การพัฒนาระบบงานประยุกต์บนระบบฐานข้อมูลแบบไคล์แอนด์-เซอร์เวอร์

APPLICATION DEVELOPMENT IN A CLIENT-SERVER DATABASE ENVIRONMENT

โดย

นายณรงค์ โต๊ะกู

นายอดุลย์ ทรงธรรมบวร

อาจารย์ที่ปรึกษา

ผศ.ดร.ศุภมิตร จิตตะยโศธร

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมบัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้า วิทยาเขตเจ้าคุณทหาร ลาดกระบัง

ปีการศึกษา 2537

ปริญญานิพนธ์ปีการศึกษา 2537

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง

การพัฒนาระบบงานประยุกต์บนระบบฐานข้อมูลแบบไคล์แอนด์-เซอร์เวอร์

APPLICATION DEVELOPMENT IN A CLIENT-SERVER DATABASE ENVIRONMENT

ผู้จัดทำ

1. นายณรงค์ โต๊ะกู
2. นายอดุลย์ ทรงธรรมบวร

.....อาจารย์ที่ปรึกษา

(ผศ.ดร.ศุภมิตร จิตตะยโศธร)

การพัฒนาระบบงานประยุกต์บนระบบฐานข้อมูลแบบไคล์แอนด์-เซอร์เวอร์

นายณรงค์ โต๊ะกู
นายอดุลย์ ทรงธรรมบวร
ผศ.ดร.ศุภมิตร จิตตะยโศธร อาจารย์ที่ปรึกษา
ปีการศึกษา 2537

บทคัดย่อ

ปริญญานิพนธ์นี้นำเสนอข้อมูลเกี่ยวกับการพัฒนาโปรแกรมประยุกต์บนระบบฐานข้อมูลแบบไคล์แอนด์-เซอร์เวอร์ โดยเลือก GUPTA Client-Server RDBMS เป็นตัวหลักในการพัฒนาซึ่งเน้นที่จะพัฒนาภายใต้เทคโนโลยีเชิงวัตถุ

นอกจากนี้ยังกล่าวถึงเทคโนโลยีไคล์แอนด์-เซอร์เวอร์และเทคโนโลยีเชิงวัตถุที่ใช้ในการพัฒนางานประยุกต์ รวมทั้งได้ศึกษาผลิตภัณฑ์และมาตราฐานการติดต่อต่าง ๆ เช่น ODBC บนระบบฐานข้อมูลแบบไคล์แอนด์-เซอร์เวอร์ที่มีอยู่ในท้องตลาดเพื่อเป็นแนวทางในการพัฒนาโปรแกรมประยุกต์บนระบบฐานข้อมูลแบบไคล์แอนด์-เซอร์เวอร์ต่อไป

APPLICATION DEVELOPMENT IN A CLIENT-SERVER DATABASE ENVIRONMENT

Narong Tohku

Adul Songtomborvorn

Assistance Professor Suphamit Chittayasothorn Advisor

1994

Abstract

This thesis intends to propose all information concerned the application development on client-server database system. GUPTA Client-Server RDBMS is specifically chosen to be the core development tool under object-oriented technology.

Besides, this thesis is also related to the client-server technology and object-oriented technology in which both of them are involved in developing application. Furthermore, many commercial client-server RDBMS products are thoroughly studied in order to be useful guidance of application development process.

สารบัญ

หน้า

ส่วนที่ 1
บทนำ

บทที่ 1
ปัญหาและวิธีในการแก้ปัญหา

ปัจจุบันคอมพิวเตอร์ส่วนบุคคล ได้ถูกนำมาเชื่อมต่อกันเป็นลักษณะโครงข่ายมากขึ้น ก่อกำเนิดระบบการใช้งานเครือข่ายแบบใหม่แบบใหม่คือ ไคล์แอนด์-เซอร์เวอร์ ซึ่งยังผลให้ระบบฐานข้อมูลเป็นเดิมคือระบบฐานข้อมูลแบบศูนย์กลาง ซึ่งถือว่าข้อมูลเป็นทรัพยากรที่ไม่สามารถทำการแบ่งกันได้ โดยแต่ละสถานีนั้นจะทำการดูแลข้อมูลและอัพเดจฐานข้อมูลรวมในแต่ละช่วงของเวลามาเป็นระบบฐานข้อมูลแบบไคล์แอนด์-เซอร์เวอร์ (Client-Server Database) ซึ่งมีลักษณะการกระจายศูนย์กลาง โดยแต่ละสถานีจะมีตัวประมวลและตัวจัดการฐานข้อมูลเป็นของตัวเอง ระบบนี้จะถือว่าข้อมูลเป็นทรัพยากรที่แบ่งกันได้ แต่ละสถานีจะเป็นตัวที่คอยดูแล และวิเคราะห์ความต้องการในข้อมูลเอง

เทคโนโลยีแบบเก่าที่ใช้ในการพัฒนาโปรแกรมประยุกต์ อาจไม่สามารถนำมาใช้ได้อีก กับระบบฐานข้อมูลแบบไคล์แอนด์-เซอร์เวอร์ ในปริญญานิพนธ์ฉบับนี้ จะกล่าวถึง การพัฒนาโปรแกรมประยุกต์ด้วยเทคโนโลยีการวิเคราะห์เชิงวัตถุ (Object-Oriented Analysis) ซึ่งเป็นการพัฒนาโปรแกรมประยุกต์เชิงวัตถุ (Object-Oriented) บนระบบฐานข้อมูลแบบไคล์แอนด์-เซอร์เวอร์ โดยมีเป้าหมายหลักของการพัฒนา 2 หัวข้อใหญ่ ๆ คือ

1) ทำให้การพัฒนาโปรแกรมประยุกต์สั้นลง และต้นทุนต่ำลง
2) ทำให้ต้นทุนในการบำรุงรักษาโปรแกรมประยุกต์ลดลงเช่นกัน

บทที่ 2
ภาพรวมของปริญญานิพนธ์

ในปริญญานิพนธ์ฉบับนี้แบ่งออกเป็น 4 ส่วนใหญ่ คือ

ส่วนที่ 1 เป็นบทนำที่กล่าวถึงปัญหา และวิธีในการแก้ปัญหาที่พบในปัจจุบัน และบอกถึงภาพรวมของปริญญานิพนธ์ทั้งหมด

ส่วนที่ 2 แบ่งออกเป็น 3 บท ดังนี้

บทที่ 3 บรรยายถึงเทคโนโลยีของไคล์แอนด์-เซอร์เวอร์ (Client-Server) แบ่งออกเป็นหัวข้อ ๆ เช่น พื้นฐานความคิดก่อนการเกิดไคล์แอนด์-เซอร์เวอร์, อุปกรณ์ในระบบไคล์แอนด์-เซอร์เวอร์, ปัญหาและส่วนเสียของไคล์แอนด์-เซอร์เวอร์, บทบาทของมัลติมีเดียกับไคลแอนด์-เซอร์เวอร์, การประยุกต์ใช้งานระบบไคล์แอนด์-เซอร์เวอร์ กับงานฐานข้อมูล

บทที่ 4 กล่าวถึงเทคโนโลยีเชิงวัตถุ (Object-Oriented Technology) แยกเป็นหัวข้อ ๆ เช่น ตำนานภาษา Object-Oriented, หลักการ-การใช้งาน-คำศัพท์ในเทคโนโลยีเชิงวัตถุ, เทคโนโลยีเชิงวัตถุกับงานฐานข้อมูล

บทที่ 5 การวิเคราะห์ระบบงานเชิงวัตถุ (Object-Oriented Analysis) แยกเป็นหัวข้อ ๆ เช่น โมเดลสารสนเทศ, โมเดลสถานะ, โมเดลประมวลผล, ตัวอย่างงานระบบฐานข้อมูลสำหรับร้านวีดีโอ (VDO Database) ที่พัฒนาโดยอ้างอิงกับทฤษฎี

ส่วนที่ 3 แบ่งออกเป็น 4 บท ดังนี้

บทที่ 6 จะกล่าวถึง GUPTA SQL Client-Server RDBMS ซึ่งเป็นซอร์ฟแวร์ที่ใช้ในการพัฒนาโปรแกรมประยุกต์ โดยแยกออกเป็นหัวข้อต่าง ๆ เช่น ตัวขับเคลื่อนฐานข้อมูลและระบบโครงข่ายที่สนับสนุน, มาตรฐานที่สนับสนุน, การจัดการวัตถุ, ชั้นของวัตถุ (Object Class), การบริหารงานกลุ่ม, การทำให้ผลดีที่สุด (Optimize) ในฟรอนท์-เอนด์, คุณสมบัติที่เป็นจุดเด่น และเครื่องมือในการพัฒนาโปรแกรมประยุกต์ โดยเนื้อหาทั้งหมดจะครอบคลุมถึง เวอร์ชั่น 4.0 และ 5.0 ของ GUPTA SQLWindows

ในบทที่ 7 กล่าวแนะนำเบื้องต้น พร้อมทั้งวิธีการใช้งานของ Microsoft Open Database Connectivity (ODBC) แบ่งออกเป็นหัวข้อ ๆ เช่น ตำนานของการเชื่อมต่อฐานข้อมูลแบบเปิด (ODBC), การติดต่อ ODBC, สถาปัตยกรรม ODBC, ชนิดของตัวขับ, ความสัมพันธ์ระหว่างโปรแกรมประยุกต์กับตัวขับ, ขั้นตอนเบื้องต้นของโปรแกรมประยุกต์ในการใช้งาน ODBC

ในบทที่ 8 จะแนะนำการสร้าง Front-End สำหรับ Client-Server ด้วย Visual Basic 3.0 แบ่งออกเป็นหัวข้อ ๆ เช่น การสร้างแหล่งข้อมูลของ ODBC จากฐานข้อมูลแบบ Client-Server, การใช้ชั้นความเข้าใจกันได้ (Compatibility Layer) ของ Microsoft Access 2.0 สำหรับ Visual Basic 3.0

ในบทที่ 9 จะเป็นการแสดงตัวอย่างฐานข้อมูลบนมาตรฐาน ODBC เช่น Microsoft SQL Server for Windows NT 4.2 และ Microsoft Access 2.0

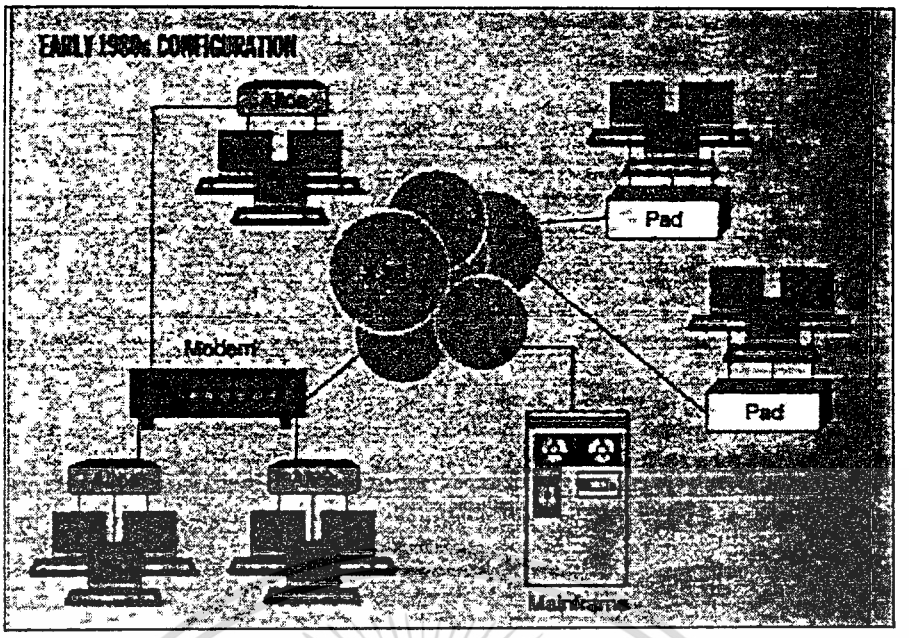
ส่วนที่ 4 เป็นบทสรุปปัญหาและวิธีการในการแก้ปัญหาพร้อมทั้งวิจารณ์งานที่ได้ทำมาทั้งหมด รวมทั้งความคิดเห็นและข้อเสนอแนะต่าง ๆ

ส่วนที่ 2
การพัฒนาโปรแกรมประยุกต์

บทที่ 3
เทคโนโลยีของไคล์แอนด์-เซอร์เวอร์ (Client-Server)

3.1 พื้นฐานความคิดก่อนการเกิดไคล์แอนด์-เซอร์เวอร์(Client/Server)

ในอดีตที่ผ่านมาทุกคนที่เกี่ยวข้องกับวงการธุรกิจ เมื่อได้ยินคำว่าคอมพิวเตอร์เมนเฟรม แล้ว จะเกิดความรู้สึกว่าเป็นเรื่องเทคโนโลยีชั้นสูง และจะไม่ค่อยกล้าที่จะเข้าไปยุ่งกับมัน จะปล่อยให้เป็นหน้าที่ของนักคอมพิวเตอร์ในฝ่ายดำเนินกรรมวิธีโดยเฉพาะ เพราะว่าการใช้งานคอมพิวเตอร์เมนเฟรมมีความยุ่งยากเกินกว่าที่คนธรรมดา ๆ จะเข้าใจได้ทำให้ข่าวสารข้อมูลต่าง ๆ ซึ่งจำเป็นต้องใช้ในการวางแผนกลยุทธ์ ถูกจำกัดอยู่เฉพาะภายในฝ่ายประมวลผลเท่านั้น หรือจะมีข้อมูลข่าวสารออกมาสนับสนุนฝ่ายการตลาดบ้างแต่ก็ไม่ตรงกับที่ฝ่ายการตลาดต้องการอย่างแท้จริง พูดง่าย ๆ ว่า ข้อมูลที่ต้องการได้ก็ไม่ได้ แต่ข้อมูลที่ไม่ต้องการกลับได้มา แต่นั่นก็ไม่ใช่ความผิดของฝ่ายประมวลผลข้อมูลเพราะไม่มีใครจะประมวลผลข้อมูลได้ถูกใจมากท่ากับการประมวลผลด้วยตนเอง ทำให้เกิดความคิดว่าถ้าหากสามารถจัดสรรข้อมูลส่วนกลางให้ถึงมือผู้ใช้ได้โดยตรงโดยผ่านคอมพิวเตอร์ส่วนบุคคลซึ่งคนส่วนใหญ่ มีความถนัดทั้งในด้านฮาร์ดแวร์และซอฟต์แวร์อยู่แล้ว จะสามารถดำเนินกรรมวิธีข้อมูลด้วยตนเอง ในรูปแบบที่ตนเองต้องการได้ ในปัจจุบันคอมพิวเตอร์ขนาดเล็กส่วนบุคคล ได้ถูกนำมาใช้งานในลักษณะแบบใช้ตัวเดียวใด ๆ ก่อน ยิ่งใช้ยิ่งได้รับข่าวสารข้อมูลมากขึ้น แต่ก็ยังไม่เพียงพอที่จะต่อสู้ในภาคธุรกิจ เพราะว่าข้อมูลต่าง ๆ ที่จะมาสนับสนุนส่วนใหญ่กระจัดกระจายอยู่ในหลายแหล่งข้อมูล การที่จะได้ข้อมูลที่ดีมาใช้งานจำเป็นต้องกว้านมาจากหลาย ๆ แหล่ง จึงเกิดความคิดว่าถ้าสามารถเชื่อมต่อระบบคอมพิวเตอร์ส่วนบุคคลเข้าด้วยกันในลักษณะโครงข่ายแล้ว จะทำให้สามารถแลกเปลี่ยนข้อมูลหรือดึงข้อมูลภายในโครงข่ายได้ตามใจชอบ ผู้ใช้ก็ไม่จำเป็นต้องไปเรียนรู้อะไรมากไปกว่าความคุ้นเคยตามปกติกับคอมพิวเตอร์และซอฟต์แวร์ที่ตัวเองมีใช้อยู่ที่บ้านเช่นถ้าถนัด โลตัสก็ใช้โลตัสประมวลผลถ้าถนัดดีเบสก็ใช้ดีเบสประมวลผลข้อมูล ถ้าถนัดเวิร์ดโพรเซสเซอร์ก็ใช้เวิร์ดโพรเซสเซอร์ประมวลผลข้อมูล เรียกว่าถนัดอะไรก็ใช้ซอฟแวร์ตัวนั้นประมวลผล ระบบไคล์แอนด์-เซอร์เวอร์ได้ถูกค้นคิดขึ้นเพื่ออำนวยความสะดวกให้ผู้ใช้ให้สามารถที่จะดึงข้อมูลจากโครงข่ายมาทำงานหรือประมวลผลด้วยกรรมวิธีที่ผู้พัฒนาถนัด ไม่ต้องไปพะวงอยู่กับการศึกษาระบบเมนเฟรมให้ปวดหัวอีกต่อไป

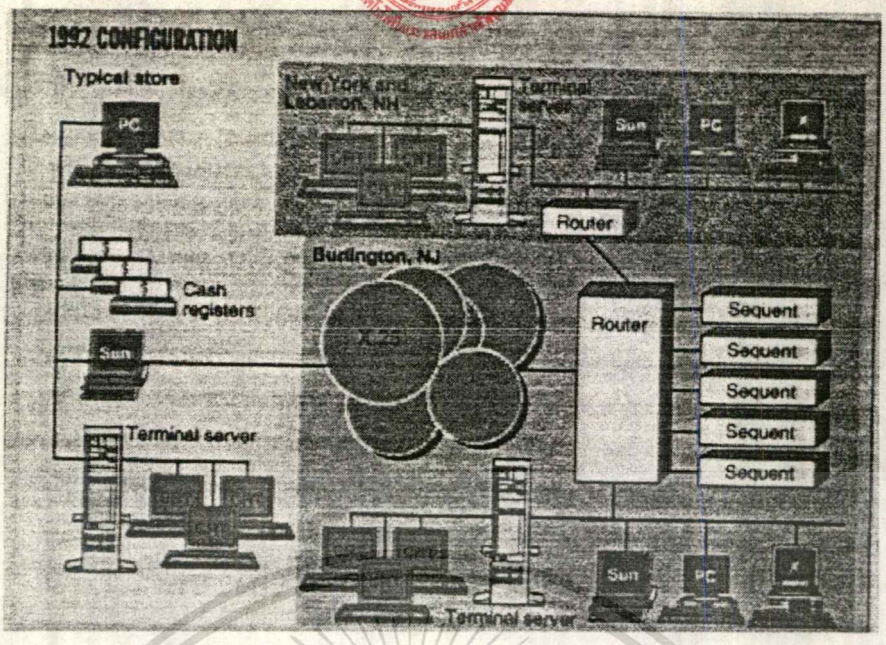


ดูแล้วก็น่าจะเป็นที่พอใจกับทุกคนเพราะทุกคนสามารถใช้ซอฟต์แวร์ที่ตนถนัด และสามารถควบคุมข่าวสารต่าง ๆ ให้ออกมาตามรูปข่าวกรองที่ตนเองต้องการได้ นี่แหละที่เราเรียกว่าไคล์แอนด์ ในส่วนของเซอร์เวอร์ ก็คือส่วนที่เป็นต้นกำเนิดหรือส่วนที่เป็นแหล่งเก็บข้อมูลหลาย ๆ แหล่งมารวม ๆ กันแล้วเรารียกว่า ระบบไคล์แอนด์-เซอร์เวอร์ระบบคอมพิวเตอร์ขนาดเล็กที่ต่ออยู่ในลักษณะของไคล์แอนด์-เซอร์เวอร์นั้น จะมีลักษณะขึ้นอยู่กับความต้องการของผู้พัฒนา ไม่ว่าจะในรูปแบบของโครงข่ายฮาร์ดแวร์ และซอฟต์แวร์ ขึ้นอยู่กับความต้องการของผู้พัฒนาทั้งสิ้น ดูแล้วน่าจะง่ายดังใจนึก แต่ถ้าลองได้ลงมือทำเข้าจริง ๆ จะพบกับปัญหามากมาย และยิ่งถ้าผู้พัฒนาขาดประสบการณ์ที่เพียงพอแล้วจะยิ่งเพิ่มความปวดหัวให้ผู้พัฒนามากขึ้นไปอีก เพราะว่าการค้นหาที่มาของเจ้าตัวปัญหาจะกระทำได้ยากมากในระบบนี้ แต่ถ้าหน่วยงานของผู้พัฒนากล้าและคิดว่าเก่งพอที่จะทดสอบระบบนี้ผลที่จะได้รับคิดว่าคุ้มค่า เพราะระบบของผู้พัฒนาจะมีความอ่อนตัวและสามารถคงอยู่ได้ไม่ว่าเทคโนโลยีคอมพิวเตอร์จะเปลี่ยนไปอย่างรวดเร็วเท่าไรก็ตาม เหตุผลสำคัญอีกอันหนึ่งที่ทำให้บริษัทฯ ส่วนใหญ่ในอนาคตเริ่มคิดหันมาใช้ไคล์แอนด์-เซอร์เวอร์ อีกเหตุผลหนึ่งก็คือ เพื่อลดต้นทุนของระบบงานคอมพิวเตอร์ เช่นผู้พัฒนาอาจสามารถนำคอมพิวเตอร์เวิร์กสเตชันสักหนึ่งหมวดมาทำงานร่วมกัน ก็จะได้พลังคำนวณสูงถึง 1500 ล้านคำสั่งต่อวินาที (MIPS) แล้ว แต่กลับใช้เงินลงทุนเพียงล้านกว่าบาทเท่านั้น และยังให้ความอ่อนตัวสำหรับพร้อมรับมือกับการเปลี่ยนแปลงในอนาคตด้วย

เมื่อหันมาใช้ไคล์แอนด์-เซอร์เวอร์ แล้วผู้ที่ต้องการใช้ข้อมูลจากเซอร์เวอร์ก็จะดึงข้อมูลจากโครงข่ายและสามารถนำมาประมวลผลด้วยซอฟต์แวร์บนเครื่องของตัวเองที่ถนัด เช่น สามารถนำข้อมูลจากฐานข้อมูลนำไปใส่ในโปรแกรมสเปรดชีตหรือเวิร์ดโพรเซสเซอร์

ปัจจุบันร้านจำหน่ายสินค้าของบริษัทเบอริงตันในสหรัฐ จำนวน 190 ร้าน ได้นำระบบ ไคล์แอนด์-เซอร์เวอร์มาใช้งานในทุกระดับ นับตั้งแต่เครื่องควบคุมการอ่านบาร์โค้ด จนกระทั่งถึงการทำรายงานแสดงภาพสถิติการขาย ร้านจำหน่ายสินค้าของเบอริงตัน ใช้ NCR 7052 ควบคุมการจำหน่ายสินค้าและบัญชีเงินสด NCR 7052 เป็นคอมพิวเตอร์ที่ใช้ระบบปฏิบัติการดอส และต่อเชื่อมเป็นพีซีเครือข่าย โดยมีไฟล์เซอร์เวอร์หนึ่งตัวประจำอยู่ที่ร้านจำหน่ายสินค้า เพื่อทำหน้าที่เป็นศูนย์กลางบริการ ไฟล์ให้กับเทอร์มินัลและพีซีเทอร์มินัลที่ทำหน้าที่ลงทะเบียนเงินสดหรือที่เรียกว่าแคชเรจิสเตอร์จะต่อเชื่อมกับโครงข่ายอิเทอร์เน็ตโดยมี Sun เวิร์กสเตชันเป็นมันสมองทำหน้าที่บริการไฟล์ ร้านจำหน่ายของเบอร์ริงตันจะสื่อสารกับระบบคอมพิวเตอร์ส่วนกลางที่สำนักงานใหญ่โดยผ่านระบบสื่อสารดาวเทียม จานขนาดเล็ก (VAST) โพรโตคอลที่ใช้ติดต่อในเครือข่ายท้องถิ่นในร้านค้าย่อยใช้ทีซีทีไอดี ในขณะที่การติดต่อกับสำนักงานใหญ่ซึ่งตั้งอยู่ห่างไกลกันหลายร้อยกิโลเมตรใช้โพรโตคอล X.25 ซึ่งนอกจากจะสามารถติดต่อกับสำนักงานใหญ่ได้โดยตรงแล้วยังสามารถติดต่อกับส่วนของบัตรเครดิต VISA และมาสเตอร์การ์ดได้อีกด้วย

รายการ(Transaction)ที่เกิดจากการจำหน่ายสินค้าจะถูกยิงผ่านระบบดาวเทียมไปยัง เราเตอร์(Router) ที่สำนักงานใหญ่ ซึ่งจะทำหน้าที่แยกข้อมูลต่าง ๆ ไปยังส่วนปลายทาง ซึ่งได้แก่ส่วนควบคุมด้านการเงิน และส่วนควบคุมด้านการธุรการเบอริงตันเลือกใช้ Oracle เป็นระบบฐานหลักสำหรับควบคุมและบริหารฐานข้อมูล โดยใช้ซีเควนคอมพิวเตอร์ เป็นแพลตฟอร์มและเลือกให้ซอฟต์แวร์ของ UNIX Labs ควบคุมระบบ On-line transaction และการอัปเดตข้อมูล ซอฟต์แวร์ตัวที่ UNIX Labs ทำขึ้นมาใช้กับการนี้ ใช้ชื่อว่าทักซิโด ตัวซอฟต์แวร์ทักซิโดตัวนี้ จะวิ่งอยู่บนชุดคอมพิวเตอร์หลาย ๆ ชุดแต่ละชุดก็จะดูดข้อมูลในส่วนของตัวเองไปโปรเซส ซึ่งเป็นวิธีการกระจายการโปรเซสในสไตล์ของไคล์แอนด์-เซอร์เวอร์ผลดีก็คือ เมื่อชุดคอมพิวเตอร์เพียงชุดนั้น ผิดกับแต่ก่อนที่ใช้เมนเฟรมเมื่อเกิดขัดข้องขึ้นงานทุกชนิดจะหยุดทันที



3.2 อุปกรณ์คอมพิวเตอร์ที่ใช้ในระบบไคล์แอนด์-เซอร์เวอร์

แนวความคิดระบบไคล์แอนด์-เซอร์เวอร์ เติบโตบนรากฐานของระบบเปิด หรือพูดง่าย ๆ ว่าถ้าพูดถึงไคล์แอนด์-เซอร์เวอร์ แล้วแทบจะแทนด้วยคำว่าระบบเปิดได้ทันทีระบบเปิดจะทำให้ค่าความเสี่ยงทางเทคโนโลยีต่ำลง เพราะว่าสามารถที่จะเลือกอุปกรณ์คอมพิวเตอร์ และสื่อสารจากยี่ห้อใด ๆ ก็ได้ ไม่ต้องโดยผูกขาดแบบสมัยก่อน เช่น บริษัท พีบีเอส ซึ่งเลือกใช้ ไคล์แอนด์-เซอร์เวอร์ เช่นกัน แต่ใช้อุปกรณ์คอมพิวเตอร์และซอฟต์แวร์ที่ต่างไปจากเบอริงตัน พีบีเอส ใช้ข้อมูลจากอินเกรส ซึ่งวิ่งบนเครื่อง VAX เซอร์เวอร์ และใช้เครื่องแมคอินทอชเป็นไคล์แอนด์ ในระยะหลังพีบีเอส นำข้อมูลตัวเดิม ซึ่งเคยวิ่งบน HP 9000 แต่ก็ยังคงวิ่งข้อมูลเครื่อง VAX คู่ขนานด้วย

พีบีเอส ใช้แมคอินทอชเป็นไคล์แอนด์ทำหน้าที่ป้อนคำถามและป้อนข้อมูลให้ระบบกับยังวางแผนที่จะนำภาษีสอบถามด้วยกราฟิกมาใช้กับเครื่องแมคด้วย โปรแกรมประยุกต์ที่ทางพีบีเอสใช้งานอยู่ในปัจจุบันมีสองโปรแกรมประยุกต์คือ โปรแกรมประยุกต์ในด้านการตรวจสอบตารางการปฏิบัติงานและโปรแกรมประยุกต์ในด้านข้อมูล ซึ่งใช้บรรจุข้อมูลโปรแกรม พีบีเอส

พนักงานของบริษัท เพลลิกลิโนต้องการใช้เมาส์ เพราะว่าพนักงานบริษัทนี้ถนัดในการใช้เมาส์มากไคล์แอนด์-เซอร์เวอร์ ก็สามารถจัดการให้เข้ากับความต้องการได้ โดยใช้ภาษาทางกราฟิกของแมคตัวไคล์แอนด์ นั้น สามารถใช้ซอฟต์แวร์หรือคอมพิวเตอร์ใดก็ได้ ขึ้นอยู่กับความต้องการของผู้ใช้เอง ไม่ว่าซอฟแวร์วิ่งบนเซอร์เวอร์จะเป็นอะไรที่ไคล์แอนด์ ก็ยังคงเหมือนเดิมคือเป็นคอมพิวเตอร์และซอฟต์แวร์ที่ตนเองถนัด

เมื่อมีซอฟต์แวร์ตัวใหม่ให้กับไคล์แอนด์ ผู้พัฒนาก็สามารถเปลี่ยนแปลงซอพต์แวร์ที่ ไคล์แอนด์ ได้โดยไม่มีผลกระทบแม้แต่น้อยกับเซิรฟ์เวอร์ เนื่องจากโครงสร้างของ ไคล์แอนด์-เซอร์เวอร์ เป็นแบบมอดูลาร์เป็นสัดเป็นส่วน แต่ละส่วนก็ทำงานเป็นอิสระแก่กัน ทำให้การขยายหรือการสับเปลี่ยนสามารถทำได้สะดวก

ข้อดีของไคล์แอนด์-เซอร์เ

คุณสมบัติ	ประโยชน์
การโยงใยขนาดเล็กให้ประสิทธิภาพการทำงานเต็มกำลัง	ทำให้มีการลดขนาดเครื่องลดและสามารถติดตั้งในสำนักงานได้
คอมพิวเตอร์อะเรย์จะมีความเร็วเป็นพัน MIPS	ระบบผู้พัฒนาสามารถจะใช้เครื่องที่มีกำลังเท่าใดก็ได้ไม่ขึ้นอยู่กับเครื่องใดเครื่องหนึ่ง
บาง Workstation อาจจะมีกำลังเท่ากับเครื่องเมนเฟรมแต่ราคาน้อยกว่า 1 ใน 10	ให้กำลังที่มากกว่าสำหรับราคาที่น้อยกว่า ระบบจะให้ความยืดหยุ่นสูงในการที่จะจัดซื้อเครื่องและเพิ่มประโยชน์ให้กับผู้พัฒนา
ระบบปฏิบัติการ	ผู้พัฒนาสามารถเลือก hardware, software และบริการจากผู้ขายหลายแห่ง
ระบบจะขยายได้ง่าย	มันง่ายที่ผู้พัฒนาจะทำการเปลี่ยนระบบของผู้พัฒนาเมื่อผู้พัฒนาต้องการเปลี่ยน
การทำงานของไคล์แอนด์ในสภาพแวดล้อมที่เป็นส่วนตัว	ผู้พัฒนาสามารถรวมเข้าไปกับระบบหรือทำงานในส่วนตัวเองได้

3.3 ส่วนเสียของระบบ ไคล์แอนด์-เซอร์เวอร์

ส่วนเสียของระบบ ไคล์แอนด์-เซอร์เวอร์ ก็คือ การหาข้อผิดพลาดเมื่อเกิดข้อผิดพลาดขึ้นจะหาได้ยากมาก เพราะผู้พัฒนาจะต้องตรวจสอบแทบทุกส่วนของระบบ ทั้งนี้เนื่องจากแต่ละส่วนใช้วิธีการตรวจสอบเฉพาะส่วนของมันเอง ซึ่งไม่เหมือนกับระบบเมนเฟรมที่ส่วนใหญ่มีเครื่องมือหรือซอฟต์แวร์ทูลที่สามารถช่วยหาสาเหตุความผิดพลาดทั้งระบบที่ค่อนข้างหาจุดบกพร่องได้ง่ายกว่า

อีกปัญหาหนึ่งที่เป็นข้อเสียของระบบไคล์แอนด์-เซอร์เวอร์ ก็คือ จำนวนเจ้าหน้าที่ที่ควบคุมการทำงานจะต้องใช้ปริมาณคนมากขึ้น

ปัญหาของไคล์แอนด์-เซิร์ฟเวอร์

ปัญหา	ความหมาย
การดูแลระบบ	ส่วนประกอบมักทำงานไม่เข้ากัน จะมีผู้รับเคราะห์เมื่อมีสิ่งผิดพลาด
ขาดเครื่องมือสนับสนุน	ในสถาปัตยกรรมไคล์แอนด์เซอร์เวอร์ ผู้พัฒนาต้องหาและสร้างเครื่องมือเอง
การฝึกหัดบ่อย	การพัฒนา Software สำหรับ MAC/WIN แตกต่างจาก COBOL หรือ C

เมื่อเปรียบเทียบทางด้าน ไคล์แอนด์ ระหว่างกราฟิกยูสเซอร์อินเตอร์เฟซวิ่งบนแมคอินทอช กับวินโดว์ส(Windows)วิ่งบนพีซีแล้วระบบไคล์แอนด์บนแมคอินทอซค่อนข้างใช้งานได้ง่ายกว่า เพราะว่าการใช้ยูสเซอร์อินเทอร์เฟซของแมคมีรูทีนที่เป็นมาตรฐาน ผิดกับวินโดว์สซึ่งยังมีรูทีนที่อยู่ในขั้นตอนการพัฒนา จึงยังขาดความเป็นมาตรฐานอยู่มาก วินโดว์สแต่ละชันก็ยังมีความแตกต่างกันและบางส่วนไม่มีการพัฒนาอย่างต่อเนื่อง

เมื่อจะใช้งานไคล์แอนด์อาจจะต้องพบกับความวุ่นวายพอสมควรในระยะแรกเพราะว่าผู้พัฒนาอาจจะต้องเรียนรู้การใช้ซอฟต์แวร์หลากหลายของไคล์แอนด์โดยเฉพาะในเวลานี้เทคโนโลยีของซอฟต์แวร์มีการเปลี่ยนแปลงเร็ว แต่ถ้าผู้พัฒนาเป็นคนประเภทหัวเก่า เช่นเคยชินกับการใช้งานบนระบบเมนเฟรมที่มักจะใช้ภาษาโคบอลหรือภาษาซีเขียนโปรแกรมแต่เพียงอย่างเดียว ผู้พัฒนาอาจพบกับความหนักใจเพราะว่าผู้พัฒนาจะต้องศึกษาการเขียนโปรแกรมบนวินโดว์สหรือบนกราฟิกยูสเซอร์อินเทอร์เฟซด้วย ซึ่งเป็นสิ่งที่นักโปรแกรมเมอร์ที่เคยเชี่ยวชาญบนเมนเฟรมอาจไม่คุ้นเคยเท่าใดนักกับการเขียนโปรแกรมแบบนี้

เนื่องจากระบบ ไคล์แอนด์-เซอร์เวอร์ เป็นระบบที่ค่อนข้างใหม่และยังอยู่ในขั้นตอนของการปรับปรุงพัฒนา . จึงทำให้ในช่วงนี้จะมีปัญหามากพอสมควร ควรอย่างยิ่งที่จะให้ผู้รับติดตั้งระบบฯ รับภาระการบำรุงรักษาระบบ ไคล์แอนด์-เซอร์เวอร์ ให้ตลอด

3.4 ความสำคัญของเครื่องมือพัฒนาซอฟต์แวร์ (Software Tool) ในระบบไคล์แอนด์

ในอดีตบริษัทที่ผลิตฐานข้อมูลมักจะละเลยเครื่องมือพัฒนาซอฟต์แวร์สำหรับบำรุงรักษาระบบฯ ปัจจุบันซอฟต์แวร์ประเภทนี้กลับได้รับการสนใจอย่างมาก

หันมาดูซอฟต์แวร์ที่สำคัญในระบบไคล์แอนด์-เซอร์เวอร์ อีกชนิดหนึ่งที่มีความสำคัญมากก็คือซอฟต์แวร์ประเภทเฝ้าดูระบบเครือข่าย (Monitor Network) ซอฟต์แวร์ประเภทนี้จะมีส่วนช่วยผู้จัดการเครือข่ายให้สามารถบริหารงานเครือข่ายได้อย่างดี โดยเฉพาะในด้านการบำรุงรักษาตัวระบบให้มีความเชื่อมั่นในระบบสูงขึ้น

3.5 บทบาทของมัลติมีเดียกับไคล์แอนด์-เซอร์เวอร์

ในอนาคตอันใกล้ซอฟต์แวร์ที่จะเป็นตัวเด่นในระบบไคล์แอนด์-เซอร์เวอร์อีกตัวหนึ่งคือซอฟต์แวร์ประเภทมัลติมีเดียซึ่งในขณะนี้กำลังเร่งการพัฒนาอย่างเต็มที่ เราคงมีโอกาสได้เห็นภาพและเสียงในมัลติมีเดีย จะทำให้มีการพัฒนาโปรแกรมประยุกต์ประเภทออนไลน์ช็อฟปิง และออนไลน์แบ็งค์ออกมาใช้กันอย่างกว้างขวาง ตัวบริษัทพีบีเอสเองก็มีแผนจะทำภาพยนตร์อิเล็กทรอนิกส์ สำหรับนักพิซีให้สามารถเรียกได้จากจอพีซีโดยไม่ต้องใช้จอโทรทัศน์แบบแต่ก่อนอีก

ส่วนบริษัทเบอริงตันเองนั้น คิดจะจัดทำซอฟต์แวร์ประเภทการประชุมทางคอมพิวเตอร์หรือที่เรียกว่า Videoconference โดยจะมีทั้งภาพและเสียงที่สมบูรณ์แบบ ซึ่งจะให้ขีดความสามารถที่เหนือกว่าการประชุมภาพทางโทรทัศน์เพราะว่าการจัดประชุมทางคอมพิวเตอร์นอกจากจะมีขีดความสามารถประเภทภาพและเสียงแล้ว ยังสามารถใช้คอมพิวเตอร์และเปลี่ยนข้อมูลได้ด้วย

อย่างไรดี คาดกันว่าก่อนที่มัลติมีเดียจะเข้ามามีบทบาทในไคล์แอนด์-เซอร์เวอร์นั้น คงจะต้องมีการพัฒนาองค์ประกอบในด้านต่างๆต่อไปนี้ก่อนคือโครงข่ายสื่อสารตัวฮาร์ดแวร์คอมพิวเตอร์ และตัวซอฟต์แวร์มัลติมีเดียเอง

ในด้านโครงข่ายสื่อสารนั้น จะต้องสามารถส่งข้อมูลความเร็วสูงในขนาดไม่ต่ำว่า 720 กิโลบิตต่อวินาที หรือประมาณ 60x12 กิโลบิตเฟรมต่อวินาทีจึงจะให้ภาพเสียงและข้อมูลที่เพียงพอสำหรับซอฟต์แวร์ประเภทมัลติมีเดีย ในด้านซอฟต์แวร์นั้น จะต้องมีการพัฒนาในด้านระบบฐานข้อมูลมัลติมีเดียให้มีขีดความสามารถสูงกว่าที่เป็นอยู่ในปัจจุบันที่รับมือได้ทั้งภาพและเสียง เช่น จะต้องสามารถค้นหาภาพและเสียงคำนวณที่ป้อนเข้าในรูปของภาพและเสียง

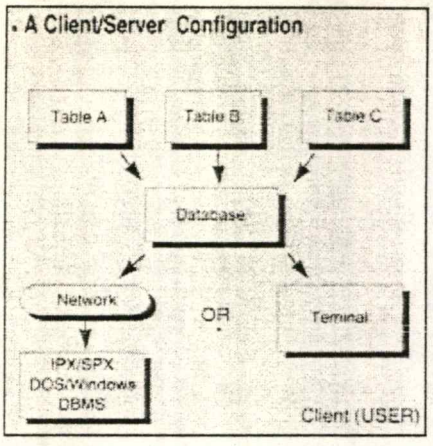
เมื่อเข้าสู่ยุคของมัลติมีเดียและสถาปัตยกรรมทางข้อมูลเดิมแบบฐานข้อมูลเชิงสัมพันธ์ (Relational Database) คงจะต้องเปลี่ยนไปเพราะว่าพื้นฐานของฐานข้อมูลเชิงสัมพันธ์รับมือได้ดีเฉพาะข้อมูลประเภทสองมิติเท่านั้น แต่ในยุคของมัลติมีเดียแล้วข้อมูลมากกว่าสองมิติ ก็จะต้องรับมือได้ด้วย ซึ่งก็คงจะต้องดูกันต่อไปว่าสถาปัตยกรรมทางข้อมูลในรูปใดจึงจะเหมาะสมกับมัลติมีเดีย

ถึงแม้ว่าลักษณะของสถาปัตยกรรมของข้อมูลในอนาคต จะยังมองไม่เห็นรูปร่างว่าจะออกมาในรูปใด แต่ก็พอจะมองเห็นว่าซอฟต์แวร์ประเภทเชิงวัตถุ (Object-oriented) จะต้องเข้ามามีส่วนในการพัฒนาระบบข้อมูลอย่างแน่นอนตัวอย่างที่พอจะมองเห็นได้ในขณะนี้ก็คือ ภาษา SQL++ ที่ได้นำฐานข้อมูลเชิงสัมพันธ์ มาใช้งานร่วมกับฐานข้อมูลเชิงวัตถุ (Object-oriented Database) ซึ่งทำให้ผู้ใช้ระบบสามารถเรียกหรือถามหาข้อมูลประเภทรูปภาพและเสียงได้อย่างสะดวก และใช้คำถามที่สั้นกะทัดรัดไม่ยุ่งยาก ผู้ใช้เพียงแต่ใช้ภาษาพูดตามปกติก็จะสามารถสอบถามหาข้อมูลอันใกล้นี้อาจจะไม่ต้องกดคีย์บอร์ดด้วยซ้ำเพียงใช้เสียงก็อาจสอบถามข้อมูลจากข้อมูลได้ทันที ซึ่งจะทำให้สามารถเรียกใช้ฐานข้อมูลได้จากโทรศัพท์ธรรมดา หรือโทรศัพท์เคลื่อนที่ก็ได้ อันจะทำให้ผู้ใช้ระบบไคล์แอนด์-เซอร์เวอร์ ไม่ว่าอยู่ ณ ที่ใดก็สามารถล้วงลึกหาข้อมูลได้อย่างสะดวก

3.6 การประยุกต์ใช้งานระบบไคล์แอนด์เซอร์เวอร์กับระบบฐานข้อมูล

3.6.1 ไคล์แอนด์เซอร์เวอร์และการรวมกันในระบบฐานข้อมูลแบบกระจาย

หลายคนมีความสงสัยว่าสถาปัตยกรรมแบบไคล์แอนด์เซอร์เวอร์นั้น นำไปรวมในระบบฐานข้อมูลแบบกระจายได้อย่างไรในระบบฐานข้อมูลแบบศูนย์กลางนั้นความสัมพันธ์ระหว่างผู้ใช้ (ไม่ว่าผู้ใช้เหล่านั้นจะติดต่อโดยเทอร์มินอลหรือพีซีติดต่อผ่านระบบ LAN ก็ตาม) และฐานข้อมูลที่เป็นศูนย์กลางนั้นสามารถแสดงลักษณะของไคล์แอนด์เซอร์เวอร์ก็ได้ โดยไคล์แอนด์เทอร์มินอลหรือพีซีจะร้องขอการบริการและข้อมูลจากเซอร์เวอร์ที่เป็นศูนย์กลาง ในกรณีที่ความสัมพันธ์เหล่านี้อยู่ในระบบกระจาย ระบบอาจจะทำหน้าที่เสมือนกับเป็นเซอร์เวอร์ในงานงานหนึ่ง ในขณะเดียวกันจะเป็นไคล์แอนด์สำหรับงานอื่น ความสัมพันธ์ในระบบแบบกระจายนี้ทั้งไคลเอนด์และเซอร์เวอร์ รวมทั้งส่วนที่เป็นฮาร์ดแวร์และซอฟต์แวร์ จะแชร์ข้อมูลและแบ่งความรับผิดชอบในการประมวลผลระบบฐานข้อมูลกัน ผู้พัฒนาระบบไคลเอ็นด์เซอร์เวอร์ยุคใหม่นั้นได้มองทางที่จะขยายรูปแบบไคลแอนด์-ซอร์เวอร์แบบกระจายเดิมเพื่อรวมพีซีไคล์แอนด์เข้าไปในระบบเครือข่าย หรืออีกนัยหนึ่งก็คือพวกเขาต้องการที่จะเปลี่ยนความสัมพันธ์แบบเดิมให้เป็นความสัมพันธ์แบบกระจายอย่างเต็มรูปแบบ ซึ่งมีข้อได้เปรียบค่อนข้างมาก โดยที่ระบบพีซีไคล์แอนด์-ซอร์เวอร์แบบกระจายจะสามารถรักษาความปลอดภัยและกำลังความสามารถของ back-end DBMS ได้ และในขณะเดียวกันก็ใช้พีซีเป็น front-end ผู้ใช้จะใช้ข้อมูลในท้องถิ่นโดยผ่านตัวขับเคลื่อน DBMS (DBMS engine)



ระบบฐานข้อมูลแบบกระจายที่ใช้พีซีนั้นก็เหมือนกับระบบแบบกระจายทั่ว ๆ ไป คือต้องอยู่ในความควบคุมของตัวจัดการการทำรายการ เครื่องพีซีของผลิตภัณฑ์ประเภท back-end (เช่น Oracle SQL Server ของ Sybase และ SQLBase Server ของ Gupta) นั้นใช้งานค่อนข้างยากและมีประสิทธิภาพค่อนข้างต่ำ ในประแกรมจัดการฐานข้อมูลที่ใช้ในพีซี (เช่น Microsoft FoxBase หรือ Borland's Paradox) ไม่ได้รวมตัวจัดการทำรายการเข้าไว้เป็นส่วนหนึ่งของระบบของฐานข้อมูลแบบกระจายด้วย

ในขณะที่ระบบฐานข้อมูลที่ใช้พีซีและผลิตภัณฑ์ที่พัฒนาฐานข้อมูลจำนวนมากได้รวม Tools เพื่อแน่ใจเกี่ยวกับความถูกต้องและความปลอดภัยของข้อมูลทางด้านแผนก MIS ส่วนใหญ่กลับหยุดที่ความคิดที่วางข้อมูลที่ต้องใช้ฐานข้อมูลที่เกี่ยวกับการปฏิบัติงาน (Critcal mission data) ไว้ในมือของผู้ใช้

ขณะที่อุตสาหกรรมได้แก้ปัญหาของตำแหน่งของข้อมูลและการจัดการทำรายการการประมวลผลพีซีไคล์แอนด์เซอร์เวอร์แบบกระจายก็จะเข้ามามีบทบาทด้วยเช่นกัน

DISTRIBUTED TRANSACTION MANAGEMENT

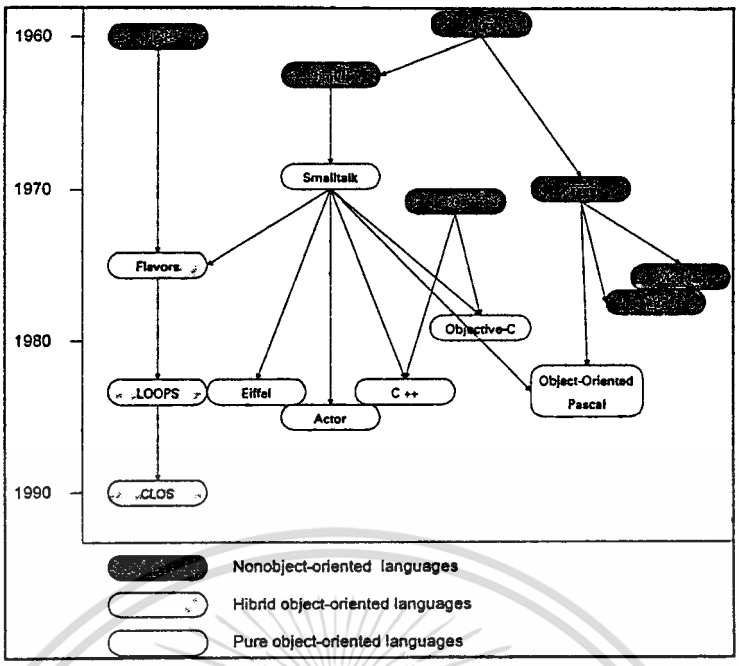
DBMS	DATA LOCATION	METHOD
Cincom Supra Server	Distributed master catalog (through DRDM)	Transparent read/write (through DRDM); transaction partnering; transparent inter-vendor read/write via DRDM-supported drivers
Sybase SQL Server 10 (through OmniSQL Gateway)	Central catalog	Transparent read/write and store-and-forward (both through Replication Server); intervendor read/write through gateways or ODBC
Gupta SQLBase Server 5.1*	Not applicable	Not applicable
Oracle 7	Central directory catalog; local data catalogs	Transparent read/write; store-and-forward; intervendor read/write through third-party, X/Open XA-compliant distributed transaction processor

* Gupta plans to implement distributed transaction management in SQLBase Server 6.0, due this summer.

ของ Whitewater Group และ Eiffel ของ Interactive Software Engineering Inc. ภาษาในกลุ่มนี้จะมีหน่วยของภาษาเป็น Object ทั้งหมดไม่ว่าจะเป็น Statement ของภาษาเช่น IF-ELSE หรือ DO-WHILE การดำเนินการของโปรแกรมจะใช้วิธีส่งแมสเสจแต่เพียงอย่างเดียว ภาษาในอีกกลุ่มหนึ่งคือกลุ่มภาษา OOP ได้แก่ C++, Objective-C, Object-BASIC, Object-COBOL, Common Lisp Object System (CLOS) และ Object-Oriented PASCAL กลุ่มนี้จะมีลักษณะทั่วไปเป็นภาษาแบบ Procedural ธรรมดาแต่มีความสามารถของภาษา OOP รวม และส่วนใหญ่จะเป็น superset ของภาษาเดิม

ในขณะที่ภาษาในกลุ่มแรกมีความเป็น OOP มากกว่าในด้านของภาษา โครงสร้างทางภาษาเป็น Object ทั้งหมดและสร้างงานได้โดยการส่งแมสเสจไปยัง Object ที่ต้องการซึ่งส่วนใหญ่มีเก็บไว้ในคลาสไลบรารีของภาษาหมดแล้วจึงทำให้พัฒนาโปรแกรมได้ง่ายและรวดเร็ว อีกทั้งยังใช้เป็นภาษาฝึกสอนได้เป็นอย่างดี ในขณะที่ความนิยมของภาษาคอมพิวเตอร์ยังตกอยู่กับ C หรือ PASCAL เป็นหลัก โปรแกรมจำนวนไม่น้อยที่พัฒนาขึ้นมาโดยภาษาเหล่านี้ความเป็นคอมแพทิเบิลของโค้ดจึงเป็นเรื่องที่สำคัญ ภาษาในกลุ่มไฮบริดได้รับความนิยมในทางปฏิบัติอย่างสูงเพราะสามารถโค้ดได้ ไม่ทิ้งจากภาษาเดิมเท่าใดนัก อีกทั้งยังเป็น superset กับ ภาษาเดิม ทำให้สามารถนำโค้ดใหม่ลิงค์กับโค้ดเก่าอย่างไม่มีปัญหาทางผู้พัฒนาและผลิตคอมไพเลอร์ก็มีความสุขเพราะเท่ากับได้กลุ่มลูกค้าเพิ่มขึ้นดังนั้นในทางปฏิบัติภาษากลุ่มไฮบริดจึงทำงานได้ดีกว่าอย่างไม่ต้องสงสัย

ภาษากลุ่มไฮบริดที่ได้รับความนิยมสูงสุดคงหนีไม่พ้น C++ และ Object-Oriented PASCAL C++ นั้นดังเพราะบารมีที่ภาษา C สั่งสมไว้ตั้งแต่ยุคปี 1980 ในฐานะที่เป็นภาษาแรกในประวัติการณ์ที่พูดได้เต็มปากว่ามีความ Portability สูงมากเพราะสามารถทำงานได้แทบทุกแพลตฟอร์ม การสร้างส่วนเพิ่มแบบ OOP ในภาษาที่ผู้พัฒนา Bjarne Stroustrup (นักวิจัยของเบลล์แลป รุ่นเดียวกับ Dennis Richie ผู้สร้างภาษา C) คิดค้นขึ้นจึงเป็นการพิ่มคุณค่าในตัวภาษา C มากขึ้นและยิ่งตอกย้ำถึงคำว่าคอมแพทิเบิลที่สามารถนำโค้ดจาก C++ ไปลิงค์รวมกับโค้ดเดิมที่สร้างด้วย C อย่างไม่มีปัญหา ปัจจุบันนี้ภาษา C++ มีคอมไพล์เลอร์อยู่ทั้งในเครื่องระดับ midi computer, workstation, PS/2 หรือ Microcomputer ทั่วไป สรุปก็คือ C++ หรือร่างแปลงอีกร่างหนึ่งของภาษา C ยังคงเป็นภาษา OOP อันดับหนึ่งในปัจจุบัน



ภาพ 1 แสดงพัฒนาการของภาษาแบบ OOP ในยุคสมัยต่าง ๆ

มาถึงภาษา Object-Oriented PASCAL อีกรีลีสหนึ่งของ PASCAL ที่ถูกพัฒนาโดยกลุ่มผู้ประกอบการทาง Software Apple Computer เป็นกลุ่มแรกที่พัฒนาส่วน OOP ให้แก่ภาษา PASCAL เอาไว้ใช้ในเครื่อง Macintosh ถัดจากนั้นทางกลุ่ม Microsoft และ Borland จึงเอาอย่างโดย Borland ออก Turbo PASCAL Version 5.5 มาชิมลางดูก่อนที่ Microsoft จะออก Microsoft Quick PASCAL ตามมาแย่งส่วนแบ่งตลาด ว่าไปแล้วส่วน OOP ที่อยู่ในภาษา Object-Oriented PASCAL นั้นไม่ใคร่จะสมบูรณ์เท่าใดนักเช่น มี Default ของตัวแปรในคลาสเป็น Public ซึ่งขัดแย่งกับหลักการ Encapsulation อย่างรุนแรงและไม่มีคุณสมบัติการถ่ายทอดแบบ Multiple Inheritance แต่ถึงอย่างไรด้วยเครดิตของ PASCAL ทำให้ภาษานี้เป็นภาษาไฮบริดอีกภาษาหนึ่งที่ประสพความสำเร็จอย่างสูง

ในส่วนของวงการ AI พัฒนาการของภาษา LISP ก็ส่งผลให้เกิดภาษา OOP แบบไฮบริดขึ้นอีกหลายภาษาอันได้แก่ Flavors ซึ่งพัฒนาโดยมหาวิทยาลัย MIT, LOOPS พัฒนาโดย Xerox PARC ที่เดียวกับที่กำเนิด Smalltalk และ CLOS หรือ Common Lisp Object System ที่พัฒนาโดยกลุ่มสมาคมผู้วิจัยเทคโนโลยีปัญญาประดิษฐ์

พัฒนาการของภาษายังมีต่อไปเรื่อย ๆ แต่ในปัจจุบันมีแนวโน้มจะเกิดภาษาใช้งานเฉพาะทางมากกว่าภาษาที่ใช้ในการเขียนโปรแกรม ภาษาดังกล่าวนี้ก็เช่น Hyper-Card ของ Apple Computer หรือพวกภาษาไฮบริดเฉพาะทางอย่าง Object-Prolog ที่ใช้ในงานปัญญาประดิษฐ์หรือพวกภาษา Script ต่าง ๆ อีกมากมาย

4.2 เทคโนโลยีเชิงวัตถุ(Object-Oriented Technology)

ในวงการคอมพิวเตอร์ในปัจจุบันน้อยคนนักที่จะกล่าวว่าไม่เคยได้ยินหรือรู้จักเรื่องราวของ Object-Oriented มาก่อนแต่ภาพพจน์ที่เรามีให้แก่ Object-Oriented นั้นถูกมองควบคู่ไปกับคำว่า Programming เสมอ OOP นั้นเรียกได้ว่าเป็นแนวความคิดใหม่ในความพยายามที่จะจัดระบบกระบวนการพัฒนาโปรแกรมให้มีระเบียบ และสามารถนำโปรแกรมที่เคยเขียนมาก่อนแล้วกลับมาใช้ใหม่ได้ เหมือนกับเรื่องที่กำลังมีการรณรงค์ให้แยกขยะเปียกออกจากขยะเส้นใย เพื่อหวังจะนำเส้นใยเหล่านั้นกลับมาใช้ใหม่ที่เรียกว่า Reusable หรือการนำกลับมาใช้ใหม่

แต่เป็นที่ทราบกันดีอยู่แล้วว่า เรื่องอะไรก็ตามที่ยากเกินจะอาศัยสามัญสำนึกของมนุษย์ในการพิจารณาเรียนรู้นั้นมนุษย์มักจะเบื่อหน่ายง่าย เข้าทำนองไม่ค่อยชอบเรื่องที่ต้องอาศัยเวลาในการทำความเข้าใจทั้ง ๆ ที่ความจริงแล้วก็ไม่ได้เป็นเรื่องยากเกินกว่าจะขบให้แตก เพราะคนคิดหลักการหรือเรื่องเหล่านี้เองก็เป็นคนเดินดินกินข้าวแกงเหมือนคนทั่ว ๆ ไป

การเขียนโปรแกรมถือเป็นเรื่องไกลตัวเกินกว่าใครที่ไม่เกี่ยวข้อง จะยอมลงมาศึกษาอย่างจริงจัง ศาสตร์ทางด้าน Programming จึงจำกัดอยู่ในวงใช้งานเท่านั้น ที่นี้พอกล่าวถึงแนวความคิดใหม่ที่เรียกกันว่า Object-Oriented ที่พาดพิงไปถึงเรื่องของโปรแกรมมิ่งเข้าให้ OOP เลยเป็นสิ่งมหัศจรรย์ไป พอเป็นสิ่งมหัศจรรย์ที่มีคนรู้น้อยเลยเกิดเป็น ภาวะเหนือมนุษย์ มีการอภิปรายอย่างกว่างขวางแต่ผิดประเด็น ทำให้คนที่ยังไม่เข้าใจยิ่งสับสนใหญ่ คนที่มีความรู้ในเรื่อง OOP กลายเป็นพวกนอกรีดไป

เมื่อ OOP กลายเป็นสิ่งเหนือมนุษย์ไปแล้ว หลายคนเกิดความกลัวเมื่อไปพบกับคำว่า Object-Oriented เพราะฝังหัวเอาไว้ก่อนว่ามันจะต้องเป็นอะไรที่ซับซ้อนและวุ่นวายมากแน่นอน แต่ในความเป็นจริงแล้วหลักการของ Object-Oriented ถูกนำมาใช้อย่างแพร่หลายมากในปัจจุบันรอบ ๆ ตัวเรามีผลิตภัณฑ์ที่เป็น Object-Oriented เต็มไปหมดอยู่ที่ว่าจะสังเกตหรือไม่เท่านั้นจุดสังเกตที่ง่ายที่สุดคือเรื่องของระบบติดต่อผู้ใช้แบบกราฟิกหรือ GUI ที่กำลังเป็นที่นิยมอยู่ในขณะนี้ อย่างใน วินโดว์ส นี่ก็ใช่ OS/2 ก็ใช่ Apple หรือ Macintosh และซอฟต์แวร์แบบ GUI ใช่ทั้งนั้น

มามองให้ไกลตัวอีกหน่อยเป็นเรื่องของระบบฐานข้อมูลนี่ก็ได้มีการบัญญัติโมเดลฐานข้อมูลแบบใหม่ที่เรียกว่า Object model ขึ้นทำให้เกิด DBMS แบบ Object-Oriented ขึ้นมาอีกหรือว่าจะเป็นเรื่องของภาษาคอมพิวเตอร์ที่อ้างอิงหลักการ Object-Oriented ที่เรียกว่าภาษา OOP ก็มีหลายภาษาได้แก Smalltalk , Eiffel , C++ ,Object-PASCAL หรือ Object-BASIC นี่ก็ได้ข่าวว่าแม้แต่ในวงการปัญญาประดิษฐ์ก็ได้สร้าง Object-Prolog ขึ้นมาแล้ว

4.2.1 หลักการกว้าง ๆ

จากชื่อ Object-Oriented ซึ่งมีผู้พยายามจะแปลว่า "...เชิงวัตถุ" ฟังดูแล้วใจหนึ่งก็ว่าเข้าท่าดีแต่อีกใจหนึ่งก็รู้สึกขัด ๆ หูพิกล ความรู้สึกที่ฟังแล้วขัด ๆ หู น่าจะเกิดจากความไม่เคยชินเสียมากกว่า จุดหักของความชินไม่ชินที่ว่านี่เกิดขึ้นเพราะว่าในช่วงเวลาแรก ๆ ที่ประเทศไทยรับเอาเทคโนโลยีคอมพิวเตอร์จากต่างชาติมานั้น เราขาดผู้แปลชื่อทับศัพท์ หากเรามีคณะกรรมาธิการพิจารณาศัพท์คอมพิวเตอร์แต่แรก ๆ นั้น ความขัดหูของชื่อเฉพาะเหล่านี้จะไม่ปรากฏเด็ดขาด ยกตัวอย่างที่เห็นได้ชัดคือ คำว่า ฝรั่งเศส อังกฤษ หรือคำว่าราชปะแตน คำเหล่านี้เป็นชื่อเฉพาะที่มาจากภาษาต่างประเทศทั้งนั้นหากแต่ในสมัยนั้นได้มีการบันทึกและเผยแพร่ในวงกว้างอย่างจริงจังจากความไม่เคยชินเลยกลายเป็นชื่อเฉพาะไป มาในสมัยนี้หากหนังสือหรือนิตยสารทุกเล่มพร้อมใจกันใช้คำว่า "คณิตกรณ์" แทนคำว่า "คอมพิวเตอร์" หมดตั้งแต่แรก ปัจจุบันนี้คำที่ฟังดูขัดหูอาจจะเป็น "คอมพิวเตอร์" ก็ได้ใครจะรู้

หลักการของ Object-Oriented อยู่ที่คำสองคำเท่านั้น คือคำว่า ออบเจกต์ กับ การถ่ายทอด ออบเจกต์คืออะไร ? ออบเจกต์คือหน่วยสนใจของระบบที่เป็น Object-Oriented ระบบแบบนี้เน้นที่ตัวปฏิบัติการ มากว่า การปฏิบัติ เช่นเรื่องของไดรว์เราถือว่า ตัวปฏิบัติการ คือตัวไดรว์ แต่การปฏิบัติคือฟังก์ชันของไดรว์เช่น อ่าน เขียน หรือ ฟอร์แมต ตัวอย่างของระบบแบบ Object-Oriented ได้แก่เชลล์ของ OS/2 และระบบที่ไม่ใช่ได้แก่เชลล์ของ DOS ในเชลล์ของ DOS หรือ COMMAND.COM นั้น การที่เราจะสั่งทำการก็อปปี้ไฟล์สักไฟล์หนึ่งจากไดรว์ A ไปยัง B เราจะต้องพิมพ์คำสั่ง

C:> COPY A:FILE1.TXT B: <ENTER>

ในขณะที่เชลล์ของ OS/2 นั้นใช้วิธีคลิกเมาท์เลือกไฟล์ในเมนูลากไอคอนของไฟล์นั้นไปดรอปลงที่ไอคอนรูปไดรว์ของไดรว์ B โปรดสังเกตความแตกต่าง ในตัวอย่างนั้น เนื่องจากระบบ Object-Oriented คำนึงถึงตัวปฏิบัติการอันได้แก่ 1. ไฟล์ 2. ไดรว์ เป็นหลักปัญหาอันเนื่องจากปฏิบัติการผิดพลาดจึงมีโอกาสเกิดขึ้นน้อยกว่า เมื่อเทียบกับความผิดพลาดอันเนื่องมาจากการคีย์คำสั่งไม่ครบหรือคีย์ผิด ที่กล่าวมานี้ไม่ได้เป็นการโจมตีจุดด้อยของระบบแบบเดิม แต่เป็นการแสดงให้เห็นมุมมองที่แตกต่างของระบบทั้งสองแบบ การคำนึงถึงตัวปฏิบัติการเป็นหลัก น่าจะลดความสับสนของของระบบไปได้มากกว่า ทั้งนี้เพราะว่าในปฏิบัติการอยู่นั้น เราเห็นความสัมพันธ์ระหว่างตัวปฏิบัติการต่าง ๆ ควบคู่ไปด้วย เมื่อเราเข้าใจความสัมพันธ์มากขึ้นเราก็สามารถใช้งานตัวปฏิบัติการเหล่านั้นได้อย่างมีประสิทธิภาพ

ระบบแบบ Object-Oriented จะเน้นที่ความเกี่ยวพันของออบเจกต์แต่ละตัวในระบบเป็นหลัก โดยดูปฏิบัติการเป็นประเด็นรองลงมา ปฏิบัติการที่ว่านี้เป็นการกระทำที่เกิดขึ้นโดยออบเจกต์หนึ่งกระทำกับออบเจกต์อีกตัวหนึ่ง เช่น มีงูตัวหนึ่ง มีไก่อีกตัวหนึ่ง งูกินไก่ แต่ไก่จะกินงูไม่ได้เพราะไก่ไม่อนุญาตให้งูกิน ออบเจกต์ "งู" สามารถกระทำการ "กิน" ออบเจกต์ "ไก่" ได้ในขณะที่

ทำไม่ได้ในทิศทางตรงกันข้ามหันมาดูระบบแบบเดิมที่เน้นปฏิบัติการเป็นหลัก ในระบบแบบนี้ไก่สามารถกินงูได้ เพราะว่าในระบบแบบนี้ไม่เคยมีการนิยามสัตว์สองประเภทนี้เลย "ไก่" และ "งู" มีภาพพจน์เป็นแค่ประธานและกรรมในประโยคเท่านั้น สรุปได้ว่า ระบบแบบ Object-Oriented นั้นมีการนิยามตัวละครในระบบ แต่ระบบแบบเดิมไม่มี

ในการนิยามลักษณะและคุณสมบัติของออบเจกต์ต่าง ๆ ในระบบนั้น สามารถทำได้ทั้งทางตรงคือใส่ข้อกำหนดไปทันที หรือใช้วิธีที่เรียกว่า "สืบสายเลือด" คืออาศัยลักษณะของออบเจกต์ที่มีอยู่แล้ว ใส่ลงในออบเจกต์ตัวใหม่ ตัวอย่างเช่น ออบเจกต์ "สัตว์" มีคุณสมบัติอย่างไรได้กำหนดเอาไว้แล้ว เมื่อต้องการกำหนดคุณสมบัติของ "สุนัข" ก็ใช้วิธีถ่ายทอดคุณสมบัติของ "สัตว์" ที่กำหนดไว้ก่อนหน้านี้แล้ว ด้วยวิธีนี้ทำให้ประหยัดเวลาและถือเป็นสิ่งที่ขาดไม่ได้ในระบบแบบนี้ทีเดียว เพราะสรรพสิ่งทั้งหลายในโลกนี้ไม่เคยเกิดมาเดี่ยว ๆ โดยไม่มีสายสกุลและผู้สืบสกุล คนก็เป็นสายสกุลของสัตว์ สัตว์ก็เป็นสายสกุลของสิ่งมีชีวิต แม้แต่โลกถ้าพิจารณาให้ดีก็มีสายสกุลเป็นดาวพระเคราะห์อยู่ เมื่อมีการสืบสายคุณสมบัติของออบเจกต์มาเป็นลำดับชั้นดังนี้แล้ว ความสัมพันธ์ระหว่างออบเจกต์จะชัดเจนยิ่งขึ้นยิ่งความสัมพันธ์ชัดมากขึ้น ก็มีผลให้การออกแบบระบบง่ายขึ้น และจะได้ระบบที่ซับซ้อนมากขึ้นเรื่อย ๆ ผู้ออกแบบสามารถออกแบบระบบขนาดใหญ่ ๆ ได้เพียงชั่วข้ามคืนโดยอาศัยออบเจกต์ที่นิยามไว้ก่อนหน้านี้แล้วโดยนักออกแบบรุ่นพี่ ลักษณะเช่นนี้จะนำไปสู่การนำกลับมาใช้ใหม่หรือ Reusable สิ่งที่ได้รับการออกแบบไว้ดีแล้วก่อนนี้จะไม่สูญหาย แต่จะเป็นแบ็คกราวน์สำหรับงานใหม่ต่อไปเรื่อย ๆ ไม่มีสิ้นสุด

4.2.2 โลกของเราก็เป็น Object-Oriented

ในโลกของเรานี่เป็นสังคมขนาดใหญ่ที่มีมนุษย์เป็นสมาชิกในสังคม คนแต่ละคนในสังคมก็มีบทบาทแตกต่างกันไป บางคนเป็นหมอ วิศวกร หรือผู้ร้ายฆ่าชิงทรัพย์จึงกล่าวได้ว่าในสังคม ๆ หนึ่งมีบุคคลอยู่มากมายหลายประเภท แต่ละคนก็มีความสัมพันธ์ซึ่งกันและกันเป็นโครงข่ายที่ซับซ้อน กิจกรรมใดๆ ในสังคมจะสามารถดำเนินไปได้ก็ด้วยการติดต่อสื่อสารระหว่างคนในสังคม เมื่อคน ๆ หนึ่งมีความต้องการในเรื่องอะไรก็ตามสักอย่างหนึ่งหากเป็นเรื่องที่เขาไม่สามารถหาคำตอบด้วยตัวเองได้ เขาจะต้องค้นหาคำตอบเอาจากคนรอบข้าง ในลักษณะความสัมพันธ์แบบถาม-ตอบ ยกตัวอย่างเช่น

"นาย ก. เป็นสถาปนิกแต่เขาขาหัก เขาไม่สามารถรักษาขาของเขาเองได้ เขาจึงต้องไปหานาย ข. ที่เป็นแพทย์ เพื่อขอให้นาย ข. ช่วยรักษาขาของเขา ในทางกลับกันหากนาย ข. ต้องการสร้างบ้านเขาคงต้องปรึกษานาย ก. เช่นกัน"

ในตัวอย่างนี้เราจะเห็นว่าทั้งนาย ก. และนาย ข. ต่างก็มีบทบาทและสถานภาพเป็นของตัวเองบทบาทของ นาย ก. คือ สถาปนิก สถานภาพก็เช่น โสด เพศชาย เป็นต้น นาย ก. อาศัยบท

บทที่ 4
เทคโนโลยีเชิงวัตถุ(Object-Oriented Technology)

4.1 ตำนานภาษาเชิงวัตถุ(Object-Oriented)

ต้นตอจริง ๆ ของภาษาที่มีแนวคิดแบบ Object-Oriented หรือที่เรียกกันว่า OOP นั้น เริ่มในช่วงปี 1950-1960 โดยภาษา LISP (ย่อมาจากคำว่า List Processing) ซึ่งเป็นภาษาที่ใช้ในงานปัญญาประดิษฐ์หรือ AI (Artificial Intelligent) เนื่องมาจาก LISP เป็นภาษาที่เริ่มรู้จักคำว่า Dynamic Binding และเห็นความสำคัญของการติดต่อผู้ใช้แบบกราฟิกที่พัฒนามาจนเป็น GUI ในปัจจุบัน ถัดจาก LIST ในราวปี 1960 หลักการของคลาสและการถ่ายทอดคุณสมบัติของคลาส(Inheritance) ถูกบัญญัติขึ้นในภาษาที่มีชื่อว่า Simula ซึ่งเป็นภาษาที่เป็นการพัฒนาโปรแกรม Simulation ที่ใช้ในงานต่าง ๆ Simula ทำให้เกิดภาษาใหม่ขึ้นมาอีก 2 ภาษาคือ Ada และ Modula-2 ภาษาทั้ง 2 ประสบความสำเร็จทางธุรกิจอย่างมาก และเป็นภาษาแรกที่มีชนิดข้อมูลที่เรียกว่า ชนิดข้อมูลแอ็บสแตรค (Abstract data type) ซึ่งยินยอมให้มีโค้ดรวมอยู่กับชนิดข้อมูลได้

ในช่วงปี 1970 ภาษา Smalltalk ก็เกิดจากผลงานวิจัยของสำนักวิจัย Xerox Palo Alto Research Center (PARC) โดยผู้คิดค้นที่มีชื่อว่า Alan Kay ภาษา Smalltalk ดึงเอาหลักการเกี่ยวกับเรื่องคลาสและการถ่ายทอดคุณสมบัติมาจากภาษา Simula-67 (ซึ่งเป็นรีลีส หนึ่งของ simaula) รวมเข้ากับระบบ การติดต่อผู้ใช้แบบกราฟิกสมบูรณ์แบบ Kay ออกแรงเข็นจนกระทั่ง Smalltalk ออกเป็นผลิตภัณฑ์รีลีสแรกมีชื่อว่า Smalltalk–80 ในปี 1981 จวบจนปัจจุบัน Digitalk ก็ได้พัฒนา Smalltalk รีลีสใหม่ออกมาอีกโดยมีชื่อว่า Smalltalk/V ตามมาด้วย Smalltalk/V for Windows ที่ทำงานภายใต้ Microsoft Windows จนกล่าวได้ว่าภาษา Smalltalk เป็นภาษาแรกที่มีลักษณะของ OOP (Object Oriented Programming) สมบูรณ์แบบที่สุด

ในขณะที่ Smalltalk เริ่มเป็นที่รู้จักนั้นเป็นช่วงเดียวกับภาษายอดนิยมอย่าง BASICS, C, PASCAL และ COBOL เริ่มเป็นที่นิยมแสดงให้เห็นว่าแนวความคิดแบบ OOP ไม่ใช่เรื่องแปลกใหม่แต่อย่างใด Procedural Programming มีขุนผลคือ C และ PASCAL เป็นหลัก ในขณะที่ฝั่ง OOP มี Smalltalk และ Simula เมื่อกระแส OOP เริ่มแผ่ขยายมากขึ้นทางกลุ่มที่พัฒนาและบริษัทผลิตซอฟต์แวร์ก็เริ่มตื่นตัวกับแนวความคิดนี้มากขึ้น เป็นผลให้เกิดภาษาแบบ OOP ในยุคที่สองขึ้น ภาษา OOP ในยุคที่สองนี้เป็นการนำเอาคุณสมบัติของ OOP ผนวกกับไวยากรณ์เดิมของภาษา Procedural เรียกว่าภาษา OOP แบบไฮบริด (Hybrid group of OOP language)

วงการคอมพิวเตอร์จึงปรากฏภาษา OOP เป็น 2 กลุ่ม กลุ่มที่ 1 คือ กลุ่ม OOP เพียว เรียกว่า Pure OOP group language ภาษาในกลุ่มนี้ได้แก่ Smalltalk, Actor ที่พัฒนามาจาก Smalltalk

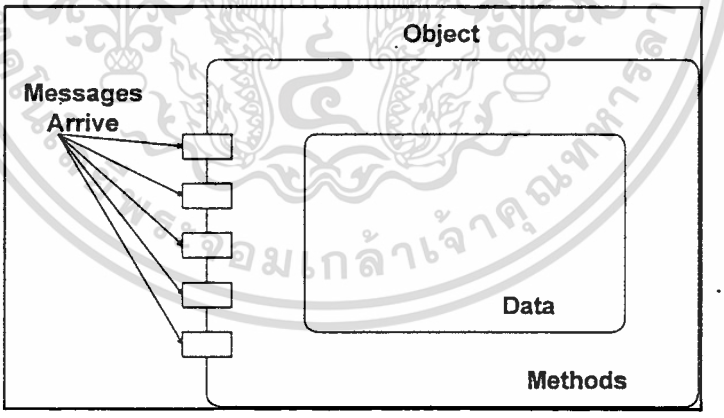
บาทที่มีอยู่คือสถาปนิก ในการตอบสนองต่อความต้องการของนาย ข. เพราะถ้านาย ก. ไม่ได้เป็นสถาปนิก เขาก็คงไม่สามารถหาคำตอบที่ถูกต้องแก่นาย ข. ได้แน่นอน

ทั้งนาย ก. และนาย ข. ในตัวอย่างนี้ต่างก็เป็น ออบเจกต์ ในระบบสังคมทั้งคู่ การสื่อสารระหว่างออบเจกต์ทำให้ระบบดำเนินไปได้อย่างปกติและมีระเบียบแบบแผน ทั้งยังง่ายต่อการเรียนรู้และแก้ไข

สรุปได้ว่า ออบเจกต์ใด ๆ ในระบบนั้นต้องสามารถนิยามได้ด้วยสถานภาพของตนเองและบทบาทที่ตอบสนองต่อออบเจกต์ตัวอื่น ๆ ในระบบ

4.2.3 ศัพท์พื้นฐานในเรื่อง Object-Oriented

คำว่าออบเจกต์ที่กล่าวมาแต่แรกแล้วนั้น ในความหมายเชิงวิชาการนิยามว่า **มันคือ**ปริมาณหนึ่งในระบบที่ประกอบขึ้นด้วยองค์ประกอบ 2 ส่วน คือ ข้อมูล และ โค้ดโปรแกรม ส่วนข้อมูลใช้เก็บสถานะของตัวมันเองเรียกว่า ข้อมูล(data) และส่วนโค้ดโปรแกรม ใช้ในการตอบสนองต่อออบเจกต์ตัวอื่นในระบบเดียวกัน เรียกว่า เมตธอด (method) ออบเจกต์ใด ๆ ในระบบจะสื่อสารกับออบเจกต์อื่นเพื่อให้บรรลุความต้องการของตน การสื่อสารนี้เป็นลักษณะ "ร้องขอและตอบสนอง" เมื่อออบเจกต์หนึ่งขอความช่วยเหลือจากอีกออบเจกต์หนึ่งเราเรียกว่า มันกำลังส่ง**แมสเสจ** (message) ไปยังออบเจกต์อื่น



ภาพ 2 โครงสร้างความสัมพันธ์ระหว่างออบเจกต์ ดาต้า เมสเสจ และเมตธอด

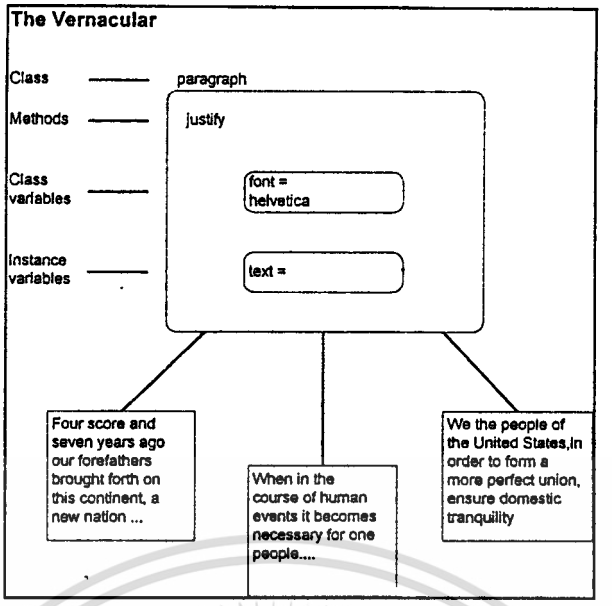
ในขณะที่ผู้พัฒนากำลังใช้งาน Microsoft Windows อยู่ ผู้พัฒนาคือออบเจกต์หนึ่งในระบบ ในขณะที่ วินโดว์ส เป็นอีกออบเจกต์หนึ่ง เมื่อผู้พัฒนาคลิกเมาส์..นั่นคือผู้พัฒนากำลังส่งแมสเสจให้กับ วินโดว์ส เพื่อรอรับผลอะไรบางอย่าง และแน่นอนที่สุดเมื่อ วินโดว์ส ตอบสนอง..นั่นหมายความว่ามันกำลังตอบสนองต่อแมสเสจของผู้พัฒนา ด้วยเมตธอดที่สร้างขึ้นเพื่อรองรับแมสเสจนั้น และเมื่อมีการตอบสนองแมสเสจครั้งหนึ่งก็จะมีผลให้สถานภาพของ วินโดว์ส เปลี่ยนแปลง

4.2.3.1 คลาสและซับคลาส

คำว่าคลาส (class) กับออบเจกต์ เป็นสิ่งคู่กัน เพราะคลาสคือเบ้าหลอมของออบเจกต์ เราจะดูว่าออบเจกต์นี้มีลักษณะหรือ characteristic อย่างไรให้ดูที่คลาส เหมือนกับที่เรากล่าวขึ้นมาลอย ๆ ว่า Dicky เอะ!! ชื่อคนหรือชื่อสุนัข หาก Dicky เป็นออบเจกต์ของคลาสสุนัข Dicky ก็คือหมาน้อยธรรมดาตัวหนึ่ง เออ..แล้ว Dicky เนี่ย..มีหางมั๊ยน้า? เราก็จะพิจารณาจากคุณสมบัติของสิ่งที่เรียกว่า สุนัข ถ้าสุนัขต้องมีหาง Dicky ก็ต้องมีหางด้วยเช่นกัน

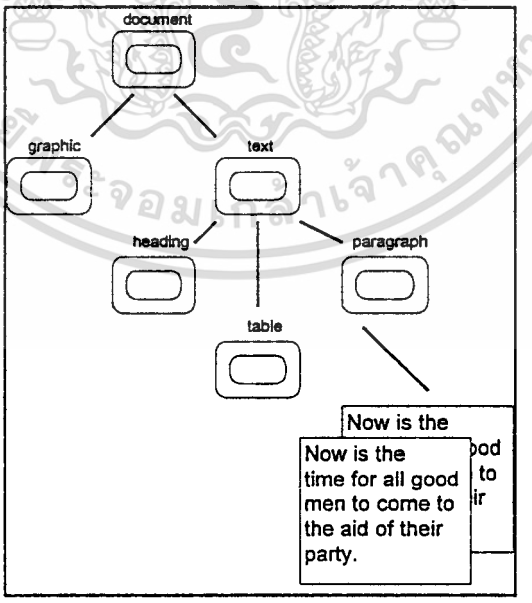
กล่าวได้ว่า Dicky เป็นตัวตนหรือ instance ของคลาสสุนัข นายวิชัยเป็นตัวตนของคลาสมนุษย์ สิ่งที่น่าศึกษาต่อก็คือ นายวิชัยจะหล่อหรือไม่ ? เป็นลักษณะเฉพาะตัว ไม่ใช่ลักษณะของคลาส ถ้าวิชัยหล่อไม่ได้หมายความว่ามนุษย์ทุกคนบนโลกต้องหล่อเหมือนวิชัยลักษณะเฉพาะตัวของออบเจกต์เช่นนี้เรียกว่า instance variable ซึ่งหมายถึงค่าตัวแปรเฉพาะของออบเจกต์ในคลาส ในขณะที่ class variable หมายถึงค่าตัวแปรร่วมของคลาส เช่น คลาสคนต้องมีมือเพียง 2 มือ เป็นต้น

สมมติว่าเราต้องการจะออกแบบโปรแกรมเวิร์ดโปรเซสเซอร์โดยใช้ Object-oriented สักตัวหนึ่ง ขั้นแรกเราได้นิยามคลาส ๆ หนึ่งขึ้นมามีชื่อว่า paragraph คลาส paragraph นี้เป็นคลาสที่ใช้เก็บคุณสมบัติของข้อมูลหนึ่งย่อหน้าของเอกสาร ประกอบด้วยตัวแปร 2 ตัวคือ font และ text ตัวแปร font เป็นตัวแปรที่ใช้เก็บรูปแบบตัวแปรตัวอักษรที่จะปรากฏให้เห็นหน้าจอ และตัวแปร text เป็นอาร์เรย์ที่เก็บข้อความเอกสารในย่อหน้านั้น จากการพิจารณาเราคงบอกได้ว่า font นั้นเป็น class variable เพราะเป็นข้อมูลร่วมของคลาส paragraph ในขณะที่ text เป็น instance variable เพราะเป็นข้อมูลเฉพาะของ instance แต่ละตัว ในย่อหน้าหนึ่งอาจเก็บข้อความ “ผมรักคุณ” แต่อีกหนึ่งย่อหน้าเก็บข้อความว่า “ผมแตกปลาย” ก็ย่อมได้



ภาพ 3 โครงสร้างของคลาส Paragraph

เนื่องจากหลักการ Object-oriented เป็นหลักการที่อาศัยการถ่ายทอดคุณสมบัติเป็นหลัก คลาสแต่ละคลาสในระบบจึงสามารถให้กำเนิดลูกหลานได้ เรียกทายาทของคลาสว่า ซับคลาส (sub-class) ซับคลาสจะรับเอาคุณสมบัติของคลาสผู้ให้กำเนิดที่เรียกว่า parent class เช่น คลาสผลไม้มีซับคลาสคือ คลาส ทุเรียน เป็นต้น ซับคลาสนี่บางทีก็เรียกกันว่า derived class หรือ child class แล้วแต่อารมณ์ของผู้เรียก



ภาพ 4 คลาสและซับคลาสของคลาส Document

เนื่องจากการกำหนดคลาสโดยอาศัยการถ่ายทอดนี้เอง ที่ทำให้เกิดคลาสขึ้นมากมายในระบบ จนเกิดสิ่งที่เรียกว่า class library ขึ้นหรือบางทีก็เรียก class hierarchy แล้วแต่อารมณ์ อีกเหมือนกัน class library นี้เปรียบเสมือนกับไลบรารีที่ใช้เก็บโปรแกรมที่จะนำมาลิงค์ต่อของโปรแกรมที่เขียนขึ้นแบบเก่า แต่แทนที่จะเป็นชุดฟังก์ชันก็เป็นโครงข่ายของคลาสในระบบแทน ถือเป็นสิ่งที่สำคัญที่สุด เพราะเป็นส่วนนิยามของออบเจกต์ทุกตัวของระบบ

คำว่า inheritance หมายถึงการถ่ายทอดคุณสมบัติ inheritance เป็นคีย์เวิร์ดสำคัญที่สุดหนึ่งในสองคำของ Object-oriented หากขาดคุณสมบัติที่ว่านี้ไป Concept ของ Object-oriented จะเป็นหมันไปทันที เพราะด้วยคำนี้ทำให้เกิดข้อดีต่าง ๆ ตามมามากมาย ได้แก่

1. การช่วยลดเวลาในการพัฒนาระบบ
2. ลดค่าใช้จ่ายผู้พัฒนา
3. ได้ระบบที่มีโครงสร้างเป็นระเบียบและปรับปรุงเปลี่ยนแปลงได้ง่าย

4.2.3.2 คำเฉพาะที่ต้องสนใจ

ความหมายและมโนภาพของคำเฉพาะที่ต้องสนใจ 2 คำ ได้แก่คำว่า Encapsulation กับ Polymorphism และขอย้ำว่าทั้งสองคำนี้เป็นคำสำคัญ

หากคุณบังเอิญไปพบสาวสวยคนหนึ่งสมมติชื่อพรพรรณ นอกจากรูปพรรณสันฐานภายนอกแล้ว คุณไม่รู้เรื่องอะไรเกี่ยวกับตัวของเธอเลย บ้านอยู่ที่ไหน? เบอร์โทรศัพท์? มีแฟนหรือยัง? แน่นอนถ้าคุณไม่กล้าพอที่จะเดินไปถาม คุณจะไม่มีวันรู้เรื่องอะไรจากตัวเธอเลย เพราะอะไรหรอ? ก็เพราะว่าคำตอบที่เป็นสถานภาพของเธอเอง เธอย่อมต้องเป็นผู้ตอบเพราะรู้ดีที่สุด

พารากราฟข้างบนเพียงจะกล่าวถึงความหมายของ Encapsulation Encapsulation คือกระบวนการปกปิดความลับของออบเจกต์ การเข้าถึงค่าสถานะภายในออบเจกต์จะกระทำได้โดยผ่านการเห็นชอบจากออบเจกต์เจ้าของเสียก่อน หมายความว่าการเปลี่ยนแปลงแก้ไขค่าตัวแปรภายในออบเจกต์จะต้องกระทำผ่านเมตธอดของออบเจกต์ดังกล่าวเท่านั้น

อีกคำหนึ่งคือ Polymorphism คำ ๆ นี้เป็นคำที่อธิบายได้ยากมากคำหนึ่งของเรื่องนี้เลยทีเดียว น่าแปลกที่คำบางคำอธิบายจนยืดยาวแต่ไม่สามารถสื่อความหมายได้ดีเท่ากับวลีสั้น ๆ วลีที่ว่านี้คือ "One interface multiple implement" (นิยามโดย Herbert Schildt) ที่ว่า "one interface" หมายถึง การบอกให้ทำเพียงอย่างเดียว ส่วน "multiple implement" หมายถึง ทำให้เป็นสิบอย่าง รวมความหมายถึงการบอกแบบเดียวแต่ได้รับการตอบสนองหลายแบบ ตัวอย่างเช่น หากบอกคนใช้ว่า "ฝน... ขอเครื่องดื่มเย็น ๆ สักแก้วซิ" ฝนอาจจะนำโค๊ก สไปร์ต หรือ โกโก้เย็น นมเย็น หรือแม้แต่คลอสเตอร์เย็น ๆ มาบริการ อย่างนี้แหละที่เรียกว่า "One interface multiple implement"

หมายความว่าในครั้งแรกที่สั่งอาจได้โค้กน้ำแข็ง ครั้งที่สองอาจได้กาแฟเย็น แต่การสั่งทั้งสองครั้งนั้นใช้คำสั่งเดียวกัน ถ้าจะเปรียบกับเรื่องการเขียนโปรแกรมก็เหมือนกับการเขียนฟังก์ชันที่มีชื่อเดียวกัน 2 ฟังก์ชันแต่ทำหน้าที่แตกต่างกัน เช่น ฟังก์ชัน Show() อาจมี 2 ฟังก์ชัน แต่มีฟังก์ชันหนึ่งวาดรูปสี่เหลี่ยมแต่อีกหนึ่งวาดวงกลม เหลือเชื่อใช่ไหม? เรื่องนี้เป็นเรื่องที่เป็นไปได้ยากในระบบธรรมดา แต่เป็นเหตุการณ์ปกติในระบบที่เป็น Object-Oriented

ในระบบที่เป็น Object-Oriented ไม่มีนิยามของคำว่า ฟังก์ชันมีแต่เมตธอด เมตธอดมีลักษณะเป็นโปรแกรมย่อยเช่นนเดียวกับฟังก์ชันแต่ต่างกันตรงที่ว่าเมตธอดเป็นโมดูลที่มีเจ้าของไม่ใช้โมดูลลอย ๆ เหมือนกับฟังก์ชัน เจ้าของที่ว่านี้คือออบเจกต์ ดังนั้นดังนั้นไม่ใช่เรื่องแปลกที่ Polymorphism เกิดมีได้ในระบบ เพราะถึงแม้ว่าจะใช้แมสเสจเหมือนกันแต่ออบเจกต์ที่รับแมสเสจต่างกัน ออบเจกต์จะตอบสนองโดยใช้เมตธอดที่มีอยู่ในตัวของมันเอง เมื่อออบเจกต์ A ส่งแมสเสจ Show ไปให้กับออบเจกต์ B จะตอบสนองโดยใช้เมตธอด Show() ที่มีอยู่ในตัวมัน ถ้าออบเจกต์ B เป็นออบเจกต์ในคลาส Rectangle ก็จะวาดรูปสี่เหลี่ยม ถ้าเป็นออบเจกต์ในคลาส Circle ก็จะวาดรูปวงกลม ทั้งหมดนี้ใช้แมสเสจเดียวกัน คือ Show

สรุปว่า ในระบบ Object-oriented การส่งแมสเสจแบบเดียวกัน สามารถตอบสนองได้หลายรูปแบบ และไม่จำเป็นต้องได้รับการตอบสนองเหมือน ขึ้นอยู่กับออบเจกต์ที่รับแมสเสจเป็นสำคัญ

ข้อมูลเปรียบเทียบระหว่างระบบทั่วไปกับระบบ Object-Oriented

หัวข้อ	ระบบทั่วไป	ระบบแบบ Object-Oriented
1. หน่วยสนใจ	ปฏิบัติการ	ตัวปฏิบัติการ
2. ส่วนประกอบของโปรแกรม	ฟังก์ชัน	ออบเจกต์
3. โครงสร้างอ้างอิง	ฟังก์ชันไลบรารี	คลาสไลบรารี
4. การเรียกใช้บริการ	ฟังก์ชันคอลล์ หรือคำสั่ง	ส่งแมสเสจ
5. การไหลของข้อมูล	ควบคุมไม่ได้	ควบคุมได้
6. โอเวอร์เฮดของระบบ	ต่ำ	สูง
7. ข้อผิดพลาดที่เกิดขึ้น	ขึ้นกับความสามารถของผู้พัฒนาระบบ	น่าจะมีโอกาสผิดพลาดน้อยกว่า
8. ขนาดงานที่เหมาะสม	ระบบขนาดกลาง	ระบบขนาดใหญ่
9. ภาษาสนับสนุน	มากมาย	ยังไม่มากนัก

4.3 Object-Oriented กับงานฐานข้อมูล

ฐานข้อมูลเชิงวัตถุ (Object-Oriented Database) หรือที่ย่อ OODB ทำลังเป็นที่สนใจในนักพัฒนาระบบฐานข้อมูลและนักวิจัยโดยทั่วไป

4.3.1 แนวความคิดฐานข้อมูล

ในยุคแรก ๆ ของการนำคอมพิวเตอร์มาใช้ในการประมวลผลข้อมูล โปรแกรมจะทำงานกับแฟ้มข้อมูลระดับกายภาพ (Physical File) โดยตรง จะต้องรู้จักการเปิดแฟ้มเปิดอินเด็กซ์(Index) ไล่ตาม อินเด็กซ์ที่ถูกต้องเพื่อเข้าหาข้อมูลให้เร็วที่สุด ผู้เขียนโปรแกรมจะต้องมีความรู้ความเข้าใจในการใช้งานระบบไฟล์ในระดับกายภาพเป็นอย่างดียิ่งถ้าเป็นโปรแกรมชนิดที่อาจถูกเรียกใช้งานกับแฟ้มข้อมูลที่ผู้อื่นร่วมใช้อยู่ด้วยในระบบหลายผู้ใช้ (Multi-user) ผู้เขียนโปรแกรมยิ่งจะต้องระวังการใช้ข้อมูลในแฟ้มมากเป็นพิเศษเพื่อหลีกเลี่ยงปัญหาต่าง ๆ ในการแก้ไขปรับปรุงข้อมูลในระหว่างที่ผู้อื่นกำลังอ่านข้อมูลชิ้นนั้นอยู่ ซึ่งโดยทั่วไปผู้เขียนโปรแกรมจะต้องทำการล็อค(lock) ข้อมูลหรือแฟ้มข้อมูลที่กำลังใช้งานและถ้าจะให้ดีจริง ๆ อาจต้องทำทรานเซ็กชัน (transaction) หรือกลุ่มงานและดูแลการกู้ข้อมูลกลับคืนเมื่อเกิดไฟดับหรือระบบคอมพิวเตอร์มีปัญหากลางคันขณะทำการปรับปรุงข้อมูลเป็นต้น

คำว่า "ยุคแรก" ที่กล่าวถึงนี้คือ เมื่อประมาณ 20 กว่าปีมาแล้วในประเทศที่ประเทศที่พัฒนาแล้ว ในสมัยนั้นมีนักวิทยาศาสตร์และบริหารระบบข้อมูลหลายท่านมีความเห็นตรงกันว่านักเขียนโปรแกรมกำลังทำงานไม่ตรงจุด เพราะมัวแต่ไปเล่นกับการเข้าถึงข้อมูลในระดับล่าง จนลืมไปว่าหัวใจของงานระบบข้อมูลนั้นอยู่ที่การเข้าใจระบบงานและการเข้าถึงในความสัมพันธ์ระหว่างข้อมูลเป็นอย่างดีต่างหาก ไม่ใช่การเล่นอินเด็กซ์หรือล็อคเรคอร์คเลย

ตัวระบบงานนั้นอาจทำการวิเคราะห์ได้โดยใช้วิธีต่างๆ มากมาย เช่น Data flow diagram หรือ A-graph ของ ISAC ผลที่ได้ก็จำทำให้เราทราบว่าควรจะพัฒนาโปรแกรมหรือโมดูลที่ทำหน้าที่อะไรบ้าง ตามที่ระบบงานต้องการ

อีกเรื่องหนึ่งคือ " ความสัมพันธ์ระหว่างข้อมูล " นั้นเป็นเรื่องใหญ่ ทุกคนเห็นพ้องต้องกันว่าโปรแกรมในระบบงานควรจะมองเห็นและทำงานกับโครงสร้างข้อมูลชนิดที่สามารถแสดงความสัมพันธ์ระหว่างข้อมูลได้ดี โดยไม่ต้องยุ่งเกี่ยวกับรายละเอียดระดับกายภาพอีกต่อไปแล้ว และนี่คือที่มาของระบบฐานข้อมูล ซึ่งมีซอฟต์แวร์ DBMS (Database Management System) สำหรับจัดการรายละเอียดระดับกายภาพพร้อมกับนำเสนอโครงสร้างข้อมูลในระดับตรรกะ (Logical Data Structure) ที่สามารถแสดงความสัมพันธ์ระหว่างข้อมูลได้เหมาะสมกับงาน

ปัญหาหนักข้อหนึ่งที่ยังเป็นหัวข้อวิจัยเลยระดับปริญญาเอกไปอีกก็คือ ทำอย่างไรจะสามารถแสดงความสัมพันธ์ระหว่างข้อมูลพร้อมทั้งกฎเกณฑ์ต่างๆ ที่ควบคุมความสัมพันธ์ให้ถูกต้อง?

ในข้อนี้เมื่อ 20 ปีที่แล้ว ทางองค์กรมาตรฐานสากลโลก CISO : International Standard Organization) ก็ได้คิดสถาปัตยกรรมฐานข้อมูลแบบ 3 ระดับขึ้น (the 3 schemas Architecture) และกำหนดว่าระดับกลางที่เป็น Conceptual Schema นั้น คือระดับที่จะใช้แสดงความสัมพันธ์ระหว่างข้อมูลดังกล่าว ทั้งนี้จะอาศัยแบบจำลองข้อมูล (Data Model) เป็นตัวแสดงความสัมพันธ์ระหว่างข้อมูลและแบบจำลองข้อมูลที่เป็นที่นิยมแพร่หลาย 3 แบบ ก็คือ Hierarchical Model หรือแบบแผนภูมิต้นไม้ Network Model หรือแบบโครงข่ายและต่อมาก็มีแบบรีเลชั่น (Relational Model) ที่กำลังเป็นที่นิยมอยู่ในปัจจุบัน

4.3.2 ปัญหาของเรคอร์ด

แบบจำลองข้อมูลที่กล่าวมาล้วนแต่ใช้เรคอร์ดเป็นโครงสร้างทั้งสิ้น บางครั้งถึงจะเลี่ยงไปเรียกอย่างอื่น เช่น tuple ในรีเลชันแนลความจริงก็คือ flat record นั่นเองการใช้เรคอร์ดสำหรับเก็บข้อมูลและแสดงความสัมพันธ์ระหว่างข้อมูล (โดยเฉพาะอย่างยิ่งแบบรีเลชันแนล) ดูจะเหมาะกับงานเก็บข้อมูลทั่วไปในทางธุรกิจและงานสำนักงาน (ในทางธุรกิจนั้นการจัดเก็บเอกสารก็เรียกว่า " Record Keeping " อยู่แล้ว) อย่างไรก็ตาม ข้อบกพร่องของการเก็บข้อมูลเป็นเรคอร์ดนั้นได้มีผู้หยิบยกมาวิจารณ์กันมาก ซึ่งจะขอนำมากล่าวในที่นี้ในบางเรื่องโดยเฉพาะที่จะไปเกี่ยวข้องกับ OODB ดังนี้

4.3.2.1. การเก็บข้อมูลเป็นเรคอร์ดนั้น มีปัญหาในกรณี Subtype Hierarchy กล่าวคือ ถ้าจะแสดงว่าเฉพาะบางส่วนของข้อมูลเท่านั้น ที่มีคุณสมบัติหรือความสัมพันธ์เฉพาะด้านบางประการ เรคอร์ดที่ได้จะมีสภาพที่ไม่น่าดูคือ ตัวเนื้อข้อมูล (record instance หรือ tuple)จะประกอบด้วยค่า null เป็นจำนวนมาก

ยกตัวอย่างเช่น การออกแบบเรคอร์ดสำหรับเก็บข้อมูลผลิตภัณฑ์เสื้อผ้า เราจะพบว่าผลิตภัณฑ์ประเภทดังกล่าวนั้นมีแอตตริบิวต์ (Attribute)หรือคุณสมบัติหลากหลายมากถ้าเป็นเสื้อก็จะมีแอตตริบิวต์เฉพาะสำหรับเสื้อเช่น แขนสั้น แขนยาว เสื้อเชิ้ตหรือ เสื้อยืด มีปกหรือไม่ ฯลฯ ถ้าผลิตภัณฑ์นั้นเป็นกางเกงหรือกระโปรงก็จะมีแอตตริบิวต์ต่างๆก็ไปอีก ดังนั้นการจัดฟิลด์หรือคอลัมน์ของเรคอร์ดนี้ถ้าจัดของผลิตภัณฑ์ทุกประเภทเข้าด้วยกัน โครงสร้างเรคอร์ดที่ได้ก็จะออกมามีขนาดยาวหลายฟิลด์โดยที่ตังเนื้อข้อมูลในเรคอร์ดเองจะใช้เพียงบางคอลัมน์ที่ตรงกับประเภทผลิตภัณฑ์ของตนเท่านั้นที่เหลือจะเป็นค่า null ไปหมด

บางท่านอาจนึกในใจว่า ถ้าเกิดเหตุการณ์อย่างนี้ก็เรื่องอะไรจะต้องไปเก็บฟิลด์ต่างๆ เหล่านั้นไว้ในไฟล์เดียวกัน ทำไมไม่แบ่งโครงสร้างเรคอร์ดออกเป็นหลายๆ ไฟล์ เช่น ไฟล์แรกสำหรับฟิลด์ที่ทุกผลิตภัณฑ์ต้องมี จำพวกราคาขายปลีก ชื่อแบบ ชื่อยี่ห้อ อะไรพวกนี้ ไฟล์ที่สองก็จะเห็นไฟล์เฉพาะของเสื้อเชิ้ต กับไฟล์เสื้อยืดได้อีก นอกนั้นก็เป็นไฟล์เฉพาะของกระโปรง ไฟล์ของกางเกง คือ 1 ไฟล์สำหรับผลิตภัณฑ์ แต่ละประเภทไปเลย การทำเช่นนั้นอาจช่วยทำให้แต่ละไฟล์มีแต่เนื้อข้อมูลไม่มี null value คือ ไฟล์ดูสวยขึ้นและประหยัดเนื้อที่ลงได้มากไม่มีส่วนสูญเปล่าแต่ผลข้างเคียงที่จะตามมาก็คือ การประมวลผลจำทำได้ยากขึ้น การ Insert , delete และ Update ข้อมูลจะต้องกระทำบนหลายไฟล์ และถ้ามีคำถามค้นหาข้อมูลจะต้องทำการ join เพราะข้อมูลแยกกันอยู่อย่างน้อยก็ไฟล์ที่เป็นข้อมูลทั่วไปจะต้อง join ไฟล์เฉพาะผลิตภัณฑ์ และการ join นี้ใครๆ ก็ทราบว่าเป็นการปฏิบัติการซึ่งช้าและกินทรัพยากรมากที่สุดของการปฏิบัติการบนฐานข้อมูล ดังนั้นการตัดสินใจแยกไฟล์ออกเป็นหลายไฟล์แทนที่จะรวมเป็นไฟล์เดียว ก็เป็นเพียงการตัดสินใจแลกกันระหว่าง Storage space กับการประมวลผลข้อมูลนั่นเอง

ในระบบที่ใช้เรคอร์ด เราถูกบีบบังคับให้เลือกทางใดทางหนึ่ง คำถามจึงมีอยู่ว่า อะไรทำให้เราต้องเลือก ? ไม่ใช่เพราะตัวโครงสร้างข้อมูลที่เป็นเรคอร์ดหรือ ?

4.3.2.2. การไม่สามารถแสดงกฎกำกับความถูกต้อง (Integrity constranintgs) ไว้บนโครงสร้างข้อมูล

เรื่องนี้เป็นเรื่องหนักใจนักวิชาการและผู้ใช้ฐานข้อมูลมืออาชีพเป็นจำนวนมาก เนื่องจากการแสดงความสัมพันธ์ระหว่างข้อมูลอยู่ในรูปของเรคอร์ดยากต่อการแทนกฎต่างๆ บนตัวโครงสร้างข้อมูล ยกตัวอย่างเช่น จะบอกอย่างไรว่าผู้ชายเท่านั้นที่ลาบวชได้ และผู้หญิงเท่านั้นที่ลดคลอดได้ ? จะบอกอย่างไรว่าพนักงานในบริษัทนี้อายุต้องไม่น้อยกว่า 18 ปี และไม่เกิน 60 ปี ? จะบอกอย่างไรว่าถ้าพนักงานคนหนึ่งเป็นโสด การอัพเดตจะต้องเปลี่ยนจากโสดไปเป็นหม้ายเลยไม่ได้เหล่านี้เป็นต้น กฎเกณฑ์ต่างๆซึ่งใช้บังคับความถูกต้องของข้อมูลและความสัมพันธ์ระหว่างข้อมูลนี้ รวมเรียกว่า Integrity constraints หรือ Integrity Rules

ในการจำลองข้อมูลแบบเก่าทั้งสามแบบ ซึ่งใช้เรคอร์ดนั้นไม่สามารถแทนกฎดังกล่าวบนโครงสร้างข้อมูลได้ แต่จะได้ในส่วนที่เป็นกฎคุมโครงสร้างข้อมูล (Static Structural Constraints) เช่น กฎที่ว่าข้อมูลใน Primary Key จะเป็น null ไม่ได้และกฎที่ว่า ข้อมูลใน Foreign Key จะต้องอยู่ในชุดของข้อมูลใน Primary Key ที่เกี่ยวข้องกันหรือไม่ก็มีค่าNull เป็นต้นกฎทั้งสองนี้เป็นเพียงสองกฎที่สามารถบังคับได้บนโครงสร้างข้อมูลแบบรีเลชัน ซึ่งจะเห็นได้ว่าไม่เกี่ยวข้องกับการควบคุมความถูกต้องของข้อมูลที่ยกเป็นตัวอย่างในย่อหน้าก่อนเลย ถ้าจะควบคุมความถูกต้องของข้อมูลเช่นนั้น (เรียกว่า Behavioral Integrity constraints) จะต้องเขียนโปรแกรมตรวจสอบเอาเอง

การเขียนโปรแกรมจัดการตรวจสอบความถูกต้องของข้อมูลเองนั้น มีข้อเสียคือ ทำให้ตัวโปรแกรมที่จะทำงานกับข้อมูลยุ่งยากขึ้น เพราะต้องคอยตรวจความถูกต้องทุกครั้งที่จะอัพเดตข้อมูล และถ้าเป็นระบบหลายผู้ใช้ หรือหลายโปรแกรมขึ้นมาอาจมีบางโปรแกรมพลาดลืมตรวจสอบในบางจุดข้อมูลก็จะเสียหายและไม่ถูกต้องตามกฎที่วางไว้ นอกจากนั้นการนำส่วนตรวจสอบดังกล่าวฝังไว้ในโปรแกรมทุกโปรแกรมที่มีการปรับปรุงแก้ไขข้อมูลได้ยังทำให้ระบบงานยากบำรุงรักษา เมื่อมีการแก้ไขกฎบางอย่าง โปรแกรมทั้งหมดจะต้องถูกแก้ไขตามไปด้วยในระบบงานขนาดกลางถึงขนาดใหญ่เรื่องตามแก้โปรแกรมเหล่านี้เป็นเรื่องเสียเวลาและค่าใช้จ่ายมาก องค์กรมาตรฐานสากลนานาชาติ (ISO : International Standard Organization) จึงกำหนดไว้ว่า ระบบฐานข้อมูลที่สมบูรณ์แบบจะต้องสามารถบังคับความถูกต้องเหล่านี้ได้ที่ระบบ DBMS โดยประกาศ (Declare) เป็นกฎไว้อย่างชัดเจนบนโครงสร้างข้อมูลระดับตรรกะ

4.3.2.3. การไม่สามารถแสดงวัตถุที่มีโครงสร้างซับซ้อน (Complex Object) ได้อย่างเหมาะสม

โครงสร้างของข้อมูลชนิดเรคอร์ดเป็นโครงสร้างที่ใช้แสดงค่าของข้อมูล (Value-Oriented data structure) เช่นแบบจำลองข้อมูลแบบรีเลชันแนล ซึ่งใช้รีเลชันเป็นโครงสร้างข้อมูลเพียงอย่างเดียว และในปัจจุบันนี้แทนรูปแบบของรีเลชันด้วยตารางที่เป็นชนิดเรียบ (Flat table) และเช่นเดียวกับที่ได้กล่าวมาแล้ว แบบจำลองข้อมูลแบบไฮราคีและโครงข่ายต่างก็ใช้โครงสร้างข้อมูลพื้นฐานเป็นเรคอร์ดทั้งสิ้น ลองนึกดูว่าอะไรจะเกิดขึ้นถ้าให้เราใช้โครงสร้างข้อมูลอย่างนี้เก็บรายละเอียดของวงจรโทรทัศน์สักวงจรหรือรถยนต์สักหนึ่งคันคงยุ่งมาก การปฏิบัติกับตาราง ดังนั้นจะต้องมีการ join กันมากกว่าจะได้คำตอบอะไรสักอย่าง เพียงแต่คิดเรื่องออกแบบโครงสร้างข้อมูลสำหรับเก็บรายละเอียดวงจรให้มีความซ้ำซ้อนน้อยที่สุด ก็หนักใจแล้วเพราะอะไหล่อุปกรณ์แต่ละชิ้นมีคุณสมบัติเฉพาะตัว และยังประกอบด้วยอุปกรณ์เล็กกว่าย่อยลงไปอีกซับซ้อนหลายขั้น อุปกรณ์แต่ละตัวก็เชื่อมต่อและมีความสัมพันธ์กับอุปกรณ์อื่นๆ ในหลายลักษณะ ที่หนักขึ้นไปอีกก็คือการแสดงการทำงานของอุปกรณ์ที่มีความสัมพันธ์กันอย่างซับซ้อนนี้ อาจต้องใช้เรื่องของแกนเวลา (time Axis) มาประกอบด้วย

4.3.3 แนวความคิดเชิงวัตถุ (Object - Oriented Concept)

จากปัญหาหลักของโครงสร้างแบบเรคอร์ดที่ได้กล่าวแล้วข้างต้นจึงมีผู้คิดแบบจำลองข้อมูลชนิดใหม่ เพื่อให้สามารถแก้ปัญหาเหล่านี้ได้ แนวความคิดดังกล่าวพยายามแทนรูปแบบของวัตถุ (Object) โดยตรงแทนที่จะใช้ค่าของข้อมูล (Value) ซึ่งจะซ่อนรายละเอียดของวัตถุต่างๆ ไว้ภายในขอบเขตของวัตถุ หลักการซ่อนรายละเอียดไว้ภายในขอบเขตหนึ่งนี้เรียกว่า Encapsulation

ซึ่งความจริงก็คือหลักการ Abstract Data type ในภาษาโปรแกรมนั่นเองในออบเจกต์นี้นอกจากจะมีโครงสร้างข้อมูลแล้วยังมีวิธีการปฏิบัติการกับข้อมูล (Method) ซ่อนอยู่ภายในด้วย การเรียกใช้ออบเจกต์จึงต้องส่งคำสั่ง (Message) ผ่านขอบเขตของวัตถุเข้าไปเพื่อให้เมตธอดทำงานตามต้องการอยู่ภายใน การแสดงความสัมพันธ์ระหว่างข้อมูลโดยใช้ออบเจกต์นี้สามารถซ่อนรายละเอียดภายในแต่ละออบเจกต์ไว้ได้ ดังนั้นความสัมพันธ์ระหว่างออบเจกต์จึงแสดงได้โดยสะดวกไม่ซับซ้อนเกินไปภายในแต่ละออบเจกต์เองก็จะประกอบด้วยออบเจกต์ย่อยๆลงไปอีกได้เรื่อย ๆ ซึ่งชั้นสุดท้ายก็จะไปถึงออบเจกต์ที่มีชนิดข้อมูล (class หรือ abstract data type) อย่างเบื้องต้นที่สุด เช่น integer หรือ enumerate data type ซึ่งเป็นชนิดข้อมูลที่ผู้ใช้กำหนดเอง เป็นต้น การแสดง Subtype สามารถแสดงได้อย่างสะดวกในระดับออบเจกต์โดยใช้เส้นโค้ง (arc) หรือตัวชี้เชิงตรรกะ (Logical Pointer) แสดงความสัมพันธ์ชนิดนี้ กฎเพื่อควบคุมความถูกต้อง (integrity constraints) ส่วนใหญ่จะอยู่ในรูปของเมตธอดภายในออบเจกต์ (เช่น Behavioral Constraints) และใช้ตัวชี้เชิงตรรกะแสดง Structural constraints เช่น Referential Integrity เป็นต้น

แนวความคิดเชิงวัตถุดังกล่าว เริ่มมีใช้ในภาษาโปรแกรมก่อน เช่น ภาษา Small talk ซึ่งพัฒนาขึ้นมาเพื่อวัตถุประสงค์นี้โดยเฉพาะ และภาษา C++ ซึ่งเพิ่มแนวความคิดของ Object เข้าไปในภาษา C ปัจจุบันแนวทางการเขียนโปรแกรมแบบเชิงวัตถุนี้ได้รับการต้อนรับจากนักเขียนโปรแกรมมาก เนื่องจากสามารถซ่อนรายละเอียดของโปรแกรมย่อยในขอบเขตของออบเจกต์ไว้ได้ ทำให้แต่ละส่วนของโปรแกรมเป็นโมดูลที่พัฒนาแยกส่วนได้สะดวก แต่ละส่วนต่อเติมได้ง่ายโดยไม่กระทบกระเทือนส่วนอื่น และยังมีชนิดของข้อมูลที่ผู้เขียนโปรแกรมกำหนดได้เองโดยสะดวก นอกจากนั้นออบเจกต์ที่เป็นซับเซ็ตส่วนย่อยของออบเจกต์อื่น (subtype หรือ subclass) ยังได้รับการถ่ายทอดทั้งโครงสร้างข้อมูลและอัลกอริธึม จากออบเจกต์ที่เป็น Super type อีกด้วย

4.3.4 ฐานข้อมูลเชิงวัตถุ (Object 0 Oriented Database)

ฐานข้อมูลเชิงวัตถุ ก็คือฐานข้อมูลที่ใช้แบบจำลองข้อมูลซึ่งแสดงข้อมูลและความสัมพันธ์ระหว่างข้อมูลในแนวทางของออบเจกต์ดังกล่าวมาข้างต้น ความแตกต่างระหว่างโปรแกรมเชิงวัตถุ (Object Oriented Program) กับฐานข้อมูลเชิงวัตถุ (Object Oriented database) โดยหลักการแล้วมีเพียงว่าในฐานข้อมูลนั้นต้องมีออบเจกต์ถาวร (Persistent Objects) ที่เก็บไว้บนหน่วยความจำสำรอง (Secondary Storage) เช่นฮาร์ดดิสก์ เป็นต้น

มีผู้คาดกันว่า OODB นี้จะมาแทน RDB (ฐานข้อมูลแบบรีเลชัน) ในอีกไม่นานข้างหน้า ตามความคิดเห็นทั่วไปนั้นเห็นว่าควรจะแทน RDB ในงานเฉพาะอย่างที่เหมาะกับออบเจกต์ เช่น งานออกแบบทางวิศวกรรม และการเก็บข้อมูลรายละเอียดของวงจรไฟฟ้าหรือเครื่องจักร หรืออะไร

ก็ตามที่เป็นวัตถุเชิงซ้อน (Complex Object) แต่อาจจะไม่เหมาะกับงานสารสนเทศและงานประมวลผลข้อมูลโดยทั่วไป

สิ่งที่ทำให้คิดเช่นนั้น นอกจากจะด้วยความจริงที่ว่าโครงสร้างข้อมูลในงานธุรกิจนั้นอยู่ในเรคอร์ดโดยธรรมชาติอยู่แล้ว ยังมีข้อคิดเชิงวิชาการฐานข้อมูลซึ่งเป็นข้อที่นักวิชาการในสาขานี้มองเห็นกันอยู่ว่าเป็นปัญหา ได้แก่

4.3.4.1. OODB ยังไม่มีทฤษฎีคณิตศาสตร์รองรับและอธิบายได้โดยชัดเจนเช่นเดียวกับ RDB RDB นั้นใช้ทฤษฎีเซ็ต (Set theory) และทฤษฎีตรรกศาสตร์ (Predicate Logic) มาอธิบายได้อย่างเหมาะสม ในขณะที่ OODB ยังไม่มี และข้อนี้ทำให้ OODB ยังถือเป็นศิลป์ (Art) ไม่ใช่ศาสตร์ (Science) การออกแบบฐานข้อมูลที่ถือเป็นศิลป์นั้นเปรียบได้เหมือนกับการสร้างตึกหรือต่อเรือสำเภาโดยไม่รู้วิชาฟิสิกส์ กลศาสตร์และวิศวกรรมโยธาพังหรือไม่พัง
ไม่อาจทราบล่วงหน้าได้และความสำเร็จของงานขึ้นกับทักษะส่วนบุคคลของผู้ออกแบบซึ่งไม่มีหลักการที่อธิบายได้แน่นอน สิ่งนี้ถือเป็นสิ่งที่ OODB ยังต้องพัฒนาต่อ

4.3.4.2. ภาษาที่ใช้กับ OODB ในปัจจุบันนี้ส่วนหนึ่งได้รับอิทธิพลมาจาก Small talk เช่น (ภาษา Opal ของ Gemstone OODBMS ที่มีชื่อเสียง)และอีกส่วนหนึ่งได้รับอิทธิพลจากการต่อเติมภาษา C เช่น (ภาษา COP และ TDL ของ Vbase OODBMS)ภาษาทั้งสองนั้นเป็น record-at--a-time คือการอ่าน (read) 1 ครั้งจะได้ข้อมูล 1 เรคอร์ดและการเขียน (Write) 1 ครั้งจะได้ 1 เรคอร์ด ดังนั้น เมื่อมาประยุกต์กับออบเจกต์ การปฏิบัติการก็จะยังปฏิบัติบนเซ็ตของออบเจกต์ไม่ได้หรือไม่ดีพอ (มีผู้พัฒนาและปรับปรุงภาษา Object - Oriented SQL อยู่)

ปัจจุบันนี้การประมวลผลโดยใช้ปฏิบัติการเป็นเซ็ตนั้นเป็นที่ยอมรับกันทั่วโลกว่าให้ประสิทธิผล (Productions) สูงกว่าแบบที่ละชิ้นข้อมูลมากมายและเป็นข้อที่ทำให้ภาษารีเลชันแนลเหนือกว่าภาษาโปรแกรมธรรมดามากมาย ถึงแม้จะมีผู้โจมตีว่าการใช้ภาษารีเลชันแนล เช่น SAL ซึ่งปฏิบัติการเป็นเซ็ตร่วมกับภาษาโปรแกรมธรรมดา ซึ่งปฏิบัติกับข้อมูลทีละชิ้น (Record-at-a-time) เช่นภาษาโคบอลหรือภาษาซีนั้นเป็นการผิดฝาผิดตัวเข้ากันไม่ได้ดี (Impedance Mismatch) และสรุปว่าภาษารีเลชันแนลชนิดที่ปฏิบัติการเป็นเซ็ตใช้ไม่ได้ต้องแก้ไขนั้นกลับเห็นว่าถ้าจะแก้ต้องแก้ให้ปฏิบัติการเป็นเซ็ตให้เหมือนกันหมดจะถูกต้องกว่า เพราะภาษาระดับเซ็ตนอกจากจะให้ประสิทธิผลที่สูงกว่าแล้วยังเหมาะกับการทำงานในระบบฐานข้อมูลแบบกระจาย (Distributed Database System) อีกด้วย เนื่องจากไม่ต้องส่งคำสั่ง Read หรือ Write จำนวนมากไปมาในโครงข่ายสื่อสารนั้นเอง ความจริงข้อนี้เป็นที่ยอมรับกันมานานทั่วโลก โดยตำราวิชาการด้านฐานข้อมูล

แบบกระจายทุกเล่ม จะอ้างอิง RDB ทั้งสิ้น ถึงแม้บางจุดในโครงข่ายจะเป็นแบบอื่นที่ไม่ใช่ RDB แต่ก็จะมีตัวแปลงให้ผู้ใช้มองเห็นเป็น RDB อยู่ข้างบน

4.3.4.3. หลักการฐานข้อมูลสำหรับงานสารสนเทศนั้น ฐานข้อมูลมีไว้เพื่อรองรับงานประมวลผล ตามความต้องการของผู้ใช้ จึงจำเป็นอยู่เองที่จะต้องมีความเป็นเอนกประสงค์ระดับหนึ่ง การนำ Method หรือ โปรแกรมย่อย ซึ่งทำงานเฉพาะกิจกับออบเจกต์ใดฝังไว้รวมกับโครงสร้างข้อมูล ก็เท่ากับเป็นการนำโปรแกรมไปผูกกับข้อมูลโดยตรง ทำให้ฐานข้อมูลซึ่งปกติมีแต่ข้อมูลก็ใหญ่และซับซ้อนมากอยู่แล้วยิ่งซับซ้อนขึ้นไปอีก นอกจากจะเป็นหนทางให้มีข้อผิดพลาด (Bug) เพิ่มขึ้นแล้ว มุมมองที่ผู้ใช้มองออบเจกต์จากงานที่ต่างกันก็อาจแตกต่างกันทำให้มีเมตธอดจำนวนมาก ซึ่งอาจไม่เกี่ยวข้องกับงานของเรา ถูกผูกรวมกับออบเจกต์เป็นการเพิ่มภาระให้แก่ฐานข้อมูลโดยไม่จำเป็น

หลักการนี้เห็นได้ชัดเจนว่าตรงข้ามกับวัตถุประสงค์ของโปรแกรมเชิงวัตถุ (Object Oriented Program) เพราะโปรแกรมนั้นเขียนขึ้นมาเพื่อวัตถุประสงค์เฉพาะงาน การใช้แนวคิดเชิงวัตถุในโปรแกรม จึงเป็นเรื่องที่ตรงกับวัตถุประสงค์ของโปรแกรมอย่างยิ่ง

4.3.4.4. หลักวิชาต่างๆ ที่จำเป็นในการพัฒนาพื้นฐานข้อมูลรวมขนาดใหญ่ หลายผู้ใช้ เช่น การควบคุมให้ผู้ใช้ใช้ข้อมูลร่วมกัน (Concurrency Control) การจัดการกู้คืนเมื่อเกิดความเสียหาย การประมวลคำถามให้เข้าถึงข้อมูลอย่างดีที่สุด (Query Optimization) ล้วนแล้วแต่จะต้องพัฒนาขึ้นใหม่ตามแนวคิดใหม่แทบทั้งสิ้น จึงยังต้องใช้เวลาอีกนานพอสมควรกว่าจะพ้นความเป็นต้นแบบในห้องทดลองออกมาสู่ประชาชนทั่วไปได้ RDB นั้นว่าไปแล้วเพิ่งเริ่มใช้กันได้ไม่นานแม้ว่าผลทางวิชาการดังกล่าว จะใช้ได้ในห้องทอลองมานานแล้วก็ตาม การต้องรื้อของเก่าทิ้ง แล้วทำใหม่หมด (ถ้าคุ้มที่จะทำ) ก็ต้องใช้เวลาอีกนานพอสมควร

จากเหตุผลดังกล่าวข้างต้น ทำให้เห็นว่า RDB จะอยู่กับงานสารสนเทศอีกนาน จุดที่จะต้องปรับปรุงจริงๆ คือเรื่องของความถูกต้องของข้อมูล (Integrity Constraints) นั้นไม่จำเป็นต้องฝังรวมกับตัวข้อมูล เช่น OODB แต่น่าจะรวมอยู่กับส่วนตรวจสอบข้อมูลของ DBMS (ซึ่งเป็นซอฟต์แวร์แยกออกจากฐานข้อมูล) มากกว่า อย่างไรก็ตามถ้าเป็นงานเฉพาะด้านและผู้ใช้คนเดียว (Single User) เช่นงานออกแบบทางวิศวกรรมหรือระบบผู้เชี่ยวชาญฐานข้อมูลเชิงวัตถุ (OODB) จะเหนือกว่า RDB มากทีเดียว

4.4 ปัญหาและวิธีการแก้ไขในระบบฐานข้อมูลแบบกระจาย

นอกจากเทคโนโลยีที่กล่าวมาแล้วนั้นยังต้องพิจารณาจำนวนองค์ประกอบที่จะช่วยเหลือตัวอื่น ๆ เมื่อผู้พัฒนาจะทำการพัฒนาระบบฐานข้อมูลแบบกระจายซึ่งองค์ประกอบที่เราจะต้องการพิจารณานี้สามารถสรุปออกมาเป็นหัวข้อหลักได้ 4 ข้อ คือ

องค์ประกอบที่ 1 การสร้างระบบฐานข้อมูลแบบกระจายนั้นจำเป็นต้องทำให้ระบบเครือข่ายแต่ละระบบสามารถเชื่อมต่อกับเครือข่ายอื่น ๆ ได้ทุกเครือข่าย ระบบเครือข่ายสื่อสารทั้ง LAN และ WAN (Wide-Area Network)นั้นเป็นระบบการเชื่อมต่อที่ไม่แข็งแรงนัก การเชื่อมต่อระหว่างสถานีที่อยู่ไกลนั้นต้องใช้เวลาในการติดต่อนานและเสียค่าใช้จ่ายสูง ปัญหาที่เกี่ยวกับเน็ตเวอร์คโพรโตคอลแบนด์วิดท์ของระบบ WAN และการจัดการระบบเครือข่ายและสถาปัตยกรรมที่แตกต่างกันระหว่าง LAN กับ WAN ทำให้การรวมระบบเครือข่ายต่าง ๆ เข้าด้วยกันนั้นประสบความยุ่งยากหลายอย่าง

โพรโตคอลบนเน็ตเวอร์คมีหลายตัวด้วยกัน เช่น IPX TCP/IP และ Apple Talk ส่วน SAN (Systems Network Architect) และ NetBEUI นั้นเป็นเน็ตเวร์คโพรโตคอลที่ต้องการเส้นทางการสื่อสารเป็นของตัวเอง อย่างไรก็ตามไม่จะใช้โพรโตคอลใดก็ตามค่าใช้จ่ายอื่น ๆ สำหับระบบ WAN ไม่ว่าจะเป็นการเช่าสายสื่อสาร (Communication Line) หรือจะใช้เซอร์เวอร์ที่ใช้ในการสื่อสาร (Communication Server) และสายโทรศัพท์สาธารณะก็ตาม ก็คงสูงอยู่ดี แต่สถานการณ์ยังไม่เลวร้ายจนเกินไปนัก น่าดีใจที่พัฒนาการของเครือข่ายการสื่อสารในขณะนี้นับได้ว่าเป็นส่วนที่มีการเจริญเติบโตมากที่สุดในอุตสาหกรรมที่เกี่ยวกับคอมพิวเตอร์ โดยในปีที่ผ่านมาได้มีการปรับปรุงในด้านติดต่อสื่อสารขึ้นหลายอย่างด้วยกันอาทิ เช่น เน็ตเวิร์คการ์ดที่มีความเร็วสูงขึ้น ปรับปรุงเทคโนโลยีที่เกี่ยวกับการบีบข้อมูลและดาต้าลิงค์สวิตชิงและการติดต่อสื่อสารแบบไม่สัมพันธ์ (Asynchronous Communication) ที่เชื่อถือได้ เป็นที่คาดหมายกันว่าค่าใช้จ่ายในการสื่อสารข้อมูลจะถูกลงมากในระยะเวลาอันใกล้นี้

ทางด้านอุปกรณ์ปลายทางก็เช่นกัน มีการพัฒนาระบบใหม่ ๆ ออกมาสนองความต้องการกันมากขึ้นเซอร์เวอร์สำหรับการสื่อสารจาก Xylogics ที่ชื่อว่า Annex สามารถใช้งานร่วมกับโพรโตคอลได้หลายตัว และสามารถสื่อสารระหว่างสถานีที่อยู่ห่างไกลขึ้นบริษัทที่เกี่ยวข้องกับระบบ LAN ก็ได้ทำการพัฒนาผลิตภัณฑ์เหล่านี้อย่างจริงจังไม่ว่าจะเป็น Novell/USL's Tuxedo และ Banyan's Vines Enterprise Data Distributionเพื่อสร้างรูปแบบการกระจายข้อมูลให้รวมเป็นส่วนหนึ่งของไฟล์เซอร์เวอร์ ซึ่งความก้าวหน้าทั้งหมดที่กล่าวมานี้เป็นความหวังที่จะช่วยแก้ปัญหาของระบบเครือข่าย ในอนาคตอันใกล้นี้

องค์ประกอบที่ 2 คือการหาข้อลงตัวของข้อจำกัดในมาตราฐานระหว่างผลิตภัณฑ์ด้านฐานข้อมูล ผู้ผลิตหลายราย เช่น Oracle, SysBase, Ingress, Progress, Informix, Borland, Cincom และ Gupta ได้มีการผลิต Software ของตนเองให้สามารถใช้รันบนเครื่องและระบบปฏิบัติการที่แตกต่างกันหลาย ๆ ตัว ถ้าหากระบบที่ใช้อยู่ไม่ใช่ระบบแบบผู้ผลิตรายเดียว การปรับปรุงระบบเครือข่ายแบบกระจาย โดยเฉพาะเรื่องที่เกี่ยวกับความเข้ากันระหว่างเซอร์เวอร์กับเซอร์เวอร์จะเป็นปัญหาที่หนักพอสมควร ยิ่งไปกว่านั้นผลิตภัณฑ์ฐานข้อมูลหลาย ๆ ตัวก็จะขึ้นกับดาต้าโมเดลที่แตกต่างกัน (เช่นโมเดลแบบ Hierarchical JMS ของบริษัท IBM หรือโมเดลแบบเครือข่ายของบริษัท Cincom) ซึ่งทำให้เกิดปัญหาเกี่ยวกับการแอ็กเซสข้อมูลและการทำรายการการตอบคำถาม (Query transaction) แม้ว่ามาตราฐานในระบบเครือข่ายจะไม่สามารถก้าวตามทันเทคโนโลยีในปัจจุบันก็ตามแต่ ขณะนี้ได้เริ่มมีการพัฒนาขึ้นมาบ้างแล้ว โดยเริ่มพัฒนาในส่วนของระบบปฏิบัติการฐานข้อมูลเชิงสัมพันธ์ (Relational DBMS) ANSI/ISO SQL IBM's SAA/SQL มาตราฐาน SQL89 และ SQL92 เมื่อเร็ว ๆ นี้ SQL Access Group ได้ทำการกำหนดรายละเอียดที่เพิ่มเข้าไปใน SQL92 และนำมาทำงานเป็นส่วนหนึ่งของมาตราฐาน SQL รุ่นต่อไปซึ่งกำหนดออกประมาณ ปี 1996 สำหรับข้อมูลแบบกระจายนั้นมาตราฐาน DRDA (Distributed Relational Database Architecture) ของบริษัท IBM ได้สร้างโพรโตคอลที่เป็นมาตราฐานเพื่อใช้สำหรับการแอ็กเซสข้อมูลที่อยู่ไกลโดยเข้าแพลทฟอร์มหลาย ๆ แพลทฟอร์ม ซึ่งเมื่อก่อนนี้สิ่งนี้ไม่สามารถทำได้ (เช่น DB2 สำหรับ VMS, AIX/6000 และ OS/2; SQL/DS สำหรับ VM และ VSE/WSA; และ OS/400 ซึ่งตามมาตราฐานดังที่กล่าวนี้ทำให้เราแน่ใจได้ว่าฐานข้อมูลที่แตกต่างกันจะสามารถทำงานร่วมกันได้

ผลิตภัณฑ์ประเภท Middleware เช่นพวกไดรเวอร์มาตราฐานบน ODBC (Open Database Connectivity) ของบริษัท Microsoft และ IDAPI ของบริษัท Borland ได้ทำการเพิ่มไดรเวอร์ที่ทำให้ดาต้าเบสเซอร์เวอร์สามารถทำงานร่วมกันได้ ดังนั้นจึงแน่ใจได้ว่าการติดต่อสื่อสารระหว่างโปรแกรม DBMS นั้นเชื่อถือได้ SQLBase Server ของ Gupta และ Supra Server ของ Cincom ได้เพิ่มไดรเวอร์เหล่านี้เข้าไปในโปรแกรมประเภท DBMS ด้วย นอกจากนั้นผลิตภัณฑ์ประเภท Middleware นี้ค่อนข้างช้า และมีข้อจำกัดบางอย่าง แต่ก็จะได้ว่าเป็นเทคโนโลยีใหม่ในขณะนี้

ผลิตภัณฑ์ของ Gateway ซึ่งได้รวมผลิตภัณฑ์และการบริการจาก Micro Decisionware และ Teradata นั้นมีแนวโน้มว่าจะมีความเร็วที่เพิ่มขึ้น โดย Gateway ได้ให้สายเชื่อมต่อราคาแพงกับผลิตภัณฑ์และแพลตฟอร์ม DBMS ที่มีชื่อเสียงหลายตัว รวมทั้ง DB2 และ AS/400

องค์ประกอบที่ 3 แต่เดิมนั้นผู้ใช้เครื่องเมนเฟรม มินิคอมพิวเตอร์ และพีซีไม่สามารถทำงานร่วมกันได้ เช่นเดียวกันกับผู้ใช้ระบบ LAN และ WAN ซึ่งไม่ได้ใช้ทรัพยากรของระบบในลักษณะเดียวกัน ไม่ได้แก้ปัญหาด้วยวิธีการเดียวกันและไม่ได้ใช้ภาษาในการสื่อสารเดียวกัน แต่ใน

ระบบฐานข้อมูลแบบกระจายนี้ผู้ใช้เหล่านี้จำเป็นต้องมีการทำงานร่วมกันและต้องทำการติดต่อสื่อสาร ในการจัดการนั้นจำเป็นขั้นตอนเพื่อให้แน่ใจว่าผู้ใช้ทุก ๆ คนจะสามารถทำงานร่วมกันได้

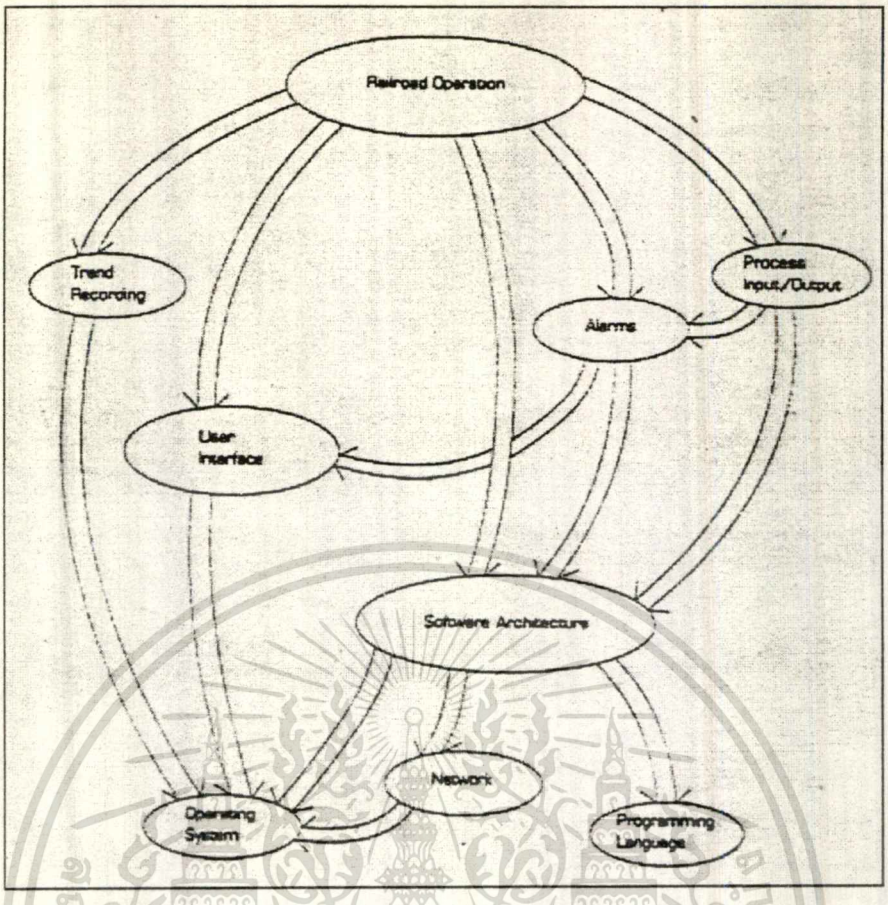
องค์ประกอบที่ 4 เป็นองค์ประกอบที่สร้างความยุ่งยากมากที่สุด คือเป็นการทำ Troubleshooting ของฐานข้อมูลแบบกระจายในระบบแบบกระจายขนาดเล็กที่สุดนั้น ฐานข้อมูลแบบกระจายจะต้องใช้คอมพิวเตอร์อย่างน้อย 2 เครื่อง ระบบปฏิบัติการ 2 ระบบ ผลิตภัณฑ์ DBMS 2 ตัว เครือข่ายการสื่อสารและฐานข้อมูลรวมทั้งโปรแกรมประยุกต์ด้วยปัญหาต่าง ๆ สามารถเกิดขึ้นได้ทุกที่ หรือในการรวมกันของแต่ละที่ก็อาจจะก่อให้เกิดปัญหาได้เช่นเดียวกัน อาจกล่าวได้ว่าจำนวนปัญหานั้นจะเพิ่มตามอัตราส่วนของเอ็กโปรเนนเชียลในขณะที่จำนวนของสถานีเพิ่มจำนวนขึ้นยิ่งกว่านั้น ปัญหาต่าง ๆ เหล่านี้อาจเกิดขึ้นเนื่องจากการใช้ Hardware และ Software จากผู้ผลิตหลายราย แต่ยังโชคดีที่ผู้ผลิตได้ตระหนักถึงความสำคัญของการทำงานร่วมกับลูกค้าดังนั้นผู้ผลิตหลายรายจึงได้ให้บริการเกี่ยวกับการออกแบบเพื่อช่วยแก้ปัญหาเหล่านี้

บทที่ 5
การวิเคราะห์เชิงวัตถุโดยทั่วไป
(Overview of Object-Oriented Analysis)

ในการพัฒนาระบบ Software ในรูปของ Object-Oriented จะต้องทราบถึงส่วนประกอบต่าง ๆ ของการที่จะวิเคราะห์ระบบในแบบ Object-Oriented ที่ส่วนใหญ่จะถูกแทนในรูปกราฟิกพอที่จะแบ่งได้ดังนี้

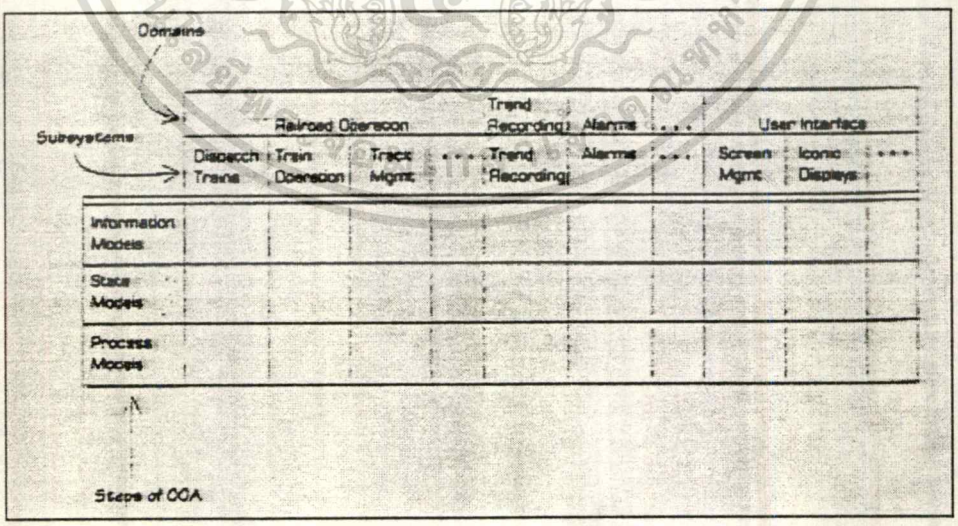
5.1. เริ่มต้นสำหรับการวิเคราะห์ (Setting Up for the Analysis)

ในการที่สร้างระบบ Software ขนาดใหญ่ เราจะต้องวิเคราะห์ถึงส่วนต่าง ๆ ที่สามารถจะแบ่งแยกตามหน้าที่ของแต่ละส่วนหรือจะเรียกส่วนต่างเหล่านี้รวมกันว่า Domain แต่ละ Domain สามารถสามารถแบ่งได้ตามหน้าที่ต่าง ๆ หรือ Object ตัวอย่างที่จะแสดงต่อไปนี้เป็นระบบควบคุมการเดินรถไฟ



ภาพ 5 เป็น Domain ชาร์ต สำหรับระบบการบริหารรถไฟ

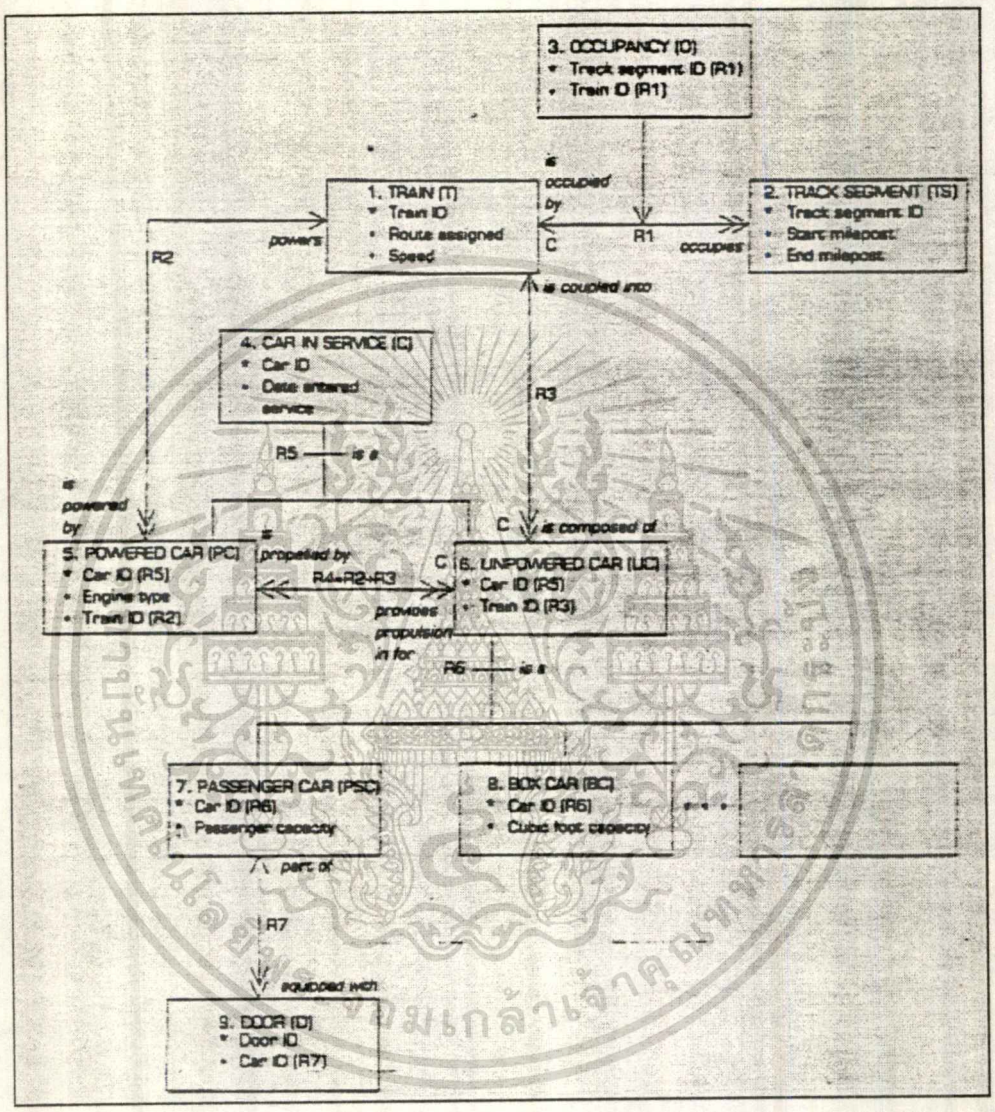
บาง Domain จะมีขนาดเล็กไม่พอที่จะแจกแจงรายละเอียดจึงจำเป็นจะต้องมีการแบ่งแยกย่อยในส่วนของ Domain นั้น ๆ ในรูปของ Subsystems โดยจะแสดงได้ดังในรูป



ภาพ 6 เป็นภาพของ Project Matrix

5.2. โมเดลสารสนเทศ(Information Models)

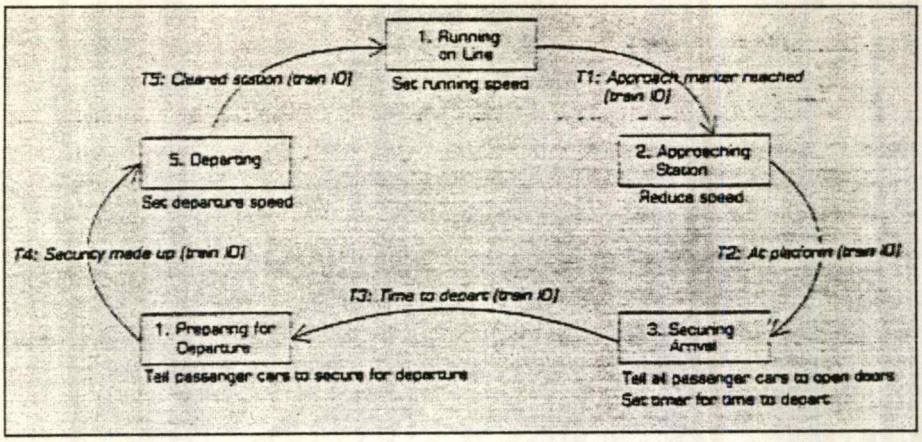
จุดประสงค์ของ Information Models คือขั้นต้นของการกำหนดนิยามของ Object ต่าง ๆ ที่ถูกสร้างขึ้นมาจาก Subsystem ภายใต้การวิเคราะห์ Object ต่าง ๆ จะแสดงในรูป ซึ่งจะเป็นการบอกถึงคุณสมบัติของแต่ละ Object และความสัมพันธ์ระหว่าง object



ภาพ 7 เป็นภาพของ Information Model

5.3. โมเดลสถานะ(State Models)

ในขณะนี้เมื่อเราได้ความสัมพันธ์ระหว่าง Object แล้วเราจะนำความสัมพันธ์ของ Object ที่มีวงจรชีวิตของ Object มาทำการแยกแบ่งเป็นลักษณะการทำงานของแต่ละ Object



ภาพ 8 เป็นภาพที่แสดง State model สำหรับวัตถุรถไฟ

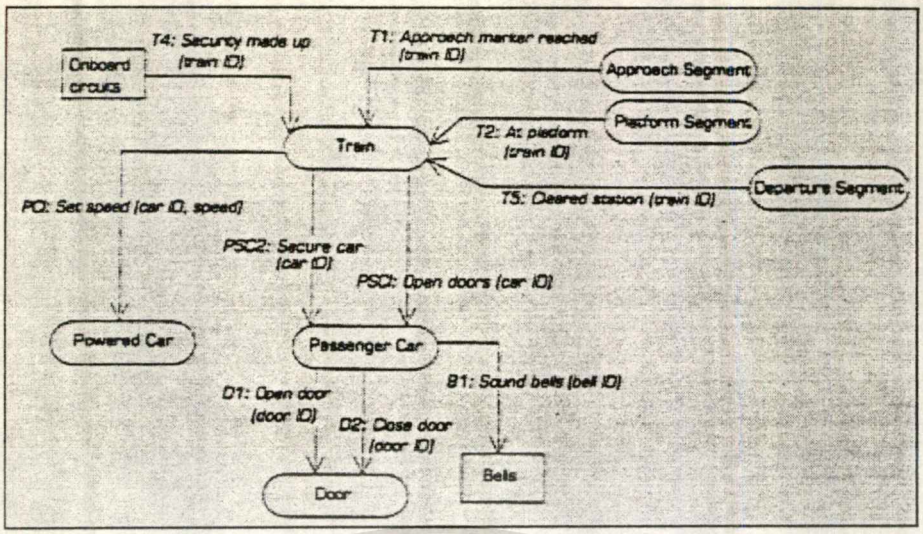
ตัวอย่างเช่นรถไฟที่วิ่งทางยาวจะต้องจอดรับคนในแต่ละสถานีจะต้องมีรูปแบบ(Platform) ดังนี้รถไฟจะต้องหยุดก่อนประตูจึงจะเปิดออกและจะมีการสั่นกระดิ่งเพื่อเตือน ผู้โดยสารและสุดท้ายก็จะถูกปิดก่อนที่รถไฟจะเคลื่อนออกจากสถานี

ดังนั้นวงจรชีวิตในรูปของ State Models จะประกอบไปด้วย 2 ส่วน คือ state และ event

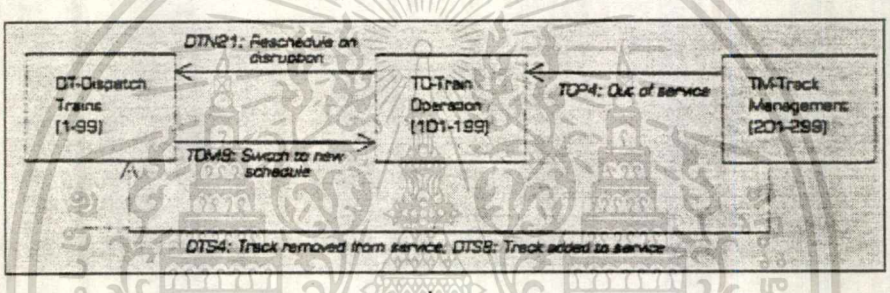
1) state จะแสดงเงื่อนไขของ Object ที่เป็นไปตามกฎและข้อตกลงต่าง ๆ
2) event จะแสดงถึงเหตุการณ์ต่าง ๆ ที่เป็นและให้ Object นั้นเคลื่อนที่จาก State หนึ่งไปยัง State อื่น ๆ

การแบ่งแยก State Models จะถูกสร้างขึ้นในทุก ๆ Object และความสัมพันธ์ของ Object นั้น รูปจะแสดงถึง State Model ของ Train Object ในแต่ละ State ก็จะมีส่วนที่อยู่ในรูปของการกระทำ (action) เมื่อวัตถุเข้ามายัง State

ในรูปแบบของ State Models นั้นจะยอมให้มีการติดต่อสื่อสารจากภายนอกเข้ามามีส่วนเกี่ยวข้องด้วยในรูป จะเป็นการแยกการสื่อสารของ Object ที่ถูกสร้างในแต่ละ Subsystem ในแต่ละ Subsystem ใน Domain เดียวกันก็จะมีโมเดลที่ติดต่อสื่อสารระหว่าง Subsystem ดังแสดงในรูป



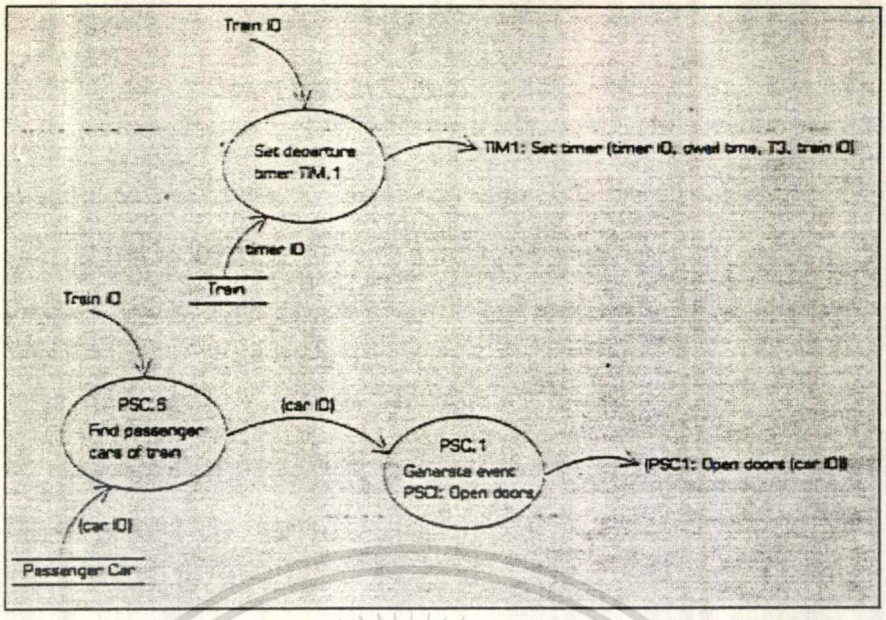
ภาพ 9 การติดต่อสื่อสารของวัตถุ



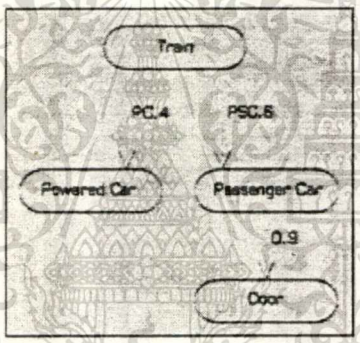
ภาพ 10 การติดต่อสื่อสารของ Subsystem

5.4. โมเดลประมวลผล (Process Models)

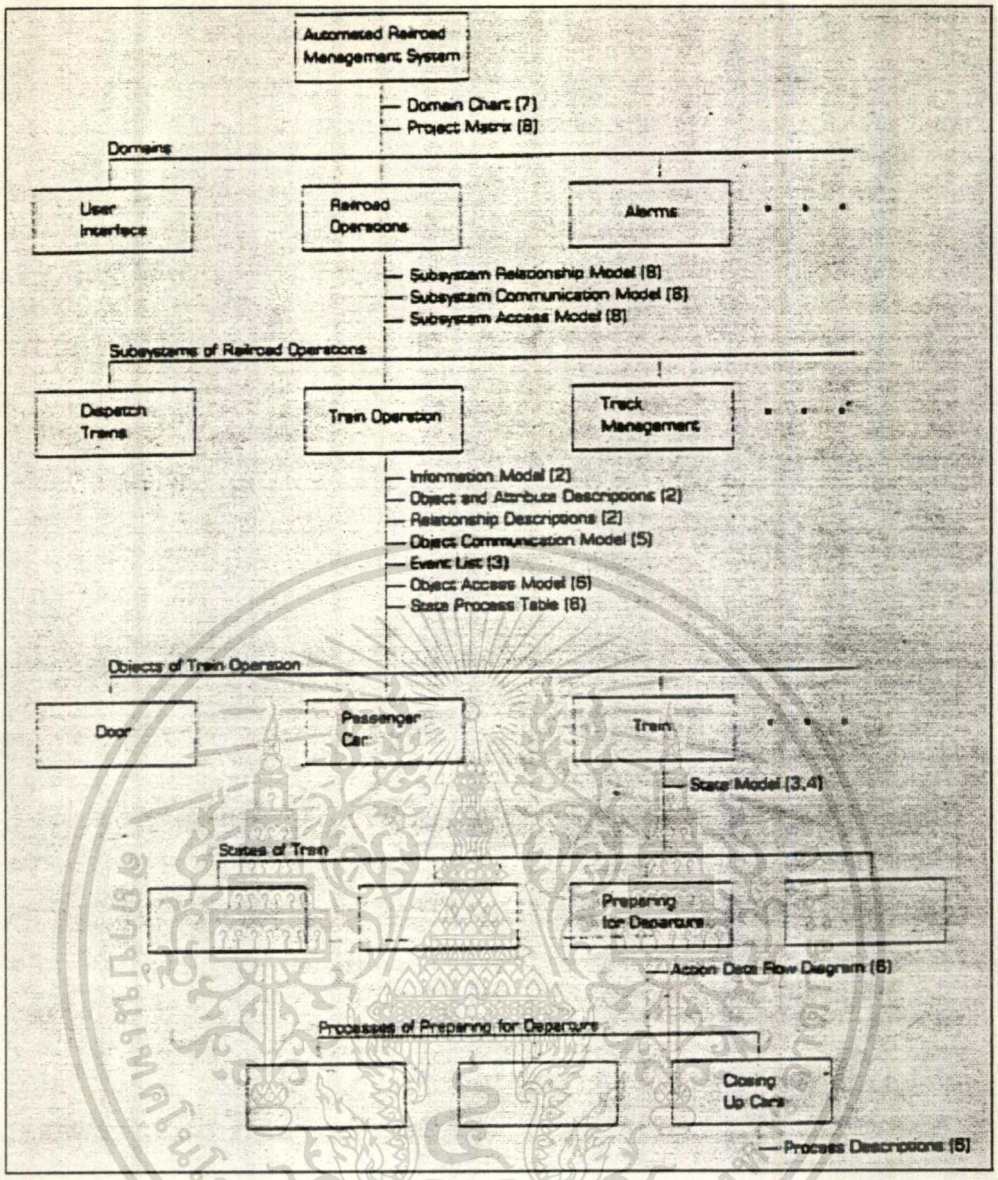
ทุก ๆ การประมวลผลในระบบจะประกอบไปด้วยการกระทำของ State Model แต่ละการกระทำในรูปของการประมวลผลและการเก็บข้อมูลของ Object ในการประมวลผลจะมีการทำงานของวัตถุนั้นและการเก็บข้อมูลของ Information Model แต่ละการกระทำจะแสดงในลักษณะรูปภาพในรูปของ action data flow diagram (ADFD) ดังแสดงในรูป ซึ่งจะถูกแยกแต่ละ State Model



ภาพ 11 Action data flow diagram



ภาพ 12 Partial object access model



ภาพ 13 งาน OOA สำหรับระบบบริหารรถไฟ

5.5 ตัวอย่างการวิเคราะห์โปรแกรมประยุกต์ระบบร้าน VDO

ข้อเท็จจริง

ในร้านวีดิโอจะประกอบไปด้วยข้อเท็จจริงที่สำคัญดังนี้

สมาชิกผู้เช่า

รายการ VDO

รายการที่จะต้องเกิดขึ้นเป็นประจำในร้าน VDO

การเช่า VDO จะประกอบไปด้วยกิจกรรมดังนี้

รายชื่อสมาชิก

ทำการตรวจสอบรายชื่อสามชิกว่าเป็นสมาชิกเก่าหรือเป็นสมาชิกใหม่

รายชื่อ VDO

ทำการตรวจสอบรายการวิดิโอที่ต้องการเช่า

ทำรายการเช่า

บันทึกรายการจากรหัสสมาชิกและรหัส VDO ลงในตารางเช่า

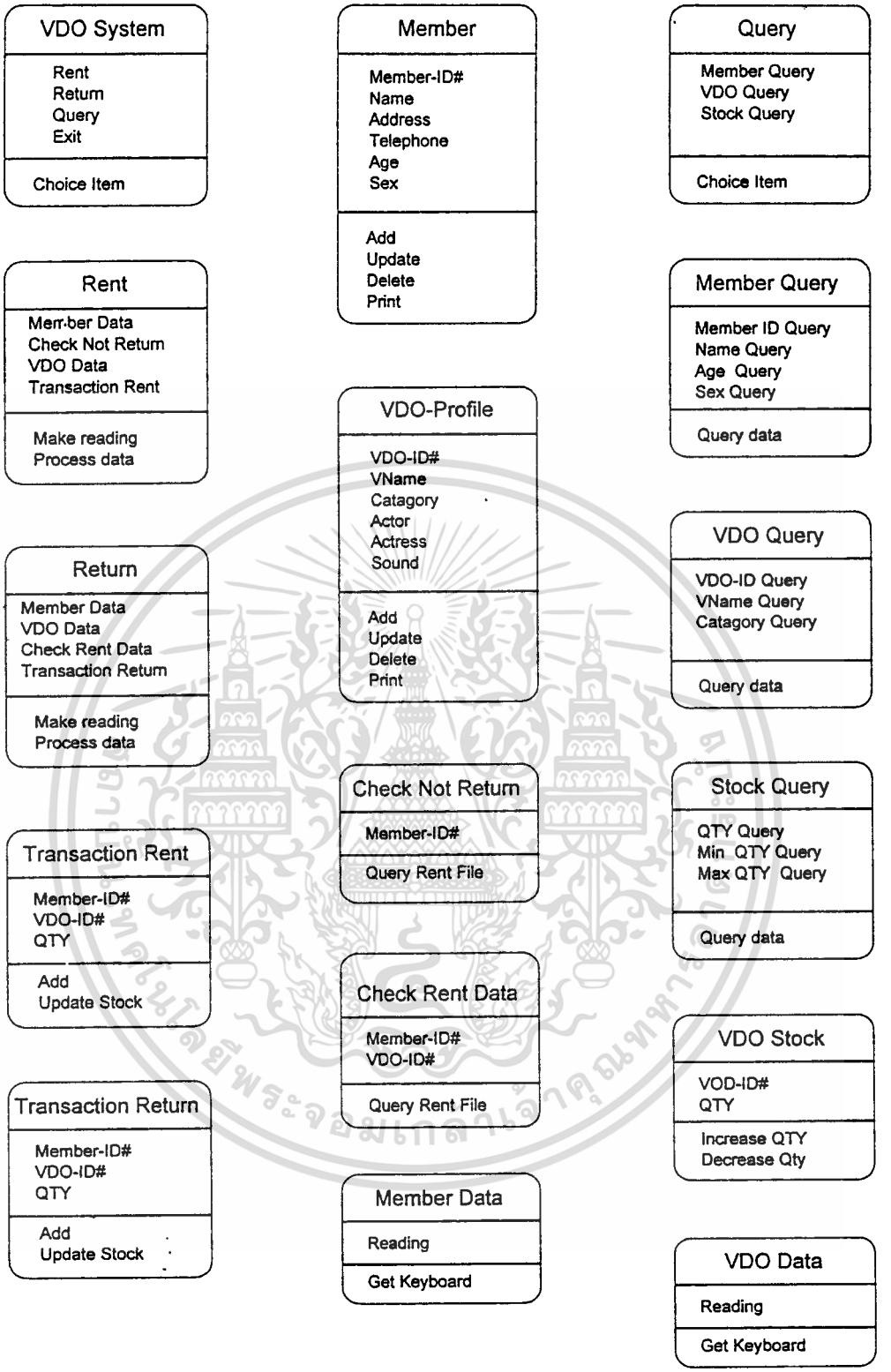
การคืน VDO จะประกอบไปด้วยกิจกรรมดังนี้

ตรวจสอบรายการเช่า

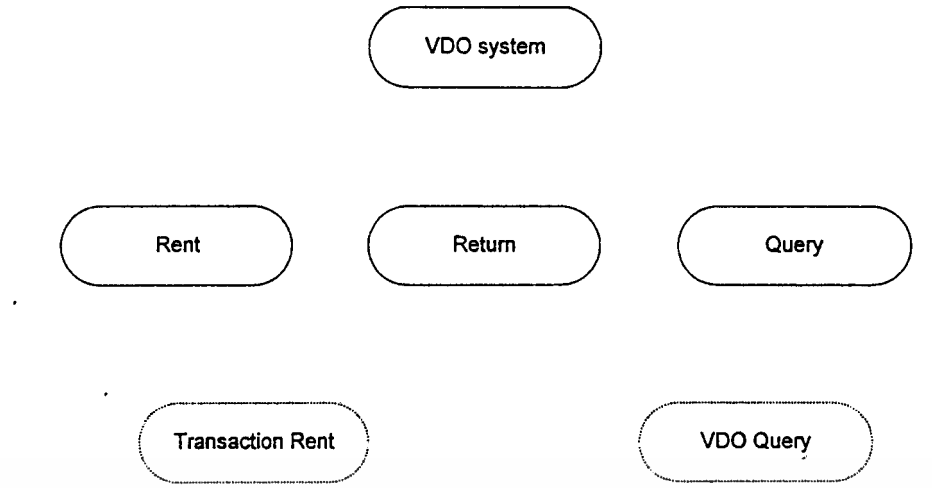
ทำการตรวจสอบถึงข้อมูลที่เช่าจากตารางเช่า

ทำรายการคืน

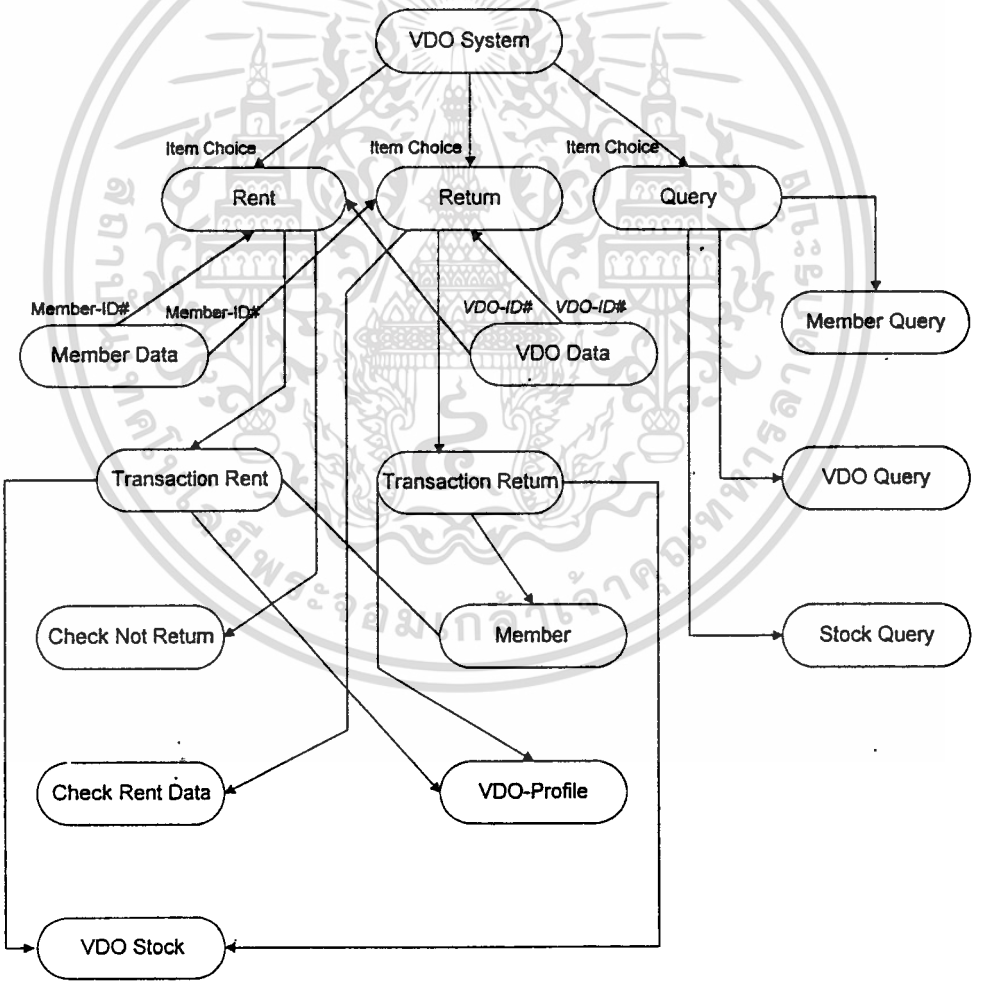
บันทึกรายการจากรหัสสมาชิกและรหัส VDO ลงในตารางคืนและลดจำนวนในตรางเช่า



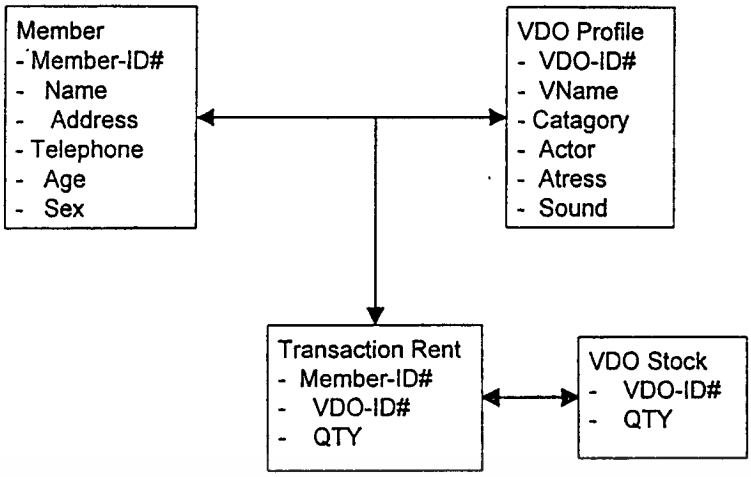
ภาพ 14 Object **ที่เกิดในร้าน** VDO



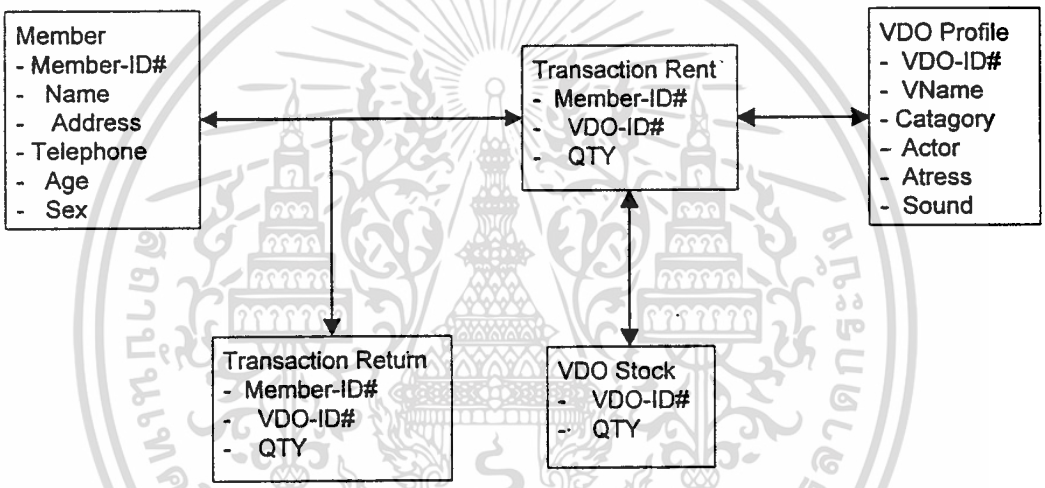
ภาพ 15 เป็นการแสดงระดับของคลาส VDO



ภาพ 16 แสดงความสัมพันธ์ระหว่างคลาสและซับคลาส



ภาพ 17 Rent Operation



ภาพ 18 Return Operation

ตารางฐานข้อมูลของระบบ VDO

Member Table

Member-ID#	Name	Address	Telephone	Age	Sex

VDO Table

VDO-ID#	Vname	Catagory	Actor	Actress	Sound

Transaction Rent Table

Member-ID#	VDO-ID#	QTY

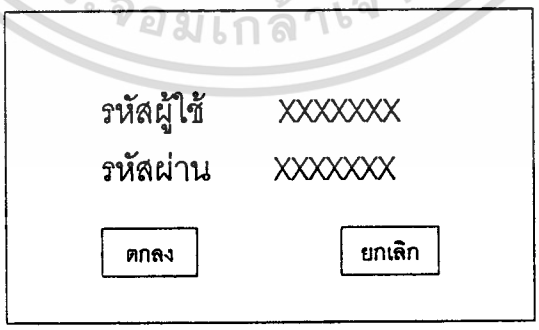
Transaction Return Table

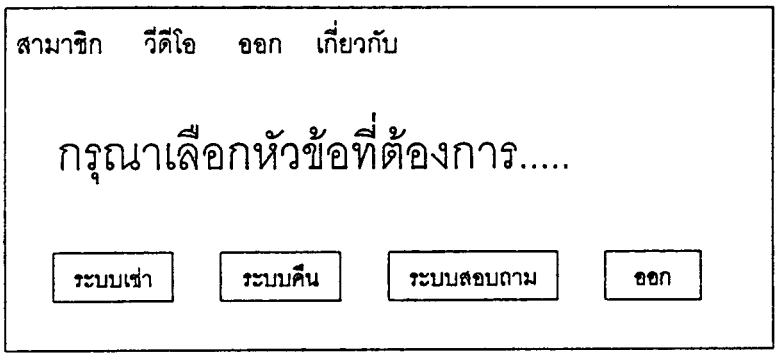
Member-ID#	VDO-ID#	QTY

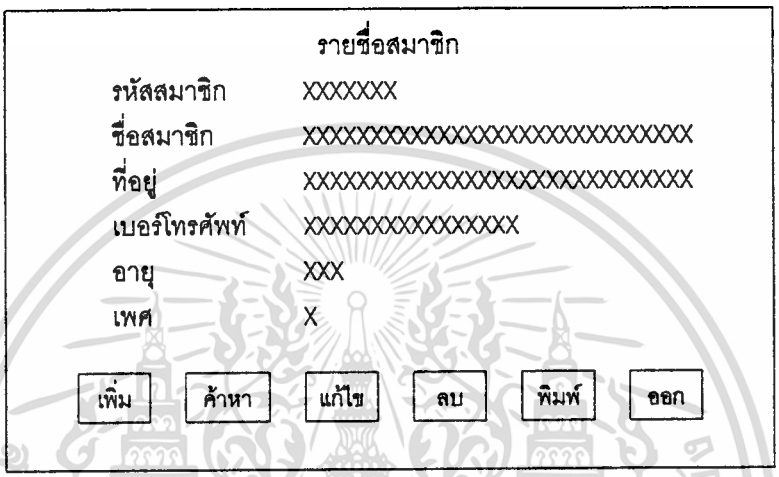
VDO Stock Table

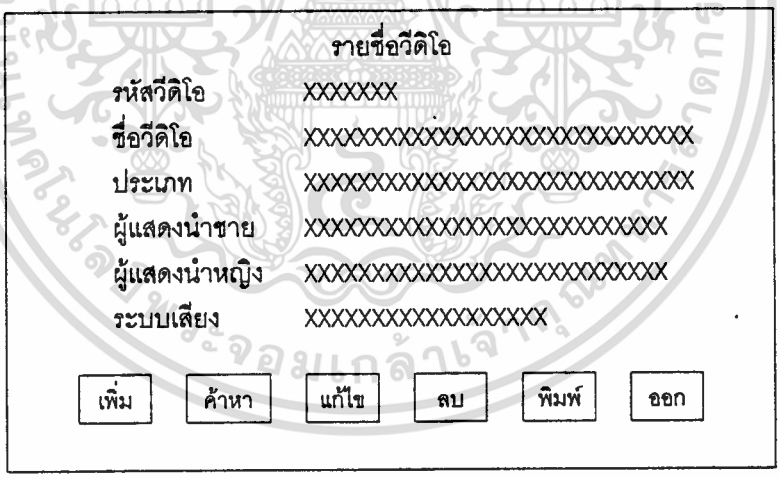
VDO-ID#	QTY

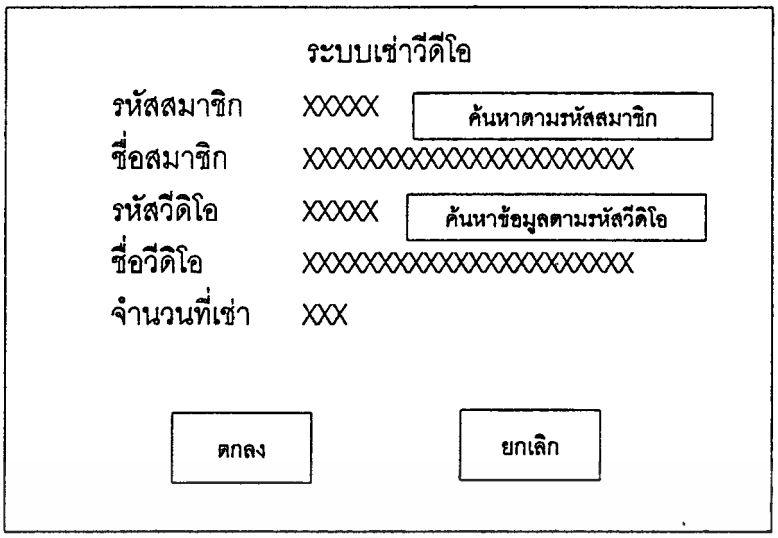
การออกแบบหน้าจอ

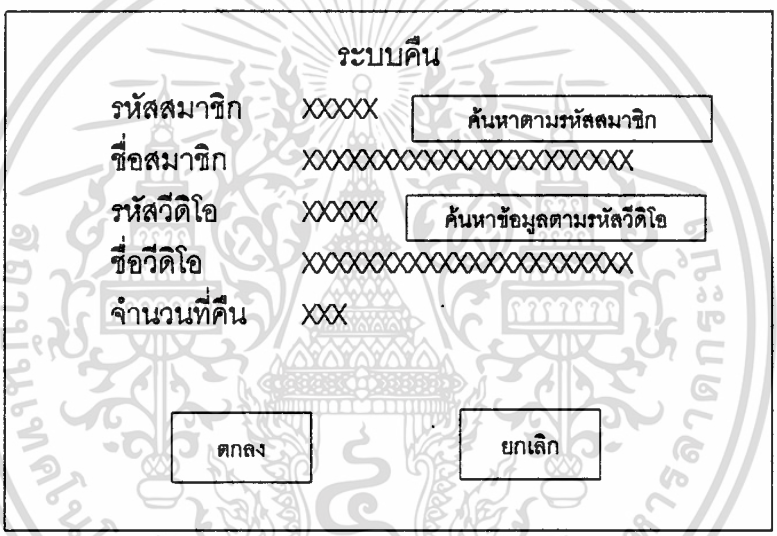












ส่วนที่ 3
ระบบฐานข้อมูลแบบ Client-Server

บทที่ 6
GUPTA SQL Client-Server RDBMS และเครื่องมือในการพัฒนา

สถาปัตยกรรมแบบไคล์แอนด์-เซอร์เวอร์ (Client-Server) เป็นการรวมเอาประโยชน์ต่าง ๆ ของซอฟท์แวร์ที่จัดการฐานข้อมูลเอาไว้ด้วยกัน ทำงานบนฮาร์ทแวร์ที่เป็นตัวหลักซึ่งบางทีอาจจะเป็นมินิคอมพิวเตอร์ หรือเมนเฟรม พร้อมด้วยสภาพแวดล้อมที่ใช้งานง่ายแบบกราฟฟิค เช่น วินโดว์ เนื่องจากซอฟท์แวร์ที่เป็นตัวจัดการฐานข้อมูลแบบไคล์แอนด์-เซอร์เวอร์เกือบทั้งหมดใช้ภาษาเอสคิวแอล (SQL) เครื่องมือที่ใช้ทำงานกับฐานข้อมูลที่ client เราเรียกว่า เอสคิวแอล **ฟรอนท์เอนท์** (SQL front-ends)

SQLWindows ในปัจจุบันนี้จะมีชุดที่แยกและชุดรวมผลิตภัณฑ์ไว้ด้วยกัน โดยในชุดผลิตภัณฑ์นี้ไม่เพียงแต่จะมีเพียงแค่ชุดพัฒนาโปรแกรมประยุกต์เท่านั้น แต่ยังประกอบไปด้วย SQLBase ตัวขับเคลื่อนฐานข้อมูลสำหรับวินโดว์; Quest และ Quest Reporter ฟอร์ม, รายงาน และเครื่องมือช่วยในการออกแบบการค้นคืน (query); ReportWindows ตัวออกแบบรายงานที่สามารถนำข้อมูลมาประกอบได้จากหลายแหล่ง และ TeamWindows เป็นเครื่องมือที่มีประสิทธิภาพมากในการจัดการพัฒนาโปรแกรมประยุกต์และการเก็บข้อมูลบนฐานข้อมูลแบบ ไคล์แอนด์-เซอร์เวอร์

ส่วนของซอฟท์แวร์ที่ทำให้ SQLWindows เข้าถึงข้อมูลในฐานข้อมูลของตัวแม่ได้เราเรียกว่า เราเตอร์ (router) ซึ่งแยกตามฐานข้อมูลที่เราต้องการเข้าถึง ซอฟท์แวร์เราเตอร์จะถูกรวมเข้ากับชุดของ SQLWindows เวอร์ชั่น 5.0 โดยไม่แยกเหมือนก่อน

6.1 ตัวขับเคลื่อนฐานข้อมูลและระบบโครงข่ายที่สนับสนุน

สามารถเลือกฐานข้อมูลที่สนับสนุนโดยผ่านซอฟท์แวร์ SQLNetwork

- GUPTA SQLBase Server
- IBM DB2
- IBM AS/400
- IBM OS/2 Database Manager
- Oracle
- Sybase SQL Server
- Microsoft SQL Server

- Informix
- Ingres
- Cincom Supra
- HP ALLBASE/SQL

และโดยผ่านมาตรฐานการเชื่อมต่อฐานข้อมูลเปิด Open Database Connectivity (ODBC)

- dBASE
- Paradox
- FoxPro
- Access
- Oracle
- Rdb
- AS/400
- DB2
- และอื่น ๆ

6.1.1 ระบบปฏิบัติการที่สนับสนุน

ระบบปฏิบัติการ	เวอร์ชั่น
DOS	PC-DOS or MS-DOS 3.1 หรือสูงกว่า
OS/2	OS/2 2.1 หรือสูงกว่า
NetWare	NetWare 3.11, 4.0 หรือสูงกว่า; NetWare Workstation for Windows
Sun Unix	Sun OS 4.1.1 หรือสูงกว่า, TCP/IP บน client
UnixWare	UnixWare 1.1.1 หรือสูงกว่า
Windows NT	Windows NT or Windows NT Advanced Server
Windows	Windows 3.1

6.1.2 ระบบเครือข่ายที่สนับสนุน

ชื่อระบบปฏิบัติการเครือข่าย	Protocols
3Com+3Open	NetBIOS
Banyan Vines	SPX/IPX
IBM LAN Server	TCP/IP
Microsoft LAN Manager	
Novell Netware	
Novell 3.1 or higher	

6.1.3 มาตรฐานที่สนับสนุน

สำหรับการเข้าถึงและแลกเปลี่ยนข้อมูล	สำหรับรายงานและฟอร์มของ e-mail
ODBC	VIM
DDE	MAPI
OLE	MHS

6.2 การจัดการวัตถุ (object)

ในการเลือก SQL front end (หรือ Package ในการพัฒนา โปรแกรมประยุกต์ อื่น ๆ) บริษัทจะต้องพิจารณาไม่เพียงแต่ลักษณะซึ่งสามารถเข้ากับ โปรแกรมประยุกต์ และให้ความสะดวกสบายสำหรับจัดการขั้นตอนในการพัฒนาด้วย เนื่องจากความต้องการของ โปรแกรมประยุกต์ มีมากขึ้น คุณจึงต้องการความสะดวกในการปรับงานของผู้พัฒนาหลาย ๆ คน และจัดการ Module ที่ต่างกันใน โปรแกรมประยุกต์ และเก็บส่วนประกอบในทางที่สามารถทำให้มันใช้ได้อีกกับโปรแกรมประยุกต์ ในอนาคต

Object เป็นส่วนประกอบที่สร้าง โปรแกรมประยุกต์ วัตถุหนึ่งสามารถเป็นอะไรก็ได้จากรูปแบบที่ซับซ้อนจนถึงรูปแบบที่เฉพาะ ซึ่งเป็นจะประกอบไปด้วย Field หรือ ปุ่มคำสั่งใน Object Oriented Programming นั้น ตัว Program Code จะเกี่ยวข้องกับวัตถุ เช่น ภาระกิจประจำเกิดขึ้นเมื่อกดปุ่ม ซึ่งวัตถุเป็นแบบการรวมตัวกันกล่าวคือมันถูกรวมเป็นส่วนของวัตถุ Object View จะเก็บ Program Code ที่เกี่ยวข้องกับรูปแบบวัตถุใน Physical file ที่แยกจากตัว From แต่ SQL จะรวม Object Code ไว้ด้วยกัน

ประสิทธิภาพของผู้พัฒนาจะดีขึ้นเมื่อวัตถุถูกพัฒนาครั้งหนึ่งและสามารถใช้ได้หลายครั้ง ทั้งในรูปแบบเดิม ๆ หรือมีการเปลี่ยนแปลงเล็กน้อย ขั้นตอนเบื้องต้นของการนำมาใช้ใหม่ของวัตถุ คือการทำสำเนาและการวาง โดยใช้ Windows Clipboard SQL Windows มีความสะดวกสบาย และมีประสิทธิภาพในการทำสำเนาวัตถุมาก ในการเลือกวัตถุหนึ่งในมุมมองแบบหยาบ จะสามารถเลือกวัตถุที่อยู่ภายใต้วัตถุที่เลือกได้ รวมถึงวัตถุที่เกี่ยวข้อง และ Program Code ที่เกี่ยวข้อง คุณสามารถที่จะ การทำสำเนา หรือ การทำสำเนา การเลือกทั้งหมดภายในหน้าต่างเดียวกัน หรือไปยังหน้าต่างที่ต่างกันใน โปรแกรมประยุกต์ หรือ ใน โปรแกรมประยุกต์ อื่น ๆ การเลือกวัตถุในมุมมองการสร้างจะเลือกรายการเดียวกันได้เหมือนเลือกอยู่ในแบบมุมมองแบบหยาบ

6.3 ชั้นของวัตถุ (Object Classes)

การใช้ class ของวัตถุจะทำให้การนำมาใช้ใหม่ของวัตถุง่ายขึ้น ใน class หนึ่ง ๆ นั้น เป็นแบบ ๆ หนึ่งของวัตถุ วัตถุทุกชิ้นของ class หนึ่ง ๆ จะมีลักษณะเบื้องต้นแบบหนึ่ง เครื่องมือทั้งหมดของการสร้างวัตถุจะมี class ถึงแม้มันจะไม่ได้กล่าวถึง ตัวอย่างเช่น ปุ่มนั้นเป็น class ๆ หนึ่ง วัตถุต่าง ๆ ของ class ปุ่มจะมีลักษณะแบบหนึ่ง เช่นการปรากฏของการถูกกดเมื่อปุ่มถูกกด เป็นต้น แนวคิดของ class คือประโยชน์ที่มากที่สุดเมื่อคุณสามารถสร้าง class ของคุณเองจาก class ที่มีอยู่

class ๆ หนึ่งจะขึ้นอยู่กับ class อีก class หนึ่งซึ่งมีการถ่ายทอดลักษณะของ class ซึ่งมันขึ้นอยู่ด้วย ตัวอย่างเช่นคุณอาจสร้าง class หนึ่งที่ชื่อว่า Next Button ซึ่งถ่ายทอดลักษณะของปุ่ม และมี Program Code ด้วยเพื่อจะเลื่อนรูปแบบหนึ่งไปยัง Record ต่อไป คุณอาจจะสร้าง class อีก class หนึ่งที่ชื่อ Master detail next button ซึ่งถ่ายทอดลักษณะทั้งหมดของ Next Button และสามารถเก็บ Program Code เพื่อจัดการรายละเอียดของ Record ร่วมกับตัว Master Record ลักษณะอีกอันหนึ่งของการถ่ายทอดคือถ้าลักษณะของ class เปลี่ยนไป class และวัตถุทั้งหมดที่ก่อนหน้านี้มีผลกับ class นั้น ๆ ก็จะเปลี่ยนแปลงไปอย่างอัตโนมัติ

SQL Windows สนับสนุนการสร้าง class กับความสามารถเพิ่มเติมที่เพียงพอ แทนที่จะสร้าง class และสร้างวัตถุจาก class นั้น คุณสามารถสร้างวัตถุและจัดเก็บวัตถุนั้นเป็น class ได้ ตัวอย่างเช่น คุณสารมารถสร้างปุ่มค้นหา และตัดสินในที่จะให้มันมีลักษณะถาวร เป็น class ที่สามารถใช้ได้เรื่อย ๆ SQL Windows สามารถให้การถ่ายทอดแบบซับซ้อนได้ ซึ่งวัตถุหรือ class สามารถถ่ายทอดลักษณะของ class หลาย ๆ class ได้

วัตถุและ class ที่ใช้หลายครั้งได้เป็นประโยชน์ในระยะยาวเมื่อมันถูกเก็บใน Library ที่แยกจาก โปรแกรมประยุกต์ SQL Windows จะให้คุณรักษา class ใน Library ซึ่งจะถูกใช้งานในโปรแกรมประยุกต์ โดยที่ทางแฟ้มข้อมูลทำงานในโปรแกรมภาษา C

6.4 การบริหารงานกลุ่ม

ธุรกิจขนาดใหญ่จะมีกลุ่มผู้พัฒนาทำงานบน โปรแกรมประยุกต์ เดียวกัน มันจึงเป็นสิ่งจำเป็นสำหรับการป้องกันมันให้ใครจะเปลี่ยนแปลงข้อมูลของใครได้ ในทางกลับกัน การเปลี่ยนแปลงโปรแกรมประยุกต์ จะหนีไม่พ้นปัญหาใหม่ ๆ และเพื่อทำงานอย่างราบรื่น มันจำเป็นต้องมีการติดตามขั้นตอนการพัฒนาก่อนที่จะมีปัญหาเกิดขึ้น

SQLWindows สนับสนุนการนำเข้าและตรวจสอบการส่งออกของข้อมูลในแต่ละส่วนของโปรแกรมประยุกต์ เมื่อ โปรแกรมประยุกต์ หรือส่วนประกอบถูกตรวจสอบออก ในลักษณะเดิม ๆ จะถูกรักษาไว้ในที่เก็บศูนย์กลางซึ่ง Lock ไว้ในขณะที่ผู้พัฒนาทำงานบนตัวสำเนา ในที่เก็บอีกที่ตัวเดิมยังถูกเก็บในลักษณะอ่านอย่างเดียว สำหรับการทดสอบโปรแกรมประยุกต์หรือทำ Executable แต่ไม่มีผู้พัฒนาคนไหนสามารถแก้ไขได้จนกว่าจะมีการตรวจสอบเข้า

SQL Windows จะรวมการควบคุม Version ภายใน การรักษา Version หลาย ๆ ตัวของโปรแกรมประยุกต์ และส่วนประกอบเหมือนการมองใหม่

6.5 Client-Server Repository

ฐานข้อมูล Client/Server จะทำที่เก็บเฉพาะของฐานข้อมูลสำหรับ โปรแกรมประยุกต์ สำหรับส่วนประกอบที่สร้าง โปรแกรมประยุกต์ และสำหรับวัตถุที่ใช้ได้หลายครั้ง ฐานข้อมูลจะดีกว่า Tree ของระบบเครือข่าย ในการจัดเก็บและจดการสภาวะการพัฒนา ฐานข้อมูล Client/Server class นำสามารถให้ที่เก็บของวัตถุแบบ Binary ในตารางฐานข้อมูล และให้คุณเก็บไม่เพียงแต่ข้อมูลเกี่ยวกับ โปรแกรมประยุกต์ แต่เก็บ โปรแกรมประยุกต์ ด้วยโดยใช้ Field วัตถุแบบ Binary สำหรับส่วนประกอบเหมือน Source Code Module หรือการออกแบบรูปแบบ

หน้าต่าง Team ของ SQL Windows สามารถใช้ที่เก็บฐานข้อมูลแบบ Client/Server ด้วยโดยใช้ Field วัตถุแบบ Binary สำหรับส่วนประกอบเหมือน Source Code Module หรือ การออกแบบรูปแบบ

หน้าต่าง Team ของ SQL Windows มีการใช้งานที่จัดเก็บอย่างมีประสิทธิภาพ รวมถึงการเก็บส่วนประกอบที่ใช้สร้าง โปรแกรมประยุกต์ Team Windows จะเก็บข้อมูลของ Data Dictionary

เกี่ยวกับฐานข้อมูลขึ้นอยู่กับ โปรแกรมประยุกต์ ถึงแม้ SQL Windows 4.1 ต้องการที่เก็บที่จะ Run ภายใต้ SQL Base มันก็สามารถอ่านและเก็บข้อมูลของ Data Dictionary จาก Engine Platform ข้อมูลจะถูกใช้ในรูบแบบการพัฒนาขึ้นอยู่กับโครงสร้างของข้อมูลซึ่งสุดท้ายจำเป็นในการใช้งาน Team Window รักษาข้อมูลเก่าก่อนเป็น Module ที่มีการทบทวนและพัฒนาและมันสามารถออกรายงานสถานะได้หลายแบบ

6.6 Deployment

เมื่อ โปรแกรมประยุกต์ ถูกพัฒนา มันจะถูกใช้บนระบบของผู้ใช้ ส่วนใหญ่ Software ที่ใช้พัฒนาจะถูกใช้เพื่อสร้าง โปรแกรมประยุกต์ และ Source File จะไม่ต้องการอีก การพัฒนาของ Runtime Executable ของสินค้าทั้งหมดและข้อตกลงทางลิขสิทธิ์ จะให้มีการแจกจ่าย อย่างไม่จำกัดของ Runtime โปรแกรมประยุกต์ files โดยไม่ต้องมีการขึ้นอยู่กับตัวใดตัวหนึ่ง

เพื่อที่จะให้ โปรแกรมประยุกต์ Executable หรือ Runtime Files ประกอบด้วย Software ของผู้ขายหลายแห่งเป็นสิ่งที่ต้องการของ โปรแกรมประยุกต์ ที่จะทำงาน SQL Windows ติดตั้ง Deployment files ไว้ใน Subdirectory แยกบนระบบพัฒนาเพื่อจะได้ ทำสำเนาได้ง่าย

6.7 SQL Decision

SQLWindows ง่านต่อการใช้ในการสร้างโปรแกรมประยุกต์เมื่อต้องการ ถ้ายิ่งความต้องการของโปรแกรมประยุกต์ยิ่งซับซ้อนเท่าใด SQLWindows ก็สามารถให้ตัวเลือกในการแก้ไขปรับกรุงและคำสั่งที่ชัดเจนในการใช้งานได้มากนั้น ความสามารถในการทำ Even Handing และส่งข้อมูลระหว่างรูปแบบนั้นดีมาก โดยมี TeamWindows ที่ให้ประโยชน์ในสภาวะแวดล้อม Client-server เพื่อจะให้ประโยชน์และประสิทธิภาพมากที่สุดสำหรับจัดการขั้นตอนของการพัฒนาได้

6.8 การ optimize สำหรับ front-end

SQLBase ได้รับการออกแบบให้ใช้กันได้กับ งานประเภทกราฟฟิก, งานด้านฐานข้อมูลที่ส่วน front-end เป็น Windows ซึ่งจะมีการดูข้อมูลในกรอบหน้าต่างที่เลื่อนขึ้นลงได้บ่อย ๆ ในขณะที่ระบบจัดการฐานข้อมูลส่วนใหญ่อนุญาตให้เลื่อนไปข้างหน้าได้เท่านั้น SQLBase สนับสนุนการเลื่อนเคอร์เซอร์ให้ไปข้างหน้าหรือกลับหลังก็ได้ในระดับของส่วนขับเคลื่อนฐานข้อมูล (database engine) เลยทีเดียว โดยผู้ใช้ Windows อยู่จะดูข้อมูลที่ถูกพักเอาไว้ โดยไม่มีการรบกวนความ

สามารถในการทำงานร่วมกัน (concurrence) และไม่มีปัญหาการแก้ไขที่สูญหาย (lost update problem) ด้วย

การสนับสนุนให้เลื่อนหน้าต่างไปข้างหน้าและกลับหลังได้นั้น นับเป็นสิ่งยุ่งยากอีกประการหนึ่งทีเดียว ส่วน back-end จะต้องจัดการ ถ้ามีการใช้ shared-lock ในขณะที่กำลังมีผู้ใช้ดูข้อมูลอยู่ ผู้ใช้คนอื่น ๆ ในโครงข่ายก็ไม่สามารถที่จะทำการแก้ไข (update) ข้อมูลได้ ดังนั้นความสามารถในการทำงานร่วมกัน (congruency) จะสูญเสียไป

SQLBase จัดการกับปัญหานี้โดยการสนับสนุนระดับที่แยกจากกัน 2 ระดับคือ Read Only และ Release Locks ในการทำงานในระดับ Read Only จะไม่มีการ lock ในระดับใด ๆ เลย ดังนั้นจึงให้ความสามารถในการทำงานร่วมกับสูงสุด ส่วนในระดับ Release Locks จะใช้ share lock ในขณะที่อ่านข้อมูลไปจนเสร็จ แล้วจึงคลาย lock ความเสี่ยงในการใช้ Release Locks ก็คือคนอื่นสามารถทำการแก้ไขข้อมูลที่กำลังดูอยู่ในกรอบหน้าต่างที่เลื่อนได้ ในกรณีที่แก้ไขข้อมูลส่วนนั้นด้วย การแก้ไขของคุณก็จะไม่มีผล

SQLBase ได้สร้างคุณสมบัติในการเตือนและป้องกันการแก้ไข เพื่อที่จะจัดการกับจุดอ่อนตรงนี้ แต่ละแถวจะมีการกำหนดรหัสประจำแถว (row ID) เพื่อช่วยเพิ่มความเร็วในการเข้าไปแก้ไขแถวนั้น และเพื่อตรวจสอบว่าแถวนั้นมีการแก้ไขหลังจากที่มันถูกอ่านไปครั้งสุดท้ายหรือไม่ เพื่อที่จะอาศัยประโยชน์จากคุณสมบัตินี้ผู้พัฒนาระบบต้องใช้รหัสประจำแถวในการบอก SQLBase ว่าแถวใดที่ได้รับการแก้ไข ถ้าแถวนั้นมีการเปลี่ยนแปลงไปแล้ว SQLBase จะป้องกันไม่ให้มีการแก้ไขเกิดขึ้นและจะเตือนโปรแกรมประยุกต์ว่ามีความขัดแย้ง (conflict) เกิดขึ้นในการแก้ไขโดยจะไม่ใช้วิธีการยกเลิกการแก้ไข

SQLBase จะมีการเก็บค่าต่าง ๆ ที่อยู่รอบ ๆ เคอร์เซอร์เอาไว้ เนื่องจากจอภาพ [ซึ่งมีอัตราการวาดใหม่ (refresh) ช้า] จะได้ไม่ต้องถูกวาดใหม่บ่อย ๆ เมื่อมีการทำคำสั่ง COMMIT ข้อมูล (ที่ถูกแก้ไข) ก็จะถูกแสดงออกมา และทั้ง share และ exclusive lock ก็จะถูกปลดออก ในขณะเดียวกัน เคอร์เซอร์ทุกตัวก็ยังคงอยู่ที่ตำแหน่งเดิมเหมือนตอนก่อนมีการ commit รอให้มีการเลื่อนเพื่อจัดการกับแถวอื่น ๆ ต่อไป ข้อดีอีกอย่างหนึ่งก็คือ แอพลิเคชันไม่จำเป็นต้องมีการดึงแถวทั้งหมดกลับโดยผ่านทางโครงข่ายอีกครั้งหนึ่งเพื่อที่จะให้ไปอยู่ในสภาวะก่อนมีการ COMMIT ดังนั้นจึงลดงานเขียนโปรแกรมและลด overhead ลงไปได้มาก ซึ่งการทำงานเมื่อมีการ rollback ก็จะมีลักษณะเช่นนี้เช่นกัน

6.9 คุณสมบัติที่เป็นจุดเด่น

SQLBase ไม่ได้สนับสนุนการใช้งาน triggers แต่ว่ามันสนับสนุนการประกาศความถูกต้องในการอ้างอิง (referential integrity) ในรูปแบบของ DB2 SQLBase ได้ขยายคุณสมบัตินี้โดยสร้างทางเลือกขึ้นมา 3 ทางได้แก่ RESTRICT, SET NULL และ CASCADE โดยทางเลือก RESTRICT จะป้องกันไม่ให้มีการลบข้อมูลในกรณีที่เมื่อลบแล้วมันจะละเมิดความถูกต้องในการอ้างอิง (คือยอมให้ลบหรือแก้ไขได้ก็ต่อเมื่อเรคคอร์ดนั้น ไม่มี foreign key ที่มีค่าตรงกับมันเท่านั้น) SET NULL ก็จะไปแก้ไขเรคคอร์ดที่อ้างอิงตัวมัน (โดยทำให้เป็น NULL) ส่วนทางเลือก CASCADE จะตามไปลบหรือแก้ไข foreign key ที่ถูกลบหรือแก้ไขด้วย

ตัว optimize ของ SQLBase ที่ทั้งคุณสมบัติ cost-based และ syntax-based แต่มันนำสถิติมาใช้อย่างจำกัด (ใช้ index, จำนวน page, จำนวนของแถวในตาราง) ในกรณีของการ optimize ตามราคา (cost-based) และมันก็จะเขียน query ใหม่ด้วย เพื่อที่จะทำให้การ optimizer รวบรวมเข้ามาและข้อมูลเหล่านี้มันนำไปใช้ประโยชน์อีกที

SQLBase มีคุณสมบัติที่ล้ำหน้าอยู่หลายอย่างซึ่งรวมทั้งการสนับสนุน BLOB (Binary Large Object) เช่น bitmap หรือ sound object และการสนับสนุนการแบ็กอัพในขณะที่ระบบกำลังทำงานอยู่ (on-line backup) นอกจากนี้ยังมี API ไว้ใช้กับภาษา C และ COBOL รวมทั้ง precompiler สำหรับภาษา COBOL เพื่อใช้งานกับระบบงานที่ใช้ DB2 SQLBase สามารถเก็บ procedure ที่มีการแยกทาง (branching) ได้

การใช้เนื้อที่ของ log file SQLBase ได้อย่างมีประสิทธิภาพ การเปลี่ยนแปลงของฐานข้อมูลจะถูกเก็บไว้ในระดับ field ในขณะที่ฐานข้อมูลส่วนใหญ่จะเก็บไว้ในระดับ record นอกจากนี้ log file ของ SQLBase สามารถที่จะเก็บไว้ในดิสก์ที่แยกกับฐานข้อมูลได้ เพื่อที่จะลดความติดขัดจากข้อมูลมากเกินไป (contention) ตัวฐานข้อมูลเองก็สามารถที่จะเก็บแยกจากกัน (เก็บไว้ในไฟล์มากกว่า 1 ไฟล์ เพื่อกระจาย load ไปบนดีไวซ์หลาย ๆ ตัว และเพื่อที่จะเพิ่มความจุในการเก็บฐานข้อมูลขนาดใหญ่

นอกจากนี้ยังได้มีการปรับปรุงด้านการใช้หน่วยความจำของฝั่ง client เมื่อพิจารณา ส่วน kernel ฝั่ง server ซึ่งต้องการหน่วยความเพียง 700K และในการติดต่อกับฝั่ง client ในแต่ละครั้งต้องการหน่วยความจำเพิ่มเพียง 50K เท่านั้น สำหรับฝั่ง client ซึ่งครั้งหนึ่งเคยใช้หน่วยความจำ 200K แต่ปัจจุบัน software ในฝั่ง client เวอร์ชั่นใหม่ใช้หน่วยความจำเพียง 50K ในส่วนของหน่วยความจำส่วนล่าง (conventional memory) โดยอนุญาตให้ load ส่วนที่เหลือขึ้นไปไว้ที่หน่วยความจำ extended

Optimizer ของ SQLBase 5.1 เป็น Cost - Based Optimizer ทุก ๆ ครั้งที่มีการสร้างอินเด็กซ์ขึ้นมาใหม่ SQLBase จะทำการสร้างค่าสถิติต่าง ๆ ขึ้นมาโดยอัตโนมัติ แต่ว่าก็มีคำสั่ง Update Statistics เอาไว้ให้ DBA ใช้อัพเดตค่าสถิติของอินเด็กซ์เก่า ๆ สำหรับบางอินเด็กซ์ หรือสำหรับทุก

อินเด็กซ์ตาราง (Table) ที่ต้องการแต่คำสั่ง Update statistics SQLBase ไม่มีความสามารถในการสุ่มตัวอย่างข้อมูลเป็นส่วนเพื่อนำมาสร้างค่าสถิติที่ต้องการได้

ถ้าหากว่าคุณใช้คำสั่ง SET PLANONLY บน SQL เซตให้ PLANONLY มีค่าเป็น ON ก่อนที่จะรันคำสั่ง SQL SQLBase ก็จะวิเคราะห์หา Execution Plan แล้วแสดงค่าของ Execution Plan นั้นออกมาให้คุณเห็น โดยที่ไม่ได้เข้าถึงข้อมูลเพื่อตอบกับคำสั่ง SQL จริง ๆ ถึงเราจะสามารถขอดูขั้นของการดำเนินงาน Execution Plan ได้ SQLBase ไม่ได้เปิดช่องให้เราสามารถเบี่ยงเบนหรือแก้ไข Execution Plan เห็นสมควรโดยตรงได้

เนื่องจาก SQLBase ไม่ได้ใช้วิธีการเก็บเอาไว้เป็นโพรซีเจอร์ แต่จะเก็บเฉพาะ Execution Plan ของประโยคโดยอาศัยชื่อของประโยคคำสั่ง SQL ที่ผู้ใช้เป็นผู้ตั้ง ใช้เป็นในการอ้างอิงถึง Execution Plan ที่เก็บเอาไว้ (ด้วยคำสั่ง SQL และผู้ใช้สามารถเรียกใช้ประโยคคำสั่งนั้นซ้ำได้ด้วยคำสั่ง Execute โดยประโยคคำสั่ง SQL เหล่านั้นสามารถกำหนดตัวแปรไว้ได้ด้วย

SQLBase ไม่มีกลไกที่ช่วยทำให้ Execution Plan ของประโยคคำสั่ง SQL ที่เก็บเอาไว้มีความเหมาะสมกับสถานะการณ์เวลาใด ๆ เพราะฉะนั้นหากมีการเปลี่ยนแปลงของข้อมูลหรือฐานข้อมูลที่มีความสำคัญต่อการเข้าถึงข้อมูลเกิดขึ้น ก็ต้องเป็นหน้าที่ของใครสักคน (อาจจะเป็น DBA โปรแกรมเมอร์ หรือผู้ใช้) ที่จะต้องไปนั่งจัดการ Restore ประโยคคำสั่งเหล่านี้ลงไปใหม่ด้วยตัวเองเพื่อให้ Optimizer สามารถวิเคราะห์หา Execution Plan ใหม่อีกครั้งหนึ่ง

ใน SQLBase เราสามารถเลือกใช้อินเด็กซ์ได้ 2 ชนิดคือ B-Tree และ Hash Index ดังนั้น SQLBase จึงค่อนข้างสะดวกต่อการใช้งานได้หลาย ๆ แบบ

การบริหารระบบและการป้อนกลับ

SQLTalk ซึ่งมีทั้งเวอร์ชั่นที่เป็นแบบตัวอักษรและกราฟิก ช่วย DBA อย่างมากจากการใช้ระบบเมนู (menu-driven) ในการสร้างฐานข้อมูล, มอบอำนาจในการใช้งาน (permission) และช่วยทำหน้าที่ในการบริหารระบบอื่น ๆ อีกด้วย

6.10 GUPTA SQL client-server RDBMS tools

6.10.1 SQLBase

6.10.2 SQLWindows

6.10.3 Quest Reporter and Quest

6.10.4 SQLTalk

6.10.5 TeamWindows

6.10.6 ReportWindows

6.10.7 SQLConsole

6.10.8 SQLRouters

6.10.1 SQLBase 5.2

เป็น SQL RDBMS ที่มีหลาย platform และสามารถทำงานข้าม platform ได้ เช่น DOS, Netware, OS/2, UNIX และ Windows NT ภายในสภาพแวดล้อมแบบกราฟิก SQLBase สนับสนุนการติดต่อกับตัวขับเคลื่อนฐานข้อมูลและระบบเครือข่าย มากมาย ดังกล่าวมาแล้วข้างต้นนอกจากนี้ GUPTA SQL/API ยังได้เตรียมฟังก์ชันก็เป็นภาษา C ไว้ถึง 117 ฟังก์ชัน โดยทำงานภายใต้ Microsoft C เพื่อผู้พัฒนาโปรแกรมประยุกต์ด้วย C และ COBOL

6.10.2 SQLWindows

SQLWindows เป็น development system สำหรับการสร้าง multi-user C/S database โปรแกรมประยุกต์ สำหรับ windows, windows NT และ OS/2 workstation

ลักษณะของ SQLWindows คือ

เหมาะกับงานการเขียนโปรแกรมที่ต้องการการทำงานร่วมกันเป็นทีมสามารถที่จะเพิ่ม productivity ได้เนื่องจาก blocks ต่าง ๆ ในโปรแกรมประยุกต์ที่สร้างไว้สามารถนำมาใช้ใหม่ได้ มีการใช้โปรแกรมแบบ Object-Oriented ที่แต่ละ object สามารถนำมาใช้ใหม่ได้ สามารถสร้างโปรแกรมประยุกต์ได้โดยไม่ต้องรู้จัก SQL และแทบไม่ต้องเขียน code เลย

6.10.2.1 โปรแกรมประยุกต์ Designer

ใช้ในการออกแบบหน้าจอ forms และ objects ในการเพิ่มหรือ modify object ต่าง ๆ ใน โปรแกรมประยุกต์ designer programmer สามารถใช้วิธี point-and-click และ drag-and-drop ใน SQLWindows จะสร้าง code และ documentation ให้โดยอัตโนมัติ object ใน โปรแกรมประยุกต์ designer รวมถึงพวก image sound และ video สำหรับการสร้าง multimedia โปรแกรมประยุกต์ด้วย

โปรแกรมประยุกต์ Designer ประกอบไปด้วย

1. Palette ของ paining และ customizing tools

2. Context-Sensitive Help

3. QuestWindow เป็น graphical data access tools ใช้สำหรับการสร้าง โปรแกรมประยุกต์ โดยไม่ต้องเขียน code

4. TableWindow จะจัดการให้ data retinal, display และ updating มีลักษณะเป็นตารางโดยอัตโนมัติ ตัว TableWindow จะถูกจัดการโดย high-level function และจะช่วยลด programming effect ได้

6.10.2.2 โปรแกรมประยุกต์ Outliner

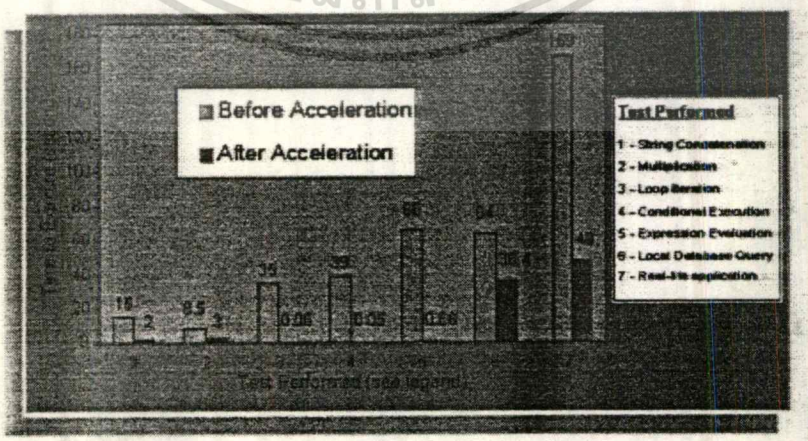
จะแสดง overall view ของส่วนประกอบต่าง ๆ ของ โปรแกรมประยุกต์ เมื่อมี โปรแกรมประยุกต์ object ถูกแก้ไขตามโดยอัตโนมัติ Outliner นี้จะทำให้การทำ document และการทำ navigate ของโปรแกรมประยุกต์ขนาดใหญ่ทำได้สะดวกขึ้น

6.10.2.3 SQLWindows โปรแกรมประยุกต์ Language(SAL)

SAL เป็นการ programming แบบ 4GL ซึ่งจะ support

1. 500 SALWindows และ Microsoft Windows function
2. ความสามารถในการสร้างและ share function ใหม่ ๆ
3. ทุก ๆ Back-End datatype
4. Variables และ Arrays
5. Control Flow
6. การ access ไปยัง externally developed code เช่น C, C++

SQLWindows 5.0 ในขณะที่ทำงานวิจัยชิ้นนี้อยู่ บริษัท GUPTA Corp. ได้ออก Version 5.0 ของ SQLWindows ซึ่งมีความสามารถเพิ่มขึ้นมาดังนี้



- ตัวคอมไพเลอร์ ที่สามารถที่จะกำเนิด Code เป็น C ได้พร้อมทั้งสามารถ optimize code ให้มีประสิทธิภาพและความเร็วมากขึ้น
- เทคโนโลยีใหม่คือ Quick Object ซึ่งจะมีการฝังที่จำเป็นลงไปในแต่ละ Object ที่เลือกให้ ทำให้พัฒนาโปรแกรมประยุกต์ได้รวดเร็วขึ้นมาก
- SQLWindows ตัวใหม่นี้สามารถวิ่งบนฐานข้อมูลของ Oracle และ Sybase ได้ดีเหมือนกับวิ่งบน SQLBase
- QuickObject ไม่เพียงสามารถเข้าถึงฐานข้อมูลแบบไคล์แอนด์-เซอร์เวอร์ที่สนับสนุนเท่านั้นแต่ยังรวมไปถึง Lotus Notes และยังเข้าถึงระบบ E-mail ได้โดยตรงเช่น Lotus cc:Mail และ Microsoft Mail
- จำนวนของฐานข้อมูลที่สามารถเข้าถึงได้ จะถูกรวมไปถึง ฐานข้อมูลที่สนับสนุน ODBC (Open Database Connectivity) ด้วย.

Extensive Integration และ Interpretability

สนับสนุน Dynamic Data Exchange (DDE), Object Linking and Embedding (OLE), Multiple Document Interface (MDI) และ external Dynamic Link Libraries (DLL) ทั้งใน โปรแกรมประยุกต์ Designer และ SAL นอกจากนั้น windows message call ต่าง ๆ ยังสามารถใช้ได้ใน SQL Windows

6.10.3 Quest

เป็น tool ที่ใช้ในการเข้าถึงข้อมูลสำหรับผู้ใช้การติดต่อกับผู้ใช้ ใช้การ point-and-click บน window ในการใช้ Quest จึงไม่ต้องการความรู้ทาง SQL Quest ประกอบด้วย

6.10.3.1 Query activity

เป็นการเข้าถึงข้อมูลใน SQL database การใช้ Quest ทำให้ผู้ใช้สามารถจะสร้าง database query โดยการ point-and-click เมื่อผู้ใช้ทำการสร้าง query แต่ละ query Quest จะสร้าง SQL code ที่ถูกต้องให้ตัว SQL - code นี้จะถูกเก็บซ่อนไว้ไม่แสดงต่อผู้ใช้ แต่ถ้าผู้ใช้ต้องการดูก็สามารถที่จะเรียกดูได้ Quest จะ provide การ link ระหว่างตารางที่มีความสัมพันธ์กันได้โดยอัตโนมัติ และหลังจากที่ query ถูก define ขึ้นมาจะสามารถเก็บไว้ใช้งานต่อไปได้

Administration ของระบบสามารถที่จะใช้ Quest ในการสร้าง join formula และ query conditions อื่น ๆ ลงใน query template ได้เพื่อให้ผู้ใช้คนอื่น ๆ สามารถใช้ template เหล่านี้สำหรับการสร้าง query ของตัวเองได้

6.10.3.2 Form activity

เป็นการเข้าถึงและการจัดการ SQL data โดยผ่าน Form activity สามารถที่จะสร้าง form-based data management interface ได้โดยอัตโนมัติ การใช้ Quest form ผู้ใช้ไม่ต้องใช้การเขียนโปรแกรมใด ๆ เลยเช่น การสร้าง data entry form

6.10.3.3 Report activity

เป็นการสร้างรายงานที่สามารถเข้าถึงข้อมูล SQL ใช้ในการออกแบบและการสร้างรายงานจากแหล่งข้อมูลใด ๆ ก็ได้ที่ Quest support user สามารถที่จะเลือก fonts graphics และ formatting option อื่น ๆ ได้ ตัวรายงานสามารถที่ถูก pre-view บนหน้าจอหลังจากเป็นที่พอใจแล้วจึงส่ง output ไปที่ printer

6.10.3.4 SQL activity

เป็นการ query หรือการใช้คำสั่งเกี่ยวกับ data management โดยใช้ SQL โดยตรงสำหรับผู้ใช้ที่คุ้นเคยกับภาษา SQL อยู่แล้วก็สามารถใช้ได้โดยตรงเลย ผลของการ query ก็สามารถส่งไปยัง report ได้เช่นเดียวกัน

6.10.3.5 Table activity

ใช้ในการสร้าง การ browse และการ edit table ผู้ใช้สามารถใช้ table activity สร้างตารางใหม่หรือ view ที่มีอยู่แล้วก็ได้ และเพื่อการเพิ่ม security และ data integrity Table activity จะถูก install โดยให้มี write privilege กับ local database แต่เป็น read-only privilege สำหรับฐานข้อมูลอื่น ๆ บน network

6.10.3.6 Catalog activity

ใช้ในการสร้างการ browse และการ modify database Catalog activity จะจัดการเกี่ยวกับ data definitions ใน database user สามารถที่จะ browse ดูได้ว่าแต่ละตาราง view index ถูกระบุอย่างไรทำให้รู้ถึงสิทธิ์ของตนในการเข้าถึงข้อมูลต่าง ๆ และเช่นเดียวกับ Table activity Catalog activity จะถูก install โดยให้มี write privilege สำหรับ local data แต่เป็น read-only privilege สำหรับฐานข้อมูลในระยะไกล

6.10.4 SQLTalk

เป็น data manager สำหรับ SQLBase Server และสำหรับฐานข้อมูลที่ GUPTA SQLNetwork support อยู่ SQLTalk จะรับ query (เป็นภาษา SQL) ที่ Input Window และทำการแสดงผลของ query นั้นออกที่ output window

SQLTalk มีการติดต่อ 2 ลักษณะด้วยกัน คือ

1.) SQLTalk/Window เป็น graphical interface ใช้บน Microsoft windows

2.) SQLTalk/Character เป็น interface แบบ DOS command line

SQLTalk/Windows

ทำงานได้โดยใช้การ point-and-click list ของฐานข้อมูล, ตาราง และผู้ใช้จะดูได้จาก menus และกรอบตอบโต้ใน SQLTalk จะประกอบด้วย window 2 window คือ input window สำหรับการ edit SQL statement

เราสามารถที่จะดูผลของ query ได้ใน format หลาย ๆ แบบเช่น SQL, ASCII, DIFF, DBF, WKS และ raw data รูปแบบเหล่านี้สามารถที่จะถูก import ลงในฐานข้อมูลได้เนื่องจากและเนื่องจาก ReportWindow สามารถที่จะรอบรับ input จาก source data ใดก็ได้ data file ที่สร้างจาก SQLTalk จึงสามารถที่จะถูกบรรจุเข้าไปใน ReportWindow เพื่อสร้าง report ได้

6.10.5 TeamWindows

เป็น Data Dictionary driven repository สำหรับข้อมูลและ reusable โปรแกรมประยุกต์ block มี Repository-based tools เพื่อสนับสนุนการเขียนโปรแกรมที่ทำงานร่วมกันเป็นทีม

TeamWindows ทำให้เกิดการ sharing overuse components และจะทำ administrative task ต่าง ๆ เช่น security, standard enforcement ฯลฯ รวมทั้ง TeamWindows จะสร้าง up-todate report บนสถานะการของทุก ๆ phase ของ โปรแกรมประยุกต์ project

TeamWindows สามารถที่จะ

- สร้าง Data Dictionary-Driven โปรแกรมประยุกต์ โดยตัว TeamWindows จะบำรุงรักษา data dictionary กลางของข้อมูลที่เกี่ยวกับโครงสร้างของฐานข้อมูล เช่น พวก, ตาราง, ชื่อ column, primary และ foreign key ความสัมพันธ์และพารามิเตอร์ อื่น ๆ ซึ่งข้อมูลเกี่ยวข้องกับโครงสร้างเหล่านี้สามารถที่จะถูก import มาจาก CASE tools ที่มีอยู่แล้วได้
- Modify data dictionary ทุก ๆ ครั้งที่มีการเปลี่ยนแปลงของโครงสร้างของฐานข้อมูลรวมทั้งแก้ไขทุก ๆ โปรแกรมประยุกต์ที่เกี่ยวข้องด้วย
- ควบคุม security ของ source code และ version เมื่อ SQLWindows ถูกพัฒนาสมบูรณ์แล้ว มันจะถูกเก็บใน repository และถูกจัดการโดย TeamWindows source code controller มี check-in

และ check-out facility ซึ่งทำให้สมาชิกของกลุ่มสามารถที่จะ share และ modify components ของ โปรแกรมประยุกต์ ได้โดยที่ repository security จะคอยปฏิเสธการเข้าถึงที่ไม่ถูกต้องของผู้ใช้และทำให้แน่ใจได้ว่าในเวลาใดเวลาหนึ่งจะมี programmer 1 คนเท่านั้นที่จะสามารถ modify component ใด ๆ นอกจากนั้น TeamWindows จะคอยตามและ Update version numbers ของโปรแกรมประยุกต์ modules ตลอดวัฏจักรของการพัฒนา

- สร้างและดูแลรักษา Template Libraries, TeamWindows จะจัดหา pre-defined templates สำหรับการสร้าง SQLWindows screen และ โปรแกรมประยุกต์ โดยใช้ data dictionary information ผู้พัฒนาสามารถที่จะ save และ reuse template เหล่านั้นได้

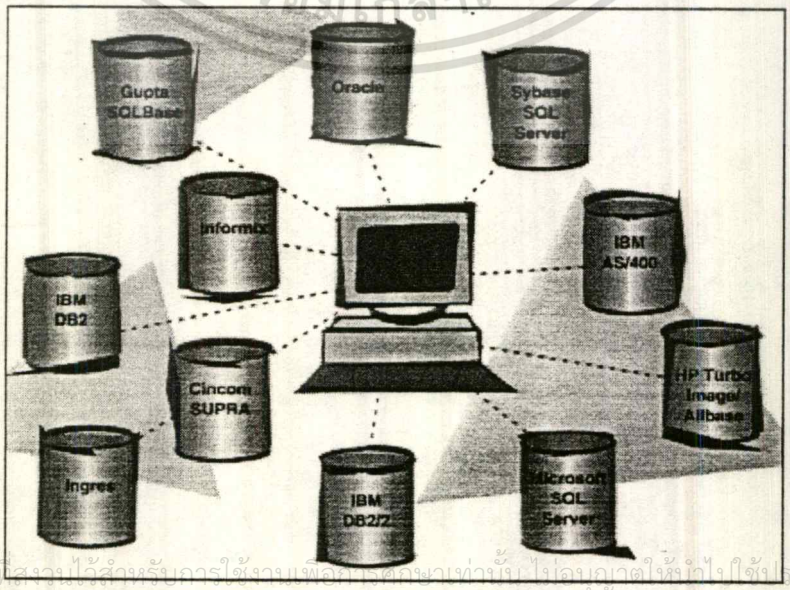
6.10.6 ReportWindows

เป็น graphical report designer & writer ซึ่งจะมี font, สี และ formatting options อื่น ๆ ให้เลือก Report ที่สร้างจาก ReportWindows นี้จะ compatible กับ report ที่สร้างจาก Quest

6.10.7 SQLColsole

SQLConsole ซึ่งมีคุณสมบัติของโปรแกรมเฝ้าติดตาม (monitor) ประสิทธิภาพของระบบ และยังมีเครื่องมือสำหรับช่วยงาน DBA อีกด้วย กล่าวคือช่วยติดตั้ง, ถอนการติดตั้ง, สร้างและลบฐานข้อมูล ช่วยจัดการทำสำรองฐานข้อมูลและแฟ้ม log และจัดการกับ server, ฐานข้อมูล และโพรเซสที่กำลังทำงานอยู่ด้วย ตามตารางหมายกำหนดการล่วงหน้า โดยตัวเฝ้าติดตามจะแสดงผลทางสถิติต่าง ๆ ออกมาเป็นลักษณะกราฟิก 3 มิติ

6.10.8 SQLRouters



เป็นตัวเชื่อมต่อระหว่าง graphical PC (Client) กับ LAN database และ SQL database SQLRouters นี้จะสามารถเข้าถึงได้อย่างรวดเร็วไป Oracle, SQL Server, Informix, Ingres และ AS/400 รวมทั้งสนับสนุนมาตรฐาน Open Database Connectivity (ODBC) เพื่อการเข้าถึง PC และ SQL DBMSs เช่น dBASE, Paradox, FoxPro, Access, SQL Server, Oracle, Rdb, AS/400 และ DB2

บทที่ 7

Microsoft Open Database Connectivity (ODBC)

ทฤษฎีการทำงานของ ODBC

ส่วนติดต่อของการเชื่อมต่อฐานข้อมูลเปิด(Open Database Connectivity, ODBC) เป็นตัวทำให้โปรแกรมประยุกต์ สามารถที่จะเข้าถึงข้อมูลในระบบจัดการฐานข้อมูล (Database Management System, DBMS) โดยใช้ภาษาค้นคืนแบบโครงสร้าง (Structure Query Language ,SQL) เป็นมาตราฐานหลักในการเข้าถึงข้อมูล

7.1 ตำนานของการเชื่อมต่อฐานข้อมูลเปิด

ในโลกของฐานข้อมูลทางการค้า โปรแกรมประยุกต์โดยทั่วไปแล้ว โปรแกรมนั้นจะระบุประสิทธิภาพของงานฐานข้อมูล DBMS ดังเช่น โปรแกรมบัญชีเงินเดือน,การวิเคราะห์ระบบการเงินหรือการบริหารสินค้าคงคลัง ดังนั้นโปรแกรมเหล่านี้จะถูกเขียนโดยใช้ SQL แบบฝังตัว (embedded SQL) เนื่องจาก SQL แบบฝังตัวนี้สามารถถ่ายไปใช้เครื่องที่มี Hardware ต่างกันและสภาพแวดล้อมของระบบปฏิบัติการที่ต่างกัน ฉะนั้น source code จะต้องทำการคอมไฟล์สำหรับสภาพแวดล้อมใหม่ ๆ ได้

SQL แบบฝังตัวไม่ได้ถูกออกแบบมาสำหรับโปรแกรมประยุกต์ที่ต้องการวิเคราะห์ข้อมูลที่ถูกเก็บในฐานข้อมูล ดังเช่น DB/2 และ Oracle และเสนอที่ทำจากตระกูลโปรแกรมประยุกต์ที่ติดต่อผู้ใช้ดังเช่น Microsoft Excel ภายใต้จุดมุ่งหมายทางการค้านั้น การประมวลผลฐานข้อมูล ในรุ่นหนึ่งของ Microsoft Excel มีการถูกคอมไฟล์ขั้นต้น ด้วย IBM precompiler และตัวอื่น ๆ ด้วย Oracle precompiler

ODBC ให้แนวทางใหม่ที่จะแยกโปรแกรมที่จะทำการประมวลผลข้อมูลสารสนเทศของฐานข้อมูลและมีทางที่ให้สำหรับโปรแกรมประยุกต์ที่จะนำเข้าข้อมูลจากแหล่งภายนอกได้ ดังนั้นมีทางเป็นไปได้มากที่จะมีการติดต่อสื่อสารสำหรับตัวแปรของวิธีการอื่น ข้อตกลงของข้อมูล (data protocol) และความสามารถของ DBMS ODBC มีวิธีการที่จะยอมให้ใช้เทคโนโลยีที่ต่างกันที่จะใช้งานด้วยการติดต่อที่เป็นมาตราฐาน วิธีการนี้นำไปสู่แนวความคิดของตัวขับฐานข้อมูลไดนามิกลิงค์ไลบรารี ที่ทำให้โปรแกรมประยุกต์สามารถที่จะกำหนดความต้องการที่จะประมวลผลแหล่งกำเนิดข้อมูลผ่านวิธีการติดต่อสื่อสารคล้ายกับตัวขับเครื่องพิมพ์ที่ทำงานภายใต้ Windows ODBC ให้มาตราฐานในการติดต่อสื่อสารกับผู้ใช้ที่ยอมให้โปรแกรมประยุกต์ทั้งสองเป็นผู้เขียนและให้ข้อมูลไลบรารีไปยังข้อมูลระหว่างโปรแกรมประยุกต์และแหล่งกำเนิดข้อมูล

7.2 การติดต่อของ ODBC

การติดต่อของ ODBC ถูกนิยามดังต่อไปนี้

- ไลบารีของ ODBC มีหน้าที่เรียกและยอมให้โปรแกรมประยุกต์ติดต่อสื่อสารไปยัง DBMS คำสั่ง SQL และรอรับผลลัพธ์
- รูปแบบของ SQL พื้นฐานวางอยู่บน X/Open และ SQL Access Group (SAG) คุณสมบัติ SQL CAE (1992)
- มีมาตราฐานข้อผิดพลาด
- มีมาตราฐานเส้นทางในการติดต่อและเข้าไปยัง DBMS
- มีมาตราฐานในการนำเสนอตามชนิดข้อมูล

ความยืดหยุ่นของการติดต่อ

- ตัวอักษรของคำสั่ง SQL สามารถที่จะรวมไปกับ source code หรือจะทำงานที่ Run Time
- เป็นออบเจกต์โค๊ดที่เหมือนกันถึงจะถูกใช้ในงานผลิตภัณฑ์ DBMS ที่ต่างกันได้
- โปรแกรมประยุกต์สามารถที่จะยกเลิกข้อตกลงในการติดต่อสื่อสารระหว่างตัวมันและผลิตภัณฑ์ของ DBMS
- ค่าของข้อมูลสามารถที่จะส่งและรับในรูปของโปรแกรมประยุกต์

การติดต่อ ODBC มี 2 ชนิดที่จะเรียกใช้

1) หน้าที่หลักอยู่บนพื้นฐานของ X/Open และ SQL Access Group ถูกเรียกใช้ในระดับของคุณสมบัติการติดต่อ
2) หน้าที่ส่วนขยายสามารถสนับสนุนหน้าที่อื่น ๆ เช่นการรวมของ Scrollable เคอร์เซอร์ และการประมวลผลแบบอะซิสโครนัส

การส่งคำสั่งของ SQL สามารถที่รวมคำสั่ง SQL เป็นอากรูเมนท์ในฟังก์ชันคอลล์ของ ODBC คำสั่งต้องการที่จะไม่ระบุเฉพาะเจาะจงสำหรับ DBMS โดยเฉพาะ

7.3 สถาปัตยกรรม ODBC

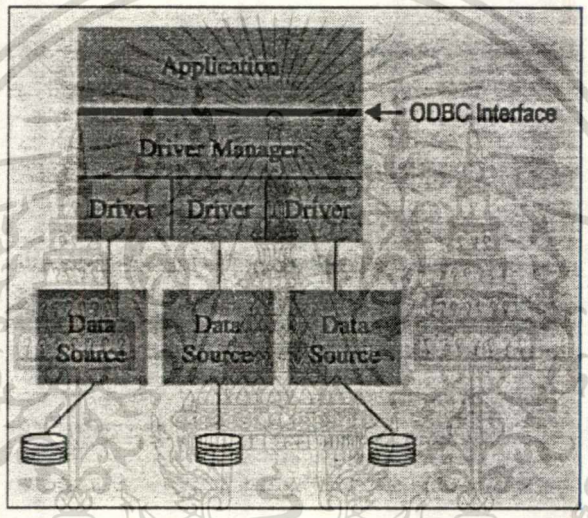
ประกอบไปด้วยส่วนประกอบ 4 ส่วนคือ

1) โปรแกรมประยุกต์ (โปรแกรมประยุกต์) เป็นตัวจัดทำการประมวลผลและเรียกฟังก์ชันของ ODBC เพื่อจะไปสั่งให้คำสั่ง SQL ทำงานและรอรับผลลัพธ์
2) ตัวจัดการตัวขับ (Driver Manager) โหลดตัวขับที่เป็นประโยชน์ของโปรแกรมประยุกต์

3) **ตัวขับ(Driver)** การประมวลผลของฟังก์ชันคอลล์ ODBC จะทำคำสั่ง SQL ที่ถูกเรียกร้องของแหล่งกำเนิดข้อมูลและรับผลลัพธ์กลับไปยังโปรแกรมประยุกต์แต่อย่างไรก็ตามตัวขับที่ได้ถูกแก้ไขให้เข้ากับโปรแกรมประยุกต์ที่จะร้องขอให้ตรงกับการที่จะสนับสนุน DBMS

4) **แหล่งกำเนิดข้อมูล (Data source)** ส่วนประกอบของฐานข้อมูลที่ผู้ใช้ต้องการประมวลผลมันจะถูกช่วยเหลือโดยระบบปฏิบัติการ, DBMS และรูปแบบของระบบเครือข่ายที่ถูกใช้เข้าถึงข้อมูลของ DBMS

ตัวจัดการตัวขับและตัวขับรอจนโปรแกรมประยุกต์เรียกฟังก์ชันคอลล์ ODBC รูปภาพด้านล่างเป็นการแสดงความสัมพันธ์ระหว่างส่วนประกอบทั้ง 4 ส่วน



7.3.1 โปรแกรมประยุกต์ (โปรแกรมประยุกต์)

โปรแกรมประยุกต์ที่ใช้การติดต่อของ ODBC จะทำหน้าที่ดังต่อไปนี้

- เรียกร้องการต่อเชื่อมหรือหน้าต่างทำงานไปยังแหล่งกำเนิดข้อมูล
- ส่งคำสั่ง SQL ที่ต้องการไปยังแหล่งกำเนิดข้อมูล
- กำหนดพื้นที่เก็บข้อมูลและรูปแบบของข้อมูลสำหรับผลของคำสั่ง SQL
- เรียกร้องผลลัพธ์
- ประมวลผลการผิดพลาด
- รายงานผลลัพธ์กลับไปยังผู้ใช้ถ้าต้องการ
- เรียกร้องการยืนยัน(commit)หรือการเรียกคืน(rollback)สำหรับควบคุมทรานเซ็กชั่น(transaction)
- ตัดการติดต่อไปยังแหล่งกำเนิดข้อมูล

โปรแกรมประยุกต์สามารถที่จะใช้การติดต่อสื่อสารภายนอกโดยการรวม Mail, ความสามารถของ Spreadsheet , การประมวลผลออนไลน์ทรานเซ็กชั่นและการสร้างรายงานโปรแกรมประยุกต์จะยอมหรือไม่ยอมโดยผู้ใช้

7.3.2 ตัวจัดการตัวขับ (Driver Manager)

ตัวจัดการตัวขับถูกกำหนดโดย Microsoft คือไดนามิกลิงค์ไลบรารี(DLL) ด้วยไลบรารีที่นำเข้ามา จุดประสงค์ของตัวจัดการตัวขับคือทำการโดยตัวขับ ตัวจัดการตัวขับมีประโยชน์ดังต่อไปนี้

- ใช้แฟ้ม ODBC.INI หรือลงทะเบียนที่จะกำหนดชื่อของแหล่งกำหนดข้อมูลที่ระบุตัวขับที่เป็นไดนามิกลิงค์ไลบรารี (DLL)
- ประมวลผลเริ่มต้นของ ODBC
- เป็นจุดป้อนหน้าที่ของฟังก์ชัน ODBC สำหรับแต่ละตัวขับ
- ตรวจสอบค่าพารามิเตอร์และลำดับสำหรับ ODBC

7.3.3 ตัวขับ(Driver)

ตัวขับก็คือแฟ้มข้อมูลที่เป็น DLL ที่ถูกทำงานภายใต้ฟังก์ชันคอลล์ของ ODBC และติดต่อกับแหล่งกำเนิดข้อมูล ตัวจัดการตัวขับจะทำการโหลดตัวขับเมื่อโปรแกรมประยุกต์เรียกฟังก์ชัน SQLBrowseConnect, SQLConnect, SQLDriverConnect

ตัวขับจะทำงานต่อไปนี้ในการตอบสนองฟังก์ชันคอลล์ของ ODBC จากโปรแกรมประยุกต์

- ทำการเชื่อมต่อไปยังแหล่งกำเนิด
- รับข้อเรียกร้องไปยังแหล่งกำเนิด
- ถ่ายเทข้อมูลจากรูปแบบอื่น ๆ ถ้าโปรแกรมประยุกต์ต้องการ
- รับผลลัพธ์กลับไปยังโปรแกรมประยุกต์
- จัดรูปแบบข้อผิดพลาดไปยังโค๊ดข้อผิดพลาดมาตราฐานและนำกลับไปยังโปรแกรมประยุกต์นั้น
- ประกาศและใช้ตัวเคอร์เซอร์ถ้าต้องการ
- ทำการกำหนดค่าเริ่มต้นทรานเซ็กชั่นถ้าแหล่งกำเนิดข้อมูลนั้นต้องการ

7.3.4 แหล่งกำเนิดข้อมูล(Data Source)

โปรแกรมประยุกต์สามารถจะสร้างการเชื่อมโยงกับผลิตภัณฑ์ DBMS บนระบบปฏิบัติการที่ต้องการ เช่น

- Oracle DBMS วิ่งบนระบบปฏิบัติการ OS/2 เข้าถึงข้อมูลโดย Novell NetWare

- แฟ้ม Xbase ในเครือข่ายและบนระบบปฏิบัติการระยะไกล โดยไม่ผ่านส่วนของการติดต่อสื่อสาร
- Tandem nonstop SQL DBMS วิ่งบนระบบปฏิบัติการ Guardian 90 โดยผ่าน Gateway
- ชนิดของตัวขับ

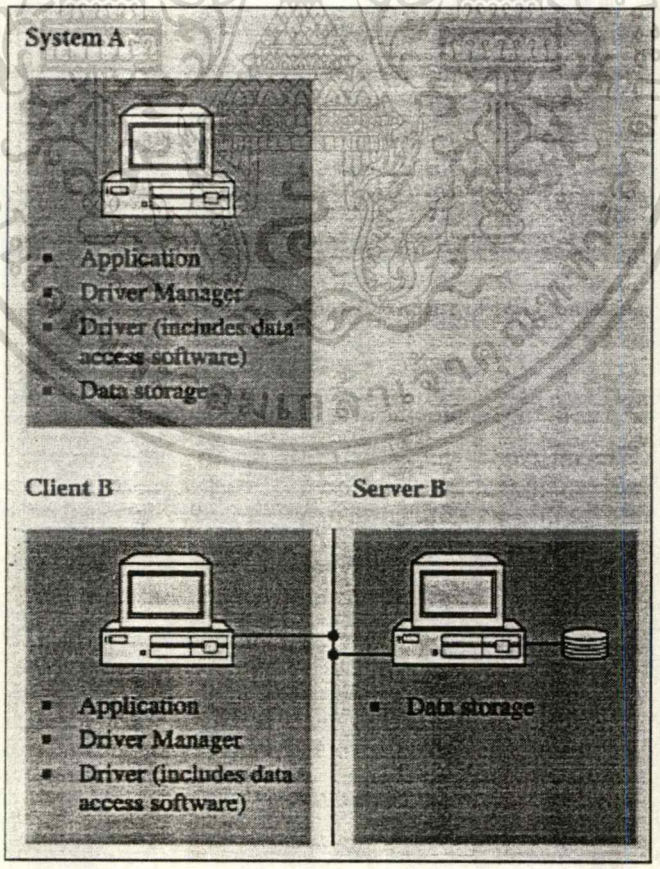
ODBC แบ่งตัวขับออกเป็น 2 ชนิดคือ

1) Single-tier เป็นตัวขับที่จะประมวลผลทั้งการเรียกของ ODBC และประโยคคำสั่ง
2) Multiple-tier เป็นตัวขับที่ประมวลผลการเรียกของ ODBC และจะส่งผ่านประโยคคำสั่ง SQL ไปยังแหล่งกำเนิดข้อมูลต่อไป

7.4.1 ส่วนประกอบของ Single-Tier

ในการพัฒนาแบบ single-tier แฟ้มฐานข้อมูลจะถูกประมวลผลโดยตัวขับโดยตรง ตัวขับจะประมวลผลประโยคคำสั่ง SQL และรับข้อมูลที่จำเป็นมาจากฐานข้อมูล

ตัวขับของ Single-tier จะจำกัดจำนวนชุดของประโยคคำสั่ง SQL โดยจำนวนชุดคำสั่ง SQL ต่ำสุด จะต้องสนับสนุนโดยตัวขับของ single-tier

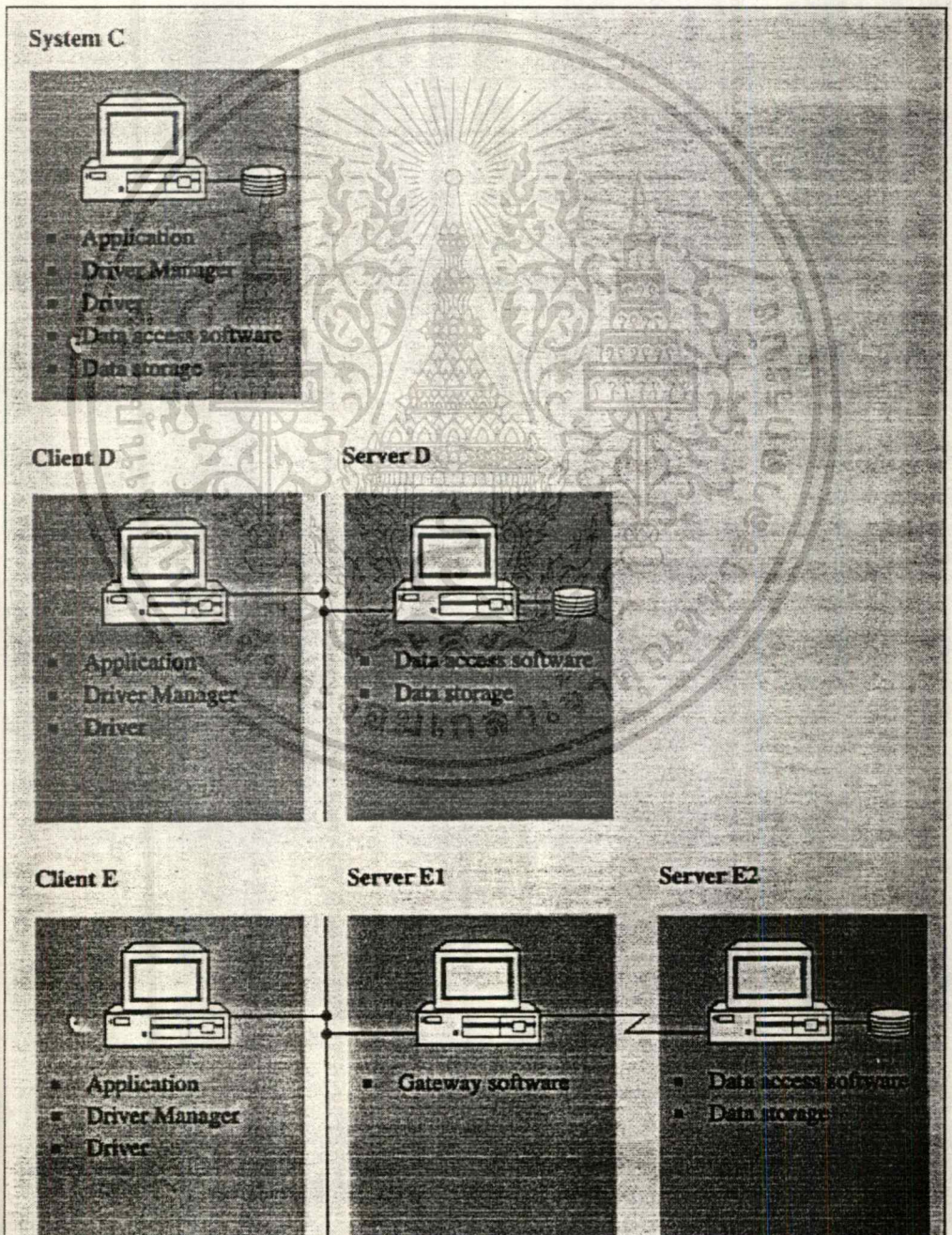


7.4.2 ส่วนประกอบของ Multiple-Tier

ในส่วนประกอบของ multiple-tier ตัวขับจะส่งความต้องการใช้ SQL ไปยัง Server ที่ประมวลผล SQL ที่ต้องการ

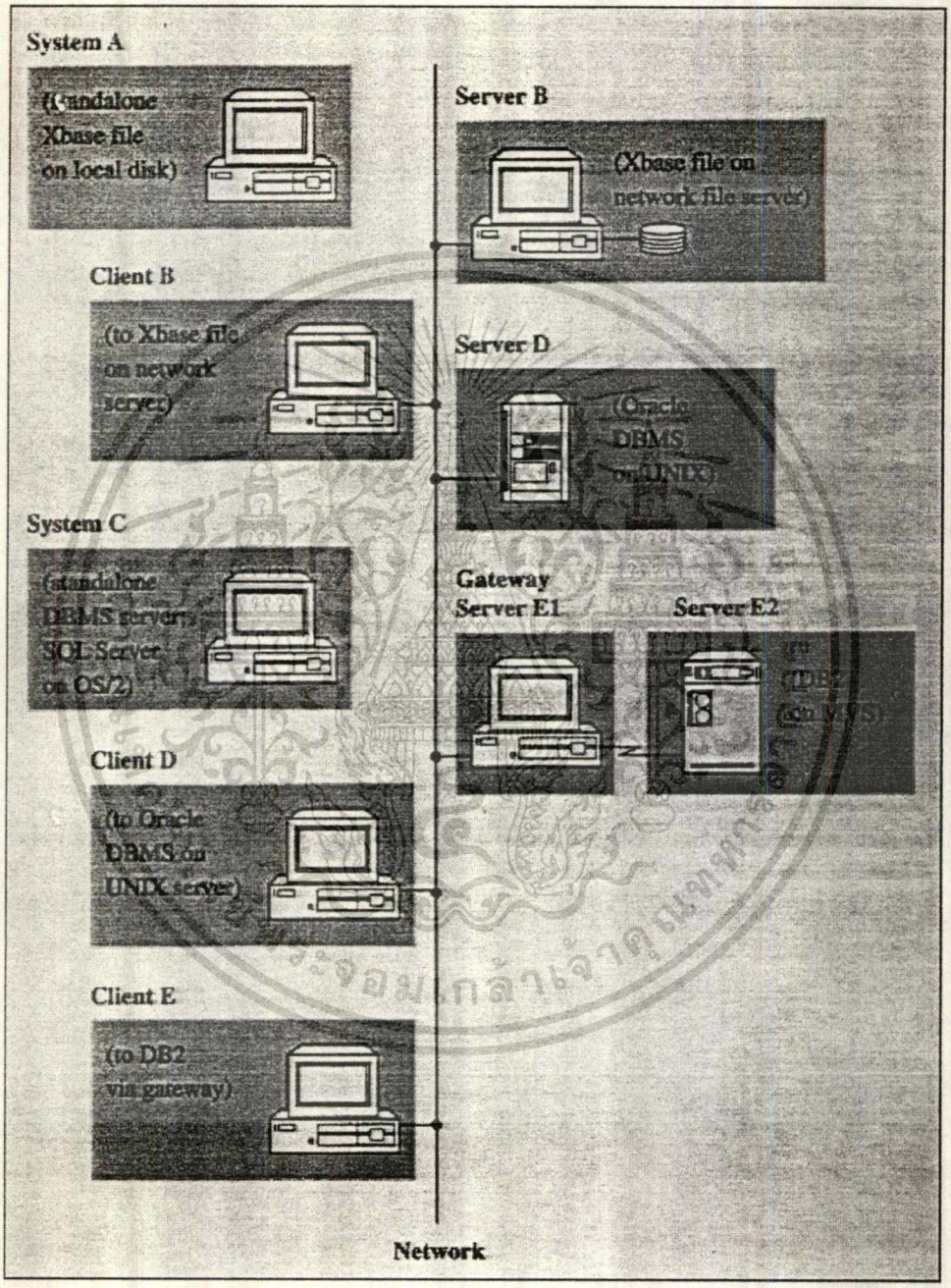
ถึงอย่างไรการติดตั้งบนระบบเดี่ยวจะถูกแบ่งแยกไปตาม platform โปรแกรมประยุกต์ ตัวขับและตัวจัดการตัวขับจะอยู่เพียงระบบใดระบบหนึ่งเท่านั้น เรียกว่า client ฐานข้อมูลและ software ที่ใช้ควบคุมการเข้าถึงฐานข้อมูลที่อยู่ในอีกแบบหนึ่ง เรียกว่า server

ส่วนประกอบของ multiple-tier อีกแบบหนึ่งคือสถาปัตยกรรมแบบ gateway ตัวขับจะส่งความต้องการใช้ SQL ไปยังตัวประมวลผล gateway ที่ซึ่งจะส่งความต้องการไปยังแหล่งกำเนิดข้อมูลอีกที

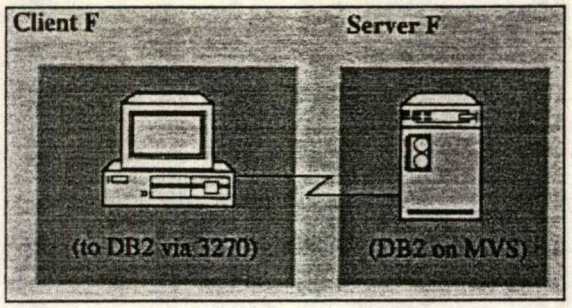


7.4.3 ตัวอย่างเครือข่าย

จากรูปแสดงถึงส่วนประกอบของระบบของระบบเครือข่ายเดี่ยว โดยในรูปจะรวมเอาตัวอย่างของชนิดของ DBMS ที่อยู่ในเครือข่ายเอาไว้ด้วย



โปรแกรมประยุกต์ สามารถจะทำการติดต่อสื่อสาร ข้ามเครือข่ายพื้นที่กว้างได้ (Wide Area Network)



7.5 ความสัมพันธ์ระหว่างโปรแกรมประยุกต์กับตัวขับ

ส่วนติดต่อของ ODBC จะช่วยให้โปรแกรมเมอร์สามารถสร้างโปรแกรมประยุกต์ ODBC โดยไม่ต้องกำหนดแหล่งข้อมูลเป้าหมายเลย ผู้ใช้สามารถเพิ่มเติมตัวขับเข้าไปในโปรแกรมประยุกต์หลังจากโปรแกรมประยุกต์นั้นถูกคอมไพล์แล้วได้ด้วย

ในจุดมาตราฐานของโปรแกรมประยุกต์คือทุก ๆ ตัวขับและแหล่งกำเนิดข้อมูล จะสนับสนุนฟังก์ชัน ODBC เดียวกัน และเรียกประโยคคำสั่ง SQL เดียวกัน อย่างไรก็ตามแหล่งกำเนิดข้อมูลข้อมูลและตัวขับ จะมีฟังก์ชันใช้งานมากมายจึงเป็นเหตุให้การติดต่อกับ ODBC ถูกกำหนดเป็นระดับ class ของ ODBC procedures และประโยคคำสั่ง SQL ที่สนับสนุนโดยตัวขับ

7.5.1 ระดับของ ODBC ที่ตรงกัน

ODBC กำหนดระดับที่ตรงกันสำหรับตัวขับใน 2 ลักษณะคือ ODBC API และไวยากรณ์ ODBC SQL (โดยรวมเอาชนิดของข้อมูลของ ODBC SQL เอาไว้ด้วย)ระดับความตรงกัน จะช่วยให้ทั้งโปรแกรมประยุกต์และผู้พัฒนาตัวขับโดยการตั้งมาตราฐานของชุดของฟังก์ชันขึ้นมา โปรแกรมประยุกต์สามารถกำหนดฟังก์ชันที่ต้องการได้ง่ายถ้าตัวขับให้การสนับสนุนฟังก์ชันนั้นด้วย

ตัวขับ สามารถที่จะพัฒนาให้สนับสนุนโปรแกรมประยุกต์โปรแกรมประยุกต์ โดยไม่ต้องกังวลกับความต้องการของแต่ละโปรแกรมประยุกต์ว่าไม่ตรงกัน

เพื่อเรียกร้องให้มีการยืนยันระหว่าง API หรือระดับที่ตรงกันของ SQL, ตัวขับต้องสนับสนุนทุก ๆ ฟังก์ชัน ในระดับที่ตรงกันโดย ต้องให้ ฟังก์ชัน นั้นตรงกับ ฟังก์ชัน ที่ DBMS ให้การสนับสนุนด้วย

โปรแกรมประยุกต์ สามารถกำหนดว่า ฟังก์ชัน นี้สนับสนุน โดยตัวขับด้วยการเรียก SQLGetInfo, SQLGetFunction และ SQLGetTypeInfo

7.5.1.1 ระดับความตรงกันของ API

ODBC API กำหนดชุดของ ฟังก์ชัน ที่มีลักษณะเดียวกัน ไปยัง ฟังก์ชัน ใน X/Open และรูปแบบของ SQL Access Group Call Level Interface. ODBC จะกำหนดชุดของ ฟังก์ชัน 2 ชุดม ระดับ 1 และระดับ 2 ดังนี้

Core API

- กำหนดและปล่อยสภาวะแวดล้อม, การติดต่อ และ Statement handles
- เชื่อมต่อกับแหล่งข้อมูล
- Prepare และ execute ประโยคคำสั่ง SQL ประมวลผล SQL แบบทันทีทันใด
- กำหนดพื้นที่ให้กับตัวแปรใน SQL และคอลัมน์ของผลลัพธ์
- รับข้อมูลจากชุดของผลลัพธ์, รับข้อมูลเกี่ยวกับชุดของผลลัพธ์
- ทำรายการยืนยันและการเรียกกลับ
- รับข้อมูลข้อผิดพลาด

Level 1 API

- ฟังก์ชันในระดับ Core API
- เชื่อมต่อกับแหล่งข้อมูลพร้อมกับตัวขับ
- ตั้งและข้ามถ้าประโยคคำสั่งและการเลือกของการเชื่อมต่อ
- ส่งค่าของตัวแปรบางส่วนหรือทั้งหมด
- รับค่าของผลลัพธ์เป็นคอลัมน์ บางส่วนหรือทั้งหมด
- รับข้อมูลเกี่ยวกับ catalog
- รับข้อมูลเกี่ยวกับตัวขับและแหล่งข้อมูล

Level 2 API

- ฟังก์ชันในระดับ core และ level 1 API
- browse connection information และแสดงแหล่งข้อมูลที่ใช้งานได้
- ส่งค่าของตัวแปรแบบ Array รับค่าของผลลัพธ์เป็นตัวแปร
- รับจำนวนของตัวแปรและรายละเอียด
- ใช้ Scrollable cursor
- รับค่าของประโยคคำสั่ง SQL ที่มีมากกว่าคืน
- รับข้อมูล catalog
- เรียก DLL

7.5.1.2 ระดับความตรงกันของ SQL

ODBC กำหนดไวยากรณ์หลักไว้คร่าว ๆ ในลักษณะเดียวกันกับ X/Open และ SQL Access Group SQL CAE . ODBC จะกำหนดไวยากรณ์น้อยสุด เพื่อพอดีกับระบบความตรงกันเบื้องต้นของ ODBC และไวยากรณ์ซับซ้อนไว้สำหรับ DBMS ที่ใช้ส่วนขยายของ SQL โดยแสดงให้ดูดังนี้

Minimum SQL Grammar

- Data Definition Language (DDL) เช่น CREATE TABLE และ DROP TABLE
- Data Manipulation Language (DML) เช่น SELECT, INSERT, UPDATE, SEARCHED และ DELETE SEARCHED
- สมการ เช่น A > B + C
- ชนิดของข้อมูล เช่น CHAR , VARCHAR หรือ LONG VARCHAR

Core SQL Grammar

- Minimum SQL grammar และ data type
- DDL เช่น ALTER TABLE , CREATE INDEX, DROP INDEX, CREATE VIEW, DROP VIEW, GRANT และ REVORK
- DML เช่น full SELECT
- สมการ subquery, set functions เช่น SUM และ MIN.
- Data type เช่น DECIMAL, NUMERIC, SMALLINT, INTEGER, REAL, FLOAT, DOUBLE PRECISION

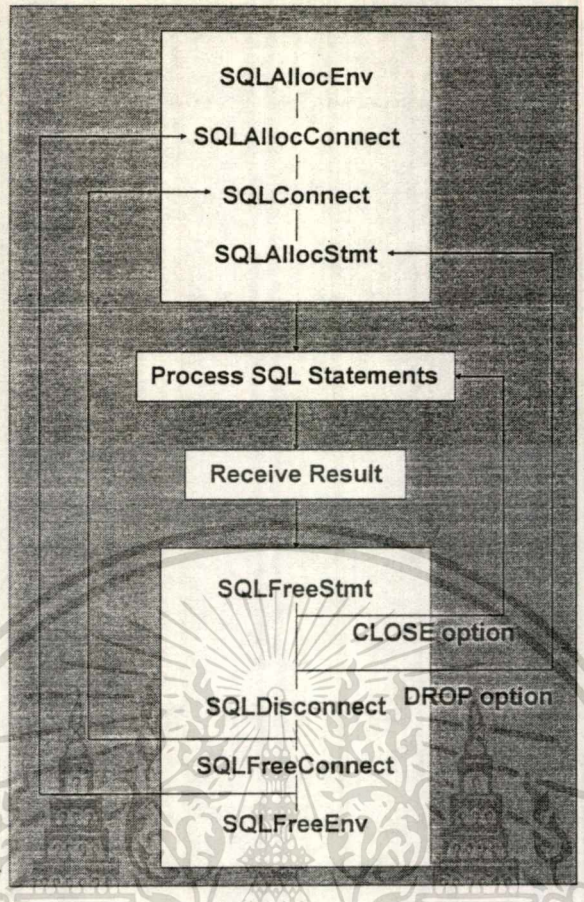
Extended SQL grammar

- minimum และ core SQL และชนิดของข้อมูล
- DML เช่น Outer Joins, positioned UPDATE, positioned DELETE, SELECT FOR UPDATE และ Unions
- สมการ scalar function เช่น SUBSTRING และ ABS, date, time และ timestamp literals
- Data type เช่น BIT, TINYINT, BIGINT, BINARY, VARBINARY, LONG VARBINARY, DATE, TIME, TIMESTAMP
- Batch SQL
- การเรียก procedure

7.6 ขั้นตอนเบื้องต้นของโปรแกรมประยุกต์

เพื่อติดต่อกับแหล่งข้อมูลโปรแกรมประยุกต์ จะทำตามขั้นตอนดังนี้

1. เชื่อมต่อไปยังแหล่งข้อมูล โดยการระบุชื่อของแหล่งข้อมูลและข้อมูลอื่น ๆ ที่จำเป็นในการที่จะเชื่อมต่อได้อย่างสมบูรณ์
2. ประมวลผลหนึ่งหรือมากกว่าประโยคคำสั่ง SQL
   - โปรแกรมประยุกต์จะใส่ข้อความ SQL ลงในบัฟเฟอร์รวมทั้งตัวแปรที่จำเป็นด้วย
   - ถ้าประโยคคำสั่งส่งผลลัพธ์กลับมา, โปรแกรมประยุกต์จะกำหนด cursor name สำหรับให้ประโยคคำสั่งหรือยอมให้ตัวขับกระทำการใด ๆ ด้วย
   - โปรแกรมประยุกต์จะมอบประโยคคำสั่งสำหรับการ prepare หรือ execution ทันที
   - ถ้าประโยคคำสั่งทำการสร้างชุดผลลัพธ์ โปรแกรมประยุกต์สามารถถามถึง attributes ของชุดผลลัพธ์นั้น
   - ถ้าประโยคคำสั่งเกิดมีข้อผิดพลาด, โปรแกรมประยุกต์จะรับข้อมูลผิดพลาดจากตัวขับและทำตามความเหมาะสม
3. การจบของประโยคคำสั่งทำได้โดยการยืนยันหรือเรียกคืนกลับ
4. จะสิ้นสุดการเชื่อมต่อก็ต่อเมื่อเสร็จสิ้นการเชื่อมโยงกับแหล่งข้อมูล



จากรูปเป็นการแสดงรายการของฟังก์ชัน ODBC ที่ถูกเรียกโดยโปรแกรมประยุกต์เพื่อเชื่อมโยงกับแหล่งข้อมูล, ประมวลผล SQL และยกเลิกการเชื่อมต่อกับแหล่งข้อมูล

บทที่ 8
การสร้าง Front-End สำหรับ Client-Server Database ด้วย Visual Basic 3.0

การสร้างโปรแกรมประยุกต์ฐานข้อมูลของฝั่ง front-end ไปยัง RDBMS back-end โดยที่ตัว RDBMS จะเน้นที่ผลิตภัณฑ์หลัก ๆ เช่น Microsoft SQL Server for Windows NT 4.2 (NTSQLS) ที่วิ่งอยู่บนระบบปฏิบัติการ Windows NT Advanced Server 3.1 (NTAS) และยังแสดงถึงการใช้ SQL Administrator และ SQL Object Manager ตัวใหม่บน NTSQLS 4.2

นอกจากนี้ยังแสดงถึงการใช้ Microsoft ODBC API และการสร้างโปรแกรมประยุกต์บน client-server RDBMS ด้วย ODBC driver ที่สนับสนุนบนไวยากรณ์ SQL ระดับ core และ ODBC ระดับ 1

การออกแบบ Visual Basic 3.0 front-end สำหรับฐานข้อมูลแบบ client-server ก็ทำในลักษณะเดียวกันกับการออกแบบโปรแกรมประยุกต์บนฐานข้อมูลแบบตั้งโต๊ะทั่ว ๆ ไป ถ้ามีโครงสร้างของตารางในฐานข้อมูลแบบตั้งโต๊ะเดิมอยู่แล้ว เราสามารถเปลี่ยนคำสั่ง OpenTable() เพื่อที่จะเปิดตารางของข้อมูลใน ODBC แทนการเปิดบนฐานข้อมูลแบบตั้งโต๊ะได้ ในลักษณะทั่ว ๆ ไป อย่างไรก็ตามการใช้ CreateDynanset() หรือ CreateSnapshot() ด้วยการผ่านค่าที่เลือกไปยังประโยคคำสั่ง SQL ก็สามารถทำได้และคำสั่ง ExecuteSQL() สามารถใช้แทนคำสั่ง Execute() เพื่อเพิ่มประสิทธิภาพบน client-server front-end.

8.1 การใช้ Microsoft SQL Server สำหรับ Windows NT version 4.2

Microsoft SQL Server for Windows NT เป็นการปรับปรุงมาจาก Microsoft SQL Server version 4.2 ที่วิ่งอยู่บน OS/2 เวอร์ชั่น 1.3 อย่างไรก็ตาม NTSQLS จะมีลักษณะคล้ายกับ Sybase Server version 4.2 โดยเพิ่มความสามารถด้านการทำงานแบบ Symmetrical Multiprocessing (SMP) ซึ่งเป็นการแบ่งงานให้ CPU ทำตั้งแต่ 2-4 ชิพ

เราสามารถที่จะวิ่ง NTSQLS ภายใต้ Windows NT 3.1 หรือ Windows NT Advanced Server 3.1 ตัว NTSQLS client สามารถที่จะใช้ได้ทั้งบน DOS, OS/2, DOS/WfwG และ สภาวะแวดล้อมของการทำงานแบบ NT

8.2 การสร้างแหล่งข้อมูลของ ODBC จากฐานข้อมูลแบบ client-server

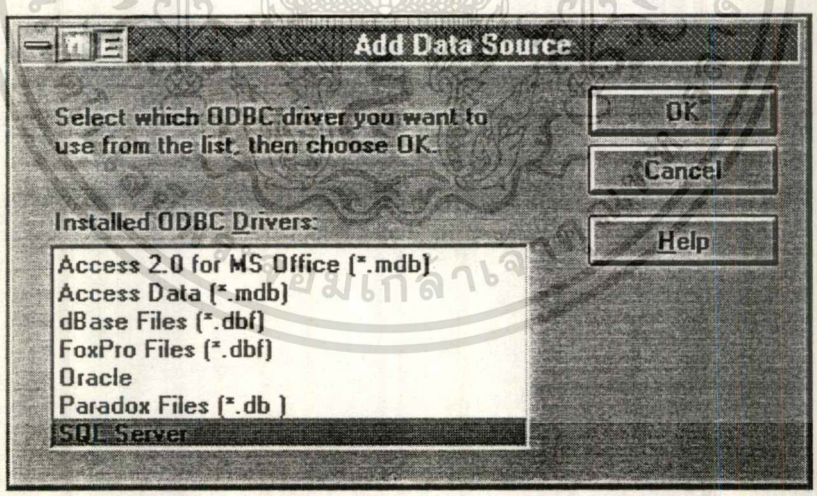
การสร้างแหล่งข้อมูลของ ODBC สามารถเลือกสร้าง ODBC ได้จาก ODBC database driver ที่รวมอยู่ใน Excel 5.0 หรือใน Microsoft ODBC Desktop Database Drivers Kit

ในส่วนต่อจากนี้จะอธิบายถึงวิธีการในการเพิ่มฐานข้อมูลชื่อ VDODB ซึ่งเป็นแหล่งข้อมูล SQL เข้าไปยังระบบ

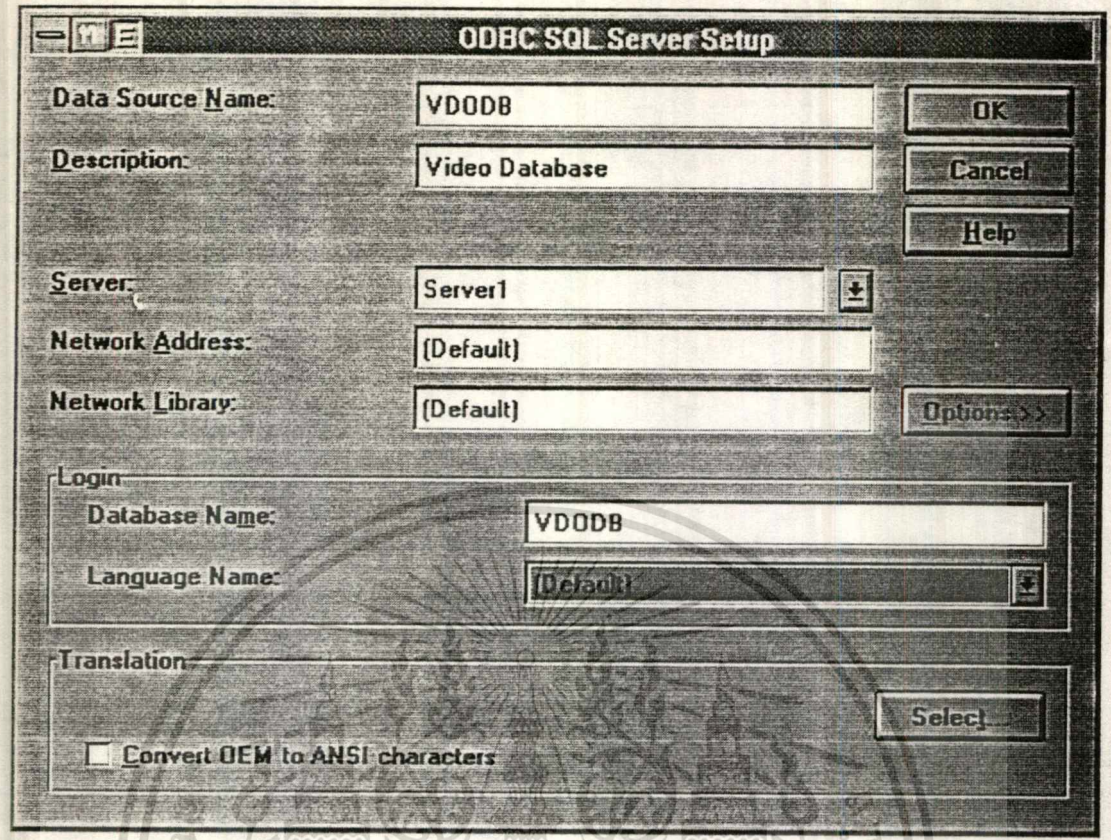
8.2.1 การเพิ่มฐานข้อมูล VDODB ไปยังแหล่งข้อมูล ODBC

การใช้ SQL Server ODBC database driver เพื่อเพิ่มฐานข้อมูล VDODB ลงไปยังแหล่งข้อมูล ODBC ทำตามขั้นตอนดังนี้

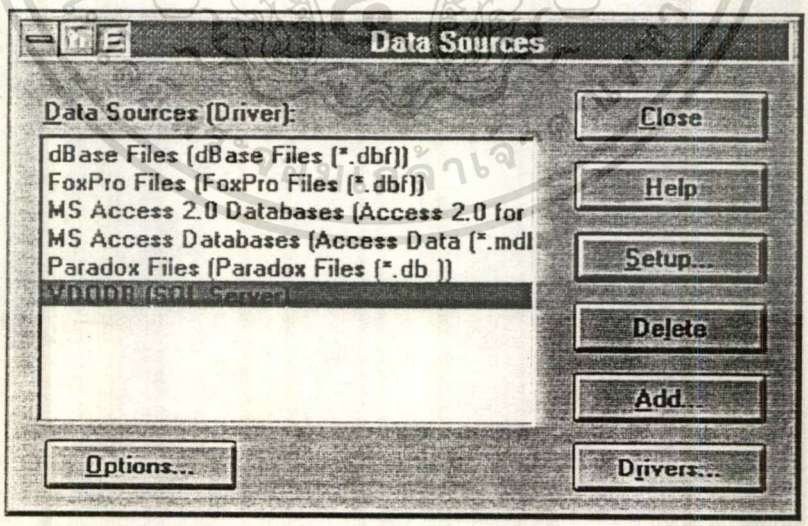
1) ใช้โปรแกรมประยุกต์ ODBC Administrator ถ้าเราติดตั้ง ODBC Desktop Database Driver Kit หรือ Microsoft Excel 5.0 ลงไปแล้ว ไอคอนของ ODBC Administrator ก็จะปรากฏอยู่ใน Control Panel
2) กดที่ปุ่ม Add ของกรอบตอบโต้ Data Source เพื่อแสดงกรอบตอบโต้ Add Data Source
3) เลือก SQL Server จากรายชื่อ ODBC Drivers ที่ติดตั้งลงไปแล้วและกดที่ปุ่ม OK เพื่อแสดงกรอบตอบโต้ ODBC SQL Server Setup



4) กดที่ปุ่ม Options เพื่อขยายทางเลือกของกรอบตอบโต้ ODBC SQL Server Setup และพิมพ์ VDODB ในกรอบข้อความเพิ่มรายละเอียดลงไปในกรอบข้อความ Description ด้วย
5) เปิดกรอบ Server และเลือก Server จากฐานข้อมูล VDODB และใส่รายละเอียดของ Network library
6) พิมพ์ VDODB ลงในกรอบข้อความ Database Name และเลือก Language Name



7) กดที่ปุ่ม OK เพื่อเปิดกรอบตอบโต้ ODBC SQL Server Setup เราก็จะได้แหล่งข้อมูลเพิ่มเติมเข้ามาในกรอบตอบโต้ Data Source



8) กดที่ปุ่ม Close ของกรอบตอบโต้ Data Sources เพื่อกลับสู่ Control Panel หรือ Program Manager

8.3 การใช้ class ความเข้าใจกันได้ (Compatibility Layer) ของ Microsoft Access 2.0 สำหรับ Visual Basic 3.0

class ความเข้ากันได้ใน Access 2.0 (หรือเรียกอีกอย่างหนึ่งว่า Mapping Layer) สำหรับ Visual Basic 3.0 แฟ้มที่ต้องการจะอยู่ในชุดพัฒนา Microsoft Access (ADT) โดยแฟ้มที่ต้องการจะอยู่ใน Microsoft Office Developer's Kit (ODK) เพื่อวัตถุประสงค์ในการอธิบายเกี่ยวกับความสามารถให้โปรแกรมประยุกต์ที่สร้างจาก Visual Basic 3.0 ใช้งานแฟ้ม .MDB ที่สร้างโดย Access 2.0 ได้

8.3.1 class ความเข้ากันได้ของ Access 2.0

แฟ้ม MSAJT112.DLL จะทำให้ Visual Basic 3.0 เข้ากันได้กับ Access 2.0 เพื่อทำการสร้าง, อ่าน และเขียนข้อมูลไปยังแฟ้ม .MDB ที่เข้ากันได้ทั้งใน Access 1.1 และ 2.0 นอกจากนี้ MSAJT112.DLL ยังทำให้การพัฒนางานฐานข้อมูลมีความยืดหยุ่นมากขึ้น อย่างไรก็ตามแฟ้มฐานข้อมูลของ Access 2.0 จะมีความสามารถพิเศษที่จะใช้งานได้ในโปรแกรมประยุกต์ของ Visual Basic ดังต่อไปนี้

- การบังคับขอบเขตความถูกต้องโดยการใช้กฎทางธุรกิจที่ระดับตาราง คุณสมบัติอันนี้ไม่ค่อยจะมีใน client-server RDBMS นักเนื่องจากกฎทางธุรกิจจะถูกจัดการโดยโปรแกรมประยุกต์ฐานข้อมูลด้วยตัวเองมากกว่า
- การบังคับความถูกต้องในการอ้างอิงในระดับฐานข้อมูลทำได้ดีขึ้น ความสัมพันธ์ระหว่าง primary และ foreign key สามารถที่จะทำให้เป็นแบบ one-to-many หรือ one-to-one ความถูกต้องในการอ้างอิงทั้งหมดจะถูกถ่ายทอดไปเมื่อเราทำการติดต่อกับตารางของ Access 2.0 ไปยังฐานข้อมูลของ Access 2.0 ตัวอื่น ๆ
- ทางเลือกในการทำ cascading updates และ deletions based บนคำจำกัดความของความสัมพันธ์แบบใหม่.
- ฟิลด์ในตารางของ Access 2.0 สามารถที่จะยอมรับ zero length strings ("") เพื่อรับทราบและแสดงว่าข้อมูลมีข้อผิดพลาด ในลักษณะเดียวกับค่า NULL ที่แสดงค่าที่ไม่ทราบ

8.3.2 การติดตั้ง class ความเข้ากันได้ของ Access 2.0

การติดตั้งสามารถที่จะติดตั้งได้จาก Access 2.0, Excel 5.0 และ Work 6.0 ในส่วนของการติดตั้งเพิ่มเติมจะทำการติดตั้งแฟ้ม MSAJT200.DLL (เวอร์ชั่น 2.0 ของตัวขับเคลื่อนฐานข้อมูล Jet) และเวอร์ชั่นใหม่ของ Jet ISAM drivers สำหรับ xBase/FoxPro, Paradox และ Btrieve โดย drivers

XBS200.DLL, PDX200.DLL และ BTRV200.DLL จะถูกเพิ่ม ลงในไดเรกทอรี่ ที่มีชื่อว่า \WINDOWS\SYSTEM

8.3.3 การแก้ไขและแทนที่แฟ้มของ Visual Basic 3.0 ในขณะติดตั้ง

การติดตั้ง class ของความเข้ากันได้จะทำการแก้ไขหรือแทนที่แฟ้มดังต่อไปนี้

ชื่อแฟ้มข้อมูล	ตำแหน่ง	วัตถุประสงค์
VBDB300.DLL	\WINDOWS\SYSTEM	ติดต่อ VB 3.0 กับ Jet 1.1
PDCTJET.DLL	\WINDOWS\SYSTEM	Crystal Report runtime
PDDIRJET.DLL	\WINDOWS\SYSTEM	Crystal Report runtime
VB.INI	\WINDOWS	Crystal Report runtime
SETUPWIZ.INI	\WINDOWS	VB 3.0 initialization
CRW.EXE	\VB\REPORTS	Crystal Report executable

บทที่ 9
ตัวอย่างฐานข้อมูลบนมาตรฐาน ODBC

9.1 Microsoft SQLServer for Windows NT 4.2

ถูกสร้างขึ้นมารอบรับฐานข้อมูลโปรแกรมประยุกต์ที่มีความสำคัญต่อการดำเนินงาน (mission critical database โปรแกรมประยุกต์) และเข้ากันได้กับ Sybase SQL Server

การออกแบบระบบ โดยให้มีจุดประสงค์หลักดังนี้

1) มุ่งมั่นให้เป็นผลิตภัณฑ์ที่มีคุณสมบัติสูงสุดบน Microsoft Windows NT
2) มีประสิทธิภาพสูงสุดและสามารถปรับแต่งได้สำหรับการ threading, scheduling, asynchronous I/O และหน้าที่ระบบอย่างอื่น ๆ
3) พยายามให้เป็นส่วนหนึ่งของ Windows NT ใช้งานง่ายเหมือนกับใช้ระบบปฏิบัติการ
4) ใช้ประโยชน์จากการบริการของ Windows NT ให้ได้มากที่สุด

ภายใต้ระบบ Windows NT SQL Server สามารถที่จะ

- พัฒนางานแบบ 32 bit ได้อย่างเต็มที่
- สนับสนุนเครื่องที่เป็น Symmetric Multiprocessing (SMP) เช่น COMPAQ SystemPro XL, AST Manhattan หรือ NCR 3450
- สนับสนุนฮาร์ดแวร์ที่เป็น RISC เช่น MIPS R4000 และ DEC Alpha AXP

สถาปัตยกรรมแบบ Multithreaded DBMS แบ่งเป็น 2 แบบ

1) สถาปัตยกรรมแบบ Single-Process
2) สถาปัตยกรรมแบบ Native thread-level multiprocessing

ประโยชน์ของการมีสถาปัตยกรรมแบบ Multithreaded ที่อยู่ใน server และ อยู่ในระบบปฏิบัติการด้วย ยังผลให้

ช่วยลดความยุ่งยากของระบบ โดยการช่วยจัดการ scheduling, memory allocation, queryและอื่น ๆ

- Smoother, preemptive operation
- Dynamic load balancing
- Greater robustness และ reliability
- An appropriate foundation สำหรับ salability

เพิ่มประสิทธิของ DBMS ให้สูงขึ้นในด้าน

- การ optimizer และการจัดการ cache
- I/O Subsystem
- โครงสร้างและการส่งผ่านข้อมูล

ยกตัวอย่างเช่นมี index แบบใหม่ที่สนับสนุน SMP database, การ update "In place", Automatic data striping, ปรับปรุง lock manager ให้ดีขึ้น, สร้างฐานข้อมูลให้เร็วขึ้น, มีระบบ Asynchronous checkpoint, "Free page scan" limit, Tree asynchronous I/O, Log allocation ที่มีประสิทธิเพิ่มขึ้น, การ backup และ restore ทำได้เร็วขึ้น, มี template ในหน่วยความจำ, การจัดเก็บสถิติแบบใหม่, การติดตั้งและโยกย้ายทำได้ง่ายขึ้น, มีตัวเฝ้าติดตามประสิทธิภาพแบบ graphics, มี event logging, มี event logging, มีการรักษาความปลอดภัยแบบร่วม, มีเครื่องมือสำหรับ DBA ใช้ในลักษณะ visual

นอกจากนี้ยังเพิ่มประสิทธิภาพในการใช้งานให้สูงขึ้นอีกด้วย เช่น

I. มีระบบ on-line, scheduled และ attended backups
II. structured exception handing
III. ต้องจัดการ log threshold
IV. สนับสนุน platform และ API ด้าน
    A. CUP ของ Intel เช่น 80386, 80486 และ pentium ทั้งเดี่ยวและหลาย CPU
    B. MIPS, DEC Alpha AXP
    C. Open Database Connectivity (ODBC)
    D. DB-Library
    E. โปรแกรมประยุกต์ที่พัฒนากับ Sybase Open Client Interface
    F. Microsoft Open Data Services (ODS)
V. สนับสนุน gateway ดังนี้
    A. IBM DB2, IBM SQL/DS
    B. CICS
    C. Pick
    D. IBM AS/400
    E. DEC Rdb
    F. Teradata
VI. ชุดพัฒนาสำหรับ SQL Server
    A. C++

B. Visual Basic

C. COBOL

9.2 Microsoft Access, version 2.0

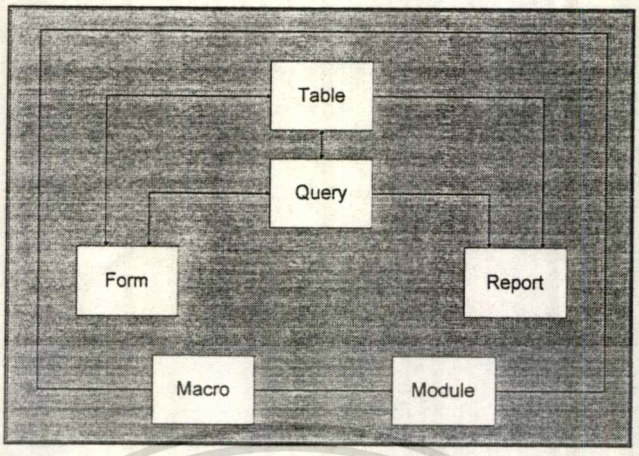
ได้มีการปรับปรุงแก้ไข้ข้อบกพร่องที่เคยมีอยู่ในรุ่น 1.1 พร้อมกับเพิ่มเติมความสามารถใหม่เข้ามาด้วยเช่น

- เพิ่มการออกแบบฐานข้อมูลเชิงสัมพันธ์ในลักษณะของกราฟิก
- รับข้อมูลขณะกรอกข้อมูล: ผู้ใช้สามารถกำหนดรูปแบบในการรับข้อมูลเองได้เช่น การรับข้อมูลหมายเลขโทรศัพท์
- Cascading Update and Deletes: ผู้ใช้สามารถเลือกให้ access ทำการเปลี่ยนแปลงหรือแก้ไว้ข้อมูลในตารางที่สัมพันธ์กันแบบอัตโนมัติได้ด้วยหากมีการเปลี่ยนแปลงหรือแก้ไขข้อมูลจากตารางหลัก
- Rushmore: ทำให้การทำงานในด้านการสืบค้นข้อมูลเร็วขึ้นกว่าเดิมมาก
- สนับสนุนการทำงานกับ ODBC อย่างเต็มที่
- คัดเลือกข้อมูลตามลำดับจำนวนที่ต้องการสืบค้น (Top-n or Top percent): การสืบค้นข้อมูลสามารถเลือกจำนวนข้อมูลจาลำดับที่จัดเรียงได้ โดยอาจจะระบุเป็นจำนวนของลำดับหรือระบุเป็นจำนวนเปอร์เซ็นต์ที่ต้องการได้
- เพิ่มหน่วยความจำสำหรับทำงานในส่วนของรายการข้อมูลที่เปลี่ยนแปลง(Transaction) ได้
- ตอบสนองการทำงานในส่วนของแบบกรอกข้อมูลและรายงานด้วยภาษาโปรแกรม Access Basic ได้โดยตรง

คุณสมบัติ

- File ข้อมูล มีเพียง 1 File มีนามสกุลเป็น MDB มีข้อมูลหลายอย่างประกอบกัน เช่น
- Table เป็นตารางข้อมูลที่ใช้เก็บข้อมูลเหมือนกับแฟ้ม DBF ของ dBASE
- Query ข้อมูลที่ถูกสร้างขึ้นมาจาก Table หรือกรอกข้อมูล
- Form รูปแบบของการแสดงข้อมูลจาก Table หรือกรอกข้อมูล
- Report รูปแบบที่สร้างขึ้นเพื่อใช้ในการออกแบบรายงาน
- Macro ชุดคำสั่งให้ทำงานโดยไม่ต้องเขียนโปรแกรม เป็นการจัดลำดับการกระทำการทำงาน
- Module เป็น Program ซึ่งเขียนด้วยภาษา Access Basic

การจัดการฐานข้อมูลด้วย ACCESS



System Table เก็บข้อมูลที่เราสร้างขึ้น เช่น

MSys Columns information about data columns

MSys Queries information about fields and expression in queries

MSys Indexes information about indexes

MSys Aces information about object security Ids

MSys Macro information about action

เครื่องมือช่วยการทำงาน

I . Wizard 2 Table, 4 Query, 5 Form, 7 Report

II. Cue Card เช่น Build database with table

A. work with table Design or Run Query

B. Design a form

C. Design Report of Mailing label

D. write Macro

III. Relation ship between table show with graphic can select

A. one - to - one

B. one - to - many

C. many - to - many

IV. Validation criteria

A. Validate data entry in many way

V. Database security

VI. Users

VII. Groups

VIII. Password Protection

IX. Permission Read, Write, Execute

X. ข้อมูลเหล่านี้เก็บใน Table System.MDA

XI. Encryption/Decryption database

XII. Transferring data with Import / Export Formats

A. Microsoft Access

B. Paradox 3.x 4.x

C. dBASE III, IV

D. Novell/Btrieve SQL (Microsoft SQL server, Sybase SQL server, Oracle)

E. FoxPro 2.0, 2.5

F. Excel 2.1, 4.0, 5.0

G. Lotus wks, wkl, wk3

XIII. Attach table เหมือน Import / Export ยกเว้น Spread Sheet Excel และ Lotus

XIV. Client/Server ใช้ ODBC diver กับคำสั่ง SQL เพื่อ Access data ของ Microsoft และ Sybase SQL Server

XV. Jet (Joint engine technology) 1.x และ 2.0

การค้นคืน:

QBE (Query by example) มี 2 แบบ คือ

Select แก้ไข data source ไม่ได้

Action สามารถแก้ไข คือ Edit, Create new table, Append, Delete ได้ โดยการสร้าง table ใหม่ คือ Dynaset

การ update ตารางเป็นแบบ Cascading Query

ใช้ Rushmore เหมือน FoxPro ใช้ 2.0 เร็วกว่า jet 1.x

Print Preview สามารถส่งข้อมูลไปสู่โปรแกรม MSWORD หรือ Excel ได้

สนับสนุน OLE, DDE

ตัวอย่าง

```
Select Count(*)
From bench
```

Where (bench.k2) AND (bench.k100=3)

count ได้ 490 record จาก 100,000 record ใช้ jet 1.x เวลา 11.53 นาที

ใช้ jet 2.0 เวลา 0.77 นาที

ข้อได้เปรียบและเสียเปรียบ:

1. Strong data-integrity

2. รูปแบบฐานข้อมูลที่แจ่มชัดที่ซึ่งทั้ง ตาราง, from, Report and โปรแกรมประยุกต์ ถูกจัดรวบรวมไว้ด้วยกัน

3. ความเร็วในการทำงานของ Access อยู่ในช่วงกลาง ๆ

4. มีรูปแบบการทำงานเป็นได้ทั้งเวิร์กสเตชันเดี่ยว ๆ ไฟล์เซิร์ฟเวอร์หลายผู้ใช้ และหน้าจอ front end for client/server

5. การรักษาความลับ และการเผยแพร่: Access มีวิธีการรักษาความลับ ที่เข้าใจง่ายสำหรับโปรแกรมประยุกต์ เช่น แฟ้ม .MDB ทั้งหมดยังสามารถเข้ารหัสไว้เพื่อป้องกันการเข้าถึงข้อมูลจากภายนอก แต่การทำเช่นนั้นจะลดความเร็วในการทำงานลง

ส่วนที่ 4
บทสรุปและวิจารณ์

บทที่ 10
บทสรุปและวิจารณ์

ในการพัฒนาระบบงานของโปรแกรมประยุกต์ในปัจจุบัน ได้มีวิธีใหม่ในการวิเคราะห์และพัฒนาคือ Object-Oriented โดยมีหลักการสำคัญหลายประการคือ Abstract Data Type, Class, SubClass inheritance และ polymorphism ซึ่งเป็นวิธีการถ่ายทอดคุณสมบัติโดยการแบ่งงานต่าง ๆ เป็นวัตถุ (Object) โดยข้อมูล (Data) และกรรมวิธี (Method) แต่ละ Object จะเป็นอิสระต่อกัน โดยจะมีการติดต่อสื่อสารระหว่าง Object ด้วยสาร (Message) เพื่อแลกเปลี่ยนข้อมูลข่าวสาร ฉะนั้นจึงทำให้การวิเคราะห์และออกแบบเป็นอิสระต่อกัน จึงทำให้ได้งานเร็วและประหยัดค่าใช้จ่ายมากขึ้น ข้อมูลต่าง ๆ ที่ได้วิเคราะห์และพัฒนายังสามารถนำมาใช้ใหม่ได้ (Reusable) การพัฒนาในระบบไคล์แอนด์-เซอร์เว่อร์ ซึ่งเป็นการกระจายฐานข้อมูลต่าง ๆ ไปตามหน่วยต่าง ๆ ของระบบเครือข่าย จะทำให้สะดวกในการที่พัฒนาเพราะไม่จำเป็นต้องทราบถึงโครงสร้างทางข้อมูลทั้งหมดเพียงแต่อาศัยภาษาค้นคืนแบบโครงสร้าง (SQL) เป็นตัวจัดในการที่จะขอข้อมูลจากแหล่งบริการอื่น ๆ

ผู้จัดทำมีความคิดเห็นว่าระบบโปรแกรมประยุกต์ที่พัฒนาแบบเชิงวัตถุ (Object-Oriented) และทำงานภายใต้ระบบเครือข่ายไคล์แอนด์-เซอร์เวอร์ เป็นวิธีที่ดีที่สุดในปัจจุบัน แต่ก็มีข้อเสียในเรื่องที่เป็นเทคโนโลยีค่อนข้างใหม่ จึงจำเป็นจะต้องมีการศึกษาเรียนรู้ใหม่ทั้งหมด และหาผู้มีประสบการณ์ในเรื่องนี้ได้น้อย

ภาคผนวก

ประมวลคำศัพท์ที่เกี่ยวข้อง

back-end เป็นคำทั่วไปที่ใช้เรียกผู้ให้บริการฐานข้อมูล (database server) หรือตัวแอพพลิเคชั่นในระบบ Client / Server

client-server database ระบบฐานข้อมูลซึ่งส่วนขับเคลื่อนฐานข้อมูลและแอพพลิเคชั่นที่ใช้ข้อมูลนั้นอยู่บนเครื่องคอมพิวเตอร์คนละเครื่องกัน และสามารถทำการติดต่อถึงกันได้โดยผ่านทางโครงข่าย ในระบบแบบนี้การประมวลผลจะถูกแยกออกไปบนเครื่องทั้งสองเครื่อง ที่ User หรือแอพพลิเคชั่นทำงานอยู่เรียกว่า Client ส่วน DBMS นั้นจะทำงานบนเครื่องที่เรียกว่า Server

database engine (ส่วนขับเคลื่อนฐานข้อมูล) เป็นส่วนหนึ่งของ DBMS ซึ่งใช้ในการจัดเก็บและจัดการกับข้อมูลตามคำสั่งจากแอพพลิเคชั่นในฐานข้อมูล

distribution transparency เป็นการจัดประเภทของระบบจัดการฐานข้อมูลแบบกระจายโดยพิจารณาจากวิธีการอ้างอิงความสัมพันธ์หรือส่วนของความสัมพันธ์ที่แอพพลิเคชั่นมองเห็น โดยทั่วไปจะแบ่งออกเป็น 3 ระดับ ได้แก่ fragmentation transparency, locationtransparency, และ local mapping transparency

entity integrity เป็น integrity ของฐานข้อมูลแบบสัมพันธ์ที่กำหนดว่า ค่าของ Primary key ห้ามเป็น Null (เว้นว่าง)

front-end application เป็นแอพพลิเคชั่นที่ทำงานที่ฝั่ง Client และมีจุดประสงค์หลักในการออกแบบให้สนับสนุนการเชื่อมต่อระหว่างผู้ใช้กับฐานข้อมูล

integrity constraint เป็นข้อกำหนดสำหรับความถูกต้องของข้อมูล ความถูกต้องที่ขึ้นกับประเภทของฐานข้อมูล (data model) เรียกว่า Structural constraint สำหรับโมเดลฐานข้อมูลแบบสัมพันธ์ จะมี Structural integrity constraint อยู่ 2 ชนิด ได้แก่ entity integrity และ referential integrity

log หรือ journal ระบบจัดการฐานข้อมูลจะทำการเก็บรายละเอียดของข้อมูลที่มีการแก้ไขไว้ ดังนั้นเมื่อข้อผิดพลาด (เช่น system fail) เกิดขึ้นกับระบบ ระบบจึงสามารถยกเลิกการแก้ไขสำหรับรายการที่ยังไม่สมบูรณ์ได้

mission critical database (ฐานข้อมูลที่มีความสำคัญต่อการดำเนินงาน) คำๆนี้มีขึ้นมาพร้อมๆกับการนำเอาเครื่องเมนเฟรมมาใช้ในการดูแลฐานข้อมูลขององค์กร ฐานข้อมูลที่เป็น

mission-critical จะมีลักษณะที่สำคัญคือ จะมีแอพพลิเคชันที่เกี่ยวข้องกับการดำเนินงานขององค์กรโดยตรง ซึ่งอาจทำให้ธุรกิจเสียหายหรือพังลงได้ ถ้าข้อมูลที่ต้องการนั้นไม่สามารถหามาให้ได้

NetWare Loadable Module (NLM) เป็นโปรแกรมหรือแอพพลิเคลันซึ่งจะทำงานที่ไฟล์เซอร์เวอร์ภาย ใต้ระบบหฏิบัติการ Novell NetWare 3.11 เป็นอย่างน้อย

precompiler เป็นตัวแปรภาษา data sublanguage เช่น SQL ที่ถูก embedded ไว้ใน host language เช่น ภาษา C , COBOL ให้เป็น host language ล้วนๆ

referential integrity เป็น integrity ของฐานข้อมูลแบบสัมพันธ์ ที่กำหนดว่า ฐานข้อมูลจะต้องไม่มีความสัมพันธ์หรือ fact ที่มีค่า foreign key ที่ไม่สอดคล้องกับค่า ของ primary key หรือเว้นว่างเอาไว้

relational database system เป็นโมเดลของฐานข้อมูลที่มีโครงสร้างข้อมูลเป็น relation มี integrity constrain ที่ประกอบด้วย entity integrity และ referential integrity และมีภาษาที่มีความสามารถเท่ากับภาษา Relational Algebra หรือ Relational Calculus เป็นอย่างน้อย

stored procedures เป็น procedure ภาษา SQL ซึ่งจะถูกเก็บไว้ในฐานะที่เป็นส่วนหนึ่งของฐานข้อมูลและตัวมันจะถูกเรียกใช้ในส่วน back-end ทั้งหมด การใช้ stroed procedures จะทำให้การประมวลผลบนฐานข้อมูลส่วนใหญ่ถูกย้ายจาก client ไปยัง server

trigger เป็น stored procedure แบบหนึ่งที่จะถูกเรียกใช้โดยอัตโนมัติเมื่อมีการทำคำสั่งภาษา SQL บางคำสั่ง (มักเป็นคำสั่งที่มีการแก้ไขข้อมูล) ผู้ผลิตบางรายเรียกมันว่า rule (กฎ) ระบบบางระบบจะนำเอา trigger มาใช้ในการดูแลความถูกต้องในการอ้างอิง (Referential Integrity) ถ้าเผื่อว่ามันไม่สนับสนุนการดูแลความถูก

กิตติกรรมประกาศ

ขอขอบคุณ คุณมนตรี หาญทวีพันธุ์ บริษัท จักรวาลคอมมิวนิเคชั่น ซีสเต็ม จำกัด ที่ให้คำปรึกษาแนะนำและข้อมูลต่าง ๆ เกี่ยวกับผลิตภัณฑ์ GUPTA

ขอขอบคุณบริษัท ไอโซเน็ท จำกัด ที่เอื้อเฟื้อข้อมูลเกี่ยวกับผลิตภัณฑ์ระบบฐานข้อมูล ไคล์แอนด์-เซอร์เวอร์ ของบริษัท ไมโครซอฟ (ประเทศไทย) จำกัด

ขอขอบคุณบริษัท ซิสโตแมท จำกัด ที่เอื้อเฟื้อสถานที่ในการจัดทำและทดสอบผลิตภัณฑ์ต่าง ๆ

ขอขอบคุณอาจารย์ศุภมิตร จิตตะยโศธร ที่ให้คำปรึกษา และแนะนำเกี่ยวกับ ปริญญานิพนธ์ ฉบับนี้

หนังสืออ้างอิง

1. ผศ.ดร.ศุภมิตร จิตะยโศธร, "อนาคตฐานข้อมูล Object-Oriented", วารสาร Computer Review. ปีที่ 10, ฉบับที่ 96,2535, หน้า 181-185.
2. วรชัย เชาว์วีระประสิทธิ์, "ภาษาคอมพิวเตอร์ในทศวรรษนี้", วารสาร Computer Review, ปีที่ 10, ฉบับที่ 96, 2535, หน้า 186-206.
3. วรชัย เชาว์วีระประสิทธิ์, "Object-Oriented เทคโนโลยีปฏิวัติวงการ", วารสาร Computer Review, ปีที่ 10, ฉบับที่ 96, 2535, หน้า 167-172.
4. ศุนย์ปืน โสมภีร์, "แนวทางใหม่กับ Client-Server", วารสาร Windows Magazine, ปีที่ 1, ฉบับที่ 4, 2536, หน้า 155-159.
5. เอก โชติชวาลวงศ์, "SQL Server", วารสารซ้อปปิ้งคอมพิวเตอร์, มีนาคม 2537, หน้า 225-270.
6. Date, C. J. , "A Guide to the SQL Standard", Addison Wesley, 1989.
7. Date, C. J. , "Database System Volume I", Addison-Wesley, 854 P., 1990.
8. Jane Richter (เรียบเรียงโดย Witch), "การกระจายข้อมูล", วารสาร Byte Thailand, ปีที่ 1, ฉบับที่ 4, 2537, หน้า 25-31.
9. Gary Entsminger, "Secrets of the Visual Basic 3.0 Masters", Indianapolis, IN,
10. Mark Hettler and Scott Higgs, "SQL Front End for Windows", Byte Magazine, 1993, Sams Publishing, 2E, ISBN 0-672-30437-6.October 1994, p.,129-138.
11. Microsoft Cor., "Microsoft SQL Server", Microsoft Press, 25 p.,1993.
12. Microsoft Cor., "Microsoft ODBC 2.0 Programmer's reference and SDK guide", Microsoft Press, 908 p., 1994.
13. Roger Jennings, "Database Developer's Guide with Visual Basic 3.0", Sams Publishing, 1134 p., 1994.
14. Sally Shlaer and Stephen J. Mellor, "Object Lifecycles Modeling the World in States", Yourdon Press, 251 p., 1992.