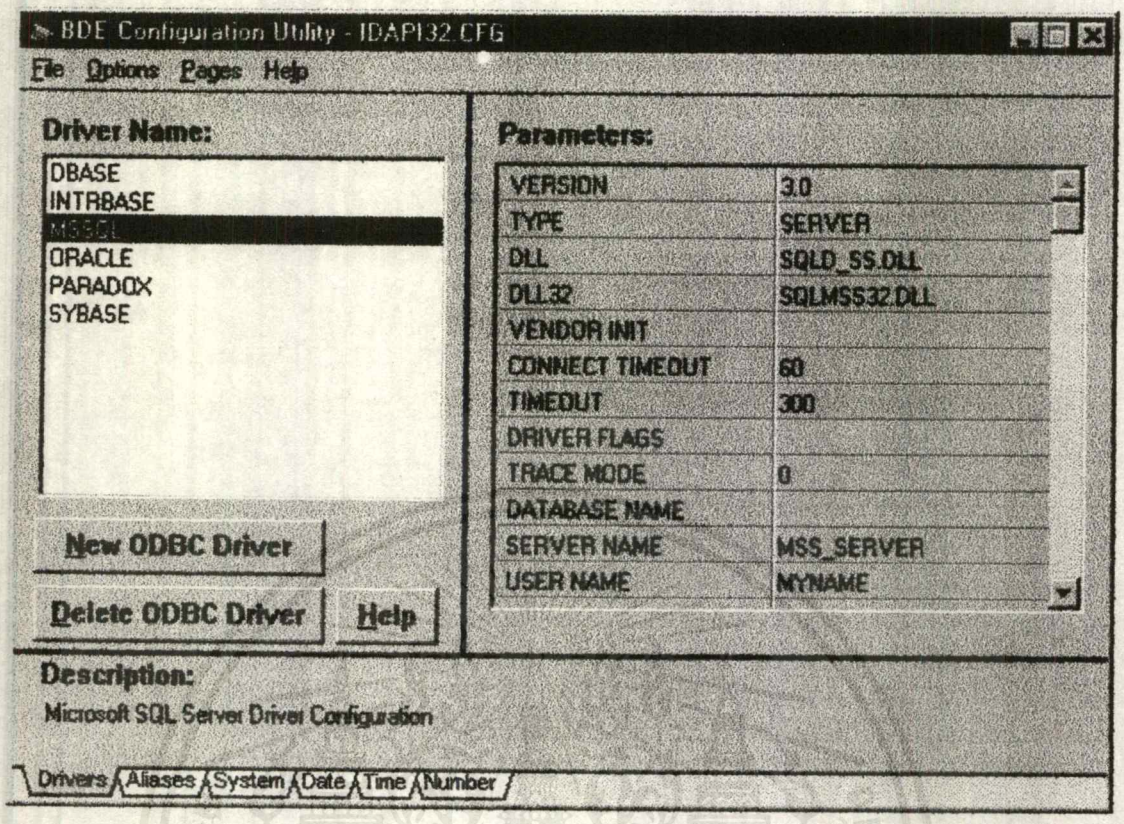
รูปที่ 3-4 แสดงความสัมพันธ์ของเดลไฟล์และเอสคิวแอลลิงค์

3.2.1.3 การกำหนดค่าบีดีอี (Configuration BDE)

ในการใช้งานเดลไฟล์ เพื่อติดต่อกับเอสคิวแอลเซิร์ฟเวอร์จำเป็นต้องมีการปรับตั้ง และกำหนดบีดีอีเพื่อให้ถูกต้องเหมาะสมกับการใช้งาน

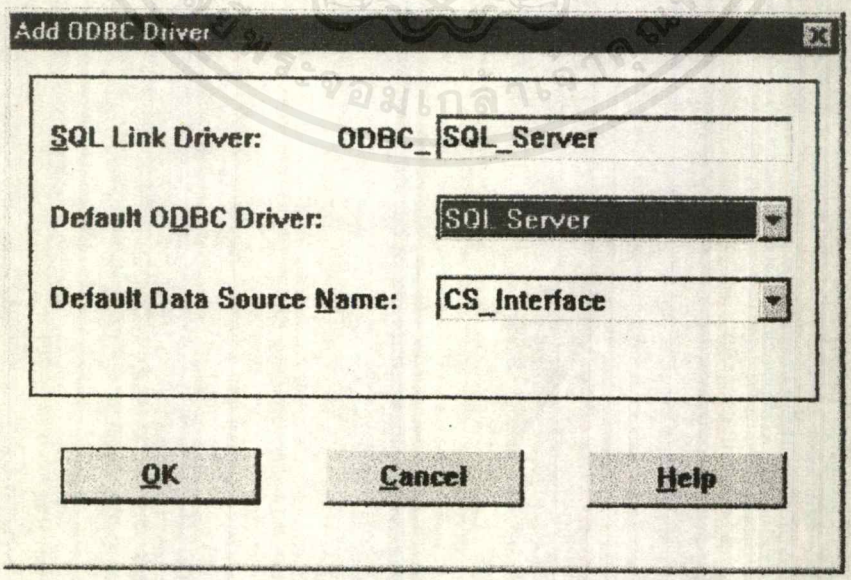
1. ทำการเรียกโปรแกรมบีดีอีคอนฟิก (BDE Config)
2. ตรวจสอบไดร์ฟเวอร์ที่จะใช้งานโดยดูจากหน้าไดร์ฟเวอร์ของบีดีอีคอนฟิก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



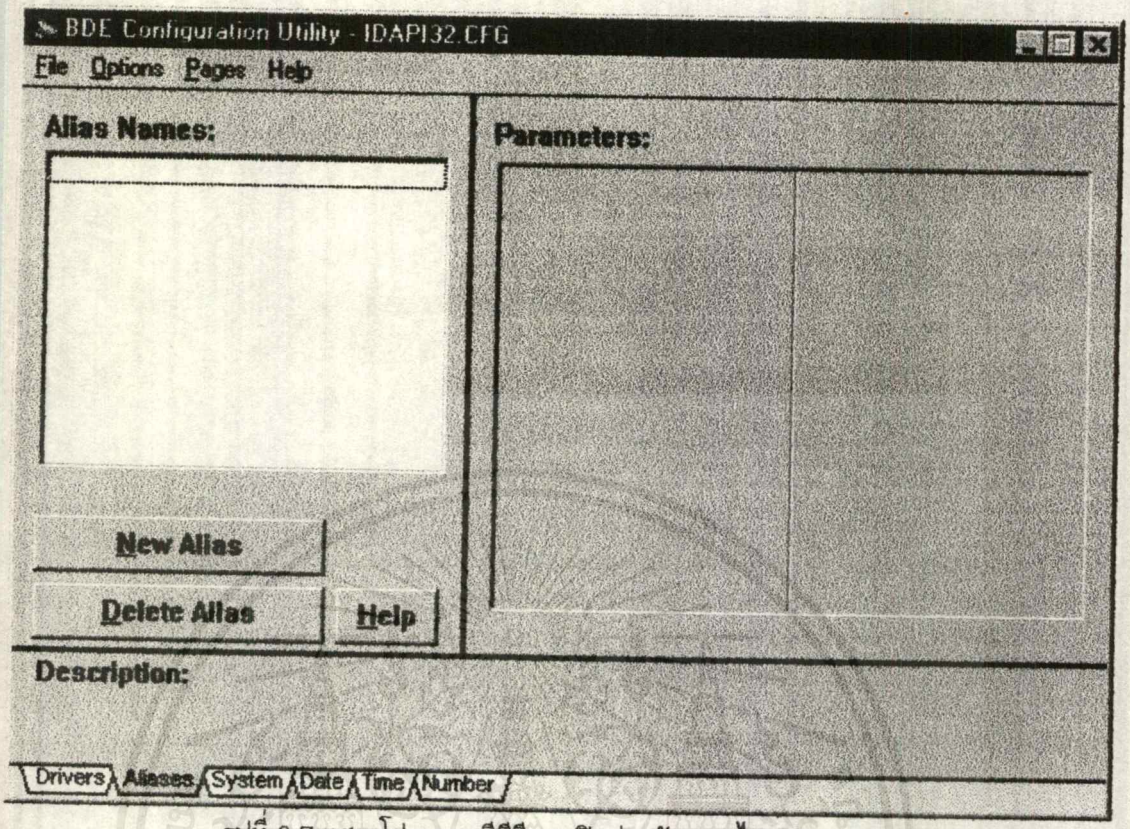
รูปที่ 3-5 แสดงโปรแกรมบีดีอีคอนฟิก

3. ถ้าต้องการใช้โอดีบีซีไดร์ฟเวอร์ เป็นเอสคิวแอลลิงค์ ให้ทำการเลือก New OBDC Driver จากนั้นใส่ชื่อที่ต้องการ แล้วเลือกโอดีบีซีไดร์ฟเวอร์และชื่อดาต้าซอสให้เรียบร้อย (ไดร์ฟเวอร์ที่จะสามารถใช้งานได้ต้องมีการกำหนดชื่อของดาต้าซอสมาก่อน โดยใช้โอดีบีซีแอดมินิสเทร์เตอร์ (ODBC Administrator))



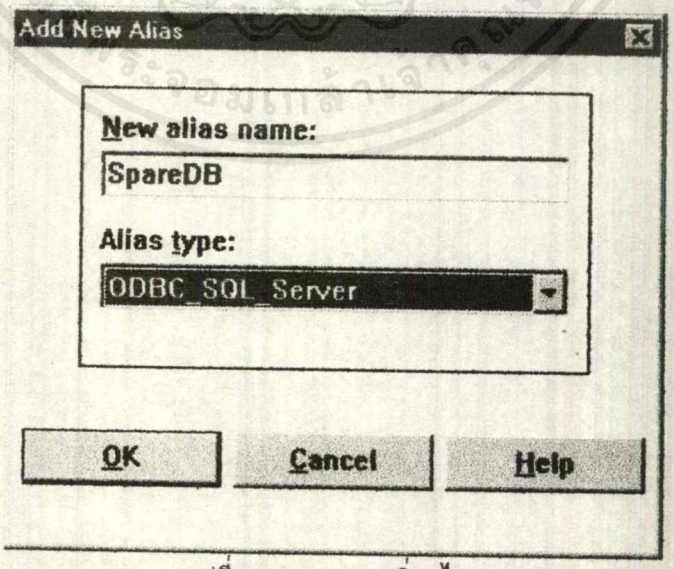
รูป 3-6 แสดงการเพิ่มโอดีบีซีดาต้าซอส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3-7 แสดงโปรแกรมบีดีอีคอนฟิกส่วนจัดการอไลแอส

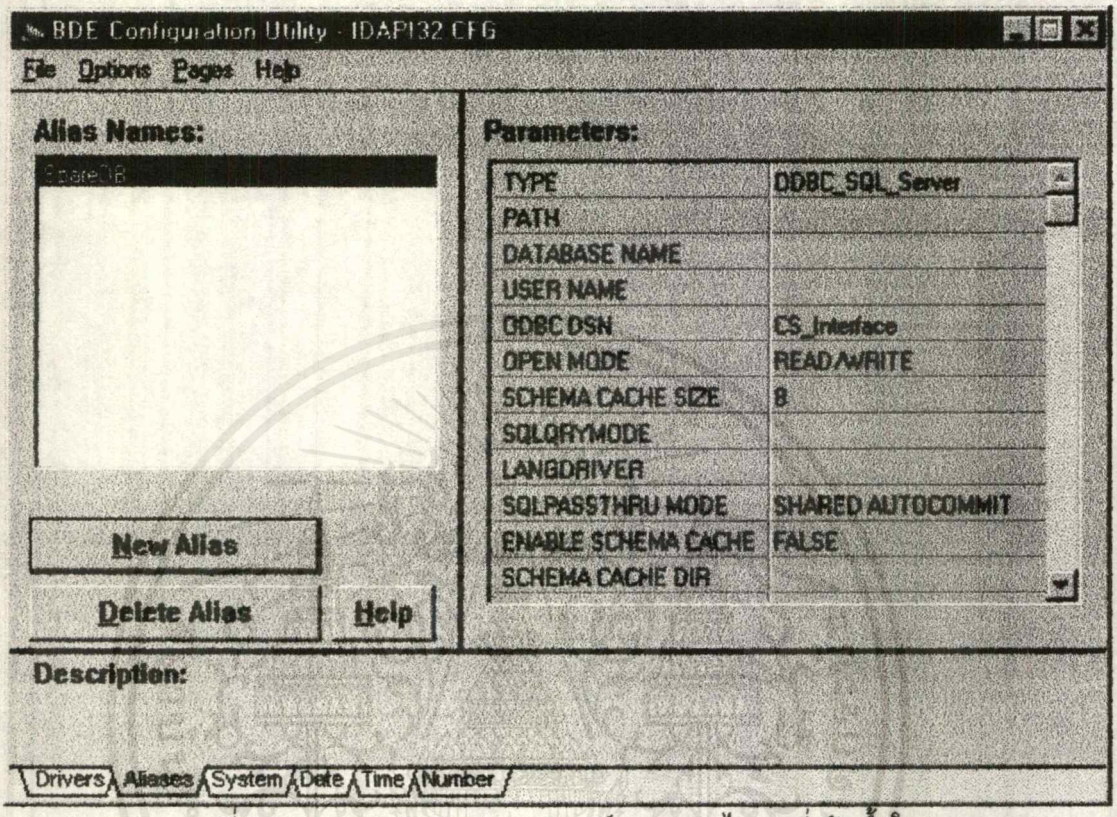
4. ทำการเลือกที่หน้าอไลแอส (Aliases) เพื่อทำการเพิ่มอไลแอสโดย อไลแอสคือชื่อที่ใช้อ้างอิงถึงเอสคิวแอลลิงค์ และฐานข้อมูลที่ต้องการติดต่อด้วย เลือก New Alias จากนั้นใส่ชื่ออไลแอสที่ต้องการแล้วเลือกชนิดของอไลแอสตามไดร์ฟเวอร์ที่ต้องการ ซึ่งได้เลือกไว้ในส่วนหน้าไดร์ฟเวอร์



รูปที่ 3-8 แสดงการเพิ่มอไลแอส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5. ทำการกำหนดค่าพารามิเตอร์ต่างๆของอไลแอสที่สร้างขึ้นใหม่ให้ถูกต้อง เช่น ชื่อฐานข้อมูล ชื่อผู้ใช้ระบบ



รูปที่ 3-9 แสดงการกำหนดค่าพารามิเตอร์ต่างๆของอไลแอสที่สร้างขึ้นใหม่

ความหมายของพารามิเตอร์ต่างๆที่สำคัญ

Type	ชนิดของ Alias นั้นใน Microsoft SQL แต่ถ้าเป็น ODBC ก็จะเป็นชื่อ Driver ที่เรากำหนดไว้
Database Name	ชื่อของ Database ที่จะใช้งานผ่าน Alias นี้ แต่ถ้าเป็น ODBC Driver ก็ไม่ต้องกำหนดเพราะถูกกำหนดขณะทำการ Set Datasource แล้ว
User Name	ชื่อ User ซึ่งใช้ทำการ Login เข้าสู่ Database Server
Open Mode	รูปแบบของการเปิดข้อมูล คือ เพื่ออ่าน เพื่อเขียน หรือเพื่ออ่านและเขียน
SQL Query Mode	บอกว่าจะทำการประมวลผล Query ณ ที่ใด แต่ถ้าไม่กำหนดก็ให้ถือว่าส่งไปที่ Server ก่อนและถ้าประมวลผลไม่ได้ก็จะทำการประมวลผลที่ Client แทน หรือทำการกำหนดให้ประมวลผลที่ Client หรือ Server ได้เลย

ตารางที่ 3-5 แสดงพารามิเตอร์ต่างๆที่สำคัญของอไลแอส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ค่าต่างๆเหล่านี้เราอาจยังไม่ต้องกำหนดก็ได้ เพราะการกำหนดสามารถทำได้ผ่านส่วนประกอบของฐานข้อมูล (Database Component) ได้เช่นกัน

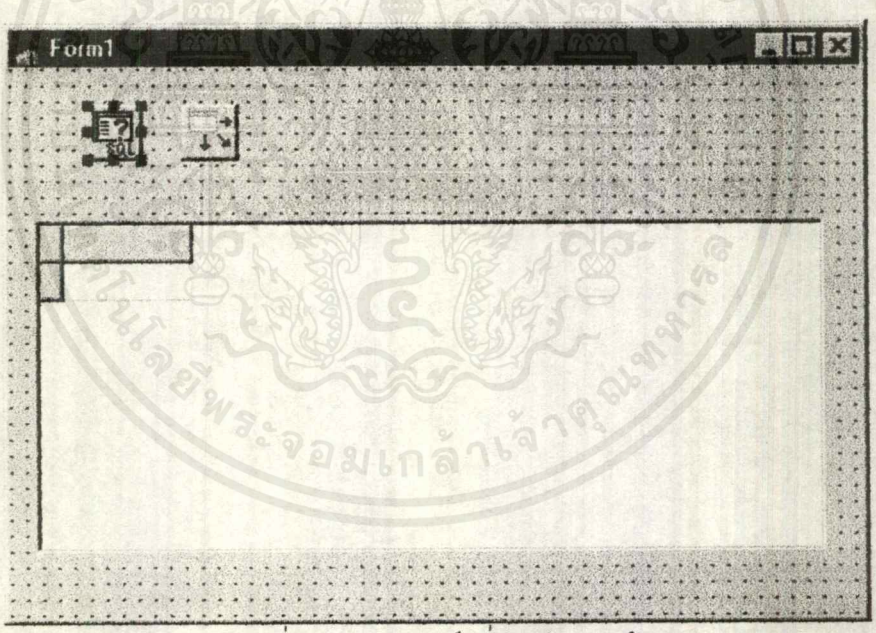
3.2.2 รายละเอียดของส่วนประกอบต่างๆในเดลไฟล์

3.2.2.1 คิวรีคอมโพเนนท์ (Query Component)

คิวรีคอมโพเนนท์ เป็นคอมโพเนนท์หลักที่ใช้ติดต่อกับฐานข้อมูล โดยใช้คำสั่งเอสคิวแอล สามารถใช้เพื่อทำการดึงข้อมูลจากฐานข้อมูล รวมทั้งใช้ส่งคำสั่งเอสคิวแอลไปยังดาต้าเบสเซิร์ฟเวอร์ สามารถดึงข้อมูลโดยทำการรวมข้อมูลจากตารางหลายๆตาราง

การใช้งานคิวรีคอมโพเนนท์อย่างง่าย

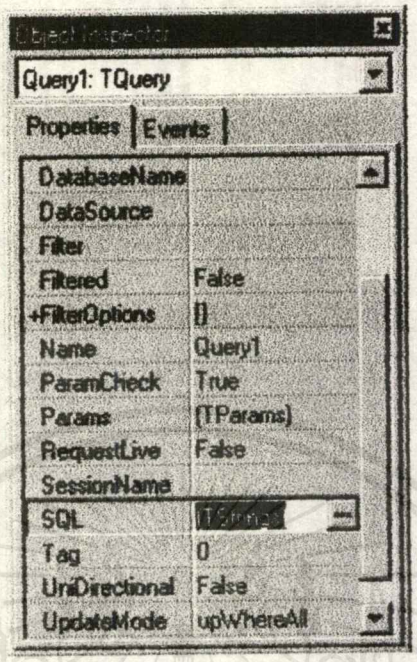
- ทำการวางคิวรีคอมโพเนนท์ลงในฟอร์ม
- ทำการวางดาต้าซอสคอมโพเนนท์ลงในฟอร์ม

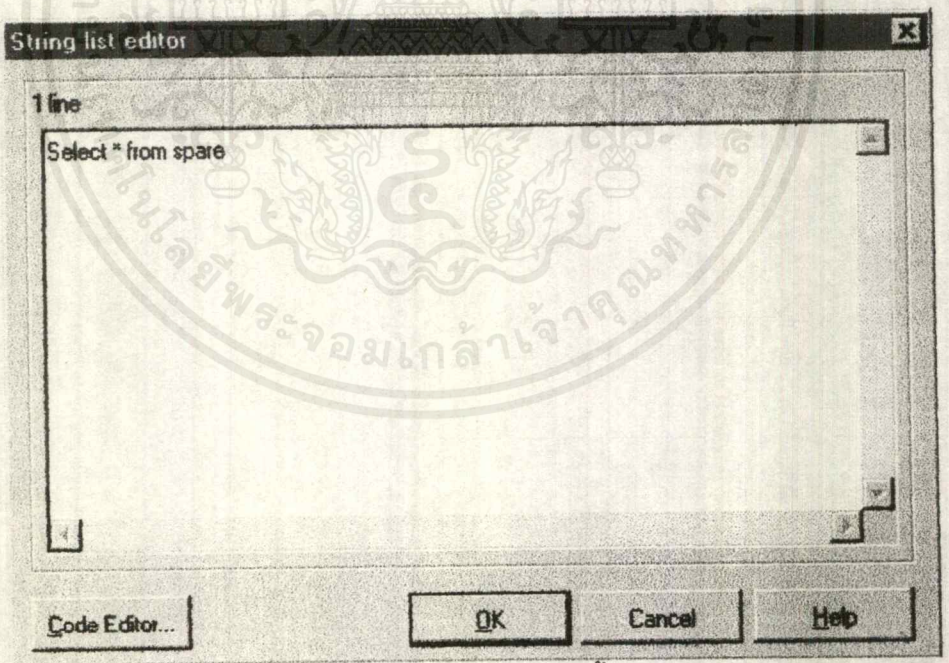


รูปที่ 3-10 แสดงฟอร์มที่มีคิวรีและดาต้าซอส

- กำหนดค่าเอสคิวแอลพร็อพเพอตี้ (SQL Property) ของคิวรีคอมโพเนนท์ ให้เป็นคำสั่งเอสคิวแอลที่ต้องการ เช่น Select * From Spare เป็นต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

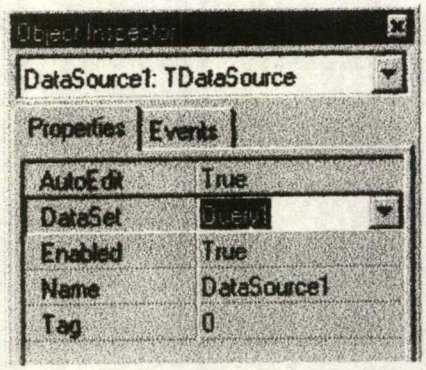




รูปที่ 3-11 แสดงการกำหนดเอสคิวแอลพร็อพเพอร์ตี้ของคิวรีคอมโพเนนท์

- กำหนดค่าดาต้าเซตพร็อพเพอร์ตี้ (Dataset Property) ของดาต้าซอสคอมโพเนนท์ ให้เป็นชื่อของคิวรีคอมโพเนนท์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3-12 แสดงการกำหนดดาต้าเซตพร็อพเพอร์ตี้ของคิวรีคอมโพเนนท์

- กำหนดค่าแอคทีฟพร็อพเพอร์ตี้ (Active Property) ของคิวรีให้เป็นจริง

จากนั้นคิวรีก็พร้อมที่จะใช้งานโดยสามารถทำการเชื่อมต่อกับดาต้าคอนโทรลอื่นๆผ่านทางดาต้าซอสคอมโพเนนท์

พร็อพเพอร์ตี้ของคิวรีคอมโพเนนท์ แสดงดังนี้

Active (Boolean)	เพื่อบอกว่า Query นั้นกำลังถูกใช้งานอยู่หรือไม่ การตั้งค่าเป็น true จะเป็นการสั่งให้ Execute คำสั่ง SQL ที่กำหนดไว้
BOF (Boolean)	มีค่าเป็น True เมื่อตัวชี้ของผลลัพธ์อยู่ที่แถวแรกของผลลัพธ์
Cached Update (Boolean)	เพื่อบอกว่าใช้งาน Cache Update หรือไม่ ถ้า Cache Update ถูกใช้งานแล้ว การเปลี่ยนแปลงข้อมูลจะถูกเก็บอยู่ที่ Client ก่อนจนกว่าจะใช้คำสั่ง Commit การเปลี่ยนแปลงจึงจะถูกส่งไปที่ Server ถ้าต้องการส่งไปยัง Server ทันทีให้ใช้คำสั่ง Apply Update การใช้งาน Cache Update ช่วยลดเวลาในการทำ Transaction ได้ โดยถ้า Transaction ยังไม่ถูก Commit จะยังคงไม่มีการเปลี่ยนแปลง รวมถึงไม่มีการ Lock Record หรือ Table ใดๆด้วย ดังนั้น User อื่นสามารถใช้งานข้อมูลนั้นๆได้จนกว่าจะมีการ Commit เกิดขึ้นจึงจะเริ่ม Lock และ Update ซึ่งก่อนที่จะมีการ Commit นั้น การ Update ที่เกิดขึ้นจะอยู่ที่หน่วยความจำของเครื่อง Client เท่านั้นทำให้ผู้ใช้คนอื่นไม่เห็นการเปลี่ยนแปลง จึงถือเป็นข้อเสียอย่างหนึ่ง โดยค่าดีฟอลท์ คือ False
ConModify (Boolean)	เพื่อบอกว่าผลลัพธ์ของ Query นี้สามารถทำการแก้ไขได้หรือไม่ ถ้าเป็น False ผลลัพธ์ของ Query นี้จะไม่สามารถแก้ไขได้ ค่านี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

	จะเป็น True ได้เมื่อตั้งค่า RequestLive Property ของ Query เป็น True ซึ่งสามารถทำได้บางกรณีเท่านั้น ดูรายละเอียดใน RequestLive Property
Database Name	เพื่อบอกว่า Query นี้ทำงานกับ Database ตัวใด โดยค่าที่เป็นไปได้คือ BDE Alias, ไดเรคทอรีของ File database กรณีที่ Database เป็นไฟล์ในเครื่องเดียวกัน หรือ Alias ซึ่งถูกตั้งใหม่โดย Tdatabase Component
Datasource	เพื่อระบุ datasource ซึ่งจะใช้เป็นตัวเชื่อมโยงค่าพารามิเตอร์จากQuery อีกตัวหนึ่ง โดยค่าพารามิเตอร์ คือค่าที่เว้นไว้เพื่อใส่ค่าก่อนที่จะทำงาน เช่น Select Brand from spare where category = :Category จากตัวอย่างจะเห็นว่า :Category เป็นพารามิเตอร์ที่ใส่ค่าได้โดยชุดคำสั่ง ParamByName ก่อนทำการ Execute โดย Parameter นี้สามารถใส่ค่าได้อีกวิธีหนึ่งโดยอาศัยค่าจากอีก Queryหนึ่ง เช่น Select category, model from spare where quantity = 10 ค่าพารามิเตอร์ (:Category) จะถูกตั้งเป็น Query ของอีก Query หนึ่งได้โดยตั้งค่า Datasource ให้เป็น Datasource ของ Query ที่มีค่าของพารามิเตอร์ที่ต้องการอยู่ เราสามารถทำให้ Query เปลี่ยนแปลงได้โดยผลจากอีก Query หนึ่งโดยไม่ต้องใช้คำสั่งอื่นๆ
EOF (Boolean)	มีค่าเป็น True เมื่อตัวชี้ผลลัพธ์อยู่ที่แถวสุดท้ายของผลลัพธ์
Field Count (Interger)	บอกจำนวน Field (Column) ของผลลัพธ์
FieldDefs (TFieldDefs)	บอกข้อมูลของแต่ละ Field ในผลลัพธ์
Fields (Array of TField)	ให้ค่าของ Tfield ตาม index ของ array ที่กำหนด เช่น Field(0) หมายถึง Field แรกในผลลัพธ์
FieldValues (Variant)	ใช้ค่าของ Field ตาม Index ซึ่งเป็นชื่อของ Field
Local (Boolean)	บอกว่า Database ที่ใช้งานอยู่นั้นอยู่บนเครื่อง Client หรือ Server
Modified (Boolean)	มีค่าเป็น True เมื่อข้อมูลกำลังถูกแก้ไข
ParamCount (Array of TParam)	บอกจำนวน Parameter ใน Query
Prepared (Boolean)	บอกว่า Query นี้มีการ Prepare หรือไม่
RecordCount (LongInt)	บอกจำนวนเรคอร์ดในผลลัพธ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

RequestLive (Boolean)	บอกว่าผลลัพธ์ของ Query นี้สามารถเปลี่ยนแปลงแก้ไขได้หรือไม่ ถ้ากำหนดให้เป็น True แล้วผลลัพธ์จะสามารถแก้ไขได้ถ้า SQL สอดคล้องกับเงื่อนไขบางอย่าง แต่ถ้า SQL ผิดเงื่อนไขก็ยังไม่สามารถทำการแก้ไขผลลัพธ์ได้ สามารถตรวจสอบได้จาก CanModify Property โดย SQL ซึ่งมีการ Join จากหลายตารางจะไม่สามารถแก้ไขได้ เนื่องจากถ้ามีการแก้ไขผลลัพธ์ ทำให้เกิด Data inconsistency
Session Name (String)	ใช้ระบุ Session component ที่ Query ต้องการใช้
SQL (String)	ใช้กำหนดคำสั่งภาษา SQL ที่ใช้ในการ Query โดยปกติจะมีไม่เกิน 1 คำสั่ง ยกเว้นในกรณีที่ Server มีรูปแบบคำสั่งพิเศษ ซึ่งจะใช้งานได้หลายๆคำสั่งพร้อมๆกัน
State	บอกว่าขณะปัจจุบัน Query มีสถานะเป็นอย่างไร เช่น กำลังแก้ไข ไม่ได้ถูกใช้งาน เป็นต้น
StmtHandle (HDBIStmt)	ให้ค่า Handle ของผลลัพธ์เพื่อใช้งานกับ BDE API โดยตรง
Text (pChar)	ใช้เพื่อตรวจสอบว่าคำสั่ง SQL ที่แท้จริงที่กำลังใช้งานเป็นอย่างไรโดยเป็นการตรวจสอบระหว่างการ Run ว่าคำสั่ง SQL ถูกต้องตามต้องการหรือไม่
Unidirectional (Boolean)	บอกว่า Query สามารถเลื่อนตำแหน่งตัวชี้ผลลัพธ์ไปได้ในทิศทางเดียวหรือไม่ โดยปกติแล้วจะเลื่อนได้ 2 ทิศทาง คือไปข้างหน้า และไปข้างหลัง ซึ่งการกำหนดเป็นทิศทางเดียวจะทำให้การใช้ Memory น้อยลง
Update Mode	ใช้เพื่อบอกว่าการ Update ข้อมูลของ Query Component จะทำโดยตรวจสอบอะไรบ้าง ซึ่งจะเป็นการระบุใน Where Clause ของคำสั่ง Update ว่าจะตรวจสอบอะไรบ้าง ซึ่งค่าที่เป็นไปได้มีดังนี้ Where All เป็นการป้องกันมากที่สุด คือเมื่อทำการ Update จะทำการตรวจสอบค่าเดิมทุกค่า ดังนั้น ถ้าระหว่างการแก้ไขของ User คนหนึ่งแล้วมี User อีกคนหนึ่งมาทำการแก้ไขในเรคอร์ดเดียวกันจะทำให้ User ที่มาทีหลังไม่สามารถแก้ไขได้ เพราะ Where Clause จะไม่สามารถพบ Record ได้เนื่องจากค่าเดิมได้ถูกแก้ไขไปแล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

	WhereKeyOnly เป็นการตรวจสอบโดยอาศัยค่าของ Key เท่านั้น WhereChange เป็นการตรวจสอบโดยใช้ทั้ง Key และค่าที่กำลังจะเปลี่ยนแปลง
UpdateObject	ใช้บอก Update Component (TUpdateSql) ที่จะใช้เวลาทำการ Update ในกรณีที่ผลลัพธ์แก้ไขไม่ได้เนื่องจาก Data inconsistency ก็ต้องบอกว่าการแก้ไขจะทำได้อย่างไร โดยระบุคำสั่งที่ใช้แก้ไขลงใน TupdateSql

ตารางที่ 3-6 แสดงค่าพร็อพเพอร์ตี้ต่างๆของคิวรีคอมโพเนนท์

Method ต่างๆ ใน TQuery

Append	ใช้เพื่อทำการแทรกเรคอร์ดใหม่ลงในผลลัพธ์ โดยผลลัพธ์ต้องอยู่ในสถานะที่แก้ไขได้ หลังจากที่ใช้ Methodนี้จะมีเรคอร์ดใหม่ที่ว่างเกิดขึ้นในผลลัพธ์
AppendRecord	ใช้เพื่อแทรกเรคอร์ดใหม่โดยระบุค่าของฟิลด์ต่างๆลงในคำสั่งเลย
ApplyUpdate	สั่งให้ทำการเปลี่ยนแปลงข้อมูลในกรณีที่ใช้ Cached Update
Cancel	ยกเลิกข้อมูลที่กำลังแก้ไขอยู่ให้กลับเป็นข้อมูลเดิม
CancelUpdate	ยกเลิกข้อมูลใน Cache ในกรณีที่ใช้ Cached Update
ClearFields	ทำการลบข้อมูลในเรคอร์ดที่กำลังแก้ไขอยู่ให้เป็นเรคอร์ดว่างๆ
Close	ใช้ยกเลิกการทำงานของ Query มีผลเหมือนกับการตั้งค่า Active ให้เป็นเท็จ
Delete	ทำการลบเรคอร์ดปัจจุบันออกจากตาราง
DisableControls	ทำการระงับการติดต่อระหว่าง Query และ Datasource เพื่อให้การทำงานบางอย่างที่ทำซ้ำๆกันใน Query เร็วยิ่งขึ้น
Edit	สั่งให้ทำการแก้ไขเรคอร์ดปัจจุบัน
EnableControl	ทำการเชื่อมการติดต่อที่ถูกระงับโดย DisableControl
ExecSQL	สั่งให้ทำคำสั่ง SQL ซึ่งคำสั่งที่ใช้มักไม่ต้องการผลลัพธ์เช่น Update, Delete
FieldByName	ให้ค่า TField ที่สัมพันธ์กับชื่อ Field ที่กำหนด
FindField	ให้ค่า Field ตามชื่อที่กำหนด ถ้าไม่มีให้กำหนดค่าเป็น nil
FindFirst	หาเรคอร์ดแรกที่ตรงตามเงื่อนไขที่กำหนดใน Filter

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

FindLast	หาเรคอร์ดสุดท้ายที่ตรงตามเงื่อนไข
FindNext	หาเรคอร์ดถัดไปที่ตรงตามเงื่อนไข
FindPrior	หาเรคอร์ดก่อนหน้านี้ที่ตรงตามเงื่อนไข
First	เลื่อนตัวชี้เรคอร์ดไปยังเรคอร์ดแรกในผลลัพธ์
Insert	สั่งให้ทำการแทรกเรคอร์ดว่างๆในผลลัพธ์ที่ตำแหน่ง เรคอร์ดปัจจุบัน
InsertRecord	สั่งให้แทรกเรคอร์ดโดยทำการระบุค่าของ Field ลงไปเลย
Last	เลื่อนตัวชี้เรคอร์ดไปยังเรคอร์ดสุดท้ายในผลลัพธ์
Locate	ทำการเลื่อนตัวชี้ไปยังเรคอร์ดที่ตรงตามเงื่อนไข โดยสามารถระบุว่าให้ทำการค้นหาแบบ Case Sensitive หรือไม่ หรือทำการค้นหาบางส่วนของ Key
MoveBy	เลื่อนตัวชี้ไปยังเรคอร์ดตามจำนวนที่กำหนด ถ้าค่าบวกทำการเลื่อนขึ้น ถ้าค่าลบทำการเลื่อนลง
Next	ทำการเลื่อนตัวชี้เดินหน้า 1 เรคอร์ด
Open	สั่งให้ทำงานตามคำสั่ง SQL กรณีที่เป็นคำสั่งที่มีผลลัพธ์เช่น Select มีผลเหมือนการตั้งค่า Active เป็น True
ParamByName	ใช้ตั้งค่าพารามิเตอร์ โดยตั้งตามชื่อพารามิเตอร์ที่กำหนด
Post	ทำการเขียนเรคอร์ดปัจจุบันลงฐานข้อมูล
Prepare	ทำการ Prepare Query ซึ่งจะประมวลผล Query เพื่อให้ทำงานได้เร็วขึ้น ใช้ในกรณีที่ใช้งาน Query เดียวหลายๆครั้ง
Prior	ทำการเลื่อนตัวชี้เรคอรด์ไปยังเรคอร์ดถัดไป
Refresh	ทำการอ่านเรคอร์ดเข้ามายังชุดผลลัพธ์ใหม่ ในกรณีของผลลัพธ์ที่สามารถแก้ไขได้ เพื่อให้ได้ข้อมูลล่าสุด
SetFields	ใช้ใส่ข้อมูลลงในเรคอร์ดโดยเรียงลำดับตาม Field
UnPrepare	ใช้ยกเลิกการ Prepare
UpdateStatus	ให้สถานะของเรคอร์ดล่าสุดใน Cache Update

ตารางที่ 3-7 แสดงเมธอดต่างๆ ใน TQuery

3.2.2.2 ที-ดาต้าเบส คอมโพเนนท์ (Tdatabase Component)

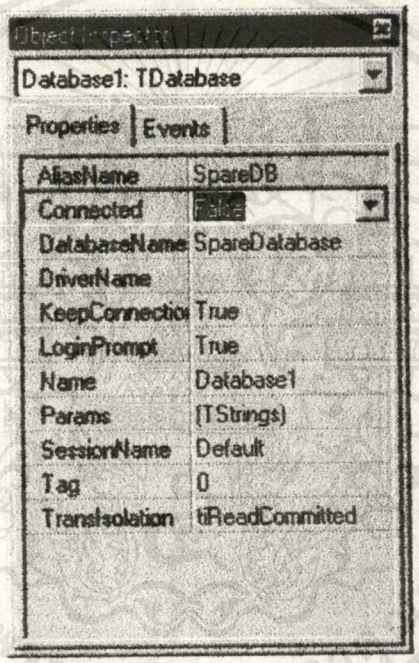
โดยปกติแล้วในเดลไฟล์สามารถทำการติดต่อกับฐานข้อมูลได้โดยไม่จำเป็นต้องมีที-ดาต้าเบส คอมโพเนนท์ เพียงแค่มีอไลแอสก็สามารถใช้งานได้ แต่ถ้าใช้ที-ดาต้าเบส คอมโพเนนท์ ทำให้เราสามารถควบคุม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การใช้งานได้มากขึ้น สามารถทำงานบางอย่างได้ เช่น สามารถควบคุมการล็อกอิน (Login) เข้าไปยังเซิร์ฟเวอร์ และสามารถควบคุมการทำทรานสแอคชั่นได้ แต่ถ้าไม่กำหนดที-ดาต้าเบส คอมโพเนนท์แล้ว เดลไฟล์จะทำการสร้างที-ดาต้าเบส (TDatabase) ชั่วคราวให้เองโดยอัตโนมัติ

ในการใช้งานง่ายๆ สามารถทำได้โดย

- กำหนดชื่ออไลแอส (AliasName) หรือ ชื่อไดร์ฟเวอร์ (DriverName) อย่างใดอย่างหนึ่ง
- กำหนดชื่อฐานข้อมูล (DatabaseName)เพื่อใช้ในโปรแกรม เรียกว่า แอพพลิเคชั่น-สเปซิฟิค-อไลแอส (Application Specific Alias)
- กำหนดคอนเน็ค (Connected) ให้มีค่าเป็นจริง



รูปที่ 3-13 แสดงการกำหนดพร็อพเพอร์ตี้ของดาต้าเบสคอมโพเนนท์

จากนั้นสามารถใช้งานได้โดยกำหนดชื่อฐานข้อมูลของ ดาต้าเซตคอมโพเนนท์ (Dataset Component) ให้เป็นตามชื่อฐานข้อมูลของที-ดาต้าเบส

ค่าพร็อพเพอร์ตี้ต่างๆของที-ดาต้าเบส คอมโพเนนท์

AliasName (Alias string)	กำหนด BDE Alias เพื่อใช้งานกับ TDatabase
Connected (Boolean)	บอกว่ามีการเชื่อมต่ออยู่หรือไม่ ก่อนการใช้งานค่านี้ต้องมีค่าเป็นค่า True จะเปลี่ยนเป็น False ก็ต่อเมื่อ Table ถูกปิดหมด ยกเว้นกำหนดให้ KeepConnection เป็น True
DatabaseName (TFileName)	กำหนด Application-Specific Alias เพื่อใช้งานมนแอพพลิ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

	เคชั่น โดยComponent อื่นๆจะเชื่อม TDatabase โดยผ่านชื่อนี้
DatasetCount (Integer)	บอกจำนวนของ Data Component ที่เชื่อมต่อกับ TDatabase อยู่
Datasets (Array of TDBDatasets)	กลุ่มของ Dataset Component ที่เชื่อมต่อกับ TDatabase อยู่
DriverName (TSymbolStr)	ชื่อ Driver ที่ใช้งานอยู่ ถ้ากำหนด AliasName แล้วค่านี้จะเป็นค่าว่าง หรือถ้ามีการกำหนดค่านี้แล้ว AliasName ก็จะเป็นค่าว่าง
Handle (HDBI DB)	Database Handle เพื่อใช้กับ BDE API โดยตรง
IsSQLBased (Boolean)	บอกว่าใช้ Driver ตัวใดอยู่ จะเป็นจริงเมื่อ Driver ที่ใช้ไม่ใชแบบมาตรฐาน (คือกรณีใช้เซิร์ฟเวอร์นั่นเอง)
KeepConnection (Boolean)	ใช้บอกว่าจะให้ Connected เป็น False หรือไม่ในกรณีที่ไม่มี Table เปิดอยู่เลย ถ้าเป็นค่า True แล้ว Connected จะเป็น True แม้ว่าจะไม่มี Table ใดเปิดอยู่เลยก็ตาม แต่ถ้าเป็น False แล้วต้องทำการ Login ใหม่ทุกครั้งที่ปิด Table จนหมดแล้วต้องการเปิดขึ้นใหม่
LoginPrompt (Boolean)	กำหนดว่าจะให้กรอบ Login ให้ใส่ User และ Password ทุกครั้งที่เปิด Database หรือไม่ โดยปกติจะเป็น True แต้ถ้าเป็น False จะต้องกำหนด User Name และ Password ที่ Params Property
Params (TStringList)	กำหนด Parameter สำหรับการเชื่อมต่อ เช่น UserName, Password, ServerName
Temporary (Boolean)	บอกว่า Tdatabase Component ว่าเป็นชนิดชั่วคราวหรือไม่
TransIsolation	กำหนดค่า Transaction Isolation ของ SQL Server TiDirtyRead จะกำหนดว่าค่าที่เปลี่ยนแปลงไปแล้วจะสามารถเห็นได้จากผู้ใช้อื่นได้ทันทีแม้ว่ายังไม่ Commit TiReadCommitted จะกำหนดว่าค่าที่เปลี่ยนแปลงไปจะเห็นได้โดยผู้ใช้คนอื่นเมื่อ Commit แล้วเท่านั้น TiRepeatableRead จะทำการกำหนดว่าระหว่างที่ทำ Transaction อยู่นั้น ค่าที่อ่านได้จะเหมือนเดิมเสมอแม้ว่าจะมีคนเปลี่ยนแปลงค่านั้นไปแล้ว

ตารางที่ 3-8 แสดงค่าพร็อพเพอร์ตี้ของที-ดาต้าเบสคอมโพเนนท์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Method ต่างๆ ของ ที-ดาต้าเบส

Close	ทำการตัดการเชื่อมต่อของฐานข้อมูล ให้ผลเช่นเดียวกับการตั้งค่า Connected เป็น False
CloseDataset	ทำการปิด Dataset Component ทั้งหมดที่ติดต่อกับ TableBase นั้น
Commit	ทำการ Commit Transaction ที่เปิดอยู่
Open	ทำการติดต่อกับ Database ให้ผลเช่นเดียวกับการตั้งค่า Connected เป็น True
Rollback	ทำการ Rollback คือ ยกเลิกการเปลี่ยนแปลงทั้งหมดที่เริ่มทำหลังจากคำสั่ง StartTransaction หมายเหตุ : การทำ Rollback ทำได้ต่อเมื่อไม่มี ActiveQuery อยู่ขณะทำ Transaction
StartTransaction	ทำการเริ่มต้น Transaction ซึ่งต้องจบ Commit หรือ RollBack

ตารางที่ 3-9 แสดงเมธอดต่างๆของที-ดาต้าเบส คอมโพเนนท์

3.2.2.3 ที-เซสชัน คอมโพเนนท์ (TSession Component)

ลักษณะเช่นเดียวกับที-ดาต้าเบส คอมโพเนนท์ ซึ่งเราไม่จำเป็นต้องกำหนดเอง แต่เดลไฟล์สามารถสร้างให้โดยอัตโนมัติ แต่ถ้าทำการกำหนดเองก็จะสามารถควบคุมลักษณะการทำงานบางอย่างกับบีดีอีได้ เช่น สามารถอ่านค่าอไลแอสที่มีอยู่มาตรวจสอบได้ สามารถอ่านค่าชื่อฐานข้อมูล (แอพลิเคชัน-สเปซิฟิค-อไลแอส) มาตรวจสอบได้ เป็นต้น

3.2.2.4 ที-เทเบิล คอมโพเนนท์ (TTable Component)

สามารถใช้เพื่อทำงานกับข้อมูลตารางเดียวได้โดยไม่จำเป็นต้องใช้คิวรี หรือไม่ต้องใช้คำสั่งเอสคิวแอล ดังนั้นจึงไม่ขึ้นอยู่กับคำสั่งเอสคิวแอลของเซิร์ฟเวอร์มากนัก โดยปกติถ้าเราไม่ต้องการข้อมูลจากหลายตาราง หรือไม่ต้องการใช้คำสั่งเอสคิวแอล เราจะเลือกใช้เทเบิล คอมโพเนนท์

โดยส่วนใหญ่ ค่าพร็อพเพอร์ตี้ต่างๆของที-เทเบิล (TTable) จะคล้ายกับที-คิวรี (TQuery) นอกจากพร็อพเพอร์ตี้บางตัว ดังนี้

Exclusive (Boolean)	กำหนดว่าจะทำการ Lock Table เพื่อไม่ให้ผู้อื่นเข้ามาใช้ขณะที่เปิด Table อยู่ เมื่อกำหนดค่าเป็น True
TableName (TFileName)	กำหนดชื่อ Table ที่ต้องการใช้งาน
TableType	ใช้บอกว่าตารางนี้เป็น ASCII, Dbase หรือ Paradox แต่ถ้าเป็น SQL Database จะไม่ได้ใช้
IndexFieldCount (Integer)	บอกจำนวน Filed ที่จะใช้เป็น Index ของ Table

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

IndexFieldNames (String)	กำหนด Field ที่จะใช้เป็น Index
IndexFieldProperty (Array of fields)	บอกข้อมูลของแต่ละ Field ที่ใช้เป็น Index

ตารางที่ 3-10 แสดงพร็อพเพอร์ตี้บางตัวของที-เทเบิลคอมโพเนนท์

Method ของ ที-เทเบิล คอมโพเนนท์

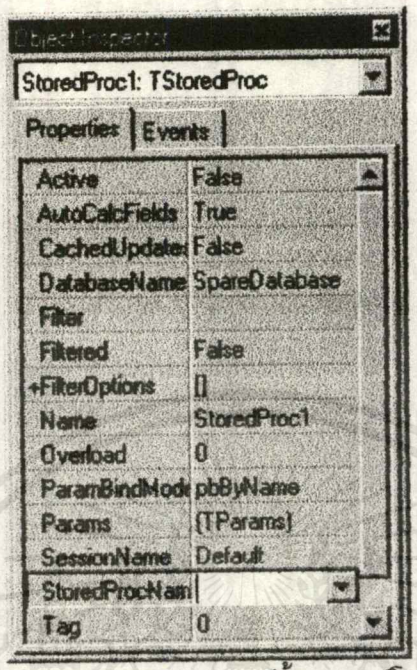
AddIndex	ทำการสร้าง Index ใหม่สำหรับ Table
DeleteIndex	ทำการลบ Secondary Index
ApplyRange	ทำการ Filter Record ซึ่งกำหนดโดย SetRangeStart และ SetRangeEnd
CancelRange	ทำการยกเลิก Range ที่จองไว้
FindKey	ทำการค้นหาเรคอร์ดที่มีค่าของ Index Field ตรงกับที่กำหนด
SetRange	ทำงานโดยกำหนดจุดเริ่มต้นและจุดสิ้นสุดของช่วง จากนั้นเรียกใช้ ApplyRange
EmptyTable	ทำการลบข้อมูลทั้งหมดของตาราง

ตารางที่ 3-11 แสดงเมธอดบางตัวของทีเทเบิลคอมโพเนนท์

3.2.2.5 สตอร์โพรซีเดอร์ คอมโพเนนท์ (Store-Procedure Component)

คอมโพเนนท์ที่ให้ผลเป็นดาต้าเซตเช่นเดียวกับตารางหรือคิวรี การเรียกใช้สามารถทำได้โดยกำหนดชื่อของสตอร์โพรซีเดอร์ ผ่านทางพร็อพเพอร์ตี้สตอร์พร็อคเนม (StoredProcName Property) จากนั้นเรียกใช้โดยคำสั่ง โอเพน (Open) หรือ เอกเซ็คพร็อค (ExecProc) ซึ่งจะให้ผลเหมือนกับการสั่งให้คิวรีทำงาน การใช้งานสามารถใช้ได้เช่นเดียวกับการใช้คิวรี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3-14 แสดงการกำหนดพร็อพเพอร์ตี้ของสตอร์โพรซีเดอร์คอมโพเนนท์

3.2.3 การใช้งานทรานสแอคชั่น

ปกติในเดลไฟล์ ถ้าไม่มีการใช้ที-ดาต้าเบส หรือไม่มีการใช้คำสั่งเกี่ยวกับทรานสแอคชัน การทำทรานสแอคชันจะทำโดยอัตโนมัติ ทุกครั้งที่ใช้คำสั่งจบ 1 คำสั่งจะมีการคอมมิททันที การใช้ทรานสแอคชันสามารถทำได้โดยผ่านที-ดาต้าเบสคอมโพเนนท์ นั่นคือจะต้องมีการตั้งค่าของฐานข้อมูลที่จะใช้งาน ให้เป็นที-ดาต้าเบสตัวนั้น คือตั้งค่าตามแอพพลิเคชั่น-สเปซิฟิค-อไลแอส ซึ่งก็คือค่าชื่อฐานข้อมูลของที-ดาต้าเบสตัวนั้นนั่นเอง

การเริ่มทำทรานสแอคชันทำได้โดยใช้สตาร์ททรานสแอคชัน เมธอด (StartTransAction Method) ของที-ดาต้าเบส จากนั้นทำคำสั่งต่างๆที่ต้องการ เมื่อครบแล้วจึงใช้คำสั่งคอมมิทของที-ดาต้าเบส ถ้าหากคำสั่งที่ทำในทรานสแอคชันไม่สำเร็จ จะต้องทำการยกเลิกการเปลี่ยนแปลงโดยใช้คำสั่ง โรลแบค (Rollback)

```
Database1.StartTransaction
TRY
        Out.Post  { ทำการเพิ่มเรคอร์ดใหม่ลงในตาราง Out }
        UpdateSpare.ParamByName('Category'). AsString := Category;
        UpdateSpare.ParamByName('Quantity'). AsInteger := 20;
        UpdateSpare.ExecSQL;  { สั่งให้ UpdateQuery ทำงาน }
        Database1.Commit { ทำงานสำเร็จครบ }
EXCEPT  { เกิดความผิดพลาด }
        MessageDlg('Error ! ',mtError,[mbok],0);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Database1.RollBack { ยกเลิกการเปลี่ยนแปลง }

ความสัมพันธ์ในการทำงานแต่ละทรานสแอคชัน สามารถกำหนดได้โดยกำหนดค่าทรานสไอโซเลชัน (TransIsolation) ของคอมโพเนนท์ที-ดาต้าเบส ซึ่งค่าที่เป็นไปได้ คือ

- ทีไอเดอร์ตี้รีดทรานสแอคชัน (TiDirtyRead TransAction) สามารถอ่านค่าที่ถูกเปลี่ยนแปลงโดยทรานสแอคชันอื่น แม้ว่าจะยังไม่คอมมิทก็ตาม ซึ่งในระดับนี้อาจทำให้เกิดความผิดพลาดได้ ถ้าอ่านค่าไปแล้ว ทรานสแอคชั่นนั้นทำการโรลแบค ค่าที่อ่านได้ก็จะผิดพลาด

- ทีไอรีดคอมมิททรานสแอคชัน (TiReadCommited TransAction) สามารถอ่านค่าได้เฉพาะค่าที่คอมมิทแล้วเท่านั้น

- ทีไอรีพีทเทเบิลรีดทรานสแอคชัน (TiRepeatableRead TransAction) ในระหว่างการทำทรานสแอคชัน ค่าที่อ่านได้จะเหมือนเดิมเสมอ แม้ว่าจะมีทรานสแอคชันอื่นมาเปลี่ยนแปลงค่านั้นก็ตาม

เออร์เรอร์-แฮนด์ลิ่ง (Error Handling)

ในเดลไฟล์ สามารถจัดการกับข้อผิดพลาดของฐานข้อมูลได้ ดังนี้

```
TRY
    { คำสั่งเปลี่ยนแปลงข้อมูล }
EXCEPT
    On e:eDBEngineError do. ....
    {คำสั่งจัดการข้อผิดพลาด}
```

โดยจะทำคำสั่งหลังจาก TRY ซึ่งถ้ามีข้อผิดพลาด จะข้ามมาทำที่คำสั่งหลัง EXCEPT ซึ่งสามารถตรวจสอบความผิดพลาดโดยตรวจสอบว่าเกิดข้อผิดพลาดชนิดใด ซึ่งประโยค On e:eDBEngineError do ... หมายความว่า ถ้ามีความผิดพลาดเกิดขึ้นจากการจัดการฐานข้อมูลก็จะทำตามคำสั่งต่อไปหลัง do.. โดยรายละเอียดความผิดพลาดให้ส่งผ่านไปยังตัวแปรชื่อ e (จากคำสั่ง e:EDBEngineError) ซึ่งเป็นตัวแปรชนิด EDBEngineError

สำหรับชนิดของความผิดพลาด และรายละเอียดของความผิดพลาด สามารถตรวจสอบได้จากค่าพร็อพเพอร์ตี้ของอี-ดีบีเอนจิ้น-เออร์เรอร์ (eDBEngineError) ซึ่งตามตัวอย่างนี้ก็คือค่าพร็อพเพอร์ตของตัวแปร e ดังนี้

ErrorCount (Integer)	บอกจำนวนของ Error ทั้งหมด
Errors (Array of tErrors)	ชุดของ tErrors

ตารางที่ 3-12 แสดงค่าพร็อพเพอร์ตี้ของอี-ดีบีเอนจิ้น-เออร์เรอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยค่าของที-เออร์เรอร์ (tError) สามารถใช้เพื่อตรวจสอบรายละเอียดของเออร์เรอร์แต่ละตัวได้จากค่าพร็อพเพอร์ตี้ต่อไปนี้

ErrorCode	รหัสของเออร์เรอร์ จากบีดีอี
Category	ชนิดของเออร์เรอร์
SubCode	รหัสย่อยของเออร์เรอร์โค้ด (Error Code)
NativeError	รหัสของเออร์เรอร์ที่ส่งมาจากเซิร์ฟเวอร์ แต่ถ้ามีค่าเท่ากับ 0 แสดงว่าเออร์เรอร์นั้น ไม่ใช่เซิร์ฟเวอร์เออร์ เรอร์
Message	ข้อความแสดงความผิดพลาด

ตารางที่ 3-13 แสดงค่าของโดยค่าพร็อพเพอร์ตี้ของที-เออร์เรอร์

การใช้งานค่าพร็อพเพอร์ตี้ของที-เออร์เรอร์สามารถทำได้ดังนี้

```
TRY
        Begin
        Out.Post;    {Post change to out table }
        form4.Database1.Commit;
        Result:=True;
        End;
EXCEPT
on e:eDBEngineError Do
Begin
        Messagedlg('ERROR : '+e.message,mtError,[mbOK],0);
        form4.Database1.Rollback  ;
End;
```

3.2.4 การทำงานกับฐานข้อมูลโดยใช้เดลไฟล์

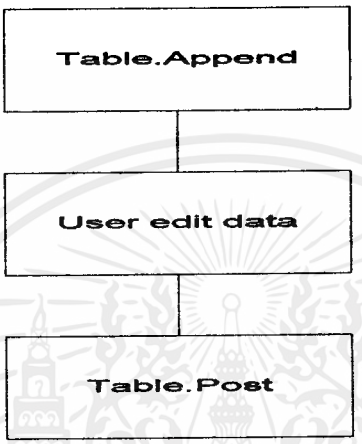
การใช้งานฐานข้อมูลโดยผ่านเดลไฟล์ สามารถทำได้หลายวิธีโดยอาจทำผ่านดาต้าคอนโทรลต่างๆของเดลไฟล์ หรืออาจทำผ่านคำสั่งเอสคิวแอลโดยตรงได้

3.2.4.1 การอินเสิร์ท (Insert)

การแทรกแถวของข้อมูลสามารถทำได้ผ่านเทเบิล-คอมโพเนนท์ (Table Component) หรือ คิวรี-คอมโพเนนท์ (Query Component) ในเทเบิล-คอมโพเนนท์สามารถทำผ่านดาต้าคอนโทรลได้ โดยการเชื่อมดาต้าคอนโทรลเหล่านั้น เช่น ดีบีอีดิท (DBEdit) ดีบีกริด (DBGrid) เข้ากับเทเบิล-คอมโพเนนท์ จากนั้นใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แอพเพน-เมธอด (Append Method) จะทำให้เกิดเรคอร์ดว่างต่อที่แถวท้ายสุดของตาราง และตัวชี้เรคอร์ดจะชี้ไปที่เรคอร์ดว่างใหม่นั้น จะทำให้ดาต้าคอนโทรลต่างๆที่เชื่อมกับเทเบิล-ออบเจคต์ (Table Object) แสดงค่าเรคอร์ดว่าง ซึ่งสามารถให้ผู้ใช้ป้อนค่าต่างๆได้ตามต้องการ เมื่อทำการป้อนค่าต่างๆจนครบแล้วก็สามารถใช้โพส-เมธอด (Post Method) เพื่อทำการเขียนข้อมูลลงในตารางได้ หรืออาจใช้แอพเพนเมธอด เพื่อเขียนข้อมูล และเพิ่มเรคอร์ดใหม่อีกครั้งก็ได้ (แอพเพนเมธอดจะเรียกโพสเมธอดโดยอัตโนมัติ)



รูป 3-15 แสดงขั้นตอนการทำงานโดยใช้แอพเพน-เมธอด

การแทรกเรคอร์ดสามารถทำได้อีกวิธีหนึ่ง คือ โดยใช้อินเสิร์ทเรคอร์ดเมธอด (InsertRecord Method) ในกรณีที่ทราบข้อมูลที่จะใส่ลงตารางอยู่แล้ว โดยการใช้อินเสิร์ทเรคอร์ด จะต้องผ่านค่าข้อมูลที่ต้องการใส่เป็นพารามิเตอร์ของเมธอดได้เลย เช่น Customer.InsertRecord([CustNoEdit.Text, CoNameEdit.Text, AddrEdit.Text,Null, Null, Null, Null, Null, Null, DiscountEdit.Text]);

อีกวิธีสามารถ ใช้อินเสิร์ทเมธอด (Insert Method) หรือ แอพเพนเมธอด (Append Method)เพื่อทำการแทรกเรคอร์ดใหม่ลงในตาราง จากนั้นใช้ฟิลด์บายเนมเมธอด (FieldByName Method) ซึ่งสามารถทำการใส่ค่าลงในฟิลด์ต่างๆได้โดยใช้ชื่อฟิลด์เป็นพารามิเตอร์ และทำการกำหนดค่าโดยตรง เช่น Out.FieldByName('Category').AsString := 'LanCard'; เมื่อทำการกำหนดค่าครบแล้วก็สามารถเรียกโพส-เมธอด (Post Method) เพื่อทำการใส่ค่าลงในตารางได้

ในแต่ละวิธีที่กล่าวมาให้ผลการทำงานไม่ต่างกันดังนั้นสามารถเลือกใช้ได้ให้เหมาะสม ตามสถานการณ์

การอินเสิร์ทผ่านคิวรีสามารถทำได้ โดยใช้ดาต้าคอนโทรลเช่นเดียวกับเทเบิล-คอมโพเนนท์ ในกรณีที่คิวรีนั้น สามารทำการแก้ไขค่าของผลลัพธ์ได้ (Modified = True) ในกรณีที่ไม่สามารถทำการแก้ไขค่าของผลลัพธ์ได้ อาจเนื่องมาจากเป็นการรวมข้อมูลจากหลายตาราง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การอินเสิร์ทยังสามารถทำได้โดยผ่านอัพเดทออบเจคต์ (Update Object) ซึ่งก็คือ ที-อัพเดท-เอสคิวแอล คอมโพเนนท์ (TUpdateSQL Component) ซึ่งจะทำการเก็บคำสั่งที่ใช้แก้ไขข้อมูลไว้ โดยจะสามารถผ่านค่าข้อมูลที่ต้องการแทรกได้ผ่านพารามิเตอร์

การอินเสิร์ทสามารถทำได้โดยผ่านคำสั่งเอสคิวแอลโดยตรง คือตั้งค่าเอสคิวแอลพร็อพเพอร์ตี้ (SQL Property) ของคิวรีให้เป็นคำสั่งอินเสิร์ท โดยตั้งค่าพารามิเตอร์ไว้ จากนั้นเมื่อเวลาจะทำการอินเสิร์ทก็ใช้คำสั่งพารามบายเนม (ParamByName) เพื่อตั้งค่าให้พารามิเตอร์จนครบ จากนั้นใช้คำสั่งเอ็กซ์เซคเอสคิวแอล (ExecSQL) เพื่อทำการอินเสิร์ทได้ เช่น ในพร็อพเพอร์ตี้ของเอสคิวแอล ตั้งค่าไว้ดังนี้

```
Insert INTO Out VALUE(:Category, :Brand, :Model)
```

เมื่อต้องการทำการแทรกค่าสามารถใส่ค่าได้ดังนี้

```
Q1.ParamByName('Category').AsString := 'LanCard';
Q1.ParamByName('Brand').AsString := 'Dlink';
Q1.ParamByName('Model') AsString := 'Combo'
Q1.ExecSQL
```

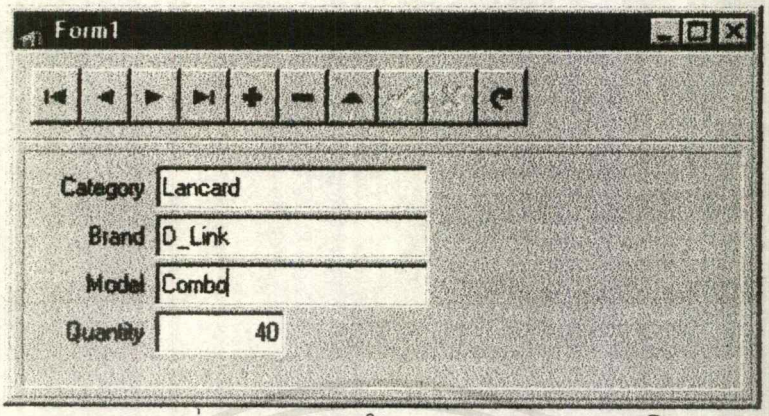
การใช้คำสั่งเอสคิวแอลโดยตรงมีข้อได้เปรียบที่ความยืดหยุ่น โดยสามารถทำการกำหนดคำสั่งเฉพาะพิเศษของเซิร์ฟเวอร์ได้ แต่โดยปกติแล้วให้ผลลัพธ์ไม่แตกต่างกันมากนัก การใช้งานจึงขึ้นกับความสะดวกและความเหมาะสมมากกว่า

3.2.4.2 การอัพเดท (Update)

การแก้ไขข้อมูล ผ่านทางเทเบิล-ออบเจคต์ (Table Object) สามารถทำได้ผ่านทางดาต้าคอนโทรลเช่นกัน โดยทำการเชื่อมดาต้าคอนโทรลเข้ากับเทเบิล-ออบเจคต์ จากนั้นจะสามารถสั่งให้เลื่อนเรคอร์ดไปข้างหน้า หรือย้อนกลับหลัง หรืออื่นๆได้ทั้งทางเมธอด และใช้ดีบีเนวิเกชัน คอนโทรล (DBNavigation Control) ก็ได้ เมื่อเลื่อนเรคอร์ดได้ตรงตามต้องการแล้ว ก็ผู้ใช้ก็สามารถทำการแก้ไขได้

สามารถทำการค้นหาข้อมูลเรคอร์ดที่ต้องการได้โดยใช้ ฟายด์คีย์เมธอด (FindKey Method) หรือกำหนดช่วงของเรคอร์ดที่ต้องการโดยใช้แอพพลายเรนจ์เมธอด (ApplyRange Method) หรือ เซทเรนจ์เมธอด (SetRange Method) ก็ได้ เมื่อการแก้ไขข้อมูลเสร็จสมบูรณ์แล้วก็ใช้โพส-เมธอด เพื่อเก็บการแก้ไขลงในตาราง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3-16 แสดงการใช้งานอัพเดทผ่านดาต้าคอนโทรล

ส่วนการอัพเดทข้อมูลผ่านทางคิวรี สามารถทำได้ในลักษณะเดียวกันกับการอินเสิร์ท

3.2.4.3 การดีลีท (Delete)

การลบข้อมูลผ่านทางเทเบิล-คอมโพเนนท์ สามารถทำได้โดยผ่านทางดาต้าคอนโทรลเช่นกัน ด้วยวิธีคล้ายกันคือ เลื่อนตัวชี้ของเรคอร์ดไปยังตำแหน่งของเรคอร์ดที่ต้องการ โดยใช้เมธอดต่างๆ หรือใช้เนวิเกชันคอนโทรล จากนั้นใช้ดีลีทเมธอด (Delete Method) เพื่อทำการลบทั้งเรคอร์ดนั้นออกไป หรืออาจใช้เอ็มตี้เทเบิลเมธอด (EmptyTable Method) เพื่อทำการลบตารางทั้งตารางก็ได้

การลบข้อมูลผ่านทางคิวรี ก็สามารถทำได้ในลักษณะทำนองเดียวกันกับการอินเสิร์ท และการอัพเดท สามารถเลือกใช้ได้ตามต้องการ

การทำงานผ่านดาต้าคอนโทรลเหมาะสำหรับข้อมูลจากผู้ใช้ในเงื่อนไขง่ายๆ เช่น การป้อนข้อมูลใหม่ ส่วนการทำงานผ่านเมธอดนั้น ให้ความยืดหยุ่นมากกว่า สามารถเลือกได้หลายวิธี แต่มีข้อเสียคือยุ่งยากมากกว่า การใช้งานจริงจึงควรใช้ผสมผสานกันไป

3.3 ลักษณะสำคัญและวิธีการใช้เพาเวอร์บิวเดอร์ในการพัฒนาโปรแกรม

โปรแกรมภาษาเพาเวอร์บิวเดอร์ (PowerBuilder) เป็นภาษาที่สนับสนุนการสร้างโปรแกรมที่ทำงานกับข้อมูลบนระบบไคล์เอนท์เซิร์ฟเวอร์ มีลักษณะสำคัญดังนี้

3.3.1 ดาต้าวินส์โดว์ (Data Window)

ดาต้าวินส์โดว์ คือ วินส์โดว์ที่ประกอบไปด้วยข้อมูล หมายถึงส่วนที่แสดงและทำงานเกี่ยวกับข้อมูลโดยทั่วไป การทำงานและประมวลผลของเพาเวอร์บิวเดอร์นั้น ดาต้าวินส์โดว์จะไม่อยู่เพียงลำพัง แต่จะอยู่บนวินส์โดว์ธรรมดา

3.3.1.1 ออบเจคต์ที่มีในดาต้าวินส์โดว์

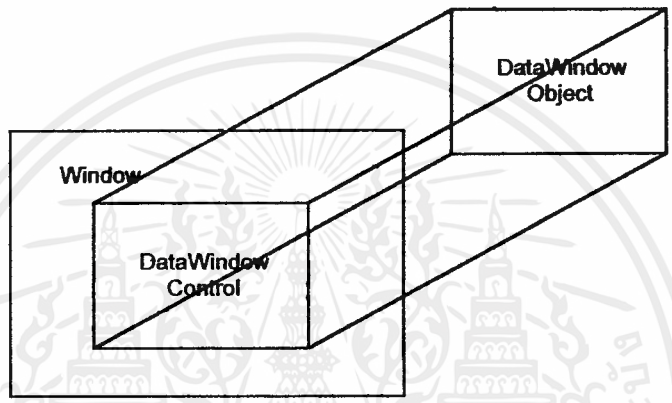
ในดาต้าวินส์โดว์ประกอบด้วยออบเจคต์ต่าง ที่ใช้ทำงานเกี่ยวกับข้อมูล ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1. ดาต้าวินส์โดว์ออบเจคต์ (DataWindow Object) คือ รูปแบบของดาต้าวินส์โดว์ ซึ่งสร้างโดยดาต้าวินส์โดว์เพนท์เตอร์ (DataWindow Painter)

2. วินส์โดว์ (Window) คือ รูปแบบของวินส์โดว์ที่สร้างโดยวินส์โดว์เพนท์เตอร์ (Window Painter)

3. ดาต้าวินส์โดว์คอนโทรล (DataWindow Control) สร้างโดยวินส์โดว์เพนท์เตอร์ ซึ่งเป็นส่วนรวบรวมดาต้าวินส์โดว์ออบเจคต์ต่างๆ



รูปที่ 3-17 แสดงความสัมพันธ์ระหว่างวินส์โดว์, ดาต้าวินส์โดว์คอนโทรลและ ดาต้าวินส์โดว์ออบเจคต์

3.3.1.2 การกำหนดแอททริบิวท์ของดาต้าวินส์โดว์

สามารถทำการกำหนดได้โดยดับเบิ้ลคลิ้กที่ดาต้าวินส์โดว์คอนโทรล เพื่อทำการเปิด Data Window Style Dialog Box

ค่าแอททริบิวท์ต่างๆของดาต้าวินส์โดว์ แสดงดังนี้

Name	ชื่อของดาต้าวินส์โดว์คอนโทรล โดยในเพาเวอร์บิวเดอร์ใช้คำนำหน้าด้วย dw_ และ d_ สำหรับดาต้าออบเจคต์
Title	เพิ่ม Title ให้กับดาต้าวินส์โดว์
Visible	กำหนดให้สามารถมองเห็นคอนโทรลได้
Enabled	กำหนดให้สามารถรับ Input จากผู้ใช้
Title Bar	เพิ่ม TitleBar ให้กับดาต้าวินส์โดว์
Control Menu	เพิ่มคอนโทรลเมนูให้กับดาต้าวินส์โดว์ คือมีฟังก์ชันให้ผู้ใช้ทำการ Move, ReSize, Minimize, Maximize และ Close DataWindow
Maximize Box	เพิ่ม Maximize Button ลงใน DataWindow สามารถกำหนดได้เมื่อทำการกำหนด Title Bar แล้วเท่านั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Minimize Box	เพิ่ม Minimize Button ลงใน DataWindow สามารถกำหนดได้เมื่อทำการกำหนด Title Bar แล้วเท่านั้น
ReSizable	อนุญาตให้ผู้ใช้สามารถทำการเปลี่ยนขนาดของ DataWindow ได้
H ScrollBar	เพิ่ม Scroll Bar ทางแนวราบให้ในดาต้าวินส์โดว์
V ScrollBar	เพิ่ม Scroll Bar ทางดิ่งให้ในดาต้าวินส์โดว์
Live Scrolling	อนุญาตให้ผู้ใช้สามารถเคลื่อนที่ไปยังแถวข้อมูลต่างๆ ในดาต้าวินส์โดว์ได้โดยใช้ ScrollBar
H Split Scroling	กำหนดเพื่อให้ทำการแสดง Split Bar ในดาต้าวินส์โดว์ออบเจคต์ เพื่อให้ผู้ใช้สามารถทำการ Scroll Window ที่แบ่งแยกกันได้อย่างอิสระ
Border	เลือก Border ให้กับคอนโทรล

ตารางที่ 3-14 แสดงค่าแอททริบิวท์ต่างๆของดาต้าวินส์โดว์

3.3.1.3 ดาต้าวินส์โดว์คอนโทรล (DataWindow Control)

ค่าแอททริบิวท์ต่างๆ ของดาต้าวินส์โดว์คอนโทรล แสดงดังนี้

Border	มีค่าเป็นจริง เมื่อมี Border รอบๆ Control
BorderStyle	ค่าที่เก็บไว้แสดงชนิดของ Border (StyleBox!, StyleLowered!, StyleRaised!, StyleShadowBox!)
BringToTop	มีค่าเป็นจริง เมื่อวางดาต้าวินส์โดว์อยู่บนคอนโทรลอื่นๆ
ControlMenu	เพิ่ม Control Menu ให้กับดาต้าวินส์โดว์ และเพิ่มฟังก์ชันที่อนุญาตให้ Move, ReSize, Minimize, Close DataWindow
DataObject	เปลี่ยน DataWindow Object ที่จะทำงานกับ Control นี้
DragAuto	มีค่าเป็นจริง เมื่อทำการกำหนดเป็น Drag Mode โดยอัตโนมัติเมื่อผู้ใช้ทำการคลิ้ก
DragIcon	ข้อความที่แสดงชื่อ Icon ที่ทำการแสดงเมื่อ Control อยู่ใน DragMode
Enables	มีค่าเป็นจริงเมื่อทำการอนุญาตคอนโทรล
Height	ค่า Integer บอกขนาดของคอนโทรลใน PBUs
HScrollBar	มีค่าเป็นจริง เมื่อมีการกำหนด Horizontal Scrollbar ใช้งาน
HSplitScroll	แสดง Split bar ในดาต้าวินส์โดว์
Icon	กำหนด Icon ที่จะใช้งาน เมื่อ Data windows Minimize
LiveScroll	ค่าเป็นจริง เมื่อต้องการให้ผู้ใช้สามารถเลื่อนค่าไปมาโดยใช้ Scroll bar

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

MaxBox	เพิ่ม Maximize Button ใน Data Window
MinBox	เพิ่ม Minimize Button ใน Data Window
ReSizable	อนุญาตให้มีการเปลี่ยนขนาดของ DataWindow
TabOrder	ค่า Integer ที่ระบุความสัมพันธ์ของ Tab Order สำหรับ Control บน Data Window
Tag	ข้อความที่ทำงานกับ Control ใช้เก็บ Microsoft Help Text
Title	เพิ่ม Title ในดาต้าวินส์โดว์
TitleBar	มีค่าเป็นจริง เพื่อเพิ่ม Title bar ลงใน Data Window
Visible	มีค่าเป็นจริง เมื่อกำหนดให้แสดงคอนโทรล
VScrollBar	มีค่าเป็นจริง เมื่อมีการกำหนด Virtical Scrollbar ใช้งาน
Width	ค่า Integer บอกความกว้างของคอนโทรลใน PBUs
X	ค่า Integer แสดงตำแหน่งในแนวราบของคอนโทรลบน PBUs
Y	ค่า Integer แสดงตำแหน่งในแนวดิ่งของคอนโทรลบน PBUs

ตารางที่ 3-15 แสดงดาต้าวินโดว์คอนโทรลแอททริบิวต์

เหตุการณ์ที่สามารถเกิดขึ้นได้กับดาต้าคอนโทรล (DataWindow Control Event) มีดังนี้

Clicked	เกิดขึ้นเมื่อผู้ใช้คลิ้กในดาต้าวินส์โดว์
Constructure	เกิดขึ้นก่อนที่ทำการกำหนดวินส์โดว์ทำงาน
DBError	สั่ง Trigger เมื่อเกิดความผิดพลาดกับฐานข้อมูล
Destructure	เกิดขึ้นเมื่อทำการปิดวินส์โดว์
DoubleClicked	เกิดขึ้นเมื่อผู้ใช้ Double Click เลือกจาก List
DragDrop	เกิดขึ้นเมื่อกำหนด Drag Mode และ Pointer ชี้ไปที่ Object ใน คอนโทรล
DragEnter	เกิดขึ้นเมื่อกำหนด Drag Mode และ Drag Object ใน Control เมื่อทำการกำหนดเป็น Off และกำหนด DragAuto Attribute แล้วจะสั่ง Trigger เมื่อเกิดเหตุการณ์ DragEnter
DragLeave	เกิดขึ้นเมื่อกำหนด Drag Mode และทำการ Trigger เมื่อมีการ Drag Object ออกจาก Control
DragWithin	เกิดขึ้นเมื่อกำหนด Drag Mode และทำการ Trigger เมื่อมีการ Drag Object ใน Control

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

EditChanged	เกิดขึ้นเมื่อข้อมูลใน Edit Control ของ Data Window เปลี่ยนแปลง
GetFocus	เกิดขึ้นกรอนการเลือก Control โดยคลิ้ก หรือ Tab
ItemChanged	เกิดขึ้นเมื่อข้อความใน Edit Control เปลี่ยนไป
ItemError	สั่ง Trigger เมื่อมีเหตุการณ์ที่เปลี่ยนแปลง Item
ItemFocusChanged	สั่ง Trigger เมื่อผู้ใช้เปลี่ยนแปลง Focus บนดาต้าวินส์โดว์
LoseFocus	เกิดขึ้นก่อนที่คอนโทรงจะ Lose Focus เพราะ Tab หรือคลิ้กนอกคอนโทรล
Other	เหตุการณ์จากวินส์โดว์อื่นทำการ Map กับเหตุการณ์ในวินส์โดว์นี้
PrintEnd	สั่ง Trigger เมื่อพิมพ์ Data Window เสร็จ
PrintPage	สั่ง Trigger เมื่อเริ่มพิมพ์ Data Window ในแต่ละหน้า
PrintStart	สั่ง Trigger เมื่อเริ่มพิมพ์ Data Window
RButtonDown	เกิดขึ้นเมื่อทำการกดเมาส์ขวา และพอยน์เตอร์ชี้อยู่ในดาต้าคอนโทรล
Resize	สั่ง Trigger เมื่อมีการเปลี่ยนขนาดดาต้าวินส์โดว์
RetrieveEnd	สั่ง Trigger เมื่อดาต้าวินส์โดว์ภายหลังเรียกการทำงานแล้ว
RetrieveStart	สั่ง Trigger เมื่อดาต้าวินส์โดว์เริ่มทำการเรียกการทำงาน
RowFocusChanged	สั่ง Trigger ทุกครั้งที่ค่า Row เปลี่ยนแปลง
ScrollHorizontal	สั่ง Trigger เมื่อผู้ใช้เลื่อน Scroll ไปมาตามแนวราบ
Scroll Vertical	สั่ง Trigger เมื่อผู้ใช้เลื่อน Scroll ไปมาตามแนวดิ่ง
SQLPreview	สั้ง Trigger ก่อนที่คำสั่งเอสคิวแอลให้ค่าผลลัพธ์
UpdateEnd	สั่ง Trigger ภายหลัง update ขัดกับข้อมูลใน Database
UpdateStart	สั่ง Trigger เมื่อ update ขัดกับข้อมูลใน Database

ตารางที่ 3-16 แสดงเหตุการณ์ของดาต้าวินส์โดว์คอนโทรล

ฟังก์ชันการทำงานของดาต้าวินส์โดว์คอนโทรล (Data Window Control Function) มีดังนี้

AcceptText	เลื่อน Edit Control Text ไปยัง Current Item Buffer Cell
Clear	ลบข้อความต่างๆจากดาต้าวินส์โดว์ แต่ไม่เก็บใน ClipBoard
ClearValues	ลบค่าต่างๆในคอลัมน์บนดาต้าวินส์โดว์
Copy	Copy จากดาต้าวินส์โดว์ไปยัง ClipBoard
Cut	ลบคอนโทรลที่เลือกออกจากดาต้าวินส์โดว์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

DBCancel	ยกเลิกการเรียกข้อมูลปัจจุบัน
DBErrorCode	คืนรหัสแสดงความผิดพลาดที่เกิดขึ้น
DBErrorMessage	คืนข้อความที่แสดงความผิดพลาดที่เกิดขึ้น
DBHandle	คืนค่า Database Connection Handle
DeleteCount	คืนค่าจำนวนเรคอร์ดที่ถูกลบจากดาต้าวินส์โดว์
DeleteRow	ทำการลบเรคอร์ดออกจากดาต้าวินส์โดว์
Drag	เริ่มต้น และสิ้นสุด Drag Mode ของคอนโทรล
Filter	ทำการ Filter ในดาต้าวินส์โดว์
FilteredCount	คืนค่าจำนวนแถวที่ถูก Filter ในดาต้าวินส์โดว์
GetClickedColumn	คืนค่าจำนวนคอลัมน์ที่ถูกคลิ้ก
GetClickRow	คืนค่าจำนวนแถวที่ถูกคลิ้ก
GetColumn	คืนค่าคอลัมน์ปัจจุบัน
GetColumnName	คืนค่าชื่อของคอลัมน์ปัจจุบัน
GetData	รับค่าข้อมูลจากคอนโทรล
GetFormat	คืนค่ารูปแบบข้อมูลในคอลัมน์ปัจจุบัน
GetItemDate	คืนค่า Specific Datawindow item ในชนิดวันที่
GetItemDateTime	คืนค่า Specific Datawindow item ในชนิด DateTime
GetItemDecimal	คืนค่า Specific Datawindow item ในชนิดทศนิยม
GetItemNumber	คืนค่า Specific Datawindow item ในชนิดตัวเลข
GetItemString	คืนค่า Specific Datawindow item ในชนิดข้อความ
GetItemTime	คืนค่า Specific Datawindow item ในชนิดเวลา
GetRow	คืนค่าจำนวนเรคอร์ดที่มีในดาต้าวินส์โดว์
GetSelectedRow	คืนค่าภายในเรคอร์ดแรกที่ถูกเลือกไว้ในดาต้าวินส์โดว์
GetSQLSelect	คืนข้อความที่เป็นประโยคภาษาเอสคิวแอล คำสั่ง Select ที่ทำงานอยู่ในดาต้าวินส์โดว์
GetText	คืนข้อความใน Edit Control
Hide	ทำดาต้าวินส์โดว์คอนโทรลให้มองไม่เห็น
InsertRow	ทำการแทรกเรคอร์ดใหม่ลงในดาต้าวินส์โดว์
PostEvent	ทำการวางเหตุการณ์ลงในลำดับของเหตุการณ์ในดาต้าวินส์โดว์คอนโทรล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Print	พิมพ์ชุดสำรองของดาต้าวินส์โดว์
PrintCancel	ยกเลิกงานพิมพ์ปัจจุบัน
Reset	ลบดาต้าวินส์โดว์
Retrieve	ส่งคำสั่งภาษาเอสคิวแอลสำหรับดาต้าวินส์โดว์
RowCount	คืนค่าจำนวนเรคอร์ดปัจจุบันในดาต้าวินส์โดว์
ScrollNextPage	เลื่อนไปข้างหน้า 1 เพจในดาต้าวินส์โดว์
ScrollNextRow	เลื่อนไปข้างหน้า 1 เรคอร์ดในดาต้าวินส์โดว์
ScrollPriorPage	เลื่อนกลับหลัง 1 เพจในดาต้าวินส์โดว์
ScrollPriorRow	เลื่อนกลับหลัง 1 เรคอร์ดในดาต้าวินส์โดว์
ScrollToRow	เลื่อนตำแหน่งไปยังหมายเลขเรคอร์ดที่ระบุ
SelectedLength	คืนค่าจำนวนตัวอักษรที่เลือกไว้ในดาต้าวินส์โดว์
SelectedLine	คืนค่าข้อความที่ถูกเลือกในคอนโทรล
SelectedStart	คืนค่าตำแหน่งของอักษรตัวแรกที่เลือกไว้ในคอนโทรล
SelectedText	คืนค่าข้อความที่เลือกไว้ในคอนโทรล
SelectRow	เลือก Item บนลิสต์ โดยพิจารณาจากตำแหน่ง
SelectText	เลือกส่วนของข้อความในคอนโทรล
SetColumn	กำหนดคอลัมน์ปัจจุบัน
SetFocus	กำหนด Focus ลงบนดาต้าวินส์โดว์
SetFormat	กำหนดรูปแบบการแสดงคอลัมน์
SetItem	กำหนดค่าของ Item ในดาต้าวินส์โดว์
SetRow	กำหนดเรคอร์ดแถวปัจจุบันให้เป็นเรคอร์ดเฉพาะ
SetSQLSelect	กำหนดประโยคภาษาเอสคิวแอล Select ในดาต้าวินส์โดว์
SetTabOrder	กำหนดค่าลำดับความสัมพันธ์ของ Tab สำหรับคอลัมน์ในดาต้าวินส์โดว์
SetText	กำหนดข้อความลงใน Edit Control
SetTrans	กำหนดทรานสแอคชันออบเจคต์เพื่อใช้ในดาต้าวินส์โดว์ โดยทำการจัดการทรานสแอคชันด้วย
SetTransObject	กำหนดทรานสแอคชันออบเจคต์เพื่อใช้ในดาต้าวินส์โดว์
Show	กำหนดให้ดาต้าวินส์โดว์คอนโทรลสามารถมองเห็นได้
TriggerEvent	ทำการ Trigger คอนโทรลของดาต้าวินส์โดว์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Update	ทำการเก็บค่าที่เปลี่ยนแปลงทั้งหมดในดาต้าวินส์โดว์ กลับลงสู่ฐานข้อมูล

ตารางที่ 3-17 แสดงฟังก์ชันการทำงานของดาต้าคอนโทรล

นอกจากนี้ยังมีฟังก์ชันขั้นสูงของดาต้าวินส์โดว์คอนโทรลอีก

Create	สร้างดาต้าวินส์โดว์
Describe	ส่งค่าเกี่ยวกับส่วนประกอบดาต้าวินส์โดว์กลับคืน
Find	ทำการค้นหาเรคอร์ดแรกที่ตรงกับเงื่อนไข
FindRequired	ทำการหาคอลัมน์ถัดไปที่ทำการกำหนดเป็นคอลัมน์ที่ต้องการ
GetChild	ส่งค่าชื่อของดาต้าวินส์โดว์ของลูกกลับคืน

ตารางที่ 3-18 แสดงฟังก์ชันขั้นสูงของดาต้าวินโดว์คอนโทรล

3.3.1.4 การใช้งานดาต้าวินส์โดว์คอนโทรล

ในการใช้งานดาต้าวินส์โดว์คอนโทรล ต้องทำการใส่ดาต้าวินส์โดว์ลงในวินส์โดว์ และทำการกำหนดขนาด และตำแหน่งเหมือนกับคอนโทรลชนิดอื่นๆในวินส์โดว์เพนท์เตอร์ (Windows Painter) และขั้นต่อไปคือกำหนดการเชื่อมต่อแบบเฉพาะกับดาต้าวินส์โดว์คอนโทรล และดาต้าวินส์โดว์ออบเจคต์ โดยในดาต้าวินส์โดว์คอนโทรล มีคุณสมบัติที่แสดงชื่อของดาต้าวินส์โดว์ออบเจคต์ที่เกี่ยวข้อง ทำให้ดาต้าวินส์โดว์ออบเจคต์มีความเหมาะสมในการใช้งานแอพพลิเคชัน

การใช้งานดาต้าวินส์โดว์คอนโทรล ต้องทำการกำหนดค่าเริ่มต้นของทรานสแอคชันออบเจคต์ เช่น sqlca และทำการเชื่อมต่อกับฐานข้อมูล ซึ่งการทำงานในส่วนนี้มีลักษณะการทำงานเหมือนเอมเบดด์เอสคิวแอล เมื่อทำการกำหนดการติดต่อกับฐานข้อมูลสำเร็จ ต้องทำการกำหนดทรานสแอคชันออบเจคต์ ให้กับดาต้าวินส์โดว์คอนโทรลก่อนที่จะทำงานฐานข้อมูล จากนี้เองใช้ฟังก์ชันของดาต้าวินส์โดว์คอนโทรลในการนำและจัดการกับข้อมูล แทนที่จะใช้เอมเบดด์เอสคิวแอล โดยใช้ฟังก์ชัน Retrieve() แทนคำสั่ง Select ในภาษาเอสคิวแอล และใช้ฟังก์ชัน InsertRow(), DeleteRecord() ในการแทรกเรคอร์ดและทำการลบเรคอร์ดข้อมูล โดยการทำงานเหล่านี้มีผลต่อดาต้าวินส์โดว์เท่านั้น ยังไม่มีผลการเปลี่ยนแปลงกับฐานข้อมูลจนกว่าจะใช้ฟังก์ชัน Update()

ในการเขียนเพาเวอร์สคริปท์ จะมีความเกี่ยวข้องกับดาต้าวินส์โดว์คอนโทรล ซึ่งเป็นการเขียนสคริปท์สำหรับตอบสนองเหตุการณ์ โดยมีฟังก์ชันที่สำคัญเกี่ยวกับการจัดการข้อมูล ได้แก่

Retrieve() ส่งประโยคภาษาเอสคิวแอลให้กับดาต้าวินส์โดว์ และทำการจัดการเกี่ยวกับเรคอร์ดข้อมูลจากฐานข้อมูลบนดาต้าวินส์โดว์

InsertRow() ทำการแทรกเรคอร์ดข้อมูลลงในดาต้าวินส์โดว์ในตำแหน่งที่ต้องการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

DeleteRow() ทำการลบเรคอร์ดข้อมูลในตำแหน่งที่กำหนด จากดาต้าวินส์โดว์

Update() รับการเปลี่ยนแปลงต่าง ได้แก่ การเพิ่ม การลบ และการเปลี่ยนแปลงเรคอร์ดข้อมูล และทำการส่งค่าที่เปลี่ยนแปลงจากดาต้าวินส์โดว์ กลับสู่ฐานข้อมูล

3.3.1.5 การกำหนดดาต้าวินส์โดว์คอนโทรลลงในดาต้าวินส์โดว์

ในวินส์โดว์เพนท์เตอร์ทำการคลิ๊กที่ไอคอนของดาต้าวินส์โดว์ และเลือก Control ! DataWindow Menu เพื่อเป็นการบอกวินส์โดว์เพนท์เตอร์ว่าต้องการใช้ดาต้าคอนโทรลลงในวินส์โดว์ จากนั้นก็ทำการใส่ดาต้าวินส์โดว์คอนโทรลลงในวินส์โดว์ จากนั้นก็ทำการกำหนดตำแหน่งของดาต้าวินส์โดว์คอนโทรลในวินส์โดว์นั้น การเลือกดาต้าวินส์โดว์ออบเจคต์

ในแต่ละดาต้าวินส์โดว์คอนโทรลต้องมีดาต้าวินส์โดว์ออบเจคต์เฉพาะ ทำได้โดยเปิดเมนูดาต้าวินส์โดว์ (โดยการคลิ๊กขวาบนดาต้าวินส์โดว้คอนโทรล) และเลือก Change DataWindow จะทำการเปิด Select DataWindow Dialog หลังจากนั้นทำการเลือกแอพพลิเคชันไลบรารี (Application Libraries) ในส่วนล่างของลิสต์บ๊อกซ์ (ListBox) ที่เก็บค่าของดาต้าวินส์โดว์ออบเจคต์ จากนั้นทำการเลือกดาต้าวินส์โดว์ออบเจคต์จากลิสต์ใน DataWindows ListBox ที่อยู่ส่วนบนสุดใน Dialog Window แล้วคลิ๊ก OK

3.3.2 การใช้งานคำสั่งภาษาเอสคิวแอลในเพาเวอร์บิวเดอร์

3.3.2.1 เอมเบดด์เอสคิวแอลในเพาเวอร์สคริปท์ (Embeded SQL in PowerScript)

```
Transaction   trans1
trans1 = CREATE Transaction
trans1.DBMS       = "ODBC"
trans1.Database   = "Video Store DB"
trans1.UserId     = "dba"
trans1.DBParm     = "Connectstring='DSN=Video Store DB'"
CONNECT USING trans1;
UPDATE customer
  SET address1='123 Main Street'
  WHERE firstname='Gina' and lastname='Gray'
  USING trans1;
DISCONNECT USING trans1;
DESTROY trans1
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากโปรแกรมตัวอย่างเป็นตัวอย่างสคริปท์ที่ใช้เอมเบดด์เอสคิวแอล โดยใน 4 บรรทัดก่อนบรรทัด Update Customer เป็นเอมเบดด์เอสคิวแอล ส่วนประโยค Connect Using trans1 และ Disconnect trans1 ก็เป็นเอมเบดด์เอสคิวแอลเช่นกัน แต่โปรแกรมส่วนที่เหลือก็เป็นสคริปท์ทั่วไป

จะเห็นได้ว่าลักษณะการเขียนสคริปท์ต่างจากสคริปท์ทั่วไป แต่ในลักษณะนี้โดยส่วนมากจะเป็นแอพพลิเคชันที่สำหรับใช้งานจริง ในที่นี้มักใช้ประโยค Connect, Update และ Disconnect ในสคริปท์เหตุการ์ณของแอพพลิเคชัน โดยจะไม่ทำการเขียนรวมอยู่ในสคริปท์เพียงสคริปท์เดียว

ประโยค Update มีลักษณะเหมือนกับภาษาเอสคิวแอลโดยทั่วไป แต่ระหว่างภาษาเอสคิวแอลกับเพาเวอร์สคริปท์มีสิ่งที่แตกต่างกัน คือ

1. ประโยคเอสคิวแอลลงท้ายด้วยเครื่องหมาย ";" แต่ในเพาเวอร์สคริปท์ไม่ต้องใช้ .
2. ในเพาเวอร์สคริปท์ใช้อักษร "&" ในการต่อบรรทัด แต่ประโยคเอสคิวแอลไม่ต้องใช้

3.3.2.2 ทรานสแอคชัน ออบเจคต์ (Transaction Object)

จากโปรแกรมตัวอย่าง แสดงให้เห็นถึงการใช้ตัวแปร trans1 แต่ในแอพพลิเคชันส่วนมาก มักจะกำหนดในชื่อตัวแปร sqlca ที่เป็นตัวแปรชนิดพิเศษในเพาเวอร์บิวเดอร์

ตัวแปร trans1 เป็นชื่อของทรานสแอคชันออบเจคต์ที่ทำการเก็บข้อมูลเกี่ยวกับการเชื่อมต่อกับฐานข้อมูล และทำหน้าที่ทุกอย่างเกี่ยงกับการติดต่อระหว่างเพาเวอร์บิวเดอร์กับฐานข้อมูล

วิธีการประกาศ การสร้าง และการทำลายทรานสแอคชันออบเจคต์ มีดังนี้

1. ประโยค Transaction trans1 เป็นการประกาศตัวแปร trans1 เป็นทรานสแอคชันออบเจคต์

2. ประโยค trans1 = CREATE Transaction เป็นการกำหนดตำแหน่งของตัวแปรในหน่วยความจำ และทำการกำหนดค่าเริ่มต้นให้กับตัวแปรนั้น

3. ประโยค DESTROY trans1 เป็นคำสั่งสุดท้ายในสคริปท์ซึ่งมีการทำงานตรงข้ามกับ CREATE

พารามิเตอร์ของทรานสแอคชันที่กี่ยวข้องกับการเชื่อมต่อกับฐานข้อมูล มีดังนี้

DBMS	ชนิดของ Database Server เช่น ODBC, SYB SQL Server 4.x
Server Name	ชื่อของ Database Server
Database	ชื่อของ Database
UserID	รหัสผู้เข้าใช้งานฐานข้อมูล
dbPass	รหัสผ่านของผู้ใช้ในการติดต่อฐานข้อมูล
LogID	รหัสของผู้ใช้ในการติดต่อกับเซิร์ฟเวอร์
LogPass	รหัสผ่านสำหรับการเข้าถึงเซิร์ฟเวอร์
Lock	ระดับการล็อคข้อมูลขณะทำทรานสแอคชัน
dbParm	พารามิเตอร์เพิ่มเติมเฉพาะของระบบจัดการฐานข้อมูลแต่ละตัว
AutoCommit	กำหนดการ commit แต่ละประโยคในระหว่างการประมวลผล โดยอัตโนมัติ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 3-19 แสดงพารามิเตอร์ของทรานสแอคชันที่กี่ยวข้องกับการเชื่อมต่อกับฐานข้อมูล

ทุกค่าพร็อพเพอร์ตี้มีชนิดเป็น String ยกเว้น Autocommit จะมีค่าเป็นบูลลีน นอกเหนือจากนี้ยังมีอีก 5 พร็อพเพอร์ตี้ ที่ใช้ในการส่งรหัสผิดพลาดให้แก่ผู้ใช้ กับข้อมูลอื่นๆ ตามประโยคภาษาเอสคิวแอล อีกด้วย

ตัวอย่างเอมเบดด์เอสคิวแอลในเพาเวอร์สคริปท์

```
CONNECT USING trans1;
UPDATE customer
        SET address1='123 Main Street'
        WHERE firstname='Gina' and lastname='Gray';
USING trans1;
DISCONNECT USING trans1;
```

ประโยค connect เป็นการทำการเชื่อมต่อกับฐานข้อมูล มีลักษณะเหมือนกับการเลือก File|Connect ในดาต้าเบสเพนท์เตอร์ (Database Painter) สำหรับประโยค Update ใช้สำหรับทำการเปลี่ยนแปลงข้อมูลในตาราง และ DisConnect มีการทำงานตรงกันข้ามกับ Connect เพื่อบอกการสิ้นสุดการเชื่อมต่อกับฐานข้อมูล

ประโยค Using เป็นประโยคที่สำคัญในเพาเวอร์บิวเดอร์ ซึ่งจะทำการส่งประโยคเอสคิวแอลที่เขียนอยู่ในสคริปท์ไปยังฐานข้อมูล ในกรณีที่แอพพลิเคชันติดต่อกับฐานข้อมูลหลายๆตัว จะต้องทำการกำหนดตัวแปรทรานสแอคชันออบเจคต์ให้ต่างกันในการเชื่อมต่อแต่ละฐานข้อมูล แต่ในกรณีที่ใช้ชื่อ sqlca สำหรับทรานสแอคชันออบเจคต์แล้ว ก็ไม่จำเป็นต้องใช้ประโยค Using

3.3.3 การเชื่อมต่อกับฐานข้อมูลโดยใช้เพาเวอร์บิวเดอร์

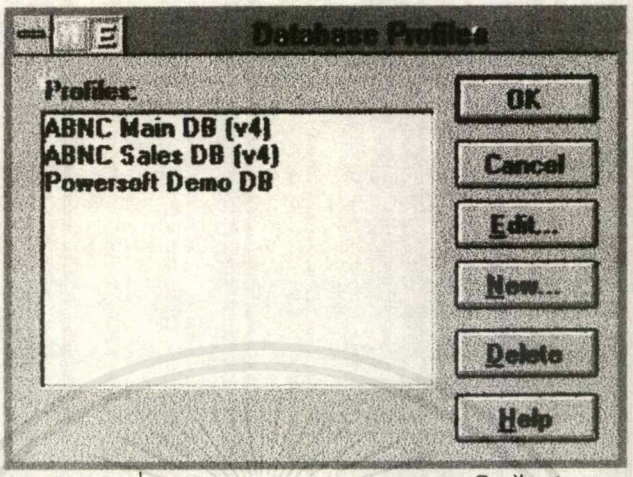
เพาเวอร์บิวเดอร์ สามารถเชื่อมต่อกับฐานข้อมูลโดยผ่านโอดีบีซีไดร์ฟเวอร์ โดยต้องทำการเพิ่มโอดีบีซีดาต้าซอสที่ต้องการก่อน จากนั้นจึงกำหนดให้เพาเวอร์บิวเดอร์ทำการติดต่อกับดาต้าซอสนั้น

เพาเวอร์บิวเดอร์ จะเก็บข้อมูลรายละเอียดการติดต่อกับฐานข้อมูลแต่ละตัวไว้ในดาต้าเบสโพรไฟล์ (Database Profile) ซึ่งเก็บในชื่อไฟล์ PB.INI การกำหนดค่าการใช้งานของฐานข้อมูลสามารถทำได้ดังนี้

1. เรียกดาต้าเบสเพนท์เตอร์ โดยเลือกจากปุ่มบน Power
2. ทำการเพิ่มโอดีบีซีดาต้าซอสของไดร์ฟเวอร์ที่ต้องการ โดยใช้โอดีบีซีแอดมินิสเทรเตอร์ หรือใช้คำสั่ง File|Configure ODBC (โดยถ้าใช้คำสั่งในเมนูบนเพาเวอร์บิวเดอร์จะต้องมีไดร์ฟเวอร์ที่ต้องการติดตั้งอยู่แล้ว) เลือกไดร์ฟเวอร์ที่ต้องการ จากนั้นเลือก Create เพื่อทำการติดตั้งดาต้าซอส
3. ทำการกำหนดค่าของดาต้าเบสโพรไฟล์ (Database Profile) โดยใช้คำสั่ง File|Connect|Setup

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เลือก New จากนั้นใส่ค่าที่ต้องการให้ครบ



รูปที่ 3-18 แสดงการกำหนดดาต้าเบสโปรไฟล์

Profile Name คือ ชื่อที่ใช้ติดต่อระหว่างแอพลิเคชันกับฐานข้อมูล

DBMS คือ ฐานข้อมูลที่ต้องการทำการติดต่อ กรณีนี้เลือกเป็น ODBC และเลือก MORE >> แล้วจะมีค่าให้ใส่เพิ่มอีกส่วนหนึ่ง

Server Name คือ ชื่อของดาต้าเบสเซิร์ฟเวอร์ที่ต้องการติดต่อ

Login ID คือ ชื่อที่ใช้เข้าทำงานกับเซิร์ฟเวอร์

Login Password คือ รหัสผ่านเพื่อทำงานกับเซิร์ฟเวอร์

DBParm คือ ค่ากำหนดสำหรับการติดต่อ กำหนดดังนี้

ConnectString = ' DSN= ; UID= ;PWD= ;

โดย DSN หมายถึง ชื่อของดาต้าซอส (Datasource Name)

UID หมายถึง ชื่อที่ใช้ติดต่อกับเซิร์ฟเวอร์

PWD หมายถึง รหัสผ่านในการเข้าทำงานกับเซิร์ฟเวอร์

การกำหนดสามารถทำการกำหนดเพียง Profile Name แล้วกำหนด DBParm ให้ถูกต้อง ก็สามารถใช้งานได้โดยไม่จำเป็นต้องกำหนดค่าจนครบ

4. สามารถใช้งานฐานข้อมูลได้โดยใช้คำสั่ง FilelConnect แล้วเลือก ProfileName ที่สร้างไว้

3.3.4 การทำงานกับฐานข้อมูลโดยใช้เพาเวอร์บิวเดอร์

การติดต่อกับฐานข้อมูลโดยใช้เพาเวอร์บิวเดอร์ จำเป็นต้องใช้ทรานสแอคชันออบเจคต์เสมอ โดยปกติใช้ทรานสแอคชันออบเจคต์ sqlca ซึ่งมีประกาศและสร้างขึ้นใช้งานโดยอัตโนมัติ ไม่จำเป็นต้องทำการประกาศและสร้างเอง การติดต่อกับฐานข้อมูลมักทำเพียงครั้งเดียวตอนเริ่มต้น คือในส่วนคำสั่งตอบสนองการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เปิด (Open Event) แอพพลิเคชัน ซึ่งต้องมีการกำหนดค่าต่างๆของ sqlca เช่น DBMS, Database เป็นต้น จากนั้นใช้คำสั่ง CONNECT เป็นการติดต่อกับฐานข้อมูลได้ ซึ่งจะทำการติดต่ออยู่ตลอดเวลาจนกว่าจะมีการใช้คำสั่ง DISCONNECT ซึ่งมักเป็นคำสั่งในส่วนตอบสนองเหตุการณ์ปิด (Close Event) แอพพลิเคชัน

3.3.5 การใช้งาน sqlca - พื้นที่การสื่อสารภาษาเอสคิวแอล (SQL Communication Area)

เพาเวอร์บิวเดอร์จะทำการประกาศ และสร้างตัวแปรทรานสแอคชันออบเจคต์ sqlca โดยอัตโนมัติ โดยข้อดีของการใช้ sqlca แทนการใช้ชื่อตัวแปรที่ทำการประกาศเป็นทรานสแอคชันออบเจคต์ คือ

1. ไม่ต้องทำการประกาศ sqlca และไม่ต้องทำการ CREATE และ DESTROY เพราะเพาเวอร์บิวเดอร์ทำให้โดยอัตโนมัติ
2. ไม่มีความจำเป็นต้องใช้ประโยค Using ในทุกๆคำสั่งสุดท้ายของประโยคภาษาเอสคิวแอล เพราะเพาเวอร์บิวเดอร์จะทำการกำหนด Using sqlca เป็นค่าดีฟอลท์

ดังนั้น ถ้าหากใช้ sqlca จะทำให้การเขียนโค้ดน้อยลง แต่ถ้าในกรณีที่ทำการติดต่อกับฐานข้อมูลมากกว่า 1 แห่ง ก็มีความจำเป็นที่ต้องประกาศตัวแปรทรานสแอคชันออบเจคต์ ที่ไม่ใช่ sqlca ด้วย

ใน sqlca จะมีพร็อพเพอร์ตี้เพื่อบอกความผิดพลาด เช่น sqlca.sqlcode หรือ sqlca.errtext สามารถนำไปใช้ทำการตรวจสอบความผิดพลาดที่เกิดขึ้นได้

3.3.6 การทำงานกับข้อมูลโดยใช้เพาเวอร์บิวเดอร์

หลังจากผ่านขั้นตอนของการติดต่อกับฐานข้อมูลเรียบร้อยแล้ว เราสามารถทำงานกับข้อมูล ได้โดยวิธีหลักๆต่อไปนี้

1.ใช้การทำงานผ่านภาษาเอสคิวแอลโดยตรง ซึ่งสามารถทำการจัดการกับข้อมูลได้ทุกรูปแบบ สามารถใช้ตัวแปรเข้าร่วมในคำสั่งเอสคิวแอลด้วยได้ ซึ่งทำให้การทำงานยืดหยุ่นมากยิ่งขึ้น เช่นสามารถนำค่าคอนโทรลต่างๆมาทำการเก็บหรือแก้ไขข้อมูลในตารางได้

การค้นหาข้อมูล (Select) ก็สามารถทำได้เช่นเดียวกัน โดยถ้าผลลัพธ์ที่ได้มีเพียงเรคอร์ดเดียว ก็สามารถใช้ Select .... INTO เพื่อทำการเก็บผลลัพธ์ในตัวแปร และสามารถนำค่าไปใช้ต่อได้ กรณีที่มีผลลัพธ์จำนวนหลายเรคอร์ดจะต้องทำการประกาศเคอร์เซอร์ (Cursor) ซึ่งเปรียบเสมือนตัวชี้ผลลัพธ์ จากนั้นทำการสั่งให้เคอร์เซอร์ทำงาน ซึ่งจะได้กลุ่มของผลลัพธ์ที่สามารถนำมาใช้ได้ทีละ 1 เรคอร์ด โดยใช้คำสั่ง Fetch นอกจากนี้ยังสามารถทำการแก้ไข (Update) ข้อมูลในตำแหน่งที่เคอร์เซอร์ชี้อยู่ได้อีกด้วย

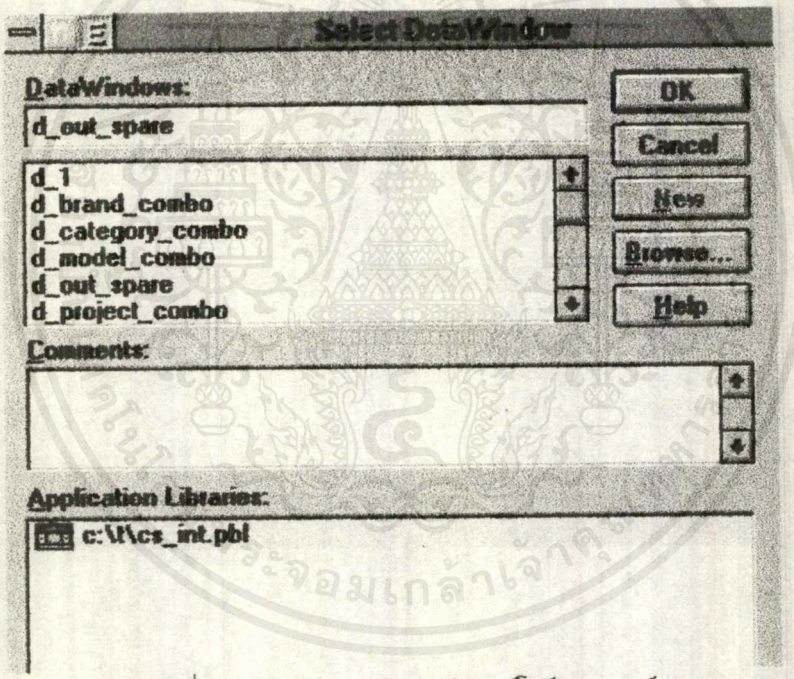
การทำงานโดยใช้คำสั่งเอสคิวแอลโดยตรง เป็นการทำงานที่ตรงที่สุด และมีความสามารถที่ดีที่สุด (ทั้งนี้ขึ้นอยู่กับความสามารถของผู้สร้างแอพพลิเคชันด้วย) แต่มีข้อเสียคือ ความยุ่งยากมนการทำงานมีมาก เพราะนอกจากจะต้องควบคุมขั้นตอนการทำงานของแอพพลิเคชันแล้ว ยังต้องควบคุมการแก้ไขและค้นหาข้อมูลเอง นอกจากนี้จะต้องมีความรู้ในภาษาเอสคิวแอลอีกด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. การทำงานผ่านดาต้าวินส์โดว์คอนโทรล และดาต้าวินส์โดว์ออบเจคต์ ซึ่งทั้งสองตัวนี้เป็นสิ่งช่วยในการอำนวยความสะดวกในการทำงานกับข้อมูล โดยไม่ต้องใช้คำสั่งภาษาเอสคิวแอล และ การแสดงข้อมูลสามารถทำได้โดยอัตโนมัติ โดยข้อมูลที่ต้องการจะถูกกำหนดระหว่างการสร้างดาต้าวินส์โดว์ออบเจคต์ และสามารถแสดงข้อมูลจากหลายตารางได้อย่างง่ายดาย

การแก้ไขข้อมูล ผู้ใช้สามารถทำได้โดยตรงผ่านดาต้าวินส์โดว์คอนโทรล หรืออาจทำการแก้ไขโดยใช้คำสั่งของดาต้าวินส์โดว์คอนโทรลได้ ซึ่งมีคำสั่งมากมายที่ครอบคลุมการทำงานได้ทั้งหมด

การทำงานโดยวิธีนี้มีข้อได้เปรียบ คือความสะดวกและง่าย ไม่จำเป็นต้องควบคุมการทำงานเอง ไม่จำเป็นต้องรู้คำสั่งเอสคิวแอล แต่ต้องทำการเรียนรู้คำสั่ง และข้อกำหนดต่างๆของดาต้าวินส์โดว์อย่างแท้จริง จึงจะสามารถทำงานได้มีประสิทธิภาพ และข้อเสียอีกประการหนึ่งคือ ผู้พัฒนาจะไม่สามารถควบคุมการทำงานได้อย่างเต็มที่



รูปที่ 3-19 แสดงการเลือกดาต้าวินส์โดว์ออบเจคต์

ในกรณีที่ต้องการความช่วยเหลือในการกำหนดดาต้าวินส์โดว์ออบเจคต์ ทำได้โดยคลิ้ก Browse เพื่อทำการเปิด Browse Data Windows Dialog Window โดยในไดอะล็อกซ์บ็อกซ์ชั้นแรกให้ทำการเลือกไลบรารี (.PBL) ที่ต้องการค้นหา และทำการกำหนดเงื่อนไขการค้นหาใน Search for Field และคลิ้ก Search แล้วใน Match Listbox จะทำการแสดงดาต้าวินส์โดว์ทั้งหมดที่ตรงกับเงื่อนไขในการค้นหา นอกจากนี้ยังสามารถกำหนดให้ค้นหาแบบพิจารณาชนิดและขนาดตัวอักษรได้อีกด้วย (Case Sensitive) โดยคลิ้กใน CheckBox

3.3.6.1 การเปิดใช้งาน Global connection

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยปกติ เราต้องการให้การติดต่อกับฐานข้อมูลยังคงดำเนินอยู่ตลอดเวลาที่ทำงานดังนั้นจึงมีความเหมาะสมที่จะทำการกำหนดการเชื่อมต่อในเหตุการณ์ Open และตำแหน่งที่เหมาะสมที่สุดในการยกเลิกการติดต่อคือ ในเหตุการณ์ Close ของแอพพลิเคชัน

3.3.6.2 การแทรก และลบเรคอร์ดข้อมูล (Insert and Delete)

การแทรกเรคอร์ดข้อมูล และการลบเรคคอร์ดข้อมูลในเพาเวอร์สคริปท์ มีลักษณะการเขียนเหมือนกับชุดคำสั่งในภาษาเอสคิวแอล

```
INSERT INTO customer
  (custid, lastname, firstname, address1, city, state)
  VALUES ('9999', 'Johnson', 'Janet',
        '123 Main Street', 'Los Angeles', 'CA');
IF sqlca.SqlCode <> 0 THEN
  MessageBox("Error on Insert",                    &
    "Error Code: " + String(sqlca.SqlDbCode) +  &
    "~nError Message= " + sqlca.SqlErrText,     &
    stopsign!)
  RETURN
END IF

DELETE FROM customer
  WHERE firstname='Janet' and lastname='Johnson';
IF sqlca.SqlCode <> 0 THEN
  MessageBox("Error on Delete",                    &
    "Error Code: " + String(sqlca.SqlDbCode) +  &
    "~nError Message= " + sqlca.SqlErrText,     &
    stopsign!)
  RETURN
END IF
```

ในกรณีที่ไม่มีค่าตรงกับเงื่อนไข จะมีการใส่ค่าของรหัสความผิดพลาดให้แก่ผู้ใช้งานโปรแกรมได้ทราบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3.6.3 การใช้ตัวแปรในประโยคภาษาเอสคิวแอล

การใช้ตัวแปรในเพาเวอร์สคริปท์ จะทำการแทนค่าลงในประโยคภาษาเอสคิวแอล โดยมีเครื่องหมาย ":" นำหน้าชื่อตัวแปร

```
String s1,s2,s3
s1 = "9999"
s2 = "Johnson"
s3 = "Janet"
INSERT INTO customer
  (custid, lastname, firstname)
VALUES (:s1, :s2, :s3);
```

3.3.6.4 คำสั่ง SELECT

ในประโยคเอมเบดด์เอสคิวแอล ใช้สำหรับอ่านค่าข้อมูล 1 เรคอร์ดจากฐานข้อมูล

```
String s1,s2
SELECT custid, address1
  INTO :s1, :s2
  FROM customer
  WHERE firstname='Gina' and lastname='Gray';
```

ในกรณีที่ต้องการกำหนดเป็นตัวแปร เพื่อรองรับค่าที่ได้จากประโยคภาษาเอลคิวแอล ให้ทำได้โดยเพิ่มคำสั่ง INTO และตามด้วยชื่อตัวแปรที่ต้องการให้มารับข้อมูลลงไปในสคริปท์

การคืนค่าจากคำสั่ง SELECT สามารถแสดงค่ารหัสความผิดพลาดได้ ดังนี้

0	ถ้ามี 1 เรคอร์ดถูกเลือกจากฐานข้อมูล
100	ถ้าไม่มีเรคอร์ดใดถูกเลือกเลย จากฐานข้อมูล
-1	ถ้ามีความผิดพลาดเกิดขึ้น หรือมีจำนวนเรคอร์ดมากเกินกว่า 1 เรคอร์ด

ตารางที่ 3-20 แสดงค่ารหัสความผิดพลาดคำสั่ง SELECT

ในกรณีที่ต้องการข้อมูลจากฐานข้อมูลมากกว่า 1 เรคอร์ด สามารถทำได้โดยการใช้เคอร์เซอร์ (Cursor)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3.7 การทำทรานสแอคชัน

การทำทรานสแอคชัน สามารถทำได้ผ่าน sqlca เช่นเดียวกัน โดยหากต้องการใช้งานทรานสแอคชัน จำเป็นต้องกำหนดค่าเริ่มต้น sqlca.Autocommit = false ก่อนที่จะมีการติดต่อกับฐานข้อมูล ซึ่งจะทำให้เพาเวอร์บิวเดอร์ทำการส่งคำสั่ง Begin TRANS ทุกครั้งที่ทำงานคำสั่งแรก นั่นคือมีการเริ่มต้นทรานสแอคชันโดยอัตโนมัติ และเมื่อต้องการจบทรานสแอคชัน จำเป็นต้องใช้คำสั่งจบ ได้แก่ Commit กรณีที่ตรวจสอบแล้วว่าการทำทรานสแอคชันเสร็จสมบูรณ์ หรือ RollBack ในกรณีที่พบว่ามีความผิดพลาดระหว่างการทำทรานสแอคชัน โดย คำสั่ง RollBack จะทำการยกเลิกการเปลี่ยนแปลงทั้งหมดที่ทำมาหลังจากเริ่มทรานสแอคชัน

```
sqlca.dbms = "ODBC"
Sqlca.Database = "CS_Int"
sqlca.UserID = "s6014031"
sqlca.DBParm =
ConnectString='DSN=CS_Int;UID=S6014031;PWD=interface;APP=PowerBuilder -
cs_int;WSID=SUNEO;DATABASE=CS_Interface"
sqlca.autocommit = false
CONNECT
        If SQLca.SQLCode <> 0 then
                MessageBox("Error on connect to database", &
                "Error Code: " + String(sqlca.SqlDbcode) + &
                "~nError Message= " + sqlca.SqlErrText, &
                StopSign!)
                HALT
        End if
Open(w_1)
```

คอมมิทเป็นการบอกความแน่นอนที่ได้เปลี่ยนแปลงข้อมูลในฐานข้อมูลแล้ว แต่โรลแบคเป็นการบอกยกเลิกการเปลี่ยนแปลงข้อมูลที่มีทั้งหมดในทรานสแอคชัน ซึ่งทั้ง 2 แบบเป็นการบอกสิ้นสุดทรานสแอคชัน โดยในเพาเวอร์บิวเดอร์สามารถใช้ได้โดยใช้ประโยค Using เพราะ Commit และ Rollback เป็นเอมเบดด์เอสคิวแอล

ออโต้คอมมิท เป็นพร็อพเพอร์ตี้ที่สำคัญมีผลต่อทรานสแอคชัน โดยมีชนิดเป็นบูลลีย และสามารถทำการกำหนดก่อนหรือหลังการติดต่อกับฐานข้อมูลก็ได้

ถ้า AutoCommit = False แล้วทุกคำสั่งในชุดคำสั่งภาษาเอสคิวแอล จะต้องทำการ Commit หรือ Rollback ถึงจะทำให้คำสั่งที่ผ่านมามีผลได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ถ้า AutoCommit = True แล้วเสมือนไม่มีทรานสแอคชัน เพราะทุกคำสั่งเอสคิวแอลจะถูกประมวลผลโดยทันที

3.3.8 การจัดการเกี่ยวกับความผิดพลาด (Error Handling)

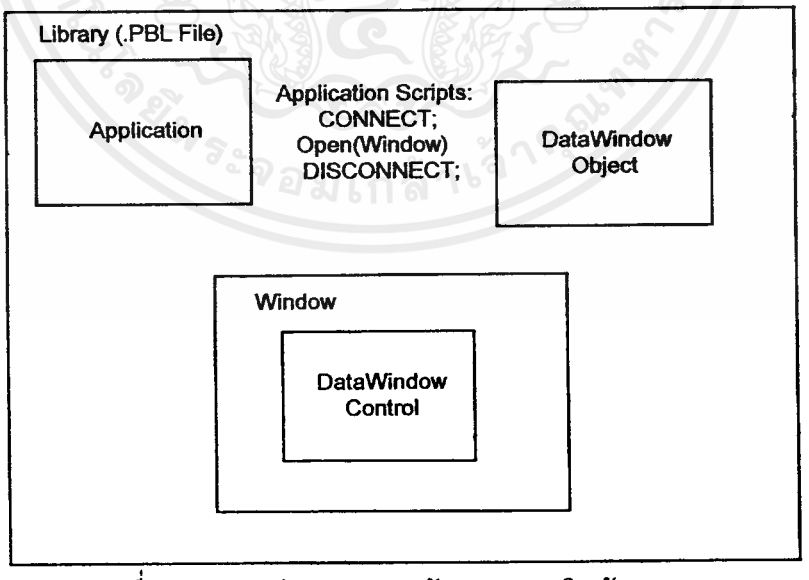
ในเพาเวอร์บิวเดอร์ เมื่อมีความผิดพลาดจากการใช้คำสั่งเอสคิวแอลเกิดขึ้น ในทรานสแอคชันออบเจคต์มีพร็อพเพอร์ตี้ที่เกี่ยวกับการแสดงความผิดพลาด และข้อมูลอื่นๆที่เกี่ยวข้องกับคำสั่งเอสคิวแอล ได้แก่

SqlCode	รหัสที่บอกถึงความสำเร็จ หรือความล้มเหลวในการทำคำสั่งเอสคิวแอล โดย 0 = NoErrors, -1 = Error และ 100 = NoRecord
SqlNRows	จำนวนเรคอร์ดที่ทำงานด้วย
SqlDBCode	รหัสความผิดพลาดที่เป็นรหัสเฉพาะของฐานข้อมูล
SqlErrText	ข้อความที่บอกความผิดพลาดที่เกิดขึ้น
SqlReturnData	ข้อความที่เกี่ยวข้องกับข้อมูลที่รับคืนจากฐานข้อมูล

ตารางที่ 3-21 แสดงพร็อพเพอร์ตี้ที่เกี่ยวกับการแสดงความผิดพลาด

3.3.9 การสร้างแอพพลิเคชันที่ทำงานกับข้อมูลอย่างง่ายโดยเพาเวอร์บิวเดอร์

ส่วนประกอบที่จำเป็นในการใช้สร้างแอพพลิเคชัน ได้แก่



รูปที่ 3-20 แสดงส่วนประกอบหลักของแอพพลิเคชัน

1. แอพพลิเคชัน ประกอบด้วย

1.1 แอพพลิเคชันออบเจคต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.2 วินส์โดว์

1.3 สคริปท์ สำหรับเหตุการณ์เพื่อทำการเปิดวินส์โดว์

1.4 ไลบรารี

2. ดาต้าวินส์โดว์ออบเจคต์ ซึ่งสร้างโดยดาต้าวินส์โดว์เพนท์เตอร์

3. ดาต้าวินส์โดว์คอนโทรล ประกอบด้วยดาต้าวินส์โดว์ออบเจคต์ และวางอยู่บนวินส์โดว์

4. คำสั่งเพาเวอร์สคริปท์ ใช้สำหรับเตรียม และจัดการกับฐานข้อมูล และดาต้าวินส์โดว์คอนโทรล โดยคำสั่งเหล่านี้จะอยู่ในสคริปท์เหตุการณ์ และแอพพลิเคชัน ได้แก่

4.1 CONNECT และคำสั่งอื่นๆที่เกี่ยวข้อง ใช้สำหรับกำหนดการเชื่อมต่อกับฐานข้อมูล ทำในสคริปท์ของเหตุการณ์เปิด (Open Script)

4.2 DISCONNECT ใช้สำหรับยกเลิกการติดต่อกับฐานข้อมูล ทำในสคริปท์ของเหตุการณ์ปิด (Close Script)

4.3 ฟังก์ชัน SetTransObj ใช้สำหรับกำหนดทรานสแอคชันออบเจคต์ โดยทั่วไปเป็น sqlca ทำในสคริปท์ของเหตุการณ์เปิด (Open Script) หรือสคริปท์ของเหตุการณ์สร้างดาต้าวินส์โดว์คอนโทรล (DataWindow Control's Constructure Script)

4.4 ฟังก์ชัน Retrieve ใช้สำหรับอ่านเรคอร์ดข้อมูลของฐานข้อมูล และแสดงบนดาต้าวินส์โดว์ ทำในสคริปท์ของเหตุการณ์เปิด สคริปท์ของเหตุการณ์การสร้างดาต้าวินส์โดว์คอนโทรล สคริปท์ของเหตุการณ์การคลิ้กปุ่มคำสั่ง (Button Click Script) สคริปท์ของเหตุการณ์การเลือกจากเมนู (Menu item's Script)

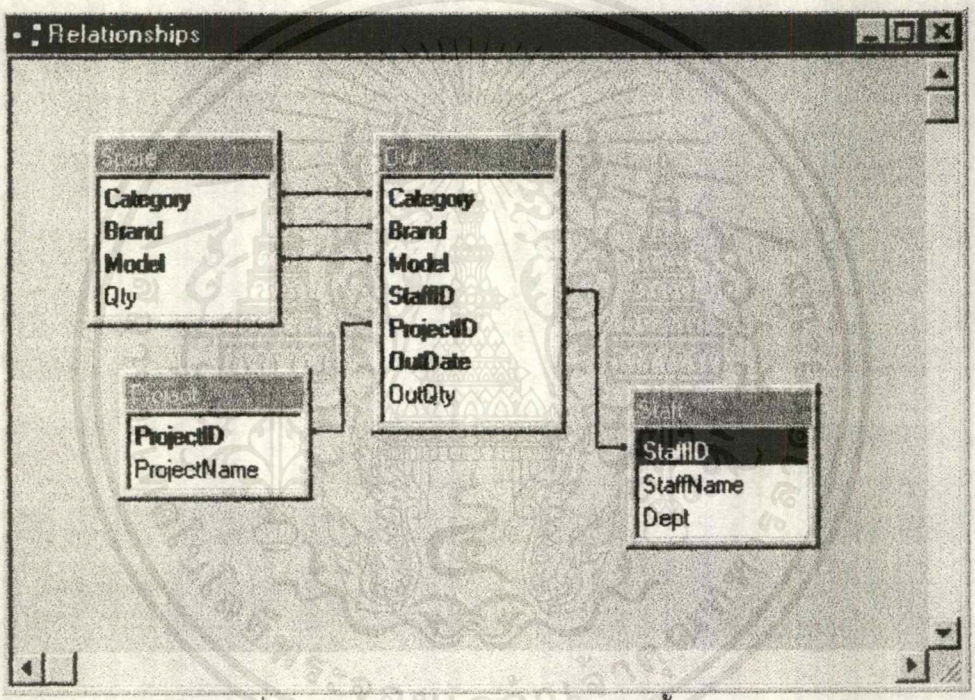
4.5 ฟังก์ชัน Update ใช้สำหรับเขียนความเปลี่ยนแปลงกลับลงสู่ฐานข้อมูล ในกรณีที่วินส์โดว์อนุญาตให้ผู้ใช้สามารถทำการเปลี่ยนแปลงข้อมูลได้ ทำในสคริปท์ของเหตุการณ์การปิด สคริปท์ของเหตุการณ์การทำลายดาต้าวินส์โดว์คอนโทรล (DataWindow Control's Destructor Script) สคริปท์ของเหตุการณ์การกดปุ่มคำสั่ง และสคริปท์ของเหตุการณ์การเลือกจากเมนู

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
การสร้างแอพพลิเคชันเพื่อทำการติดต่อกับฐานข้อมูล

4.1 กรณีศึกษา : แอพพลิชันโปรแกรมทำงานกับฐานข้อมูล

ในการทำงานได้ทดลองสมมติกรณีตัวอย่างในการทำงานกับฐานข้อมูล โดยลักษณะการทำงานจะมีการทำงานร่วมกับตาราง 4 ตาราง ซึ่งตามตัวอย่างจะเป็นลักษณะของข้อมูลเกี่ยวกับอุปกรณ์คอมพิวเตอร์ และผู้ใช้ทำการเบิกอุปกรณ์ต่างๆที่มีอยู่ไปใช้งานตามแผนกต่างๆขององค์กร มีความสัมพันธ์ ดังนี้



รูปที่ 4-1 แสดงความสัมพันธ์ระหว่างตารางทั้งหมด

ตาราง Spare เป็นตารางที่เก็บรายละเอียดของอุปกรณ์แต่ละชนิด ว่าชนิด (Category)ใด รุ่น (Model) ใด และยี่ห้อ (Brand) อะไร และมีจำนวน (Qty) เท่าใด

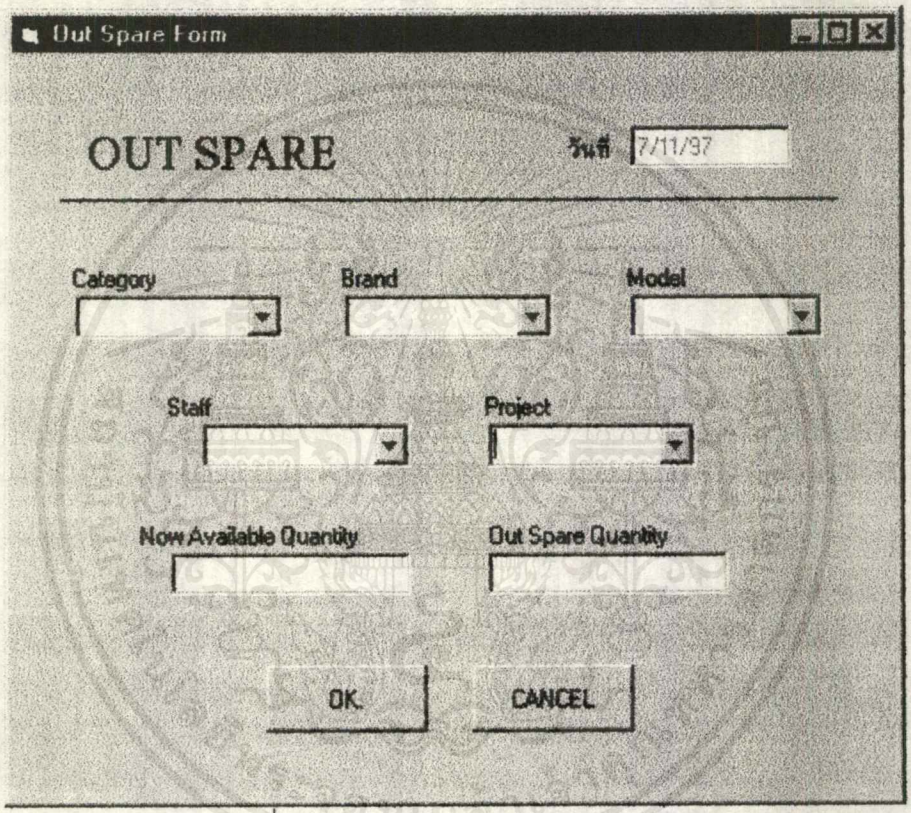
ตาราง Out เป็นตารางที่เก็บข้อมูลเกี่ยวกับการเบิกอุปกรณ์ไปใช้งาน โดยทำการเก็บข้อมูลว่าอุปกรณ์ที่เบิกไปเป็นชนิดใด (Category) ยี่ห้ออะไร (Brand) และรุ่นใด (Model) นอกจากนี้ยังเก็บด้วยว่า บุคคลใดเป็นผู้เบิก (StaffID) เบิกไปใช้ในงานอะไร (ProjectID) เมื่อใด (OutDate) และจำนวนเท่าใด (OutQty)

ตาราง Staff และ Project เก็บข้อมูลของผู้เบิก และ งาน ซึ่งสัมพันธ์กับข้อมูลในตาราง Out

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตามตัวอย่างเป็นส่วนการรับข้อมูลจากผู้ใช้ และประมวลผลการเบิกอุปกรณ์ โดยรับข้อมูลว่าอุปกรณ์ที่ต้องการเบิกเป็นอุปกรณ์ใด และทำการตรวจสอบว่าอุปกรณ์ที่ต้องการมีจำนวนเท่าใด มีเพียงพอสำหรับการเบิกหรือไม่ ข้อมูลที่รับจากผู้ใช้ครบหรือไม่ จากนั้นทำการจัดการกับข้อมูลใน 2 ลักษณะ คือ

1. ทำการเก็บข้อมูลการเบิกอุปกรณ์ ลงในตาราง Out
2. ทำการแก้ไขจำนวนอุปกรณ์คงเหลือ ในตาราง Spare
3. ข้อมูลของผู้เบิกและงาน ได้มาจากตาราง Staff และ Project



รูปที่ 4-2 แสดงหน้าจอของแอพพลิเคชัน

ลักษณะการใช้งาน

1. ผู้ใช้สามารถป้อนข้อมูลที่ส่วนใดก่อนก็ได้

2. การป้อนข้อมูลที่เกี่ยวกับรายละเอียดของอุปกรณ์ จะเป็นลักษณะของการเลือกข้อมูลจากคอมโบบ็อกซ์ (Combo Box) 3 ตัว คือ ชนิด ยี่ห้อ และรุ่น โดยต้องทำการเลือกตามลำดับ เมื่อทำการเลือกชนิดของอุปกรณ์แล้ว ค่าของยี่ห้อที่สามารถเลือกได้ต้องผ่านการกรองตามชนิดมาแล้วเท่านั้น ในทำนองเดียวกันกับ ค่าของรุ่นที่สามารถเลือกได้ ก็ต้องผ่านการกรองตามชนิด และยี่ห้อมาด้วย โดยรายละเอียดเกี่ยวกับอุปกรณ์มาจากตาราง Spare

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. เมื่อผู้ใช้ทำการกำหนดรายละเอียดของอุปกรณ์ที่ต้องการเบิกเรียบร้อยแล้ว โปรแกรมจะแสดงจำนวนที่เหลืออยู่ของอุปกรณ์นั้นให้ผู้ใช้ทราบ ซึ่งได้ค่ามาจากที่เก็บในตาราง Spare

4. เมื่อผู้ใช้ทำการระบุข้อมูลเกี่ยวกับการเบิกอุปกรณ์ครบแล้ว และกดปุ่ม OK แล้ว โปรแกรมจะทำการตรวจสอบดังนี้

4.1 มีการระบุข้อมูลครบทุกช่องหรือไม่

4.2 มีการระบุจำนวนอุปกรณ์ที่ต้องการเบิก เกินกว่าจำนวนที่มีอยู่จริงหรือไม่

4.3 มีการระบุจำนวนที่ต้องการเบิก เป็นค่าศูนย์หรือไม่

ถ้าหากพบความผิดพลาด ก็ทำการแจ้งให้ผู้ใช้ทราบ และกลับมายังหน้าจอที่รับข้อมูลอีกครั้ง

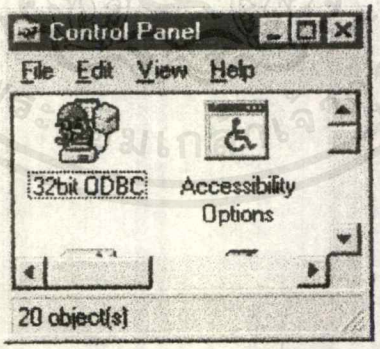
5.ถ้าการระบุข้อมูลทุกอย่างถูกต้อง โปรแกรมจะทำการคำนวณจำนวนอุปกรณ์ที่คงเหลือใหม่ หลังจากการเบิกไปแล้ว และทำการแก้ไขข้อมูลในตาราง Spare จากนั้น ทำการเพิ่มเรคอร์ดข้อมูลรายละเอียดในการเบิกอุปกรณ์ ลงในตาราง Out

4.2 การสร้างการเชื่อมต่อกับโอดีบีซี (ODBC Connection)

ในการที่จะเชื่อมต่อกับเอสคิวแอลเซิร์ฟเวอร์ จะต้องมีการสร้างโอดีบีซีดาต้าซอส (ODBC Datasource) ก่อน มีขั้นตอนดังนี้

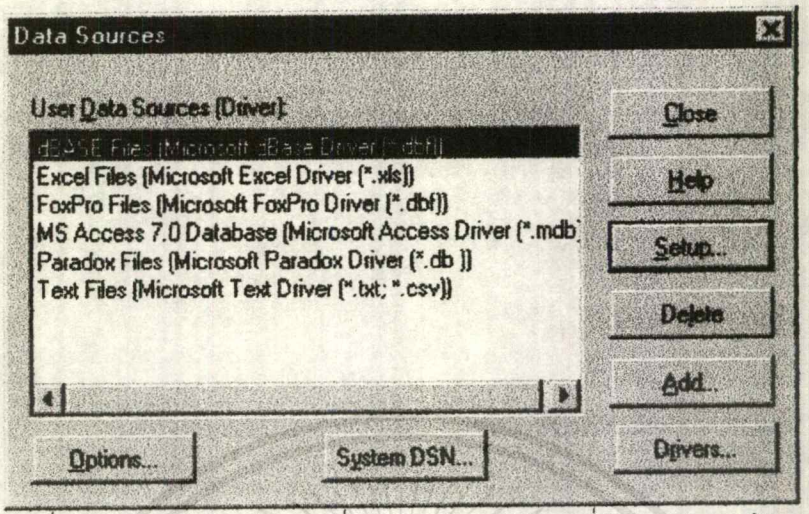
1. ทำการรันโอดีบีซี (32 bit ODBC) เพื่อทำการสร้างดาต้าซอส โดยข้อมูลเกี่ยวกับดาต้าซอสเก็บอยู่ในไฟล์ ODBC.INI

2. เลือกคำสั่ง ADD ใน Datasource Form



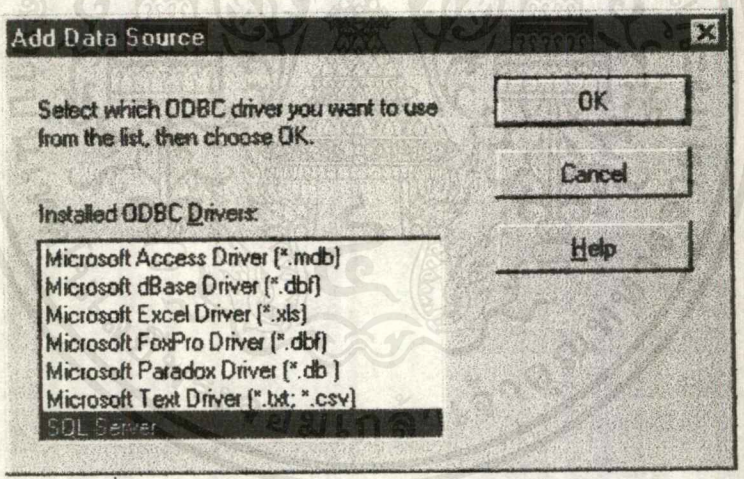
รูปที่ 4-3 แสดงการเลือกเพื่อทำสร้างการเชื่อมต่อโอดีบีซี โดยเรียก 32bit ODBC

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4-4 หน้าจอของดาต้าซอสเพื่อทำการเลือกดาต้าซอสที่ต้องการทำงานด้วย

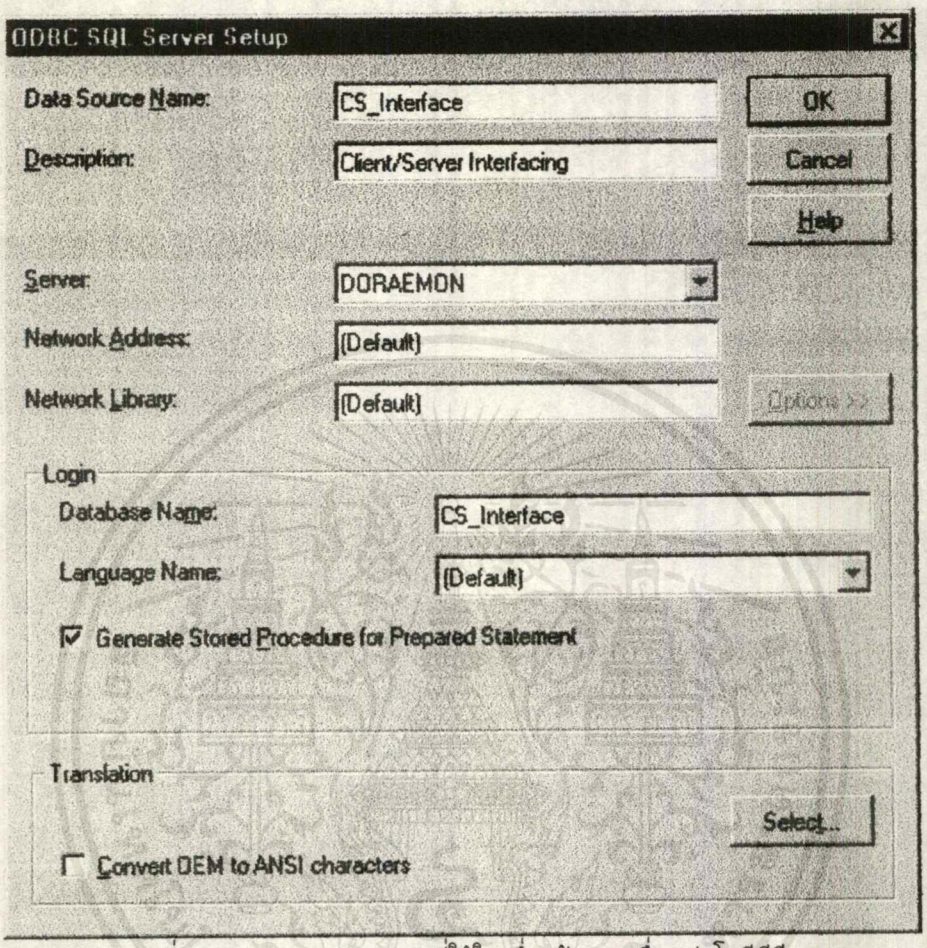
3. เลือกชนิดของดาต้าซอสที่จะทำการติดต่อด้วย ในฟอร์ม Add Data Source แล้วคลิ้ก OK



รูปที่ 4-5 แสดงการเลือกชนิดของดาต้าซอสเป็น SQL Server

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4. ทำการระบุรายละเอียด เกี่ยวกับชื่อดาต้าซอส (Datasource Name) ชื่อเซิร์ฟเวอร์ (Server Name) และชื่อดาต้าเบส (Database Name) ในฟอร์ม ODBC SQL Server Setup



รูปที่ 4-6 แสดงการกำหนดค่าที่ใช้ในเพื่อสร้างการเชื่อมต่อโอดีบีซี

หลังจากทำการกำหนดเสร็จเรียบร้อยแล้ว คลิ้ก OK ก็เป็นอันว่าทำการสร้างการเชื่อมต่อกับโอดีบีซีเสร็จเรียบร้อย

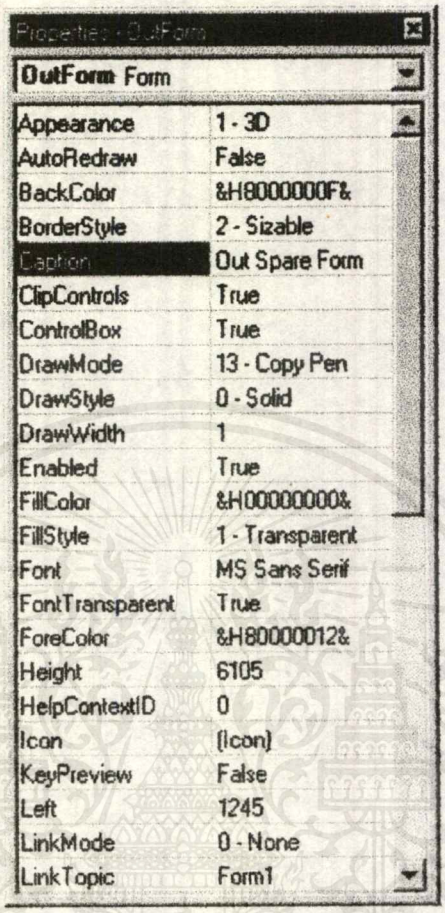
4.3 การทดลองสร้างแอพพลิเคชันตัวอย่างโดยใช้วิชวลเบสิค (Visual Basic)

ขั้นตอนและวิธีในการสร้างแอพพลิเคชันในกรณีศึกษานี้โดยวิชวลเบสิค

1. ทำการกำหนดการเชื่อมต่อกับระบบการจัดการฐานข้อมูล (SQL Server) โดยใช้โอดีบีซีไดร์ฟเวอร์ (ODBC Driver) (หมายเหตุ ดูวิธีจากการสร้างการเชื่อมต่อกับโอดีบีซี)

2. สร้างฟอร์มใหม่เพื่อสำหรับใช้งาน โดยกำหนดขนาด และตำแหน่งให้เหมาะสม และกำหนดค่าพร็อพเพอร์ตี้อื่นๆของฟอร์ม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



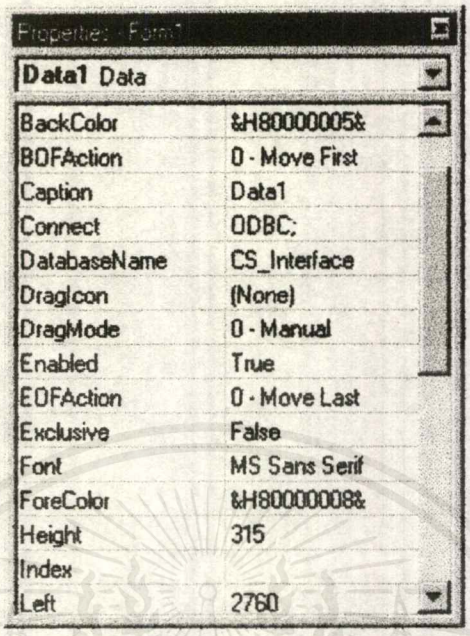
รูปที่ 4-7 แสดงพร็อพเพอร์ตี้ของฟอร์ม OutSpare

3. ในเหตุการณ์ FormLoad ให้ทำการกำหนด

```
Dim  MyDB as Database
Set  MyDB = OpenDatabase("CS_Interface",False,False,"ODBC")
```

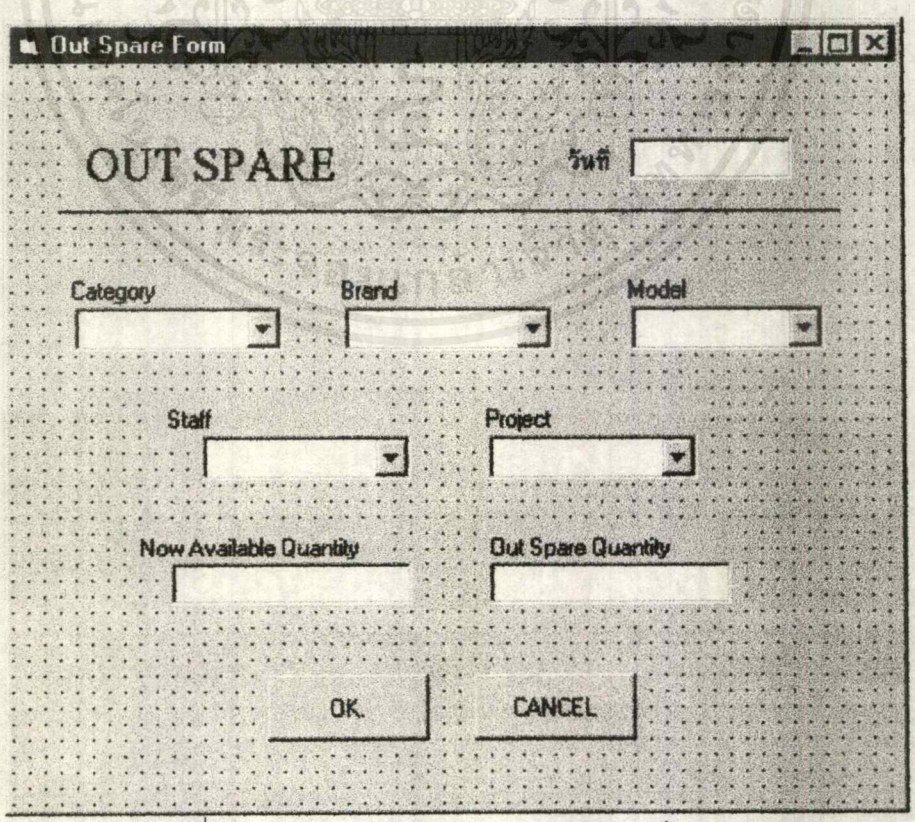
เพื่อเป็นการกำหนดการติดต่อกับฐานข้อมูลที่ใช้โอดีบีซีในการติดต่อ และมีชื่อว่า CS_Interface นอกจากวิธีนี้แล้วยังมีอีกวิธีที่ใช้ในการสร้างการติดต่อกับฐานข้อมูล โดยอาศัยคอนโทรลที่มีชื่อว่า Data หรืออีกตัวหนึ่งคือ RDO (มีเฉพาะใน Enterprise Edition) กำหนดค่าพร็อพเพอร์ตี้ DatabaseName ด้วยชื่อของฐานข้อมูลที่ต้องการใช้งาน RecordSource โดยเลือกชื่อตารางในฐานข้อมูลที่ต้องการทำงานด้วยจากลิสต์ และ Connect ซึ่งบอกถึงวิธีการเชื่อมต่อ ในที่นี้คือ ODBC

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4-8 แสดงการกำหนดพร็อพเพอร์ตี้ของดาต้าคอนโทรลเพื่อเชื่อมต่อกับฐานข้อมูล

4. กำหนดส่วนประกอบต่างๆที่ต้องการใช้งาน ในที่นี้ใช้ 5 ComboBox (Category, Brand, Model, StaffName และ ProjectName) กับ 3 TextBox (OutQty, AvailQty และ OutDate)



รูปที่ 4-9 แสดงการจัดส่วนประกอบต่างๆของฟอร์ม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5. การนำข้อมูลจากฐานข้อมูลมาแสดงแก่ผู้ใช้ นำมาจากตาราง Spare และเมื่อรับข้อมูลจากผู้ใช้แล้วจะนำไปเก็บในตาราง Out และมีบางส่วนที่ไปเปลี่ยนแปลงข้อมูลในตาราง Spare วิธีในการนำข้อมูลจากฐานข้อมูลมาแสดงมี 2 วิธีคือ

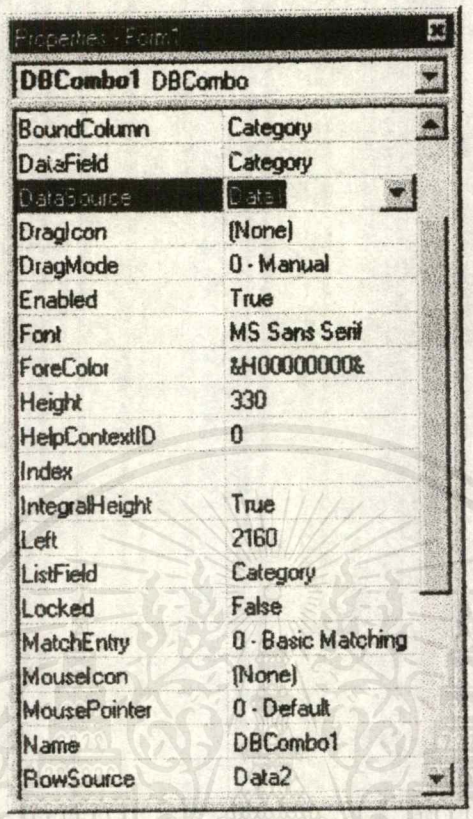
ใช้คอนโทรลที่ทำงานเกี่ยวกับข้อมูลเป็นตัวดึงข้อมูลมาแสดง โดยการกำหนดค่าพร็อพเพอร์ตี้ RowSource เป็นชื่อของคอนโทรล Data ที่สร้างในตอนแรกเพื่อสำหรับการอ้างอิงข้อมูล และ DataSource เป็นชื่อของคอนโทรล Data ที่จะนำไปทำงานด้วย ในกรณีที่มีการเรียกดูข้อมูลจากตารางหนึ่ง เพื่อใช้ทำงานกับอีกตารางหนึ่ง ต้องมีคอนโทรล Data ของแต่ละตารางของมันเอง นอกจากนี้ยังต้องกำหนดพร็อพเพอร์ตี้ ListField เพื่อบอกว่าต้องการดึงข้อมูลจากฟิลด์ใด จากตารางที่เลือกใน Data ของพร็อพเพอร์ตี้ RowSource และทำการกำหนดพร้อพเพอร์ตี้ DataField เป็นการบอกว่าต้องการทำงานกับข้อมูลบนฟิลด์ใดจากตารางที่เลือกใน Data ของพร็อพเพอร์ตี้ DataSource

อีกวิธีหนึ่งในการทำงานกับข้อมูล คือ โดยการเขียนโค้ดเพื่อทำการเรียกข้อมูลออกมาใช้งาน โดยผ่านเรคอร์ดเซต

```
Dim MyDB as Database
Dim MyDS as RecordSet
Set MyDB = OpenDatabase("CS_Interface",False,False,"ODBC")
Set MyDS = MyDB.OpenRecordset("SELECT DISTINCT Category FROM Spare,
dbopendynaset)
```

จากโค้ดตัวอย่างเป็นการเปิดเรคอร์ดเซตที่ทำการเก็บข้อมูลฟิลด์ Category จากตาราง Spare

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4-10 แสดงการกำหนดพร็อพเพอร์ตี้ของ DBCombo ผูกกับดาต้าคอนโทรล

6. ในกรณีที่ต้องทำการรับข้อมูลจากผู้ใช้ทีละขั้น จะต้องนำค่าที่เลือกไว้มาเก็บในตัวแปร เพื่อนำไปคิวรีครั้งต่อๆไป เช่นในกรณี Category ใช้ในการคิวรีหา Brand และ Category กับ Brand ใช้ในการคิวรีหา Model เป็นต้น

7. เมื่อทำการรับข้อมูลเกี่ยวกับอุปกรณ์จากการคลิ้กที่ ComboBox จากผู้ใช้แล้วจนครบ ต้องทำการคิวรีหาจำนวนอุปกรณ์นั้นที่คงเหลืออยู่ โดยเปิดเรคอร์ดเซตด้วยคำสั่งภาษาเอสคิวแอล ดังนี้ SELECT * FROM Spare WHERE Category= &CategoryValue AND Brand=&BrandValue AND Model=&ModelValue และจากนั้นทำการอ้างฟิลด์จากเรคอร์ดเซตโดยใช้ MyDS!Quantity แทนค่าในฟิลด์ Quantity ของตาราง Spare

8. สร้างปุ่มรับคำสั่ง OK และ CANCEL

9. ในเหตุการณ์การคลิ้ก OK ต้องทำการตรวจสอบว่ารับข้อมูลจากผู้ใช้ครบถ้วน และถูกต้องหรือไม่ ถ้าไม่สมบูรณ์ ต้องทำการแจ้งแก่ผู้ใช้ และจัดการกับข้อผิดพลาดที่เกิดขึ้น ในที่นี้คือให้ผู้ใช้ทำการระบุข้อมูลใหม่ แต่ถ้าถูกต้อง ก็จะทำการเก็บเรคอร์ดข้อมูลการเบิกอุปกรณ์ลงในตาราง Out และแก้ไขจำนวนอุปกรณ์คงเหลือในตาราง Spare โดยการทำ 2 คำสั่งนี้คือ Insert และ Update จะทำในรูปของทรานสแอคชัน เพื่อ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ป้องกันความผิดพลาดที่อาจเกิดขึ้นกับข้อมูล วิธีในการทำ Insert และ Update ทำได้ 2 วิธี คืออาศัยคำสั่งภาษาเอสคิวแอลโดยตรง หรืออาศัยการทำงานโดยเมธอดในเรคอร์ดเซต

การ Update ทำโดยภาษาเอสคิวแอลโดยใช้ UPDATE...SET.... และทำผ่านเรคอร์ดเซตโดยใช้ Edit และ UpdateMethod ทำได้ดังนี้

```
SqlStmt = "SELECT * FROM Spare WHERE Category=&CategoryValue -
          AND Brand=&BrandValue AND Model=&ModelValue"
Set   MyDS = MyDB.OpenRecordset (SqlStmt,dbOpenDyanaset)
MyDS.Edit
MyDS!Quantity = NewValue
MyDS.Update
```

ส่วนการ Insert ทำได้โดยคำสั่งภาษาเอสคิวแอล INSERT....INTO และ โดยผ่านเรคอร์ดเซตโดยใช้ AddNew และ Update Method ดังนี้

```
SqlStmt = "SELECT * FROM Out"
Set  MyDS = MyDB.OpenRecordSet(Sqlstmt,dbOpenDynaset)
MyDS.AddNew
MyDS!Category = CategoryValue
MyDS!Brand = BrandValue
   *** | ***     ทำการใส่ข้อมูลในแต่ละฟิลด์จนครบ
MyDS.Update
```

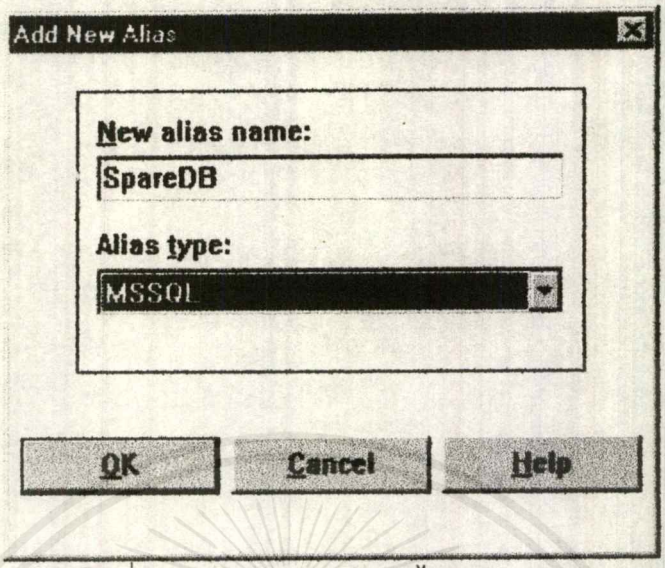
10. โดยการคลิ้ก CANCEL เป็นการจบแอพพลิเคชันในส่วนเบิกอุปกรณ์

4.4 การทดลองสร้างแอพพลิเคชันตัวอย่างโดยใช้เดลไฟล์ (Delphi)

ขั้นตอนและวิธีในการสร้างแอพพลิเคชันในกรณีศึกษานี้โดยเดลไฟล์

1. ทำการกำหนดค่าของอไลแอสให้เรียบร้อย ตามตัวอย่างจะใช้ฐานข้อมูลบนเอสคิวแอลเซิร์ฟเวอร์ โดยอาศัยเอสคิวแอลลิงค์ ซึ่งมีมากับ เดลไฟล์ไคลเอนท์เซิร์ฟเวอร์ 2.0 (Delphi Client/Server 2.0)

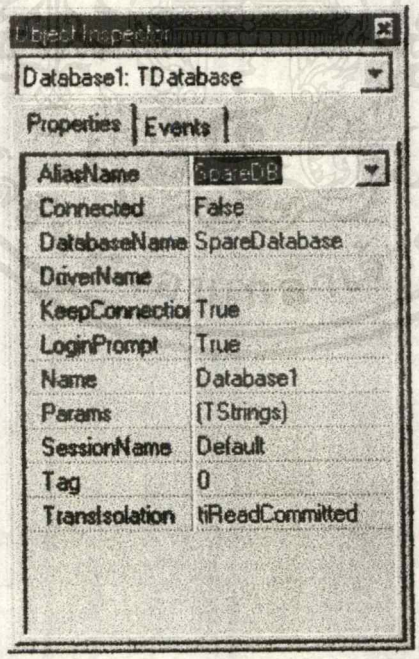
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4-11 แสดงการกำหนดค่าอไลแอส

2. ทำการสร้างฟอร์มใหม่ 1 ฟอร์ม

3. สร้าง Database Component ขึ้นใหม่ 1 ตัวและ ทำการกำหนดค่าให้มีความสัมพันธ์กับอไลแอสที่ได้สร้างไว้



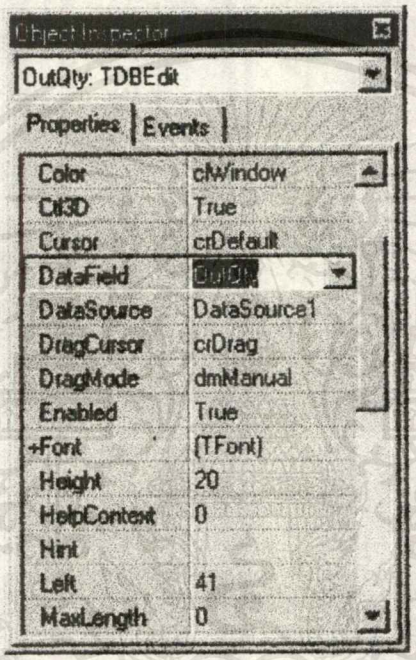
รูปที่ 4-12 แสดงการกำหนดค่าพร็อพเพอร์ตี้ของดาต้าเบสคอมโพเนนท์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

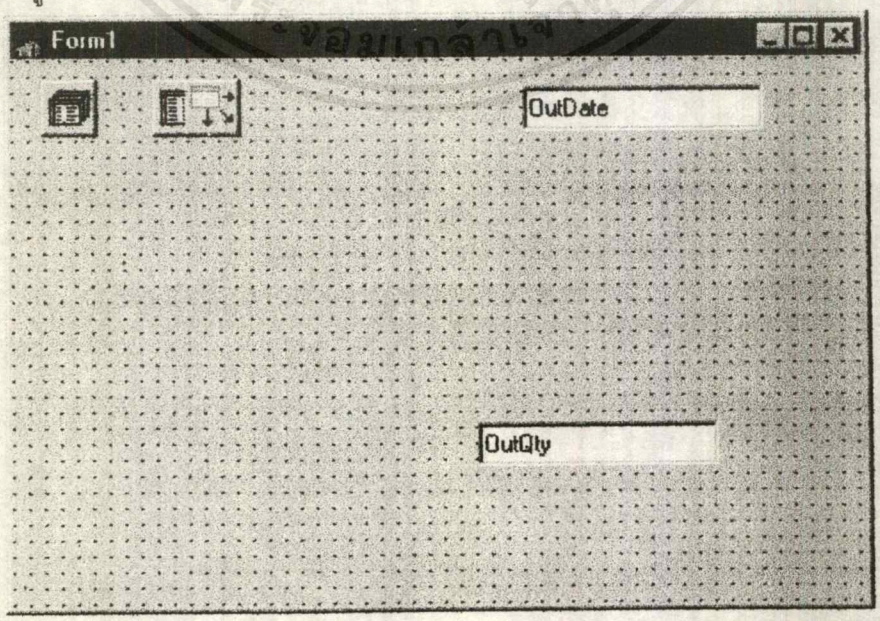
4. เนื่องจากข้อมูลหลักของแอพพลิเคชัน อยู่ในตาราง Out ดังนั้นให้สร้าง Table Component 1 ตัว ตั้งชื่อว่า out และทำการกำหนดค่า Database Name ให้สัมพันธ์กับ Database Component จากนั้นเลือกค่า TableName ให้เป็นตาราง Out และกำหนดค่า Active = True

5. สร้าง Datasource Component ขึ้นใหม่ 1 ตัว และกำหนดค่าให้สัมพันธ์กับ Table Component

6. สร้าง DBEdit Control เพื่อใช้กับฟิลด์ OutDate และ OutQty ซึ่งทำได้โดยตั้งค่า Datasource ให้สัมพันธ์กับตาราง Out และตั้งค่าฟิลด์ข้อมูลเป็น OutDate และ OutQty ตามลำดับ



รูปที่ 4-13 แสดงการกำหนดความสัมพันธ์ให้กับคอนโทรล ดีบีอีดิท (DBEdit)



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

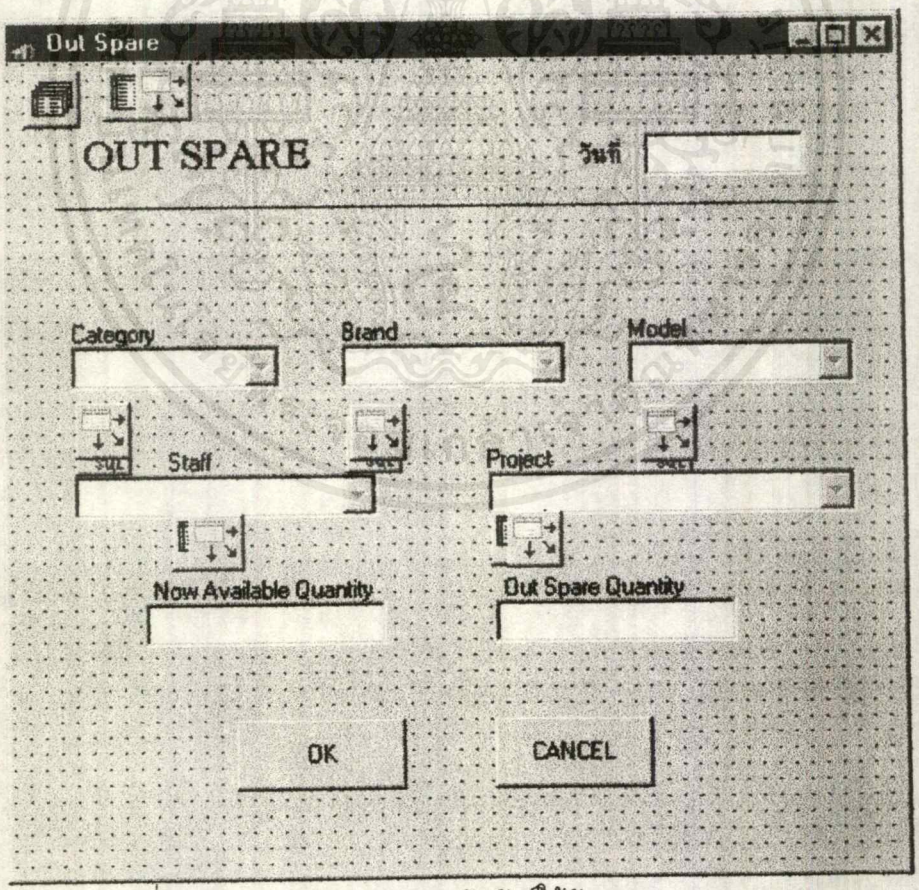
รูปที่ 4-14 แสดงชนิดของคอมโพเนนท์ที่สัมพันธ์กันบนฟอร์ม

7. สร้าง DBLookUp Combo ขึ้น 5 ตัวเพื่อใช้กับฟิลด์ Category, Brand, Model, StaffID และ ProjectID ตามลำดับ ในการที่จะให้ DBLookUp Combo สามารถดึงค่าจากตารางอื่นมาแสดงได้นั้น จำเป็นต้องมีการสร้าง Query Component ไว้ เพื่อเป็นแหล่งข้อมูลของแต่ละตัว ดังนั้นให้ทำการสร้าง Query Component และ Datasource 5 ชุด กำหนดให้สัมพันธ์กับ Combo Box ทั้ง 5 ตัว

พร็อพเพอร์ตี้ที่สำคัญของ TDBLookupComboBox

Datafield	กำหนดฟิล์ดข้อมูลซึ่งจะรับค่าจากคอมโบไปใช้
DataSource	ดาต้าซอร์สของตารางที่ต้องการรับค่าจากคอมโบไปใช้
ListSource	ดาต้าซอร์สของตารางที่คอมโบจะอ่านค่ามาแสดงให้ผู้ใช้เลือก
ListFIeld	กำหนดฟิล์ดข้อมูลซึ่งจะรับค่าจากตารางมาแสดงในคอมโบ

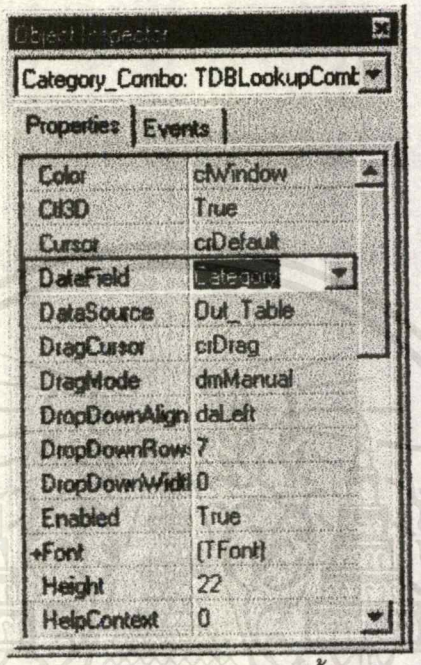
ตารางที่ 4-1 แสดงพร็อพเพอร์ตี้ที่สำคัญของ TDBLookUpCombo



รูปที่ 4-15 แสดงการกำหนดความสัมพันธ์ให้กับ DBLookUpCombo

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

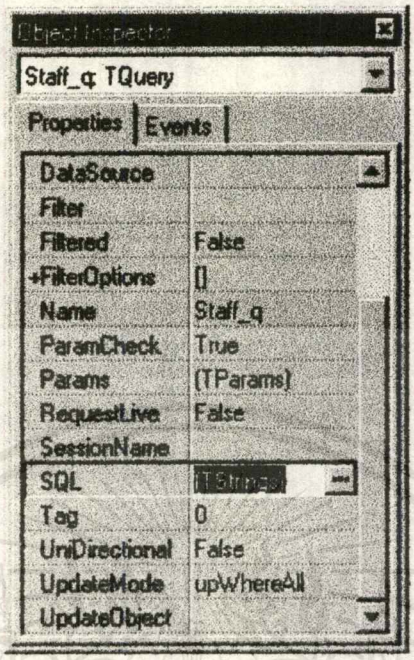
8. ในการที่จะให้ DBLookUpComboสามารถดึงค่าจากตารางอื่นมาแสดงได้นั้น จำเป็นต้องมีการสร้างคิวรีคอมโพเนนท์ไว้ เพื่อเป็นแหล่งข้อมูลของแต่ละตัว ดังนั้นให้ทำการสร้างคิวรีคอมโพเนนท์ และดาต้าซอส 5 ชุด กำหนดให้สัมพันธ์กับ Combo Box ทั้ง 5 ตัว



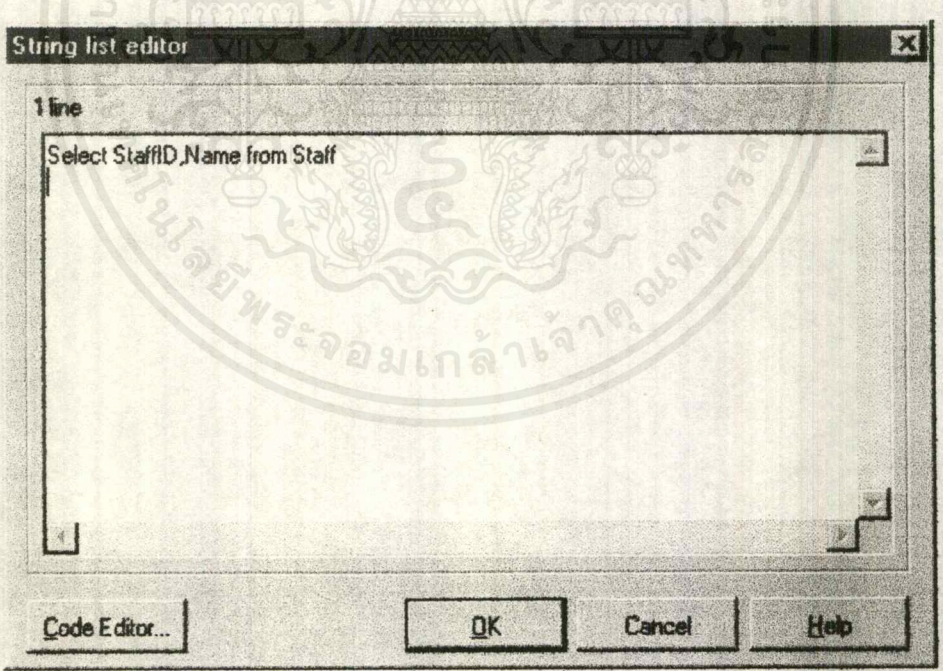
รูปที่ 4-16 แสดงการกำหนดพร็อพเพอร์ตี้ของคิวรีคอมโพเนนท์

9. ตั้งค่า SQL ของ Query Component ซึ่งสัมพันธ์กับฟิลด์ StaffID และ ProjectID ให้เป็นคำสั่งเอสคิวแอลที่จะได้ผลลัพธ์เป็นข้อมูลที่จะแสดงใน Combo Box ทั้ง 2 ตัว เช่น SELECT StaffID, Name FROM Staff สำหรับฟิลด์ StaffID จากนั้นทำการกำหนดค่า ListSource ของ ComboBox เป็นค่าของ Datasource ของคิวรีที่สร้างไว้ เพื่อเป็นการบอกให้ ComboBox ดึงข้อมูลจากคิวรีนั้นๆ และกำหนดค่า ListField ให้เป็นค่าของฟิลด์ที่ต้องการแสดงใน Combo Box และกำหนดค่า KeyField และ DataField ให้เป็นค่าของฟิลด์ที่ต้องการให้นำข้อมูลไปใส่ในที่นี้ คือ Field StaffID และ ProjectID ตามลำดับ จากนั้นทำการกำหนด Active = True

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4-17 แสดงการกำหนดเอสคิวแอลพร็อพเพอร์ตี้ของคิวรีคอมโพเนนท์

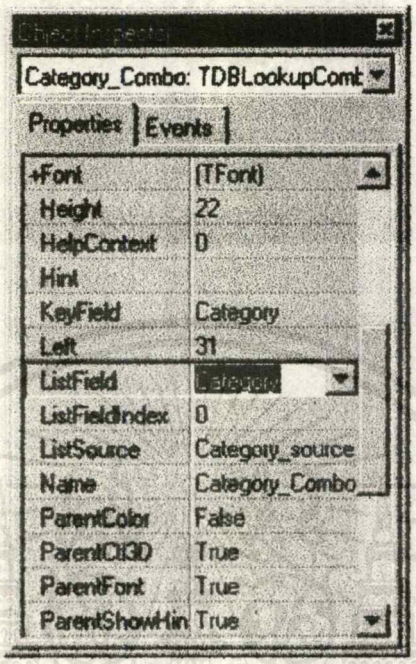


รูปที่ 4-18 แสดงการกำหนดคำสั่งเอสคิวแอลเพื่อใช้ในคิวรีคอมโพเนนท์

10. กำหนดค่า SQL ของ Query ซึ่งสัมพันธ์กับฟิลด์ Category ให้เป็น SELECT DISTINCT Category FROM Spare จากนั้นกำหนด Active = True และกำหนดค่าของ Category ComboBox ให้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สัมพันธ์กับคิวรีนี้ โดยตั้งค่า ListSource ให้เป็น Datasource ของคิวรี แล้วกำหนด List Field และ Data Field เป็นค่าของ Field Category ได้ตามต้องการ



รูปที่ 4-19 แสดงการกำหนดลิสต์ฟิลด์ และลิสต์ซอสใน LookupComboBox

11. สำหรับฟิลด์ Brand และ Model ให้ตั้งค่า List Field ให้สัมพันธ์กับ Datasource ของคิวรี แล้วกำหนดค่า SQL ของคิวรีดังนี้

ฟิลด์ Brand กำหนดให้ดึงข้อมูลฟิลด์ Brand ซึ่งมี Category ตามที่กำหนด โดยเว้นไว้เป็นตัวแปรเพื่อกำหนดค่าในภายหลัง ใช้คำสั่งภาษาเอสคิวแอล คือ SELECT Brand FROM Spare WHERE Category=:Category

ฟิลด์ Model กำหนดค่าเป็นตัวแปรเช่นกัน ทั้งในส่วนของค่า Brand และ Category ใช้คำสั่งภาษาเอสคิวแอล คือ SELECT Model FROM Spare WHERE Category=:Category AND Brand=:Brand

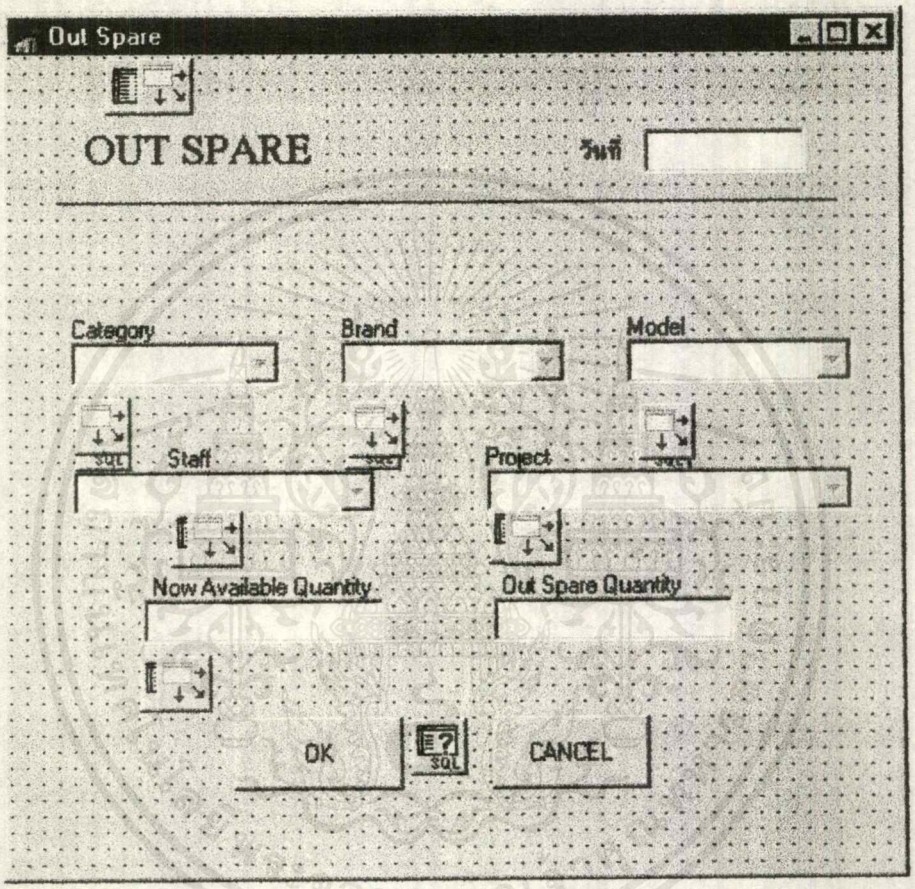
12. สร้างคิวรีและ DataSource อีก 1 ชุดเพื่อแสดงข้อมูลเกี่ยวกับจำนวนคงเหลือของอุปกรณ์ โดยกำหนดไว้เป็นตัวแปร เพื่อรับค่าที่ทำการคิวรีได้ จาก SELECT Quantity FROM Spare WHERE Category=:Category AND Brand=:Brand AND Model=:Model

13. สร้าง DBEdit หรือ DBText อีก 1 ตัวเพื่อใช้รับข้อมูลจากคิวรีที่แสดงค่าอุปกรณ์คงเหลือ

14. สร้างปุ่มคำสั่ง OK และ CANCEL

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

15. สร้างคิวรีอีก 1 ตัว เพื่อใช้ในการแก้ไขจำนวนอุปกรณ์คงเหลือ โดยใช้ตัวแปรเพื่อให้สามารถระบุค่าได้ เมื่อต้องการใช้ ดังนี้ UPDATE Spare SET Quantity=:Quantity WHERE Category=:Category AND Brand=:Brand AND Model=:Model AND OldQuantity=:OldQuantity

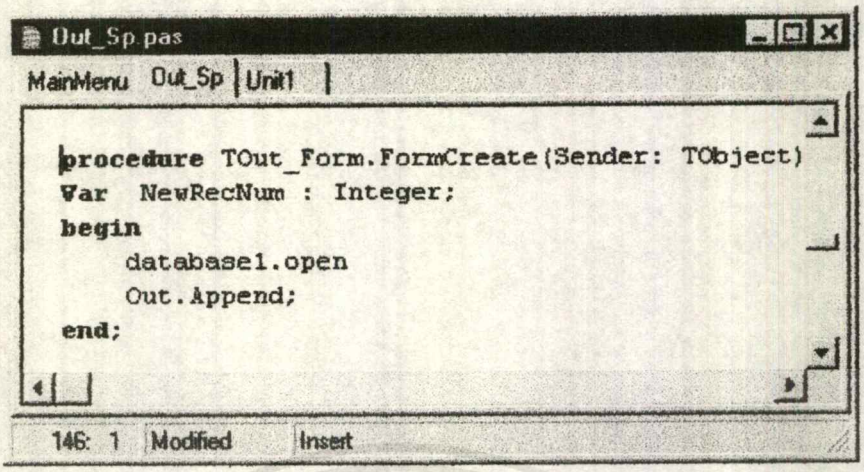


รูปที่ 4-20 แสดงหน้าจอการออกแบบโดยรวม

16. สร้างคำสั่งสำหรับเหตุการณ์ FormCreate ให้ทำการเปิดฐานข้อมูล และเพิ่มเรคอร์ดข้อมูลใหม่ลงในตาราง Out ดังนี้

```
Database1.Open
Out.Append
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4-21 แสดงชุดคำสั่งตอบสนองเหตุการณ์ FormCreate

17. สำหรับคำสั่งส่วนที่ทำการจัดการ ComboBox 3 ตัว (Category, Brand, Model) ให้ทำงานสัมพันธ์กันนั้น จะต้องทำงานโดยการตรวจสอบทุกครั้งที่มีการเปิด และปิด ComboBox (OnCloseUp Event) ว่าค่าใน ComboBox แต่ละตัวมีค่าเป็นอย่างไร โดยดูเริ่มจาก Category ถ้าไม่เป็นค่าว่างก็จะทำการส่งตัวแปรให้กับคิวรีของฟิลด์ Brand เพื่อทำการดึงข้อมูลฟิลด์ Brand มาแสดงใน ComboBox และในฟิลด์ Model ก็มีลักษณะเดียวกัน คือพิจารณาทั้งค่า Category และ Brand จากนั้นก็ทำการคิวรีหาจำนวนของอุปกรณ์ที่เหลืออยู่จากตาราง Spare

```
Procedure TOut_Form.Update_Combo;
Begin
  If Category_Combo.Text <> '' Then
  Begin
    Model_q.Active := false;        {ปิดคิวรีที่ใช้แสดงค่า Brand และ Model}
    Brand_q.Active:= false;
                                    {ส่งค่า Parameter ไปให้คิวรีที่ใช้แสดงค่า Brand }
    Brand_q.ParamByName('Category').Asstring := Category_Combo.Text;
    Brand_Combo.Listfield := 'Brand';
    Brand_Combo.Keyfield := 'Brand';
    Brand_q.Active:=True;           {สั่งให้ทำงาน}
  End
  Else
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
 Begin
   Brand_q.Active:=false;
   Model_q.Active:=false;
 End;

 If Brand_Combo.Text <> '' Then
 Begin
   Model_q.Active := false;
                                {ส่งค่า Parameter ไปให้คิวรีที่ใช้แสดงค่า Model }
   Model_q.ParamByName('Category').Asstring := Category_Combo.Text;
   Model_q.ParamByName('Brand').Asstring := Brand_Combo.Text;
   Model_Combo.Listfield := 'Model';
   Model_Combo.Keyfield := 'Model';
   Model_q.Active:=True;

 End
 Else
  Begin
    Model_q.Active:=false;
  End;

 Update_Avail_qty;      {ปรับค่าของอุปกรณ์ที่เหลือ }
End;


Procedure TOut_Form.Update_Avail_Qty;
Begin
                          {ตรวจว่าค่าที่ต้องการใช้ใน Query มีค่าครบ}
If (Category_Combo.Text <> '') AND
     (Brand_Combo.Text <> '') AND
     (Model_Combo.Text <> '') Then
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Begin
  Avail_q.Active := false;
                                {ส่งค่า Parameter ไปให้คิวรีที่ใช้ปรับค่าของอุปกรณ์ที่เหลือ}
  Avail_q.ParamByName('Category').Asstring := Category_Combo.Text;
  Avail_q.ParamByName('Brand').Asstring := Brand_Combo.Text;
  Avail_q.ParamByName('Model').Asstring := Model_Combo.Text;
  Avail_q.Active := True;
  Avail_Qty.Datafield := 'Quantity';     {สั่งให้แสดงค่า}
 End
 Else
  Begin
   Avail_q.Active := false;
  End;
End;
```

18. สร้างคำสั่งส่วนที่ตอบสนองการคลิ๊ก OK โดยทำการตรวจสอบเงื่อนไข ความครบถ้วน ถูกต้องของข้อมูล ถ้าถูกต้องจะทำการเริ่มต้นทรานสแอคชั่น แล้วทำการส่งค่าตัวแปรไปยังคิวรีที่เตรียมไว้สำหรับการแก้ไขจำนวนอุปกรณ์คงเหลือ จากนั้นทำการตรวจสอบว่าการทำงานสำเร็จหรือไม่ ถ้าสำเร็จก็จะทำการเพิ่มเรคอร์ดข้อมูลการเบิกอุปกรณ์ลงในตาราง Out โดยเพียงแค่ใช้คำสั่ง Post ของ Table Component ที่จะทำการเก็บข้อมูลจากหน้าจอไว้ในฐานข้อมูล และในส่วนนี้ต้องตรวจสอบความผิดพลาดอยู่ตลอดเวลา ถ้าหากมีความผิดพลาดเกิดขึ้น ให้ทำการ RollBack เพื่อยกเลิกการเปลี่ยนแปลง แต่ถ้าไม่มี ให้ทำการ Commit เพื่อยืนยันการเปลี่ยนแปลง และจบทรานสแอคชั่น

```
procedure TOut_Form.Button1Click(Sender: TObject);
Var Button : Integer;
    NewRecNum : Integer;
    Result : Boolean;
    e      : TDBError;

begin
  Result := false;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                    {ทำการตรวจสอบเงื่อนไข ความครบถ้วน ถูกต้องของข้อมูล}
If Out_Qty.Text <> '' Then
  If StrToInt(Out_Qty.Text) <= 0 then
  Begin
    Application.MessageBox('Out spare quantity not valid','ERROR',mb_OK);
    Exit;
  End;

IF (Date_Now.Text <> '') AND
   (Category_Combo.Text <> '') AND
   (Brand_Combo.Text <> '') AND
   (Model_Combo.Text <> '') AND
   (Staff_Combo.Text <> '') AND
   (Project_Combo.Text <> '') AND
   (Out_Qty.Text <> '') Then
 Begin
   If ( StrToInt(Out_Qty.Text) <= StrToInt(Avail_Qty.Text)) Then
    Begin          {ข้อมูลครบถ้วนถูกต้อง}
                   {ส่งค่า Parameter เพื่อปรับค่าในตาราง Spare}
       Update_Qty.ParamByName('Quantity').AsInteger :=
       StrToInt(Avail_Qty.Text)- StrToInt(Out_Qty.Text);
       Update_Qty.ParamByName('Category').Asstring :=
Category_Combo.Text;
       Update_Qty.ParamByName('Brand').Asstring := Brand_Combo.Text;
       Update_Qty.ParamByName('Model').Asstring := Model_Combo.Text;
        { for sure that no one modify quantity that read before }
        { then we must check that current quantity = old quantity }
       Update_Qty.ParamByName('Old_Quantity').AsInteger :=
       StrToInt(Avail_Qty.Text);

      Database1.StartTransaction;          {เริ่มต้น Transaction}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
try
 Update_Qty.ExecSQL;          {แก้ไขข้อมูล}
 If Update_Qty.RowsAffected <= 0 Then
 Begin
          {แก้ไขไม่สำเร็จ }
      Application.MessageBox('Error update quantity (available quantity
      have been change) try again!','ERROR',mb_OK);
   Update_Avail_Qty;        {ปรับค่าอุปกรณ์คงเหลือ}
   Database1.Rollback
 end
 Else          { การแก้ไขตาราง Spare ทำสำเร็จ}
 Begin
  Out.Post;              {ใส่ข้อมูลลงในตาราง Out }
  Database1.Commit;
  Result:=True;
 End;

except
on e:eDBEngineError Do
 Begin
  Messagedlg('ERROR : '+e.message,mtError,[mbOK],0);
  form4.Database1.Rollback  ;
 End;
End;


If Result = true Then
Begin                    {การแก้ไขข้อมูลทำได้เรียบร้อย}
 Out.Appen;              {เริ่มรับข้อมูลรายการต่อไป }
 Update_Avail_Qty;
End
Else              {การแก้ไขข้อมูลทำไม่สำเร็จ}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        Update_Avail_Qty;
      End
     Else
      Begin
        Button:=Application.MessageBox('Do not have enough
spare!','ERROR',mb_OK);
      End;
   End
   Else
     Button:=Application.MessageBox('Data entry not complete','ERROR',mb_OK);
end;
```

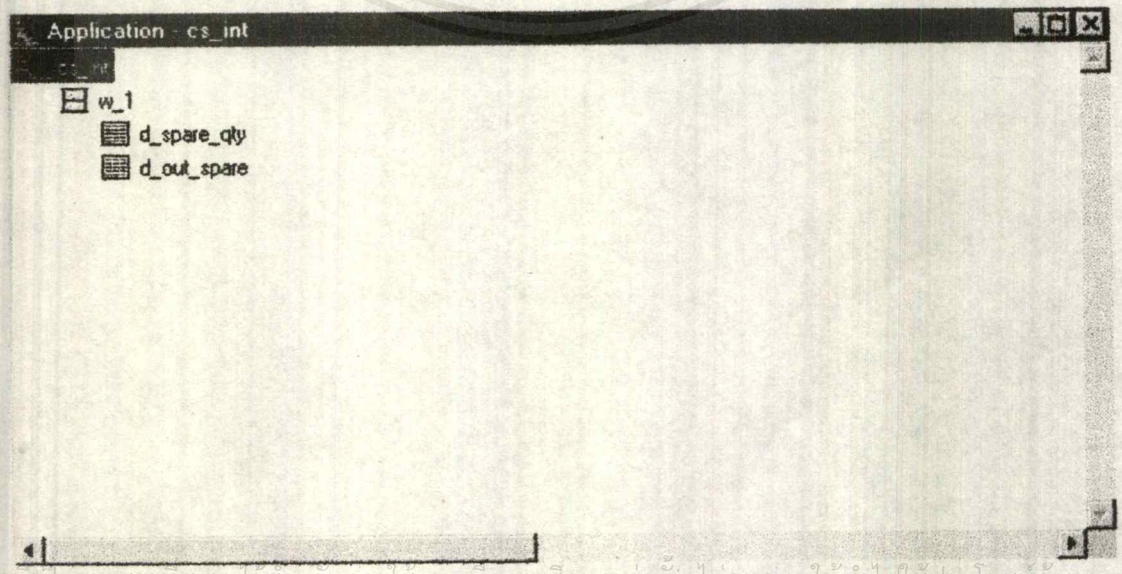
19. คำสั่งในส่วนการตอบสนองการคลิ้ก CANCEL ให้จบการทำแอพพลิเคชันส่วนการเบิกอุปกรณ์

4.5 การทดลองสร้างแอพพลิเคชันตัวอย่างโดยใช้เพาเวอร์บิวเดอร์ (Power Builder)

ขั้นตอนและวิธีในการสร้างแอพพลิเคชันในกรณีศึกษานี้โดยเพาเวอร์บิวเดอร์

1. ทำการติดตั้งดาต้าซอส และกำหนดฐานข้อมูลที่ต้องการติดต่อเพื่อใช้งานให้เรียบร้อย ในที่นี้ใช้เอสคิวแอลเซิร์ฟเวอร์ (SQL Server) โดยทำการกำหนดการเชื่อมต่อผ่านโอดีบีซีดาต้าซอส (ODBC Datasource)

2. ทำการสร้างแอพพลิเคชันใหม่ โดยใช้แอพพลิเคชันเพนท์เตอร์ (Application Painter) และตั้งชื่อว่า CS_Int



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 4-22 แสดงโครงสร้างของแอพพลิเคชัน

3. ทำการสร้างวินส์โดว์ใหม่ 1 ตัว ชื่อ w_1

4. สร้างดาต้าวินส์โดว์ออบเจคต์ตัวแรก โดยกำหนดให้ทำการรับข้อมูลจากตาราง Out ทั้งหมด และนำทุกฟิลด์มาแสดง และทำการเลือกรูปแบบเป็น Free Form และตั้งชื่อเป็น d_out_spare และทำการจัดตำแหน่งของฟิลด์ตามต้องการ

5. เนื่องจากจำเป็นต้องอาศัยข้อมูลที่แสดงจำนวนอุปกรณ์คงเหลือ จากตาราง Spare จึงต้องทำการสร้างดาต้าวินส์โดว์ออบเจคต์อีกตัวหนึ่ง โดยให้รับข้อมูลจากตาราง Spare เฉพาะฟิลด์ Qty จากนั้นเพื่อให้สามารถเชื่อมโยงข้อมูลให้สัมพันธ์กับอีกตารางหนึ่งได้ จึงต้องกำหนด Retrieve argument (หมายถึง ตัวแปรที่สามารถระบุค่าได้ระหว่างการสั่งดึงข้อมูลจากฐานข้อมูล) 3 ตัว คือ Category, Brand และ Model ซึ่งมีชนิดเป็น String แล้วนำตัวแปรทั้ง 3 ตัวไปทำการกำหนดเงื่อนไขการเลือกข้อมูล (Where) โดยคำสั่งภาษาเอสคิวแอลที่ใช้ มีลักษณะดังนี้

```
SELECT  Quantity  FROM Spare
WHERE  Category = :Category AND Brand = :Brand AND
          Model = :Model
```

และทำการตั้งชื่อดาต้าวินส์โดว์ออบเจคต์เป็น d_avail_qty

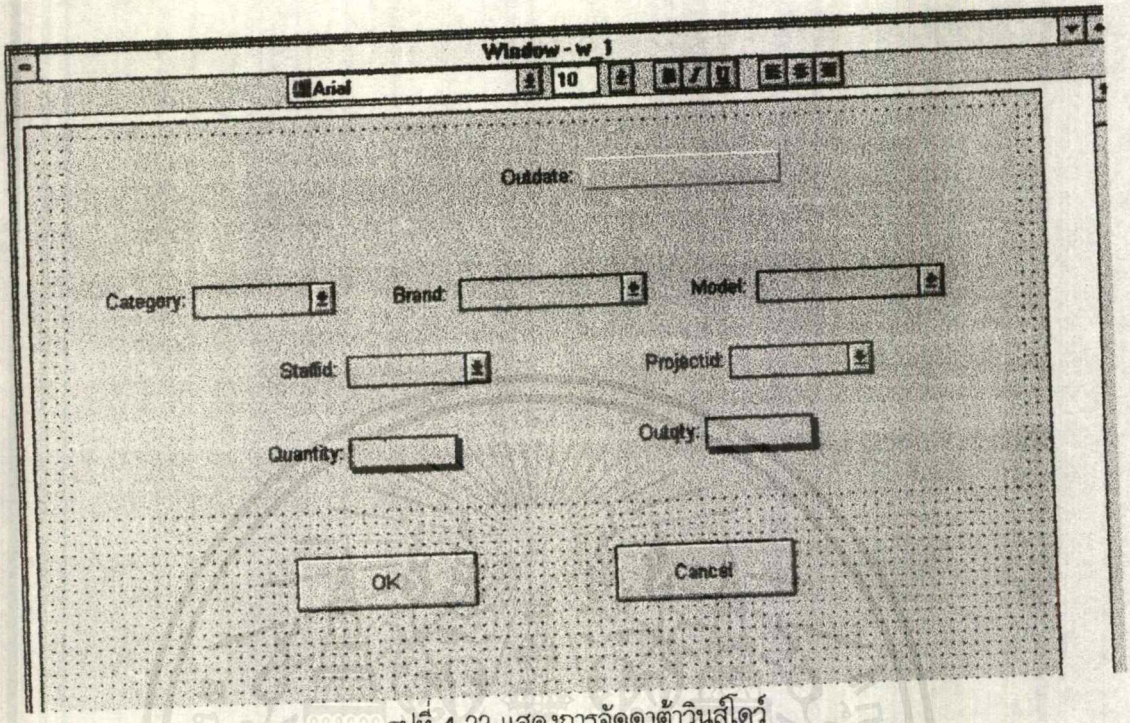
6. เนื่องจากต้องมีส่วนควบคุม สำหรับใช้ในการเลือกค่า (Drop Down List box Control) ซึ่งสามารถทำได้โดยสร้างเป็นดาต้าวินส์โดว์ออบเจคต์แยกต่างหาก 5 ตัว คือ d_category_combo, d_brand_combo , d_model_combo , d_staff_combo และ d_project_combo ซึ่งจะได้ค่ามาจากฟิลด์ Category, Brand , Model ของตาราง Spare StaffID และ ProjectID ของตาราง Staff และ Project ตามลำดับ โดยมีจุดพิเศษที่ต้องกำหนดให้ค่าไม่ซ้ำกัน โดยใส่ตัวเลือก Distinct ไปด้วย ทั้งนี้เพราะมีโอกาสเป็นไปได้ที่ในแต่ละฟิลด์จะมีค่าซ้ำกันได้

7. ที่ดาต้าวินส์โดว์ออบเจคต์ d_out_spare ให้ทำการ Edit Style ของคอลัมน์ที่ต้องการให้ทำการเลือกค่าได้ (Category, Brand, Model,StaffID,ProjectID) ให้เป็น Drop Down DataWindow แล้วกำหนดให้ไปใช้ดาต้าวินส์โดว์ออบเจคต์ที่สร้างไว้แล้วคือ d_category_combo, d_brand_combo , d_model_combo , d_staff_combo และ d_project_combo ตามลำดับ

8. ที่วินส์โดว์ w_1 ให้ทำการสร้างดาต้าวินส์โดว์คอนโทรล 2 ตัวชื่อ dw_1 และ dw_2 ตามลำดับ โดย dw_1 กำหนดให้ใช้ดาต้าวินส์โดว์ออบเจคต์ d_out_spare และ dw_2 กำหนดให้ใช้ดาต้าวินส์โดว์ d_avail_qty

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

9. ทำการจัดตำแหน่งของ dw_1 และ dw_2 ให้เหมาะสมตามต้องการ



รูปที่ 4-23 แสดงการจัดดาต้าวินส์โดว์

10. สร้างสคริปท์ของเหตุการณ์เปิด (Open Event) ของแอพพลิเคชั่น ให้ทำการติดต่อกับฐานข้อมูล และเปิดวินส์โดว์ w_1 ซึ่งเป็นวินส์โดว์หลัก

```
sqlca.dbms = "ODBC"
Sqlca.Database = "CS_Int"
sqlca.UserID = "s6014031"
sqlca.DBParm =
"ConnectString='DSN=CS_Int;UID=S6014031;PWD=interface;APP=PowerBuil
de       r - cs_int;WSID=SUNEO;DATABASE=CS_Interface"
sqlca.autocommit = false

CONNECT;
If SQLca.SQLCode <> 0 then
   MessageBox("Error on connect to database", &
                  "Error Code: " + String(sqlca.SqlDbcode) + &
                  "~nError Message= " + sqlca.SqlErrText, &
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                StopSign!)
        HALT
        End if
        Open(w_1)
```

11. สร้างสคริปท์เพื่อตอบสนองเหตุการณ์ปิด (Close Event) ให้ทำการยกเลิกการเชื่อมต่อกับฐานข้อมูลโดยใช้คำสั่ง Disconnect ;

12. สร้างสคริปท์สำหรับเหตุการณ์เปิดของ w_1 ให้ทำการกำหนดทรานสแอคชันออบเจคต์ของดาต้าวินส์โดว์ทั้ง 2 ตัว จากนั้นทำการเพิ่มแถวว่างๆใน dw_1 และกำหนดให้แสดงข้อมูลของแถวนี้ (คือค่าว่างในทุกๆฟิลด์) เพื่อการป้อนข้อมูล

```
        long n
        dw_1.SetTransObject(sqlca)
        n = dw_1.insertrow(0)
        dw_1.scrolltorow(n)
        dw_1.setfocus()
        dw_2.setTransObject(Sqlca)
```

13. ต่อไปทำการกำหนดการทำงานของ Drop Down DataWindow ทั้ง 3 ตัว (category , brand , model)ให้ทำการเปลี่ยนแปลงชอบเขตของการเลือกตัวถัดไป ซึ่งการทำงานนี้ต้องสร้างในรูปของคำสั่งที่ตอบสนองเหตุการณ์ ItemChanged ของ dw_1 ซึ่งจะเกิดทุกครั้งที่มีการเปลี่ยนแปลงค่าของคอลัมน์ใน dw_1 ดังนั้นในสคริปท์จะต้องมีการตรวจสอบคอลัมน์ที่เปลี่ยนแปลง ว่าเป็นคอลัมน์ใด โดยใช้คำสั่ง GetColumnName() ซึ่งจะได้ชื่อของคอลัมน์ที่มีการเปลี่ยนแปลง จากนั้นจะนำไปตรวจสอบว่าเป็นคอลัมน์ใดใน 3 คอลัมน์ที่ต้องการ ถ้าเป็นคอลัมน์อื่นจะไม่สนใจ

```
datawindowchild State_category_combo, State_brand_combo, State_model_combo
string ls_sqltext , ls_columnname , ls_categoryvalue , ls_brandvalue , ls_modelvalue
        long ll_current_row
ls_columnname = this.GetColumnName()

choose case ls_columnname
CASE "category"
        dw_1.Getchild('brand',State_brand_combo)
        dw_1.Getchild('model',State_model_combo)
        state_brand_combo.setitem(ll_current_row,"brand","")
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        state_model_combo.setitem(ll_current_row,"model","")
        ll_current_row = Getrow()
        ls_categoryvalue = Gettext()
        ls_sqltext = "Select brand from spare where category = '" +
        ls_categoryvalue + "'"
        State_brand_combo.setsqlselect(ls_sqltext)
        State_brand_combo.settransobject(sqlca)
        State_brand_combo.Retrieve()

        /* clear value of listing in model combo */
        State_model_combo.setsqlselect("select model from spare where category =
        '")
        State_model_combo.settransobject(sqlca)
        State_model_combo.Retrieve()
CASE "brand"
        dw_1.Getchild('Model',State_Model_combo)
        state_model_combo.setitem(ll_current_row,"model","")

        ll_current_row = Getrow()
        ls_categoryvalue = getitemstring(ll_current_row,"category")
        ls_brandvalue = Gettext()
        ls_sqltext = "Select model from spare where category = '" +
        ls_categoryvalue + "'" + " AND brand = '" + ls_brandvalue + "'"
        State_model_combo.setsqlselect(ls_sqltext)
        State_model_combo.settransobject(sqlca)
        State_model_combo.Retrieve()

CASE "model"
        ll_current_row = Getrow()
        ls_categoryvalue = getitemstring(ll_current_row,"category")
        ls_brandvalue = getitemstring(ll_current_row,"brand")
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
ls_modelvalue = gettext()

dw_2.retrieve(ls_categoryvalue,ls_brandvalue,ls_modelvalue)
END CHOOSE
```

14. การเปลี่ยนแปลงขอบเขตของการเลือก Drop Down DataWindow สามารถทำได้โดย เรียกคำสั่ง GetChild() เพื่อกำหนดตัวแปรอ้างอิงไปยังดาต้าวินส์โดว์นั้นๆก่อน จากนั้นกำหนดขอบเขตการเลือกโดยกำหนดคำสั่งเอสคิวแอลที่ใช้เลือกโดยตรง โดยใช้คำสั่ง SetSQLText ซึ่งคำสั่งเอสคิวแอลคำสั่งนั้น จะเป็นคำสั่งที่ใช้ค่าของ Drop Down DataWindow ตัวอื่นเป็นตัวจำกัด โดยค่าที่เป็นค่าปัจจุบันของข้อมูลในคอลัมน์นั้นจะหาได้จากคำสั่ง GetText() ซึ่งจะใช้ค่าของข้อมูลในคอลัมน์ที่ถูกเลือกอยู่ กรณีที่ต้องการข้อมูลของคอลัมน์ที่ไม่ได้ถูกเลือกอยู่ จะต้องใช้คำสั่ง GetItemString ซึ่งจะต้องบอกค่าของแถวปัจจุบัน สามารถหาได้จาก GetRow()

15. เมื่อทำการเปลี่ยนขอบเขตการเลือกแล้ว ต้องสั่งให้ Drop Down DataWindow นั้น ทำการอ่านข้อมูลด้วยคำสั่ง Retrieve()

16. กรณี Model นั้น เมื่อผู้ใช้ทำการเลือกรุ่นเรียบร้อยแล้ว ก็จะต้องแสดงค่าของจำนวนอุปกรณ์ที่มีอยู่ โดยผ่านค่าของ Retrieval Argument ทั้ง 3 ตัวไปให้แก่ dw_2 เพื่อให้แสดงข้อมูลที่ต้องการ

17. เป็นไปได้ในกรณีที่ผู้ใช้เลือกค่าของ Drop Down DataWindow ตัวใดตัวหนึ่งแล้ว ค่านั้นเป็นค่าเดิม (ทำการเลือกใหม่แล้ว แต่ยังคงเลือกอันเดิม) ทำให้เหตุการณ์ ItemChanged ไม่เกิดขึ้น ซึ่งจะทำให้การทำงานผิดพลาด ดังนั้นจะต้องทำการสร้างคำสั่งในเหตุการณ์คลิ้ก ซึ่งจะเกิดทุกครั้งที่กด dw นี้ ให้ทำการบังคับให้เกิดเหตุการณ์ ItemChanged ทุกครั้งที่มีการกดที่ Drop Down DataWindow ตัวใดตัวหนึ่ง

```
datawindowchild State_category_combo,
State_brand_combo, State_model_combo
string ls_sqltext , ls_columnname , ls_categoryvalue , ls_brandvalue,
ls_modelvalue
long ll_current_row , li_columnNum
li_columnNum = this.Getclickedcolumn()
ls_columnname = This.Describe("#" + String(li_columnnum) + ".Name")

choose case ls_columnname
CASE "category"
    dw_1.triggerEvent(Itemchanged!)
CASE "brand"
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        dw_1.triggerEvent(Itemchanged!)
    CASE "model"
        dw_1.triggerEvent(Itemchanged!)
    END CHOOSE
```

18. ที่วินส์โดว์ w_1 ทำการเพิ่มปุ่ม 2 ปุ่ม คือ OK และ CANCEL จากนั้นกำหนดสคริปท์ให้แต่ละตัว และสำหรับปุ่ม CANCEL ให้ทำการปิดแอพพลิเคชันนี้ โดยใช้คำสั่ง HALT CLOSE

19. สำหรับปุ่ม OK จะต้องทำการสร้างสคริปท์ เพื่อให้ทำการตรวจสอบความครบถ้วน และถูกต้องของข้อมูล ซึ่งทำได้โดยการรับค่าของคอลัมน์ต่างๆมา โดยใช้คำสั่ง GetItemString, GetItemDateTime, GetItemNumber ตามแต่ชนิดข้อมูลในคอลัมน์นั้น โดยก่อนทำการดึงข้อมูลจะต้องเรียกคำสั่ง AcceptText() ก่อนเพื่อให้ค่าที่ผู้ใช้ป้อนไว้ถูกเก็บลงไปยังคอลัมน์นั้น

20. ทำการตรวจสอบค่าของคอลัมน์ต่างๆ ตามเงื่อนไข ถ้ามีความผิดพลาดก็ทำการแจ้งให้ผู้ใช้ได้ทราบ

21. ทำการแก้ไขข้อมูลในตาราง Spare โดยใช้คำสั่งภาษาเอสคิวแอลโดยตรง ตามค่าที่ได้จากการคำนวณ

22. ทำการเพิ่มเรคอร์ดข้อมูลใหม่ลงในตาราง Out โดยใช้คำสั่ง Update() ซึ่งจะทำการเก็บค่าที่มีอยู่ในแต่ละคอลัมน์ ลงในฐานข้อมูลตามที่กำหนดไว้

23. ถ้าการทำงานทั้ง 2 ขั้นตอน ประสบความสำเร็จให้ใช้คำสั่ง COMMIT และถ้าหากล้มเหลวตัวใดตัวหนึ่ง ให้ใช้คำสั่ง RollBack

```
/* this Button will update out table and check weather
        - data entry complete
        - out quantity is ok ( out quantity <> 0 and out quantity <= spare
quantity)  */

integer li_resultcode
string ls_category , ls_brand , ls_model , ls_staffid , ls_projectid
integer li_outqty , li_availqty
long ll_current_row , n
datetime ld_outdate
integer li_remain_qty

ll_current_row = dw_1.Getrow()
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/* get value of all field to check weather it is complete or not */

dw_1.AcceptText()
ls_category = dw_1.getitemstring(ll_current_row,"category",PRIMARY!,false)
ls_brand = dw_1.getitemstring(ll_current_row,"brand",PRIMARY!,false)
ls_model = dw_1.getitemstring(ll_current_row,"model",PRIMARY!,false)
ls_staffid = dw_1.getitemstring(ll_current_row,"staffid",PRIMARY!,false)
ls_projectid = dw_1.getitemstring(ll_current_row,"projectid",PRIMARY!,false)
ld_outdate =
dw_1.getitemdatetime(ll_current_row,"outdate",PRIMARY!,false)
li_outqty = dw_1.getitemnumber(ll_current_row,"outqty",PRIMARY!,false)

ll_current_row = dw_2.Getrow()
if ll_current_row > 0  Then
li_availqty = dw_2.getitemnumber(ll_current_row,"quantity",PRIMARY!,false)
Else
        SetNull(li_availqty)
End if

/* check all of they is not empty and have enough spare to out */
if ( (ls_category <> "") AND (ls_brand <> "") AND (ls_model <> "") AND &
(ls_staffid <> "") AND (ls_projectid <> "") AND (NOT isnull(ld_outdate)) &
AND (NOT isnull(li_outqty)) AND (li_outqty <> 0) AND (li_outqty <= &
li_availqty) ) then

li_remain_qty = (li_availqty - li_outqty)

UPDATE spare SET quantity = :li_remain_qty where category =
:ls_category AND brand = :ls_brand AND model = :ls_model AND quantity
= :li_availqty;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
if sqlca.Sqlcode <> 0 THEN
        Messagebox("Error on update",                    &
                "Error Code : " + String(Sqlca.SqlDbcode) +  &
                "~nError Message= " + sqlca.SqlErrText,   &
                Stopsign!)
                ROLLBACK;
                RETURN;
Else   /* update complete then insert new row to out table */

li_resultcode = dw_1.Update()
if li_resultcode <> 1 then
        ROLLBACK;
        RETURN;
else
        /* data entry complete */
        COMMIT;
        /* insert new empty row for next input */
        dw_1.SetTransObject(sqlca)
        n = dw_1.insertrow(0)
        dw_1.scrolltorow(n)
        dw_1.setfocus()
        dw_2.setTransObject(Sqlca)
        dw_2.Reset()
End if
End if
ELSE
  Messagebox("Error update ",                    &
                "Data is not complete !",        &
                Stopsign!)
End if
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.6 การเปรียบเทียบความแตกต่างในแต่ละภาษา

สำหรับการพิจารณาเปรียบเทียบความแตกต่างของทั้ง 3 ภาษา ได้แก่ วิชวลเบสิค เดลไฟล์ และเพาเวอร์บิวเดอร์ ทำการพิจารณาจาก 3 ลักษณะด้วยกัน คือ .

4.6.1 ความแตกต่างในลักษณะการใช้งาน (User)

สำหรับในแง่ของผู้ใช้งานแอพพลิเคชันโปรแกรมที่พัฒนาจาก 3 ภาษา จะไม่พบความแตกต่างที่เด่นชัดนัก เนื่องจากถ้าการสร้างแอพพลิเคชันโปรแกรมทำงานชนิดเดียวกัน มีการออกแบบลักษณะหน้าจอ และส่วนที่ทำการติดต่อกับผู้ใช้ได้เหมือนกัน (ยกเว้นการติดต่อกับผู้ใช้ในลักษณะพิเศษบางอย่างที่ต้องอาศัยคอมโพเนนท์เฉพาะบางตัว) สำหรับข้อแตกต่างที่สามารถสังเกตได้ คือในด้านความเร็วในการทำงาน ซึ่งมีผลจากปัจจัยหลายอย่าง ไม่ว่าจะเป็นกลไกภายในของแต่ละตัว ลักษณะการเชื่อมต่อ และลักษณะการเขียนโปรแกรมการทำงานภายในโปรแกรม ซึ่งในโครงงานไม่ได้เน้นการศึกษาเปรียบเทียบในส่วนนี้

4.6.2 ความแตกต่างในลักษณะการพัฒนาแอพพลิเคชันโปรแกรม (Developper)

สำหรับในแง่ของการพัฒนาแอพพลิเคชันโปรแกรมนั้น สามารถพิจารณาการเปรียบเทียบได้หลายส่วนคือ

ลักษณะและโครงสร้างภาษา	● ในเดลไฟล์ มีลักษณะภาษาเป็นออบเจคต์ปาสคาล (Object Pascal) ซึ่งมีประสิทธิภาพมาก เพราะเป็นลักษณะออบเจคต์โอเรียนท์ (Object Oriented) แต่เนื่องจากมีข้อกำหนดต่างๆในการใช้งานค่อนข้างมาก จึงยากสำหรับผู้เริ่มต้น ● ในวิชวลเบสิค และเพาเวอร์บิวเดอร์ มีลักษณะภาษาที่ค่อนข้างง่าย ผู้เริ่มต้นจึงสามารถทำความเข้าใจได้ง่ายกว่าในภาษาเดลไฟล์
ลักษณะการสร้างแอพพลิเคชัน	● ในวิชวลเบสิค และเดลไฟล์มีลักษณะ การสร้างแอพพลิเคชั่นที่คล้ายกัน โดยการนำคอมโพเนนท์ต่างๆมาประกอบใช้งานร่วมกัน ลักษณะของ DB Component มีลักษณะการแบ่งแยกออกเป็นระดับตั้งแต่ Database มาถึง Control ย่อยต่างๆ โดยการสร้างแอพพลิเคชันจำเป็นต้องทำการกำหนดค่าพร็อพเพอร์ตี้ เพื่อให้คอมโพเนนท์ต่างๆเชื่อมต่อกันได้สมบูรณ์ ซึ่งค่อนข้างยุ่งยากสำหรับผู้เริ่มต้น โดยในเดลไฟล์จะใช้การกำหนด พร็อพเพอร์ตี้สำหรับทำงานกับข้อมูลได้มากกว่าวิชวลเบสิค ที่ต้องอาศัยการเขียนโค้ดประกอบกับการกำหนดพร็อพเพอร์ตี้ ● ในเพาเวอร์บิวเดอร์นั้น การสร้างแอพพลิเคชั่นที่ทำงานกับข้อมูลในฐานข้อมูล มีลักษณะการสร้างโดยผ่านออบเจคต์เพียงตัวเดียว คือ Data Windows Object ซึ่งการใช้งานออบเจคต์นี้ ใช้งานโดย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

	อัตโนมัติ กล่าวคือ เพียงทำการกำหนดคำสั่งเอสคิวแอลที่จะใช้ แล้วเลือกลักษณะของการแสดงผลเท่านั้น จึงมีลักษณะการทำงานที่ง่ายกว่าในกรณีที่การทำงานไม่ซับซ้อนมากนัก
การควบคุมการทำงาน (Control)	● ในวิชวลเบสิค และเดลไฟล์มีคอนโทรลมากกว่า และคอนโทรลมีลักษณะการใช้งานที่ง่ายกว่า โดยทั้งวิชวลเบสิค และเดลไฟล์จะใช้คอนโทรลหลายๆตัวมาประกอบกัน เพื่อทำการสร้างแอพพลิเคชัน ดังนั้นจึงสามารถทำการกำหนดการทำงานแบ่งเป็นส่วนย่อยๆได้ง่าย ● ในเพาเวอร์บิวเดอร์ มีคอนโทรลเพียงตัวเดียว คือ Data Windows Control การตอบสนองเหตุการณ์ที่เกิดขึ้นกับส่วนประกอบย่อยๆ จึงจำเป็นต้องเขียนคำสั่งตรวจสอบเหตุการณ์ที่เกิดขึ้นเอง ทำให้การทำงานซับซ้อนกว่าในวิชวลเบสิค และเดลไฟล์

ตารางที่ 4-2 แสดงความแตกต่างในลักษณะการพัฒนาแอพพลิเคชัน

4.6.3 ความแตกต่างในลักษณะการทำงานกับระบบฐานข้อมูล (DB Admistrator)

สำหรับในมุมมองของผู้ควบคุมดูแลระบบฐานข้อมูล (Database Administrator) นั้นจากการศึกษาไม่สามารถสรุปได้ชัดเจนว่าตัวใดน่าจะเหมาะสมที่สุด ดังนั้นจึงได้ทำการทดสอบโดยใช้โปรแกรมเอสคิวแอลเทรซ (SQL Trace) เพื่อทำการตรวจสอบว่าตัวใดส่งคำขอไปยังเอสคิวแอลเซิร์ฟเวอร์มากที่สุดเมื่อทำงานอย่างเดียวกัน ซึ่งในการทดสอบนี้จะบอกได้เพียงว่าตัวใดส่งคำขอไปยังเอสคิวแอลเซิร์ฟเวอร์มากน้อยเท่าใด โดยสรุปไม่ได้แน่ชัดว่าตัวที่ส่งคำขอน้อยจะทำงานได้เร็วกว่า แต่สิ่งที่น่าจะสรุปได้คือ การส่งคำขอน้อยกว่าน่าจะส่งผลให้ส่วนเซิร์ฟเวอร์ทำงานน้อยกว่า ทำให้ปัญหาที่เกิดขึ้นกับส่วนเซิร์ฟเวอร์น้อยลงซึ่งน่าจะเหมาะสมมากกว่าตามความเห็นของผู้ดูแลระบบฐานข้อมูล

```
Connect  Visual Basic 4.0 S6014019   MR.X    4294854981
SQL      Visual Basic 4.0    select usertype,type,name from systypes where usertype>=100 select
502,",USER_NAME() exec sp_server_info 500 select 501,",1 where 'a'='A' set textsize 2147483647 set
ansi_defaults on set cursor_close_on_commit off set implicit_transactions off    10        10        0
         0        COMP_ISO        S6014019        MR.X    4294854981
SQL      Visual Basic 4.0    select substring('NY',status/1024&1+1,1) from master..sysdatabases where
name=DB_NAME()          10        10        0        0        COMP_ISO        S6014019
         MR.X    4294854981
SQL      Visual Basic 4.0    SELECT Config, nValue FROM MSysConf      30        0        1
         0        COMP_ISO        S6014019        MR.X    4294854981
SQL      Visual Basic 4.0    select substring('NY',status/1024&1+1,1) from master..sysdatabases where
name=DB_NAME()          10        0        0        0        COMP_ISO        S6014019
         MR.X    4294854981
SQL      Visual Basic 4.0    SET TEXTSIZE 2147483646          0        0        0        0
         COMP_ISO        S6014019        MR.X    4294854981
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
RPC     Visual Basic 4.0    sp_special_columns "spare", NULL, NULL, "V", "T", "U", "spare", NULL,
NULL, "V", "T", "U"         253     40      7       0       COMP_ISO        S6014019
        MR.X    4294854981
RPC     Visual Basic 4.0    sp_columns "spare", NULL, NULL, NULL, "spare", NULL, NULL, NULL
        450     70      7       0       COMP_ISO        S6014019        MR.X
        4294854981
SQL     Visual Basic 4.0    SET TEXTSIZE 2147483647         10      10      0       0
        COMP_ISO        S6014019        MR.X    4294854981
RPC     Visual Basic 4.0    sp_datatype_info 1, 1           10      10      0       0
        COMP_ISO        S6014019        MR.X    4294854981
RPC     Visual Basic 4.0    sp_datatype_info 12, 12         10      10      0       0
        COMP_ISO        S6014019        MR.X    4294854981
RPC     Visual Basic 4.0    sp_datatype_info -1, -1         10      0       0       0
        COMP_ISO        S6014019        MR.X    4294854981
SQL     Visual Basic 4.0    SET TEXTSIZE 2147483646         0       0       0       0
        COMP_ISO        S6014019        MR.X    4294854981
RPC     Visual Basic 4.0    sp_statistics "Spare", "s6014031", NULL, "%", "N", "Q", "Spare",
"s6014031", NULL, "%", "N", "Q"     763     140     8       0       COMP_ISO
        S6014019        MR.X    4294854981
Connect Visual Basic 4.0 COMP_ISO S6014019      MR.X    4294854981
SQL     Visual Basic 4.0    select usertype,type,name from systypes where usertype>=100 select
502,",USER_NAME() exec sp_server_info 500 select 501,",1 where 'a'='A' set textsize 2147483647 set
ansi_defaults on set cursor_close_on_commit off set implicit_transactions off   10      10      0
        0       COMP_ISO        S6014019        MR.X    4294854981
SQL     Visual Basic 4.0    SELECT DISTINCT "Category"  FROM "s6014031"."Spare"     0
        0       0       0       COMP_ISO        S6014019        MR.X    4294854981
RPC     Visual Basic 4.0    sp_special_columns "staff", NULL, NULL, "V", "T", "U", "staff", NULL,
NULL, "V", "T", "U"         10      10      0       0       COMP_ISO        S6014019
        MR.X    4294854981
RPC     Visual Basic 4.0    sp_columns "staff", NULL, NULL, NULL, "staff", NULL, NULL, NULL
        10      10      0       0       COMP_ISO        S6014019        MR.X
        4294854981
SQL     Visual Basic 4.0    SET TEXTSIZE 2147483647         10      10      0       0
        COMP_ISO        S6014019        MR.X    4294854981
RPC     Visual Basic 4.0    sp_datatype_info 1, 1           10      10      0       0
        COMP_ISO        S6014019        MR.X    4294854981
RPC     Visual Basic 4.0    sp_datatype_info 12, 12         10      10      0       0
        COMP_ISO        S6014019        MR.X    4294854981
RPC     Visual Basic 4.0    sp_datatype_info -1, -1         10      10      0       0
        COMP_ISO        S6014019        MR.X    4294854981
SQL     Visual Basic 4.0    SET TEXTSIZE 2147483646         10      10      0       0
        COMP_ISO        S6014019        MR.X    4294854981
RPC     Visual Basic 4.0    sp_statistics "Staff", "s6014031", NULL, "%", "N", "Q", "Staff",
"s6014031", NULL, "%", "N", "Q"     40      40      1       0       COMP_ISO
        S6014019        MR.X    4294854981
```

รูปที่ 4-24 แสดงบางส่วนของคำขอที่วิชวลเบสิค ส่งไปยังส่วนเซิร์ฟเวอร์

```
Connect COMP_ISO        S6014031                4294838061
SQL     exec sp_server_info 18      70      10      1       0       COMP_ISO
        S6014031                4294838061
SQL     use CS_Interface    0       0       0       0       COMP_ISO        S6014031
                4294838061
SQL     set textsize 32767  0       0       0       0       COMP_ISO        S6014031
                4294838061
SQL     select name from sysusers where uid = user_id()     260     10      5       0
        COMP_ISO        S6014031                4294838061
Attention COMP_ISO      S6014031                4294838061
Connect COMP_ISO        S6014031                4293992281
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
SQL     exec sp_server_info 18      10      10      0       0       COMP_ISO
        S6014031                    4293992281
SQL     use CS_Interface  10        10      0       0       COMP_ISO        S6014031
                4293992281
SQL     set textsize 32767  0       0       0       0       COMP_ISO        S6014031
                4293992281
SQL     select name from sysusers where uid = user_id()  0      0       0       0
        COMP_ISO        S6014031                    4293992281
Attention COMP_ISO      S6014031                    4293992281
SQL     Select StaffID,Name from Staff      40      0       4       0       COMP_ISO
        S6014031                    4293992281
SQL     select user_name (uid), object_name (id), type, crdate from sysobjects where type in ('U', 'V', 'S')
and id = object_id ('s6014031.Borrow')      220     10      2       0       COMP_ISO
        S6014031                    4293992281
SQL     select c.name, t.type, c.length, c.status, t.name, c.prec, c.scale from syscolumns c, systypes t
where c.id = object_id ('s6014031.Borrow') and c.usertype = t.usertype order by colid ASC       270
        20      3       0       COMP_ISO        S6014031                    4293992281
SQL     select name, indid, status from sysindexes where id = object_id ('s6014031.Borrow') and indid
between 1 and 254 order by indid ASC  30        10      0       0       COMP_ISO
        S6014031                    4293992281
SQL     select INDEX_COL ('Borrow', x.indid, c.colid) from sysindexes x, syscolumns c where x.id =
object_id ('s6014031.Borrow') and x.id = c.id and x.name = 'PK_Borrow_2__13' and c.colid <= keycnt
order by c.colid      50        20      0       0       COMP_ISO        S6014031
        4293992281
SQL     select c.name, c.status, o.name from syscolumns c, sysobjects o where c.id = object_id
('s6014031.Borrow') and c.cdefault *= o.id order by colid ASC       10      10      0       0
        COMP_ISO        S6014031                    4293992281
Connect COMP_ISO        S6014031                    4293992281
SQL     exec sp_server_info 18      10      0       0       0       COMP_ISO
        S6014031                    4293992281
SQL     use CS_Interface  0         0       0       0       COMP_ISO        S6014031
                4293992281
SQL     set textsize 32767  0       0       0       0       COMP_ISO        S6014031
                4293992281
SQL     SELECT Category ,Brand ,Model ,StaffID ,ProjectID ,BorrowDate ,BorrowQty ,DueDate
FROM Borrow ORDER BY  Category ASC , Brand ASC , Model ASC , StaffID ASC , ProjectID ASC ,
BorrowDate ASC   30      10      1       0       COMP_ISO        S6014031
        4293992281
SQL     Select DISTINCT Category from Spare         20      10      1       0
        COMP_ISO        S6014031                    4293992281
SQL     Select ProjectID,Name  from project     30      0       1       0       COMP_ISO
        S6014031                    4293992281
SQL     select user_name (uid), object_name (id), type, crdate from sysobjects where type in ('U', 'V', 'S')
and id = object_id ('s6014031.ReturnS')     30      10      1       0       COMP_ISO
        S6014031                    4293992281
SQL     select c.name, t.type, c.length, c.status, t.name, c.prec, c.scale from syscolumns c, systypes t
where c.id = object_id ('s6014031.ReturnS') and c.usertype = t.usertype order by colid ASC      20
        20      0       0       COMP_ISO        S6014031                    4293992281
SQL     select name, indid, status from sysindexes where id = object_id ('s6014031.ReturnS') and indid
between 1 and 254 order by indid ASC  0         0       0       0       COMP_ISO
        S6014031                    4293992281
SQL     select INDEX_COL ('ReturnS', x.indid, c.colid) from sysindexes x, syscolumns c where x.id =
object_id ('s6014031.ReturnS') and x.id = c.id and x.name = 'PK_ReturnS_1__10' and c.colid <= keycnt
order by c.colid      10        10      0       0       COMP_ISO        S6014031
        4293992281
```

รูปที่ 4-25 แสดงบางส่วนของคำขอที่เดลไฟล์ ส่งไปยังส่วนเซิร์ฟเวอร์

```
Connect  PowerBuilder - cs_int COMP_ISO      S6014031        SUNEO  4294898973
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
SQL PowerBuilder - cs_int select usertype,type,name from systypes where usertype>=100
select 502,",USER_NAME() exec sp_server_info 500 select 501,",1 where 'a'='A' set textsize
2147483647 set ansi_defaults on set cursor_close_on_commit off set implicit_transactions off 10
10 0 0 COMP_ISO S6014031 SUNEO 4294898973
SQL PowerBuilder - cs_int select substring('NY',status/1024&1+1,1) from
master..sysdatabases where name=DB_NAME() 10 10 0 0 COMP_ISO
S6014031 SUNEO 4294898973
RPC PowerBuilder - cs_int sp_datatype_info 0, 0 20 20 0 0
COMP_ISO S6014031 SUNEO 4294898973
SQL PowerBuilder - cs_int set implicit_transactions on 0 0 0 0
COMP_ISO S6014031 SUNEO 4294898973
SQL PowerBuilder - cs_int SET TEXTSIZE 32767 0 0 0 0
COMP_ISO S6014031 SUNEO 4294898973
SQL PowerBuilder - cs_int SELECT project.projectid , project.name FROM project
ORDER BY project.projectid ASC 10 10 0 0 COMP_ISO
S6014031 SUNEO 4294898973
SQL PowerBuilder - cs_int SELECT DISTINCT spare.category FROM spare 10
10 0 0 COMP_ISO S6014031 SUNEO 4294898973
SQL PowerBuilder - cs_int SELECT staff.staffid , staff.name , staff.dept
FROM staff ORDER BY staff.staffid ASC 0 0 0 0
COMP_ISO S6014031 SUNEO 4294898973
SQL PowerBuilder - cs_int SELECT DISTINCT spare.brand FROM spare 0
0 0 0 COMP_ISO S6014031 SUNEO 4294898973
SQL PowerBuilder - cs_int SELECT DISTINCT spare.model FROM spare 10
10 0 0 COMP_ISO S6014031 SUNEO 4294898973
SQL PowerBuilder - cs_int Select distinct brand from spare where category = '' 10
10 0 0 COMP_ISO S6014031 SUNEO 4294898973
SQL PowerBuilder - cs_int select model from spare where category = '' 0 0
0 0 COMP_ISO S6014031 SUNEO 4294898973
SQL PowerBuilder - cs_int Select distinct brand from spare where category = 'mouse' 0
0 0 0 COMP_ISO S6014031 SUNEO 4294898973
SQL PowerBuilder - cs_int select model from spare where category = '' 10 10
0 0 COMP_ISO S6014031 SUNEO 4294898973
SQL PowerBuilder - cs_int Select distinct brand from spare where category = 'mouse' 10
10 0 0 COMP_ISO S6014031 SUNEO 4294898973
SQL PowerBuilder - cs_int select model from spare where category = '' 10 10
0 0 COMP_ISO S6014031 SUNEO 4294898973
SQL PowerBuilder - cs_int Select distinct model from spare where category = 'mouse' AND
brand = 'microsoft' 10 0 0 0 COMP_ISO S6014031
SUNEO 4294898973
SQL PowerBuilder - cs_int Select distinct model from spare where category = 'mouse' AND
brand = 'microsoft' 0 0 0 0 COMP_ISO S6014031
SUNEO 4294898973
SQL PowerBuilder - cs_int SET FMTONLY ON select spare.category from spare SET
FMTONLY OFF 10 10 0 0 COMP_ISO S6014031
SUNEO 4294898973
SQL PowerBuilder - cs_int SET FMTONLY ON select spare.brand from spare SET
FMTONLY OFF 10 0 0 0 COMP_ISO S6014031
SUNEO 4294898973
SQL PowerBuilder - cs_int SET FMTONLY ON select spare.model from spare SET
FMTONLY OFF 10 10 0 0 COMP_ISO S6014031
SUNEO 4294898973
SQL PowerBuilder - cs_int SET TEXTSIZE 2147483647 0 0 0
0 COMP_ISO S6014031 SUNEO 4294898973
SQL PowerBuilder - cs_int SET FMTONLY ON SELECT spare.quantity FROM spare
SET FMTONLY OFF 10 10 0 0 COMP_ISO S6014031
SUNEO 4294898973
SQL PowerBuilder - cs_int SELECT spare.quantity FROM spare WHERE (
spare.category = 'mouse' ) and ( spare.brand = 'microsoft' ) and ( spare.model = 'test2' ) 0
0 0 0 COMP_ISO S6014031 SUNEO 4294898973
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
SQL     PowerBuilder - cs_int       SET FMTONLY ON select quantity from spare SET FMTONLY
OFF     0       0       0       0       COMP_ISO        S6014031        SUNEO
        4294898973
SQL     PowerBuilder - cs_int       SET FMTONLY ON select  category from spare SET
FMTONLY OFF     0       0       0       0       COMP_ISO        S6014031
        SUNEO   4294898973
SQL     PowerBuilder - cs_int       SET FMTONLY ON select  brand from spare SET FMTONLY
OFF     0       0       0       0       COMP_ISO        S6014031        SUNEO
        4294898973
SQL     PowerBuilder - cs_int       SET FMTONLY ON select  model from spare SET FMTONLY
OFF     0       0       0       0       COMP_ISO        S6014031        SUNEO
        4294898973
SQL     PowerBuilder - cs_int       SET FMTONLY ON select  quantity from spare SET
FMTONLY OFF     10      10      0       0       COMP_ISO        S6014031
        SUNEO   4294898973
SQL     PowerBuilder - cs_int       update spare SET quantity =24 where category ='mouse' AND
brand ='microsoft' AND model ='test2' AND quantity =25  0       0       0       0
        COMP_ISO        S6014031        SUNEO   4294898973
SQL     PowerBuilder - cs_int       SET FMTONLY ON select  Category from out SET FMTONLY
OFF     0       0       0       0       COMP_ISO        S6014031        SUNEO
        4294898973
```

รูปที่ 4-26 แสดงบางส่วนของคำขอที่เพาเวอร์บิวเดอร์ ส่งไปยังส่วนเซิร์ฟเวอร์

จากผลการทดสอบสรุปได้ว่าเพาเวอร์บิวเดอร์ จะส่งคำขอไปยังส่วนเซิร์ฟเวอร์น้อยที่สุด ในขณะที่วิชวลเบสิคจะส่งมากที่สุด โดยทั้งวิชวลเบสิค และเพาเวอร์บิวเดอร์จะส่งคำขอในลักษณะคล้ายกัน ส่วนในเดลไฟล์จะส่งคำขอในลักษณะที่ต่างออกไป แต่ทั้งนี้จำนวนคำขอที่ส่งออกไปยังส่วนเซิร์ฟเวอร์น่าจะขึ้นอยู่กับปัจจัยอื่นด้วยดังนี้

1. ส่วนมิดเดิ้ลแวร์ที่ใช้แตกต่างกันคือ ในวิชวลเบสิคใช้ ODBC 32 bit และในเพาเวอร์บิวเดอร์ใช้ ODBC 16 bit แต่ในเดลไฟล์ใช้ SQL Link

2. ลักษณะของการเขียนโปรแกรม มีการใช้คำสั่งในลักษณะใด โดยการเขียนคำสั่งเพื่อดึงข้อมูลอาจทำได้หลายแบบ ซึ่งแต่ละแบบก็จะให้ผลต่างกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
สรุปและวิจารณ์

วิชวลเบสิค และเดลไฟล์มีแนวคิดในการเขียนโปรแกรมคล้ายกัน คือมีการแบ่งส่วนของแอพพลิเคชันที่ติดต่อกับฐานข้อมูล ออกเป็นส่วนๆอย่างชัดเจน ได้แก่

- ส่วนมิดเดิ้ลแวร์ คือ โอดีบีซีของวิชวลเบสิค และ เอสคิวแอลลิงค์และบีดีอีของเดลไฟล์
- ส่วนกลไกฐานข้อมูล ซึ่งทำหน้าที่ประมวลผลคำสั่งเอสคิวแอล และจัดการกับผลลัพธ์ คือ เจทเอนจิ้นของวิชวลเบสิค และบีดีอีของเดลไฟล์
- ส่วนดาต้าคอนโทรล และดาต้าแอคเซสออบเจคต์ ซึ่งเป็นการทำงานหลักของแอพพลิเคชันที่เกี่ยวกับฐานข้อมูล ทำหน้าที่จัดการผลลัพธ์ เปลี่ยนแปลงข้อมูล รับคำสั่งต่างๆ แบ่งได้เป็นหลายประเภท เช่น ตาราง เรคอร์ดเซต คิวรี ซึ่งในแต่ละชนิดก็มีวิธีการใช้งานและ หน้าที่แตกต่างกันไป ซึ่งทั้งในเดลไฟล์ และวิชวลเบสิคมีลักษณะคล้ายกันมาก แต่ในเดลไฟล์จะลักษณะของดาต้าคอนโทรลที่เห็นได้ชัดเจนกว่า นั่นคือทุกๆดาต้าออบเจคต์จะมีดาต้าคอนโทรลที่ทำหน้าที่เดียวกันได้เสมอ การสร้างแอพพลิเคชันจึงทำได้ง่ายโดยใช้ดาต้าคอนโทรล และอาจไม่จำเป็นต้องมีการเขียนโค้ดเลย
- ส่วนดาต้าคอนโทรลที่ทำการรับ และแสดงผลข้อมูล มีหลายประเภทให้เลือกใช้งานตามลักษณะของงาน เช่น ลิสต์บอกซ์ อีดิทบอกซ์ คอมโบบอกซ์ และ กริด โดยในวิชวลเบสิคจะมีคอนโทรลเหล่านี้น้อยกว่า เดลไฟล์จึงสามารถให้ความสะดวกได้มากกว่าเมื่อพิจารณาจากส่วนนี้ทั้งวิชวลเบสิคและเดลไฟล์ แบ่งส่วนของออบเจคต์และคอนโทรลออกจากกันได้อย่างชัดเจนในแต่ละแอพพลิเคชัน จะมีคอนโทรลย่อยๆมาก ดังนั้นจะสามารถตอบสนองต่อเหตุการณ์ได้มากกว่า และสามารถแยกส่วนการทำงานได้อย่างชัดเจน แต่จุดอ่อน คือ อาจเป็นความยุ่งยากในการสร้างแอพพลิเคชันมากกว่าเพาเวอร์บิวเดอร์

สำหรับในเพาเวอร์บิวเดอร์ จะมีแนวคิดการใช้งานที่ต่างออกไป การทำงานกับฐานข้อมูลที่ต่างออกไป โดยจะมีส่วนติดต่อกับฐานข้อมูลในลักษณะเดียวกับวิชวลเบสิค และเดลไฟล์ คือ สามารถติดต่อผ่านโอดีบีซีได้ หรือสามารถใช้ไดร์ฟเวอร์เฉพาะก็ได้เช่นกัน ในส่วนการทำงานกับฐานข้อมูล จะมีส่วนหลักๆคือ ทรานสแอคชันออบเจคต์ ซึ่งทำหน้าที่คล้ายกับดาต้าเบสออบเจคต์ในเดลไฟล์ และวิชวลเบสิค คือทำหน้าที่ควบคุมการเริ่มต้น และจบทรานสแอคชัน และควบคุมเกี่ยวกับการเชื่อมต่อกับฐานข้อมูลทั้งหมด การทำงานกับฐานข้อมูลทุกอย่างต้องผ่านทรานแอคชันออบเจคต์เสมอในการทำงานกับข้อมูล เพาเวอร์บิวเดอร์จะไม่แบ่งแยกการทำงานออกเป็นส่วนคอนโทรลย่อยๆ แต่จะมีคอนโทรลหลักเพียงตัวเดียว คือ ดาต้าคอนโทรล ซึ่งจะทำหน้าที่รับข้อมูลและแสดงผลข้อมูลรวมทั้งตอบรับเหตุการณ์ต่างๆได้ซึ่งดาต้าคอนโทรลจะทำการเชื่อมกับดาต้าออบเจคต์ ซึ่งจะทำหน้าที่เชื่อมต่อกับข้อมูลทำการดึงข้อมูลและ ส่งข้อมูลลงในตารางที่กำหนด โดยการออกแบบและสร้างดาต้าออบเจคต์สามารถสร้างได้ง่ายโดยอัตโนมัติ เพียงกำหนดรูปแบบที่ต้องการและ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กำหนดคำสั่งเอสคิวแอลที่ต้องการใช้ในการดึงข้อมูล โดยสามารถกำหนดได้ในลักษณะของคิวบีอี(QBE) โดยไม่จำเป็นต้องรู้คำสั่งเอสคิวแอลก็ได้

จุดเด่นของการทำงานแบบนี้คือ ความง่ายที่สามารถสร้างส่วนติดต่อกับ ดาต้าเบสโดยอัตโนมัติ ไม่จำเป็นต้องสร้างโดยอาศัยส่วนการทำงานย่อยๆมาประกอบกัน ข้อด้อยที่สำคัญคือ เนื่องจากมี่คอนโทรลเพียงตัวเดียวคือ ดาต้าคอนโทรลดังนั้นเหตุการณ์ ที่สามารถตอบรับได้จะมีไม่มากนัก (สามารถทำงานได้ครบถ้วนแต่จำเป็นต้องมีการตรวจสอบเหตุการณ์เพิ่มเติม) ถ้าต้องการเหตุการณ์ที่เกิดกับส่วนประกอบย่อยๆ จะต้องเขียนโปรแกรมตรวจสอบเอง จุดเด่นอีกจุดหนึ่งคือ จะมีฟังก์ชันและพรอพเพอตี้ให้เลือกใช้หลากหลายมาก ถ้าสามารถเรียนรู้การทำงานอย่างแท้จริง ซึ่งนับว่าเป็นเรื่องยากพอสมควรที่จะเรียนรู้และใช้คำสั่งได้อย่างถูกต้อง

ลักษณะเด่นที่สำคัญอีกอย่างหนึ่งของเพาเวอร์บิวเดอร์ คือสามารถทำงานในลักษณะของโปรแกรมพัฒนางานลักษณะเก่าได้ นั่นคือใช้คำสั่งที่เป็นภาษาโปรแกรมทั้งหมด โดยไม่ต้องใช้ออบเจคต์อื่นๆอีก โดยสามารถใช้คำสั่งเอสคิวแอล ได้ทันทีในส่วนของโปรแกรม โดยสามารถใช้ตัวแปรในโปรแกรมเป็นตัวช่วยในการผ่านและ รับค่าข้อมูลได้ทันที ในกรณีที่ผลลัพธ์มีมากกว่า 1 แถวก็สามารถประกาศ เคอร์เซอร์(CURSOR) ซึ่งสามารถใช้ดึงข้อมูลทีละแถวได้ นอกจากนี้สามารถใช้ไดนามิกเอสคิวแอล (Dynamic SQL) ซึ่งเป็นคำสั่งเอสคิวแอลที่สามารถเปลี่ยนแปลงได้หลายรูปแบบในขณะทำงาน ดั้งนั้นจะเห็นว่าเพาเวอร์บิวเดอร์ มีความยืดหยุ่นในการสร้างแอพพลิเคชันมาก มีทางเลือกในการทำงานได้หลายทางตั้งแต่แบบง่ายๆไปจนถึง แบบที่ซับซ้อนที่สุด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ขอขอบคุณ อาจารย์บรรจง ปิยธำรง อาจารย์ที่ปรึกษา ที่คอยชี้แนะ และให้คำปรึกษาตลอดเวลาการทำโครงงานการเชื่อมต่อระบบไคล์เอนท์เซิร์ฟเวอร์

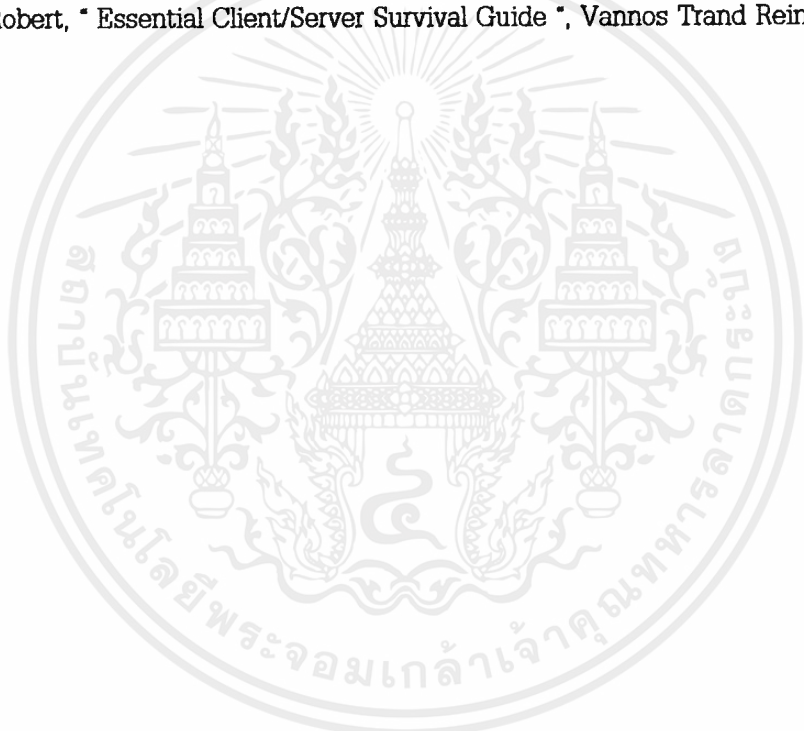
ขอขอบคุณ อาจารย์วิบูลย์ พร้อมพานิชย์ และเจ้าหน้าที่ห้องไอโซเน็ท ที่เอื้อเฟื้ออุปกรณ์ และสถานที่สำหรับทำโครงงานการเชื่อมต่อระบบไคล์เอนท์เซิร์ฟเวอร์

ขอขอบคุณ บิดามารดา เพื่อนๆ และอาจารย์ทุกท่าน ที่ให้กำลังใจ และความช่วยเหลือจนการทำโครงงานการเชื่อมต่อระบบไคล์เอนท์เซิร์ฟเวอร์สำเร็จ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

1. William R. Vaughn, " Visual Basic and SQL Server ", Microsoft Press, 620p, 1996
2. Dan Rahmel and Ron Rahmel, " Developing Client/Server Application with Visual Basic 4.0 ", SAMS, 1038p, 1996
3. Josepth D. Booth, " Delphi Client/Server Developer's Guide ", M&T Books, 639p, 1997
4. David Mc Clananan, " Power Builder 4 A Developer's Guide ", M&T Books, 873p, 1995
5. Brian J. Smith, " Foundation of Power Builder 5.0 Programming ", IDG Books, 704p, 1996
6. Orfali Robert, " Essential Client/Server Survival Guide ", Vannos Trand Reinhold, 527p, 1994



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้