การพัฒนาเครื่องมือสำหรับเปรียบเทียบคุณสมบัติของระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์

(The development of a software tool for object relational DBMS comparison)

โดย

นายถิรภัทร มีสำราญ

รหัส 43067143

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน
หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
ภาคการเรียนที่ 1 ปีการศึกษา 2545
คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	การพัฒนาเครื่องมือสำหรับเปรียบเทียบคุณสมบัติของระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์
นักศึกษา	นายถิรภัทร มีสำราญ
อาจารย์ที่ปรึกษา	รศ.ดร. ศุภมิตร จิตตะยโศธร
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2545

บทคัดย่อ

การพัฒนาเครื่องมือสำหรับเปรียบเทียบคุณสมบัติของระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ มีจุดประสงค์เพื่อนำไปใช้ในการให้น้ำหนักสำหรับระบบการจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ ที่ได้ผลิตออกมาในเชิงการค้า โดยผู้ใช้ จะให้คะแนนคุณสมบัติต่างๆ ตามความต้องการใช้งาน ซึ่งคุณสมบัติดังกล่าว จำมาจากข้อมูลของแต่ละผลิตภัณฑ์ เทียบเคียงกับคุณสมบัติในทางทฤษฎีของระบบการจัดการฐานข้อมูล เพื่อช่วยให้การตัดสินใจเลือกผลิตภัณฑ์ใดๆ เป็นไปด้วยความรอบคอบและรัดกุมที่สุด ตรงตามความต้องการของผู้ใช้

Title	The development of a software tool for object relation DBMS comparison.
Student	Mr.Thiraphar Meesumrarn
Advisor	Associate Prof. Dr. Suphamit Jittayasothon
Level of Study	Master of Science in Information Technology
Major	Information Science
Academic Year	2002

ABSTRACT

The development of a software tool for object relational DBMS comparison has a purpose to use for weighing in a software tool for object relational DBMS that were commercially produced. According to the main purpose, the users will mark any property in which come from each information of the products by comparing with theoretical property of relational DBMS. The information will useful for making user's decision precisely and accurately.

กิตติกรรมประกาศ

โครงการพัฒนาระบบงานนี้สำเร็จได้เพราะได้รับการส่งเสริม และสนับสนุนจากบุคคลหลายท่าน กระผมจึงใคร่ขอกราบขอบพระคุณ

- บิดาและมารดาที่ได้อบรมสั่งสอนและสนับสนุนส่งเสียให้ได้เล่าเรียนจบประสบความสำเร็จในการศึกษา
- อาจารย์ศุภมิตร จิตตะยโศธรที่ได้ให้ความกรุณาให้คำปรึกษาแนะนำสิ่งต่าง ๆ ในโครงการพัฒนาระบบงาน
- อาจารย์ทุก ๆ ท่านที่ได้ประสิทธิ์ประสาทความรู้ หลักวิชาการต่าง ๆ เพื่อเป็นพื้นฐานในการดำเนินชีวิตและการทำงาน
- เจ้าหน้าที่คณะเทคโนโลยีสารสนเทศทุกท่านที่ให้คอยอำนวยความสะดวกอย่างดี
- เพื่อน IS10 ทุกท่านที่ได้กำลังใจ และให้คำแนะนำดี ๆ เสมอมา

นายถิรภัทร มีสำราญ

สารบัญ

หน้า

บทคัดย่อภาษาไทย..... I

บทคัดย่อภาษาอังกฤษ..... II

กิตติกรรมประกาศ..... III

สารบัญ..... IV

สารบัญตาราง..... VI

สารบัญรูปภาพ..... VII

บทที่

1. บทนำ..... 1
   - 1.1 ลักษณะของปัญหา..... 1
   - 1.2 พิจารณาบทความตีพิมพ์..... 1
   - 1.3 เนื้อหาของรายงาน..... 2
2. หลักการที่เกี่ยวข้อง..... 3
   - 2.1 ระบบจัดการฐานข้อมูลเชิงสัมพันธ์..... 3
   - 2.2 ระบบจัดการฐานข้อมูลเชิงวัตถุ..... 6
   - 2.3 ระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์..... 8
   - 2.4 SQL3..... 16
3. คุณสมบัติในระบบจัดการฐานข้อมูล..... 18
   - 3.1 Informix-Universal server..... 18
   - 3.2 Oracle 8..... 32
4. หลักเกณฑ์การให้คะแนน..... 46
   - 4.1 แบบสอบถามคะแนนคุณสมบัติ..... 46
   - 4.2 แบบสอบถามคะแนนประเภทงาน..... 48
5. การพัฒนาเครื่องมือเปรียบเทียบคุณสมบัติของระบบจัดการฐานข้อมูลเชิง-วัตถุสัมพันธ์..... 49
   - 5.1 การออกแบบระบบ..... 49

6. สรุปและข้อเสนอแนะ 61
6.1 สรุปผลการพัฒนาระบบ 61
6.2 ข้อเสนอแนะ 61
บรรณานุกรม..... 62
ภาคผนวก..... 63
ประวัติผู้เขียน..... 67

สารบัญตาราง

หน้า

ตารางที่

3.1 การเปรียบเทียบ Informix-Universal server กับข้อกำหนดของ Stonebrakers ORDBMS..... 31

3.2 Conclusion on Oracle8 and Stonebrakers definition..... 39

3.2 การเปรียบเทียบ Informix-Universal Server และ Oracle กับข้อกำหนดของ Stonevrakers ORDBMS 43

4.1 แสดงแบบสอบถามคะแนนคุณสมบัติของผลิตภัณฑ์ 46

5.1 แสดง Data Dictionary ของ TABLE PRODUCT..... 59

5.2 แสดง Data Dictionary ของ TABLE WORK..... 59

5.3 แสดง Data Dictionary ของ TABLE FEATURE 59

5.4 แสดง Data Dictionary ของ TABLE PRODUCTSCORE 60

5.5 แสดง Data Dictionary ของ TABLE WORKSCORE 60

5.6 แสดง Data Dictionary ของ TABLE FEATURESCORE 60

สารบัญรูปภาพ

หน้า

ภาพที่

2.1 ตัวอย่างง่ายๆ ของ SQL92- ที่แปลมาจากโค้ดภาษา C 4

2.2 รูปแบบ SQL-92 สำหรับตาราง Employee 5

2.3 ตารางแอปพลิเคชั่นของระบบฐานข้อมูล 9

3.1 แสดงองค์ประกอบหลักของ DataBlade module 19

3.2 แสดงองค์ประกอบหลัก 4 ส่วนของ DataBlade module 20

3.3 แสดง Comples types ที่ Informix Universal server สนับสนุน..... 23

3.4 แสดง ระบบ Cartridges 34

3.5 แสดงกระบวนการของการเรียกใช้การทำงานจากภายนอก 35

5.1 แสดง Context Diagram ของเครื่องมือเปรียบเทียบฯ 51

5.2 แสดง Level 0 ของเครื่องมือเปรียบเทียบฯ 52

5.3 แสดง Level 1 ของ Process ออกแบบคำถาม 54

5.4 แสดง Level 1 ของ Process การให้คะแนนผลิตภัณฑ์ 55

5.5 แสดง Level 1 ของ Process การให้คะแนนงาน 56

5.6 แสดง Level 1 ของ Process การพิจารณาผล 57

5.7 แสดง Entity Relationship Diagram..... 58

6.1 แสดงจอภาพสำหรับออกแบบคำถาม 63

6.2 แสดงจอภาพสำหรับการให้คะแนนผลิตภัณฑ์ 64

6.3 แสดงจอภาพสำหรับการให้คะแนนความสนใจเฉพาะงาน 65

6.4 แสดงจอภาพสำหรับการตัดสินใจเลือกผลิตภัณฑ์ 66

6.5 แสดงจอภาพสำหรับการตัดสินใจเลือกผลิตภัณฑ์แบบกราฟ 66

บทที่ 1
บทนำ

1.1 ลักษณะของปัญหา

เมื่อองค์กรหรือหน่วยงานใดๆ มีความต้องการจัดหาวัสดุ ครุภัณฑ์ เข้ามาใช้ในหน่วยงานแล้ว จะต้องมีการกำหนดคุณลักษณะของสิ่งที่เราต้องการ ยื่นซองประกวดราคา แล้วทำการพิจารณาคุณลักษณะที่ผู้ผลิตแต่ละรายได้เสนอแข่งขันกันเข้ามา

แต่เราจะทราบได้อย่างไรว่า สิ่งที่ผู้จำหน่ายผลิตภัณฑ์ยื่นซองประกวดราคาเข้ามานั้น ตรงกับความต้องการหรือมีคุณสมบัติครบถ้วนตามที่เรากำหนดไว้ หากมีคุณสมบัติดังกล่าวครบถ้วน จะทราบได้อย่างไรว่าผลิตภัณฑ์ของใครทำงานได้ดีกว่า ในราคาเหมาะสม เป็นการยากที่เราจะพิจารณาสิ่งต่างๆ เหล่านี้ได้อย่างรอบคอบและครบถ้วน โดยเฉพาะอย่างยิ่งหากขาดความเข้าใจในผลิตภัณฑ์เหล่านั้นอย่างชัดเจน

ไม่ว่าจะเป็นผลิตภัณฑ์ใดๆ รวมถึงผลิตภัณฑ์สำหรับจัดการฐานข้อมูล เมื่อนำไปใช้งานในหน่วยงานใดๆ แล้ว การพิจารณาเลือกซื้อ ก็ควรให้ได้ตรงกับความต้องการ ทั้งคุณภาพและราคา แต่จะทราบได้อย่างไรว่า ผลิตภัณฑ์ไหนเป็นตัวเลือกที่เหมาะสมที่สุด เพราะผลิตภัณฑ์ทางด้านฐานข้อมูลก็มีอยู่ด้วยกันหลายประเภท เช่น ฐานข้อมูลเชิงสัมพันธ์ (RDBMS) ฐานข้อมูลเชิงวัตถุ (ODBMS) หรือฐานข้อมูลเชิงวัตถุสัมพันธ์ (ORDBMS) เป็นต้น นอกจากนั้นฐานข้อมูลแต่ละประเภทต่างก็มีผู้ผลิตออกมาจำหน่ายอยู่หลายรายด้วยกัน ดังนั้นจึงเป็นการยากที่เราจะพิจารณาได้อย่างครบถ้วน

1.2 พิจารณาบทความตีพิมพ์

Lisbeth Bergholt เปรียบเทียบคุณสมบัติที่มีของระบบจัดการฐานข้อมูล กับข้อกำหนดของ stonebraker แต่ไม่ได้กล่าวถึงความสามารถของคุณสมบัติที่มี ทำให้ไม่สามารถตัดสินใจได้ว่า ผลิตภัณฑ์ของใคร ดีกว่า

DB Magazine แสดงคุณสมบัติทาง physical ที่ควรมีในระบบจัดการฐานข้อมูล แต่ไม่ระบุเกณฑ์ที่ใช้

1.3 เนื้อหาของรายงาน

จากปัญหาและการพิจารณาบทความตีพิมพ์ดังกล่าวข้างต้น เห็นได้ว่า ยังไม่มีงานชิ้นใดที่ทำการเปรียบเทียบผลิตภัณฑ์ฐานข้อมูลเชิงวัตถุสัมพันธ์ ดังนั้นเนื้อหาของรายงานฉบับนี้จะบรรยายถึง คุณลักษณะของระบบจัดการฐานข้อมูล โดยมีขอบเขตอยู่ที่ระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์เท่านั้น โดยเริ่มต้นจากทฤษฎีที่เกี่ยวข้องของระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ ทั้งในส่วนของระบบจัดการฐานข้อมูลเชิงสัมพันธ์ และเชิงวัตถุ พร้อมทั้งกล่าวถึงข้อด้อยของทั้งสองระบบว่ามีอะไรบ้าง และเหตุดังกล่าวนำมาเป็นจุดเริ่มต้นของระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ได้อย่างไร

บทที่ 3 พิจารณาผลิตภัณฑ์เป็นรายตัว โดยศึกษาเฉพาะ Oracle 8i และ Informix Universal Server เท่านั้น

บทที่ 4 หลักเกณฑ์การให้คะแนน โดยเป็นการให้คะแนนแบบ 3 ตัวเลือก

บทที่ 5 เป็นการออกแบบซึ่งประกอบไปด้วยขั้นตอนการออกแบบ แผนภาพบริบท และ ER Diagram

บทที่ 6 สรุปและข้อเสนอแนะในมุมมองของการพัฒนาโปรแกรม พร้อมทั้งการใช้งานโปรแกรม

บทที่ 2
หลักการที่เกี่ยวข้อง

ฐานข้อมูลเชิงวัตถุสัมพันธ์ เป็นระบบจัดการฐานข้อมูลที่มีคุณสมบัติของฐานข้อมูลเชิงสัมพันธ์และเชิงวัตถุอยู่ด้วยกัน เพื่อลดจุดบกพร่องที่เคยมีในระบบจัดการฐานข้อมูลทั้ง 2 รูปแบบ ในบทนี้จะกล่าวถึงระบบจัดการฐานข้อมูลเชิงสัมพันธ์ และเชิงวัตถุ ว่าฐานข้อมูลทั้งสอง มีลักษณะอย่างไร และมีข้อบกพร่องใด จึงทำให้เกิดระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ขึ้นมา

2.1 ระบบจัดการฐานข้อมูลเชิงสัมพันธ์

ปี 1974 นักวิจัยของ IBM ชื่อ "Ted Codd" ได้ตีพิมพ์เอกสารที่อ้างถึงวิธีการใหม่เพื่อการจัดการ "การใช้ธนาคารข้อมูลขนาดใหญ่ร่วมกัน" ในเอกสารดังกล่าว Codd ได้นิยาม 2 วัตถุประสงค์สำหรับการจัดการการใช้ข้อมูลร่วมกัน อันดับแรกคือ data independence นั่นคือ applications ที่ถูกสร้างเพื่อใช้งานฐานข้อมูลจะไม่ขึ้นอยู่กับลักษณะทางกายภาพใดๆ ทั้งปวง เนื่องจากฐานข้อมูลจะจัดการสิ่งเหล่านั้นเอง ลำดับถัดมา เขาอ้างถึงกฎต่างๆ เพื่อความถูกต้องในการใช้ข้อมูลร่วมกัน นั่นคือ consistent โดยการกำจัดสิ่งที่ซ้ำซ้อนในขั้นตอนการออกแบบฐานข้อมูล

ในเอกสารของ Codd จงใจที่จะยกเลิกข้อพิจารณาอื่นๆ ว่าทำอย่างไร model ถึงจะถูกทำให้เป็นผล เขาต้องการที่จะกำหนด abstraction ของปัญหาในการจัดการ information ได้แก่ การสร้างกรอบแนวความคิด และดำเนินการภายใต้แนวความคิดนั้น

รูปแบบเชิงสัมพันธ์ของ Codd ประกอบด้วย 3 ส่วนคือ รูปแบบข้อมูล หมายถึง วิธีการทำงานในภาษาชั้นสูง, ชุดของการออกแบบ ซึ่งทำให้แน่ใจได้ว่าได้ลดปัญหาความซ้ำซ้อนของข้อมูล รูปแบบเชิงสัมพันธ์ของ Codd มองข้อมูลเหมือนกับการเก็บลงในตาราง ประกอบไปด้วยตัวแปรของ rows (หรือ record) คล้ายๆ กับสมุดโทรศัพท์ หรือการบันทึกวันเกิด, วันตาย และวันแต่งงาน จะเป็นการดีถ้าทำเป็นตาราง โดยแต่ละรายการประกอบด้วยข้อมูลที่แตกต่างกัน แต่อยู่ในรูปแบบเดียวกันทุกรายการ

รูปแบบเชิงสัมพันธ์ยังอ้างถึงตัวดำเนินการทางตรรกะจำนวนมาก ซึ่งนำมาใช้กับข้อมูล ตัวดำเนินการเหล่ามีความหมายรวมถึงการได้มาซึ่งแถวบางแถว (ทุกชื่อในสมุดโทรศัพท์ที่มีชื่อต้นว่า "Brown") คอลัมน์บางคอลัมน์ (ชื่อและหมายเลขเท่านั้น) หรือข้อมูลจากการรวมกันของ table (ใครที่แต่งงานแล้ว พร้อมกับระบุหมายเลขโทรศัพท์)

โดยการยืมเทคนิคทางด้านคณิตศาสตร์ ทางด้านตรรกศาสตร์ Codd ได้มาซึ่งชุดของการออกแบบ ที่สามารถใช้รับประกันว่าโครงสร้างฐานข้อมูลนั้นปราศจาก ความซ้ำซ้อน อันเป็นปัญหาในระบบต่างๆ ต่อมาได้แพร่หลายอย่างกว้างขวาง เช่น แนวความคิดพื้นฐานทางทฤษฎีของ Normal Forms ระบบภายใต้กฎของ normal form สามารถแน่ใจได้ว่า การออกแบบฐานข้อมูลนั้นปลอดจากความซ้ำซ้อน และปัญหาอื่นๆ ของความผิดปกติในข้อมูล

2.1.1 การทำงานของฐานข้อมูลเชิงสัมพันธ์

เริ่มจากปี 1970 กลุ่มของโครงงานวิจัยคู่ขนานยังไม่ได้ทำงานบน RDBMS มันเป็นสิ่งที่ยากมาก จนกระทั่งในปี 1980 ผลิตภัณฑ์ทางด้าน RDBMS ได้รองรับงานระดับสูง ได้แก่ การประมวลผลออนไลน์ ซึ่งให้การสนับสนุนที่ดีแก่เทคโนโลยียุคแรกๆ

ถึงแม้ว่า เทคโนโลยีทางด้าน RDBMS จะเพิ่งเกิด แต่ก็ได้รับความนิยม เพราะผลิตภัณฑ์ที่ออกมาในตอนแรกๆ มันถูกกว่า เร็วกว่า และง่ายกว่า ในการสร้างระบบข้อมูลข่าวสาร สำหรับจำนวนที่เพิ่มขึ้นของ application, การสนับสนุนในทางการเงินที่มากขึ้นเกี่ยวกับฮาร์ดแวร์และใช้คนน้อยลงRDBMS ได้ทำในสิ่งที่เป็นไปได้ในการจัดการระบบข้อมูลข่าวสารนั้น ขณะที่มุมมองทางด้านการจัดการธุรกิจ รู้สึกว่าแพง

เพื่อแสดงให้เห็นชัดๆ ถึงความแตกต่างระหว่าง วิธีการเชิงสัมพันธ์ กับ วิธีที่มีอยู่ก่อนเชิงสัมพันธ์ โปรแกรมภาษา C 400 บรรทัด สามารถถูกแทนโดยใช้ชุดคำสั่งของ SQL-92 ตามที่แสดงในรูปที่ 2.1

```
CREATE TABLE Employees (
    Name     VARCHAR(128),
    DOB      DATE,
    Salary   DECIMAL(10,2),
    Address  VARCHAR(128)
);
```

รูปที่ 2.1 ตัวอย่างง่ายๆ ของ SQL-92 ที่แปลมาจากโค้ดภาษา C

โค้ดในรูปที่ 2.1 มีความสามารถมากกว่าการทำงานที่เขียนด้วยภาษา C เพราะ RDBMSs ได้ให้การรับรองการทำรายการ การเปลี่ยนแปลงของข้อมูล โดยทำการ locking, logging, backup และ recovery ให้โดยอัตโนมัติ เพื่อรับรองว่าข้อมูลที่ได้จัดเก็บลงไปนั้น มีความถูกต้อง อีกทั้ง

RDBMS ได้เตรียมคุณสมบัติด้านความปลอดภัยเอาไว้ ในตารางต่างๆ ในฐานข้อมูลเดียวกัน สามารถกำหนดการเข้าถึงที่แตกต่างกันในแต่ละกลุ่มของผู้ใช้ได้ นี่คือคุณสมบัติทั้งหมดที่มีอยู่ หมายถึงผู้พัฒนาให้ความสนใจที่คุณสมบัติของระบบและใช้การทำงานอันน้อยนิดบนเทคนิคที่ซับซ้อนเหล่านั้น

ด้วย RDBMSทุกวันนี้ ได้แสดงให้เห็นตาราง Employees อย่างง่ายๆ ในรายการที่ 1-1 การกำหนดสิ่งอื่นๆ เพิ่มเติมนั้นแสดงไว้ในรูปที่ 2.2

```
CREATE TABLE Employees (
    FirstName    VARCHAR(32) NOT NULL,
    Surname  VARCHAR(64) NOT NULL,
    DOB      DATE NOT NULL,
    Salary     DECIMAL(10,2) NOT NULL
                        CHECK ( Salary > 0.0 ),
    Address_1    VARCHAR(64) NOT NULL,
    Address_2    VARCHAR(64) NOT NULL,
    City       VARCHAR(48) NOT NULL,
    State      CHAR(2) NOT NULL,
    ZipCode  INTEGER NOT NULL,
    PRIMARY   KEY ( Surname, FirstName, DOB )
);
```

รูปที่ 2.2 รูปแบบ SQL-92 สำหรับตาราง Employees

ทุกวันนี้ทั่วโลก ตลาดสำหรับ ซอฟต์แวร์ RDBMS, การบริการ และโปรแกรมประยุกต์ ได้มีการใช้ฐานข้อมูลเชิงสัมพันธ์มากกว่า 5 หมื่นล้านเหรียญทุกปี ภาษาสำหรับฐานข้อมูลคือ SQL-92 ยังคงง่าย ให้ความคุ้นเคย และมีความยืดหยุ่นสำหรับจัดการข้อมูลทางธุรกิจใดๆ โครงสร้างทั้งหลายทางวิศวกรรมภายใต้ระบบฐานข้อมูลเชิงสัมพันธ์ เพื่อบรรลุถึงระดับของประสิทธิภาพทั้งปริมาณผลลัพธ์ของข้อมูลและเวลาตอบสนองของ query จากข้อมูลข่าวสารขนาดใหญ่

2.1.2 ปัญหาของ RDBMSs

เริ่มต้นมาประมาณปี 1980 ข้อบกพร่องหลายๆ อย่างในผลิตภัณฑ์ของ RDBMS เริ่มได้รับคำเตือนจำนวนมาก ข้อบกพร่องอันดับแรกคือ ส่วนสำคัญของภาษาเชิงสัมพันธ์ ได้แก่ SQL-92 มีข้อจำกัดในความเกี่ยวข้องกับส่วนสำคัญหลายๆ ส่วน ตัวอย่างเช่น SQL-92 สนับสนุนกลุ่มของข้อมูลที่มีอยู่อย่างจำกัด ซึ่งมีให้เฉพาะตัวเลข(Number) และสายอักขระ (string) แต่แอปพลิเคชั่นทางด้านฐานข้อมูลส่วนใหญ่ เริ่มที่จะรวมเอา object ที่มีความซับซ้อน ตัวอย่างเช่น จุดทางภูมิศาสตร์, ข้อความ และสัญญาณข้อมูลดิจิตอล ปัญหาตามมาก็คือจะมีการใช้งานข้อมูลนั้นอย่างไร แนวคิดก็คือ คำถามง่ายๆ ของโครงสร้างข้อมูลที่ซับซ้อน กลับต้องใช้ query ของ SQL-92 ที่ยืดยาว

ข้อบกพร่องลำดับที่สองคือ รูปแบบเชิงสัมพันธ์ได้รับ ข้อบกพร่องทางโครงสร้าง ตารางเชิงสัมพันธ์บางและไม่ได้ตระเตรียมการสนับสนุนที่ดีสำหรับโครงสร้างเชิงซ้อน ตัวอย่างเช่น เซตและอาร์เรย์ เช่นเดียวกัน ก็ให้ความมั่นใจได้กับชนิดของความสัมพันธ์ เช่น subtype ระหว่าง object ของฐานข้อมูล ซึ่งยากในการแสดงให้เห็นในรูปแบบ (subtype เกิดขึ้นเมื่อเรากล่าวถึงสิ่งใดสิ่งหนึ่ง เช่น salesPerson เป็น subtype ของสิ่งๆ หนึ่ง ได้แก่ Employee) SQL-92 สนับสนุนเฉพาะแถวของตารางหรือคอลัมน์

ข้อบกพร่องลำดับที่ 3 คือ เทคโนโลยี RDBMS นั้น ไม่สนับสนุนวิศวกรรมซอฟต์แวร์ทางด้าน Object ซึ่งแพร่หลายในวงการอุตสาหกรรม วิธีการทาง OO ช่วยลดต้นทุนและปรับปรุงคุณภาพของระบบสารสนเทศ โดยการหรือวิธีการคิดคำนวณสมัยใหม่ สามารถซ่อนไว้ภายใต้กลุ่มของวิธีการติดต่อ สิ่งเหล่านี้ยอมให้นักพัฒนาโปรแกรมใช้งานฟังก์ชันการทำงานที่ซับซ้อนโดยไม่จำเป็นต้องเข้าใจว่ามันทำงานอย่างไรรับเอามุมมองทาง object ไปพัฒนาซอฟต์แวร์ ซึ่งเป็นการรวมกันของข้อมูล และวิธีการของสิ่งที่มีอยู่ในโลกความเป็นจริง เข้าไว้ในหน่วยทางซอฟต์แวร์ หรือ component โครงสร้างข้อมูลที่ซับซ้อน หรือวิธีการคิดคำนวณสมัยใหม่สามารถซ่อนไว้ภายใต้กลุ่มของวิธีการติดต่อ สิ่งต่างๆ เหล่านี้ยอมให้นักพัฒนาโปรแกรมใช้งานฟังก์ชันการทำงานที่ซับซ้อนโดยไม่จำเป็นต้องเข้าใจว่ามันทำงานอย่างไร

แม้ว่ารูปแบบเชิงสัมพันธ์จะเป็นการจัดการที่ดีสำหรับข้อมูลข่าวสารต่างๆ แต่การเกิดกลุ่มปัญหาขึ้นสำหรับเทคโนโลยีทางด้าน RDBMS ทำให้มันควรได้รับการปรับปรุง

2.2 ระบบจัดการฐานข้อมูลเชิงวัตถุ

ระบบการจัดการฐานข้อมูลเชิงวัตถุ (OODBMS) มาจากเทคนิคการโปรแกรมแบบ OO ที่อยู่ในการจัดการข้อมูล สำหรับแอปพลิเคชั่นที่มากมาย ประสิทธิภาพ ความยืดหยุ่น และ การบริหารค่าใช้

จ่ายของ OODBMSs มีความสามารถที่ดีกว่า RDBMSs หรือ ORDBMSs ข้อได้เปรียบเด่นๆ ของ OODBMSs คือความสามารถในการรวมภาษาเชิงวัตถุ และ DBMS เข้าด้วยกัน โดยแท้จริงแล้ว ส่วนมาตรฐานหลักๆ ของ OODBMSs ได้แก่ การจัดการ กลุ่มฐานข้อมูลวัตถุ (ODMG) นั่นคือการกำหนดระบบของ OODBMS ให้มีความสามารถทางด้านฐานข้อมูลเข้ากับ ความสามารถการโปรแกรมเชิงวัตถุ แนวความคิดที่ซ่อนอยู่ภายใต้สิ่งนี้ คือการใช้งานของผู้พัฒนาโปรแกรม มันจะเป็นประโยชน์เมื่อไม่สนใจที่จะถามว่า object ทำงานอย่างไรภายใต้การติดต่อที่ให้มานั้น การจัดเก็บและเรียกคืนจะทำอย่างไร นักพัฒนาทั้งหลายมีหน้าที่สร้างแอปพลิเคชั่นนั้นโดยใช้คุณสมบัติการโปรแกรมภาษาเชิงวัตถุ เช่น C++ Samlltalk หรือ Java และ OODBMS จะดูแลเรื่อง การเก็บข้อมูล, การควบคุมความคงสภาพ และสื่อบันทึกข้อมูล

ในความสามารถที่รวมกันอยู่นี้ OODBMSs ได้รับความสามารถอันมากมายเหล่านั้น และใช้งานตามคุณสมบัติหลักที่ได้มาจากรูปแบบเชิงวัตถุเพื่อที่จะแก้ปัญหาประเภทข้อมูลที่จำกัดที่บังคับอยู่ใน SQL-92 คุณสมบัติของ OODBMSs ส่วนมาก ระบบสามารถเพิ่มขยายได้ โดยใช้เทคนิคนี้ OODBMS สามารถใช้ วัตถุที่ซับซ้อนเป็นส่วนหนึ่งของแอปพลิเคชั่นและเก็บมันโดยตรง OODBMS สามารถเรียกใช้การอ้างถึงวิธีการบนวัตถุเหล่านี้ อย่างใดอย่างหนึ่งโดยเรียกผ่านไปยังวัตถุหรือ ใช้วิธีติดต่อแบบ query และสุดท้าย โครงสร้างมากมายที่กำหนดไว้ใน SQL-92 ถูกครอบคลุมโดยแนวความคิดทางด้าน OO ตัวอย่างเช่น การถ่ายทอดคุณสมบัติ (inheritance) และ การใช้เซต และ อาร์เรย์

ผลิตภัณฑ์ทางด้าน OODBMS เริ่มปรากฏให้เห็นในหน่วยงานทางการศึกษาและ สร้างความสนใจให้ทางด้านธุรกิจเมื่อปี 1990 และวันนี้มูลค่าทางการตลาดของผลิตภัณฑ์ OODBMS อยู่ที่ 500 ล้านเหรียญต่อปี ในจำนวนแอปพลิเคชั่นทั้งหลาย ตัวที่มีชื่อเสียงที่สุดได้แก่ คอมพิวเตอร์ช่วยในการออกแบบ หรือการผลิต (CAD/CAM) ค่าใช้จ่ายสำหรับการสร้างระบบที่แข็งแกร่งเพื่อจัดการทั้งฐานข้อมูลและแอปพลิเคชั่นคือการทำสมดุลย์ของประสิทธิภาพการส่งต่อของระบบ

2.2.1 ปัญหาของ OODBMSs

เป็นที่น่าเสียดาย ความสามารถอันมากมายของ OODBMS ถ้าการเรียนรู้สิ่งเหล่านี้ได้แพร่หลายเมื่อ 20 ปีที่แล้ว อันดับแรก ผู้ผลิต OODBMS ได้พบความยากลำบากอีกครั้งของการเชื่อมโยงการออกแบบฐานข้อมูลให้เข้าใกล้กับกับการออกแบบแอปพลิเคชั่น การบำรุงรักษาและการปรับปรุง ระบบข้อมูลข่าวสารทางที่มีพื้นฐานทางด้าน OODBMS ยากต่อการดำเนินการ อย่างที่สอง พวกเขาต้องเรียนรู้การใช้งานภาษานั้นอีกครั้ง เช่น SQL-92 สร้างผลกำไรให้อย่างมากมายจนองค์กรสามารถซื้ออุปกรณ์อัตโนมัติต่างๆ ได้ตามที่ต้องการ คุณสามารถซื้อฮาร์ดแวร์ได้เสมอ

แต่ไม่ใช่เวลา อย่างที่สาม ได้มีการค้นพบข้อเท็จจริงอีกครั้งว่ามีข้อบกพร่องของรูปแบบข้อมูลมาตรฐานนำไปสู่การออกแบบที่ผิดพลาดและไม่ความไม่เข้ากัน

ทั้งๆ ที่เป็นข้อบกพร่อง แต่เทคโนโลยีทางด้าน OODBMS ได้เตรียมวิธีการที่มีประสิทธิภาพสำหรับประเภทของการจัดการปัญหาของข้อมูล แนวความคิดหลากหลายที่ถูกริเริ่มโดน OODBMS มีให้เห็นจริงแล้ว โดยสิ่งเหล่านั้นมีประโยชน์อย่างมากและสามารถพบได้ใน ORDBMSs ระบบเชิงวัตถุสัมพันธ์ได้รวมเอาคุณสมบัติทั้ง complex object extensibility, encapsulation, inheritance และวิธีการติดต่อที่ดีกับภาษาแบบ OO

2.3 ระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์

ในทางด้านธุรกิจแล้ว ส่วนใหญ่จะใช้งานระบบจัดการฐานข้อมูลเชิงสัมพันธ์ เหตุที่มีการใช้เชิงสัมพันธ์เพราะว่ามันง่าย มีภาษา query และเข้าใจได้เป็นอย่างดีภายใต้ทฤษฎีทางด้านคณิตศาสตร์ (เช่น Oracle, Sybase และ Informix มีส่วนแบ่งทางการตลาดที่ใหญ่มากในตลาดนี้) ความท้าทายใหม่ที่เกิดขึ้นสำหรับระบบเชิงสัมพันธ์ นั่นคือการมาถึงของวิธีการสำหรับ client server และวิธีการสำหรับ internet การนำเสนอของระบบ มีการวิวัฒนาการไปยังการนำเสนอในภาษาเชิงวัตถุ และจากความสนใจนั้น ระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์จึงได้เกิดขึ้น

ขอบเขตของการวิจัย ได้ให้ความสนใจอย่างมากในการจัดเก็บวัตถุ สิ่งหนึ่งที่ได้รับความสนใจมากนั่นคือการใช้การวิเคราะห์ การออกแบบ และ การเขียนโปรแกรม เชิงวัตถุ ภายใต้รูปแบบระบบการจัดการฐานข้อมูลเชิงวัตถุ โดยไม่ใช้วิธีการทางคณิตศาสตร์ เหมือนที่เราได้จากภายใต้รูปแบบการจัดการฐานข้อมูลเชิงสัมพันธ์ โดยปราศจาก relational algebra ระบบการจัดการฐานข้อมูลเชิงวัตถุจะทำความเข้าใจได้ไม่ดีไปกว่าระบบการจัดการฐานข้อมูลเชิงสัมพันธ์

ได้มีความพยายามรวม 2 วิธีการดังกล่าวเข้าด้วยกัน ในส่วนต่างๆ ทุกส่วนของการพัฒนาระบบการจัดการขนาดใหญ่ การรวมกันของสองวิธีการต้องการความเข้าใจเป็นอย่างดีในรูปแบบวัตถุและเชิงสัมพันธ์ ว่ามันมีอะไรที่คล้ายกัน หรือแตกต่างกัน แนวคิดนั้นนะต้องรวมอยู่ในรูปแบบเดียวโดยมีคุณสมบัติของทั้งสองวิธีการ

[Sto96] ได้กำหนดหมวดหมู่ใหม่ของระบบการจัดการฐานข้อมูล นั่นคือ ระบบการจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ เขามอง ระบบการจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์เป็นประเภทหนึ่งของระบบการจัดการฐานข้อมูล ที่พยายามรวมระบบจัดการฐานข้อมูลเชิงสัมพันธ์ และระบบจัดการฐานข้อมูลเชิงวัตถุ เข้าไปเป็นประเภทของ DBMS

[Sto96] บรรยายความต้องการที่แตกต่างกันในการเก็บข้อมูลอย่างง่ายๆ โดยการเปรียบเทียบแอปพลิเคชั่นผ่านทางตารางขนาด 2x2 แกนทั้ง 2 คือ ข้อมูลที่ซับซ้อน และความต้องการในการค้น

หาข้อมูลที่มีความสะดวกแตกต่างกัน Stonebraker กล่าวว่า ความต้องการในตารางกรอบบนขวา ไม่มีทางพบได้ใน DBMS ที่มีอยู่ หมายความว่ามันไม่สามารถจัดการข้อมูลที่ซับซ้อน และให้การตอบสนองที่ดีในการค้นหาข้อมูล แต่ ORDBMS ให้ความต้องการที่อยู่ในกรอบบนขวาของตาราง สิ่งที่เขาไดบรรยายไม่ได้บอกอย่างชัดเจนว่ามุมบนขวาของตารางจะให้ระบบฐานข้อมูลที่เหมาะสม RDBMS จำนวนหนึ่งได้มีการพัฒนาขึ้น เพื่อความเป็นไปได้ในการใช้งานข้อมูลที่นอกเหนือไปจากข้อมูลง่ายๆ เพื่อให้ได้ความสะดวกในการค้นหาฐานข้อมูลเชิงวัตถุ

รายงานฉบับนี้ทำการเปรียบเทียบคุณสมบัติต่างๆกับ 2 DBMSคือ

- Oracle 8
- Informix Universal Server

และยังมี DBMS อื่นที่น่าสนใจได้แก่

- DB2/2 or DB2/6000
- Sybase
- Ingres

[Sto96] ได้กำหนด DBMS ใหม่นี้ว่าเป็น ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์อย่างสมบูรณ์ เขาแสดงให้เห็นความต้องการที่แตกต่างผ่านทางตารางแอปพลิเคชั่นขนาด 2x2 แกนทั้งสองคือความซับซ้อนของข้อมูล และความต้องการความสะดวกในการค้นหาที่แตกต่างกัน stonebraker กล่าวว่า DBMS ที่มีอยู่นั้นไม่เพียงพอกับความต้องการที่อยู่ในมุมบนขวาของตารางนี้ นั่นหมายถึงมันไม่สามารถจัดการกับข้อมูลที่มีความซับซ้อน และในขณะเดียวกันก็ไม่ได้ให้ความสะดวกในการค้นหา

Query	Ex. The usual EMP-DEPT System	Ex. Databases including maps, slides, videos and operations on theese
No Query	Ex. A standard text processing system	Ex.Systems with a tigth integration with a oo-programming language ie C++ , Smalltalk.
	Simple Data	Complex Data

รูปที่ 2.3 ตารางแอปพลิเคชั่นของระบบฐานข้อมูล

ORDBMS พบความต้องการที่อยู่ด้านมุมบนขวาของตาราง DBMS ใหม่ที่เขากล่าวถึงเรียกว่า ฐานข้อมูลเชิงวัตถุสัมพันธ์อย่างสมบูรณ์

ฐานข้อมูลเชิงวัตถุสัมพันธ์อย่างสมบูรณ์นั้น ต้องประกอบด้วยคุณสมบัติหลัก 4 ประการ ได้แก่

1. Base type extension.
2. Complex objects.
3. Inheritance.
4. A rule system.

2.3.1 BASE TYPE EXTENSION

ความสามารถในการสร้างชนิดใหม่ มีเงื่อนไขอยู่หลายประการ ทั้ง ADT แบบง่ายๆ หรือซับซ้อน ซึ่งเป็นประโยชน์อย่างมาก ในการที่จะออกแบบแอปพลิเคชั่นให้ฉลาดกว่า และง่ายกว่า เช่น การขยายส่วนการเลียนแบบชนิดข้อมูลใหม่และฟังก์ชัน

ประเภทข้อมูลพื้นฐาน ต้องมีคุณสมบัติดังต่อไปนี้

2.3.1.1 Dynamic Linking

Object-relational DBMS ที่ดี ต้องเชื่อมโยงอย่างทันทีกับฟังก์ชันที่กำหนดโดยผู้ใช้ แต่สิ่งเหล่านั้นไม่ได้อยู่ใน DBMS address space จนกระทั่งเขาต้องการ ย่อหน้าต่อไปนี้อธิบายว่าทำไมความต้องการนี้ต้องนี้สำคัญ ใน object-relational DBMS ที่ดี มันต้องง่ายสำหรับผู้ใช้จะเพิ่มชนิดข้อมูลใหม่, ตัวกระทำ หรือฟังก์ชัน ต้องรื้อระบบโดยการติดตั้งชนิด หรือฟังก์ชันใหม่ หมายถึงการติดตั้งต้องถูกกำหนด บางวันก็ล่วงหน้า โดยผู้บริหารระบบส่วนกลาง ผลก็คือ การเชื่อมโยงแบบ static ไม่ค่อยสะดวกสำหรับผู้ใช้

ฟังก์ชันที่กำหนดโดยผู้ใช้ถูกขัดเกลาอีกครั้งโดยผู้พัฒนาบ่อยๆ เมื่อสิ่งนี้ปรากฏ ฟังก์ชันต้องถูกกำหนดอีกครั้งเพื่อ DBMS ถ้า DBMS ทำการเชื่อมโยงแบบ static ของฟังก์ชัน แล้ว DBMS จะต้อง down, relinked และ reinstall การที่ระบบ down และ rebuild มันสิ่งที่หนักในการจ่ายเพื่อแนวคิดนี้

การติดตั้ง Object-relational DBMS คาดหวังให้มีหมายเลขเนื้อหาของฟังก์ชันที่กำหนดโดยผู้ใช้ ถ้าเขาเชื่อมโยงแบบ static ใน DBMS space ในขณะที่ติดตั้ง DBMS แล้ว "footprint" ของ DBMS มีขนาดมหึมา มันเป็นไปได้สำหรับขนาดของ DBMS ที่ถูกเพิ่มโดยปัจจัยของ 5 หรือแม้แต่ 10 เป็นผลของการเชื่อมโยงแบบ static ของ library ขนาดใหญ่ของฟังก์ชันที่กำหนดโดยผู้ใช้ สิ่งนี้เป็นพิเศษสำหรับฟังก์ชันการจดจำรูปแบบ และเข้าใจรูปภาพ

ซึ่งมีหลากหลายและขนาดใหญ่ ในระบบปฏิบัติการส่วนมากนั้น ได้เพิ่มภาระของระบบปฏิบัติการ และการจัดการ paging ด้วยการเพิ่ม footprint และ ถ้า footprint กลายเป็นปัญหาใหญ่ บางระบบปฏิบัติการจะเกิดการ "choke"

ด้วยเหตุนี้ เหตุผลก็คือจากความสามารถของระบบปฏิบัติการ ไปยังความสะดวกของผู้ใช้ ความสะดวกของผู้ใช้ object-relational ที่ดีฟังก์ชันที่ถูกกำหนดโดยผู้ใช้ต้องมีการเชื่อมโยงแบบ dynamic

ดังนั้นต้องมีความเป็นไปได้ในการเพิ่มชนิดข้อมูลพื้นฐานใหม่ที่กำหนดโดยผู้ใช้ขณะที่กำลังทำงาน การหยุดการทำงานของฐานข้อมูลแล้ว ค่อยเพิ่มชนิดข้อมูลลงไปใหม่นั้น ยอมรับไม่ได้

2.3.1.2 Client or server activation

คุณสมบัติที่ต้องการอย่างที่ 2 คือเพื่อให้ฟังก์ชันทำงาน ข้อกำหนดอย่างชัดเจนคือกระตุ้นฟังก์ชันบนเซิฟเวอร์ใน address space ต่างๆ ของ DBMS ฟังก์ชันต้องถูกเรียกสำหรับการคัดเลือกทุกอย่างที่เป็นไปได้ของ เรคคอร์ด และผ่าน instance ของจุดและโดยรอบ ถ้าฟังก์ชันถูกกระตุ้นในกระบวนการคล้ายๆ กันของ DBMS เมื่อการกระตุ้นนี้เป็นการเรียกโดยโพรซีเดอร์ภายใน และมีค่าใช้จ่ายต่ำในการเรียก

อีกนัยหนึ่ง ถ้าฟังก์ชันวิ่งในคนละ address space บนเครื่องที่แตกต่างกัน แล้ว RPC ต้องถูกใช้ และargument ต้องถูกสำเนาไปยัง address space ที่ต่างกัน ค่าใช้จ่ายของ RPC เรียกว่า substantial และ ประสิทธิภาพจะแย่ลงอย่างมากในกรณีนี้ ตัวอย่าง ถ้าค่าใช้จ่ายของการให้ฟังก์ชันทำงาน และ/หรือ การสำเนา argument ไม่ใหญ่มากเมื่อเปรียบเดียบด้วยค่าใช้จ่ายของ RPC แล้ว RPC จะเพิ่มความสำคัญของค่าใช้จ่าย ถ้าฟังก์ชันถูกเรียกหลายครั้ง ประสิทธิภาพจะลดลง

ฟังก์ชันที่ผู้ใช้กำหนดขึ้นสำหรับชนิดข้อมูลพื้นฐานใหม่ ต้องมีความสามารถในการถูกเรียกใช้ทั้งจาก server หรือ client

2.3.1.3 Security

กับการทำงานฝั่งเซิฟเวอร์ ที่นั่นอันตรายเสมอ เมื่อผู้ใช้กำหนดฟังก์ชันขึ้นมา เขา เอาอาจไม่ได้ตั้งใจ (หรือตั้งใจ) ละเมิดความปลอดภัยโดยการอ่านหรือเขียนข้อมูลบนฐานข้อมูลกับการทำงานฝั่งเซิฟเวอร์ ที่นั่นอันตรายเสมอ เมื่อผู้ใช้กำหนดฟังก์ชันขึ้นมา เขา เอาอาจไม่ได้ตั้งใจ (หรือตั้งใจ) ละเมิดความปลอดภัยโดยการอ่านหรือเขียนข้อมูลบนฐานข้อมูล มันเลี่ยงไม่ได้ที่ระบบจะสนับสนุนการทำงานในฝั่งเซิฟเวอร์ ทำให้เกิดความเสี่ยงทางด้านความปลอดภัย

ถ้ากระตุ้นฝั่งไคลน์เอนท์เพื่อใช้งาน แล้วฟังก์ชันที่รันในกระบวนการด้วย ID ของผู้ใช้ ถ้ากระตุ้นฝั่งเซิฟเวอร์เพื่อใช้ฟังก์ชันที่รันในกระบวนการด้วย DBMS ใช้ ID ของผู้ใช้ พิจารณาฟังก์ชันที่ถูกตรวจแก้อย่างสมบูรณ์และทำคำสั่งไม่ถูกต้องตามกฎ หรือทำให้ข้ามไม่เข้าถึงแอดเดรส ถ้าฝั่งไคล์เอนท์กระตุ้นอยู่ในที่กำหนด แล้วกระบวนการไคลน์เอนท์จะชนแม้น่ารำคาญแต่เหตุการณ์ก็ถูกบรรจุ อย่างไรก็ตาม ถ้าการกระตุ้นใช้งานที่ฝั่งเซิฟเวอร์แล้วกระบวนการ DBMS จะชนกัน บางทีก็ยกเลิกผู้ใช้ทั้งหมดที่ติดต่อเข้ามายังระบบ

และถ้าไคลน์เอนท์แครช วิธีแก้ปัญหาต้องใช้รหัสของผู้ใช้ อย่างไรก็ตาม ถ้าเซิฟเวอร์แครช แล้วมันไม่ชัดเจนว่าปัญหานั้นจะค้างอยู่กับ DBMS หรือฟังก์ชันของผู้ใช้ สิ่งนี้ไม่ชัดเจนอย่างมากในกรณีที่ฟังก์ชันทำการกระโดดเข้าไปในส่วนรหัสของ DBMS การทำอย่างนี้แม้จะยากในการแยกต้นตอของปัญหา เพราะ DBMS ไม่สนับสนุนให้มีการควบคุม มันจะแครช อย่างไรก็ตามปัญหาที่แท้จริงจะอยู่กับฟังก์ชันของผู้ใช้

สิ่งที่ดูไม่ดีอย่างมาก พิจารณากรณีฟังก์ชันที่ประสงค์ร้าย กับการกระตุ้นฝั่งไคลน์เอนท์ ฟังก์ชันสามารถอ่าน (และ/หรือ ทำลาย) ทุกๆ การอ่านหรือเขียนโดยไคลน์เอนท์นี้ ด้วยตัวมันเอง มันคือปัญหาความปลอดภัย อย่างไหรก็ตาม มันไกลจากความจริงถ้ากระตุ้นการใช้งานฝั่งเซิฟเวอร์ ในกรณีนี้ ฟังก์ชันสามารถอ่านและทำลายทุกข้อมูลในฐานข้อมูล เหตุผลสำหรับทุกข้อมูลของ DBMS สามารถเขียนลงไฟล์ที่อ่านและเขียนได้ด้วยกระบวนการของ DBMS ด้วยการกระตุ้นฝั่งเซิฟเวอร์ ผู้ดูแลระบบฐานข้อมูลต้อง ไว้ใจฟังก์ชันที่ผู้ใช้กำหนดว่าไม่มีประสงค์ร้าย ในหลายๆ สภาพแวดล้อมนี้ ระดับของความไว้ใจ ไม่เหมาะสม

ฟังก์ชันที่ผู้ใช้กำหนดขึ้นสำหรับชนิดข้อมูลพื้นฐานใหม่ ต้องไม่ทำให้เกิดการทำงานที่ผิดๆ บนฐานข้อมูล

2.3.1.4 Callback

คือการยอมให้เขาวิ่ง queries ในฟังก์ชัน โดยเฉพาะโปรแกรมไคลเอนท์สามารถรวมคำสั่ง DBMS ผ่าน API ไคลเอนท์ ฟังก์ชันที่กำหนดโดยผู้ใช้ต้องสามารถใช้กับส่วนติดต่อที่เหมือนๆ กัน เพื่ออ้างถึงบริการของ DBMS วางให้แตกต่าง ฟังก์ชันต้องกระทำอย่างแน่นอนด้วยวิธีเดียวกัน ไม่ว่ามันจะในแอปพลิเคชั่นโปรแกรมหรืออ้างถึงโดย DBMS เป็นผลจากผู้ใช้ ใช้คำสั่ง SQL ที่บรรจุด้วยฟังก์ชัน

โน้ต คำสั่ง SQL อาจจะอ้างโดยฟังก์ชันที่กระทำ callback ในทางกลับกัน ภายในฟังก์ชันที่กำหนดโดยผู้ใช้ยังทำการ callback สิ่งนี้อาจหมายถึงการมีอยู่อย่างทั่วไปของ callbacks และ object-relation DBMS ที่ดี สนับสนุนสิ่งนี้โดยไม่สะสม

ฟังก์ชันที่ผู้ใช้กำหนด ต้องสามารถเรียกใช้ฟังก์ชันอื่น store procedure หรือ queries ในแอปพลิเคชั่นอื่น ที่มีลักษณะเดียวกัน .

2.3.1.5 User-defined access methods

วิธีการเข้าถึงใหม่ๆ ที่เพิ่มเข้าไปต้องเผื่อไว้สำหรับผู้กำหนดชนิดข้อมูลพื้นฐานใหม่ มันต้องมีความเป็นได้ได้ในการเพิ่มฟังก์ชันซึ่งถูกใช้โดยเครื่องมือทางฐานข้อมูล ไม่ใช่ชนิดข้อมูลทุกตัวจะเป็นชุดที่ดีสำหรับ B-trees

2.3.1.6 Arbitrary-length data-types

มันเป็นไปได้ที่จะมีชนิดข้อมูลที่กำหนดโดยผู้ใช้ ที่ไม่มีการจำกัดความยาว บางครั้งกล่าวได้ว่า BLOBs ที่แสดงในระบบเชิงสัมพันธ์เพียงพอที่จะสนับสนุนชนิดข้อมูลที่ไม่จำกัดความยาว ซึ่งไม่ถูกต้องตาม [Sto96] BLOBs ไม่ใช้ชนิดข้อมูล เพราะว่ามันไม่มี operation

2.3.2 COMPLEX OBJECTS

ความแตกต่างที่สำคัญของความสามารถในการจัดการข้อมูลที่ซับซ้อนของ OODB และ RDB ชุดของชนิดพื้นฐานใน RDBs แย่มากเมื่อเทียบกับ OODB [Sto96] ฉะนั้นกล่าวได้ว่าความสะดวกเหล่านี้มีอยู่ใน ORDB

2.3.2.1 Complex objects

มันต้องสนับสนุนชนิดซับซ้อนอย่างมากมาย อย่างน้อยที่สุดต้องมีชนิดที่ซับซ้อนดังต่อไปนี้

- Type constructors
- set of
- record of
- reference

ฟังก์ชันผู้ใช้กำหนดต้องมีความสามารถจัดการ complex types ฟังก์ชันผู้ใช้กำหนดต้องมีการสนับสนุนสำหรับการใช้งาน complex types

คล้ายๆ กับ ชนิดผู้ใช้กำหนด complex data types ต้องไม่จำกัดความยาว

มันต้องมี SQL สนับสนุน complex types เช่น dot-notation สำหรับเรคคอร์ด, [] สำหรับ array และ * สำหรับ references

2.3.3 INHERITANCE

2.3.3.1 Data and function inheritance

ลักษณะหลักของ ORDB ตาม [Sto96] คือสนับสนุน inheritance

ทั้งข้อมูลและ ฟังก์ชัน นั้นเป็นสิ่งจำเป็น inheritance ของข้อมูลนั้น ใช้กับประเภทข้อมูลเท่านั้น วิธีการสร้างตารางโดยปราศจากการใช้ชนิดข้อมูล ไม่ระบุชนิด และตารางนั้น ไม่สามารถใช้ประโยชน์จากการ inheritance

เขากล่าวว่า Inheritance ที่มีประสิทธิภาพที่สุดได้มาจากการ inheritance ของฟังก์ชัน [Sto96] เห็นว่าฟังก์ชันกำหนดโดยผู้ใช้ และ method เป็นแนวคิดอย่างเดียวกัน เขากล่าวสนับสนุนฟังก์ชัน

2.3.3.2 Overloading

มันมีความเป็นไปได้ที่จะกำหนดนิยามของฟังก์ชัน และ subtypes Overloading ต้องถูกสนับสนุน

2.3.3.3 Inheritance of types, not tables. Types and tables in different concepts

ชนิดและตารางเป็นแนวความคิดที่แตกต่างกัน ตารางเป็น container ที่ประกอบไปด้วยชนิดต่างๆ ภายใน แนวทางที่ว่านั้นก็คือสามารถมีได้หลายๆ ตาราง ในชนิดแบบใดแบบหนึ่ง แต่ละตารางมีการกำหนดการ inheritance คุณสมบัติ ถ้าตารางถูกสร้าง โดยไม่มีชื่อของชนิดอย่างนั้นแล้วตารางจะเป็นชนิด anonymous และไม่สามารถใช้ประโยชน์การ inheritance เพราะฉะนั้นเขาแนะนำให้สร้างชนิดและกำหนดให้กับตาราง มากกว่าต้องการสร้างตาราง

2.3.3.4 Multiple inheritance

ต้องสนับสนุน Multiple inheritance มีแอปพลิเคชั่นมากมายที่ต้องการ [Sto96] กล่าว เขายังแนะนำถึงปัญหาของความกำกวมในการ inheritance จาก super types ทั้งคู่ในที่นี้ ในกรณีของความกำกวม เขากล่าวว่า DBMS จะ error ขณะ runtime ดังนั้นผู้ใช้งาน DBMS ต้องรับภาระการตัดสินใจในสิ่งที่กำกวมนั้น เขายังแนะนำถึงความเป็นไปได้ในการเพิ่ม attribute ส่วนประกอบเสมือนในชนิดข้อมูล (ฟังก์ชันที่กำหนดไว้บางแห่งในลำดับชั้นของ inheritance) การใช้ attribute เสมือน ผู้ใช้สามารถอ้างถึงตำแหน่งของฟังก์ชัน ที่ inheritance มาหลากหลาย

กล่าวสั้นๆ ได้ว่า เมื่อใช้รูปแบบที่มีการทำ multiple inheritance ควรพิจารณาถ้ามันจำเป็นสิ่งหนึ่งที่สามารถใช้ได้อย่างดีคือ การใช้ส่วนประกอบ (กลุ่มรวม) ในรูปแบบถ้ามีความต้องการทำ multiple inheritance ฟังก์ชันที่ inherited จำเป็นต้องถูกกำหนดขึ้นมาก่อน เรา

สามารถเจาะจงลงไปยังภาษาต่างๆ มีวิธีแก้ปัญหา inheritance ที่แตกต่างกัน (เช่น Self, JAVA, C++, Eiffel) ซึ่งแสดงให้เห็นความเอาใจใส่ในเรื่องนี้

2.3.4 A RULE SYSTEM

Rules และ triggers ต้องมีอยู่ใน ORDB และจะต้องมีคุณลักษณะตามดังต่อไปนี้ [Sto96]:

2.3.4.1 Events and actions

Rules และ triggers เป็นประโยชน์อย่างมากในแอปพลิเคชั่นส่วนใหญ่ มันถูกใช้บ่อยๆ เพื่อทำให้แน่ใจถึงความ consistency ของฐานข้อมูล รูปแบบทั่วๆ คือ on event - do action กฎของระบบต้องรวมความสามารถของการปฏิบัติคำสั่งว่าต้องทำก่อนหรือหลังเหตุการณ์ของ process on event - do action คือ trigger ทั่วๆ ไป ตามธรรมดาแล้วนั้น trigger สนับสนุนเฉพาะ update/insert/delete query ในกรณีที่จำเป็น query ยังมีความเป็นไปได้ เพื่อระบุเหตุการณ์ให้ trigger ทำงาน

2.3.4.2 Integration of rules with inheritance and type extension

กฎของระบบควรถูกรวมเข้ากับแนวคิดเชิงวัตถุสัมพันธ์

2.3.4.3 Rich execution semantics for rules

กฎของระบบต้องมีความสามารถในการสนับสนุนการกระทำที่แตกต่างกัน เช่น immediate/deferred และ same/different transaction

2.3.4.4 No infinite loops

กลไกเชิงวัตถุสัมพันธ์ ควรสืบหาลูปใน on event - do action ตามกฎของระบบและทำให้กลับเป็นอย่างเดิมจากการทำคำสั่งตามขั้นตอนของระบบ

2.3.5 สรุป

คุณสมบัติที่ต้องการเพื่อสนับสนุน 4 ลักษณะพื้นฐานมีดังนี้

1. Base type extension
   - Dynamic linking of user-defined functions
   - Client or server activation of user-defined functions
   - Secure user-defined functions
   - Callback in user-defined functions
   - User-defined access-methods
   - Arbitrary-length data-types

2. Complex objects
   - Type constructors
     - set of
     - record of
     - reference
   - ฟังก์ชันผู้ใช้กำหนดบน complex types คล้ายกับ ฟังก์ชันผู้ใช้กำหนดบน base types:
     - dynamic linking
     - client or server activation
     - secure user defined functions
     - callback
   - Arbitrary length complex data types
   - SQL-support
3. Inheritance
   - Data and function inheritance
   - Overloading
   - Inheritance of types, not tables
   - Multiple inheritance
4. A rule system
   - Events and actions
   - Integration of rules with inheritance and type extension
   - Rich execution semantics for rules
   - No infinite loops

2.4 SQL3

SQL93 เป็นภาษาหลักที่ขยายมาตรฐานของ SQL92 ด้วยการขยายส่วนเพิ่มเติมของฟังก์ชัน ทำให้ใช้งานสะดวกใน SQL

ส่วนหลักๆ ใน SQL3 ที่อยู่ในส่วนของวัตถุ ประกอบด้วย

- ชนิดที่กำหนดโดยผู้ใช้ ซึ่งเป็น Abstract Data Types (ADTs) ได้
- มีตัวสร้างสำหรับ row type และ reference type
- มีตัวสร้างสำหรับ type ที่เป็นกลุ่ม (sets, lists and multi sets)

- สนับสนุน Binary Large Objects (BLOBs) และ Character Large Object (CLOBs)
- ผู้ใช้สร้าง Function และ Procedure ได้

2.4.1 ชนิดที่กำหนดโดยผู้ใช้ (ADTs)

โปรแกรมเมอร์ที่สร้างแอปพลิเคชั่น สามารถที่จะนิยาม Abstract Data Types (ADTs) ด้วยการซ่อน attrubute และ operations ใน entity ตัวเดี่ยวๆ operation มีลักษณะเหมือนๆ กับ procedure ซึ่งเรียกส่วนการทำงานใน SQL3 นอกจากนี้ยังสนับสนุน inheritance รวมทั้ง multiple inheritance

2.4.2 Row types และ Reference types

Row type เป็นลำดับของชื่อ field และชนิดของข้อมูลคู่กัน ซึ่งเทียบเท่ากบการสร้างตาราง กล่าวได้ว่า 2 row type จะเทียบเท่าได้ ถ้า:

- ทั้งสอง row มีหมายเลข fields เดียวกัน
- Fields ทุกๆ คู่มีตำแหน่งของชนิดที่สอดคล้องกัน

ชื่อของ row type มีไว้เพื่อให้ง่ายต่อการใช้งาน ถ้าเรากำหนดชื่อให้กับมัน นอกจากนั้นมังยังเรียงตามชนิดที่กำหนดโดยผู้ใช้ โดยไม่มีการ encapsulated โครงสร้างภายใน

ชื่อของ row types สามารถใช้ในการอ้างถึงชนิด เนื่องจากมันเป็นค่าที่ไม่ซ้ำจึงเจาะจงลงไปยัง instance ของ row type ได้

2.4.3 Collection Types

ใน SQL3 นั้น สนับสนุนไปยัง column ของตารางข้อมูลแบบเซต รายการหรือหลายๆ เซต นอกเหนือจากค่าปกติ

บทที่ 3

คุณสมบัติในระบบจัดการฐานข้อมูล

ในที่นี้ เราจะทำการค้นหาและเปรียบเทียบคุณสมบัติตามความต้องการที่กล่าวไว้เมื่อบทที่แล้ว ว่าแต่ผลิตภัณฑ์ที่เราสนใจ ซึ่งได้แก่ Oracle 8 และ Informix US มีคุณสมบัติครบถ้วนหรือไม่

3.1 INFORMIX-UNIVERSAL SERVER AS AN OBJECT-RELATIONAL DATABASE MANAGEMENT SYSTEM

เอกสารฉบับนี้อ้างถึง Informix-Universal Server (IUS) ในความสัมพันธ์ของคุณสมบัติที่ต้องการในการจัดประเภทของฐานข้อมูลเชิงวัตถุสัมพันธ์

3.1.1 A SHORT INTRODUCTION TO INFORMIX-UNIVERSAL SERVER

ตามคำกล่าวที่ให้โดย Informix, Informix-Universal server คือ ความพยายามในการขยายออกอย่างเต็มที่ของ ระบบการจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ ซึ่งได้รับการออกแบบอย่างชัดเจนเพื่อการจัดการสมบูรณ์, ชนิดข้อมูลที่ซับซ้อน

INFORMIX-Universal server เป็นผลิตภัณฑ์ที่ถูกพัฒนาโดยการรวมเอา Illustra code ที่ดีที่สุด กับ สิ่งที่ดีที่สุดของ Informix DSA (Dynamic Scalable Architecture) code

คำจำกัดความที่ให้ไว้ในเอกสารฉบับนี้อ้างถึง INFORMIX-Universal Server รุ่น 9.12 ข้อมูลเพิ่มเติมสามารถพบได้จาก http://www.informix.com

3.1.2 INFORMIX AND THE DEFINITION BY STONEBRAKER

ตามนิยามที่ให้โดย Stonebraker ฐานข้อมูลเชิงวัตถุสัมพันธ์ที่สมบูรณ์นั้นต้องสนับสนุนคุณสมบัติดังต่อไปนี้

- Base type extension
- Complex Objects
- Inheritance
- A rule system

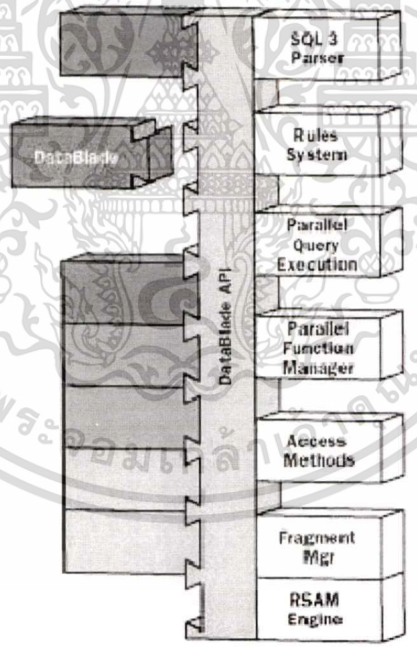
จากคุณสมบัติที่สำคัญ 4 ประการ จะอ้างอิงไปยัง Informix-Universal server

3.1.2.1 Base Type Extension

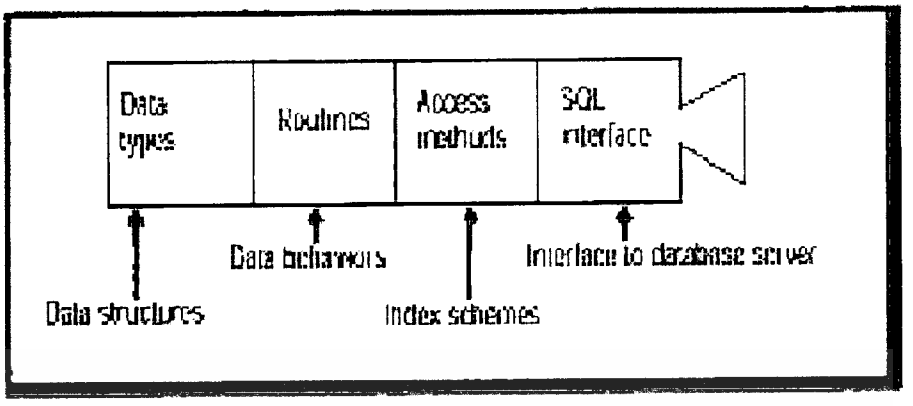
Informix-Universal Server สามารถเพิ่มการจัดการชนิดข้อมูลใหม่ โดยวิธีของ DataBlade modules

DataBlade Modules เป็นส่วนซอฟต์แวร์มาตรฐาน ซึ่งเสียบไปในฐานข้อมูลเพื่อขยายความสามารถของมัน DataBlade modules สามารถรับได้จาก Informix และจากผู้ผลติร่วมอื่นๆ หรือสามารถสร้างจาก DataBlade modules ของคุณ

DataBlade module คือชุดรวมของวัตถุฐานข้อมูล และโค้ด ซึ่งขยายเซิฟเวอร์ฐานข้อมูลโดยการเพิ่มวิธีการใหม่ไปยังเซิฟเวิอร์ฐานข้อมูล DataBlade module ทำให้เซิฟเวอร์ฐานข้อมูลเตรียมระดับขั้นการสนับสนุนชนิดข้อมูลใหม่ ซึ่งมันเตรียมรวมเข้าไว้กับชนิดของข้อมูล DataBlade module เป็นส่วนหนึ่งของ object-oriented package คล้ายๆ กับคลาสของ C++ ที่ encapsulates ความชำนาญเป็นพิเศษในชนิดข้อมูลเช่น รูปภาพ



รูปที่ 3.1 แสดงให้เห็นถึงองค์ประกอบหลักของ DataBlade module



รูปที่ 3.2 ภาพแสดงถึงองค์ประกอบหลัก 4 ส่วนของ DataBlade module

3.1.2.1.1 Data types

Data type คือชนิดข้อมูลที่ผู้ใช้กำหนดขึ้น หรือเป็นชุดรวมของข้อมูลที่ผู้ใช้กำหนดขึ้น ค่าของชนิดข้อมูลที่ผู้ใช้กำหนดขึ้น สามารถจัดเก็บ ตรวจสอบโดยใช้ queries หรือเรียกใช้โดย routine ผ่านค่า argument ไปยังฟังก์ชันของฐานข้อมูล และทำดัชนีในทำนองเดียวกันกับชนิดข้อมูลที่มีมาให้อยู่แล้ว ส่วนประกอบต่างๆ ของชนิดข้อมูลจะกำหนดโครงสร้างข้อมูลสำหรับ Universal Server

3.1.2.1.2 Routines

Routines สามารถที่จะกระทำกับชนิดข้อมูลที่กำหนดโดย DataBlade module ผู้พัฒนาจะรู้เป็นอย่างดีในชนิดข้อมูลต่างๆ ของเซิฟเวอร์ฐานข้อมูล รวมถึงชนิดข้อมูลที่กำหนดโดย DataBlade modules อื่น องค์ประกอบ routines กำหนด data behavior สำหรับ Universal Server

3.1.2.1.3 Access methods

วิธีการเข้าถึง กระทำบนตารางและดัชนี ซึ่งถูกจัดการโดยเซิฟเวอร์ฐานข้อมูล ผู้พัฒนา DataBlade module สามารถทำดัชนีบนชนิดข้อมูลใหม่ โดยใช้กรรมวิธีการเข้าถึง หรือเพิ่มวิธีการเข้าถึงใหม่ๆ ลงไป กรรมวิธีการเข้าถึงกำหนดวิธีการทำดัชนีสำหรับ Universal Server

3.1.2.1.3 SQL interface

ส่วนการติดต่อ SQL เป็นชุดรวมของฟังก์ชันที่สอดคล้องกับมาตรฐาน และส่งไปยังส่วนบริการที่คาดเอาไว้ ส่วนการติดต่อ SQL ช่วยให้ DataBlade modules ให้บริการร่วม

กันกับ DataBlade modules อื่นๆ องค์ประกอบของส่วนการติดต่อ SQL กำหนดวิธีการติดต่อกับ Universal Server

- **DYNAMIC LINKING**

ฟังก์ชันที่กำหนดโดยผู้ใช้ อาจถูกเขียนใน Stored Procedure Language (SPL) และภาษาในยุคที่ 3 เหมือนกับ C/C++ ทั้งชนิดและฟังก์ชันสามารถติดตั้งได้โดยไม่ต้องหยุดการทำงานของฐานข้อมูล ฟังก์ชัน SPL ถูกคอมไพล์เป็นส่วนหนึ่งของ p-code และถูกอินเตอร์พรีทโดยฐานข้อมูล และภาษาอื่นๆ ที่คอมไพล์ไปยัง dynamic-link module

- **CLIENT OR SERVER ACTIVATION**

ปัจจุบันทุกฟังก์ชันถูกกระทำบนเซิฟเวอร์

- **SECURITY**

สำหรับผลของประสิทธิภาพที่ Informix ได้เลือกให้ process กระทำกับฟังก์ชั่นที่กำหนดโดยผู้ใช้ ให้มีการเข้าถึงยังพื้นที่ในบัฟเฟอร์ที่ใช้ร่วมกัน ซึ่งอาจเป็นสาเหตุให้ข้อมูลเกิดข้อผิดพลาดถ้าตัวชี้เริ่มต้นไม่ถูกใช้

- **CALLBACK**

ฟังก์ชันที่กำหนดโดยผู้ใช้สามารถทำทุกอย่างได้ในการเข้าถึงฐานข้อมูล ยกเว้นเพียงอย่างเดียว ถ้าฟังก์ชันนั้นใช้คำสั่ง select ฟังก์ชันนั้นไม่สามารถแก้ไขตารางที่อยู่ในขณะการ select

- **USER-DEFINED ACCESS METHODS**

ค่าของชนิดข้อมูลที่ผู้ใช้กำหนดสามารถจัดเก็บ ตรวจสอบโดยใช้ query หรือเรียกใช้โดย routine ผ่านค่า arguments ไปยังฟังก์ชันของฐานข้อมูล และทำดัชนีในทำนองเดียวกันกับชนิดข้อมูลที่มีมาให้อยู่แล้ว

วิธีการเข้าถึงที่ปฏิบัติการบนตารางและดัชนี ถูกจัดการโดยเซิฟเวอร์ฐานข้อมูล ผู้พัฒนา DataBlade modul สามารถทำดัชนีกับชนิดข้อมูลใหม่ โดยใช้วิธีการเข้าถึงหรือเพิ่มวิธีการเข้าถึงใหม่ๆ ลงไป องค์ประกอบของการเข้าถึงกำหนดการทำดัชนีสำหรับ Universal server

- **ARBITRARY-LENGTH TYPES**

Universal Server ยอมให้บันทึก large object และ เข้าถึง และ เปลี่ยนแปลงรายละเอียด หลักจากเรียกขึ้นมาจากเซิฟเวอร์ฐานข้อมูล large object เป็นวัตถุข้อมูลที่ในทางจินตภาพแล้วเก็บลงในคอลัมน์ของตาราง แต่ในทางกายภาพแล้วการจัดเก็บเป็น

อิสระจากคอลัมน์ large object จัดเก็บแยกออกมาจากตาราง เพราะชนิดของการจัดเก็บนั้นมีขนาดใหญ่มากกว่าผลรวมของข้อมูล

Universal Server สนับสนุน large objects: smart large object และ simple large objects

- **Smart Large Object**

กับ smart large object มันเป็นไปได้ที่จะค้นหา อ่านขึ้นมา และเขียนไปยังส่วนของวัตถุ smart large object สามารถประกอบไปด้วยชนิดข้อมูลแบบ large object ดังต่อไปนี้

Character Large Object (CLOB) CLOB เป็น smart large object ที่จัดเก็บรายการข้อความ เช่น PostScript หรือแฟ้มแบบ HTML CLOB สามารถจัดเก็บและเรียกคืนได้ และมีคุณสมบัติของฐานข้อมูลเช่นการคืนสภาพ และทำ transaction rollback

Binary Large Object (BLOB) BLOB เป็น smart large object ที่สามารถจัดเก็บชนิดต่างๆ ของข้อมูลแบบไบนารี่ รวมทั้งรูปภาพ BLOB สามารถจัดเก็บ เรียกคืนได้ และมีคุณสมบัติของฐานข้อมูลเช่นการคืนสภาพ และทำ transaction rollback

Smart large object สามารถถูกใช้เพื่อเก็บชนิดข้อมูลที่ผู้ใช้กำหนด เช่น ส่วนของวีดิโอ หรือเสียง รูปภาพ ข้อความขนาดใหญ่ และ spatial object เช่นภาพวาด และแผนที่

- **Simple Large Object**

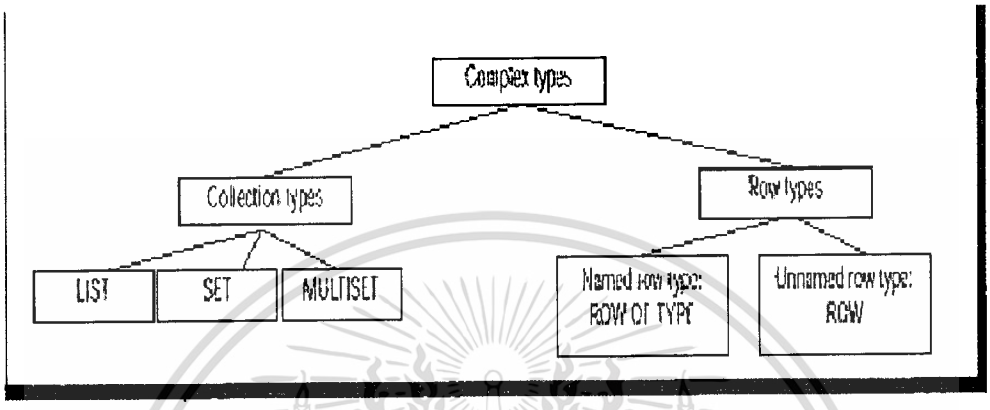
Simple Large Object คือประเภทของ large object ที่มีขนาดไม่เกิน 231 ไบต์ และในทางปฏิบัติถูกจำกัดโดยขนาดความจุของดิสก์ของเรา Universal Server สนับสนุนชนิดข้อมูลแบบ simple-large-object ดังต่อไปนี้

- BYTE จัดเก็บข้อมูลแบบ binaryy
- TEXT จัดเก็บข้อมูลแบบ text

ไม่เหมือนกับ smart large objects, simple large object ไม่สนับสนุนการเข้าถึงแบบสุ่มไปยังข้อมูล เมื่อเราถ่ายโอน simple large object ระหว่างแอปพลิเคชั่นของไคลเอน และเซิฟเวอร์ฐานข้อมูล มันต้องทำการถ่ายโอนค่าทั้งหมดของ BYTE หรือ TEXT

3.1.2.2 Complex Objects

complex object คล้ายๆ กับ complex data type ซึ่งคือชนิดข้อมูลที่ผู้ใช้กำหนดสามารถมีชนิดข้อมูลได้หลายชนิด ของข้อมูลใดๆ และชนิดที่รวมกัน



รูปที่ 3.3 แสดง Complex types ที่ Informix Universal server สนับสนุน

3.1.2.2.1 ARRAYS

Informix-Universal Server ไม่สนับสนุน functionality ทั้งหมด ที่เป็น constructor ของชนิดอาร์เรย์แบบปกติ โดยตัวอย่าง มันเป็นไปไม่ได้ที่จะเพิ่มสมาชิกเข้าไป หรือ select สมาชิกจากตำแหน่งของอาร์เรย์ที่ระบุ

ส่วนของ functionality ที่หวัง ซึ่งสนับสนุนใน Informix-Universal server นั้น สนับสนุนโดยชุดของข้อมูล "List"

รายการ คือชุดรวมของสมาชิกซึ่งยอมให้เกิดค่าซ้ำ สมาชิกแต่ละตัวในรายการมีลำดับในชุดรวม คำสั่งของสมาชิกในรายการโต้ตอบกับคำสั่งซึ่งถูกใส่ลงไปใน LIST

ตัวอย่างต่อไปนี้สร้างตารางซึ่งคอลัมน์ MONTH_SALES เป็น LIST อันดับแรกบันทึก (สมาชิก)ลงใน LIST ในตำแหน่งที่ 1 อาจจะเหมือนกับเดือนมกราคม สมาชิกลำดับที่สอง อยู่ในตำแหน่งที่ 2 เป็นกุมภาพันธ์ ดังนี้

```
CREATE TABLE sales_person
(
    name CHAR(30),
    month_sales LIST(MONEY NOT NULL)
);
```

3.1.2.2.2 COMPOSITES

INFORMIX-Universal server ประกอบด้วย 2 ส่วน ได้แก่ named และ unnamed เรียกว่า "Named row type" และ "Unnamed row type"

- Named Row Type

named row type คือหนทางหนึ่งในการสร้างชนิดข้อมูลใหม่ named row type สามารถกำหนดตารางหรือคอลัมน์

ตัวอย่างต่อไปนี้แสดงวิธีการสร้างชนิดของตารางโดยกำหนด named row type ในคำสั่ง create

```
CREATE ROW TYPE person_t
(
    name VARCHAR(30),
    address VARCHAR(20),
    city VARCHAR(20),
    state CHAR(2),
    zip INTEGER,
    bdate DATE
);
CREATE TABLE person OF TYPE person_t;
```

คำสั่งแรกสร้างชนิดข้อมูลใหม่คือ PERSON_T คำสั่งถัดมาสร้างตาราง person พร้อมด้วยสมาชิกของชนิด PERSON_T

ในตัวอย่างถัดไป ชนิดข้อมูลใหม่สร้างโดย named row type และชนิดดังกล่าวถูกใช้ในการนิยามคอลัมน์ในตาราง employee

```
CREATE ROW TYPE address_t
(
    street VARCHAR(20),
    city VARCHAR(20),
```

```
    state CHAR(2),
    zip VARCHAR(9)
);

CREATE TABLE employee
(
    name VARCHAR(30),
    address address_t,
    salary INTEGER
);
```

- Unnamed Row type

จุดเด่นที่สำคัญระหว่าง named และ unnamed row types คือมันเป็นไปไม่ได้ที่จะกำหนด unnamed row type ให้กับตาราง unnamed row type ถูกใช้ในการกำหนดชนิดให้กับคอลัมน์ หรือ field เท่านั้น นอกจากนี้ unnamed row type ถูกระบุโดยโครงสร้างเพียงอย่างเดียว อย่างไรก็ตาม named row type จะถูกระบุด้วยชื่อของมัน คำสั่งต่อไปนี้กำหนด 2 unnamed row types ให้กับคอมลัมน์ของตาราง student

```
CREATE TABLE student
(
    s_name ROW(f_name VARCHAR(20), m_init CHAR(1),
    l_name VARCHAR(20) NOT NULL),
    s_address ROW(street VARCHAR(20), city VARCHAR(20),
    state CHAR(2), zip VARCHAR(9))
);
```

คอลัมน์ S_NAME และ S_ADDRESS ของตาราง student จะมีหลาย fields แต่ละ field ของ unnamed row type สามารถมีชนิดข้อมูลที่แตกต่าง ถึงแม้ว่าตาราง student มี 2 คอลัมน์ unnamed row types กำหนด fields ทั้งหมด 7 fields: F_NAME, M_INIT, L_NAME, STREET, CITY, STATE และ ZIP

3.1.2.2.3 SETS

INFORMIX-Universal server สนับสนุน sets 2 ชนิด คือ ordinary set และ multiset

- Set

set คือชุดรวมที่ไม่มีลำดับของสมาชิก ซึ่งสมาชิกแต่ละตัวต้องมีเพียงอันเดียว ตัวอย่างต่อไปนี้แสดงให้เห็นถึงวิธีการใช้ SET

```
CREATE TABLE employee
(
    name CHAR(30),
    address CHAR (40),
    salary INTEGER,
    dependents SET(VARCHAR(30) NOT NULL)
);
```

คำสั่งด้านบนเป็นการสร้างตารางซึ่งพนักงานแต่ละคนจะขึ้นอยู่กับบริษัท ที่ได้กำหนดไว้เป็น SET

- Multiset

Multiset เป็นชุดรวมของสมาชิก ซึ่งสมาชิกสามารถซ้ำกันได้ ตัวอย่างต่อไปนี้แสดงถึงวิธีการใช้งาน MULTISET

```
CREATE TABLE employee
(
    name CHAR(30),
    address CHAR (40),
    salary INTEGER,
    bonus MULTISET(MONEY NOT NULL)
);
```

คำสั่งสร้างตารางซึ่งคอลัมน์โบนัสเป็น MULTISET ในคอลัมน์ของโบนัส โบนัสซึ่งลูกจ้างรับจากค่าล่วงเวลาสามารถจัดเก็บได้

3.1.2.2.4 REFERENCES

References ในเวอร์ชันปัจจุบันของ IUS ยังไม่สนับสนุน

3.1.2.2.5 SQL SUPPORT OF COMPLEX DATA TYPES

- Row type

การเข้าถึง field เดี่ยวๆ ใน row types นั้นถูกสนับสนุนโดย dot notation สามารถใช้ไม่ว่าจะเป็นชนิด named tow type หรือ unnammed row type

ตัวอย่างดังต่อไปนี้แสดงถึงวิธีการใช้ dot notation

```
CREATE ROW TYPE address_t
(
    street VARCHAR(20),
    city VARCHAR(20),
    state CHAR(2),
);

CREATE ROW TYPE employee_t
(
    name VARCHAR(30),
    address address_t
    salary INTEGER
);

CREATE TABLE employee OF TYPE employee_t;
SELECT address.city, address.state
FROM employee
```

คำสั่ง SELECT รวม filed city และ state จากค่าที่ส่งกลับมาของคอลัมน์ address

- Collection Type

หนทางเดียวในการ select, insert, update หรือลบสมาชิกตัวใดตัวหนึ่งในชุดรวมคือผ่านมาจากภายนอก หรือชุดคำสั่ง SPL นอกจากนี้ คุณไม่สามารถกระทำ sub-queries บนคอลัมน์ที่เป็นชุดรวมของชนิด

ชุดรวมของสมาชิกสามารเลือกโดยใช้คำหลัก IN ในคำหลัก IN สามารถรวมเข้ากับ NOT เพื่อค้นหาชุดนั้นต้องไม่บรรจุสมาชิกที่แน่นอน

3.1.2.3 Inheritance

Universal Server สนับสนุน inheritance เฉพาะ named row types และ typed tables

Universal Server สนับสนุนเฉพาะ single inheritance กับ single inheritance แต่ละ subtype หรือ sub-table มีเฉพาะ 1 super-type หรือ super-table

ความหมายก็คือ INFORMIX-Universal server ไม่สนับสนุน multiple inheritance

3.1.2.3.1 TYPE INHERITANCE

ชนิดแบบ inheritance ประยุกต์กับ named row types เท่านั้น คุณสามารถใช้ inheritance กับกลุ่มของ named row types ไปยังชนิดลำดับชั้น ซึ่งแต่ละ subtype inherits จะแสดงข้อมูล (data fields) และ behavior (routines, aggregates และ operators) ของ supertype ภายใต้การถูกกำหนดของมัน

ตัวอย่างต่อไปนี้สร้าง PERSON_T super-type ของชนิดลำดับชั้น และsubtype EMPLOYEE_T นั้น inherits fields ทั้งหมดของ PERSON_T

```
CREATE ROW TYPE person_t
(
    name VARCHAR(30) NOT NULL,
    address VARCHAR(20),
    city VARCHAR(20),
    state CHAR(2),
    zip INTEGER,
    bdate DATE
);
```

```
CREATE ROW TYPE employee_t
(
    salary INTEGER,
    manager VARCHAR(30)
)
UNDER person_t;
```

มันมีความเป็นไปได้ที่กำหนดหลายๆ subtypes ภายใต้ super-type เดียว ถึงแม้ว่า inheritance เดียว ต้องการสืบทอดทุกๆ subtype จากหนึ่ง และ super-type เดียวเท่านั้น ไม่มีข้อจำกัดในการปฏิบัติบนความลึกและความกว้างของชนิดลำดับชั้น

3.1.2.3.2 TYPED TABLE INHERITANCE

ทุกๆ ตารางในตารางลำดับชั้นต้องถูกกำหนดเพื่อ named row type ในชนิดลำดับชั้นที่คล้ายคลึงกัน เมื่อคุณสร้าง sub-table ภายใต้ super-table, sub-table มีการสืบทอดทุกคุณสมบัติของ super-table รวมถึงสิ่งต่อไปนี้

- All columns of the super-table
- Constraint definitions
- Storage options
- Indexes
- Referential integrity
- Triggers
- The access method

คำสั่ง CREATE TABLE ต่อไปนี้ กำหนดลำดับชั้นตารางอย่างง่ายๆ

```
CREATE TABLE person OF TYPE person_t;

CREATE TABLE employee OF TYPE employee_t
UNDER person;

CREATE TABLE sales_rep OF TYPE sales_rep_t
UNDER employee;
```

3.1.2.3.3 OVERLOAD OF FUNCTIONS

Routines สามารถ overloaded พื้นฐานบนชนิดและลำดับของพารามิเตอร์ มันเป็นไปไม่ได้ที่จะ overload ชนิดที่ส่งกลับ

เมื่ออ้างถึง routine เซิฟเวอร์ฐานข้อมูลค้นหา signature ที่ตรงกับชื่อของ routine และ arguments ถ้า routine กับ signature ตรงกันมันจะทำงาน ถ้าไม่มีคู่ที่แน่นอน เซิฟเวอร์ฐานข้อมูลจะค้นหา routine ตัวแทน โดยจับคู่ arguments จากซ้ายไปขวา

3.1.2.4 Rules

IUS สนับสนุน rules โดยใช้ triggers

3.1.2.4.1 UPDATE AND QUERY RULES

Triggers สามารถถูกกระต้นโดย INSERT, DELETE หรือ UPDATES. trigger ไม่สนับสนุน Select ดังนั้นกฎในการ update จะมี แต่ query ไม่มี

3.1.2.4.2 RULES IN SAME OR DIFFERENT TRANSACTION

ทุกกฎถูกจัดการใน active transaction ถ้า trigger ต้องการที่จะจัดเก็บข้อมูล แม้ว่า transaction นั้น roll back มันต้องหาวิธีที่จะทำ โดยที่ DB ไม่รู้ เช่น การเขียนลงไฟล์

3.1.2.4.3 RULES INTEGRATION WITH OTHER OBJECT-RELATIONAL FEATURES

ไม่มีข้อจำกัดในวิธีการการทำงานที่ใกล้เคียงของ trigger สิ่งหนึ่งที่กล่าวถึงคือเรื่องของการ callback

IUS มีการจำกัดจำนวนในการใช้งาน trigger โดยมีเพียงหนึ่ง INSERT และ DELETE trigger ที่สามารถกำหนดได้ในแต่ละตาราง และถ้า DELETE กลุ่มของตารางที่ลดหลั่นกันลงมา DELETE trigger ที่จะถูกสร้างขึ้น การทำ Multiple UPDATE trigger ถูกยอมได้ถ้าใช้กับคอลัมน์ที่แตกต่างกัน

Trigger อาจจะถูกสร้างจาก procedure ตราบเท่าที่ procedure ไม่สามารถถูกเรียกมาจากคำสั่งแก้ไขตาราง

3.1.2.4.4 IMMEDIATE OR DEFERRED EXECUTION OF RULES

แต่ละ trigger การทำงานสามารถกำหนดการกระทำ BEFORE, AFTER และ สำหรับ EACH ROW ของการกระตุ้นคำสั่ง

3.1.2.4.5 DETECTION OF LOOPS

Loops ใน trigger ต้องถูกค้นหา แต่ต้องไม่มากไปกว่า 16 ระดับของ trigger ถึงจะยินยอม Loops ในฟังก์ชันที่กำหนดโดยผู้ใช้ จะสืบหาได้ถ้าเขาใช้ stack-space (เรียกใช้แบบ recursive) ฟังก์ชันจะถูกใช้ตราบเท่าที่เขาต้องการ DB ไม่สามารถค้นหา loops

3.1.3 การเปรียบเทียบ INFORMIX-UNIVERSAL SERVER กับข้อกำหนดของ STONEBRAKERS ORDBMS

ข้อกำหนดของ Stonebrakers		Informix
Base type extension	Dynamic linking	Yes
	Client and server execution	Yes
	Security modules	No
	Callback	Yes
	User defined access methods	Yes
	Arbitrary length types	Yes
Complex types	Array	No - only lists
	Composites	Yes
	Set	Yes - single and multiple
	Structure	
	References	No
	SQL support	Yes (nothing for arrays)
Inheritance	Data and Function inheritance	Yes
	Overload of functions and methods	Yes
	Types and table is separate concepts	Yes
	Multiple inheritance	No
Rules	Query-rules	No
	Rules on queries as well as changes in the database	
	Events	No

ข้อกำหนดของ Stonebrakers		Informix
	Restriction in rules implementation	No but restriction in rule definition
	Rules in separate transactions	
	No limitations on rule functionality	
	Immediate/deferred rules	Yes
	Same or different transaction	No
	Loop detection	Yes-some

ตารางที่ 3.1 การเปรียบเทียบ INFORMIX-UNIVERSAL SERVER กับข้อกำหนดของ STONEBRAKERS ORDBMS

3.1.4 OTHER ASPECT OF INFORMIX-UNIVERSAL SERVER

3.1.4.1 Overload of Operators

ตัวกระทำสามารถ overloaded บนพื้นฐานของชนิดที่ถูกกระทำ ตัวกระทำ overload เป็นสาเหตุให้ตัวกระทำไปกระตุ้นฟังก์ชันที่กำหนดโดยผู้ใช้

3.2 ORACLE8 AS AN OBJECT-RELATIONAL DATABASE MANAGEMENT SYSTEM

ในส่วนของรายงานฉบับนี้เพื่อดูว่า ผลิตภัณฑ์ทางด้านฐานข้อมูล Oracle 8 มีเงื่อนไขเตรียมสู่ฐานข้อมูลเชิงวัตถุ

3.2.1 A SHORT INTRODUCTION TO ORACLE8

Oracle 8 เป็นรุ่นล่าสุดของระบบจัดการฐานข้อมูลจากบริษัท Oracle

บริษัท Oracle เป็นผู้ผลิตรายใหญ่บนตลาด DBMS ที่ไม่ใช่ระบบ mainframe และเป็นการตรงประเด็นอย่างมากในการรวม Oracle 8 ไว้ในเอกสารฉบับนี้

เปรียบเทียบกับ Oracle 8 ในรุ่นที่ผ่านมาได้เพิ่มคุณสมบัติทางวัตถุจำนวนหนึ่ง และวัตถุประสงค์ของบทนี้เพื่อตรวจสอบดูในรายละเอียด

รุ่นของ Oracle 8 ที่ใช้ในบทนี้คือรุ่น 8.0.3 ดังนั้นเนื้อหาอาจไม่สมบูรณ์เมื่อดูในรุ่นถัดๆ มา

สำหรับข้อมูลเพิ่มเติมของบริษัท Oracle และ Oracle 8 พบได้จาก http://www.oracle.com

3.2.2 ORACLE8 AND STONEBRAKERS DEFINITION

- User defined types
- Handling complex objects
- Inheritance
- Rule handling

Stonebraker ได้กำหนดเงื่อนไขจำนวนหนึ่งที่ฐานข้อมูลต้องมีก่อนที่จะกล่าวว่าเป็นเชิงวัตถุสัมพันธ์อย่างสมบูรณ์ (สำหรับรายละเอียดโปรดดูได้จากบทที่ผ่านมา)

ในส่วนต่อไปนี้เราจะสำรวจว่า Oracle 8 พบความต้องการเหล่านี้อย่างไร

3.2.2.1 User Defined Types

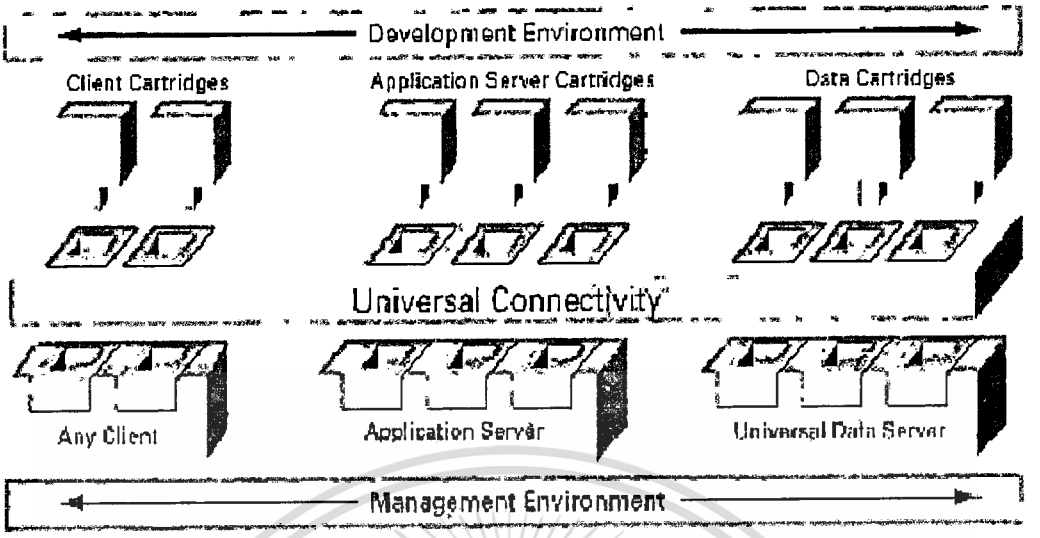
Oracle 8 มีความเป็นไปได้ในการกำหนด Cartridges

Cartridges สามารถกล่าวได้ว่าเป็นการเรียงกันของ plug-in modules ซึ่งสามารถถูกใช้เพื่อเพิ่มการทำงานให้กับ Oracle 8 ท่ามกล่างสิ่งต่างๆ Cartridges ข้อมูล สามารถบรรจุชนิดใหม่และวิธีการทำงานที่คล้ายคลึงกัน

Cartridges สามารถ (คล้ายกับตัว kernel ของฐานข้อมูลเอง) เรียกใช้ชุดคำสั่งภายนอก (ใช้ร่วมกัน) บนระบบปฏิบัติการซึ่งเป็นไปได้ที่จะใช้ชุดคำสั่งร่วมกัน

ทางเลือกของ Oracle 8 นั่นคือ Cartridges ข้อมูล สำหรับการประมวลผลภาพ การจัดการวีดิโอ และสำหรับค้นหาข้อความ

ในสถาปัตยกรรมของ Oracle 8 Cartridges สามารถติดตั้งลงบน client (เฉพาะส่วนติดต่อผู้ใช้ของ cartridges เท่านั้น) แอปพลิเคชั่นเซิฟเวอร์ (เงื่อนไขทางธุรกิจไม่ใช่ความสามารถของชนิด) และ เซิฟเวอร์ฐานข้อมูล (เงื่อนไขทางธุรกิจ และความสามารถของชนิด ด้วยการทำงานที่คล้ายคลึงกัน)



รูปที่ 3.4 แสดงระบบ Cartridges

3.2.2.1.1 DYNAMIC LINKING

Cartridges สามารถเพิ่มไปยังฐานข้อมูลในรูปแบบ dynamic ทำให้ไม่ต้องปิดระบบฐานข้อมูลในการเพิ่ม cartridge

3.2.2.1.2 CLIENT OR SERVER EXECUTION

แนวความคิดของ Cartridge ใน Oracle 8 ไม่ได้อยู่ในตัวมัน แต่ให้โอกาสในการเปลี่ยนวิธีการสำหรับการกระทำที่กำหนดโดยผู้ใช้ ถ้าชนิดถูกประกาศร่วมกับวิธีการของมัน (เรียกว่า method ใน Oracle 8) แล้วการกระทำคือการปฏิบัติการบนเซิฟเวอร์

อย่างไรก็ตามไม่มีสิ่งใดที่จะป้องกันผู้พัฒนาในการทำ 2 cartridges ในตัวเดียวกัน cartridges ตัวหนึ่ง สามารถบรรจุชนิดที่ประกาศขึ้น และวิธีการที่จะทำงานบนเซิฟเวอร์ฐานข้อมูล และวิธีการอื่นๆ ซึ่งสามารถถูกเรียกใช้บนเซิฟเวอร์แอปพลิเคชั่นหรือเซิฟเวอร์ฐานข้อมูล

ถึงแม้ว่าการแบ่งส่วนมีความเป็นไปได้ มันนำไปสู่เชิงวัตถุหนทางเชิงวัตถุทั้งหมด และไม่มีการแยกส่วนบรรจุข้อมูลและวิธีการ อย่างไหรก็ตามอย่างไรก็ตามถึงแม้ว่าการแยกส่วนไม่ได้ถูกทำ แต่ข้อเท็จจริงก็ยังคงอยู่ สำหรับระบบเชิงวัตถุวิธีการเข้ากระทำบนไคลเอ็นท์ไม่ใช่บนเซิฟเวอร์

ปัญหาอื่นในส่วนพิจารณานี้คือถึงแม้ว่าคำสั่งการทำงานนี้อยู่บนเซิฟเวอร์ มันอาจจะไม่มีอะไรที่ Stonebraker เรียกว่าเป็นการทำงานบนเซิฟเวอร์ นี่ก็เป็นเพราะการนำเสนอด้านความปลอดภัย ที่ซึ่งคำสั่งการทำงานไม่ปลดภัยในการเข้าทำงานในหน่วยความจำ

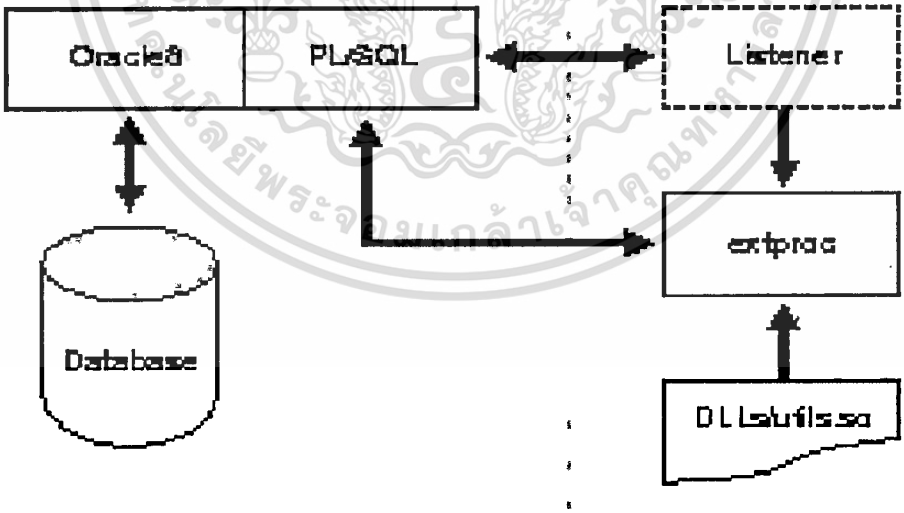
แบบแยกส่วน ในส่วนแก่นของฐานข้อมูล (ดูรูป 6.2.1.3) ชนิดของการทำงานบางที่ไม่มีอะไรที่อยู่ในหมวดหมู่ของ Stonebraker ในการเข้าทำงานกับเซิฟเวอร์

การทำงานบนเซิฟเวอร์ที่แท้จริง ความเป็นไปได้เท่านั้นคือเขียนการทำงาน cartridge ใน PL/SQL ในการโปรแกรมภาษาของ Oracle 8

อย่างไรก็ตามไม่มีความชัดเจนถึงวิธีการทำตามแนวความคิดของ Stonebraker มันเป็นทางเลือกอิสระของการทำงานให้สามารถแสดงให้เห็นผลได้ สำหรับบทสรุปเบื้องต้นในตอนนี้คือ Oracle 8 ในความเห็นของ Stonebraker คือ สิ่งนี้มีความเป็นมิตรในการแปลความหมาย

3.2.2.1.3 SECURITY

การทำงานของ Cartridge เขียนขึ้นในภาษาที่อาจมีปัญหาทางด้านความปลอดภัย (ภาษาในยุคที่ 4 บางภาษา, ภาษาในยุคที่ 3 ส่วนใหญ่ และโค้ดในแอสแซมบลี) ต้องถูกแทนที่ในชุดคำสั่งที่ใช้ร่วมกัน และถูกเรียกผ่านกระบวนการที่แยกออกมา นี่ต้องแน่ใจว่าแก่นของฐานข้อมูลเองต้องไม่ถูกทำให้ผิดโดยข้อผิดพลาดจากนักเขียนโปรแกรมในโค้ดของ cartridge



รูปที่ 3.5 แสดงกระบวนการของการเรียกใช้การทำงานจากภายนอก

3.2.2.1.4 CALLBACK

ทุกวิธีการทำงานร่วมกัน มีความเป็นไปได้ในการทำการเรียกกลับของฐานข้อมูล ทั้งวิธีการทำงานแบบ safe cartridge และ unsafe cartridge ที่อยู่ในชุดคำสั่งทำงานร่วมกัน

3.2.2.1.5 USER DEFINED ACCESS METHODS (INDEXES)

Oracle 8 ให้โอกาสในการสร้างดัชนีบนหลายๆ attribute รวมกัน ให้สำหรับชนิดอย่างไรก็ตามตราบเท่าที่ชนิดเป็นหนึ่งในชนิดมาตรฐานของ Oracle 8 นั่นคือ ไม่ใช่กับชนิดที่กำหนดโดยผู้ใช้ หรือชนิดที่ซับซ้อน

เป็นไปไม่ได้ที่จะทำดัชนีตารางที่ซ้อนกัน หรืออาร์เรย์

นี่ไม่เป็นไปตามความต้องการของ Stonebraker ความเป็นไปได้เพื่อสร้างโครงสร้างการเข้าถึงที่กำหนดโดยผู้ใช้ หรือสร้างดัชนีพื้นฐานบนค่าฟังก์ชัน (เช่น เพื่อสร้างดัชนีพื้นฐานการปรับเปลี่ยนเป็นพิมพ์ใหญ่กับ attribute แบบตัวอักษร) Oracle อ้างถึงสิ่งนี้ อาจมีความเป็นไปได้จากรุ่น 8.1 ของ Oracle 8 (ระหว่าง 1998)

สรุปก็คือ Oracle 8 ไม่มีในสิ่งที่ Stonebraker ต้องการในกรณีนี้

3.2.2.1.6 INFINITELY LARGE TYPES

Oracle ไม่กล่าวถึงข้อจำกัดสำหรับชนิดที่กำหนดโดยผู้ใช้ในเอกสารอ้างอิงทางเทคนิค

อย่างไรก็ตามมีขนาดที่จำกัดของ large binary objects (4 GB) และมันต้องกำหนดขนาดที่จำกัดนี้ประยุกต์สู่ชนิดที่กำหนดโดยผู้ใช้(โดยเฉพาะในระบบ 32 บิท)

3.2.2.2 Complex Objects

ความต้องการน้อยที่สุดของ Stonebraker นั่นคือชนิดที่ซับซ้อนต้องถูกสนับสนุนในฐานข้อมูลเชิงวัตถุสัมพันธ์ (ในศัพท์ของ Oracle 8 ชนิดต้องไม่แยกจากชนิดที่กำหนดโดยผู้ใช้ เราได้มีการปรึกษากันในส่วนที่แล้ว)

3.2.2.2.1 ARRAYS

Oracle 8 สนับสนุน อาร์เรย์ ใน Oracle 8 คือ VARRAYs (มาจาก VARying ARRay)

ตัวอย่างของการประกาศอาร์เรย์ของ Oracle 8 นั่นคือ:

```
CREATE TYPE MY_BANANAS as VARYING ARRAY (1000) OF BANANAS;
```

นี่เป็นการสร้างชนิด (MY_BANANAS) ซึ่งประกอบด้วยข้อมูลถึง 1000 BANANAS (อย่างการประกาศที่ผ่านมา)

ไม่มีข้อจำกัดบนชนิดของสมาชิกในอาร์เรย์

3.2.2.2.2 COMPOSITES

Oracle 8 สนับสนุน Composites ใน Oracle 8 เรียกว่า OBJECT

ตัวอย่างก็คือ:

```
CREATE TYPE BANANAS AS OBJECT
(   weight NUMBER(10,3),
    ripeness RIPENESS_FACTOR,
    MEMBER FUNCTION get_eaten RETURN BOOLEAN
)
```

แต่ละ attribute สามารถมีชนิดต่างๆ (RIPENESS_FACTOR เข้าใจว่าเป็นชนิดที่ประกาศซ้ำ)

3.2.2.2.3 SETS

Oracle 8 สนับสนุน Sets ใน Oracle 8 เรียกว่า TABLE

ตัวอย่างของ VARRAY ดูคล้ายกับสิ่งนี้ถ้าชนิด TABLE ถูกใช้เป็นตัวแทนของ VARRAY

```
CREATE TYPE MY_BANANAS as TABLE OF BANANAS;
```

3.2.2.2.4 REFERENCES

Oracle 8 สนับสนุน References ใน Oracle 8 เรียกว่า REF

Reference ใน Oracle 8 ไม่มีตัวชี้ไปยัง actual object อย่างใดอย่างหนึ่งเพราะว่า reference ยังไม่มีในตอนเริ่มแรก หรือเพราะว่าวัตถุซึ่งถูก reference นั้นถูกลบ

Reference คือ DANGLING ที่ State สามารถทดสอบก่อนมีการใช้ reference

ตัวอย่างที่มีการใช้ reference คือ

```
CREATE TYPE WORKING_PERSON_TYPE AS OBJECT
(name VARCHAR2(80), EMPLOYER REF EMPLOYER_TYPE);
```

3.2.2.2.5 SQL SUPPORT FOR COMPLEX TYPES

Oracle 8 สนับสนุนชนิดทั้งหมดข้างต้นโดยตรงใน SQL โดยการใช้ dot-notation สำหรับ composites และ references และ [] สำหรับ arrays

Oracle รับประกันว่า Oracle 8 สอดรับกับมาตรฐานที่เพิ่มขึ้นใน SQL 3 สำหรับฐานข้อมูลเชิงวัตถุสัมพันธ์ แต่ตั้งแต่มีมาตรฐาน SQL 3 มันยังไม่สิ้นสุด สิ่งนี้เป็นไปได้ยากที่จะทดสอบ

Inheritance ระหว่างชนิดไม่ถูกสนับสนุนในรุ่นปัจจุบันของ Oracle 8 แต่ Oracle สัญญาว่ามันจะมีส่วนดังกล่าวในรุ่นถัดไป (บางทีอาจเป็นรุ่น 8.1 ในปี 1998)

นั่นจะสนับสนุนสำหรับ single inheritance เท่านั้น และไม่มีแผนที่จะสนับสนุน multiple inheritance

3.2.2.3 Inheritance

พิจารณา Oracle 8 จากมุมมองด้านวัตถุ การไม่มีชนิดใดๆ ของ inheritance เป็นความสูญเสียหลัก แต่ Oracles เข้าใจว่า สิ่งนี้ไม่ใช่คุณสมบัติจำเป็นในฐานข้อมูลเชิงวัตถุสัมพันธ์

นี่เป็นการเริ่มต้นของความแตกต่างในความต้องการของ Stonebraker ที่ฐานข้อมูลเชิงวัตถุสัมพันธ์ต้องสนับสนุน ไม่เพียงแต่เฉพาะ inheritance แต่ยังรวมถึง multiple inheritance แม้ว่า multiple inheritance เป็นการค้านของเชิงวัตถุ มันยังไม่มีคำถามเกี่ยวกับ inheritance ที่เป็นประโยชน์และรูปลักษณ์ที่สำคัญ ของการพัฒนาทางด้านเชิงวัตถุ (และเชิงวัตถุสัมพันธ์)

3.2.2.3.1 FUNCTION AND METHOD OVERLOAD

มีความเป็นไปได้ในการทำ overload function ใน Oracle 8 (มันมีอยู่ใน Oracle 7) โดย overloading จำนวนและชนิดของพารามิเตอร์ มันเป็นไปไม่ได้ที่จะ overload ค่าที่ส่งกลับเท่านั้น

3.2.2.4 Rule Handling

กฎเรียกว่า triggers ใน Oracle 8 (อย่างน้อยกฎก็ทำบางอย่างนอกจากความมั่นใจในมาตรฐานของความคงสภาพข้อมูล คล้ายๆ กับ แม่-ลูก, ยอมให้มี null เป็นต้น)

- RULES ON QUERIES AS WELL AS CHANGES

Oracle 8 ไม่สนับสนุน trigger บน queries ในฐานข้อมูล การเปลี่ยนเดียวที่สามารถเกิดขึ้นได้เมื่อ triggers ถูกเรียกใช้งาน

- RULES IN SEPARATE TRANSACTIONS

  Oracle ไม่สนับสนุนกฎใน transaction แยกส่วน

- NO LIMITATIONS ON RULE FUNCTIONALITY

  โดยทั่วไป Oracle 8 ยอมให้ทุกชนิดของการทำงานใน triggers (inserts, updates, และการเรียกใช้คำสั่งทำงาน) บางการปฏิบัติการ สามารถใช้ลูป และตรงนั้นจะถูกป้องกันจากการเข้าทำงาน

- IMMEDIATE/DEFERRED EXECUTION OF RULES

  Oracle 8 ให้ความจำกัดที่เป็นไปได้ในการเลือกว่าเมื่อไร trigger จะทำงาน แต่เพียงเท่านั้นที่ triggers จะทำงานก่อนหรือหลัง statement

- LOOP DETECTION

  Oracle 8 ค้นพบลูปในการทำงานของ triggers แต่ได้เฉพาะในระดับของตารางเท่านั้น สิ่งนี้หมายถึง Oracle 8 ได้ยกเลิก trigger แม้สิ่งนั้นไม่ได้เป็นต้นเหตุของลูปและสิ่งนี้จำกัดการทำงานบางส่วน

ตั้งแต่นั้นมานี่เป็นปัญหาที่แก้ไม่ตก (เปรียบได้จากการยุติปัญหาของการเพิ่มสมรถนะของระบบ) มันยากที่จะโทษ Oracle สำหรับเรื่องนี้

3.2.3 CONCLUSION ON ORACLE8 AND STONEBRAKERS DEFINITION

ข้อกำหนดของ Stonebrakers		Oracle
Base type extension	Dynamic linking	Yes
	Client and server execution	Can be programmed
	Security modules	Yes
	Callback	Yes
	User defined access methods	No
	Arbitrary length types	Yes
Complex types	Array	Yes
	Composites	

ข้อกำหนดของ Stonebrakers		Oracle
	Set	Yes
	Structure	Yes
	References	Yes
	SQL support	Yes - SQL3 compliant.
Inheritance	Data and Function inheritance	No
	Overload of functions and methods	Yes
	Types and table is separate concepts	
	Multiple inheritance	
Rules	Query-rules	
	Rules on queries as well as changes in the database	No - Changes only
	Events	
	Restriction in rules implementation	
	Rules in separate transactions	No
	No limitations on rule functionality	Yes
	Immediate/deferred rules	Partially - before/after insert but not deferred until commit
	Same or different transaction	
	Loop detection	Yes-but somewhat restrictive

ตารางที่ 3.2 CONCLUSION ON ORACLE8 AND STONEBRAKERS DEFINITION

3.2.4 FURTHER OBJECT-RELATIONAL ASPECTS IN ORACLE8

ข้อเสนอ Oracle 8 อย่างน้อยประการหนึ่ง คือการเพิ่มความสนใจโดยไม่ถูกครอบคลุมโดยความต้องการของ Stonebrakers นั่นคือแนวคิดมุมมองเชิงวัตถุ

3.2.4.1 Object Views

ใน Oracle 8 มีการออกแบบกลไกเฉพาะบางอย่างเพื่อให้มันมีความเป็นไปได้ (หรือง่ายกว่า) เพื่อยกหรือเปลี่ยนจากฐานข้อมูลเชิงสัมพันธ์อย่างเดียว ไปยังเชิงวัตถุสัมพันธ์ หรือรูปแบบเชิงวัตถุจริงๆ สิ่งนี้จะกลับจะทำให้มันง่ายในการใช้เครื่องมือเชิงวัตถุในการพัฒนาคำสั่งการทำงานใหม่ๆ

กลไกดังกล่าวเรียกว่า Object views และ ถูกใช้เชื่อมโยงเข้ากับชนิดของ attribute ที่ซับซ้อน (กำหนดโดยผู้ใช้) เพื่อเลือกในฐานข้อมูลเชิงสัมพันธ์ (คล้ายๆ คอลัมน์ของมุมมองปกติที่ได้รับเชื่อมสู่การเลือก)

ซึ่งหมายเลขของวัตถุถูกสร้างโดยฐานข้อมูลตามปกติ มันเป็นหน้าที่ของผู้พัฒนาในการกำหนดหมายเลขของวัตถุ รวมเข้ากับคอลัมน์ในการกำหนด select (โดยชนิดนี้จะรวมเข้ากับคีย์หลักเพื่อใช้ในการ select)

ถ้า object view มีมากกว่า 1 ตารางแล้ว มันก็ยังเป็นหน้าที่ของผู้พัฒนาที่จะระบุวิธีการ insert วัตถุใหม่ หรือ update วิธีการใดวิธีการหนึ่งต้องถูกกระทำ นี่ถูกกระทำโดยการระบุ trigger INSTEAD-OF ซึ่งจะเข้ามาแทนที่ insert หรือ update

หลังจากนี้การระบุวัตถุสามารถถูกปฏิบัติโดยวัตถุอื่นในฐานข้อมูล

Oracle ได้ตั้งใจให้คุณสมบัตินี้ถูกใช้เมื่อระบบมีการปรับเปลี่ยนมาจากเชิงสัมพันธ์ไปยังเชิงวัตถุ ในลักษณะแบบค่อยๆ เป็นค่อยไป โดยปราศจากการเปลี่ยนแปลงพื้นฐานรูปแบบเชิงสัมพันธ์

มันน่าสนใจเพราะ Stonebraker ไม่มีสิ่งที่คล้ายกันเช่นนี้ (แม้ในทางทฤษฎี) ดูเหมือนเป็นคุณสมบัติที่มีประโยชน์มาก อย่างไรก็ตามสิ่งที่ Stonebraders สนใจไม่ใช่การย้ายระบบจากรูปแบบเชิงสัมพันธ์ ไปสู่รูปแบบเชิงวัตถุ แต่มันเป็นคุณสมบัติที่จำเป็นหลังจากการย้ายได้กระทำแล้ว

3.2.5 GENERAL DATABASE ASPECTS

ในส่วนนี้จะดูลักษณะพื้นฐานฐานข้อมูลใน Oracle 8

3.2.5.1 Transaction Management

Oracle 8 ให้วิธีการ transaction เดิมทั้งหมดที่พบในระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ขนาดใหญ่คล้าย atomic transactions และ consistency

เท่าที่เรามีพบไม่มีอะไรใหม่ และกลไก transaction ชั้นสูงคล้ายกับ instance co-operative transactions ใน Oracle 8

3.2.5.2 Concurrency Control

การควบคุมสภาวะพร้อมกันใน Oracle 8 เป็นสิ่งที่ดีมาก Oracle 8 (ทำใน Oracle 7) lock บนระดับเรคคอร์ด หรือระดับวัตถุ เพื่อให้มีจำนวน transaction มากๆ ทำงานพร้อมกันบนตารางใดๆ (ในตาราง เชิงสัมพันธ์ปกติ หรือเชิงวัตถุ)

3.2.5.3 Query Optimization

Oracle 8 มี 2 วิธีการทำ query optimization

- Rule based optimization เป็นการใช้วิธีเข้าถึง ที่ถูกกำหนดโดยชุดของกฎที่ตายตัว วิธีการ optimization นี้ ถูกใช้ในรุ่นแรกๆ ของฐานข้อมูล Oracle วิธีการ rule-based ไม่ให้วิธีที่ดีแก่นักเขียนแอปพลิเคชั่นโปรแกรม เพื่อวิธีการ optimize ที่ดี (นี่อาจจะเข้าใจได้ง่ายสำหรับนักเขียนโปรแกรมตั้งแต่ เขาหรือเธอรู้การแจกจ่ายของข้อมูลในตาราง)
- Cost based Optimization เป็นการใช้วิธีการเข้าถึง มีพื้นฐานของค่าใช้จ่ายในระบบ โดยพิจารณาการแจกจ่าย account และ cardinality ของข้อมูล ขนาดของตาราง และสมรถนะของระบบ I/O วิธีการนี้เริ่มต้นจาก i Oracle 7 และมีคำสั่งการทำงานที่ดีตั้งแต่รุ่น 7.3 ของฐานข้อมูล วิธีการนี้ยอมให้นักเขียนโปรแกรมแอปพลิเคชั่นสร้าง hint สำหรับ optimize เพื่อวิธีการที่ดีที่สุด

Oracle ได้เพิ่ม query optimizers (ทั้ง 2 วิธีการ) เพื่อจัดการคุณสมบัติเชิงวัตถุสัมพันธ์ query optimizers ใน Oracle 7 มีประสิทธิภาพโดยสมบูรณ์ ไม่มีเหตุผลเพื่อสรุปเอาว่ากรณีนี้ไม่มีใน Oracle 8 (ข้อเท็จจริงประการหนึ่งที่รับประกันโดย Oracle คือ cost based optimizer ได้ถูกปรับปรุงมาตั้งแต่ Oracle 7)

3.2.5.4 Versioning of Data

Oracle 8 ไม่สนับสนุน Versioning ของข้อมูล

3.2.5.5 Replication and Distribution

Oracle 8 ไม่สนับสนุน replication หรือ distribution ของวัตถุ มันทำสำหรับข้อมูลเชิงสัมพันธ์ Oracle สัญญาว่าคำสั่งทำงานนี้จะถูกรวมในรุ่นถัดไป (8.1)

3.2.5.6 Schema Evolution

มีปัญหาใน Oracle 8 กับการอ้างถึงการขยายของชนิด มันเป็นไปไม่ได้ในการขยายชนิดกับ attribute พิเศษ (ที่คล้ายกับการขยายตารางด้วยคอลัมน์พิเศษ)

นี่หมายถึงชนิดต้องถูกกำหนดอย่างถูกต้องจากตอนต้นๆ มาก และถ้าต่อมามันจำเป็นเพื่อการขยายชนิดแล้ว การปรับเปลี่ยนขนาดที่ใหญ่ของฐานข้อมูลทั้งหมด กลายเป็นสิ่งจำเป็น

ตั้งแต่เริ่มต้นมันเป็นประสบการณ์ของเรา ซึ่งจำเป็นเพื่อทำระหว่างช่วงชีวิตของระบบใดๆ บ่อยๆ ซึ่งดูคล้ายๆ กับเป็นอุปสรรคสำคัญ

3.2.5.7 Integration to Host Languages

Oracle 8 สนับสนุนภาษาการเขียนโปรแกรมของมันเอง (PL/SQL - ผลของการทำงานอย่างสมบูรณ์ที่ขยายมาตรฐานของ SQL) ทั้งระดับเซิฟเวอร์และไคลเอนท์ (เมื่อใช้เครื่องมือพัฒนาจาก Oracle) แต่มันเป็นไปได้ที่ใช้ได้จากภาษาทั่วๆ ไปเพื่อติดต่อกับฐานข้อมูล

ส่วนเชิงสัมพันธ์ของ Oracle 8 อาจจะถูกเข้าถึงโดยตรงโดยภาษา C, C++, COBOL และภาษาอื่นๆ ผ่านการใช้ของ pre-compilers และการสนับสนุนของภาษาใดๆ โดยใช้ชุดคำสั่งร่วมกันผ่านการเรียกใช้ส่วนติดต่อของ Oracle อย่างไรก็ตาม เราไม่รู้สึกถึงการใช้ส่วนติดต่อ OCI สามารถพิจารณารวมความจริงไปยัง class ของภาษาเจ้าของ และส่วนติดต่อ OCI จะไม่ถูกกล่าวถึงเพิ่มเติมในบทนี้

เมื่อดูส่วนเชิงวัตถุสัมพันธ์ของ Oracle 8 อย่างไรก็ตามมีในปัจจุบันเท่านั้นสนับสนุนโดยตรงสำหรับ C++ ผ่านการใช้ของ object type translator (OTT) และแคชวัตถุ การรวมกับ JAVA ยังไม่สนับสนุนแต่ Oracle รับประกันว่ามันจะมีในอนาคต

OTT เป็นเครื่องมือกำหนดโครงสร้างเพื่อใช้เมื่อสร้างโปรแกรมด้วย C++ ในการเข้าถึงฐานข้อมูล

Object cache คือพื้นที่หน่วยความจำบนไคลน์เอ็นท์ ซึ่งวัตถุถูกใช้โดยโปรแกรมไคลน์เอ็นท์ สิ่งนี้ยอมให้มีการเข้าถึงรีวกว่ากับวัตถุ ขณะที่ปฏิบัติคำสั่งกับแอปพลิเคชั่น เพราะ reference object สามารถ cached บนตัวแทนของไคลน์เอ็นท์ i โดยรับมาจากฐานข้อมูลในขณะที่มันถูกใช้ object cache เป็นคุณสมบัติที่ถูกพบในฐานข้อมูลเชิงวัตถุหลายๆ ตัว

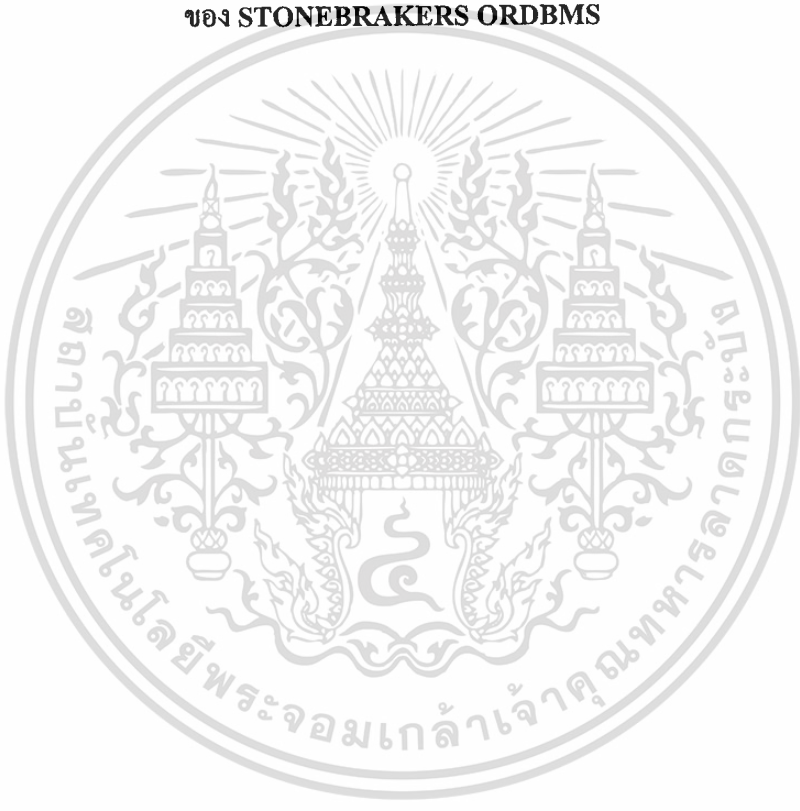
3.3 การเปรียบเทียบ INFORMIX-UNIVERSAL SERVER และ Oracle กับข้อกำหนดของ STONEBRAKERS ORDBMS

ข้อกำหนดของ Stonebrakers		Informix	Oracle
Base type extension	Dynamic linking	Yes	Yes
	Client and server execution	Yes	Can be programmed
	Security modules	No	Yes
	Callback	Yes	Yes

ข้อกำหนดของ Stonebrakers		Informix	Oracle
	User defined access methods	Yes	No
	Arbitrary length types	Yes	Yes
Complex types	Array	No - only lists	Yes
	Composites	Yes	
	Set	Yes - single and multiple	Yes
	Structure		Yes
	References	No	Yes
	SQL support	Yes (nothing for arrays)	Yes - SQL3 compliant.
Inheritance	Data and Function inheritance	Yes	No
	Overload of functions and methods	Yes	Yes
	Types and table is separate concepts	Yes	
	Multiple inheritance	No	
Rules	Query-rules	No	
	Rules on queries as well as changes in the database		No - Changes only
	Events	No	
	Restriction in rules implementation	No but restriction in rule definition	
	Rules in separate transactions		No
	No limitations on rule functionality		Yes
	Immediate/deferred rules	Yes	Partially – before/after insert but not deferred until commit

ข้อกำหนดของ Stonebrakers		Informix	Oracle
	Same or different transaction	No	
	Loop detection	Yes-some	Yes-but somewhat restrictive

ตารางที่ 3.3 การเปรียบเทียบ INFORMIX-UNIVERSAL SERVER และ Oracle กับข้อกำหนดของ STONEBRAKERS ORDBMS



บทที่ 4
หลักเกณฑ์ในการให้คะแนน

การกำหนดคะแนนในเครื่องมือนี้ ใช้วิธีการของแบบสอบถาม (Questionnaire) นำมาทำมาตราส่วนประมาณค่า หรือจัดอันดับ โดยใช้มาตราส่วนประมาณค่าแบบกำหนดตัวเลข (Numerical Rating Scale) เป็นแบบวัดที่กำหนดตัวเลขแทนความสามารถ โดยมีเกณฑ์การแบ่งระดับและแปลความหมายดังนี้

ถ้าให้ 0 = ไม่มี/ไม่ระบุ

1 = น้อย

2 = ปานกลาง

3 = มาก

แบบสอบถามดังกล่าว จะถูกจัดให้อยู่ในรูปของตาราง เนื่องจากใช้ตัวเลือกเหมือนกัน โดยแบบสอบถามจะประกอบด้วย 2 ส่วนดังต่อไปนี้ คือ

4.1 แบบสอบถามคะแนนคุณสมบัติ

แบบสอบถามคุณสมบัติ เป็นแบบสอบถามที่ใช้กับบริษัทให้คำปรึกษาเกี่ยวกับผลิตภัณฑ์ หรือบริษัทที่จัดจำหน่ายผลิตภัณฑ์

คุณสมบัติ		Informix				Oracle			
		3	2	1	0	3	2	1	0
Base type extension	Dynamic linking	/				/			
	Client and server execution		/				/		
	Security modules				/		/		
	Callback		/			/			
	User defined access methods	/							/
	Arbitrary length types	/					/		
Complex types	Array			/		/			
	Composites	/				/			

คุณสมบัติ		Informix				Oracle			
		3	2	1	0	3	2	1	0
	Set	/					/		
	Structure				/		/		
	References				/		/		
	SQL support		/			/			
Inheritance	Data and Function inheritance		/						/
	Overload of functions and methods		/				/		
	Types and table is separate concepts	/							/
	Multiple inheritance				/				/
Rules	Query-rules				/				/
	Rules on queries as well as changes in the database				/				/
	Events				/				/
	Restriction in rules implementation				/				/
	Rules in separate transactions				/				/
	No limitations on rule functionality				/	/			
	Immediate/deferred rules	/					/		
	Same or different transaction				/				/
	Loop detection			/				/	

ตารางที่ 4.1 แสดง แบบสอบถามคะแนนคุณสมบัติของผลิตภัณฑ์

4.2 แบบสอบถามคะแนนประเภทงาน

แบบสอบถามประเภทงาน เป็นแบบสอบถามที่ให้คะแนนโดยคณะกรรมการ การกำหนดคุณลักษณะของผลิตภัณฑ์ที่จะนำมาใช้ในงาน โดยคะแนนดังกล่าวจะเป็นผลที่ได้จากการประชุมภายในคณะกรรมการ ซึ่งแม้จะเป็นงานประเภทเดียวกัน คะแนนที่ให้อาจไม่เท่ากัน ขึ้นอยู่กับความสนใจของกรรมการในแต่ละชุด

เมื่อได้คะแนนทั้งสองส่วนแล้ว จึงจะนำคะแนนที่ได้นั้นมาคำนวณหาค่าการประเมินผลของผลิตภัณฑ์ ว่าผลิตภัณฑ์ใดผ่านการพิจารณาคัดเลือก และนำส่งผลที่ได้นั้นไปยังกรรมการเปิดซอง เพื่อใช้ประกอบการพิจารณาเทียบกับปัจจัยอื่นๆ เช่น ราคา เป็นต้น

บทที่ 5
การพัฒนาเครื่องมือสำหรับเปรียบเทียบคุณสมบัติของระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์

เครื่องมือสำหรับเปรียบเทียบคุณสมบัติของระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ มีพื้นฐานมาจากการรวบรวมคุณสมบัติต่างๆ ที่พบได้ในระบบจัดการฐานข้อมูลฯ แล้วนำค่าที่ได้มาเปรียบเทียบกันเพื่อหาระบบจัดการฐานข้อมูลฯ ที่เราให้ความสนใจมากที่สุด ดังรายละเอียดต่อไปนี้

5.1 การออกแบบระบบ

เครื่องมือเปรียบเทียบคุณสมบัติฯ มีเป้าหมายเพื่อช่วยให้ข้อมูลสำหรับกรรมการพิจารณาผล ในการตัดสินใจเลือกระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ แต่ทั้งนี้ ต้องมีการรวบรวมข้อมูลที่จำเป็นสำหรับระบบ โดยเครื่องมือนี้จะเกี่ยวข้องกับบุคคล 4 กลุ่มด้วยกัน ประกอบด้วย

- ผู้เชี่ยวชาญ ทำหน้าที่ในการให้รายละเอียดคุณสมบัติของระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์
- บริษัท/ผู้ให้คำปรึกษา ทำหน้าที่ให้คะแนนความสามารถของผลิตภัณฑ์ที่นำมาพิจารณาผล
- กรรมการกำหนดคุณสมบัติ ทำหน้าที่กำหนดคะแนนความสนใจ สำหรับงานที่จะนำระบบจัดการฐานข้อมูลฯ มาใช้
- กรรมการพิจารณาผล รวบรวมข้อมูลข้างต้น นำมาตัดสินใจเพื่อเลือกระบบจัดการฐานข้อมูลที่นำเสนอโดย บริษัท/ผู้ให้คำปรึกษา

5.1.1 หลักการทำงานของระบบ

เพื่อให้ตอบสนองการทำงานของผู้ใช้งานแต่ละกลุ่ม ดังนั้นระบบต้องมีคุณสมบัติดังนี้

- ผู้เชี่ยวชาญ สามารถออกแบบคุณสมบัติต่างๆ ที่มีในระบบจัดการฐานข้อมูล อยู่ในรูปของคำถาม เพื่อผู้ให้คำปรึกษา หรือ กรรมการกำหนดคุณสมบัติ นำไปกำหนดคะแนน
- บริษัท/ผู้ให้คำปรึกษา กำหนดคะแนนความสามารถในแต่ละคุณสมบัติ จากแบบคำถามที่ผู้เชี่ยวชาญได้ออกแบบเอาไว้ ให้กับระบบจัดการฐานข้อมูลที่นำมาทดสอบ
- กรรมการกำหนดคุณสมบัติ กำหนดคะแนนความสนใจเฉพาะงาน จากแบบคำถามที่ผู้เชี่ยวชาญได้ออกแบบเอาไว้

- กรรมการพิจารณาผล เรียกดูคะแนนรวมเพื่อพิจารณาตัดสินใจ

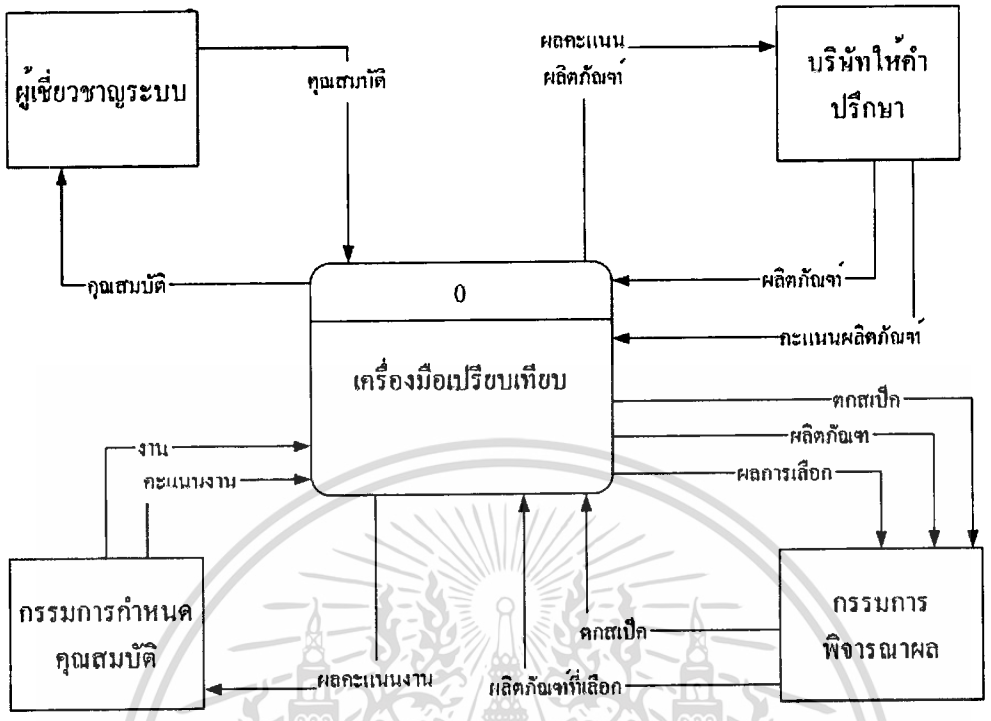
5.1.2 Process Modeling

เมื่อได้ข้อกำหนดดังกล่าวแล้ว นำมาพิจารณาสร้างเครื่องมือเปรียบเทียบคุณสมบัติของระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ ดังรายการต่อไปนี้

5.1.2.1 Data Flow Diagram (Context Diagram)

จากข้อกำหนดดังกล่าว จะได้ External Entity ซึ่งสัมพันธ์กับระบบ ดังนี้

- ผู้เชี่ยวชาญระบบ
  - รับคำถามเกี่ยวกับคุณสมบัติจากระบบ
  - ส่งข้อมูลคุณสมบัติกลับไปยังระบบ
- บริษัทให้คำปรึกษา
  - ระบุผลิตภัณฑ์ที่จะนำมาทดสอบในระบบ
  - ให้คะแนนความสามารถของคุณสมบัติ
  - รับรายงานผลคะแนนของผลิตภัณฑ์
- กรรมการกำหนดคุณสมบัติ
  - ระบุประเภทงานที่จะนำระบบจัดการฐานข้อมูลมาใช้
  - ให้คะแนนความสนใจในคุณสมบัติ ตามประเภทของงาน
  - รับรายงานผลคะแนนของประเภทงาน
- กรรมการพิจารณาผล
  - ถามข้อมูลที่ไม่เข้าคุณสมบัติจากระบบ (ตกสเป็ค)
  - รับข้อมูลที่ไม่เข้าคุณสมบัติจากระบบ (ตกสเป็ค)
  - เลือกผลิตภัณฑ์ที่ต้องการเข้าทดสอบ
  - พิจารณาตัดสินเลือกผลิตภัณฑ์
  - รับรายงานผลการตัดสินเลือกผลิตภัณฑ์



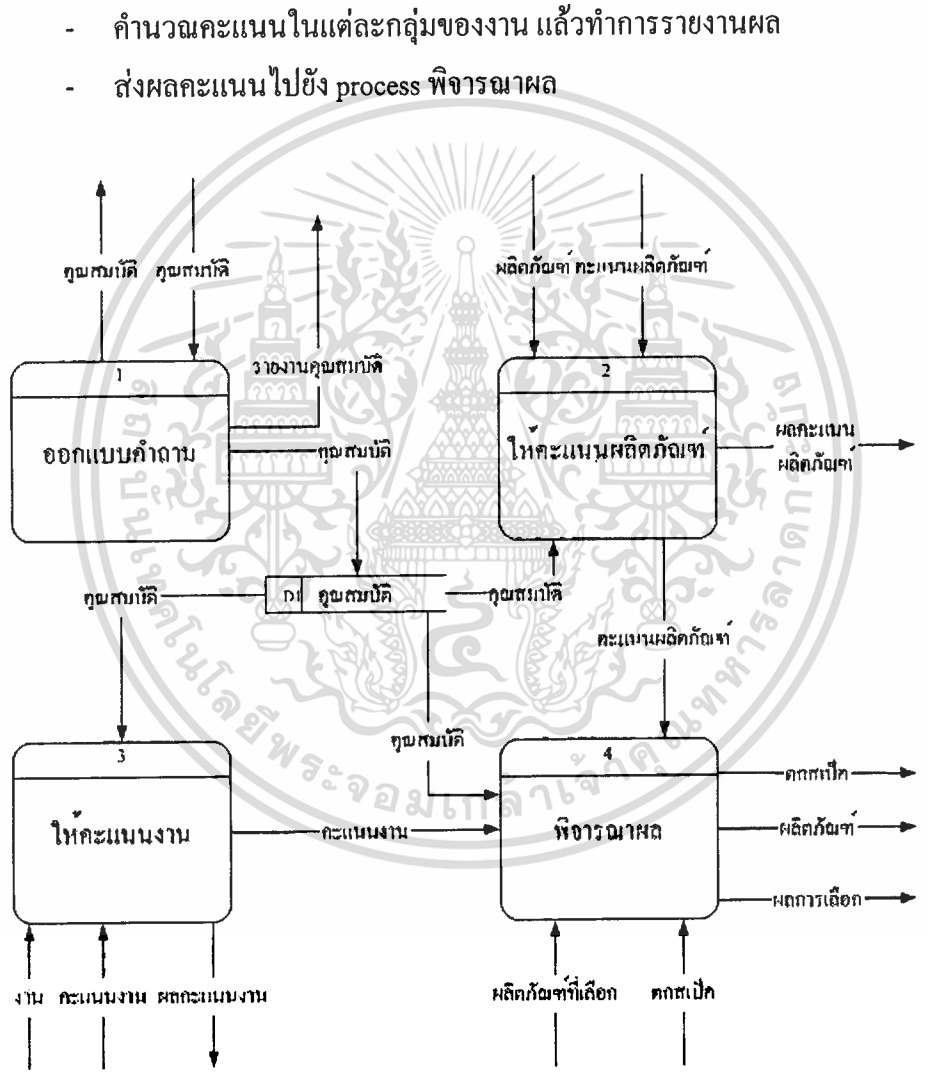
รูปที่ 5.1 แสดง Context Diagram ของเครื่องมือเปรียบเทียบ ฯ

5.1.2.2 Data Flow Diagram ระดับที่ 0

ในระดับที่ 0 ได้แบ่งระบบออกเป็น process ย่อย ได้ 4 process ดังนี้

- Process ที่ 1 ออกแบบคำถาม
  - สอบถามคุณสมบัติไปยังผู้เชี่ยวชาญระบบซึ่งเป็น external entity
  - รับคุณสมบัติซึ่งผู้เชี่ยวชาญได้แจ้งเข้ามา
  - บันทึกคุณสมบัติไปยังแฟ้มข้อมูล
  - รายงานรายการคุณสมบัติออกไปให้ยังผู้เชี่ยวชาญระบบ
- Process ที่ 2 ให้คะแนนผลิตภัณฑ์
  - บริษัทให้คำปรึกษา ระบุผลิตภัณฑ์ให้กับระบบ
  - Process รับเอาคุณสมบัติมาจากแฟ้มคุณสมบัติ
  - บริษัทให้คำปรึกษา ให้คะแนนความสามารถคุณสมบัติของแต่ละผลิตภัณฑ์
  - คำนวณคะแนนในแต่ละกลุ่มของผลิตภัณฑ์ แล้วทำการรายงานผล
  - ส่งผลคะแนนไปยัง process พิจารณาผล

- Process ที่ 3 ให้คะแนนงาน
  - กรรมการกำหนดคุณสมบัติ กำหนดประเภทงานที่จะนำระบบจัดการฐานข้อมูลมาใช้งาน
  - กำหนดคะแนนความสนใจแต่ละประเภทงาน ให้กับคุณสมบัติ (คะแนนได้จากการประชุมคณะกรรมการ)
  - Process รับเอาคุณสมบัติมาจากแฟ้มคุณสมบัติ
  - คำนวณคะแนนในแต่ละกลุ่มของงาน แล้วทำการรายงานผล
  - ส่งผลคะแนนไปยัง process พิจารณาผล



รูปที่ 5.2 แสดง Level 0 ของเครื่องมือเปรียบเทียบฯ

- Process ที่ 4 พิจารณาผล
  - นำคะแนนผลิตภัณฑ์ และคะแนนงานในแต่ละกลุ่มของคุณสมบัติ มาคำนวณหาค่าคะแนนอีกครั้ง
  - หาข้อมูลที่ไม่เข้าคุณสมบัติ โดยแยกในแต่ละผลิตภัณฑ์
  - รายงานผลข้อมูลที่ไม่เข้าคุณสมบัติ
  - รายงานผลการตัดสิน

5.1.2.3 Data Flow Diagram ระดับที่ 1 ของการออกแบบคุณสมบัติ

ทำหน้าที่ในการสร้างแบบฟอร์ม เพื่อนำไปใช้ในการให้คะแนนผลิตภัณฑ์ และคะแนนงาน พร้อมทั้งจัดกลุ่มของคุณสมบัติ โดยประกอบด้วย process ดังต่อไปนี้

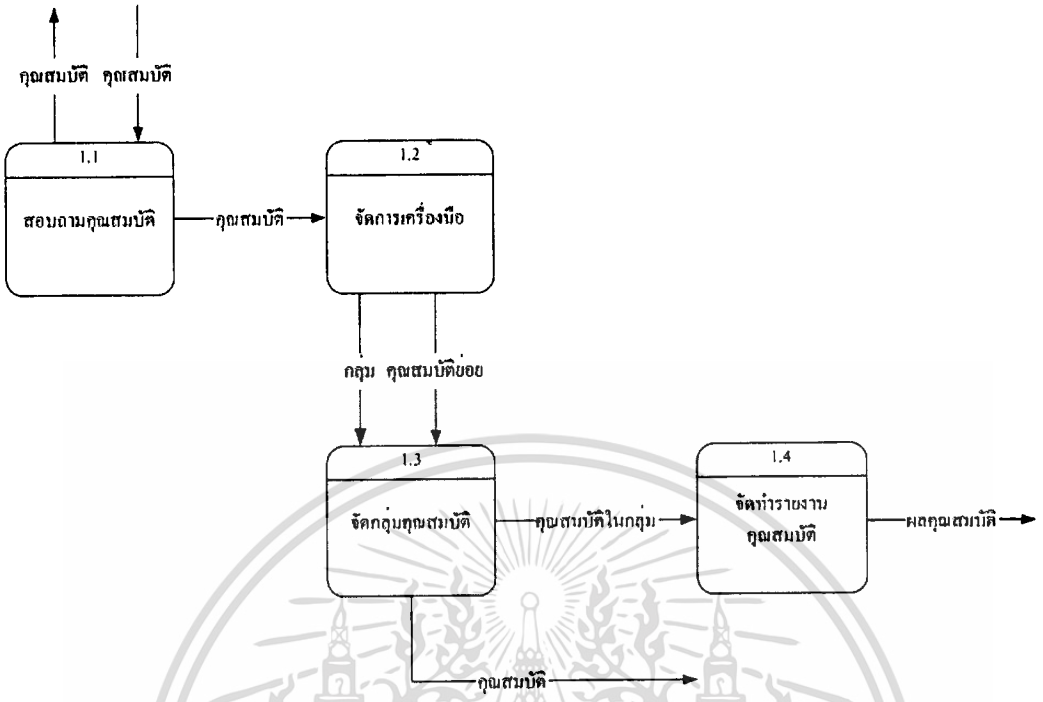
- Process ที่ 1.1 สอบถามคุณสมบัติ

  สอบถามคุณสมบัติจากผู้เชี่ยวชาญระบบ และรับคุณสมบัติที่ได้รับการตอบกลับมา
- Process ที่ 1.2 จัดการเครื่องมือ

  กำหนดตำแหน่งของหน้าจอ ในแบบฟอร์มคำถาม รวมถึงระบุว่า object ใดเป็นคุณสมบัติ และ object ใดไม่ใช่
- Process ที่ 1.3 จัดกลุ่มคุณสมบัติ

  คุณสมบัติหลักย่อมมีคุณสมบัติลูก และคุณสมบัติลูกหากยังไม่ใช่ node สุดท้าย ก็สามารถที่จะกระจายกลุ่มต่อไปเรื่อยๆ Process นี้จะช่วยในการจัดกลุ่ม รวมทั้งกระจายกลุ่ม หากคุณสมบัติดังกล่าว มี node ลูกต่อไปเรื่อยๆ
- Process ที่ 1.4 จัดทำรายงานคุณสมบัติ

  จัดทำรายงานคุณสมบัติทั้งหมดที่ได้บันทึกลงไป พร้อมทั้งแยกกลุ่มในแต่ละคุณสมบัติ ในลักษณะของแผนภูมิต้นไม้ (tree)



รูปที่ 5.3 แสดง Level 1 ของ Process ออกแบบคำถาม

5.1.2.4 Data Flow Diagram ระดับที่ 1 ของการให้คะแนนผลิตภัณฑ์

ทำหน้าที่ในการให้คะแนนกับผลิตภัณฑ์ โดยยึดจากแบบฟอร์มสอบถามที่ได้ถูกสร้างมาแล้ว การให้คะแนนนั้น จะให้ครบทุกผลิตภัณฑ์ หรือทีละตัวก็ได้ ประกอบด้วย process ดังต่อไปนี้

- Process ที่ 2.1 เลือกคุณสมบัติ

  เลือกคุณสมบัติจากแบบฟอร์มสอบถาม

- Process ที่ 2.2 กำหนดคะแนน

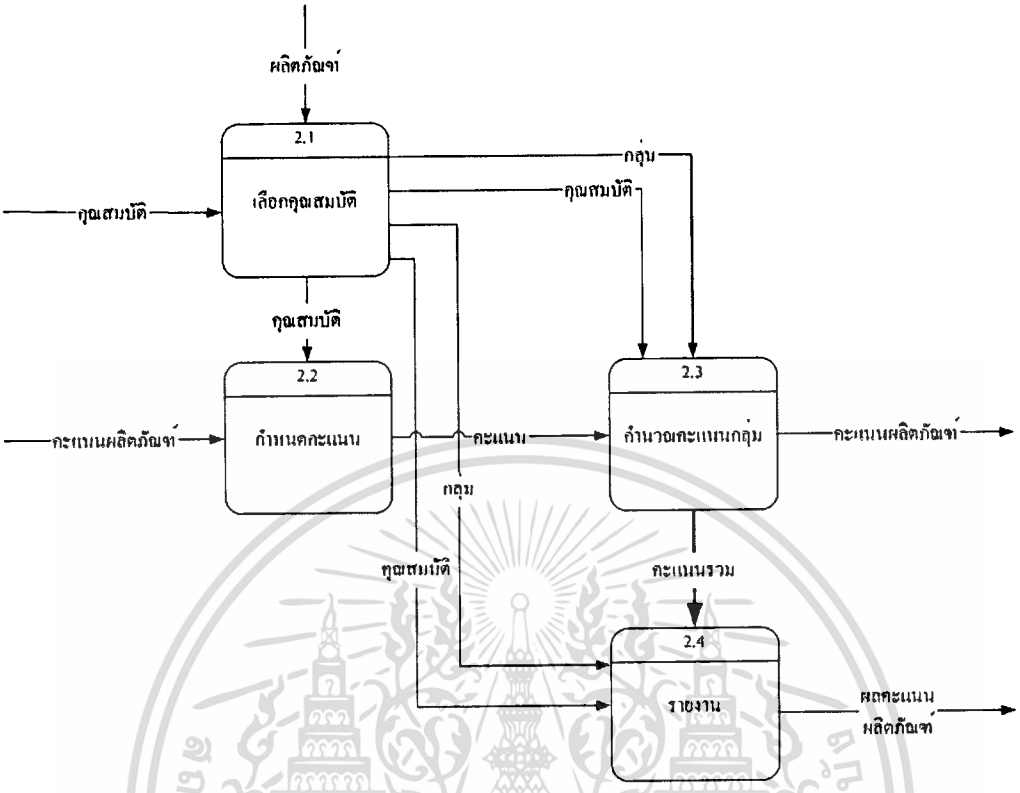
  ให้คะแนนความสามารถในแต่ละกลุ่มของผลิตภัณฑ์ พร้อมทั้งหมายเหตุ (ถ้ามี)

- Process ที่ 2.3 คำนวณคะแนนกลุ่ม

  ในแต่ละกลุ่มจะมีคะแนนเต็มไม่เกิน 3 คะแนน process นี้จะนำคะแนนจากคุณสมบัติย่อย มาคำนวณ เพื่อให้ได้สัดส่วนที่ไม่เกิน 3 คะแนน

- Process ที่ 2.4 รายงาน

  จัดทำรายงานที่ได้บันทึกคะแนนให้กับผลิตภัณฑ์ ว่าผลคะแนนทั้งหมดเป็นเท่าไร



รูปที่ 5.4 แสดง Level 1 ของ Process การให้คะแนนผลิตภัณฑ์

5.1.2.5 Data Flow Diagram ระดับที่ 1 ของการให้คะแนนงาน

ทำหน้าที่ในการให้คะแนนงาน โดยยึดจากแบบฟอร์มสอบถามที่ได้ถูกสร้างมาแล้ว การให้คะแนนนั้น จะให้ครบทุกประเภทงาน หรือทีละตัวก็ได้ (การทำงานเหมือนกับการให้คะแนนผลิตภัณฑ์) ประกอบด้วย process ดังต่อไปนี้

- Process ที่ 3.1 เลือกคุณสมบัติ

  เลือกคุณสมบัติจากแบบฟอร์มสอบถาม

- Process ที่ 3.2 กำหนดคะแนน

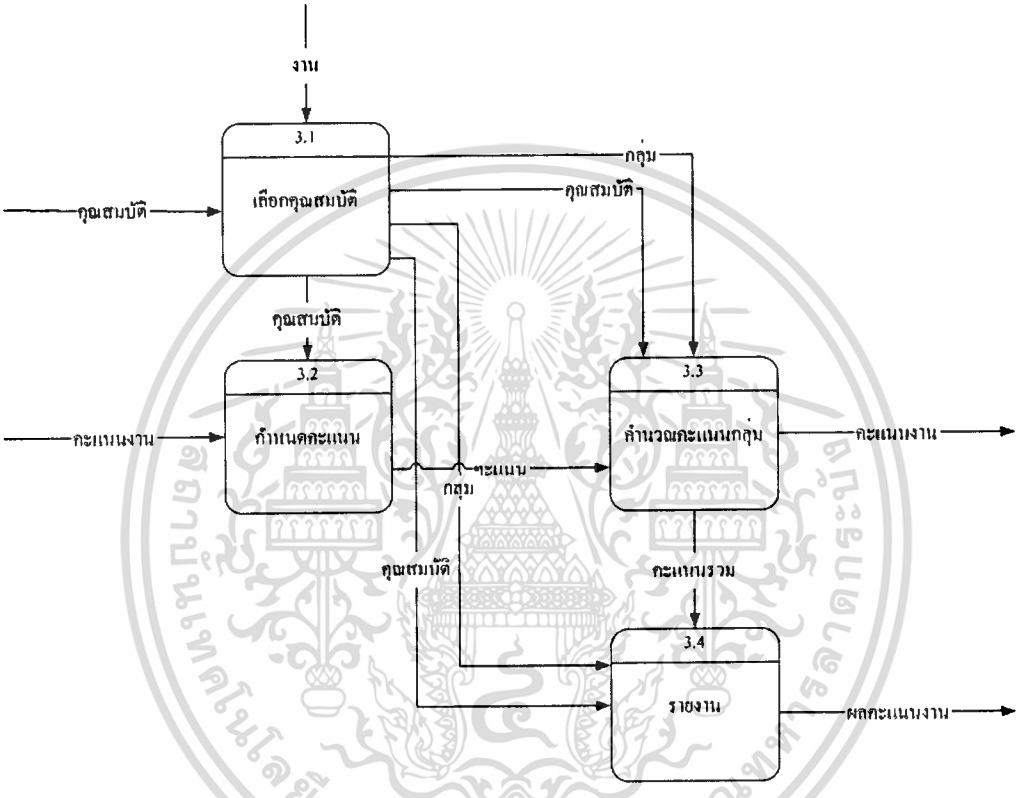
  ให้คะแนนความสามารถในแต่ละกลุ่มของประเภทงาน พร้อมทั้งหมายเหตุ (ถ้ามี)

- Process ที่ 3.3 คำนวณคะแนนกลุ่ม

  ในแต่ละกลุ่มจะมีคะแนนเต็มไม่เกิน 3 คะแนน process นี้จะนำคะแนนจากคุณสมบัติย่อย มาคำนวณ เพื่อให้ได้สัดส่วนที่ไม่เกิน 3 คะแนน

- Process ที่ 3.4 รายงาน

  จัดทำรายงานที่ได้บันทึกคะแนนให้กับประเภทงาน ว่าผลคะแนนทั้งหมดเป็นเท่าไร



รูปที่ 5.5 แสดง Level 1 ของ Process การให้คะแนนงาน

5.1.2.6 Data Flow Diagram ระดับที่ 1 ของการพิจารณาผล

หน้าที่หลักคือแสดงข้อมูลคุณสมบัติและคะแนนทั้งหมด เพื่อประกอบการตัดสินใจเลือกผลิตภัณฑ์ ประกอบด้วย process ดังต่อไปนี้

- Process ที่ 4.1 เปรียบเทียบผลิตภัณฑ์

  เปรียบเทียบคุณสมบัติต่างๆ ของแต่ละผลิตภัณฑ์ ว่ามีอัตราส่วนแตกต่างกันเท่าไร

- Process ที่ 4.2 แสดงรายการตกสเป็ค

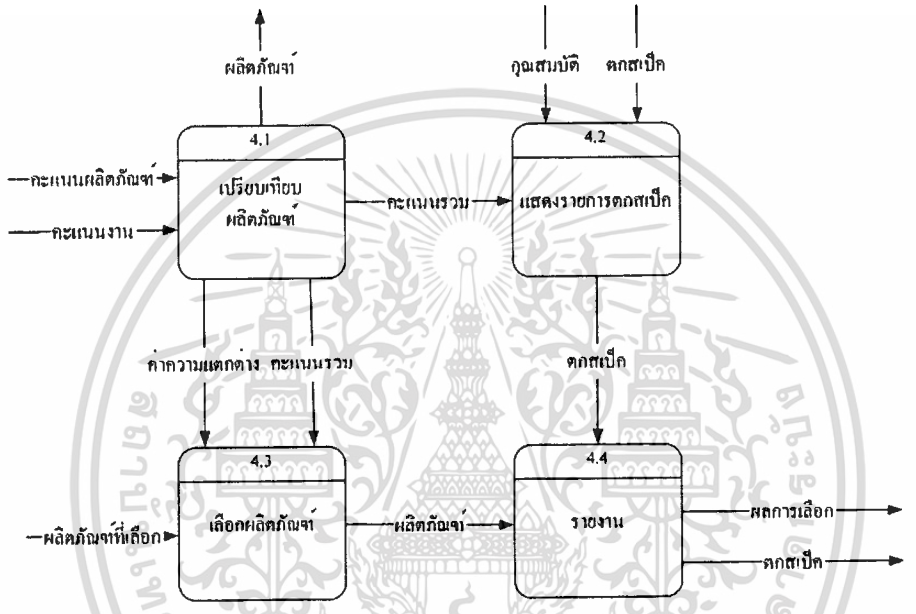
  แสดงรายการที่ไม่เข้าสเป็คของแต่ละผลิตภัณฑ์ โดยแยกตามความต้องการว่าสนใจเรียกดูผลิตภัณฑ์ใดตัวใด

- Process ที่ 4.3 เลือกผลิตภัณฑ์

  เลือกผลิตภัณฑ์ที่ โดยพิจารณาจากคุณสมบัติที่ได้ทำการเปรียบเทียบให้เห็น

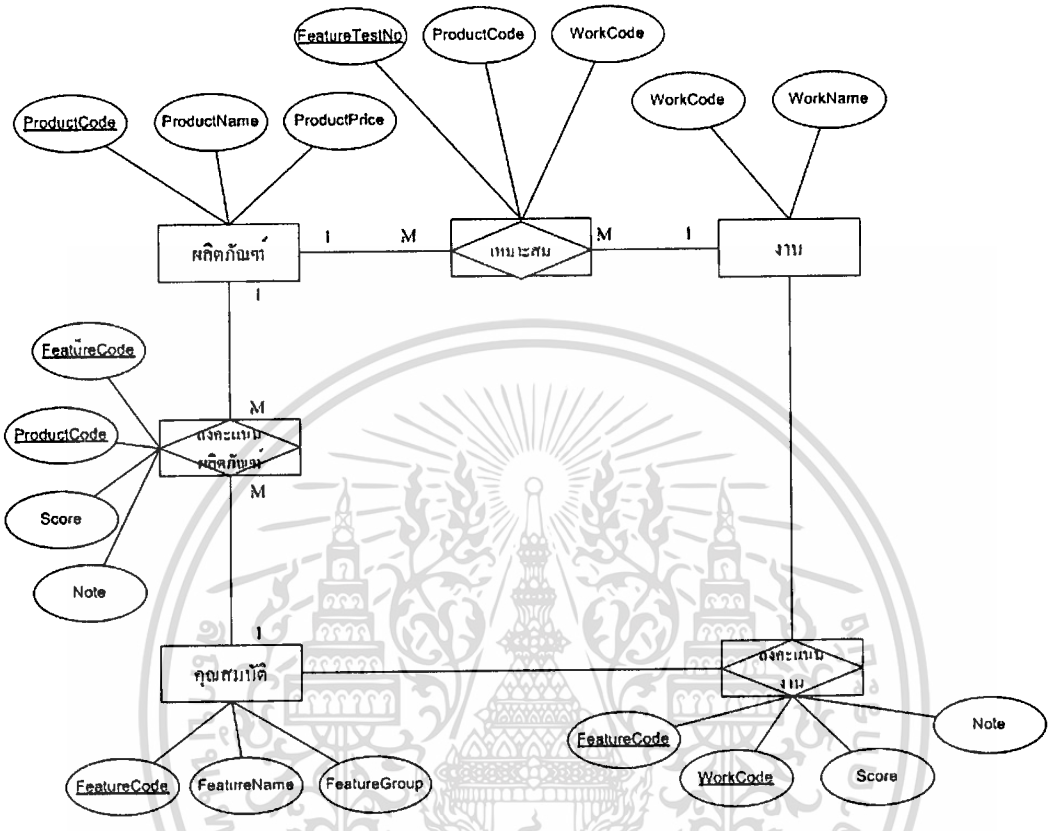
- Process ที่ 4.4 รายงาน

  จัดทำรายงานที่ผลการตัดสิน พร้อมรายละเอียดประกอบการพิจารณาจากคณะกรรมการ



รูปที่ 5.6 แสดง Level 1 ของ Process การพิจารณาผล

5.1.3 ER-Diagram



รูปที่ 5.7 แสดง Entity Relationship Diagram

จากรูปที่ 5.7 แสดงความสัมพันธ์ระหว่างตารางข้อมูลนั้นสามารถนำมาออกแบบตารางข้อมูลได้ดังต่อไปนี้

- ตาราง PRODUCT เก็บข้อมูลของผลิตภัณฑ์
- ตาราง WORK เก็บประเภทของงาน
- ตาราง FEATURE เก็บคุณสมบัติ และความสัมพันธ์ของคุณสมบัติ
- ตาราง PRODUCTSCORE เก็บคะแนนของคุณสมบัติ
- ตาราง WORKSCORE เก็บคะแนนของคุณสมบัติ
- ตาราง FEATURETEST เก็บผลการพิจารณา

5.1.4 Data Dictionary

Name	Description	Type	Size	PK/FK	Derive.
PRODUCTCODE	รหัสผลิตภัณฑ์	C	3	PK	
PRODUCTNAME	ชื่อผลิตภัณฑ์	C	40		
PRODUCTPRICE	ราคาผลิตภัณฑ์	N			

ตารางที่ 5.1 แสดง Data Dictionary ของ TABLE PRODUCT

Name	Description	Type	Size	PK/FK	Derive.
WORKCODE	รหัสประเภทงาน	C	3	PK	
WORKNAME	ชื่อประเภทงาน	C	40		

ตารางที่ 5.2 แสดง Data Dictionary ของ TABLE WORK

Name	Description	Type	Size	PK/FK	Derive.
FEATURECODE	รหัสคุณสมบัติ	C	3	PK	
FEATURENAME	รายละเอียดคุณสมบัติ	C	40		
FEATUREGROUP	กลุ่มคุณสมบัติ	C	3		

ตารางที่ 5.3 แสดง Data Dictionary ของ TABLE FEATURE

Name	Description	Type	Size	PK/FK	Derive.
PRODUCTCODE	รหัสผลิตภัณฑ์	C	3	PK	
FEATURECOD	รหัสคุณสมบัติ	C	3	PK	
RATING	คะแนนความสนใจ	N			
NOTE	หมายเหตุ/เทคนิคที่ใช้	C	40		

ตารางที่ 5.4 แสดง Data Dictionary ของ TABLE PRODUCTSCORE

Name	Description	Type	Size	PK/FK	Derive.
WORKCODE	รหัสประเภทของงาน	C	3	PK	
FEATURECODE	รหัสคุณสมบัติ	C	3	PK	
RATING	คะแนนความสนใจ	N			
NOTE	หมายเหตุ/เทคนิคที่ใช้	C	40		

ตารางที่ 5.5 แสดง Data Dictionary ของ TABLE WORKSCORE

Name	Description	Type	Size	PK/FK	Derive.
FEATURETESTNO	ลำดับรายการตัดสินใจ	C	3	PK	
WORKCODE	รหัสงาน	C	3	PK	
PRODUCTCODE	รหัสผลิตภัณฑ์	C	3	PK	

ตารางที่ 5.6 แสดง Data Dictionary ของ TABLE FEATURECODE

บทที่ 6
สรุปและข้อเสนอแนะ

6.1 สรุปผลการพัฒนาระบบ

ผลจากการพัฒนาเครื่องมือสำหรับเปรียบเทียบคุณสมบัติของระบบการจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์มีดังนี้

- มีความง่าย และยืดหยุ่นแก่ผู้ใช้ในการตั้งคำถามด้วยตนเอง
- เพิ่มทางเลือกในการตัดสินใจเลือกผลิตภัณฑ์ใดผลิตภัณฑ์หนึ่ง หากค่าเปรียบเทียบที่ได้นั้นมีความแตกต่างกันมาก แต่โดยนำเอาปัจจัยด้านราคามาเปรียบเทียบด้วย
- ในการใช้งาน แยกกลุ่มผู้ใช้อย่างชัดเจน นั่นคือ ในแต่ละกลุ่ม ไม่มีความจำเป็นในการเรียนรู้การใช้งานของอีกกลุ่ม

6.2 ข้อเสนอแนะ

การพิจารณาจัดซื้อจัดจ้าง ผลิตภัณฑ์หรือบริการนั้น ไม่จำกัดอยู่เฉพาะระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์แต่เพียงอย่างเดียว หากมีผลิตภัณฑ์ หรือบริการประเภทอื่นๆ อีก ดังนั้น แนวทางในการพัฒนาโปรแกรมต่อไป สามารถพัฒนาให้เครื่องมือสามารถใช้กับงานทั่วๆ ไปได้มากกว่านี้ ทั้งนี้เพราะโครงสร้างหลักๆ ของโปรแกรมไม่ได้ยึดอยู่กับเฉพาะระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์แต่เพียงเท่านั้น (โดยการเพิ่มกลุ่มของประเภทงานเข้าไป) เพราะงานหลักๆ ของเครื่องมือชนิดนี้ อ้างอิงไปยังทฤษฎีระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์มากกว่าเทคนิคทางด้านโปรแกรม

อีกประการหนึ่งคืออาจเพิ่มในส่วนการติดตามผลของการตรวจรับ และ/หรือ ให้มีการเปรียบเทียบที่ข้ามสถาปัตยกรรมกันอีกก็มีความเป็นไปได้

บรรณานุกรม

Lisbeth Bergholt, DTI. et al. 1998, **Database Management Systems: Relational, Object-Relational, and Object-Oriented Data Models.** [Online]. Available http://www.cit.dk/COT/reports/reports/Case4/05-v1.1/cot-4-05-1.1.pdf

Michael Stonebraker, Paul Brown. 1999, **Object-Relational DBMSs Tracking the new great wave.** Morgan Kaufmann Publishers, Inc.

Paul Brown. 2001, **Object-Relational Database Development: A Plumber's Guide.** [Online]. Available http://vig.pearsoned.com/samplechapter/0130194603.pdf

ศิริเพ็ญ มากบุญ. 2533, **การวิจัยเบื้องต้น.** วิทยาลัยครูนครสวรรค์.

ภาคผนวก

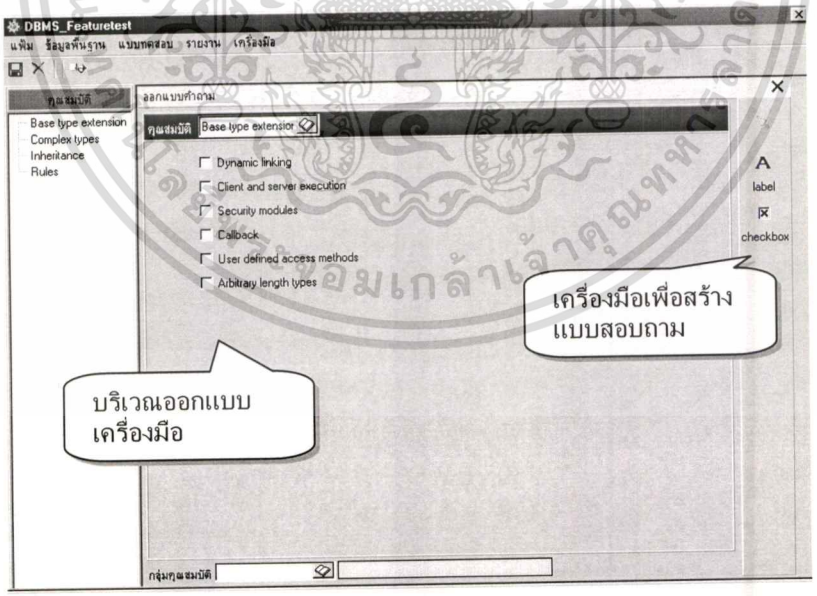
การใช้งานโปรแกรม

โปรแกรมตอบสนองการใช้งานของกลุ่มผู้ใช้แบ่งออกได้ดังต่อไปนี้

ผู้ใช้กลุ่มที่ 1: ผู้ใช้ที่มีความชำนาญ

ทำหน้าที่ในการรวบรวมคุณสมบัติของระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ มาบันทึกให้กับระบบ ดังนั้นสรุปงานในส่วนนี้ได้ดังต่อไปนี้

- ผู้ใช้สามารถสร้างแบบสอบถามได้ด้วยตนเอง ดังนั้นจึงต้องมีเครื่องมือที่จำเป็นสำหรับการสร้างแบบสอบถามให้ผู้ใช้ได้เลือกใช้
- มีความสามารถในการจัดการและแยกแยะการทำงานของเครื่องมือได้
- ผู้ใช้เลือกเครื่องมือพร้อมจัดกลุ่มตามคุณสมบัติ
- หากมีคุณสมบัติย่อย ผู้ใช้สามารถกระจายกลุ่มเครื่องมือได้ โปรแกรมจะทำการจัดกลุ่มย่อยให้โดยอัตโนมัติ

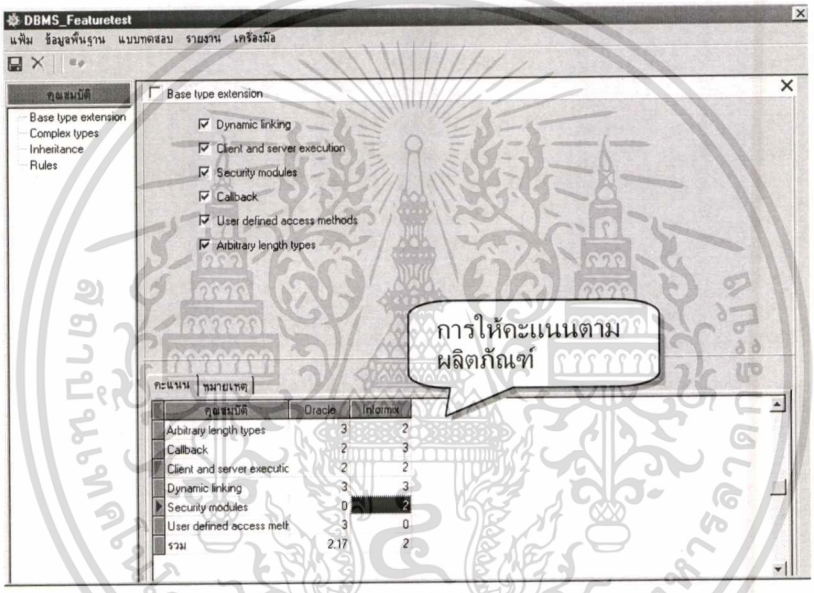


รูปที่ 6.1 แสดงจอภาพสำหรับออกแบบคำถาม

ผู้ใช้กลุ่มที่ 2: ผู้ให้คำปรึกษาเกี่ยวกับผลิตภัณฑ์

ทำหน้าที่ให้ข้อมูลว่าผลิตภัณฑ์ที่ตนเองชำนาญนั้น มีคุณสมบัติใด และมีความสามารถเพียงใด โดยจะนำข้อมูลที่ได้มาบันทึกใส่ไว้ในระบบ ตามคำถามที่ได้ออกแบบไว้โดยผู้ใช้กลุ่มแรก ซึ่งเกณฑ์การให้คะแนนจะมีตั้งแต่ 0 ถึง 3 คะแนน และนำคะแนนย่อยไปหาคะแนนในแต่ละกลุ่ม ซึ่งก็มีตั้งแต่ 0 ถึง 3 เช่นกัน ดังนั้นสรุปงานในส่วนนี้ได้ดังต่อไปนี้

- เลือกระบบจัดการฐานข้อมูลที่ต้องการจะนำมาใส่คะแนน
- ป้อนคะแนนตามหัวข้อของคุณสมบัติ

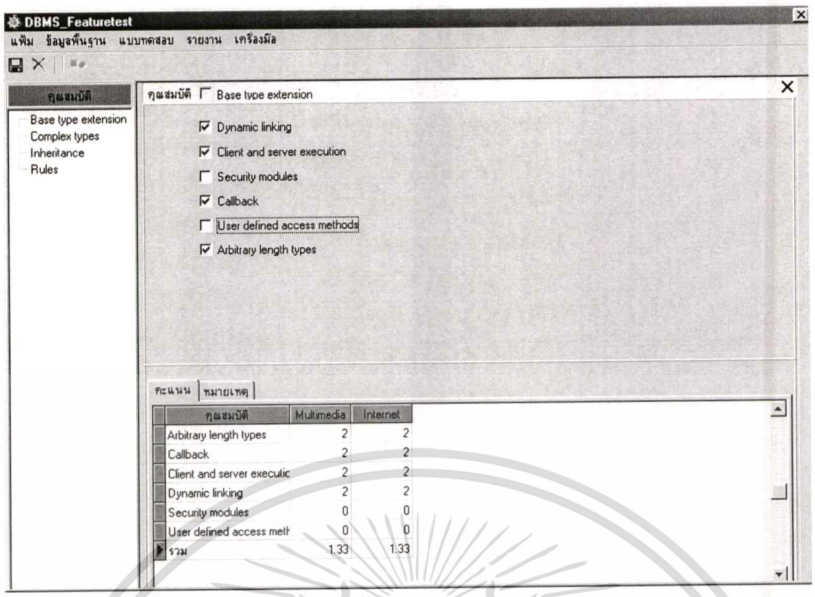


รูปที่ 6.2 แสดงจอภาพสำหรับการให้คะแนนผลิตภัณฑ์

ผู้ใช้กลุ่มที่ 3: คณะกรรมการกำหนดคุณสมบัติ

ทำหน้าที่เลือกและให้คะแนนคุณสมบัติ ตามประเภทของงานโดยจะนำข้อมูลที่ได้มาบันทึกใส่ไว้ในระบบ ตามคำถามที่ได้ออกแบบไว้โดยผู้ใช้กลุ่มแรก ซึ่งเกณฑ์การให้คะแนนจะมีตั้งแต่ 0 ถึง 3 คะแนน และนำคะแนนย่อยไปหาคะแนนในแต่ละกลุ่ม ซึ่งก็มีตั้งแต่ 0 ถึง 3 เช่นกัน ดังนั้นสรุปงานในส่วนนี้ได้ดังต่อไปนี้

- เลือกประเภทงานที่ต้องการจะนำมาใส่คะแนน
- ป้อนคะแนนตามหัวข้อของคุณสมบัติ

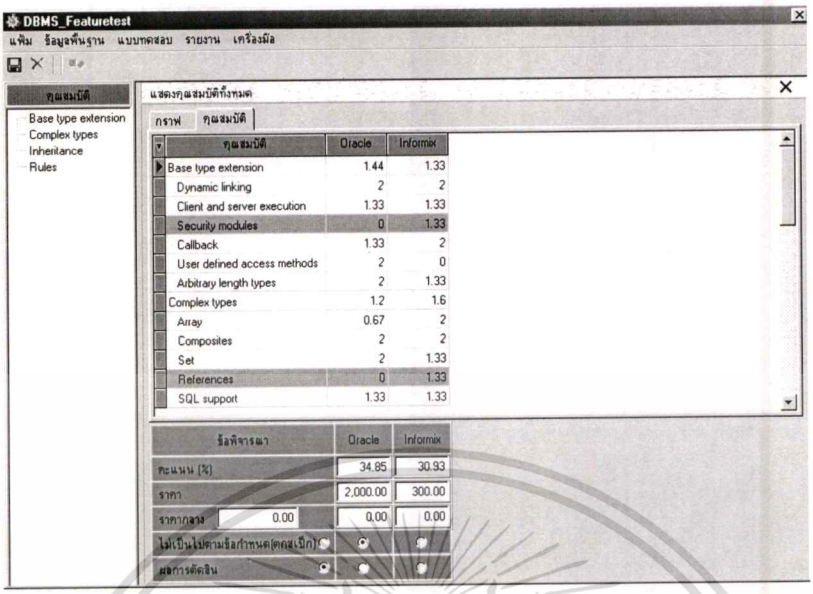


รูปที่ 6.3 แสดงจอภาพสำหรับการให้ความสนใจเฉพาะงาน

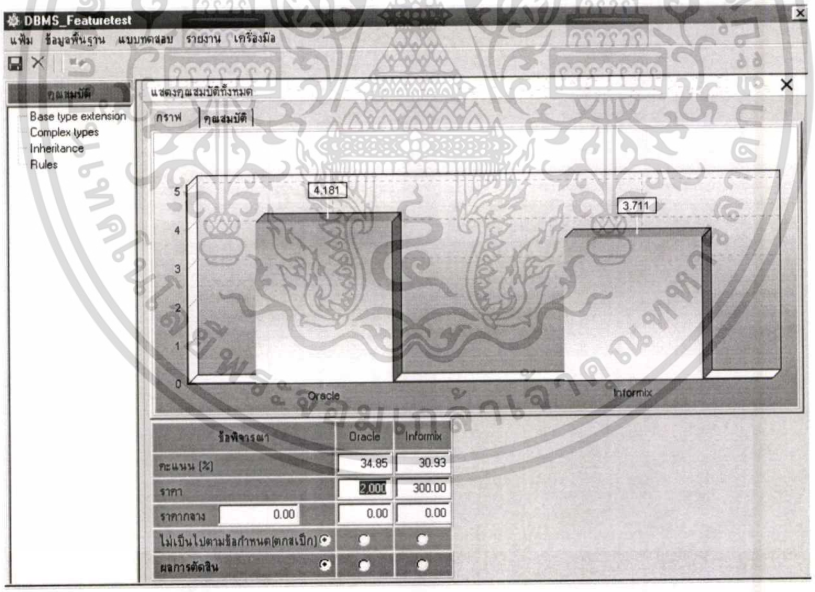
ผู้ใช้กลุ่มที่ 4: คณะกรรมการเปิดซอง

ทำหน้าที่ตัดสินว่าผลิตภัณฑ์ใดที่ได้รับความสนใจที่จะนำมาใช้งาน สำหรับข้อมูลประกอบการพิจารณาประกอบด้วยดังต่อไปนี้

- คะแนนรวมในแต่ละคุณสมบัติ แยกตามผลิตภัณฑ์ที่สนใจ
- แสดงคุณสมบัติที่ไม่ตรงตามข้อกำหนด
- เปรียบเทียบราคาของแต่ละผลิตภัณฑ์ โดยเทียบเคียงกับราคากลาง
- เลือกผลิตภัณฑ์เพียงตัวใดตัวหนึ่ง หรือยังไม่เลือกก็ได้ (ถ้ายังไม่ได้ตัดสินใจ)
- แสดงผลลัพธ์ในรูปแบบของกราฟได้ โดยแยกสีตามสีของผลิตภัณฑ์ (โปรแกรมกำหนด)



รูปที่ 6.4 แสดงจอภาพสำหรับการตัดสินใจเลือกผลิตภัณฑ์



รูปที่ 6.5 แสดงจอภาพสำหรับการตัดสินใจเลือกผลิตภัณฑ์แบบกราฟ

ประวัติผู้เขียน

ชื่อ-สกุล	นายถิรภัทร มีสำราญ
สถานที่เกิด	นครราชสีมา
ประวัติการศึกษา	วท.บ. (วิทยาการคอมพิวเตอร์)
	สถาบันราชภัฏนครราชสีมา
ประวัติการทำงาน	สถาบันราชภัฏนครสวรรค์