การพัฒนาระบบต้นแบบของแอพพลิเคชั่นฐานข้อมูลเชิงสัมพันธ์แบบไคลเอนท์เซิร์ฟเวอร์

(Client/Server Relational Database System Development)

โดย

นางสาว พัชรินทร์ มีไพบูลย์ 35104292

นางสาว พิมพ์ภัค แสนสุทธิ์ 35104301

อาจารย์ที่ปรึกษา

ดร.วรวัฒน์ ลิ้มโภคา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์

สาขาวิชาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

ปริญญานิพนธ์ปีการศึกษา 2538

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาระบบต้นแบบของแอพพลิเคชั่นฐานข้อมูลเชิงสัมพันธ์แบบไคลเอนท์เซิร์ฟเวอร์

ผู้จัดทำ

1. นางสาว พัชรินทร์ มีไพบูลย์ 35104292
2. นางสาว พิมพ์ภัค แสนสุทธิ์ 35104301

_____ อาจารย์ที่ปรึกษา

(ดร.วรวัฒน์ ลิ้มโภคา)

การพัฒนาระบบต้นแบบของแอพพลิเคชั่นฐานข้อมูลรีเลชั่นแนลแบบไคลเอ็นท์เซิร์ฟเวอร์

พัชรินทร์ มีไพบูลย์
พิมพ์ภัค แสนสุทธิ์
ดร. วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา
ปีการศึกษา 2538

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เรียบเรียงจากขั้นตอนการทำงานทั้งหมดในการสร้างระบบงานประยุกต์เพื่อใช้ในการจัดการระบบบริหารงานบุคคลและจัดการทรัพยากรมนุษย์โดยพัฒนาขึ้นเป็นระบบต้นแบบ เริ่มจากศึกษาระบบงานที่มีอยู่เดิมและกำหนดขั้นตอนการออกแบบโดยอ้างอิงกับ ทฤษฎีของ Structured Systems Analysis and Design Method (SSADM) บ้างบางส่วน เนื่องจากเป็นวิธีที่ทันสมัย สำหรับผลิตภัณฑ์ที่นำมาเป็นเครื่องมือในการออกแบบและพัฒนาระบบงานประยุกต์ในครั้งนี้ได้ใช้ผลิตภัณฑ์ ERwin/ERX 2.0 ของบริษัท Logic Works ในการสร้างตารางเก็บข้อมูลที่มีความสัมพันธ์ของตารางคล้ายกับ ER model และ Delphi ของบริษัท Borland International inc. ในการสร้างระบบงานประยุกต์ต่างๆโดยนำหลักการวิเคราะห์และออกแบบโปรแกรมแบบออปเจค - โอเรียนเต็ดมาใช้ โดยได้พัฒนาบนฐานข้อมูลแบบเดสค์ทอปก่อนจากนั้นจึงพัฒนาเพื่อเป็นระบบไคล์เอ็นท์เซิร์ฟเวอร์โดยได้ใช้ ตัวจัดการฐานข้อมูลคือ SQL Server ที่มี Windows NT เป็นตัวปฏิบัติการ

CLIENT/SERVER RELATIONAL DATABASE SYSTEM DEVELOPMENT

Patcharin Meepaibul
Pimpuk Sansuth
Dr. Vorawat Limpoka
1995

Abstract

This thesis is based on the development of Personnel Administration and Human Resource Management system to be a prototype. First , is to study an existing application and assign design by reference in partial Structure System Analysis AND Design Method (SSADM) because of being a new technology method. And by means of Object-Oriented Methodology to design and implement an application on a client/server database. The product used in a design and an implementation is ERwin/ERx 2.0 from Logic Works and Delphi from Borland international Inc. . At the first time,the application is implemented on the desktop database and the will be upsized to the SQL Server .

สารบัญ

สารบัญรูปภาพ

สารบัญตาราง

บทที่ 1
บทนำ

ปริญญานิพนธ์ฉบับนี้เขียนขึ้นเพื่อประกอบโครงงานของนักศึกษาชั้นปีที่ 4 ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง ซึ่งในโครงงานนี้เป็นการพัฒนาระบบต้นแบบของแอพพลิเคชั่นฐานข้อมูลแบบไคลเอนท์เซิร์ฟเวอร์ รายละเอียดในปริญญานิพนธ์นี้ประกอบไปด้วย

บทที่ 2 เป็นรายละเอียดเกี่ยวกับทฤษฎีที่ใช้ในการพัฒนาระบบ ได้แก่ ระเบียบวิธีเอสเอสเอดีเอ็ม, โมเดลเชิงสัมพันธ์, โมเดล ER, ผังการไหลของข้อมูล, หลักการของไคล์เอนท์เซิร์ฟเวอร์ ,หลักการของออปเจค-โอเรียนเต็ด และการสนับสนุนของ Delphi ในการพัฒนาแอพพลิเคชั่นฐานข้อมูล

บทที่ 3 เป็นรายละเอียดของผลที่ได้จากการนำเอาทฤษฎีในบทที่ 2 มาใช้ในการวิเคราะห์และออกแบบระบบ ซึ่งได้แก่ โมเดล ER , ผังการไหลของข้อมูล และออปเจค-โอเรียนเต็ดโมเดล

บทที่ 4 เป็นผลที่ได้จากการพัฒนาระบบด้วย Delphi ซึ่งจะเป็นรายละเอียดเกี่ยวกับระบบงานในส่วนหลักๆ และหน้าจอที่ได้

บทที่ 5 เป็นแนวทางในการพัฒนาระบบนี้ต่อไป รวมทั้งปัญหาที่เกิดขึ้น

บทที่ 2
ทฤษฎีที่ใช้ในการพัฒนาระบบ

2.1 ระเบียบวิธีเอสเอสเอดีเอ็ม (SSADM)

เอสเอสเอดีเอ็ม เป็นหนึ่งในวิธี*โครงสร้าง(Structure Method)*ที่ถูกใช้อย่างกว้างขวางและสมบูรณ์ที่สุดวิธีหนึ่ง

2.1.1 คุณลักษณะพิเศษของวิธีโครงสร้าง

- เป็นวิธีดำเนินงานที่จะจัดองค์ประกอบของโปรเจ็คท์ให้เป็นกิจกรรมต่าง ๆ (Activities) ที่ถูกระบุชัดเจนและแยกย่อย และระบุขั้นตอนและบทบาทระหว่างกันของกิจกรรมเหล่านั้น
- เป็นวิธีที่ใช้แผนผังและแผนภาพต่าง ๆ เพื่อช่วยอธิบายให้ชัดเจนยิ่งขึ้นซึ่งสามารถเข้าทั้งผู้ใช้และผู้พัฒนา

2.1.2 หลักการเบื้องต้นของเอสเอสเอดีเอ็ม

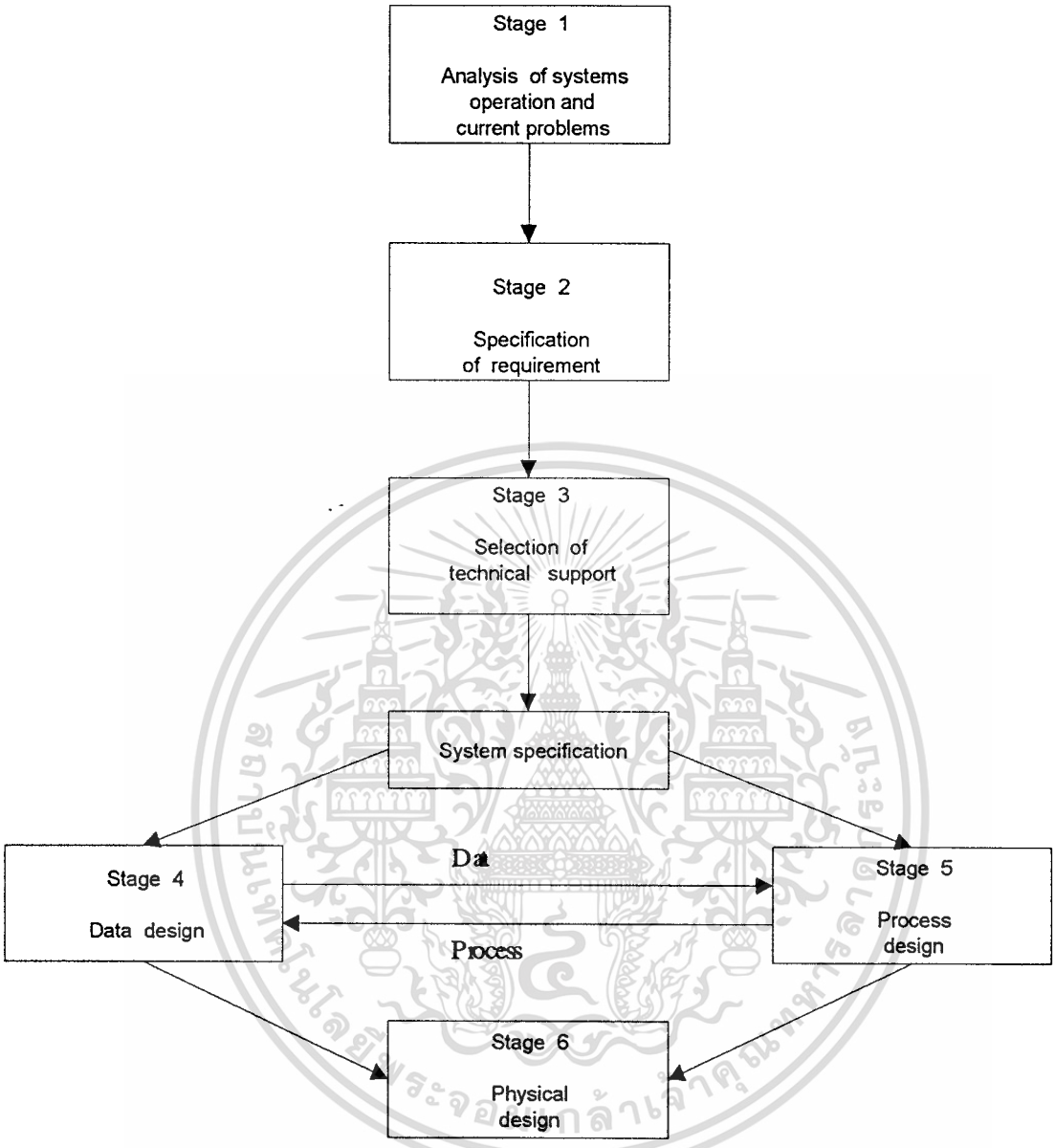
ไอสเอสเอดีเอ็ม เป็นวิธีการแบบอาศัยการเคลื่อนที่ของข้อมูล (Data-Driven) โดยมีสมมุติฐานเบื้องต้นว่าระบบจะมีโครงสร้างข้อมูล(Data Structure) มูลฐานทั่วไปซึ่งจะมีการเปลี่ยนแปลงน้อยมาก แม้ว่าความต้องการทางกรรมวิธี (Processing Requirement) จะเปลี่ยนแปลง

เทคนิคโครงสร้าง(Structured techniques) ของเอสเอสเอดีเอ็ม จะสร้างกรอบงาน (frame work) ของขั้นตอน(step) และระดับ (stage) โดยแต่ละอันได้ถูกกำหนดข้อมูลและข้อมูลออก โดยจะมีแบบฟอร์ม และเอกสารประกอบเพิ่มเติมข้อมูลที่อยู่ในแผนภาพ ดังนั้น เอสเอสเอดีเอ็มจะประกอบไปด้วยคุณลักษณะที่สำคัญ 3 ประการดังนี้

- Structure กำหนดกรอบงานของขั้นตอนและระดับ และข้อมูลเข้าและออกของมัน
- Techniques กำหนดว่าขั้นตอนและงานต่างทำอย่างไร
- Document กำหนดว่าผลงานของแต่ละขั้นตอนถูกแสดงอย่างไร

2.1.3 โครงสร้างของเอสเอสเอดีเอ็ม

จากรูปที่ 1 แสดงถึงขั้นตอนการดำเนินงานตามวิธีเอสเอสเอดีเอ็ม โดยอาจมีการเริ่มต้นด้วยเฟสการ ศึกษาความเป็นไปได้ของทางเลือกต่าง ๆ ก่อนก็ได้ แต่ละขั้นตอนระดับจะถูกแยกย่อยเป็นขั้นตอนต่าง ๆ ซึ่งกำหนด ข้อมูลเข้าและข้อมูลออก และงานที่ต้องกระทำ ผลลัพธ์ของแต่ละขั้นตอนและการติดต่อระหว่างขั้นตอนย่อยจะถูกระบุอย่างชัดเจน



รูปที่ 2-1 แผนภาพแสดงระดับขั้นตอนของเอสเอสเอดีเอ็มโปรเจ็คท์

โครงสร้างของวิธีดำเนินการดังรูปแสดงให้เห็นถึงคุณลักษณะพิเศษหลายประการของเอสเอสเอดีเอ็ม

- ระบบปัจจุบัน(Current System) จะถูกศึกษาก่อนเพื่อสร้างความเข้าใจในสภาวะแวดล้อมของระบบใหม่
- วิว(view)ของระบบปัจจุบัน จะถูกใช้ในการสร้างข้อกำหนดคุณลักษณะ (Specification) ของระบบที่ต้องการ แต่อย่างไรก็ตามระบบที่ต้องการจะไม่ผูกติดกับรูปแบบที่ใช้สร้างระบบปัจจุบัน

- ข้อกำหนดคุณลักษณะของสิ่งที่ต้องการจะต้องมีรายละเอียดมากพอที่จะสร้างทางเลือกทางเทคนิคที่ละเอียดได้

การออกแบบในรายละเอียดจะต้องสมบูรณ์ในระบบเชิงตรรกะ(Logical System) ก่อนที่จะพิจารณาถึงเรื่องเกี่ยวกับการสร้างการออกแบบเชิงตรรกะ(Logical Design) จะถูกแปลง (Translate)ไปเป็นการออกแบบเชิงกายภาพ(Physical Design) โดยใช้กูฎเบื้องต้น(Simple Rule) การออกแบบที่ได้จะถูกปรับแต่ง(Tune) โดยเทคนิคของการควบคุมการออกแบบเชิงกายภาพ ก่อนจะลงมือสร้างจริง

ระดับขั้นที่ 1 วิเคราะห์การดำเนินงานในระบบและปัญหาปัจจุบัน

ระบบปัจจุบันจะถูกตรวจสอบวิเคราะห์ด้วยเหตุผลหลายประการดังนี้

- นักวิเคราะห์ระบบจะต้องเรียนรู้ระบบคำศัพท์(Terminology) และการทำงานในสภาวะแวดล้อมของผู้ใช้
- ระบบเดิมอาจจะนำมาใช้สร้างรูปแบบพื้นฐานของระบบใหม่
- ข้อมูลที่ต้องการในระบบอาจได้จากการตรวจสอบวิเคราะห์ระบบปัจจุบัน
- ทำให้ผู้ใช้ได้รับการแนะนำเทคนิคต่าง ๆที่ดี
- ขอบเขตของการตรวจสอบวิเคราะห์จะได้ถูกกำหนดไว้อย่างชัดเจน

เหตุผลที่ 3 แสดงให้เห็นถึงหลักการสำคัญข้อหนึ่งของเอสเอสเอดีเอ็มที่ว่า โครงสร้างมูลฐานของข้อมูลในระบบจะไม่ถูกเปลี่ยนแปลงอยู่ตลอดเวลา แม้ว่าในระบบใหม่อาจมีการเปลี่ยนแปลงหน้าที่การทำงาน ข้อมูลมูลฐานที่ใช้ในการทำงานจะไม่เปลี่ยนแปลงมากนัก

ถ้าปรากฏว่าไม่มีระบบปัจจุบัน ขั้นตอนนี้จะเป็นเพียงการนำเสนอโครงงานในขั้นต้นและการเริ่มสร้างเอกสารข้อกำหนดต่าง ๆ ขึ้นมาใหม่

ระดับขั้นที่ 2 การกำหนดคุณลักษณะของสิ้นที่ต้องการ

เริ่มต้นด้วยการสร้างวิวเชิงตรรกะของระบบปัจจุบัน เพื่อจะสนใจว่ามีมีฟังชั่นการทำงานใดบ้างในระบบปัจจุบัน และตัดสินใจว่าอะไรบ้างที่ควรจะคงมีอยู่ในระบบใหม่

ความแตกต่างของระบบทั้งสองอยู่ที่ ทางเลือกทางธุรกิจของระบบ (Business System Option) โดยใช้ทางเลือกทางธุรกิจของระบบที่ได้เลือกไว้เป็นพื้นฐาน ข้อกำหนดในรายละเอียดของระบบที่ต้องการ จะถูกสร้างขึ้นและถูกตรวจสอบอย่างละเอียด

ระดับขั้นที่ 3 การคัดเลือกเทคนิคในการสร้าง

ในขั้นตอนนี้ ทีมพัฒนาจะรวบรวมข้อมูลที่ได้เพื่อประมวลผลทางเลือกในการสร้างที่ต่างกันหลายทางสำหรับระบบ แต่ละทางเลือกจะมีค่าใช้จ่ายและผลประโยชน์(Cost and Benefit) ประเมินไว้ เพื่อช่วยผู้ใช้ในการตัดสินใจเลือก และอาจจะมีการเลือกฮาร์ดแวร์ของระบบด้วย

ระดับขั้นที่ 4 การออกแบบข้อมูลเชิงตรรกะ

ในขั้นตอนนี้จะทำการออกแบบข้อมูล(ทางตรรกะ) ซึ่งเป็นข้อมูลที่ต้องการทั้งหมด โดยอาศัยเทคนิควิเคราห์เชิงสัมพันธ์ (Relational Analysis Technique) เพื่อจัดกลุ่มของดาต้าไอเท็ม (Data Item) ในระบบเพื่อใช้ในการตรวจสอบสองทาง (cross-check) กับนิยามข้อมูล (Data Definition) ที่สร้างในระดับขั้นที่ 2

การออกแบบข้อมูลขั้นสุดท้ายจะถูกตรวจสอบกับกรรมวิธี(ทางตรรกะ)ที่พัฒนาในระดับขั้นที่ 5 เพื่อให้มั่นใจว่าข้อมูลที่ต้องการใช้ในกรรมวิธีต่าง ๆ อยู่ครบในขั้นตอนการออกแบบข้อมูล

ระดับขั้นที่ 5 การออกแบบกรรมวิธีเชิงตรรกะ

ข้อกำหนดที่สร้างในระดับขั้นที่ 2 จะถูกขยายความให้มีรายละเอียดในระดับสูงขึ้น เพื่อที่ว่าผู้สร้างจะได้รับรายละเอียดที่จำเป็นในการสร้างระบบ นิยามกรรมวิธี(Process Definition) จะถูกตรวจสอบกับนิยามข้อมูลที่ได้จากระดับขั้นที่ 4

ระดับขั้นที่ 6 การออกแบบเชิงกายภาพ

เมื่อการออกแบบเชิงตรรกะเสร็จสมบูรณ์ทั้งข้อมูลและกรรมวิธี ก็จะถูกแปลงไปเป็นแบบที่จะทำงานบนสภาวะแวดล้อมเป้าหมาย การออกแบบเชิงกายภาพในขั้นต้นจะมีการปรับแต่งในกระดาษก่อนที่จะลงมือสร้าง เพื่อที่ว่าจะได้ตรงตามความต้องการด้านประสิทธิภาพ(Performance Requirement) ของระบบ

ในขั้นตอนนี้จะต้องการใช้เอกสารจำนวนมาก ในการสร้างและแปลงการออกแบบจากเชิงตรรกะไปเป็นเชิงกายภาพ

2.1.4 เทคนิคโครงสร้างที่ได้เลือกใช้ในการทำโครงงานนี้

เทคนิคของเอสเอสเอดีเอ็มจะสร้างมาตรฐานในการที่จะบอกว่าแต่ละขั้นตอนและแต่ละงานต้องทำอย่างไรบ้าง กฎไวยากรณ์(Rules of Syntax) และการใช้เครื่องหมาย(Notation) จะเสริมด้วยข้อแนะนำว่าควรจะนำมาใช้กับขั้นตอนนั้นอย่างไร เทคนิคแผนภาพ(Diagrammatic Technique) ของ เอสเอสเอดีเอ็มทำนำมาใช้ในโครงงานนี้ประกอบไปด้วย

- แผนภาพแสดงการไหลของข้อมูล(Data Flow Diagram) หรือ ดีเอฟดี(DFD)
- แผนภาพแสดงโครงสร้างข้อมูลเชิงตรรกะ(Logical Data Design) หรือ แอลดีเอส(LDS)

2.1.5 ความสัมพันธ์ระหว่าง ดีเอฟดี กับ แอลดีเอส

แอลดีเอสจะสะท้อนภาพของโครงสร้างข้อมูลที่ถูกเก็บ ส่วนดีเอฟดีจะแสดงให้เห็นการเคลื่อนที่ของข้อมูลระหว่างระบบกับดาต้าสตอร์(Data Store)

กฎที่ใช้ควบคุมความสัมพันธ์ระหว่าง ดีเอฟดี กับ แอลดีเอสมีดังนี้

- แต่ละดาต้าสตอร์ควรจะเป็นตัวแทนของเอ็นติตี้ทั้งเอ็นติตี้

เนื่องจากทั้งเอนติตี้และดาต้าสตอร์ ต่างก็ใช้แสดงข้อมูลที่ถูกเก็บ จึงมีความสัมพันธ์กันอย่างใกล้ชิดมาก และเนื่องจากเอนติตี้คือ กลุ่มของดาต้าไอเท็ม ที่มีความสัมพันธ์กันจึงไม่มีเหตุผลสมควรที่จะอยู่แยกกันคนละดาต้าสตอร์ และเช่นเดียวกันเนื่องจากแอลดีเอสเป็นวิวของข้อมูลที่มีรายละเอียดมากกว่าจึงเป็นไปไม่ได้ว่า ดาต้าสตอร์จะแสดงกลุ่มของข้อมูลที่มีขนาดเล็กกว่า 1 เอนติตี้

โดยมีกฎอธิบายดังน

- ◆ ดาต้าสตอร์ จะต้องเกี่ยวข้องกับ เอนติตี้ มากกว่าหรือเท่ากับหนึ่ง
- ◆ เอนติตี้ใด ๆ จะปรากฏอยู่ในดาต้าสตอร์เพียงดาต้าสตอร์เดียว แต่มีข้อยกเว้นดังนี้
- ◆ ถ้าในระบบปัจจุบันมักใช้ข้อมูลนั้นแบบซ้ำซ้อน(Duplicate) ดังนั้นอาจเป็นไปได้ว่า เอนติตี้หนึ่งอาจอยู่บนหลายดาต้าสตอร์
- ◆ ดาต้าสตอร์ที่ไม่เสถียร (Transient data store) จะไม่แสดงข้อมูลที่ถูกเก็บ และจะไม่สัมพันธ์กับเอนติตี้ ในแอลดีเอส

- ดาต้าไอเท็มใน ดีเอฟดีจะต้องเป็นของ เอนติตี้อันใดอันหนึ่ง

อีกความสัมพันธ์หนึ่งระหว่าง ดีเอฟดี กับ แอลดีเอส คือความสัมพันธ์ระหว่าง ดาต้าไอเท็มที่เคลื่อนที่ในระบบกับที่เป็นของเอนติตี้ ซึ่งสามารถใช้ในการตรวจสอบข้อมูลที่เข้าและออกดาต้าสตอร์เพื่อให้แน่ใจว่า ดาต้าไอเท็มที่ระบุถึงข้อมูลที่ไหลอยู่ ปรากฏอยู่ในคำอธิบายเอนติตี้ที่เกี่ยวข้องกับดาต้าสตอร์

ดาต้าไอเท็ม ที่ไม่สมควรปรากฏใน แอลดีเอสคือ

- ◆ ข้อมูลไม่เสถียร เช่นเป็นผลลัพธ์ระหว่างทาง
- ◆ ข้อมูลที่คำนวณได้มาเป็นข้อมูลออกของระบบและข้อมูลที่เคลื่อนที่บางตัวอาจจะมีแค่เลเบล(Label) เช่นข้อผิดพลาด (Error) ซึ่งไม่มีความสัมพันธ์กับวิวของแอลเอสดี

2.2 โมเดลเชิงสัมพันธ์ (Relational Model)

โมเดลเชิงสัมพันธ์เป็นโมเดลที่ใช้ในการอธิบายความสัมพันธ์ของข้อมูลที่ถูกเก็บด้วยระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (Relational Database Management System : RDBMS) ซึ่งเป็นผลงานของ ดร.คอดด์ (Codd) ที่ได้เสนอผลงานวิจัยให้ชาวโลกได้รู้จักในปี พ.ศ. 2513 โดยมีบรรดานักวิชาการทางคอมพิวเตอร์ก็ได้ให้ความสนใจและทุ่มเททำการวิจัยเกี่ยวกับโมเดลนี้มากมาย จนในปัจจุบันได้แพร่หลายไปมาก มีการนำไปใช้งานกับเครื่องระดับตั้งแต่เมนเฟรมลงไปจนถึงเครื่องระดับไมโครด้วย และก็เป็นที่ยอมรับกันแล้วว่า บรรดาผู้ใช้ระบบฐานข้อมูล (โดยเฉพาะผู้ที่ทำงานด้วยเครื่องระดับมินิ และระดับไมโคร) จะมีความคุ้นเคยกับโมเดลเชิงสัมพันธ์นี้มากกว่าอีก 2 โมเดล คือ โมเดลเชิงแตกสาขา (Hierarchical model) และโมเดลเชิงโครงข่าย (Network model) ที่มีมาก่อนหน้านี้

นอกเหนือจากความแพร่หลายของโมเดลเชิงสัมพันธ์นี้แล้ว ข้อดีของโมเดลเชิงสัมพันธ์ที่มีมากกว่าอีก 2 โมเดล ดังนี้

1. โมเดลเชิงสัมพันธ์เป็นโมเดลที่สามารถสร้างความเข้าใจได้ง่ายกว่า เพราะภาพลักษณ์ของข้อมูลที่เก็บโดยโมเดลเชิงสัมพันธ์จะมาจากมุมมองของผู้ใช้ ซึ่งจะมีความซับซ้อนน้อยกว่าภาพลักษณ์ของข้อมูลที่เก็บโดยอีก 2 โมเดล

2. ระบบส่วนใหญ่ที่ใช้โมเดลเชิงสัมพันธ์นี้มักจะมีเครื่องมือที่ช่วยให้ผู้ใช้สามารถจัดการกับข้อมูลที่เก็บอยู่ได้ง่ายกว่าข้อมูลที่จัดเก็บด้วยโมเดลแบบอื่น

3. โมเดลเชิงสัมพันธ์นี้มีเครื่องมือที่ช่วยให้ผู้ใช้สามารถค้นพบปัญหาที่เกิดขึ้นในการออกแบบระบบฐานข้อมูลได้ง่าย และยังง่ายต่อการแก้ไขการออกแบบที่ผิดพลาดนั้นด้วย

4. โมเดลเชิงสัมพันธ์เป็นโมเดลที่มีความสอดคล้องกับหลักการของฐานข้อมูล ผู้ใช้ไม่ต้องพะวงกับรายละเอียดของการจัดเก็บข้อมูลเหมือนกับการจัดข้อมูลของโมเดลอื่น

5. ภาษาที่ใช้ในการจัดการกับข้อมูลที่จัดเก็บด้วยระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (ภาษา SQL : Structure Query Language) เป็นภาษาแบบ set oriented ซึ่งจะต่างกับภาษาที่ใช้ในการจัดการกับข้อมูลที่จัดเก็บด้วยระบบจัดการฐานข้อมูลของโมเดลอื่นที่เป็นภาษาแบบ record-at-a-time

แม้ว่าโมเดลเชิงสัมพันธ์จะมีข้อดีหลายประการดังได้กล่าวไปแล้ว แต่ในปัจจุบันก็ยังมีจุดอ่อนที่มีการอ้างอิงถึงเสมอ คือ ระบบจัดการฐานข้อมูลแบบโมเดลเชิงสัมพันธ์นี้มักจะมีประสิทธิภาพในการใช้งานสู้อีก 2 โมเดลไม่ได้ โดยเฉพาะในการประยุกต์ใช้งานขององค์กรขนาดใหญ่ จุดอ่อนนี้ก็ได้มีการแย้งกลับมาในแง่ที่ว่า โมเดลเชิงสัมพันธ์เป็นโมเดลที่มีอายุการพัฒนาน้อยกว่าอีก 2 โมเดล จึงเป็นไปได้ว่า การพัฒนาที่ผ่านมาของโมเดลเชิงสัมพันธ์ก็ยังมีจำนวนระดับขั้นที่ได้พัฒนาไปแล้วน้อยกว่าอีก 2 โมเดล ดังนั้นหากต้องการเปรียบเทียบการทำงานระหว่างโมเดลเชิง

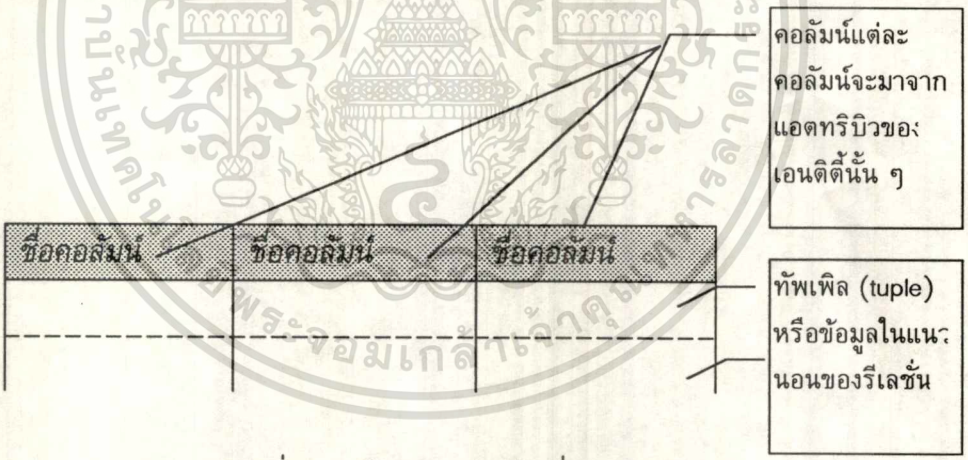
สัมพันธ์กับโมเดลอื่นก็ควรที่จะทำการเปรียบเทียบที่ระดับจำนวนขั้นการพัฒนาที่เท่ากันจึงจะสม
เหตุสมผล

2.2.1 ศัพท์เฉพาะของโมเดลเชิงสัมพันธ์

ในหัวข้อนี้จะกล่าวถึงโมเดลเชิงสัมพันธ์ โดยกำหนดนิยามและกล่าวถึงคำศัพท์ต่าง ๆ ที่
เกี่ยวข้องกับโมเดลนี้

จากการที่ข้อมูลที่เก็บด้วยโมเดลเชิงสัมพันธ์จะถูกเก็บไว้ในตารางที่จะถูกเรียกว่า
"*รีเลชั่น*" โดยที่รีเลชั่นทุกรีเลชั่นจะอยู่ในรูปของตาราง แต่ตารางบางตารางอาจไม่เป็นรีเลชั่นก็ได้
ดังนี้ตารางที่มีลักษณะเป็นรีเลชั่นจะต้องมีคุณลักษณะดังนี้

1. แต่ละช่องของตารางจะบรรจุข้อมูลได้เพียงค่าเดียว
2. ชื่อหัวข้อในแต่ละคอลัมน์มีความแตกต่างกัน อันเป็นชื่อของแอตทริบิวท์ของเอนติตี้
3. ค่าข้อมูลที่อยู่ในแต่ละคอลัมน์ คือ ค่าของแอตทริบิวท์ตามที่ระบุหัวข้อไว้ที่หัวของ
คอลัมน์นั้น ๆ
4. การเรียงลำดับคอลัมน์ไม่ถือว่ามีความสำคัญ
5. ข้อมูลแต่ละแถวจะต้องแตกต่างกัน
6. การเรียงลำดับแถวไม่ถือว่ามีความสำคัญ



รูปที่ 2-2 ลักษณะของรีเลชั่น

ตารางที่มีคุณลักษณะดังกล่าวจะเรียกว่า *รีเลชั่น*

ดังนั้น เราจะได้นิยามของ*ฐานข้อมูลเชิงสัมพันธ์* คือ ฐานข้อมูลที่เกิดจากการรวบรวม
รีเลชั่นต่าง ๆ ที่มีความสัมพันธ์ (relationship)ระหว่างกัน

เราจะเรียกข้อมูลแต่ละแถวในแนวนอนของรีเลชั่นว่า *ทัพเพิล (tuple)* และเรียกข้อมูลแต่
ละแถวในแนวตั้งหรือแนวคอลัมน์ว่า *แอตทริบิว (attribute)* โดยที่คำว่า *คีย์ (key)* จะหมายถึง
ข้อมูลที่เกิดจากแอตทริบิวท์ 1 ตัวหรือหลายตัวก็ได้

แต่ละรีเลชั่นจะต้องมีสิ่งที่เรียกว่า *คีย์หลัก (primary key)* คือ ข้อมูลของแอตทริบิวท์
ตัวหรือมากกว่า 1 ตัวก็ได้ที่สามารถใช้เป็นตัวเจาะจงบอกเราได้ว่ากำลังอ้างอิงถึงข้อมูลทัพเพิลใด

ส่วนคีย์ที่เป็นแอตทริบิวท์ของรีเลชั่นอื่นที่ซ้ำกับแอตทริบิวท์ที่เป็นคีย์หลักของรีเลชั่นหนึ่งจะเรียกว่า *คีย์นอก (foreign key)* เช่น รีเลชั่น A มีแอตทริบิวท์รหัสประจำตัวเป็นคีย์หลัก แล้วในรีเลชั่น B มีแอตทริบิวท์รหัสประจำตัวเช่นเดียวกับ A เราจะเรียก แอตทริบิวท์ดังกล่าวของรีเลชั่น B ว่าเป็นคีย์นอกของคีย์หลักของรีเลชั่น A ในกรณีที่รีเลชั่นมีแอตทริบิวท์หรือกลุ่มแอตทริบิวท์ที่มีคุณสมบัติเป็นคีย์หลักได้อยู่หลายแอตทริบิวท์ เราจะเรียกแอตทริบิวท์ที่มิได้ถูกเลือกเป็นคีย์หลักว่า *คีย์คู่แข่ง (candidate key)* หรือ *คีย์สำรอง (alternate key)* และแอตทริบิวท์อื่นๆที่เหลือที่มิได้เป็นคีย์หลักและไม่ได้เป็นส่วนใดส่วนหนึ่งของคีย์หลักก็จะถูกเรียกว่า *นันคีย์ (nonkey attribute)*

คำว่า *โดเมน (domain)* จะหมายถึง กรอบของค่าต่างๆ ที่เป็นไปได้ เช่น โดเมนของแอตทริบิวท์วันที่ ก็จะหมายถึงค่าของวันที่ที่เป็นไปได้ คือ มีค่าเท่ากับ 1 ถึง 31 ในเดือนที่ลงท้ายด้วยคำว่า “คม” และมีค่าเท่ากับ 1 ถึง 30 ในเดือนที่ลงท้ายด้วยคำว่า “ยน” และในเดือนกุมภาพันธ์อาจจะมีค่าเท่ากับ 1 ถึง 28 หรือ 29 ก็ได้ แต่ในการเก็บค่าข้อมูลลงในรีเลชั่นนั้น บางกรณีที่เรามีการกำหนดโดเมนให้กับแอตทริบิวท์แล้ว แต่ข้อมูลที่จะถูกเก็บเข้าไปอาจถูกบรรจุเข้าไปในภายหลัง ลักษณะนี้จะทำให้เกิด *ค่าว่าง (Null value)* ขึ้นชั่วขณะก่อนที่จะมีการบรรจุค่าข้อมูลที่อยู่ในโดเมนที่กำหนดไว้เข้าไป ดังนั้น คำว่า “ค่าว่าง” จึงหมายถึงค่าที่ยังมิทราบแน่ชัดว่าแอตทริบิวท์นั้นจะมีค่าเป็นค่าใด หรือ ค่าของข้อมูลที่ไม่อยู่ในโดเมนที่กำหนด โดยมีข้อบังคับไว้ว่า แอตทริบิวท์ที่ทำหน้าที่เป็นคีย์หลักของรีเลชั่นจะมีค่าข้อมูลเป็นค่าว่างไม่ได้เสมอ เพราะจะทำให้การเข้าถึงข้อมูลในทัพเพิลนั้นกระทำไม่ได้

เมื่อมีการจัดเก็บข้อมูลในฐานข้อมูลใดๆแล้ว ข้อมูลจะถูกแยกออกเป็นกลุ่มของข้อมูลเป็นชุดที่ประกอบด้วยแอตทริบิวท์ต่างๆที่มีความสัมพันธ์กัน เช่น การเก็บข้อมูลของบุคลากรในโรงเรียนก็อาจแยกเก็บเป็นกลุ่มของข้อมูลนักเรียน, กลุ่มข้อมูลของอาจารย์, และกลุ่มข้อมูลของนักการภารโรง เป็นต้น กลุ่มของข้อมูลแต่ละกลุ่มนี้จะเรียกว่า *เอนติตี้ (entity)* ซึ่งแต่ละเอนติตี้จะประกอบไปด้วยแอตทริบิวท์ต่างๆ ที่มีความสัมพันธ์กัน เช่น เอนติตี้ของนักเรียนก็จะประกอบไปด้วยชื่อ, นามสกุล, ที่อยู่, ชั้นเรียน เป็นต้น

จากการแยกจัดเก็บข้อมูลออกเป็นเอนติตี้นี้ แต่ละเอนติตี้ก็จะมีความสัมพันธ์กัน ความสัมพันธ์ระหว่างเอนติตี้สามารถแบ่งออกเป็น 3 ชนิด คือ

- ความสัมพันธ์แบบหนึ่งต่อหนึ่ง (one to one)
- ความสัมพันธ์แบบหนึ่งต่อกลุ่ม (one to many)
- ความสัมพันธ์แบบกลุ่มต่อกลุ่ม (many to many)

ความสัมพันธ์แบบหนึ่งต่อหนึ่ง

ความสัมพันธ์แบบหนึ่งต่อหนึ่งระหว่างเอนติตี้ก็หมายความว่า เมื่อเอนติตี้หนึ่งมีข้อมูลของคีย์หลักค่าหนึ่ง ค่าของข้อมูลดังกล่าวก็จะมีความสัมพันธ์กับค่าของข้อมูลของคีย์หลักของอีก

เอนติตี้หนึ่งเพียงค่าเดียวเท่านั้น เช่น หากเรากำหนดให้ความสัมพันธ์ระหว่างเอนติตี้นักเรียนกับเอนติตี้ผู้ปกครองเป็นแบบหนึ่งต่อหนึ่งแล้วก็จะหมายความว่า การที่เราอ้างถึงนักเรียนคนใดคนหนึ่งก็จะสามารถอ้างถึงผู้ปกครองได้เพียงคนเดียวเท่านั้น และในทางตรงกันข้ามก็ต้องเป็นจริงด้วย คือ เมื่อเราอ้างถึงผู้ปกครองคนใดคนหนึ่งแล้วก็จะสามารถอ้างถึงนักเรียนได้เพียงคนเดียวเท่านั้น

ชื่อนักเรียน	*ชื่อผู้ปกครอง*
A	a
B	b
C	c

รูปที่ 2-3 ความสัมพันธ์แบบหนึ่งต่อหนึ่ง

ความสัมพันธ์แบบหนึ่งต่อกลุ่ม

ความสัมพันธ์แบบหนึ่งต่อกลุ่มระหว่างเอนติตี้ก็หมายความว่า เมื่อเอนติตี้หนึ่งมีข้อมูลของคีย์หลักค่าหนึ่ง ค่าข้อมูลดังกล่าวก็จะมีความสัมพันธ์กับค่าข้อมูลของคีย์หลักของอีกเอนติตี้หนึ่งได้หลายค่า เช่น หากเรากำหนดให้ความสัมพันธ์ระหว่างเอนติตี้นักเรียนกับเอนติตี้ผู้ปกครองเป็นแบบหนึ่งต่อกลุ่มแล้วก็จะหมายความว่า การที่เราอ้างถึงนักเรียนคนใดคนหนึ่งก็จะสามารถอ้างถึงผู้ปกครองได้หลายคน และในทางตรงกันข้ามก็จะหมายความว่า เมื่อเราอ้างถึงผู้ปกครองคนใดคนหนึ่งแล้วก็จะสามารถอ้างอิงถึงนักเรียนได้คนเดียวเท่านั้น แต่ถ้าผู้ปกครองที่เราอ้างถึงเป็นคนละคนกันก็อาจจะอ้างอิงถึงนักเรียนคนเดียวกันก็เป็นได้

ชื่อนักเรียน	*ชื่อผู้ปกครอง*
A	a
B	a
C	c

รูปที่ 2-4 ความสัมพันธ์แบบหนึ่งต่อกลุ่ม

ความสัมพันธ์แบบกลุ่มต่อกลุ่ม

ความสัมพันธ์แบบกลุ่มต่อกลุ่มระหว่างเอนติตี้ก็หมายความว่า ค่าข้อมูลของคีย์หลักของเอนติตี้หนึ่งที่ต่างกันอาจอ้างอิงถึงค่าข้อมูลของคีย์หลักของอีกเอนติตี้หนึ่งได้ค่าเดียวหรือหลายค่าก็ได้ เช่น หากเรากำหนดให้ความสัมพันธ์ระหว่างเอนติตี้นักเรียนและเอนติตี้ผู้ปกครองเป็นแบบ

กลุ่มต่อกลุ่มแล้วก็จะหมายความว่า การที่เราอ้างถึงนักเรียนคนใดคนหนึ่งหรือหลายคนก็จะสามารถอ้างอิงถึงผู้ปกครองคนเดียวกันได้ และในทางกลับกัน การที่เราอ้างถึงผู้ปกครองคนหนึ่งหรือหลายคนก็จะสามารถอ้างถึงนักเรียนคนเดียวกันได้

ชื่อนักเรียน	*ชื่อผู้ปกครอง*
A	a
B	a
C	c
C	d

รูปที่ 2-5 ความสัมพันธ์แบบกลุ่มต่อกลุ่ม

เมื่อเราได้แอตทริบิวท์ต่าง ๆมาประกอบกันเป็นเอนติตี้ และสามารถกำหนดความสัมพันธ์ระหว่างเอนติตี้ได้แล้ว ขั้นตอนต่อไปก็คือการเขียนภาพแสดงความสัมพันธ์ระหว่างเอนติตี้ต่าง ๆด้วยโมเดล ER (Entity-Relationship Model)

2.3 โมเดล ER (Entity-Relationship Model)

โมเดล ER เป็นแผนภาพที่ถูกออกแบบมาเพื่อแสดงความสัมพันธ์ระหว่างเอนติตี้ต่าง ๆในรูปแบบที่เป็นรูปธรรมมากขึ้น เมื่อมีการใช้โมเดล ER แสดงความสัมพันธ์ระหว่างเอนติตี้แล้วก็จะไม่ต้องมีคำอธิบายความสัมพันธ์ใด ๆอีก เพราะโมเดล ER ประกอบไปด้วยสัญลักษณ์ต่าง ๆที่แสดงถึงคุณลักษณะของเอนติตี้และแอตทริบิวท์ได้ในตัวเองแล้ว

หลังจากที่เราสามารถหาความสัมพันธ์ระหว่างข้อมูลต่าง ๆโดยแสดงความสัมพันธ์ของข้อมูลเหล่านั้นด้วยโมเดล ER แล้ว ขั้นตอนต่อไปนี้ก็คือการเปลี่ยนความสัมพันธ์ของข้อมูลที่อยู่ในโมเดล ER ให้อยู่ในรูปของรีเลชั่น หลังจากนั้นก็ทำการปรับปรุงรีเลชั่นที่ได้ให้มีความซ้ำซ้อนน้อยที่สุด ซึ่งการปรับปรุงดังกล่าวก็จะมีทฤษฎีที่ต้องอ้างถึงคือ กฎของความคงสภาพ (Integrity Rule) ,ฟังก์ชั่นการขึ้นต่อกัน(Functional Dependency) และทฤษฎีที่สำคัญที่สุดก็คือ ทฤษฎีการนอร์มัลไลซ์ (Normalization) ดังจะกล่าวต่อไปนี้

2.3.1 การแปลงความสัมพันธ์ของข้อมูลจากโมเดล ER ไปสู่รูปของรีเลชั่น

การแปลงความสัมพันธ์ของข้อมูลจากโมเดล ER ไปสู่ในรูปของรีเลชั่นมีขั้นตอนดังนี้

1. สำหรับแต่ละเอนติตี้ที่ไม่ใช่เอนติตี้แบบอ่อนของโมเดล ER เราจะสร้างเป็นรีเลชั่นโดยมีทุกแอตทริบิวท์ที่เป็นแอตทริบิวท์ธรรมดา (simple attribute) มาประกอบกัน แล้วทำการเลือกแอตทริบิวท์ใดแอตทริบิวท์หนึ่งหรือกลุ่มของแอตทริบิวท์มาทำหน้าที่เป็นคีย์หลักของรีเลชั่น
2. สำหรับแต่ละเอนติตี้แบบอ่อน เราจะสร้างรีเลชั่นที่เกิดจากการรวมกันของแอตทริบิวท์ธรรมดาของเอนติตี้นั้น โดยที่รีเลชั่นนี้จะมีคีย์หลักคือคีย์รวม (combine key)ที่เกิดจากการรวมกันของคีย์หลักของเอนติตี้แบบอ่อน(partial key)กับคีย์หลักของเอนติตี้ที่มันต้องอ้างถึง(ในกรณีนี้คีย์นี้จะเรียกว่าเป็นคีย์นอก)
3. สำหรับความสัมพันธ์ระหว่างเอนติตี้แบบหนึ่งต่อหนึ่ง เราจะสร้างรีเลชั่นจากความสัมพันธ์ดังกล่าวได้ 2 ลักษณะ คือ เลือกคีย์หลักของเอนติตี้ใดเอนติตี้หนึ่งมาเป็นคีย์หลักของรีเลชั่นนี้ แล้วให้คีย์หลักของอีกเอนติตี้หนึ่งมาเป็นคีย์นอกของรีเลชั่นนี้ โดยถ้าความสัมพันธ์นี้มีแอตทริบิวท์ก็ให้นำแอตทริบิวท์เหล่านั้นมารวมอยู่ในรีเลชั่นนี้ด้วย
4. สำหรับความสัมพันธ์ระหว่างเอนติตี้แบบหนึ่งต่อกลุ่ม เราจะสร้างรีเลชั่นจากความสัมพันธ์ดังกล่าวโดยนำเอาคีย์หลักของเอนติตี้ฝั่งที่มีความสัมพันธ์แบบกลุ่มมาเป็นคีย์หลักของรีเลชั่นนี้ แล้วให้นำเอาคีย์หลักของเอนติตี้ฝั่งที่มีความสัมพันธ์แบบหนึ่งมาเป็นคีย์นอกของรีเลชั่นนี้ โดยถ้าความสัมพันธ์นี้มีแอตทริบิวท์ก็ให้นำเอาแอตทริบิวท์เหล่านั้นมารวมอยู่ในรีเลชั่นนี้ด้วย
5. สำหรับความสัมพันธ์ระหว่างเอนติตี้แบบกลุ่มต่อกลุ่ม เราจะสร้างรีเลชั่นจากความสัมพันธ์นี้โดยนำเอาคีย์หลักของทั้งสองเอนติตี้มาประกอบกันเป็นคีย์หลักของรีเลชั่นนี้ โดยถ้าความสัมพันธ์นี้มีแอตทริบิวท์ก็ให้นำเอาแอตทริบิวท์เหล่านั้นมารวมอยู่ในรีเลชั่นนี้ด้วย
6. สำหรับเอนติตี้ใดที่มีแอตทริบิวท์ที่มีค่าข้อมูลแบบหลายค่า (multivalue attribute หรือ repeating group) ก็ให้สร้างรีเลชั่นใหม่โดยมีคีย์หลักของเอนติตี้นั้นรวมกับแอตทริบิวท์ดังกล่าวเป็นคีย์หลักของรีเลชั่นนี้
7. สำหรับความสัมพันธ์ระหว่างเอนติตี้ที่เกิดจากเอนติตี้มากกว่า 2 เอนติตี้ ให้สร้างรีเลชั่นของความสัมพันธ์นี้โดยนำคีย์หลักของทุกเอนติตี้มาประกอบกันเป็นคีย์หลักของเอนติตี้นี้ โดยถ้าความสัมพันธ์นี้มีแอตทริบิวท์ ก็ให้นำแอตทริบิวท์เหล่านั้นมารวมอยู่ในรีเลชั่นนี้ด้วย

2.4 ทฤษฎีของระบบฐานข้อมูลเชิงสัมพันธ์

2.4.1 กฎของความคงสภาพ (Integrity Rule)

กฎของความคงสภาพของโมเดลเชิงสัมพันธ์เป็นทฤษฎีที่ช่วยยืนยันความถูกต้องของความสัมพันธ์ระหว่างข้อมูลว่า รีเลชั่นใดที่เป็นไปตามกฎของความคงสภาพนี้แล้วย่อมจะมีความสัมพันธ์ระหว่างข้อมูลอย่างถูกต้องอยู่ตลอดเวลา ไม่ว่ารีเลชั่นนั้นจะมีการเปลี่ยนแปลงแก้ไขข้อมูลไปในรูปแบบใดก็ตาม

กฎของความคงสภาพมีความหมายอยู่ 2 ลักษณะ คือ *กฎความคงสภาพของเอนติตี้ (entity integrity rule)* และ *กฎความคงสภาพของการอ้างอิง (referential integrity rule)* ดัง

อธิบายได้ดังนี้

1.*กฎความคงสภาพของเอนติตี้* กล่าวว่า

"แอตทริบิวท์ทุกตัวที่เป็นส่วนของคีย์หลักจะต้องไม่เป็นค่าว่าง"

หมายความว่า คีย์หลักของทุกรีเลชั่นจะไม่สามารถเก็บค่าของข้อมูลที่เป็นค่าว่างได้ เหตุผลของข้อกำหนดนี้ก็คือ เพื่อให้การเข้าถึงข้อมูลในทัพเพิลใดๆของรีเลชั่นมีความเป็นไปได้เสมอ เพราะถ้าคีย์หลักของทัพเพิลใดมีค่าข้อมูลเป็นค่าว่างแล้ว ก็จะส่งผลให้การเข้าถึงข้อมูลในทัพเพิลนั้นไม่สามารถกระทำได้อย่างแน่นอน

2.*กฎความคงสภาพของการอ้างอิง* กล่าวว่า

"ถ้าเรามีรีเลชั่น R2 ซึ่งมี FK เป็นคีย์นอกที่อ้างถึงคีย์หลัก PK ในรีเลชั่น R1 สำหรับทุกค่าของ FK ใน R2 จะต้อง

ก. มีค่าเท่ากับค่า PK ในทัพเพิลใดทัพเพิลหนึ่งในรีเลชั่น R1

หรือ

ข. มีค่าของแอตทริบิวท์ทุกตัวใน FK เป็นค่าว่าง"

หมายความว่า แอตทริบิวท์ใดๆ ที่เป็นคีย์หลักของรีเลชั่นหนึ่ง เมื่อมีการนำแอตทริบิวท์นั้นไปเป็นคีย์นอกของอีกรีเลชั่นหนึ่ง การเป็นคีย์นอกของแอตทริบิวท์นั้นจะต้องมีโดเมนเดียวกันกับแอตทริบิวท์ที่เป็นคีย์หลัก ทั้งนี้ก็เพื่อให้การนำรีเลชั่นมาใช้งานร่วมกัน (การนำรีเลชั่นมา join กัน) กระทำได้อย่างถูกต้อง คือ ทุกแอตทริบิวท์ที่เป็นคีย์นอกจะมีต้องมีข้อมูลซ้ำกับข้อมูลของแอตทริบิวท์ที่เป็นคีย์หลักอย่างแน่นอน แต่อาจจะมีค่าข้อมูลของแอตทริบิวท์ที่เป็นคีย์หลักเป็นข้อมูลที่ไม่ได้อยู่ในโดเมนของแอตทริบิวท์ที่เป็นคีย์นอกก็ได้ นั่นคือ โดเมนของคีย์นอกจะต้องเล็กกว่าหรือเท่ากับโดเมนของคีย์หลักเสมอ

รีเลชั่น R1

คีย์หลักของ R1	*คีย์อื่น ๆของ R1*
A	1
B	2
C	3
D	4

รีเลชั่น R2

คีย์หลักของ R2	*คีย์นอกของ R1*
a	A
b	B
c	B
d	C

รูปที่ 2-6 กฎความคงสภาพ

2.4.2 ฟังก์ชั่นการขึ้นต่อกัน (Functional Dependency)

ฟังก์ชั่นการขึ้นต่อกันเป็นข้อกำหนดที่ช่วยให้เราเห็นถึงความสัมพันธ์ของแอตทริบิวท์ต่าง ๆที่อยู่ในรีเลชั่น ทั้งนี้เพราะแอตทริบิวท์ต่าง ๆที่อยู่ในเอนติตี้เดียวกันก็เป็นไปได้ที่แอตทริบิวท์เหล่านั้นจะมีความสัมพันธ์กันเองโดยที่ความสัมพันธ์นี้อาจเกี่ยวข้องหรือไม่เกี่ยวกับความสัมพันธ์ที่มันมีต่อคีย์หลักของเอนติตี้นั้นก็เป็นได้ ซึ่งการที่แอตทริบิวท์เหล่านั้นมีความสัมพันธ์กันเองจะเป็นสิ่งที่เราต้องพิจารณาแยกออกเป็นรีเลชั่นย่อย ๆ เพราะแอตทริบิวท์ของแต่ละรีเลชั่นก็ควรจะมีความสัมพันธ์กับคีย์หลักของรีเลชั่นของตนเองเท่านั้น

กำหนดรีเลชั่น R ถ้ามีแอตทริบิวท์ Y ของ R เป็นฟังก์ชั่นที่ขึ้นต่อแอตทริบิวท์ X ของรีเลชั่น เราสามารถเขียนแทนได้ด้วยสัญลักษณ์

R.X ---> R.Y

อ่านว่า R.X มีฟังก์ชั่นการขึ้นอยู่กับ R.Y

หรือ R.X มีฟังก์ชั่นการเลือก R.Y

หรือ R.Y ขึ้นอยู่กับ R.X

นิยาม R.X มีฟังก์ชั่นการขึ้นอยู่กับ R.Y ก็ต่อเมื่อ ทุกค่าข้อมูลของแอตทริบิวท์ X ใน R จะมีค่าข้อมูลของแอตทริบิวท์ Y ใน R ได้เพียงค่าเดียวเสมอ โดยที่แอตทริบิวท์ X และ Y อาจจะเป็น*คีย์แบบรวม (composit key)* ก็ได้

ตัวอย่างเช่น เรามีรีเลชั่นของนักเรียน ซึ่งประกอบด้วยแอตทริบิวท์ดังนี้ รหัสนักเรียน,ชื่อ, นามสกุล, ที่อยู่ เราสามารถกล่าวได้ว่า ชื่อ ขึ้นอยู่กับ รหัสนักเรียน เพราะในการกำหนดรหัสนักเรียนขึ้นมา 1 ค่า จะทำให้เราเลือกได้ชื่อนักเรียนออกมา 1 ชื่อเสมอ เช่น นักเรียนรหัส 315 ก็จะหมายถึง พิชัย แต่ในทางกลับกันเราไม่สามารถกล่าวได้ว่ารหัสนักเรียน “ขึ้นอยู่กับ” ชื่อ เพราะนักเรียนอาจมีชื่อซ้ำกันได้ เช่น ถ้าเราระบุชื่อนักเรียน “มณี” ก็อาจจะได้รหัสนักเรียนออกมา 2 ค่า เป็นต้น

นิยาม R.X มีฟังก์ชั่นการขึ้นอยู่กับ R.Y อย่างเต็มที่ (R.Y fully functionally dependent on R.X) ก็ต่อเมื่อ R.Y ขึ้นอยู่กับ R.X และไม่ขึ้นอยู่กับข้อมูลบางส่วนของ R.X โดยที่แอตทริบิวท์ X และ Y อาจจะเป็นคีย์แบบรวมก็ได้

2.4.3 การนอร์มัลไลซ์ (Normalization)

การนอร์มัลไลซ์เป็นการออกแบบฐานข้อมูลแบบที่เป็นมาตรฐานที่สุด ออกแบบโดย Codd โดยมีวัตถุประสงค์ของการออกแบบก็เพื่อลดความซ้ำซ้อนของความสัมพันธ์ของข้อมูลให้เหลือน้อยที่สุด (minimum redundancy) ซึ่งตามมาตรฐานปกติจะมีอยู่ 3 ระดับ คือ

1NF (First Normal Form)

2NF (Second Normal Form)

3NF (Third Normal Form)

โดยที่รีเลชั่นใดที่ยังไม่สอดคล้องตามรูปแบบนอร์มัล (normal form) ทั้งสามก็จะต้องมีการแยกรีเลชั่นนั้นๆออกเป็นรีเลชั่นย่อยๆต่อไปอีก (decomposition method)

ต่อมาได้มีการออกแบบเพิ่มเติมอีก 2 ระดับ คือ

4NF (Fourth Normal Form)

5NF (Fifth Normal Form)

หากรีเลชั่นใดมีมาตรฐานถึงรูปแบบนอร์มัลระดับที่ 5 (5NF) แล้วก็จะมั่นใจได้ว่า รีเลชั่นนั้นจะไม่มีความซ้ำซ้อนของความสัมพันธ์ของข้อมูลอย่างแน่นอน

นอกจากนี้ยังมีการออกแบบรูปแบบนอร์มัลเพิ่มเติมระหว่างรูปแบบนอร์มัลที่ 3 (3NF) และรูปแบบนอร์มัลที่ 4 (4NF) โดย Boyce และ Codd ซึ่งมีชื่อว่า Boyce-Codd Normal Form (BCNF) อีกด้วย

รูปแบบนอร์มัลระดับที่ 1

การปรับรีเลชั่นให้อยู่ในรูปแบบนอร์มัลระดับที่ 1 คือการปรับจากรีเลชั่นที่ไม่นอร์มัล (unnormalized relation) เช่น รีเลชั่นที่มีข้อมูลของแอตทริบิวท์บางช่องมีมากกว่า 1 ค่า (มีแอตทริบิวท์ที่มีข้อมูลเป็น repeating group)

นิยาม รีเลชั่นจะอยู่ในรูปนอร์มัลระดับที่ 1 (1NF) ก็ต่อเมื่อโดเมนของแต่ละแอตทริบิวท์ประกอบด้วยข้อมูลที่เป็นหน่วยย่อยที่สุด (A relation is in first normal form (1NF) if and only if all underlying simple domains contain atomic values only.)

สิ่งที่ได้จากการที่รีเลชั่นอยู่ในรูปแบบนอร์มัลระดับที่ 1 ก็คือรีเลชั่นยังคงมีความซ้ำซ้อนของความสัมพันธ์ระหว่างข้อมูลอยู่มากมาย เพราะนิยามของรูปแบบนอร์มัลระดับที่ 1นี้กำหนดเพียงเฉพาะว่า แต่ละแอตทริบิวท์ของรีเลชั่นจะมีโดเมนที่มีสมาชิกเป็นหน่วยที่เล็กที่สุดเท่านั้น มิได้เป็นการลดความซ้ำซ้อนของความสัมพันธ์ระหว่างข้อมูลแต่ประการใด

รูปแบบนอร์มัลระดับที่ 2

นิยาม รีเลชั่นจะอยู่ในรูปแบบนอร์มัลระดับที่ 2 (2NF) ก็ต่อเมื่อรีเลชั่นนั้นอยู่ในรูปแบบนอร์มัลระดับที่ 1 แล้ว และทุกแอตทริบิวที่ไม่เป็นส่วนใดส่วนหนึ่งของคีย์หลักจะต้องขึ้นอยู่กับคีย์หลักของรีเลชั่นอย่างเต็มที่ (A relation is in 2NF if and only if it is in 1NF and every nonkey atrribute fully depends on the primary key)

สิ่งที่ได้จากการที่รีเลชั่นอยู่ในรูปแบบนอร์มัลระดับที่ 2 คือ ข้อมูลของบางแอตทริบิวท์ที่ไม่ใช่คีย์หลักอาจมีความสัมพันธ์กันเองโดยที่ไม่มีความสัมพันธ์กับคีย์หลักเลย ซึ่งความสัมพันธ์ดังกล่าวนี้ถือว่าเป็นความซ้ำซ้อนประการหนึ่งของรีเลชั่นนั้นๆ ที่จะต้องทำการลดรูปด้วยรูปแบบนอร์มัลระดับต่อไป

รูปแบบนอร์มัลระดับที่ 3

นิยาม รีเลชั่นจะอยู่ในรูปแบบนอร์มัลระดับที่ 3 (3NF) ก็ต่อเมื่อรีเลชั่นนั้นอยู่ในรูปแบบนอร์มัลระดับที่ 2 แล้ว และทุกแอตทริบิวท์ที่ไม่เป็นส่วนหนึ่งส่วนใดของคีย์หลักจะต้องไม่เป็นฟังก์ชั่นที่ขึ้นต่อกันเอง (A relation is in 3NF if and only if it is in 2NF and every nonkey attribute is nontransitively dependent on the primary key)

โดยปกติแล้ว สิ่งที่ได้จากการที่รีเลชั่นอยู่ในรูปแบบนอร์มัลระดับที่ 3 คือ รีเลชั่นจะไม่มีความซ้ำซ้อนอีกต่อไปโดยที่จะสอดคล้องกับรูปแบบนอร์มัลระดับที่ 4 และ 5 ด้วย แต่ก็มีรีเลชั่นบางลักษณะที่จะต้องทำให้อยู่ในรูปแบบนอร์มัลของ Boyce-Codd (BCNF) รีเลชั่นดังกล่าวจะมีลักษณะดังนี้

1) เป็นรีเลชั่นที่มีหลายคีย์คู่แข่ง

และ 2) เป็นคีย์คู่แข่งที่เกิดจากการรวมกันของคีย์ย่อย (candidate key เป็น combine key)

และ 3) คีย์ย่อยที่ประกอบขึ้นเป็นคีย์คู่แข่งนั้นมีส่วนซ้อนทับ(overlap)กันอยู่

นิยาม เราเรียกแอตทริบิวท์ (หรือกลุ่มของแอตทริบิวท์) ใดๆ ก็ตามที่สามารถเลือก (determine) แอตทริบิวท์ตัวอื่นๆได้ว่า *ตัวเลือก (determinant)*

รูปแบบนอร์มัลของ Boyce-Codd

นิยาม รีเลชั่นใดๆจะจัดอยู่ในรูปแบบนอร์มัลของ Boyce-Codd เมื่อตัวเลือกทุกตัวเป็นคีย์คู่แข่ง

สำหรับการขจัดตัวเลือกที่ไม่ได้เป็นคีย์คู่แข่งออกไปก็มีวิธีการดังนี้

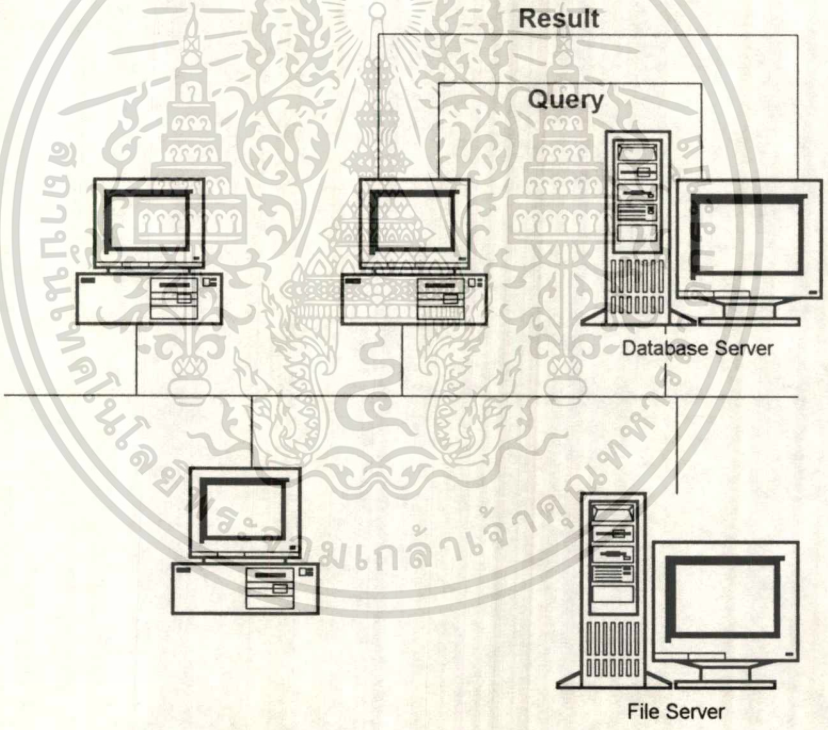
1) ดึงเอาแอตทริบิวท์ที่ขึ้นกับตัวเลือกที่ไม่ใช่คีย์คู่แข่งออกไปสู่รีเลชั่นใหม่

2) กำหนดให้ตัวเลือกที่เกี่ยวข้องเป็นคีย์ของรีเลชั่นนี้

รีเลชั่นส่วนใหญ่เมื่อถูกทำให้อยู่ในรูปแบบนอร์มัลของ Boyce-Codd แล้วก็จะเป็นรีเลชั่นที่ไม่มีความซ้ำซ้อนของความสัมพันธ์ระหว่างข้อมูลหลงเหลืออยู่ ดังนั้นจึงไม่ขอกล่าวถึงรายละเอียดของรูปแบบนอร์มัลระดับที่ 4 และ 5

2.5 หลักการของไคลเอนท์เซิร์ฟเวอร์

ไคลเอนท์เซิร์ฟเวอร์ เป็นโครงสร้างของระบบคอมพิวเตอร์รูปแบบหนึ่งที่แบ่งแยกการประมวลผลข้อมูลออกเป็น 2 ระบบ โดยฝั่งไคลเอนท์ (ผู้ขอใช้บริการ) จะถูกเรียกว่าระบบฟรอนท์เอ็น (Front-end system) หรือส่วนดาต้าเบสแอพพลิเคชั่นทำงานอยู่และฝั่งดาต้าเบสเซิร์ฟเวอร์ (ผู้ให้บริการ) จะถูกเรียกว่าระบบแบคเอ็น (Back-end system) หรือส่วนที่เป็นระบบการจัดการฐานข้อมูลจริง ๆทำงานอยู่ ซึ่งระบบฟรอนท์เอ็นนี้จะจัดการการประมวลผลเกี่ยวกับหน้าจอและอินพุทเอาท์พุทของผู้ใช้และระบบแบคเอ็นจะจัดการการประมวลผลข้อมูลและการทำเข้าถึงดิสค์ เช่นเมื่อผู้ใช้บนระบบฟรอนท์เอ็นสร้างคิวรี่ (query) เพื่อสอบถามข้อมูลจากดาต้าเบสเซิร์ฟเวอร์ ส่วนฟรอนท์เอ็น (Front-end) แอพพลิเคชั่นจะส่งการร้องขอให้เซิร์ฟเวอร์โดยผ่านระบบเครือข่าย ส่วนเซิร์ฟเวอร์ก็จะทำการค้นหาข้อมูลที่ผู้ใช้ต้องการแล้วส่งข้อมูลกลับไปให้ ดังรูปที่ 7



รูปที่2- 7 ระบบไคลเอนท์เซิร์ฟเวอร์

วัตถุประสงค์หลักของระบบไคลเอนท์เซิร์ฟเวอร์คือการอนุญาติให้แอพพลิเคชั่นของผู้ใช้บริการเข้ามาเรียกใช้ข้อมูลที่ถูกจัดการโดยผู้ให้บริการได้ โดยผู้ให้บริการสามารถรันอยู่ในเครื่องที่ตั้งในที่ห่างไกลกับเครื่องที่ผู้ขอใช้บริการรันอยู่

โดยทั่วไปแล้วระบบไคลเอนท์จะถูกใช้ทำงานกับพีซี และส่วนดาต้าเบสเซิร์ฟเวอร์สามารถทำงานบนเครื่องใดก็ได้ตั้งแต่พีซีไปจนถึงเมนเฟรม

2.5.1 ชนิดของการประมวลผลไคลเอนท์เซิร์ฟเวอร์

รูปแบบของแอพพลิเคชั่นของระบบไคลเอนท์เซิร์ฟเวอร์แบ่งได้เป็น 6 ประเภท โดยมีรายละเอียดดังนี้

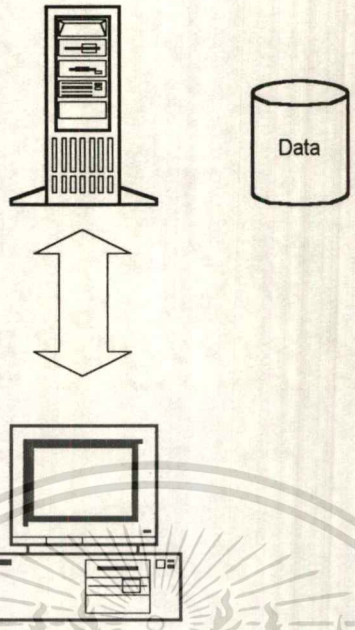
2.5.1.1 ไคลเอนท์เซิร์ฟเวอร์ที่ทำงานบนเครื่องเดียวกัน(Stand-alone Client/Server)

แอพพลิเคชั่นประเภทนี้จะมีผู้ขอใช้บริการประมวลผลอยู่บนเครื่องเดียวกับที่ให้บริการทำการประมวลผลดังในรูปที่ 8 ลักษณะการทำงานเช่นนี้จะเป็นการบั่นทอนประสิทธิภาพการประมวลผลสำหรับระบบจัดการฐานข้อมูลลงบ้าง แต่ความเร็วในการสื่อสารระหว่างผู้ขอใช้บริการกับผู้ให้บริการจะสูงมาก ผู้ให้บริการจะยังสามารถที่จะทำงานได้โดยการประมวลผลร่วมกับแอพพลิเคชั่นอื่น ๆ ของผู้ขอใช้บริการ ในกรณีที่มีผู้ของใช้บริการและผู้ให้บริการหลาย ๆ ตัวรันอยู่บนฮาร์ดแวร์แพลตฟอร์มเดียวกันการใช้มัลติโปรเซสเซอร์อาจจะช่วยเพิ่มประสิทธิภาพการทำงานขึ้นได้ แต่ว่าจะไม่สามารถนำเอาเทคโนโลยีด้านการประมวลผลแบบกระจายหรือ การประมวลฐานข้อมูลแบบกระจายมาใช้ในกรณีนี้ได้เลย

รูปที่ 2-8 ไคลเอนท์เซิร์ฟเวอร์ประเภทที่ 1

2.5.1.2 แสตนอโลนแลนไคลเอนท์เซิร์ฟเวอร์ (Stand-alone LAN Client/Server)

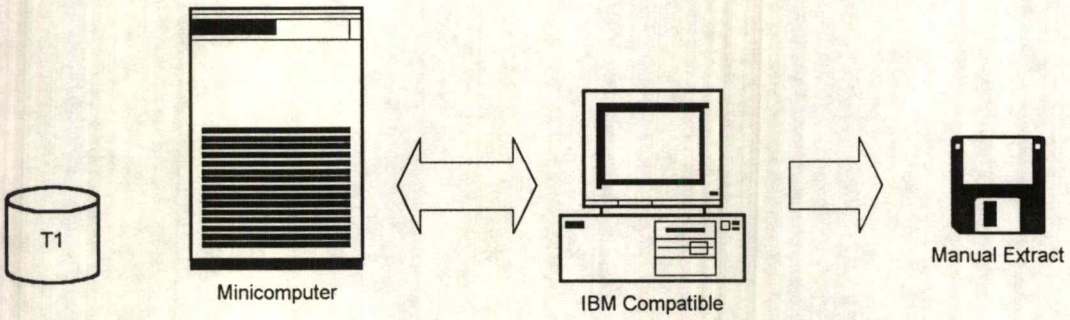
ระบบไคลเอนท์เซิร์ฟเวอร์แบบนี้จะเป็นรูปแบบของไคลเอนท์เซิร์ฟเวอร์ในวงแลนวงหนึ่งมีการทำงานของผู้ขอใช้บริการแต่ละตัวอาจจะรับผิดชอบงานด้านการนำเสนอข้อมูลประมวลผลธุรกิจและลอจิกทางด้านฐานข้อมูลในขณะที่ผู้ให้บริการจะรับผิดชอบในเรื่องของการเรียกใช้ข้อมูลสำหรับผู้ขอใช้บริการภายในวงแลน ข้อเสียของระบบนี้เมื่อเทียบกับระบบในรูปที่ 8 คือการสื่อสารระหว่างผู้ขอใช้บริการกับผู้ให้บริการที่ทำโดยผ่านการเชื่อมต่อของแลนจะช้ากว่าการใช้หน่วยความจำร่วมกันของระบบที่ 1 มาก



รูปที่ 2-9 ไคลเอนท์เซิร์ฟเวอร์ประเภทที่ 2

2.5.1.3 แมนนวลเอ็กแทรกต์ไคลเอนท์เซิร์ฟเวอร์ (Mannual extract Client/Server)

ในรูปที่ 10 จะแสดงให้เห็นถึงรูปแบบของแอพพลิเคชั่นไคลเอนท์เซิร์ฟเวอร์ที่การประมวลผลกระทำได้โดยเรียกใช้ข้อมูลบางส่วนของทั้งหมดที่ได้ทำการย้ายไปเก็บไว้ในเครื่องของผู้ขอใช้บริการ ข้อมูลส่วนนี้ถูกสร้างขึ้นด้วยวิธีการกระจายข้อมูลแบบแมนนวลเอ็กแทรกต์ ลักษณะการทำงานของแอพพลิเคชั่นสามารถเกิดขึ้นโดยผู้ใช้ส่งคำสั่งไปยังผู้ให้บริการเพื่อเรียกใช้ข้อมูล ซึ่งในกรณีนี้มักจะถูกกำหนดให้ทำการอ่านอย่างเดียว การคัดข้อมูลและทำการย้ายนั้นเป็นสิ่งที่สำคัญและจำเป็นต้องทำ เพราะว่าโดยปกติแล้วข้อมูลทั้งหมดมักจะไม่ได้อยู่ในรูปแบบที่ผู้ใช้ต้องการ ตัวอย่างเช่น ผู้ใช้อาจจะต้องการดูข้อมูลสรุป หรือข้อมูลที่ได้ทำการรวบรวมแล้วมากกว่าที่จะดูข้อมูลโดยละเอียด การรวบรวมข้อมูลหรือทำสรุปจะกระทำที่เครื่องของผู้ขอใช้บริการ ข้อเสียอย่างหนึ่งที่อาจจะเกิดขึ้นคือข้อมูลในส่วนที่เก็บอยู่ที่เครื่องของผู้ขอใช้บริการอาจจะไม่ถูกต้องกับความเป็นจริง ถ้าข้อมูลส่วนดังกล่าวกำลังถูกเรียกใช้โดยผู้ขอใช้บริการ และในขณะเดียวกันก็กำลังถูกเปลี่ยนแปลงที่ผู้ให้บริการ

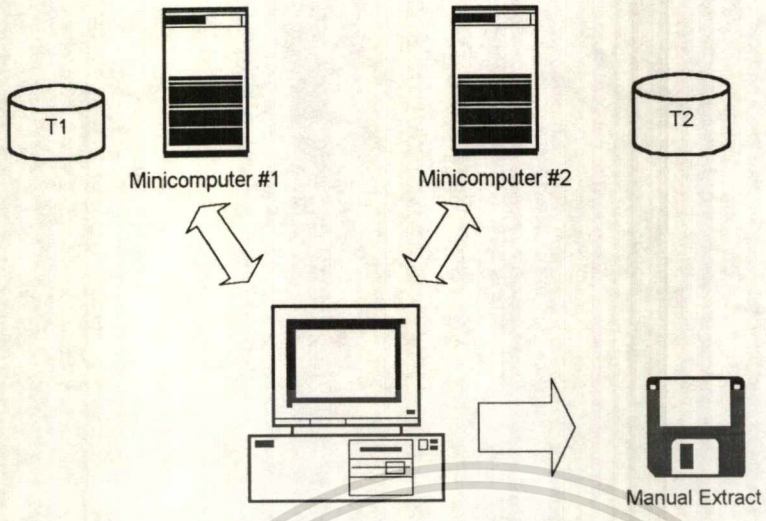


รูปที่ 2-10 ไคลเอนท์เซิร์ฟเวอร์ประเภทที่ 3

2.5.1.4 ซิงเกิลไซต์อัพเดทไคลเอนท์เซิร์ฟเวอร์ (Single-site update Client/Server)

ในรูปที่ 11 จะแสดงลักษณะแอพพลิเคชั่นไคลเอนท์เซิร์ฟเวอร์ประเภทนี้จะมีความสามารถที่สูงขึ้น โดยมันจะสามารถส่งคำสั่งที่ประกอบด้วยคำสั่งหลายคำสั่งส่งไปยังผู้ให้บริการหลาย ๆ ตัวที่อยู่ห่างไกลได้ แต่ข้อมูลที่ทำการเรียกใช้จากผู้ให้บริการแต่ละตัวมักจะไม่มีความสัมพันธ์กัน ทั้งนี้เนื่องจากว่าผู้ให้บริการแต่ละตัวไม่ได้ต่อเชื่อมกันเป็นเครือข่ายเดียวกันและไม่มีผู้ให้บริการตัวใดทำหน้าที่เป็นตัวกลางในการสื่อสารระหว่างผู้ให้บริการโดยการใช้เส้นทางเครือข่ายผ่านทางผู้ขอใช้บริการอีกทอดหนึ่ง (Two phase commit protocol) จากสาเหตุอันนี้ ทำให้การประมวลผลแบบนี้อนุญาตให้ผู้ขอใช้บริการสามารถที่จะทำการแก้ไขข้อมูลที่ผู้ให้บริการได้เพียงตัวเดียวเท่านั้น ถ้าหากข้อมูลที่เก็บอยู่ ณ หน่วยเก็บข้อมูลของผู้ให้บริการต่าง ๆ มีความสัมพันธ์กัน การที่ผู้ใช้คำสั่งให้มีการแก้ไขข้อมูลที่เก็บอยู่ ณ หน่วยเก็บข้อมูลของผู้ให้บริการตัวอื่นด้วย ถ้าแอพพลิเคชั่นของผู้ใช้บริการสามารถสนับสนุนให้ตัวผู้ขอใช้บริการทำหน้าที่เป็นตัวกลางระหว่างผู้ให้บริการทั้งหลายแล้ว ข้อจำกัดข้างต้นก็สามารถที่จะแก้ไขได้

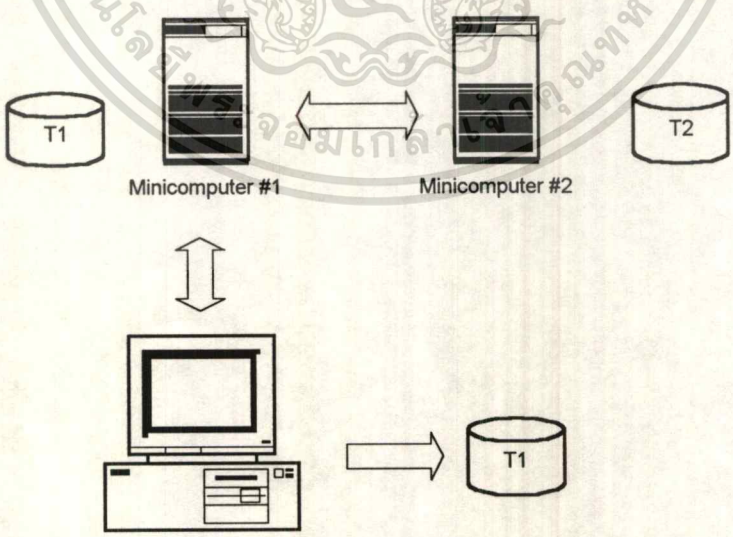
ถึงแม้ว่าประมวลผลแบบนี้จะสามารถแก้ไขข้อมูลที่ผู้ให้บริการได้เพียงหนึ่งตัว แต่ก็ยังมีความเป็นไปได้ที่อาจจะเกิดเดดล็อคขึ้นในเวลาที่มีผู้ใช้หลาย ๆ คนเรียกใช้ข้อมูลพร้อม ๆ กัน ดังนั้นจึงจำเป็นที่จะต้องมีระบบมาทำการควบคุมการเรียกใช้ข้อมูลด้วย การกระจายข้อมูลของไคลเอนท์เซิร์ฟเวอร์ประเภทนี้อาจทำได้โดยใช้วิธีแมนนวลเอ็กแทรกต์



รูปที่ 2-11 ไคลเอนท์เซิร์ฟเวอร์ประเภทที่ 4

2.5.1.5 มัลติไซต์อัพเดทไคลเอนท์เซิร์ฟเวอร์ (Multi-site update Client/Server)

ลักษณะแอพพลิเคชั่นประเภทนี้ดังแสดงในรูปที่ 12 จะสนับสนุนการติดต่อระหว่างผู้ให้บริการแต่ละตัว ดังนั้นผู้ใช้จึงสามารถที่ออกคำสั่งประเภทที่จะแก้ไขข้อมูลที่เก็บอยู่หลายที่ได้ ถ้ามองในอีกแง่หนึ่งก็คือข้อมูลที่เก็บอยู่ ณ ที่ต่างๆ กัน สามารถที่จะมีความสัมพันธ์กันได้ ลักษณะของไคลเอนท์เซิร์ฟเวอร์ประเภทนี้จะเป็นประเภทแรกที่มีความสามารถในเรื่องการกระจายฐานข้อมูลและเมื่อมีความสามารถในเรื่องนี้แล้ว การกระจายข้อมูลจะถูกการทำด้วยวิธี สแนพชอต (Snapshots) จากผู้ใช้บริการฐานข้อมูลที่จะเป็นวิธีแมนนวลเอ็กแทรกต์

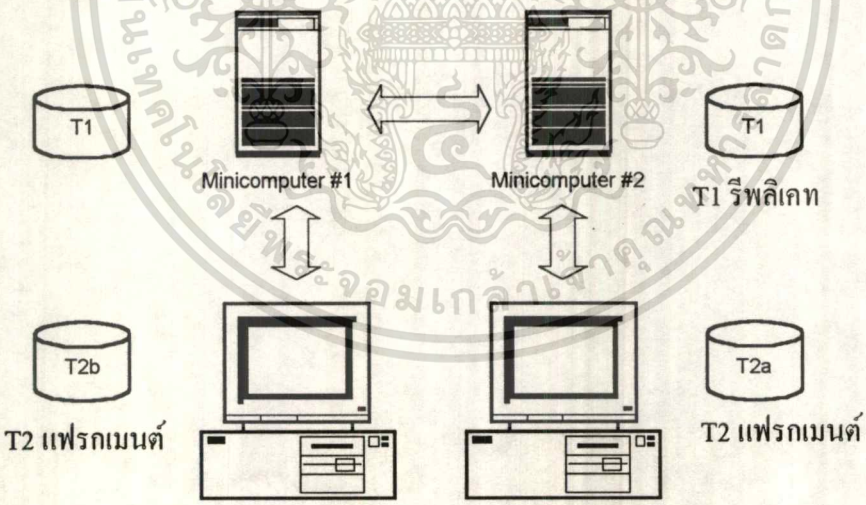


รูปที่ 2-12 ไคลเอนท์เซิร์ฟเวอร์ประเภทที่ 5

2.5.1.6 ไคลเอนท์เซิร์ฟเวอร์แบบระบบฐานข้อมูลแบบกระจาย (Distribute database Client/Server)

เป็นระบบไคลเอนท์เซิร์ฟเวอร์ที่ใช้แอพพลิเคชั่นฐานข้อมูลแบบกระจายและใช้การประมวลผลแบบดิสทริบิวเต็ดรีเควสต์ (Distribute request) ดังแสดงในรูปที่ 13 ลักษณะของไคลเอนท์เซิร์ฟเวอร์ประเภทนี้ ผู้ให้บริการฐานข้อมูลจะสนับสนุนทั้งการคัดแบ่งข้อมูล หรือการทำก็อปปี้ข้อมูลทั้งหมดไปเก็บไว้ตามหน่วนเก็บข้อมูลของผู้ให้บริการต่าง ๆ ซึ่งทำให้การอ่านข้อมูลสามารถทำได้ด้วนความรวดเร็วแต่การแก้ไขข้อมูลอาจจะต้องใช้เวลามากกว่าเพราะว่าจะต้องมีการติดต่อกันระหว่างผู้ให้บริการซึ่งอาจจะไม่ใช่เพียงแค่ 2 ตัว
ดังนั้นเทคโนโลยีทางด้านการสื่อสารจึงมีบทบาทสำคัญในการที่จะขจัดปัญหาในเรื่องของความเร็ว

ความสามารถที่จำเป็นสำหรับแอพพลิเคชั่นประเภทนี้ คือ การที่แอพพลิเคชั่นจะไม่จำเป็นต้องรู้ตำแหน่งของผู้ให้บริการ หรือตำแหน่งที่เกิดการประมวลผลฐานข้อมูล การควบคุมประสิทธิภาพโดยรวมของระบบ การควบคุมความถูกต้องของข้อมูลที่กระจายเห็นอยู่ตามที่ต่าง ๆ และการควบคุมการทำการกระจายข้อมูลซึ่งเป็นส่วนสำคัญของระบบไคลเอนท์เซิร์ฟเวอร์ประเภทนี้



รูปที่ 2-13 ไคลเอนท์เซิร์ฟเวอร์ประเภทที่ 6

2.5.2 ข้อดีข้อเสียของระบบไคลเอนท์เซิร์ฟเวอร์

2.5.2.1 ข้อดีของระบบไคลเอนท์เซิร์ฟเวอร์

- เนื่องจากระบบไคลเอนท์เซิร์ฟเวอร์มีการแบ่งแยกการประมวลออกระหว่างส่วนไคลเอนท์และส่วนเซิร์ฟเวอร์ ซึ่งทำให้การประมวลผลข้อมูลจะทำที่เซิร์ฟเวอร์ ทำให้ความเร็วของระบบจัดการฐานข้อมูลไม่ขึ้นกับความเร็วของเวิร์คสเตชั่น (Workstation) ดังนั้น

เวิร์คสเตชั่นที่ใช้ไม่จำเป็นต้องเป็นเครื่องที่มีประสิทธิภาพสูง เพียงแต่สามารถให้ระบบฟรอนท์เอ็นทำงานได้ก็เพียงพอ

- เนื่องจากมีการแบ่งแยกการประมวลผลออกเป็น 2 ฝั่ง ทำให้ลดโหลด (load) ในการติดต่อบนระบบเครือข่ายระหว่างเซิร์ฟเวอร์และไคลเอนท์ ถ้าเปรียบเทียบกับระบบโครงสร้างแบบรวมศูนย์ จะต้องมีการส่งไฟล์ฐานข้อมูลทั้งไฟล์ไปกลับระหว่างผู้ให้บริการและผู้ใช้บริการอยู่ตลอดเวลาที่มีการเรียกใช้ข้อมูล แต่สำหรับระบบไคลเอนท์เซิร์ฟเวอร์จะเป็นการส่งคิวรี่และผลลัพธ์ที่ได้จากดาต้าเบสเซิร์ฟเวอร์ เพราะการใช้ประโยคำสั่งเอสคิวแอลในแอพพลิเคชั่นของระบบไคลเอนเซิร์ฟเวอร์สามารถที่จะสร้างตารางข้อมูลบรรจุผลลัพธ์ที่ได้จากการรวม ตัดทอน และเปลี่ยนแปลงข้อมูลจากตารางข้อมูลจากผู้ให้บริการแล้วค่อยส่งคำสั่งเอสคิวแอลไว้ในตัวมันเองโดยทำการเก็บไว้ในลักษณะของสตอรโพรซีเดอร์ (Store procedure) ซึ่งผู้ใช้จะทำการเรียกใช้โดยออกคำสั่งสั้น ๆ ให้ผู้ให้บริการเรียกประโยคคำสั่งนั้น ๆ ออกมาทำงานจะเป็นการช่วยลดปริมาณข้อมูลที่ส่งผ่ายเข้าไปในเครือข่ายได้ทางหนึ่ง แต่ก็มีปัญหาอยู่ว่ายังไม่มีการกำหนดมาตรฐานในเรื่องของสตอร์โพรซิเดอร์ขึ้นมาและระบบจัดการฐานข้อมูลหลายตัวยังไม่สนับสนุนความสามารถในเรื่องนี้จากการแบ่งแยกออกเป็นไคลเอนท์เซิร์ฟเวอร์ทำให้ส่วนที่ไคลเอนท์ทำงานอยู่และแพลตฟอร์มสามารถเป็นอะไรก็ได้ ซึ่งแพลตฟอร์มที่ใช้อาจจะเป็นพีซีที่เข้ากันได้กับพีซีของไอบีเอ็ม(IBM) ,แมคอินทอช (MACINTOSH) , ยูนิกซ์เวิร์คสเตชั่น (UNIX Workstation) นอกจากนั้นยังสามารถใช้กับระบบปฏิบัติการได้หลายตัว เช่น ดอส (DOS) ,พีซีดอส (PC-DOS) ,ไมโครซอฟท์วินโดว์ (MS-WINDOWS) ,ไอบีเอ็มโอเอสทู (IBM OS/2) หรือ แอปเปิ้ล (APPLES SYSTEM 7) ทำให้เวิร์คสเตชั่นสามารถใช้แอพพลิเคชั่นตัวใดก็ได้ไสการเข้าถึงฐานข้อมูล
- ระบบไคลเอนท์เซิร์ฟเวอร์สามารถรักษาความคงสภาพของข้อมูล (Data integrity)ได้โดยระบบจัดการฐานข้อมูลจะไม่อนุญาตให้ผู้ใช้เข้าถึงฐานข้อมูลจากภายนอก เช่นอาจจะทำการเข้ารหัสไฟล์เพื่อป้องกันผู้ใช้ดูข้อมูลจากภายนอก นอกจากนั้นระบบจัดการฐานข้อมูลยังสามารถทำการแบคอัพไปยังเทปแบบเรียลไทม์(real time) ได้ คือขณะที่ฐานข้อมูลกำลังถูกใช้ก็มีการแบคอัพไปยังเทป การทำดิสค์มิเรอร์ (Disk Mirroring) ซึ่งการทำสิ่งเหล่านี้เพื่อรักษาความถูกต้องของข้อมูลจากการเกิดการเสียหายของระบบหรือไฟดับ
- สำหรับกำหนดขนาดของผู้ใช้บริการและผู้ให้บริการได้อย่างอิสระและสามารถลดและขยายในภายหลัง
- แอพพลิเคชั่นต่าง ๆ สามารถใช้ข้อมูลบนผู้ให้บริการร่วมกันได้

2.5.2.2 ข้อเสียของระบบไคลเอนเซิร์ฟเวอร์

- เสียค่าใช้จ่ายในการดูแลระบบ และบำรุงรักษาระบบ

- ฮาร์ดแวร์ที่ทำหน้าที่เป็นดาต้าเบสเซิร์ฟเวอร์จะต้องเป็นเครื่องที่มีประสิทธิภาพสูงซึ่งทำให้เสียค่าใช้จ่ายสูง
- ซอฟท์แวร์ที่เป็นระบบจัดการฐานข้อมูลจะมีราคาสูง
- มีผลกระทบจากการกระจายข้อมูลต่อประสิทธิภาพของระบบ
- การบริการระบบข้อมูลทำได้ลำบาก ในระบบที่ข้อมูลทุกอย่างเก็บรวบรวมอยู่ที่ส่วนกลาง การควบคุมจะกระทำได้สะดวก แต่เมื่อเรากระจายการพัฒนาระบบงานการประมวลผลแอพพลิเคชั่นและการจัดเก็บข้อมูลออกไปแล้ว ความง่ายและความสะดวกในการควบคุมจะสูญเสียไป ซึ่งจะมีปัญหาในเรื่องต่าง ๆ ดังนี้
    - ◆ การจัดการไลบรารี่ของโปรแกรมของโปรแกรมต่าง ๆ ที่เก็บกระจายกันอยู่
    - ◆ การจัดการข้อมูลที่เก็บอยู่ตามที่ต่าง ๆ
    - ◆ การตรวจสอบและเพิ่มประสิทธิภาพในการทำงานของระบบ
    - ◆ การสำรองข้อมูลที่เก็บอย่างกระจายในระบบ
    - ◆ การจัดการระบบเครือข่าย

2.6 โอเพนดาต้าเบสคอนเน็คติวิตี้ (โอดีบีซี) (Open DataBase Connectivity : ODBC)

2.6.1 ความหมายของคำว่าโอดีบีซี

Open Database Connectivity คือวิธีการติดต่อและเข้าถึงจากแอพพลิเคชั่นสู่ระบบการจัดการฐานข้อมูล (ระบบจัดการฐานข้อมูล) โดยใช้ภาษา SQL เป็นมาตราฐานการเข้าถึงข้อมูล ความสามารถในการเชื่อมต่อแบบนี้ทำให้แอพพลิเคชั่นสามารถเข้าถึงฐานข้อมูลได้หลายรูปแบบซึ่งทำให้ผู้พัฒนาโปรแกรมสามารถพัฒนาโปรแกรมไปได้โดยไม่ต้องทำการระบุชนิดของระบบการจัดการฐานข้อมูล

แต่เดิมนั้นการพัฒนาโปรแกรมประยุกต์ที่ใช้งานเกี่ยวกับฐานข้อมูล การเข้าใช้ฐานข้อมูลโปรแกรมเหล่านี้จะทำผ่านการเรียกใช้เอ็มเบดเด็ด เอสคิวแอล (Embedded SQL) ซึ่งในขณะนั้นวิถีทางแบบนี้ก็ดูจะไปได้ทีเดียว เพราะว่าตัวโปรแกรมสามารถทำการเปลี่ยนรูปแบบของระบบไม่ว่าจะเป็นทางด้านฮาร์ดแวร์ หรือ ซอฟท์แวร์ได้หลายรูปแบบ รวมทั้งระบบปฏิบัติการด้วย (โดยการคอมไพล์ใหม่ทุกครั้งที่มีการย้ายระบบ)

อย่างไรก็ตามในการพัฒนาโปรแกรมในระบบที่มีความแตกต่างกัน เช่น การเรียกใช้ข้อมูลของออราเคิลจาก ไมโครซอฟท์ เอ็กซ์เซล(Microsoft Excel) วิธีการเข้าถึงข้อมูลแบบเดิมนั้นจะต้องทำการพรีคอมไพล์โค้ดของเอ็กซ์เซลและออราเคิลโดยการใช้ไอบีเอ็มพรีคอมไพล์เลอร์(IBM

Precompiler) และออราเคิลพรีคอมไพล์เลอร์ (Oracle Precompiler) ตามลำดับ ซึ่งจะเห็นว่าเป็นการยุ่งยากมากทีเดียว

สำหรับวิธีการต่อเชื่อมแบบโอดีบีซีจะให้ความสะดวกในการติดต่อข้อมูลมากกว่าวิธีการดั้งเดิม โดยการกำหนดมาตรฐานการต่อเชื่อมของข้อมูล (Data protocol, DBMS capability) และแนวทางนี้ได้ทำให้เกิดความคิดที่จะสร้างไดร์เวอร์การติดต่อกับของงานข้อมูลขึ้นมา (DLL)

2.6.2 ข้อดีของการติดต่อโดยใช้โอดีบีซี

- ฟังก์ชั่นของโอดีบีซีอนุญาตให้แอพพลิเคชั่นติดต่อกับระบบจัดการฐานข้อมูล ได้โดยสะดวก (การทำคำสั่งเอสคิวแอลและการรับผลลัพธ์)
- ใช้ภาษาเอสคิวแอลตามมาตรฐานSQL CAE, X/Open และSQL Access Group (SAG)
- มีการกำหนดการส่งกลับรหัสความผิดพลาด (Error Code) เป็นมาตรฐานเดียวกัน
- เป็นวิธีการมาตรฐานในการติดต่อกับระบบจัดการฐานข้อมูล
- มีการกำหนดชนิดของข้อมูล(Data Type)เป็นมาตรฐาน
- ชุดคำสั่งเอสคิวแอลสามารถกำหนดได้แม้ในขณะรัน
- สามารถเขียนโปรแกรมชุดเดียวแต่สามารถเข้าใช้ระบบจัดการฐานข้อมูลได้หลายตัว
- ตัวโปรแกรมไม่ต้องรับผิดชอบในการดูแลการติดต่อข้อมูลกับระบบจัดการฐานข้อมูล
- ค่าข้อมูลสามารถถูกส่งหรือรับได้ในรูปแบบที่สะดวกขึ้น

2.6.3 องค์ประกอบของโอดีบีซี

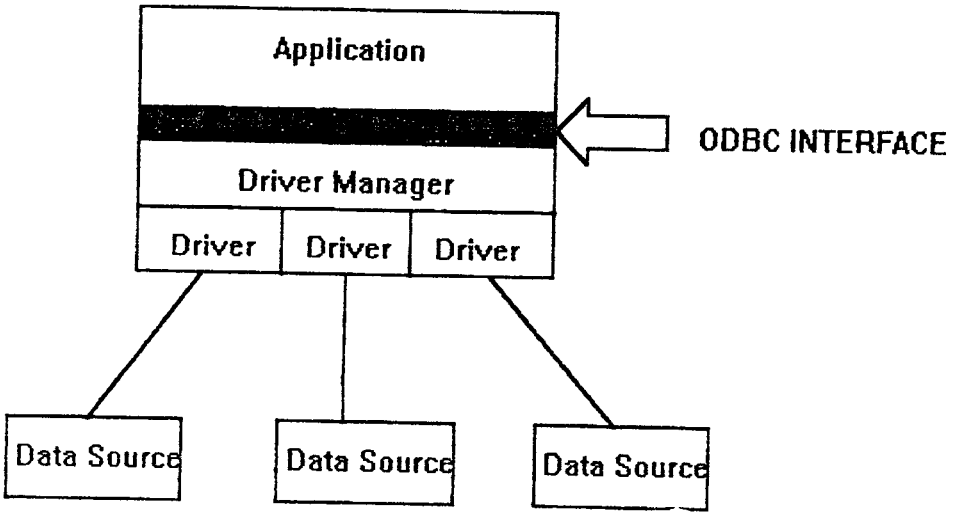
สถาปัตยกรรมของโอดีบีซีประกอบด้วย 4 ส่วนสำคัญ

1.แอพพลิเคชั่น ทำหน้าที่ประมวลผลและเรียกใช้ฟังก์ชั่นของโอดีบีซีตามคำสั่งภาษา SQL พร้อมทั้งทำการรับผลลัพธ์ด้วย

2.ตัวจัดการไดร์เวอร์(Driver Manager) ทำหน้าที่โหลดไดร์เวอร์และเชื่อมต่อกับแหล่งข้อมูล

3.ไดร์เวอร์(Driver)ทำหน้าที่ประมวลผลการเรียกใช้ฟังก์ชั่นของโอดีบีซี ส่งคำสั่ง SQL ไปสู่แหล่งข้อมูลที่ถูกต้องและทำการส่งผลลัพธ์กลับให้แอพพลิเคชั่น และในบางครั้งไดร์เวอร์จะทำหน้าที่แปลงคำสั่งที่ส่งมาให้อยู่ในรูปที่สนับสนุนโดยระบบจัดการฐานข้อมูลแต่ละชนิดอีกด้วย

4.แหล่งข้อมูล(Data Source)เป็นแหล่งข้อมูลที่ผู้ใช้ต้องการเข้าถึง



รูปที่2-14 องค์ประกอบของโอบีดีซี

ตัวโปรแกรมจะเรียกใช้การเชื่อมโอบีดีซี ในการทำงานต่อไปนี้

1. ร้องขอการต่อเชื่อมกับแหล่งข้อมูล
2. ส่งคำสั่งเอสคิวแอลสู่แหล่งข้อมูล
3. กำหนดพื้นที่การจัดเก็บและรูปแบบของข้อมูล ที่เป็นผลลัพธ์จากเอสคิวแอล รีเควสซ์ (SQL request)
4. ร้องขอผลลัพธ์
5. ประมวลผลและจัดการกับข้อผิดพลาด
6. รายงานผลให้กับผู้ใช้ (ถ้าจำเป็น)
7. ร้องขอการคอมมิท (Commit) หรือ โรลแบ็ค (Roll back) สำหรับควบคุมการประมวลผลทรานแสคชั่น (Transaction)
8. ยกเลิกการติดต่อกับแหล่งข้อมูล

2.7 หลักการของออปเจค-โอเรียนเต็ด

2.7.1 ที่มาของการเขียนโปรแกรมแบบออปเจค-โอเรียนเต็ด

ในอดีตที่ผ่านมาการพัฒนาโปรแกรมต่างๆ จะถูกแบ่งออกเป็น 2 ส่วนใหญ่ๆ คือ ส่วนของข้อมูล และส่วนของตัวโปรแกรม เช่น โพรซีเยอร์ (Procedure) ต่างๆ ซึ่งแนวความคิดในการพัฒนาโปรแกรมทั้ง 2 ส่วนดังกล่าวก็ได้ถูกพัฒนาขึ้นมาตามลำดับ ซึ่งการพัฒนาดังกล่าวนั้นถูกแยกจากกันตลอดมา ดังตาราง

รูปแบบการจัดการข้อมูล	**รูปแบบการเขียนโปรแกรม**
- ข้อมูลอยู่ภายในโปรแกรม (Data within Program)	*- การเขียนโปรแกรมแบบซีเควนเชียล*

มีลักษณะดังนี้

1. ข้อมูลหรือตัวแปรทุกตัวเป็นของส่วนรวม (global variable)
2. ส่งผลให้ข้อมูลของโพรซีเยอร์หนึ่งถูกเข้าถึงได้โดยอีกโพรซีเยอร์หนึ่ง
3. เกิดข้อผิดพลาดขึ้นได้ง่าย

- *การเขียนโปรแกรมแบบลำดับ (Sequential Programming)*
- *การเขียนโปรแกรมแบบโมดูลาร์ (Modular Programming)*
- *การเขียนโปรแกรมแบบสตรัคเจอร์ (Structure Programming)*

- *ข้อมูลอยู่ภายในโพรซีเยอร์ (Data within Subroutine)* หมายถึงว่าแต่ละโพรซีเยอร์มีข้อมูลหรือตัวแปรอยู่ในขอบเขตของโพรซีเยอร์นั้น (local varibles)
- ข้อมูลถูกเก็บอยู่ภายนอกโปรแกรม *(Data outside Program)* เป็นการเก็บข้อมูลไว้ในไฟล์ธรรมดาแยกจากโปรแกรม
- มีการใช้ข้อมูลร่วมกันระหว่างโปรแกรมต่างๆ *(Sharing Data)* นั่นคือเริ่มเป็นฐานข้อมูลประเภทต่างๆ

ถึงแม้ว่าแนวความคิดในการจัดการข้อมูลและการเขียนโปรแกรมจะถูกพัฒนาขึ้นเรื่อยๆก็ตาม แต่อย่างไรก็ดี เราก็ยังไม่สามารถบอกถึงความสัมพันธ์ที่ชัดเจนระหว่างข้อมูลกับส่วนของโปรแกรมหรือโพรซีเยอร์ต่างได้ เช่นว่า ตัวแปรตัวนี้สามารถถูกอ้างถึงได้โดยโพรซีเยอร์ใดได้บ้าง หรือว่าโพรซีเยอร์นี้เป็นการกระทำกับข้อมูลหรือตัวแปรใดบ้าง หน้าที่เหล่านี้เป็นของผู้เขียนโปรแกรมที่จะกำหนดขอบเขตของการเข้าถึงข้อมูลต่างๆ ให้เป็นไปอย่างที่ควรจะเป็น เช่น ตัวแปรตัวนี้ควรจะมีขอบเขตอยู่ในโพรซีเยอร์นี้เท่านั้น หรือว่า ตัวแปรนี้ควรถูกเข้าถึงได้จากทุกส่วนของโปรแกรม ซึ่งถ้าหากว่าผู้เขียนโปรแกรมไม่ระมัดระวัง ก็อาจจะทำให้ข้อมูลบางตัวถูกเข้าถึงโดยโพรซีเยอร์ที่ไม่ต้องการได้ ซึ่งอาจก็ให้เกิดผลกระทบหรือข้อผิดพลาดที่คาดไม่ถึงได้ ดังนั้นจึงเกิดแนวความคิดใหม่ที่จะนำทั้ง 2 ส่วนนี้ คือส่วนของข้อมูล และส่วนของโปรแกรม มารวมเข้าด้วยกัน และแนวความคิดนี้ถูกเรียกว่า *การเขียนโปรแกรมแบบออปเจค-โอเรียนเต็ด (Object-Oriented Programming : OOP (โอ-โอ-พี))*

2.7.2 การเขียนโปรแกรมแบบออปเจค-โอเรียนเต็ด

หลักการสำคัญในการเขียนโปรแกรมแบบออปเจค-โอเรียนเต็ดก็คือ *ชนิดของข้อมูลแบบนามธรรม (Abstract Data Type)* ซึ่งก็คือการนำเอาข้อมูลและชุดของการทำงาน (operation) มารวมเข้าไว้ด้วยกัน ซึ่งการทำงานก็จะเป็นตัวกำหนดคุณสมบัติของชนิดของข้อมูลนั้นๆ

การกำหนดการทำงานของชนิดของข้อมูลแบบนามธรรมนั้นเราเรียกว่า การนิยามคลาส (class definiton) ซึ่งเป็นการกำหนดสิ่งต่าง ๆ ดังต่อไปนี้

- จะเรียกใช้ชนิดของข้อมูลนั้นได้อย่างไรบ้าง
- โครงสร้างข้อมูลของชนิดของข้อมูลนั้นเป็นอย่างไร

การทำงานของคลาสมีชื่อเรียกอีกอย่างหนึ่งว่า *วิธีการ (method)* ซึ่งเปรียบได้กับฟังก์ชั่นหรือโพรซีเยอร์ในภาษาที่ไม่ใช่ภาษาออปเจค-โอเรียนเต็ดนั่นเอง

ถ้าเราจะเปรียบเทียบคลาสกับสิ่งที่มีอยู่ในภาษาโปรแกรมทั่วไปๆ เช่น ภาษาปาสคาล (เนื่องจากว่าการพัฒนาระบบในโครงงานนี้ใช้ Delphi ซึ่งภาษาที่ใช้ก็คือ ภาษาปาสคาลที่มีคุณสมบัติทางออปเจค-โอเรียนเต็ดนั่นเอง) ก็จะเปรียบเทียบได้กับ เรคคอร์ด (structure)ในภาษาปาสคาล แต่คลาสจะเป็นเรคคอร์ดที่มีฟังก์ชั่นหรือโพรซีเยอร์เป็นส่วนประกอบอยู่ด้วย นอกจากนี้ก็จะยังมีคุณสมบัติอื่นๆเพิ่มเติมมาจากเรคคอร์ดทั่วไปๆ ก็คือ เอนแคปซูเลชั่น (encapsulation) ,การถ่ายทอดคุณสมบัติ (inheritance) และโพลีมอร์ฟิซึ่ม (polymorphism) ซึ่งรายละเอียดเหล่านี้จะกล่าวถึงโดยละเอียดต่อไป

ออปเจค (Object) ก็คือตัวแปรของคลาสต่างๆ ที่เราได้ประกาศไว้นั่นเอง หรือเปรียบเทียบได้กับว่าเราประกาศเรคคอร์ดไว้ในภาษาปาสคาล ต่อมาก็ประกาศตัวแปรที่มีชนิดของข้อมูลเป็นเรคคอร์ดนั้น

โดยที่ข้อมูลของออปเจคนั้นก็คือข้อมูลที่ถูกกำหนดไว้ในคลาสนั่นเอง เราจะเรียกข้อมูลเหล่านั้นว่าคุณสมบัติของออปเจค ซึ่งมีศัพท์ที่ใช้แทนหลายคำ เช่น attribute, propery หรือ instance variable และวิธีการที่สามารถจะกระทำกับออปเจคนั้นได้ก็คือ วิธีการเรียกใช้ที่กำหนดไว้ในคลาส

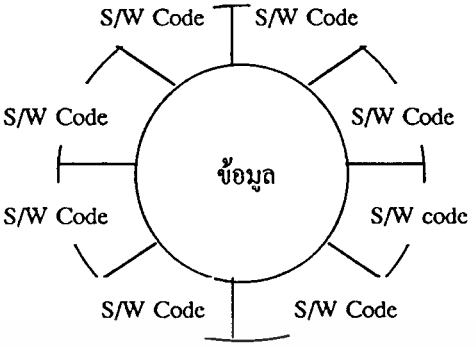
ตัวอย่างเช่น เรากำหนดคลาส automated vehicle ขึ้นประกอบไปด้วยคุณสมบัติดังต่อไปนี้

- ขนาด
- สิ่งที่บรรทุก
- ความเร็ว
- ตำแหน่ง

และมีวิธีการในการเข้าถึงข้อมูล ดังต่อไปนี้

- move_to
- load
- unload
- lift

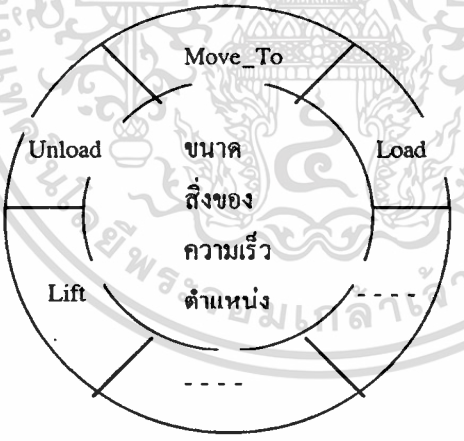
เราจะใช้สัญลักษณ์ดังต่อไปนี้แทน ออปเจค



รูปที่ 2-15 สัญลักษณ์ที่ใช้แทนออปเจค

ในที่นี้ ข้อมูลก็คือคุณสมบัติของออปเจคและ software code ก็คือ สิ่งที่แสดงถึงความสามารถของ ออปเจค นั้น ซึ่งก็คือวิธีการที่ออปเจคนั้นสามารถทำงานได้นั่นเอง การที่เราจะเข้าถึงข้อมูลของออปเจคใดๆจะต้องทำโดยผ่านวิธีการต่างๆของออปเจคนั้นเท่านั้น

เมื่อเรากำหนดให้ ออปเจค vehicle1 เป็น ออปเจค ในคลาส automate vehicle ออปเจค vehicle1 ก็จะมีโครงสร้างข้อมูลและวิธีการในการเรียกใช้และเข้าถึงตามที่เรากำหนดไว้ในการนิยามคลาส ดังรูป



รูปที่ 2-16 ออปเจค vehicle1

การเปลี่ยนแปลงคุณสมบัติของออปเจคจะต้องทำโดยผ่านวิธีการต่างๆซึ่งก็คือ move_to , unload , load ,lift เป็นต้น เช่น การเปลี่ยนแปลงตำแหน่งของออปเจค vehicle1 นี้ก็จะต้องผ่านวิธีการ move_to หรือ การเปลี่ยนแปลงสิ่งของที่บรรจุ ก็จะต้องทำโดยผ่านวิธีการ load หรือ unload เป็นต้น

จากตัวอย่าง จะเห็นได้ว่าเมื่อเราใช้วิธีการเขียนโปรแกรมแบบออปเจค-โอเรียนเต็ด แล้วจะทำให้เราสามารถมองเห็นได้ชัดขึ้นว่าโพรซีเยอร์หรือฟังก์ชั่นที่เราเขียนขึ้นสามารถเข้าถึง

ข้อมูลตัวใดได้บ้าง และจะไม่มีการเข้าถึงข้อมูลโดยพละการ คือ จะต้องผ่านวิธีการที่ถูกกำหนดไว้เท่านั้น

คลาสและคลาสย่อย (Class and Subclass)

ในการเขียนโปรแกรมแบบออปเจค-โอเรียนเต็ดนั้นนอกจากการกำหนดคลาสแล้วก็ยังมีกำหนดคลาสย่อยอีกด้วย คลาสย่อย ก็คือ คลาส ๆ หนึ่งแต่ว่ามีคุณสมบัติของคลาสนั้นสืบทอดมาจากคลาสอีกคลาส หนึ่ง

ตัวอย่างเช่น เรากำหนดคลาสของบุคคล ขึ้นมาเป็นคลาสแรก มีคุณสมบัติดังต่อไปนี้

- ชื่อ
- นามสกุล
- วันเกิด
- ที่อยู่

แล้วเราก็กำหนดคลาสใหม่ขึ้นมาให้ชื่อว่า คลาสพนักงาน และให้คลาสพนักงานนี้เป็นคลาสย่อยของคลาสบุคคล ดังนั้นคลาสพนักงาน ก็จะมีคุณสมบัติ ชื่อ นามสกุล วันเกิด และที่อยู่ด้วยเช่นเดียวกับคลาสบุคคล โดยทั่วไปแล้ว คลาสที่เป็นต้นแบบนั้นจะถูกเรียกว่า คลาสพื้นฐาน หรือ ซุปเปอร์คลาส (Base class หรือ Super class)และคลาสที่สืบทอดคุณสมบัติมาจากคลาสพื้นฐานหรือซุปเปอร์คลาส ก็จะถูกเรียกว่าคลาสย่อย

จากลักษณะการเขียนโปรแกรมแบบออปเจค-โอเรียนเต็ด ดังกล่าวส่งผลให้โปรแกรมที่ได้มีลักษณะดังนี้ ซึ่งถือเป็นลักษณะสำคัญของการเขียนโปรแกรมแบบออปเจค-โอเรียนเต็ด ด้วย

1. **Encapsulation** หมายถึงว่า ข้อมูลต่าง ๆนั้นจะถูกเก็บซ่อนไว้ภายในออปเจค ซึ่งทำให้การเข้าถึงข้อมูลของออปเจคนั้นโดยออปเจคอื่นๆนั้นเป็นไปอย่างมีระบบและกฎเกณฑ์ ช่วยป้องกันไม่ให้ข้อมูลนั้นถูกเปลี่ยนแปลงหรือแก้ไขไปโดยไม่ตั้งใจ ตัวอย่างเช่น ถ้าหากว่าเราเขียนโปรแกรมแบบ ฟังก์ชั่น-โอเรียนเต็ด และมีการประกาศตัวแปรเป็นของส่วนรวม (global variable) ไว้ ตัวแปรตัวนั้นก็จะถูกเข้าถึงได้โดยทุกๆ โพรซีเยอร์ การเปลี่ยนแปลงค่าตัวแปรนั้นก็จะส่งผลกระทบไปยังโพรซีเยอร์อื่นๆที่อ้างถึงตัวแปรนั้นด้วย แต่ถ้าเป็นการเขียนโปรแกรมแบบออปเจค-โอเรียนเต็ด แล้ว ข้อมูลจะถูกเก็บไว้ในออปเจคแต่ละออปเจค การเข้าถึงข้อมูลจะต้องผ่านวิธีการที่กำหนดไว้เท่านั้น ดังนั้นก็จะไม่มีข้อผิดพลาดในการเปลี่ยนแปลงค่าของข้อมูลแล้วไปกระทบกับโพรซีเยอร์อื่นๆอีก

ประโยชน์อีกข้อหนึ่งของ Encapsulation ก็คือ ออปเจคต่างๆไม่ต้องสนใจว่าข้อมูลต่างๆของออปเจคอื่นๆจะมีรูปแบบอย่างไร เพราะว่าการเข้าถึงก็ยังคงทำได้โดยผ่านวิธีการเดิม ส่วนขั้นตอนต่างๆในวิธีการที่กระทำกับข้อมูลนั้น ก็เป็นเรื่องภายในของออปเจคที่เป็นเจ้าของข้อมูลนั้น ตัวอย่างเช่น ออปเจค queue ซึ่งมีการเข้าถึงได้ 3 วิธี คือ insert,remove และ empty ถ้าแต่เดิมเราทำการสร้าง queue โดยใช้โครงสร้างข้อมูลแบบอาร์เรย์ และต้องการจะเปลี่ยนเป็น

โครงสร้างข้อมูลแบบ linked list ก็ไม่ต้องไปแก้ไข procedure อื่นๆที่อ้างถึง queue นี้ เนื่องจากว่าวิธีการเข้าถึง queue นี้ก็ยังคงใช้วิธีการเดิม ออปเจคอื่นๆก็ไม่ต้องสนใจและรับรู้ว่าโครงสร้างภายในของออปเจคอื่นๆนั้นมีการเปลี่ยนแปลงไปอย่างไร เพราะส่วนที่ใช้เชื่อมต่อกันนั้นก็ยังเหมือนเดิม

2. **Inheritance** หมายถึง การสืบทอดลักษณะระหว่างกันของคลาส ต่างๆ เช่น เราสร้างคลาสรถยนต์ขึ้นมา ประกอบไปด้วยคุณสมบัติดังนี้

- ขนาดเครื่องยนต์
- ลักษณะตัวถัง
- สี
- ขนาดของรถ

ต่อจากนั้นเราสร้างคลาสย่อยของ คลาสรถยนต์ ให้ชื่อว่า คลาสฟอร์ด คุณสมบัติของ คลาสฟอร์ด ก็จะประกอบไปด้วย 2 ส่วนคือ ส่วนที่ได้รับสืบทอด(inherit)มาจาก คลาสรถยนต์ และส่วนที่เป็นคุณสมบัติของตัวมันเอง ดังนี้

(ส่วนที่สืบทอดมาจาก คลาสรถยนต์)
- ขนาดเครื่องยนต์
- ลักษณะตัวถัง
- สี
- ขนาดของรถ

(ส่วนที่เป็นของตัวมันเอง)
- ประกันภัย
- การบำรุงรักษา
- รุ่น

คุณสมบัติการสืบทอดนี้ ทำให้เราสามารถนำคลาสต่างๆที่เคยถูกสร้างมาแล้วมาใช้ใหม่ (reusablity)ได้โดยการให้เป็นซุปเปอร์คลาส และสร้างคลาสที่ต้องการซึ่งมีคุณสมบัติเพิ่มเติมจากคลาสเดิมได้โดยให้คลาสใหม่ที่จะสร้างขึ้นดังกล่าวนั้นสืบทอดคุณสมบัติต่างๆจากคลาสที่เราสร้างไว้แต่เดิมได้

3.**Polymorphism**

แต่เดิมเมื่อเราเขียนโปรแกรมจัดการสิ่งของที่มีวิธีการร่วมกันแต่มีขั้นตอนภายในวิธีการนั้นแตกต่างกัน เช่น เอกสาร , รูปภาพ ต่างก็สามารถสั่งพิมพ์ได้ แต่ขั้นตอนในการพิมพ์นั้นอาจจะแตกต่างกันไป เราจะต้องมีประโยค case ในโปรแกรมเพื่อทำการพิจารณาว่าสิ่งที่ต้องการจะพิมพ์ว่าเป็นอะไร แล้วค่อยไปเรียกโพรซีเยอร์ที่ทำงานนั้นๆ เช่น print_text() หรือ print_image ให้ทำงาน ถ้าหากว่าเรามีประเภทของสิ่งพิมพ์มากเราก็จะต้องตั้งชื่อโพรซีเยอร์ที่จัดการพิมพ์

สิ่งต่าง ๆนั้นให้แตกต่างกันไปและจดจำลำบาก เมื่อมีการเปลี่ยนแปลงชื่อโพรซีเยอร์หนึ่งก็จะต้องไปตามแก้โปรแกรมทุก ๆที่ที่มีการเรียกใช้โพรซีเยอร์นั้น หรือถ้ามีของเพิ่มขึ้นมาเราก็ต้องไปแก้ประโยค case ที่เขียนไว้เดิมให้รู้จักกับสิ่งพิมพ์อันใหม่ แต่ถ้าเป็นการเขียนโปรแกรมแบบออปเจค-โอเรียนเต็ด นั้นคุณสมบัติ Polymorphism จะช่วยให้เราไม่ต้องตั้งชื่อโพรซีเยอร์ที่ทำงานอย่างเดียวกัน (print) แต่ว่าสำหรับออปเจคแตกต่างกันไป (เอกสาร ,รูปภาพ) ให้แตกต่างกัน เพราะในการเขียนโปรแกรมแบบ ออปเจค-โอเรียนเต็ด นั้น 1 วิธีการ จะมีได้หลายหน้าที่ หรือกล่าวอีกอย่างหนึ่งว่า วิธีการที่มีชื่อ 1 ชื่อสามารถทำงานได้หลายอย่าง หมายถึงว่าในแต่ละออปเจคที่สามารถสั่งพิมพ์ได้ก็จะมีวิธีการในการเข้าถึงคือ print เหมือนกันหมด แต่ขั้นตอนภายในวิธีการ print ของออปเจคต่าง ๆนั้นเป็นอย่างไรก็จะอยู่ภายในออปเจคนั้น

จะเห็นได้ว่าเราก็ไม่ต้องตั้งชื่อโพรซีเยอร์หรือวิธีการให้แตกต่างกันไปและไม่ต้องมีประโยค case ในการพิจารณาเรียกใช้วิธีการต่าง ๆ เพราะว่าส่วนของคอมไพเลอร์จะเป็นผู้พิจารณาพารามิเตอร์ของวิธีการที่เราเรียกใช้และไปเรียกวิธีการของออปเจคที่เหมาะสมให้เอง

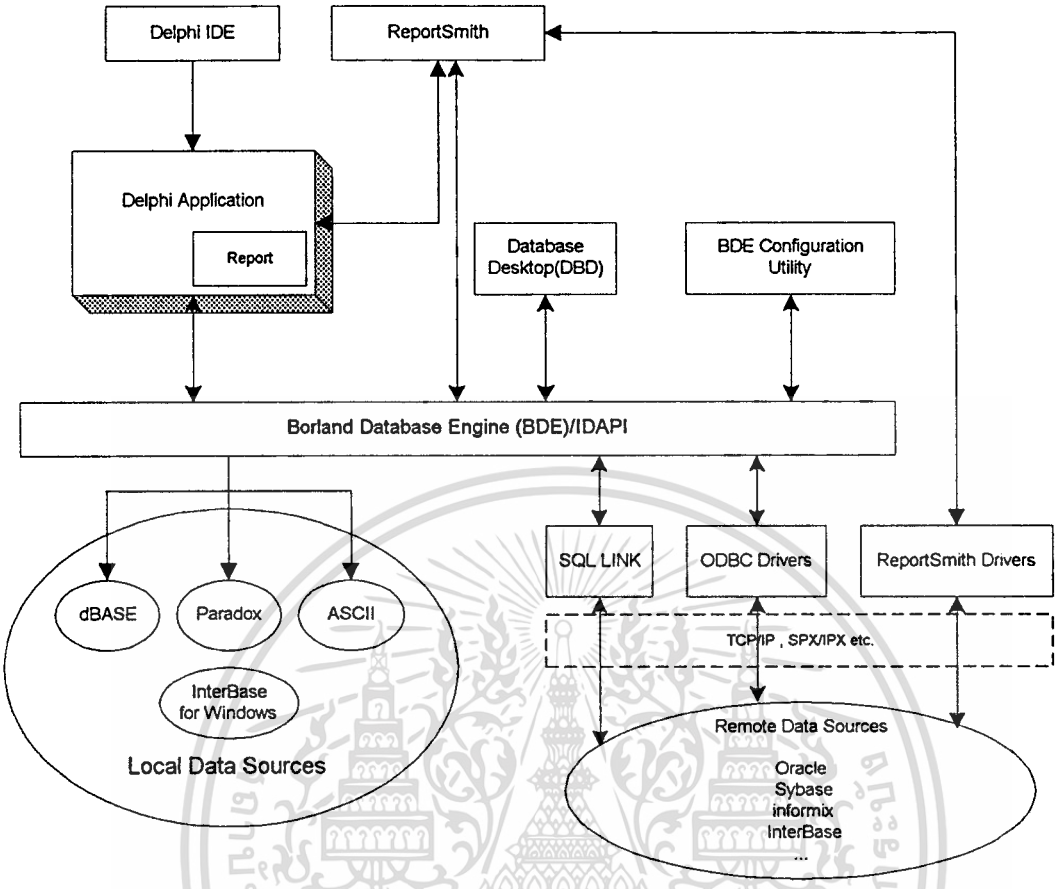
2.8 สภาพแวดล้อมในการพัฒนาระบบ

2.8.1 ความสามารถและคุณลักษณะพิเศษทางด้านงานฐานข้อมูลของDelphi

การสร้างแอพพลิเคชั่นงานฐานข้อมูลโดยDelphiจะอาศัย

- เครื่องมือพัฒนางานด้านฐานข้อมูลของDelphi(Delphi database development tool)
- ดาต้า-แอคเซส คอมโพเนนท์ของDelphi(Delphi data-access component)
- ดาต้า-อแวร์ จียูไอ คอมโพเนนท์ของDelphi(Delphi data-aware GUI component)

โดยใช้คอมโพเนนท์ในการติดต่อกับบอร์เครือข่ายท้องถิ่นด์ ดาต้าเบส เอนจิ้น หรือบีดีอี (Borland Database Engine , BDE) ซึ่งจะทำหน้าที่ติดต่อกับฐานข้อมูลอีกที ภาพดังต่อไปนี้จะแสดงความสัมพันธ์ของ เครื่องมือต่าง ๆ และ แอพพลิเคชั่นงานฐานข้อมูล กับ บีดีอี และ แหล่งข้อมูล (Datasource)



รูปที่ 2-17 แสดงสถาปัตยกรรมด้านงานฐานข้อมูลของDelphi

ตารางดังต่อไปนี้สรุปคุณลักษณะพิเศษทางงานฐานข้อมูลของDelphi

ตารางที่ 2-1 สรุปคุณลักษณะพิเศษทางด้านงานฐานข้อมูล

เครื่องมือ	วัตถุประสงค์
Data Access Component	เข้าถึง ฐานข้อมูล,ตาราง,สตอร์โพรซิเดอร์
Data Control Component	ช่วยเหลือให้ผู้ใช้ติดต่อกับตารางในฐานข้อมูล
Database Desktop	สร้างตาราง,สร้างอินเดกซ์ และ ทำคิวรี่สำหรับพาราดอกซ์ และ ดีเบส ,เข้าถึง และ แก้ไขเพิ่มเติมข้อมูลสำหรับฐานข้อมูลเอสคิวแอล
ReportSmith	สร้าง,เรียกดู และพิมพ์รายงาน
Borland Data Engine (BDE)	เข้าถึงข้อมูลในตาราง พาราดอกซ์ และ ดีเบส และ จากดาต้าเบสเซิร์ฟเวอร์ของโลคอลInterBase (Local InterBASE)
BDEConfiguration	สร้างและจัดการเกี่ยวกับเอเลียส(alias)ที่ใช้ในการติดต่อกับฐานข้อมูลโดยบีดีอี

Local InterBase Server	เป็นเอสคิวแอลเซิร์ฟเวอร์แบบผู้ใช้คนเดียว แบบมัลติอินสแตนท์(multi-instance) เพื่อไว้สร้างและทดสอบแอพพลเคชั่น ก่อนที่จะย้ายแพลตฟอร์มไปใช้ผลิตภัณฑ์อื่น เช่น Oracle, Sybase, Informix หรือ InterBASE บนรีโมทเซิร์ฟเวอร์
Interbase SQL Link	เป็นเนทีฟไดรเวอร์ที่เชื่อมต่อแอพพลิเคชั่น กับ โลคอล InterBaseเซิร์ฟเวอร์

และตารางต่อไปนี้จะแสดงรายการของคุณลักษณะพิเศษทางงานฐานข้อมูลเพิ่มเติมที่มีใน Delphi ฉบับไคลเอนท์เซิร์ฟเวอร์ คุณลักษณะพิเศษเหล่านี้จะเพิ่มความสามารถในการเข้าถึงรีโมท เอสคิวแอล ดาต้าเบสเซิร์ฟเวอร์ ตัวอย่างเช่น Sybase,Microsoft SQL Server,Informix และ InterBase

ตารางที่ 2-2 คุณลักษณะพิเศษเพิ่มเติมของ Delphi ฉบับไคลเอนท์เซิร์ฟเวอร์

เครื่องมือ	วัตถุประสงค์
SQL Drivers	ทั้ง เอสคิวแอลลิงค์ (SQL Links) และ รีพอร์ทสมิธ (ReportSmith) จะเป็นเนทีฟไดรเวอร์ทำหน้าที่ให้การติดต่อระหว่างแอพพลิเคชั่นกับเอสคิวแอล เซิร์ฟเวอร์ เช่น Oracle,Sybase,ไมโครซอฟท์ เอสคิวแอล เซิร์ฟเวอร์, อินฟอร์ม และInterBase
Visual Query Builder	สร้างคำสั่งภาษาเอสคิวแอลโดยจัดการกับตารางและคอลัมน์แบบวิชวล

2.8.2 สถาปัตยกรรมฐานข้อมูลของDelphi

Delphi ใช้ออปเจคโอเรียนเต็ด คอมโพเนนท์(Object-Oriented Component) ในการสร้างแอพพลิเคชั่นงานฐานข้อมูลเช่นเดียวกับแอพพลิเคชั่นอื่นๆที่ไม่ใช่แอพพลิเคชั่นงานฐานข้อมูล ดาต้าเบสคอมโพเนนท์ (Database Component) จะเหมือนกับคอมโพเนนท์มาตรฐานทั่วไป ที่มีแอททริบิวท์ หรือเรียกว่า พรอพเพอร์ตี้ (Property) ซึ่งจะถูกตั้งค่าโดยโปรแกรมเมอร์ในขณะที่ทำการออกแบบ และอาจจะตั้งค่าในขณะที่กำลังรันโปรแกรมก็ได้

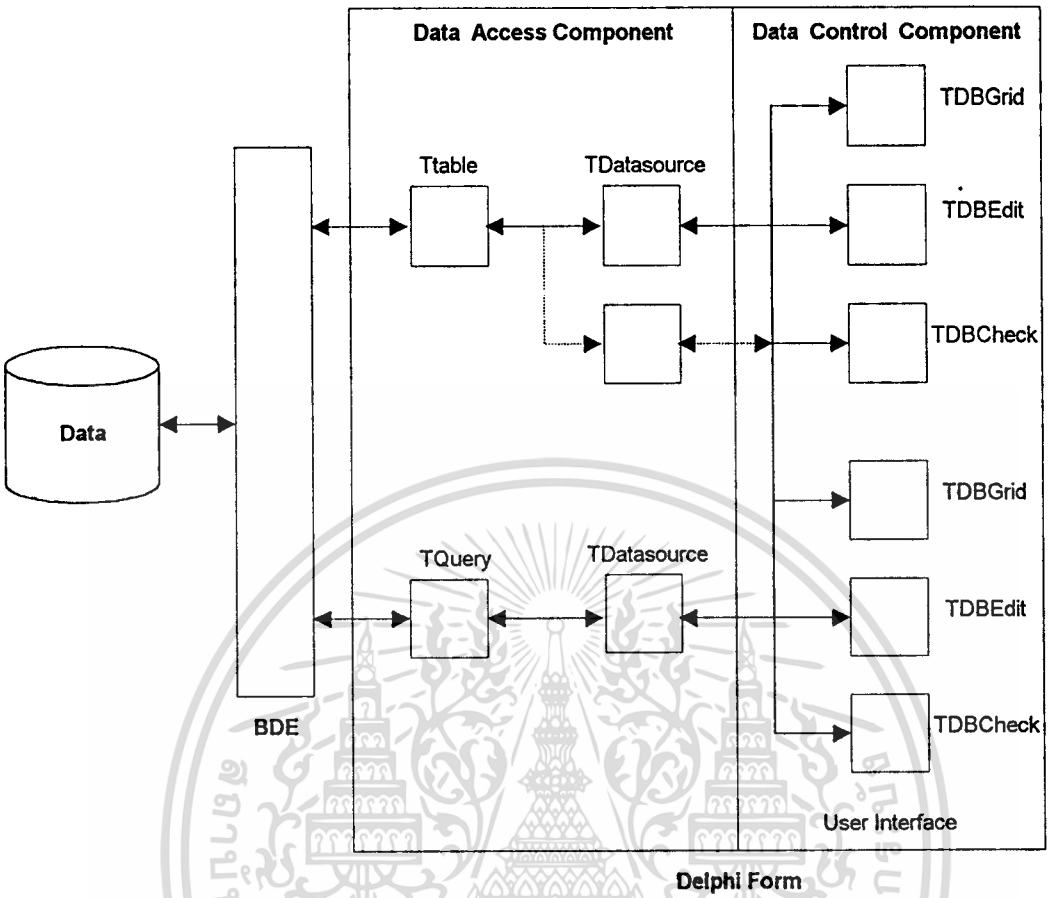
ดาต้าเบสคอมโพเนนท์จะมีค่าที่ถูกตั้งไว้ให้แล้ว (Default) เพื่อให้มันสามารถทำงานได้โดยไม่ต้องเขียนโปรแกรมเพิ่มหรือเพิ่มเพียงเล็กน้อย

Delphi ประกอบด้วย 2 คอมโพเนนท์พาเลท (Component Palette) ที่มีดาต้าเบสคอมโพเนนท์ คือ

- หน้าสำหรับคอมโพเนนท์ที่เข้าถึงเข้ามูล (Data-Access Page) จะมีออปเจคที่ทำให้การเข้าถึงข้อมูลง่ายขึ้น โดยการเอนแคปซูเลท(Encapsulate)แหล่งข้อมูล เช่น ฐานข้อมูลที่จะติดต่อ,ตารางในฐานข้อมูล หรือ ฟิลด์(Field)ของข้อมูลที่ต้องการ ตัวอย่างของออปเจคเหล่านี้ได้แก่ TTable , TQuery , TDatasource และ Treport
- หน้าสำหรับคอมโพเนนท์ที่ควมคุมข้อมูล (Data-Control Page) จะมีคอมโพเนนท์ไว้ติดต่อกับผู้ใช้เพื่อแสดงข้อมูลในรูปแบบต่าง ๆ ซึ่งจะเหมือนกับคอมโพเนนท์มาตรฐานที่ใช้ติดต่อกับผู้ใช้ เพียงแต่ข้อมูลเหล่านั้นมาจากตารางในฐานข้อมูล ตัวอย่างของคอมโพเนนท์ควบคุมข้อมูลที่ใช้บ่อย ๆ ได้แก่ TDBEdit , TDBNavigator และ TDBGrid

ดาต้าเซทคอมโพเนนท์(Dataset Component) เช่น TTable , TQuery และ TStoreProc จะมองไม่เห็นในขณะรันโปรแกรม แต่มีไว้ให้แอพพลิเคชั่นติดต่อกับข้อมูลของมันผ่านบีดีอี คอมโพเนนท์ควบคุมข้อมูลจะติดต่อกับดาต้าเซทคอมโพเนนท์โดยผ่านทาง Tdatasource คอมโพเนนท์ เพื่อให้การติดต่อกับข้อมูลแบบมองเห็นได้

จากภาพแสดงให้เห็นว่า คอมโพเนนท์เข้าถึงข้อมูลกับคอมโพเนนท์ควบคุมข้อมูลสัมพันธ์กับข้อมูล ,สัมพันธ์ซึ่งกันและกัน และสัมพันธ์กับส่วนติดต่อกับผู้ใช้แอพพลิเคชั่นอย่างไร



รูปที่ 2-18 สถาปัตยกรรมของดาต้าเบสคอมโพเนนท์

2.8.3 หลักการพัฒนาแอพพลิเคชั่นงานฐานข้อมูล

2.8.3.1 รูปแบบของการพัฒนา

เนื่องจากการออกแบบแอพพลิเคชั่น จะต้องขึ้นกับโครงสร้างฐานข้อมูลที่จะเข้าถึง โดยฐานข้อมูลจะต้องถูกกำหนดล่วงหน้าก่อนที่จะทำการพัฒนาแอพพลิเคชั่น

รูปแบบในการพัฒนาแอพพลิเคชั่นงานฐานข้อมูลของ Delphi เป็นไปได้ 4 รูปแบบคือ

- ไม่มีฐานข้อมูลอยู่ก่อน หรือจำเป็นต้องทำการกำหนดใหม่หมด สามารถทำได้โดย

  ♦ ใช้ ดาต้าเบส เดสค์ทอป ยูทิลิตี้ (Database Desktop Utility) เพื่อกำหนดตารางแบบพาราดอกซ์และดีเบส

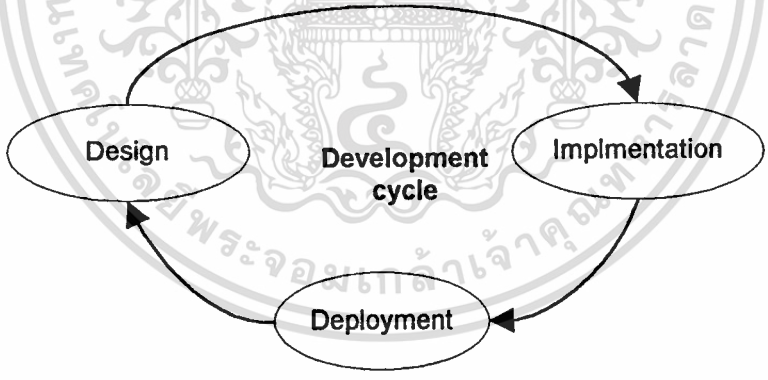
  ♦ สำหรับ เอสคิวแอลเซิร์ฟเวอร์ สามารถใช้เครื่องมือที่เป็นของเซิร์ฟเวอร์ หรือของดาต้าเบส เดสค์ทอปก็ได้ ตัวอย่างเช่น สำหรับ โลคอลอินเตอร์เบสเซิร์ฟเวอร์ สามารถใช้ วินโดว์ ไอเอสคิวแอล (Windows ISQL) ในการกำหนดฐานข้อมูล

- มีฐานข้อมูลปรากฏอยู่บนแหล่งข้อมูล ที่เป็นเดสค์ทอป หรือ เป็นเครือข่ายท้องถิ่น (LAN) และฐานข้อมูลเข้าถึงข้อมูล ณ ที่แห่งนั้น ถ้าบีดีอีและแหล่งข้อมูลอยู่บนเครื่องเดียวกันกับแอพพลิเคชั่นแล้ว แอพพลิเคชั่นนั้นจะเป็นแบบแสตนด์อโลน (Stand-Alone)
- ฐานข้อมูลปรากฏบนแหล่งข้อมูลแบบเดสค์ทอป และจะถูกอัพไซซ์ (Upsize) เป็น เอสคิวแอลเซิร์ฟเวอร์ แบบนี้จะเรียกว่าเป็นโลคอลเอสคิวแอลเซิร์ฟเวอร์
- ฐานข้อมูลอยู่บนเอสคิวแอลเซิร์ฟเวอร์ และแอพพลิเคชั่นจะเข้าถึงข้อมูลที่อยู่ที่เซิร์ฟเวอร์ แบบนี้จะเรียกว่าเป็นไคลเอ็นท์เซิร์ฟเวอร์แบบมาตรฐาน

2.8.3.2 วงจรในการพัฒนาแอพพลเคชั่นงานฐานข้อมูล

การพัฒนาแอพพลิเคชั่นงานฐานข้อมูลมี 3 ขั้นตอนหลักคือ

- การออกแบบ และ ทำต้นแบบ (Prototyping)
- การสร้าง (Implement)
- การส่งมอบระบบใช้งานจริง (Deployment) และการบำรุงรักษา (maintenance)



รูปที่ 2-19 วงจรในการพัฒนา

สำหรับแอพพลิเคชั่นแบบไคลเอนท์เซิร์ฟเวอร์ งานของฐานข้อมูล และงานของแอพพลิเคชั่นจะแยกกันเด่นชัดเนื่องจากจะรันบนแพลตฟอร์มที่ต่างกัน และระบบปฏิบัติการมักจะต่างกันด้วย เช่น ยูนิกซ์เซิร์ฟเวอร์ และวินโดว์ 3.1 ไคลเอนท์

ดังนั้นในการพัฒนา ความรับผิดชอบในการแบ่งว่างานใดควรทำโดยดาต้าเบสเซิร์ฟเวอร์ งานใดควรทำโดยไคลเอนท์แอพพลิเคชั่นจะอยู่ในขั้นตอนการออกแบบ โดยทั่วไปแล้วจะมีการแบ่งงานกันอย่างชัดเจนระหว่าง 2 ฝั่ง แต่กรรมวิธีทางฐานข้อมูลเช่น สตอร์โพรซิเดอร์บางอย่างก็สามารถทำได้โดยไคลเอ็นท์แอพพลิเคชั่น ดังนั้นจึงขึ้นกับลักษณะของการนำ

ระบบไปใช้จริงที่คาดหวังไว้ ,ความต้องการของแอพพลิเคชั่น และข้อควรพิจารณาอื่น ๆ ดังนั้นในการออกแบบสามารถจะกำหนดไว้การทำงานจะอยู่ฝั่ง ไคลเอนท์ หรือฝั่งเซิร์ฟเวอร์

2.8.4 การส่งมอบแอพพลิเคชั่นเพื่อนำไปใช้จริง (Deploying an application)

หมายถึงการนำแอพพลิเคชั่นไปให้ผู้ใช้จริงใช้ และจัดหาซอฟแวร์ที่จำเป็นที่ผู้ใช้จะต้องมีไว้ใช้ในการรันแอพลิเคชั่น ในสภาวะแวดล้อมของผู้ใช้ สำหรับแอพพลิเคชั่นที่ไม่ใช่งานฐานข้อมูลของDelphi จะใช้เพียงไฟล์ .EXE เท่านั้น เนื่องแอพพลิเคชั่นของDelphi ไม่ต้องการ ตัวแปลภาษาขณะทำงาน (run-time interpreter) หรือ ไดนามิก ลิงค์ ไลบรารี (DLL)

2.8.4.1 ไฟล์ที่ต้องการในการส่งมอบแอพพลิเคชั่นเพื่อนำไปใช้จริง (แบบสแตนด์อโลน)

โดยทั่วไปในการส่งมอบแอพพลิเคชั่นงานฐานข้อมูลเพื่อนำไปใช้จริง จะต้องสร้างแพคเกจ (Package) ที่ประกอบด้วยไฟล์ทั้งหมดที่ผู้ใช้ต้องการในการรันแอพพลิเคชั่นและเข้าถึงแหล่งข้อมูล

ไฟล์เหล่านั้นได้แก่

- ไฟล์ .EXE และ ไฟล์ .DLL (ถ้ามี) ของแอพพลิเคชั่น
- ไฟล์ที่เป็นส่วนประกอบต่าง ๆ เช่น ไฟล์ .README , ไฟล์ .HLP หรือไฟล์สำหรับ ออนไลน์เฮลพ์ (OnLine Help)
- บีดีอี ซัพพอร์ท สำหรับการเข้าถึงฐานข้อมูล (เดสค์ทอป หรือ เซิร์ฟเวอร์)
- รีพอร์ทสมิธ รันไทม์ (ReportSmith Runtime)สำหรับการพิมพ์รายงาน
- ถ้าแอพพลเคชั่นใช้ วีบีเอ็กซ์คอนโทรล (VBX Control) จะต้องใช้ ไฟล์ BIVBX11.DLL

2.8.4.2 การส่งมอบบีดีอีซัพพอร์ทเพื่อนำไปใช้งานจริง (Deploying BDE support)

เมื่อจะส่งมอบแอพพลิเคชั่นงานฐานข้อมูล จะต้องมั่นใจว่าแพลตฟอร์มของเครื่องไคลเอนท์ได้รับการติดตั้ง บีดีอีรุ่นที่ถูกต้อง ใน Delphi จะประกอบด้วย บีดีอีที่แจกจ่ายต่อได้ (Redistributable BDE) ซึ่งจะมียูทิลิตี้ในการติดตั้งตัวมันเอง ดังนั้นในการส่งมอบแอพพลิเคชั่นเพียงแต่รวมก๊อปปี้ของแผ่น บีดีอีที่แจกจ่ายต่อได้เข้าไว้ด้วยเท่านั้น

ตารางที่ 2-3 แสดงไฟล์บีดีอีที่แจกจ่ายต่อได้

IDAPI01.DLL	BDE API DLL
IDBAT01.DLL	BDE Batch Utilities DLL
IDQRY01.DLL	BDE Query DLL

IDASCI01.DLL	BDE ASCII Driver DLL
IDPDX01.DLL	BDE Paradox Driver DLL
IDDBAS01.DLL	BDE dBASE Driver DLL
IDR10009.DLL	BDE Resources DLL
ILD01.DLL	Language Driver DLL
IDODBC01.DLL	BDE ODBC Socket DLL
ODBC.NEW	Manager DLLMicrosoft ODBC Driver , version 2.0
ODBCINST.NEW	Microsoft ODBC Driver installation DLL , version 2.0
TUTULITY.DLL	BDE Tutility DLL
BDECFG.EXE	BDE Configuration Utility
BDECFG.HLP	BDE Configuration Utility help
IDAPI.CFG	BDE (IDAPI) Configuration File

- ไดรเวอร์ภาษา (Language driver)

ดีบีอีจะมีไดรเวอร์ภาษามาให้เพื่อให้แอพพลเคชั่นเลือกใช้ โดยไดรเวอร์ภาษา DLL จะโหลดไดรเวอร์ที่ระบุโดยตารางของพาราดอกซ์ และดีเบส หรือใน IDAPI.CFG สำหรับดาต้าเบสเซิร์ฟเวอร์ ไดรเวอร์ภาษาจะเป็นไฟล์ .LD ถูกติดตั้งอยู่ในซับไดเรคทอรี่ LANGDRV ของบีดีอีไดเรคทอรี่

- โอดีบีซี ซอคเกต (ODBC Socket)

บีดีอีจะมาพร้อมกับโอดีบีซี ซอคเกต ซึ่งถูกรับรองโดย ไมโครซอฟท์ โอดีบีซี 2.0 ไดรเวอร์ เมเนเจอร์ (Microsoft ODBC 2.0 Driver Manager)

2.8.4.3 ไฟล์ที่ต้องการในการส่งมอบแอพพลิเคชั่นเพื่อนำไปใช้จริง (แบบไคลเอนเซิร์ฟเวอร์)

นอกเหนือจากไฟล์ที่ได้กล่าวมาแล้วในหัวข้อ 2.5.4.2 การส่งมอบแอพพลิเคชั่นแบบไคลเอนท์เซิร์ฟเวอร์จะต้องทำการติดตั้ง บอร์แลนด์ เอสคิวแอล ลิงค์ ที่ถูกต้องเหมาะสม ซึ่งไม่ใช่ส่วนหนึ่งของ บีดีอี และจะทำการติดตั้งแยกกันต่างหาก

เซิร์ฟเวอร์แต่ละชนิดจะมีกลุ่มของไฟล์สำหรับ เอสคิวแอลลิงค์ และเพิ่มไฟล์ที่ถูกใช้โดยทุกเซิร์ฟเวอร์ คือ BRLOM800.LD ซึ่งเป็นไดรเวอร์ภาษาชื่อโรมัน8

Oracle

ไฟล์ดังต่อไปนี้ทำให้เอสคิวแอลลิงค์ติดต่อกับเซิร์ฟเวอร์ของOracleได้ และนอกจากนี้แอพพลเคชั่นจะต้องมี ออราเคิลไคลเอนท์ไฟล์ (Oracle client file) สำหรับเชื่อมต่อกับโปรโตคอลสื่อสารระดับล่างอย่างเช่น TCP/IP

ตารางที่ 2-4 ไฟล์ออราเคิลเอสคิวแอลลิงค์

ชื่อไฟล์	คำอธิบาย
SQL_ORA.DLL	Borland SQL Link Oracle Driver
SQL_ORA.HLP	Online help file
SQL_ORA.CNF	BDE Configutation File for Oracle Driver
ORA6WIN.DLL	Oracle Version 6.x client-side DLL
ORA7WIN.DLL	Oracle Version 7.x client-side DLL
SQL13WIN.DLL	Oracle client-side DLL
SQLWIN.DLL	Oracle client-side DLL
COREWIN.DLL	Oracle client-side DLL
ORAWE850.LD	Language driver based on DOS code page 850

Sybase และ Microsoft SQL Server

ไฟล์ดังต่อไปนี้ทำให้ เอสคิวแอลลิงค์ ติดต่อกับ เซิร์ฟเวอร์ของSybase ได้ และนอกจากนี้แอพพลเคชั่นจะต้องมี ไซเบสไคลเอนท์ไฟล์ (Sybase client file) สำหรับเชื่อมต่อกับโปรโตคอลสื่อสารระดับล่างอย่างเช่น TCP/IP

ตารางที่ 2-5 ไฟล์ไซเบสเอสคิวแอลลิงค์

SQL_SS.DLL	Borland SQL Links Sybase Driver
SQL_SS.HLP	Borland SQL Links Sybase Driver Help
SQL_SS.CNF	BDE Configuration File for Sybase Driver
W3DBLIB.DLL	Sybase/Microsoft SQL Server client-side DLL
DBNMP3.DLL	Sybase/Microsoft SQL Server client-side DLL for Named Pipes
SYDC437.LD	Language driver based on DOS code page 850
SYDC850.LD	Language driver based on DOS code page 437

Informix

ไฟล์ดังต่อไปนี้ทำให้ เอสคิวแอลลิงค์ ติดต่อกับ เซิร์ฟเวอร์ของInformixได้ และนอกจากนี้แอพพลเคชั่นจะต้องมี อินฟอร์มิกไคลเอนท์ไฟล์ (Informix client file) สำหรับเชื่อมต่อกับโปรโตคอลสื่อสารระดับล่างอย่างเช่น TCP/IP

ตารางที่ 2-6 ไฟล์ อินฟอร์มิกเอสคิวแอลลิงค์

SQLD_INF.DLL	Borland SQL Link Informix Driver
SQLD_INF.HLP	Online Help File
SQL_INF.CNF	BDE Configuaration File for Informix Driver
LDLLSQLW.DLL	Informix client-side DLL
ISAM.IEM	Informix error message file
RDS.IEM	Informix error message file
SECURITY.IEM	Informix error message file
SQL.IEM	Informix error message file

InterBase

ไฟล์ดังต่อไปนี้ทำให้ เอสคิวแอลลิงค์ ติดต่อกับ เซิร์ฟเวอร์ของ InterBaseได้ และนอกจากนี้แอพพลเคชั่นจะต้องมี อินเตอร์เบสไคลเอนท์ไฟล์ (InterBASE client file) สำหรับเชื่อมต่อกับโปรโตคอลสื่อสารระดับล่างอย่างเช่น TCP/IP

ตารางที่ 2-7 ไฟล์อินเตอร์เบสเอสคิวแอลลิงค์

SQLD_IB.DLL	Borland SQL Link InterBase Driver
SQLD_IB.HLP	Borland SQL Link InterBase Driver Help
SQL_IB.CNF	BDE Configuration File for InterBase Driver
CONNECT.EXE	InterBase connection diagnostic tool
CONNECT.HLP	InterBase Windows connection diagnostic help file
GDS.DLL	InterBase API DLL
REMOTE.DLL	InterBase Networking interface DLL
INTERBAS.MSG	InterBase error message file

2.8.5 การสร้างแอพพลิเคชั่นบนไคลเอนท์เซิร์ฟเวอร์โดยใช้ Delphi

ประเด็นสำคัญที่ควรคำนึงในการสร้างแอพพลิเคชั่นบนไคลเอนท์เซิร์ฟเวอร์ได้แก่

2.8.5.1 ความสามารถในการย้ายแพลทฟอร์มเทียบกับประสิทธิภาพ(Portability versus Optimization)

ความสามารถในการย้ายแพลทฟอร์ท หมายถึง ความยืดหยุ่นของฐานข้อมูลและแอพพลิเคชั่นที่จะรันบนเซิร์ฟเวอร์ที่แตกต่างกันได้

ประสิทธิภาพ หมายถึงว่า แอพพลิเคชั่นสามารถใช้ประโยชน์จากคุณลักษณะพิเศษต่าง ๆ ของระบบนั้นได้มากขนาดไหน

ความสามารถในการรองรับเซิร์ฟเวอร์ได้หลายชนิด (Server Portability)

ในการออกแบบแอพพลิเคชั่นบนไคลเอนท์เซิร์ฟเวอร์ จะมีข้อได้เปรียบเสียเปรียบระหว่าง ความสามารถในการเคลื่อนย้ายแพลตฟอร์ม กับประสิทธิภาพ เนื่องจากการใช้คุณลักษณะพิเศษของเซิร์ฟเวอร์ชนิดใดโดยเฉพาะเจาะจง จะทำให้ประสิทธิภาพเพิ่มขึ้น แต่สามารถในการเคลื่อนย้ายไปแพลตฟอร์มอื่นจะลดลง

ความสามารถในการเคลื่อนย้ายแพลตฟอร์มของ Delphi ได้มาจากการใช้ TTable และ TQuery โดย Ttable นั้นจะมีความสะดวกที่ในการเปลี่ยนชนิดเซิร์ฟเวอร์ แต่การใช้ TQuery เพื่อส่งคำสั่งภาษาเอสคิวแอลไปให้เซิร์ฟเวอร์นั้นจะมีประสิทธิภาพเพิ่มขึ้น แต่ถ้าจะให้มีความสามารถในการเคลื่อนย้าย ไวยากรณ์ของภาษาเอสคิวแอลก็จะต้องเป็นไปตามมาตรฐาน ANSI เพื่อที่จะได้ไม่ต้องแก้ไขในเวลาที่เปลี่ยนเซิร์ฟเวอร์

ความสามารถในการรองรับการสื่อสารแบบไคลเอนท์เซิร์ฟเวอร์

ขึ้นกับความต้องการของแอพพลิเคชั่น ซึ่งอาจจะต้องรองรับโปรโตคอลการสื่อสารได้หลายแบบ ตัวอย่างเช่น TCP/IP และ NOVELL SPX เพื่อจะแน่ใจได้ว่า ซอฟท์แวร์สื่อสารที่ถูกติดตั้งไว้ในแพลตฟอร์มของไคลเอนท์สามารถใช้ได้

2.8.5.2 การติดต่อกับดาต้าเบสเซิร์ฟเวอร์

บอร์แลนด์เอสคิวแอลลิงค์ จะทำให้แอพพลิเคชั่นติดต่อผ่านบีดีอีไปยังรีโมทดาต้าเบสเซิร์ฟเวอร์ได้ เอสคิวแอลลิงค์ไดรเวอร์จะถูกใช้ในการเชื่อมต่อกับ Oracle ,Sybase ,Microsof SQL Server และInformix

โดยใช้ บีดีอี คอนฟิกกูเรชั่น ยูทิลิตี้ ทำการตั้งค่าเอเลียสสำหรับแต่ละแหล่งข้อมูลที่แอพพลิเคชั่นต้องการติดต่อ ค่าเอเลียสเหล่านี้จะถูกเลือกเป็นค่าของพรอพเพอร์ตี้DatabaseName ของ TTable และTQuery

การเชื่อมต่อ

ไคลเอนท์แอพพลิเคชั่น จะใช้โปรโตคอลสื่อสารใดก็ได้ที่เซิร์ฟเวอร์รองรับ โดยจะต้องกำหนดค่าเอสคิวแอลลิงค์ไดรเวอร์ ให้ตรงกับโปรโตคอลที่ต้องการ

- การใช้ TCP/IP

การใช้ TCP/IP เป็นโปรโตคอลเพื่อการติดต่อกับดาต้าเบสเซิร์ฟเวอร์ต้องคำนึงถึง

◆ จะต้องติดตั้ง ซอฟท์แวร์สื่อสาร TCP/IP และ วินซอคไดรเวอร์ (WINSOCK Driver)ที่ถูกต้องบนเครื่องไคลเอนท์

◆ ไอพี แอดเดรส (IP Address) ของเซิร์ฟเวอร์จะต้องถูกระบุในไฟล์ HOST ของไคลเอนท์

◆ หมายเลขพอร์ตของเซิร์ฟเวอร์ จะอยู่ในไฟล์ SERVICE ของไคลเอนท์

◆ แอพพลิเคชั่นจะค้นหาไดเรคทอรี่ที่มีไฟล์ .DLL ที่มันต้องการ จะต้องตรวจสอบพาธ (PATH)ในไฟล์ AUTOEXEC.BAT

- การใช้โอดีบีซี

แอพพลิเคชั่นของ Delphi สามารถเข้าถึง โอดีบีซีดาต้าซอร์ส เช่น ดีบีทู (DB2) ,บีทรีฟ (Btrieve) หรือ ไมโครซอฟท์ แอคเซส (Microsoft Access) ผ่านทางบีดีอี โดยใช้ยูทิลิตี้ในการจัดการคอนฟิกของบีดีอีในการตั้งค่าการติดต่อโอดีบีซี ไดรเวอร์

การติดต่อโอดีบีซี ไดรเวอร์ จะต้องมี

◆ โอดีบีซี ไดรเวอร์ ที่ได้มาจากผู้ขายเซิร์ฟเวอร์ (Vendor-Supplied)

◆ ไมโครซอฟท์ โอดีบีซี ไดรเวอร์ เมเนเจอร์ (Microsoft ODBC Driver Manager)

◆ บีดีอี เอเลียส ที่ติดตั้งโดยใช้ยูทิลิตี้ในการจัดการคอนฟิกของบีดีอี หรือโดยใช้Delphi

2.8.5.3 การควบคุมทรานสแอคชั่น (Transaction Control)

แอพพลิเคชั่นในDelphiสามารถควบคุมทรานสแอคชั่นได้ 2 รูปแบบคือ

- อย่างเป็นนัย(Implicitly) Delphi จะเริ่มต้นและคอมมิท(commit) ทรานแสคชั่นโดยอัตโนมัติเมื่อแอพพลิเคชั่นเรียกคำสั่งโพสท์ (Post Method)

- อย่างเปิดเผย(Explicitly) ระดับของการควบคุมขึ้นกับความต้องการของแอพพลิเคชั่น สามารถทำได้ 2 ทางคือ

◆ **ผ่าน TDatabase คอมโพเนนท์**

สามารถเลือกวิธีควบคุมได้ดังนี้

* ***StartTransaction*** - เริ่มทรานแสคชั่นตามระดับไอโซเลชั่น(Isolation Level) ที่ระบุไว้ และถ้าทรานแสคชั่นกำลังทำงานอยู่ Delphi จะรายงานความผิดพลาดที่เกิดขึ้น

* ***Commit*** - คอมมิททรานแสคชั่นปัจจุบันที่กำลังแอคทีฟอยู่บนฐานข้อมูล และถ้าไม่มีทรานแสคชั่นใดแอคทีฟอยู่เลย Delphiจะรายงานความผิดพลาดที่เกิดขึ้น

* ***Rollback*** - โรลแบคทรานแสคชั่นปัจจุบันที่กำลังแอคทีฟ และการแก้ไขใดๆ ในฐานข้อมูลที่เกิดขึ้นหลังจากการคอมมิทครั้งล่าสุดจะถูกยกเลิก

ถ้าแอพพลิเคชั่นต้องการใช้คุณลักษณะพิเศษในการควบคุมทรานแสคชั่นของเซิร์ฟเวอร์ตัวใดโดยเฉพาะ จะต้องตั้งค่า SQLPASSTHRUMODE ให้เป็น NOT SHARE เพื่อไม่ให้การเรียกใช้พาสธรูเอสคิวแอล (Passthru)มีผลกระทบต่อทรานแสคชั่นอื่น

◆ **ผ่านทาง พาสธรู เอสคิวแอล (Passthrough SQL)ใน TQuery**

โดยวิธีนี้แอพพลิเคชั่นจะอาศัยการควบคุมทรานสแอคชั่นที่เป็นของเซิร์ฟเวอร์แต่ละชนิด จึงจะต้องเข้าใจว่าเซิร์ฟเวอร์แต่ละควบคุมทรานสแอคชั่นอย่างไร การตั้งค่า SQLPASSTHRUMODE เพื่อเป็นการกำหนดว่าการเรียกใช้ พาสธรู เอสคิวแอล กับการเรียกใช้บีดีอีมาตรฐาน จะใช้การติดต่อกับฐานข้อมูลร่วมกันหรือไม่ โดยสามารถตั้งค่าได้ดังนี้

* ***SHARED AUTOCOMMIT*** - จะทำการคอมมิททีละโอเปอเรชั่นทำกับข้อมูล 1 โรว์ ซึ่งเหมาะกับ ดาต้าเบสเดสค์ทอป แต่ไม่เหมาะกับเซิร์ฟเวอร์ เพราะต้องสตาร์ทและคอมมิททุก ๆ ทรานแสคชั่นสำหรับแต่ละโรว์ ซึ่งจะเป็นภาระหนักทางด้านการจราจรในเครือข่าย (Network Traffic)

* ***SHARED NOAUTOCOMMIT*** - แอพพลิเคชั่นจะต้องกำหนดการสตาร์ทและคอมมิทอย่างเด่นชัด การตั้งค่าแบบนี้อาจทำให้เกิดความผิดพลาดได้ในสภาวะแวดล้อมที่มีผู้ใช้จำนวนมาก และมีการทำงานสูงเมื่อผู้ใช้หลายคนทำการแก้ไขข้อมูลโรว์เดียวกัน

* ***NOT SHARED*** - หมายความว่า พาสธรู เอสคิวแอล กับ งานของ Delphi จะติดต่อกับฐานข้อมูลโดยแยกการติดต่อออกจากกัน

2.8.5.4 การอัพไซซ์ (Upsizing)

คือการโยกย้ายแอพพลิเคชั่นบนเดสค์ทอปไปเป็นแอพพลิเคชั่นบนไคลเอนท์เซิร์ฟเวอร์

การอัพไซซ์มี 2 รูปแบบใหญ่ๆ คือ

- การอัพไซซ์ฐานข้อมูลจากเดสค์ทอปไปเป็นเซิร์ฟเวอร์
- การอัพไซซ์แอพพลิเคชั่นโดยเน้นพิจารณาหลักการไคลเอนท์เซิร์ฟเวอร์

<u>การอัพไซซ์ฐานข้อมูล</u>

การย้ายฐานข้อมูลประกอนด้วยขั้นตอนดังต่อไปนี้

- ระบุเมต้าดาต้า (Metadata) บนเซิร์ฟเวอร์ โดยขึ้นกับโครงสร้างฐานข้อมูลแบบเดสค์ทอปที่มีอยู่
- ย้ายข้อมูลจากเครื่องเดสค์ทอปไปยังเครื่องเซิร์ฟเวอร์
- ประเด็นสำคัญที่จะต้องพิจารณาได้แก่
  - ความแตกต่างของดาต้าไทป์ (Data Type)
  - ความปลอดภัย (Security) และความคงสภาพ (Intrigity)ของข้อมูล
  - การควบคุมทรานแสคชั่น
  - การพิสูจน์ความถูกต้องของข้อมูล (Data Validation)
  - การล็อค (Locking)

ใน Delphi จะมีการอัพไซซ์ฐานข้อมูลได้ 2 วิธี คือ

- ใช้ ดาต้าเบส เดสค์ทอป ยูทิลิตี้ โดยเลือกยูทิลิตี้ก็อปปี้ตาราง(Tools | Utilities | Copy) จากตารางเดสค์ทอปไปเป็นรูปแบบเอสคิวแอล (SQL format)
- สร้างแอพพลิเคชั่นโดยใช้ คอมโพเนนท์ TBatchMove

ทั้ง 2 วิธีนี้จะทำการก็อปปี้โครงสร้างตาราง และย้ายฐานข้อมูลไปยังเซิร์ฟเวอร์เป้าหมาย โดยอาจจะมีการแก้ไขตาราง ซึ่งขึ้นอยู่กับฐานข้อมูล ตัวอย่างเช่น ชนิดของข้อมูล (Datatype) อาจจะไม่ตรงกัน

<u>การอัพไซซ์แอพพลิเคชั่น</u>

ตามทฤษฎีแล้ว แอพพลิเคชั่นใน Delphi ที่ถูกออกแบบมาเพื่อเข้าถึงข้อมูลแบบโลคอลนั้น จะสามารถเข้าถึงข้อมูลบนรีโมทเซิร์ฟเวอร์ได้โดยที่มีการแก้ไขเล็กน้อยในตัวแอพพลิเคชั่น เช่น แก้ไขพรอพเพอร์ตี้ DatabaseName ของ TTable หรือ TQuery

แต่ในทางปฏิบัติแล้วอาจมีความแตกต่างที่สำคัญระหว่างการเข้าถึงข้อมูลแบบโลคอล กับการเข้าถึงข้อมูลแบบรีโมท เช่น เพื่อเพิ่มประสิทธิภาพของแอพพลิเคชั่นแบบไคลเอนท์เซิร์ฟเวอร์ ควรจะใช้ TQuery ในการดึงข้อมูลจำนวนมากแทนการใช้ Ttable และการนำสตอร์โพรซิเดอร์มาใช้ เพื่อให้เซิร์ฟเวอร์ทำฟังก์ชั่น (Funtion) ทางคณิตศาสตร์ และฟังชั่นเกี่ยวกับการนับผลรวมแบบต่าง ๆ (Aggregate Funtion) เพราะโดยปกติแล้ว เซิร์ฟเวอร์จะมีประสิทธิ

ภาพในการทำงานสูงกว่าจึงทำการคำนวนได้เร็วกว่า และยังเป็นการลดปริมาณการสื่อสารในเครือข่ายลงด้วย เพราะว่าฟังก์ชั่นเหล่านั้นต้องทำกับโรว์ของข้อมูลจำนวนมาก จึงควรให้เซิร์ฟเวอร์คำนวณให้เสร็จก่อนและส่งผลลัพธ์กลับมา

บทที่ 3
การวางแผนและขั้นตอนการทำงาน

หลังจากการศึกษาและวิเคราะห์ออกแบบระบบแล้วก็จะได้ผลลัพธ์ออกมา 2 ส่วน คือในส่วนที่เป็นโมเดลของข้อมูล ซึ่งก็คือ โมเดล ER และอีกส่วนหนึ่งก็คือ ส่วนที่เป็นโมเดลของการทำงานของระบบ ซึ่งก็คือ แผนผังการไหลของข้อมูล ในบทนี้จะกล่าวถึงรายละเอียดต่างๆของโมเดลทั้งสอง รวมทั้งโมเดลที่เพิ่มขึ้นมาเพื่อใช้ในการเขียนโปรแกรม นั่นคือ ออปเจค-โอเรียนเต็ดโมเดล

3.1 แผนผังการไหลของข้อมูลในระบบ

การทำงานของระบบในโครงงานนี้จะแบ่งได้ออกเป็น 3 ส่วน คือ

3.1.1 ส่วนดูแลข้อมูลของผู้สมัครงาน ,พนักงานและบริษัท

ในส่วนนี้เป็นงานเกี่ยวกับการเปลี่ยนแปลงแก้ไขข้อมูลต่างๆ ซึ่งการเปลี่ยนแปลงแก้ไขข้อมูลนี้อาจเกิดจากผู้ใช้ หรือจากการทำงานส่วนอื่นๆ เช่น ผลจากการนำบันทึกการรูดบัตรไปประมวลผลก็จะได้จำนวนชั่วโมงทำงานต่างๆบันทึกลงในฐานข้อมูล หรือจากการแก้ไขเปลี่ยนแปลงข้อมูลส่วนอื่นๆ เช่น เมื่อผู้ใช้ทำการแก้ไขบันทึกการรูดบัตรที่มีข้อผิดพลาดให้ถูกต้องก็จะต้องมีการคำนวณเวลาทำงานใหม่ ซึ่งก็จะทำให้มีการแก้ไขข้อมูลในส่วนอื่นๆเปลี่ยนแปลงไปด้วย

3.1.2 ส่วนของการคำนวณเวลาทำงาน

การทำงานของส่วนนี้เป็นการนำเอาบันทึกการรูดบัตรของพนักงานมาประมวลผลเพื่อหาจำนวนชั่วโมงทำงานต่างๆ ชั่วโมงขาด ลา มาสาย เพื่อนำไปคำนวณรายได้ต่อไป สำหรับการทำงานในส่วนนี้มีอินพุทหลักคือ บันทึกการรูดบัตร ซึ่งเป็นที่เก็บข้อมูลว่า พนักงานคนใดมารูดบัตรในวันเวลาใดบ้าง และในการรูดแต่ละครั้งนั้นมีฟังก์ชั่นอย่างไร การทำงานในส่วนนี้จะเริ่มต้นจากการตรวจสอบจำนวนของบันทึกการรูดบัตรในวันที่ต้องการคำนวณเสียก่อน เพื่อในกรณีที่ว่าไม่พบบันทึกการรูดบัตรของพนักงานคนนั้น จะได้ทำการตรวจสอบต่อไปว่าพนักงานได้มีการลาล่วงหน้าหรือไม่ หรือว่าพนักงานขาดงาน ส่วนในกรณีที่พบบันทึกการรูดบัตรของพนักงาน ระบบก็จะทำการตรวจสอบลำดับของการรูดบัตรว่าถูกต้องหรือไม่ ซึ่งดูจากความสัมพันธ์ระหว่างฟังก์ชั่นของการรูดบัตรในแต่ละครั้ง สำหรับในโครงงานนี้กำหนดให้มีฟังก์ชั่นการรูดบัตร 6 ฟังก์ชั่น คือ 'in', 'out', 'abs-in', 'abs-out', 'ot-in และ 'ot-out' ซึ่งมีความหมายถึงการเข้าทำงาน, การเลิกงาน , การลางานเข้าและออกระหว่างวัน และ การเข้าและเลิกทำงานล่วงเวลาตามลำดับ การตรวจสอบลำดับของบันทึกการรูดบัตรของพนักงานนั้นจะทำโดยการจับคู่ระหว่างบันทึกที่พนักงานรูดเข้ามา ซึ่งอาศัยหลักการของสแตค (stack) โดยการพุช (push) บันทึกการ

รูดบัตรลงไปในสแตคและดูว่าบันทึกอันถัดไปนั้นสามารถจับคู่กันได้หรือไม่ ถ้าไม่สามารถจับคู่กันได้ก็พุชลงไปในสแตคต่อ แต่ถ้าสามารถจับคู่กันได้ก็ทำการพอพ (pop) บันทึกการรูดบัตรที่อยู่บนสุดในสแตคออกไป ทำอย่างนี้ไปจนกว่าจะหมดบันทึกการรูดบัตรของพนักงานในวันนั้น ถ้าผลสุดท้ายสแตคว่างก็แสดงว่าสามารถจับคู่ได้ นั่นคือ ลำดับถูกต้อง แต่ถ้าไม่สามารถจับคู่ได้ก็แสดงว่าลำดับการรูดบัตรไม่ถูกต้อง

หลังจากที่ทำการตรวจสอบลำดับของบันทึกรูดบัตรของพนักงานเสร็จเรียบร้อยแล้ว ในกรณีที่ลำดับการรูดบัตรถูกต้อง ก็จะสามารถนำไปคำนวณเวลาทำงานได้ต่อไป ส่วนในกรณีที่ลำดับการรูดบัตรไม่ถูกต้องก็จะได้ทำการบันทึกถึงข้อผิดพลาดที่เกิดขึ้นเพื่อแจ้งให้ผู้ใช้ทราบและมาทำการแก้ไขต่อไป

3.1.3 ส่วนของการคำนวณรายได้

การทำงานอีกส่วนหนึ่งที่สำคัญของระบบก็คือการคำนวณรายได้ให้กับพนักงาน ซึ่งในระบบนี้จะทำโดยถือว่ามีการคำนวณรายได้เดือนละครั้งสำหรับการจ้างงานทุกๆ ประเภท โดยการคำนวณรายได้ของพนักงานนั้นจะประกอบไปด้วยการคำนวณรายได้จริงของพนักงาน ซึ่งมีที่มาแตกต่างกันไปตามแต่ประเภทการจ้างงานของพนักงาน ต่อจากนั้นก็จะนำรายได้จริงที่คำนวณได้มาทำการหักภาษี ณ ที่จ่าย, หักเงินประกันสังคม และหักเงินกองทุนสำรองเลี้ยงชีพ และก็จะมีการพิจารณาว่าในเดือนนั้นพนักงานมีเงินได้หรือเงินหักอื่นๆ หรือไม่ เช่น ค่าใช้จ่ายในการเดินทางที่บริษัทจ่ายให้ หรือ เงินหักซึ่งเกิดจากการขาด ลา มาสายเกินจำนวนครั้งที่บริษัทกำหนด ซึ่งข้อมูลในส่วนนี้จะมีทั้งส่วนที่ผู้ใช้จะเป็นผู้ป้อนเข้ามา เช่น ค่าเดินทาง ค่ารักษาพยาบาล และมีส่วนที่ระบบคำนวณให้ เช่น การคำนวณเงินหักจากการมาสายเกินจำนวนครั้งที่กำหนด เป็นต้น

สำหรับการคำนวณรายได้จริงของพนักงานซึ่งมีที่มาแตกต่างกันตามแต่ประเภทการจ้างงาน สามารถแยกได้ดังนี้

- พนักงานรายเดือนก็จะมีรายได้จากเงินเดือนประจำ โบนัส(ถ้าเดือนนั้นเป็นงวดการจ่ายโบนัสของบริษัท) การทำงานล่วงเวลาในวันทำงานและวันหยุด
- พนักงานรายวัน รายได้ของพนักงานจะมาจากจำนวนชั่วโมงทำงานจริงของพนักงานทั้งในวันทำงาน และการทำงานล่วงเวลาทั้งในวันทำงานและวันหยุด
- พนักงานรายชิ้น รายได้ของพนักงานจะมาจากการทำงานรายชิ้นในเดือนนั้น
- พนักงานรายวัน+รายชิ้น จะมีรายได้จากชั่วโมงทำงานจริงของพนักงานทั้งในวันทำงาน และการทำงานล่วงเวลาทั้งในวันทำงานและวันหยุด รวมกับการทำงานรายชิ้น ซึ่งมีเงื่อนไขเกี่ยวกับจำนวนชิ้นที่น้อยที่สุดที่จะต้องทำได้ โดยจำนวนชิ้นที่นำมาคำนวณรายได้นั้นจะเป็นจำนวนที่เกินมาจากจำนวนที่กำหนดเท่านั้น ไม่ใช่จำนวนทั้งหมดที่ทำได้ดังเช่นพนักงานรายชิ้น

ข้อสังเกตสำหรับการคำนวณรายได้สำหรับพนักงานรายวันหรือรายวัน+รายชิ้นนั้นจะต้องทำการตรวจสอบด้วยว่าในเดือนที่ทำการคำนวณรายได้นั้น มีการเปลี่ยนแปลงค่าจ้างแรงงานภาย

ในเดือนนั้นหรือไม่ เพื่อที่จะได้นำเงินค่าจ้างแรงงานมาคำนวณให้ถูกต้องกับวันที่ทำงาน

3.2 โมเดล ER ของระบบ

ในส่วนของโมเดล ER ของระบบนี้จะแบ่งออกเป็นข้อมูลผู้สมัครงาน ข้อมูลของพนักงาน และข้อมูลของบริษัท ในส่วนของพนักงานนั้นจะประกอบไปด้วย ส่วนที่เป็นข้อมูลทั่วไป และส่วนที่เป็นข้อมูลเกี่ยวกับการคำนวณเวลาทำงานและการคำนวณรายได้ สำหรับส่วนของบริษัทนั้นส่วนใหญ่แล้วจะเป็นเรื่องเกี่ยวกับนโยบายของบริษัท ซึ่งจะถูกนำไปใช้ในการประกอบในการคำนวณเวลาทำงานและคำนวณรายได้ของพนักงาน

- ข้อมูลของผู้สมัครงาน

ข้อมูลทั่วไปของผู้สมัครงานจะอยู่ในเอนติตี้ applicant ซึ่งมี Applicant_ID (รหัสผู้สมัคร) และ Company_ID (รหัสของบริษัทที่สมัครงาน) เป็นคีย์หลัก ข้อมูลอื่นๆของผู้สมัครงานที่เก็บก็เช่น ชื่อ นามสกุล วุฒิการศึกษา ตำแหน่งและวันที่ที่สมัคร เป็นต้น นอกจากนี้แล้วก็ยังมีข้อมูลประสบการณ์การทำงานของผู้สมัครงาน ,การเข้าร่วมการอบรมของผู้สมัครงาน ซึ่งข้อมูลการเข้าร่วมการอบรมนี้จะมี Applicant_ID, Company_ID ร่วมกับ Subject (หัวข้อการอบรม) และ Running_number (ครั้งที่) เป็นคีย์หลักซึ่งเป็นคีย์นอกมาจาก Applicant และ lecturing ซึ่งเก็บข้อมูลการอบรมต่างๆ

- ข้อมูลของพนักงาน

ข้อมูลทั่วไปของพนักงานจะประกอบไปด้วยข้อมูลทั่วไปของผู้สมัครงาน ซึ่งจะมี Applicant_ID เป็นคีย์นอกในเอนติตี้ employee และข้อมูลของพนักงานที่เพิ่มเข้ามาอยู่ในเอนติตี้ employee ซึ่งมี Employee_id และ Company_id เป็นคีย์หลัก ข้อมูลของพนักงาน ตัวอย่างเช่น วันที่เข้าทำงาน , ประเภทของพนักงาน , ประเภทการจ้างงาน และสถานะของพนักงาน ซึ่งข้อมูล 3 ข้อมูลหลังนี้จะเป็นคีย์นอกมาจาก employee_type ,hiring_type และ employee_status ตามลำดับ

ในการเข้าทำงานของพนักงานจะต้องมีผู้ค้ำประกัน ข้อมูลในส่วนนี้จะถูกแทนด้วยเอนติตี้ guaruntee ซึ่งเก็บข้อมูลว่าพนักงานคนนี้มีผู้ค้ำประกันเป็นใคร และวงเงินค้ำประกันเป็นเงินเท่าใด ซึ่งความสัมพันธ์ระหว่างพนักงานกับผู้ค้ำประกันนี้เป็นความสัมพันธ์แบบกลุ่มต่อกลุ่ม นั่นคือ พนักงาน 1 คนมีผู้ค้ำประกันได้หลายคน และผู้ค้ำประกัน 1 คนสามารถค้ำประกันพนักงานได้หลายคน ส่วนข้อมูลของผู้ค้ำประกันนั้นจะถูกแทนด้วยเอนติตี้ guaruntist

หลังจากที่พนักงานเข้าทำงานแล้วก็จะมีการประเมินผลทุกๆไตรมาสในแต่ละปี ซึ่งผลจากการประเมินนี้จะถูกแทนด้วยเอนติตี้ evaluation_result โดยจะเก็บข้อมูลว่าในไตรมาสนั้น ปีนั้นพนักงานได้รับคะแนนจากการประเมินเป็นเท่าไร โดยมี Employee_ID, Company_ID , Quarter_no และ Year เป็นคีย์หลัก โดยความสัมพันธ์ระหว่างพนักงานและผลการประเมินจะเป็นแบบหนึ่งต่อกลุ่ม

นอกจากนี้แล้วก็ยังมีส่วนที่เก็บข้อมูลเกี่ยวกับรายได้ของพนักงาน ได้แก่ประวัติเงินเดือน (salary_history), ประวัติโบนัส (bonus_history) และประวัติค่าจ้างแรงงาน (wage_history) ซึ่งข้อมูลเหล่านี้จะถูกดึงไปใช้ในการคำนวณรายได้ของพนักงาน ในการเก็บข้อมูลเหล่านี้จะมี Employee_ID, Company_ID และ วันที่เปลี่ยนแปลงเงินเดือน (หรือ วันที่เปลี่ยนแปลงโบนัส หรือ วันที่เปลี่ยนแปลงค่าจ้างแรงงาน) เป็นคีย์หลัก ความสัมพันธ์ระหว่างพนักงานและประวัติรายได้ต่าง ๆนี้ก็เป็นความสัมพันธ์แบบหนึ่งต่อกลุ่มเช่นเดียวกัน

เมื่อพนักงานทำงานไปแล้วมีการปรับเปลี่ยนตำแหน่งในการทำงาน (position_history) ซึ่งจะเก็บข้อมูลว่าพนักงานคนใดเคยทำงานในตำแหน่งใดบ้าง และเริ่มทำเมื่อใด

ข้อมูลสำคัญอีกส่วนหนึ่งของพนักงานก็คือ ข้อมูลที่เกี่ยวข้องกับการคำนวณเวลาทำงานและรายได้จากการทำงาน ได้แก่ บันทึกการรูดบัตร(TAM record) ของพนักงานในแต่ละวัน ซึ่งเป็นอินพุทหลักของการคำนวณเวลาทำงาน , บันทึกการลางานล่วงหน้า (Advance absent record) ซึ่งนำไปใช้ในกรณีที่ไม่พบบันทึกการรูดบัตรของพนักงานในวันที่ต้องการคำนวณ จะได้ทำการตรวจสอบต่อไปได้ว่าพนักงานนั้นได้ลาไว้แล้วล่วงหน้าหรือไม่ ก่อนที่จะตัดสินว่าพนักงานคนนั้นขาดงานไป เมื่อนำบันทึกการรูดบัตรของพนักงานไปคำนวณเวลาทำงานแล้ว ผลลัพธ์ที่ได้ก็คือ บันทึกการทำงาน (Work record) ซึ่งก็คือผลสรุปว่าจากบันทึกการรูดบัตรนั้นสามารถทำการคำนวณเวลาทำงานได้หรือไม่ และถ้าคำนวณได้ พนักงานทำงานได้กี่ชั่วโมง แบ่งเป็นเวลาทำงานปกติกี่ชั่วโมงและทำงานล่วงเวลากี่ชั่วโมง มีการขาดงาน เช่น มาสาย หรือ ขาดงานไปทั้งวันหรือไม่ ข้อมูลเกี่ยวกับการขาดงานนี้จะอยู่ในส่วนของบันทึกการลาหยุด (Absent day)

สำหรับพนักงานรายวันและพนักงานรายวัน+รายชิ้นแล้ว ก็ยังมีส่วนที่เก็บว่าพนักงานทำชิ้นงานใดได้กี่ชิ้นบ้างในแต่ละวัน (Piece working record)

เมื่อได้ข้อมูลเกี่ยวกับการทำงานมาครบแล้วก็สามารถที่จะคำนวณรายได้ของพนักงานได้ ซึ่งผลที่ได้จากการคำนวณรายได้ให้พนักงานนี้ จะเรียกว่า ข้อมูลรายได้ (Income data) ซึ่งจะมีส่วนที่เป็นส่วนที่พนักงานทุกคนมีเหมือนกันหมด เช่น รายได้ทั้งหมด, ภาษี ,เงินประกันสังคม , เงินกองทุนสำรองเลี้ยงชีพ เป็นต้น และส่วนที่แตกต่างกันไปสำหรับพนักงานที่มีประเภทการจ้างงานต่างกันไป (Daily work, Piece and daily work, Piece work, monthly work) เช่น พนักงานรายเดือนจะไม่มีรายได้จากการทำงานชั่วโมงปกติ เพราะว่ามีเงินเดือนอยู่แล้ว แต่พนักงานรายวันหรือพนักงานรายวัน+รายชิ้นจะมี หรือพนักงานรายชิ้นก็จะไม่มีรายได้จาการทำงานในชั่วโมงทำงาน เพราะว่ารายได้นั้นมาจากการทำงานรายชิ้นเพียงอย่างเดียว

นอกจากรายได้จากการทำงานแล้ว ก็ยังมีส่วนที่เป็นเงินได้หรือเงินหักเพิ่มเติม (other income data , other deduction data) ซึ่งผู้ใช้จะเป็นผู้ใส่รายละเอียดในส่วนนี้ และนำไปคำนวณในขั้นตอนของการคำนวณรายได้ต่อไป

- ข้อมูลของบริษัท

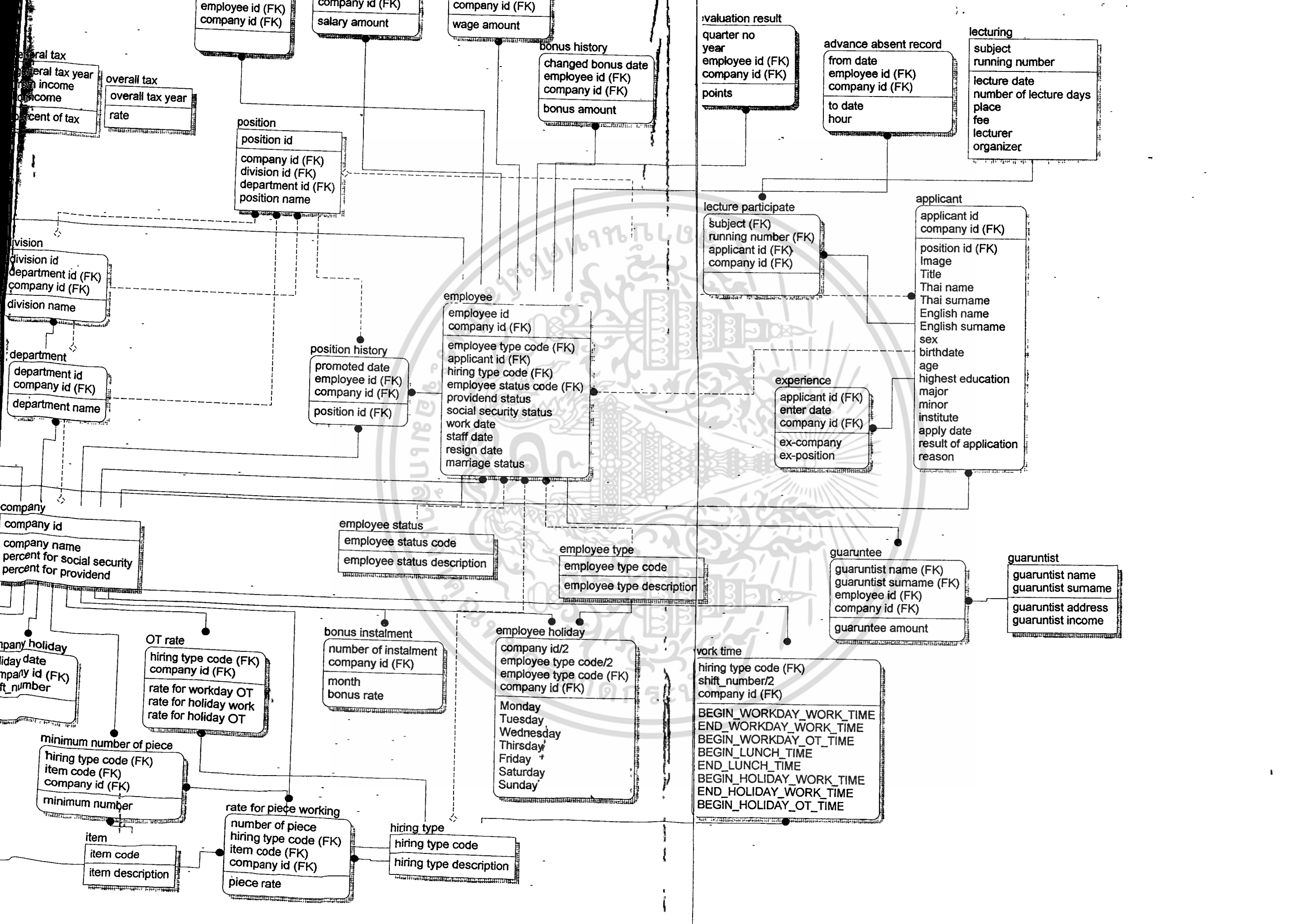
ข้อมูลเกี่ยวกับบริษัทนั้นอาจจะมองออกได้เป็น 2 ส่วนคือ ส่วนนโยบายต่างๆ และส่วนที่เกี่ยวกับโครงสร้างของบริษัท ซึ่งในแต่ละส่วนมีรายละเอียดดังต่อไปนี้

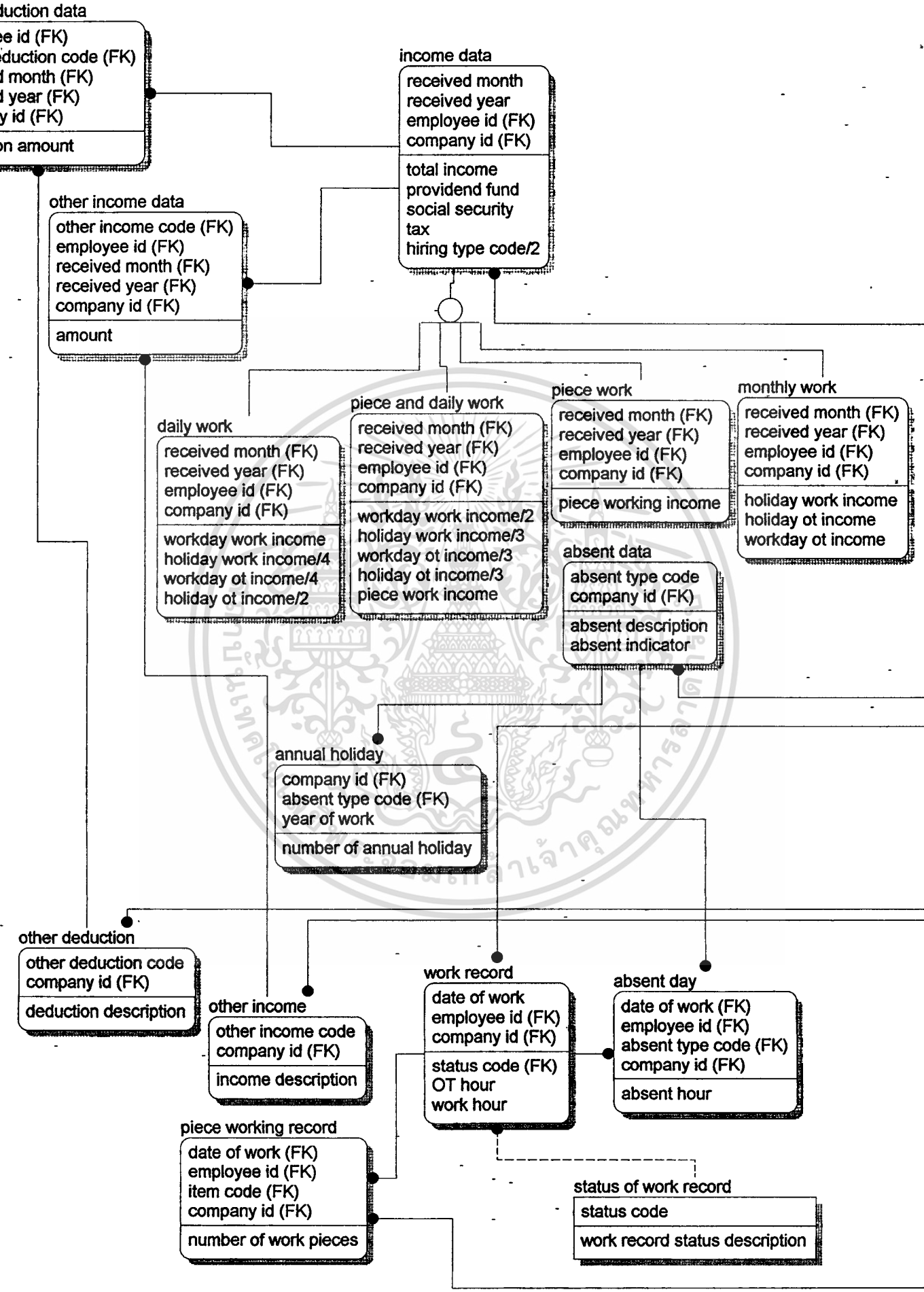
ส่วนที่เกี่ยวกับโครงสร้างของบริษัท คือการเก็บข้อมูลว่าบริษัทนี้ประกอบไปด้วยฝ่าย (Department)ใดบ้างและฝ่ายต่างๆนั้นประกอบไปด้วยส่วน(Division)ใดบ้าง รวมทั้งการเก็บข้อมูลเกี่ยวกับตำแหน่งงาน(Position)ในบริษัทว่าในบริษัท ฝ่ายและส่วนต่างๆนั้นมีตำแหน่งงานใดบ้าง

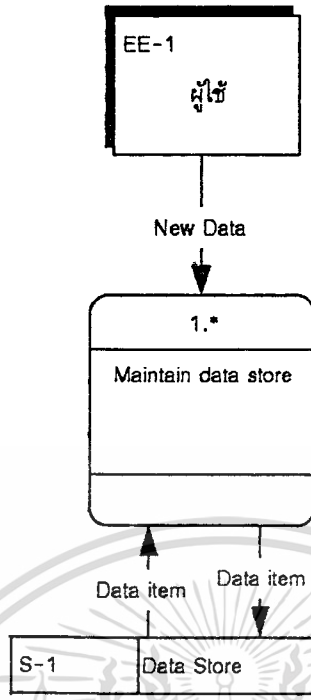
ส่วนนโยบายบริษัทนั้น เป็นส่วนที่ถูกนำไปใช้มากในการทำงานของระบบทั้งการคำนวณเวลาทำงานและการคำนวณรายได้ ข้อมูลที่ถูกนำไปใช้ในขั้นตอนของการคำนวณเวลาทำงาน ก็ได้แก่ ข้อมูลเกี่ยวกับวันหยุดของบริษัท(Company holiday) เพื่อตรวจสอบว่าวันที่ต้องการคำนวณเวลาทำงานนั้นเป็นวันหยุดของพนักงานหรือไม่ , ข้อมูลเกี่ยวกับเวลาทำงานของพนักงาน (Work Time) ว่าพนักงานที่มีการจ้างงานประเภทต่างๆนั้นมีเวลาเริ่มงานและเลิกงานในเวลาเท่าใด

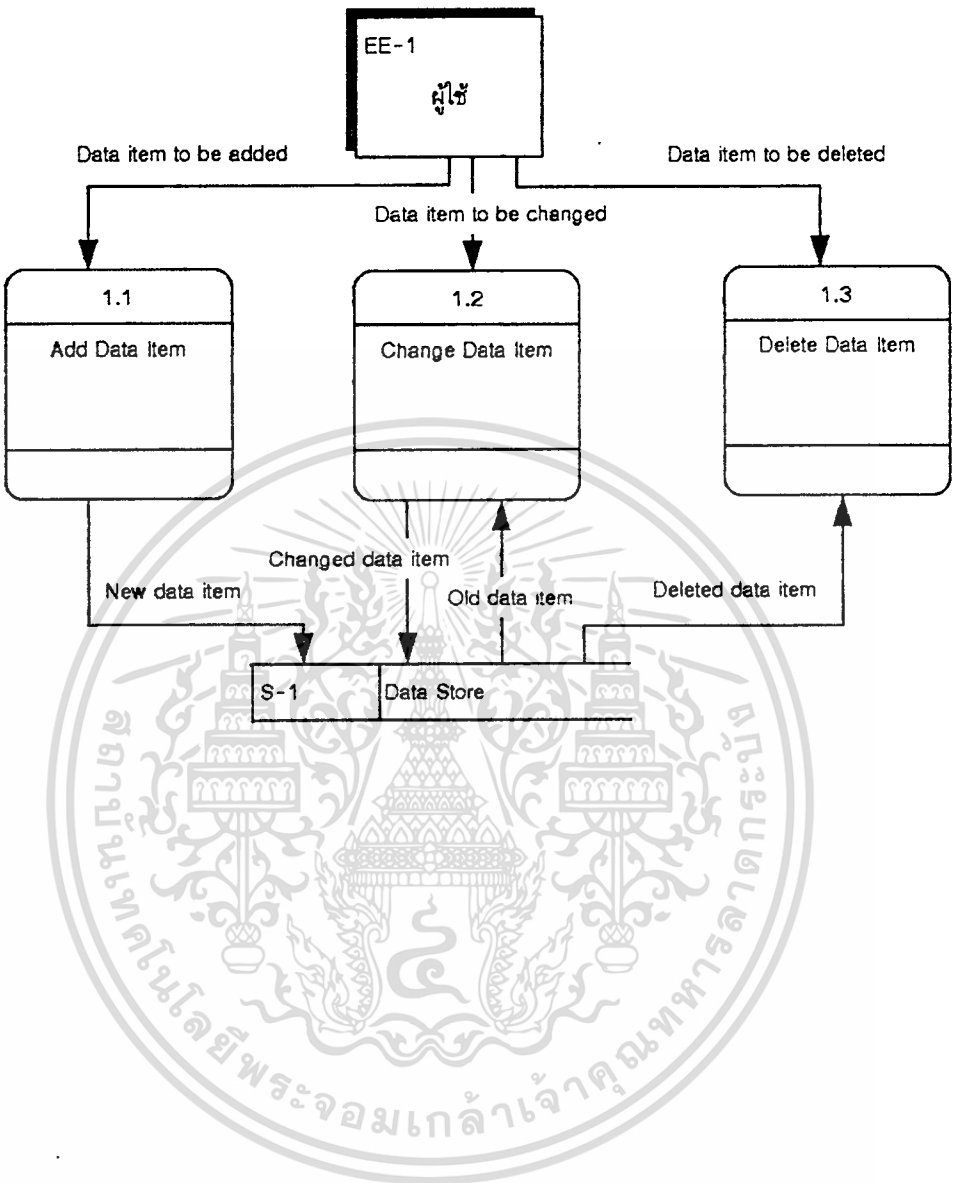
สำหรับข้อมูลที่ถูกนำไปใช้ในการคำนวณรายได้ของพนักงาน ได้แก่ อัตราค่าล่วงเวลา (Rate ต่างๆใน Hiring Type) ของพนักงานตามแต่ประเภทของการจ้างงานทั้งในการทำงานล่วงเวลาในวันทำงาน, การทำงานในวันหยุด และการทำงานในช่วงเวลาล่วงเวลาของวันหยุด , อัตราการคำนวณสำหรับการทำงานรายชิ้น (Rate for piece working) ที่บอกว่าแต่ละชิ้นงานที่พนักงานทำได้นั้นจะคิดเงินให้ในอัตราชิ้นละเท่าใด ซึ่งในส่วนนี้ สำหรับพนักงานรายวัน+รายชิ้นก็จะมีสิ่งที่ต้องนำมาพิจารณาเพิ่มด้วยก็คือ ปริมาณชิ้นงานที่น้อยที่สุดที่พนักงานจะต้องทำได้ (Minimum Piece)ตัวอย่างเช่น สำหรับพนักงานรายวัน+รายชิ้นแล้ว จะต้องทำชิ้นงานรหัส 01 ให้ได้มากกว่า 15 ชิ้นขึ้นไปจึงจะคำนวณรายได้ให้ ตัวอย่างเช่น พนักงานทำชิ้นงานรหัส 01 ได้ 30 ชิ้น ส่วนที่จะนำมาคำนวณรายได้ให้ก็คือ 15 ชิ้นที่เกินมาจากปริมาณที่กำหนด ,อัตราภาษี (Tax Formula) นำมาใช้ในการคำนวณภาษีหัก ณ ที่จ่าย

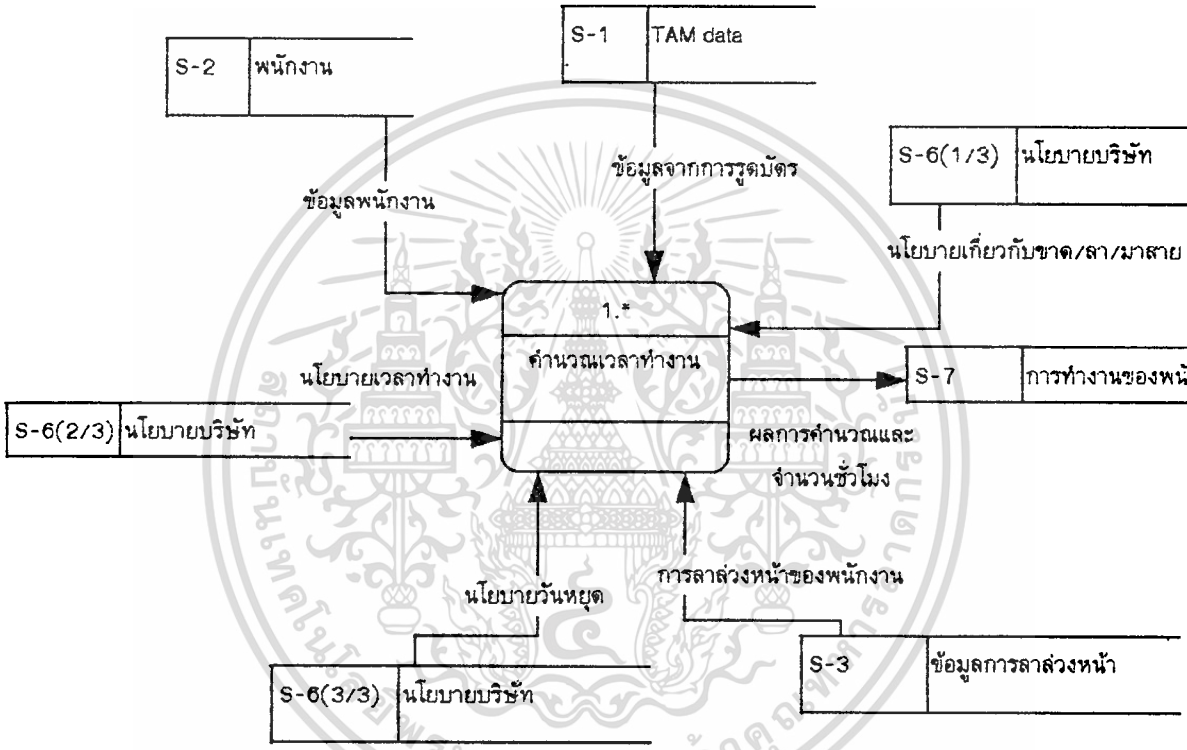
นอกจากในส่วนของนโยบายบริษัทก็ยังมีข้อมูลเกี่ยวกับการลาหยุดอีกด้วย ซึ่งจะแบ่งออกเป็น 2 ประเภท คือ การลาหยุดประจำปี (Annual holiday) และการลาหยุดประเภทอื่นๆ (Other absent) สำหรับการลาหยุดประจำปีนั้น จะต้องคำนึงถึงอายุงานของพนักงานด้วยว่าทำงานมากี่ปีแล้วมีสิทธิที่จะลาได้กี่วัน และในส่วนของการลาหยุดประเภทอื่นๆ นั้นนอกจากจะมีการให้กำหนดรหัสการหยุดงานแบบต่างๆแล้ว ก็ยังสามารถกำหนดนโยบายที่นำไปใช้ในการคำนวณรายได้ได้อีกด้วย นั่นคือ จะสามารถกำหนดได้ว่าเมื่อพนักงานลาหยุดประเภทใดๆ ไปเกินกว่าที่กำหนดไว้แล้ว จะมีผลต่อการคำนวณรายได้ โดยจะมีการนำเอาอัตราที่กำหนดไว้ในส่วนนี้ไปพิจารณาในการคำนวณรายได้ด้วย

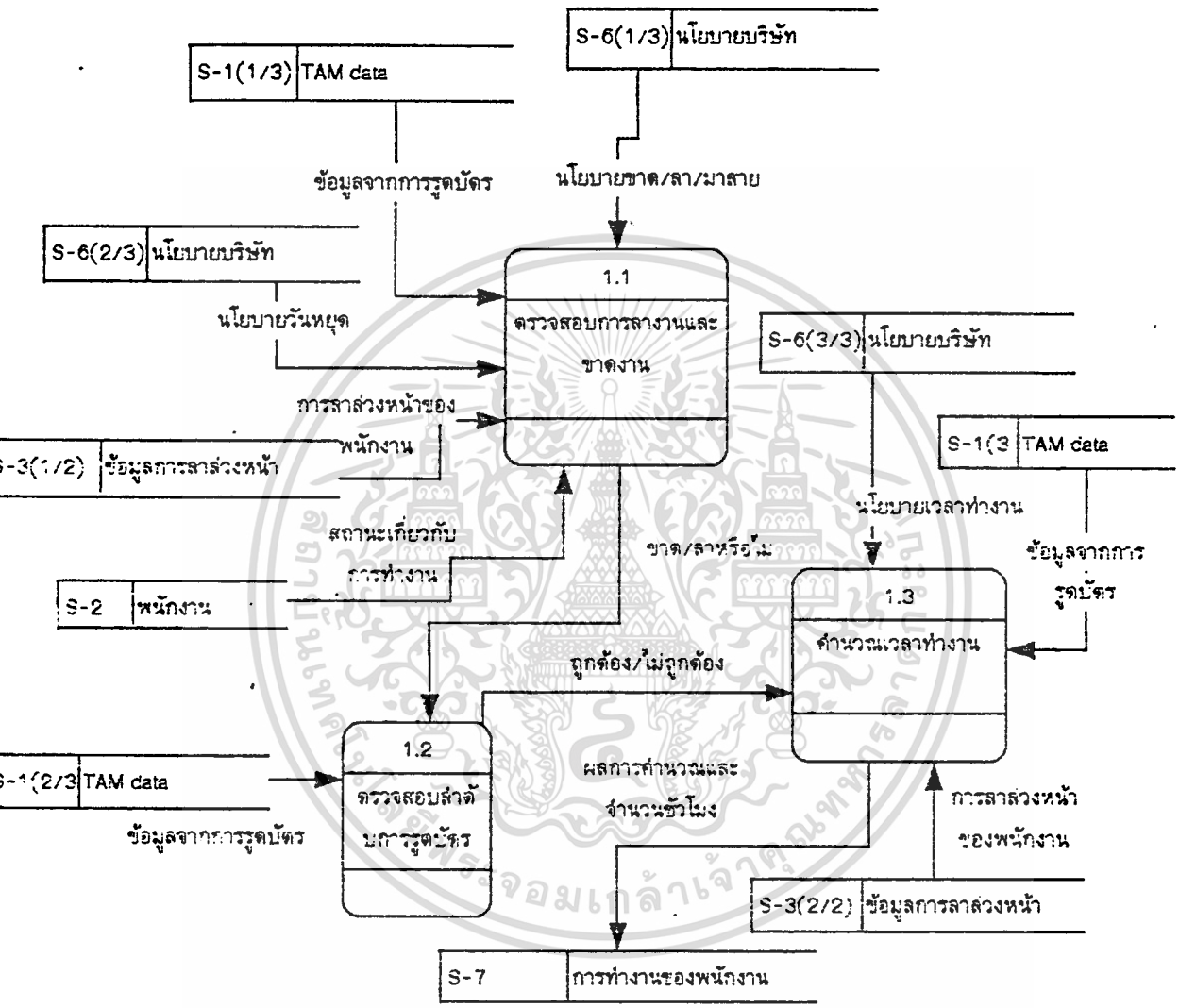

employee id (FK)
company id (FK)
company id (FK)
salary amount
company id (FK)
wage amount
bonus history
changed bonus date
employee id (FK)
company id (FK)
bonus amount
valuation result
quarter no
year
employee id (FK)
company id (FK)
points
advance absent record
from date
employee id (FK)
company id (FK)
to date
hour
lecturing
subject
running number
lecture date
number of lecture days
place
fee
lecturer
organizer
eral tax
eral tax year
income
income
cent of tax
overall tax
overall tax year
rate
position
position id
company id (FK)
division id (FK)
department id (FK)
position name
lecture participate
subject (FK)
running number (FK)
applicant id (FK)
company id (FK)
applicant
applicant id
company id (FK)
position id (FK)
Image
Title
Thai name
Thai surname
English name
English surname
sex
birthdate
age
highest education
major
minor
institute
apply date
result of application
reason
vision
division id
department id (FK)
company id (FK)
division name
department
department id
company id (FK)
department name
position history
promoted date
employee id (FK)
company id (FK)
position id (FK)
employee
employee id
company id (FK)
employee type code (FK)
applicant id (FK)
hiring type code (FK)
employee status code (FK)
providend status
social security status
work date
staff date
resign date
marriage status
experience
applicant id (FK)
enter date
company id (FK)
ex-company
ex-position
company
company id
company name
percent for social security
percent for providend
employee status
employee status code
employee status description
employee type
employee type code
employee type description
guaruntee
guaruntist name (FK)
guaruntist surname (FK)
employee id (FK)
company id (FK)
guaruntee amount
guaruntist
guaruntist name
guaruntist surname
guaruntist address
guaruntist income
pany holiday
iday date
mpany id (FK)
ft_number
OT rate
hiring type code (FK)
company id (FK)
rate for workday OT
rate for holiday work
rate for holiday OT
bonus instalment
number of instalment
company id (FK)
month
bonus rate
employee holiday
company id/2
employee type code/2
employee type code (FK)
company id (FK)
Monday
Tuesday
Wednesday
Thirsday
Friday
Saturday
Sunday
work time
hiring type code (FK)
shift_number/2
company id (FK)
BEGIN_WORKDAY_WORK_TIME
END_WORKDAY_WORK_TIME
BEGIN_WORKDAY_OT_TIME
BEGIN_LUNCH_TIME
END_LUNCH_TIME
BEGIN_HOLIDAY_WORK_TIME
END_HOLIDAY_WORK_TIME
BEGIN_HOLIDAY_OT_TIME
minimum number of piece
hiring type code (FK)
item code (FK)
company id (FK)
minimum number
item
item code
item description
rate for piece working
number of piece
hiring type code (FK)
item code (FK)
company id (FK)
piece rate
hiring type
hiring type code
hiring type description

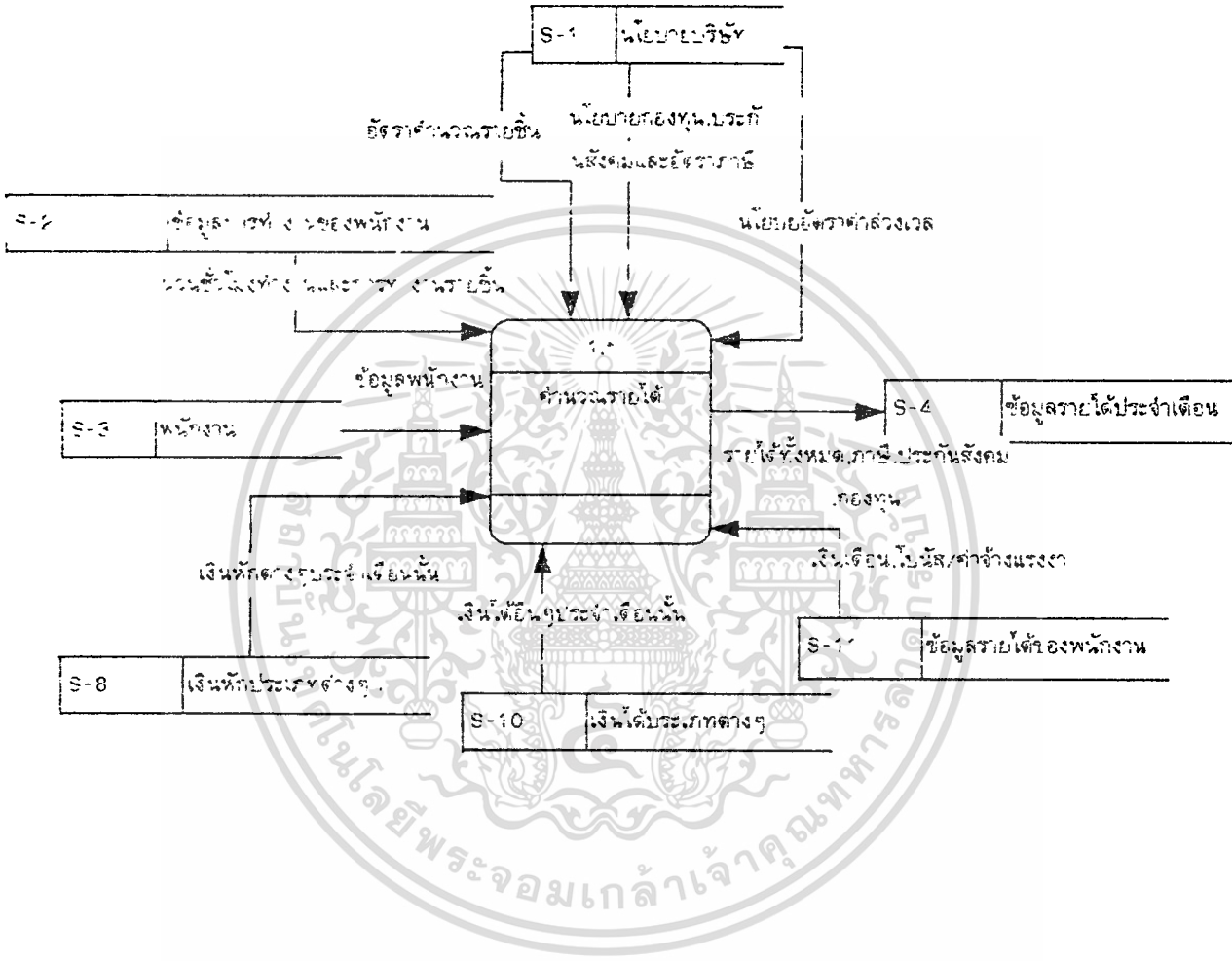

duction data
ee id (FK)
eduction code (FK)
d month (FK)
d year (FK)
y id (FK)
on amount
income data
received month
received year
employee id (FK)
company id (FK)
total income
providend fund
social security
tax
hiring type code/2
other income data
other income code (FK)
employee id (FK)
received month (FK)
received year (FK)
company id (FK)
amount
daily work
received month (FK)
received year (FK)
employee id (FK)
company id (FK)
workday work income
holiday work income/4
workday ot income/4
holiday ot income/2
piece and daily work
received month (FK)
received year (FK)
employee id (FK)
company id (FK)
workday work income/2
holiday work income/3
workday ot income/3
holiday ot income/3
piece work income
piece work
received month (FK)
received year (FK)
employee id (FK)
company id (FK)
piece working income
monthly work
received month (FK)
received year (FK)
employee id (FK)
company id (FK)
holiday work income
holiday ot income
workday ot income
absent data
absent type code
company id (FK)
absent description
absent indicator
annual holiday
company id (FK)
absent type code (FK)
year of work
number of annual holiday
other deduction
other deduction code
company id (FK)
deduction description
other income
other income code
company id (FK)
income description
work record
date of work
employee id (FK)
company id (FK)
status code (FK)
OT hour
work hour
absent day
date of work (FK)
employee id (FK)
absent type code (FK)
company id (FK)
absent hour
piece working record
date of work (FK)
employee id (FK)
item code (FK)
company id (FK)
number of work pieces
status of work record
status code
work record status description

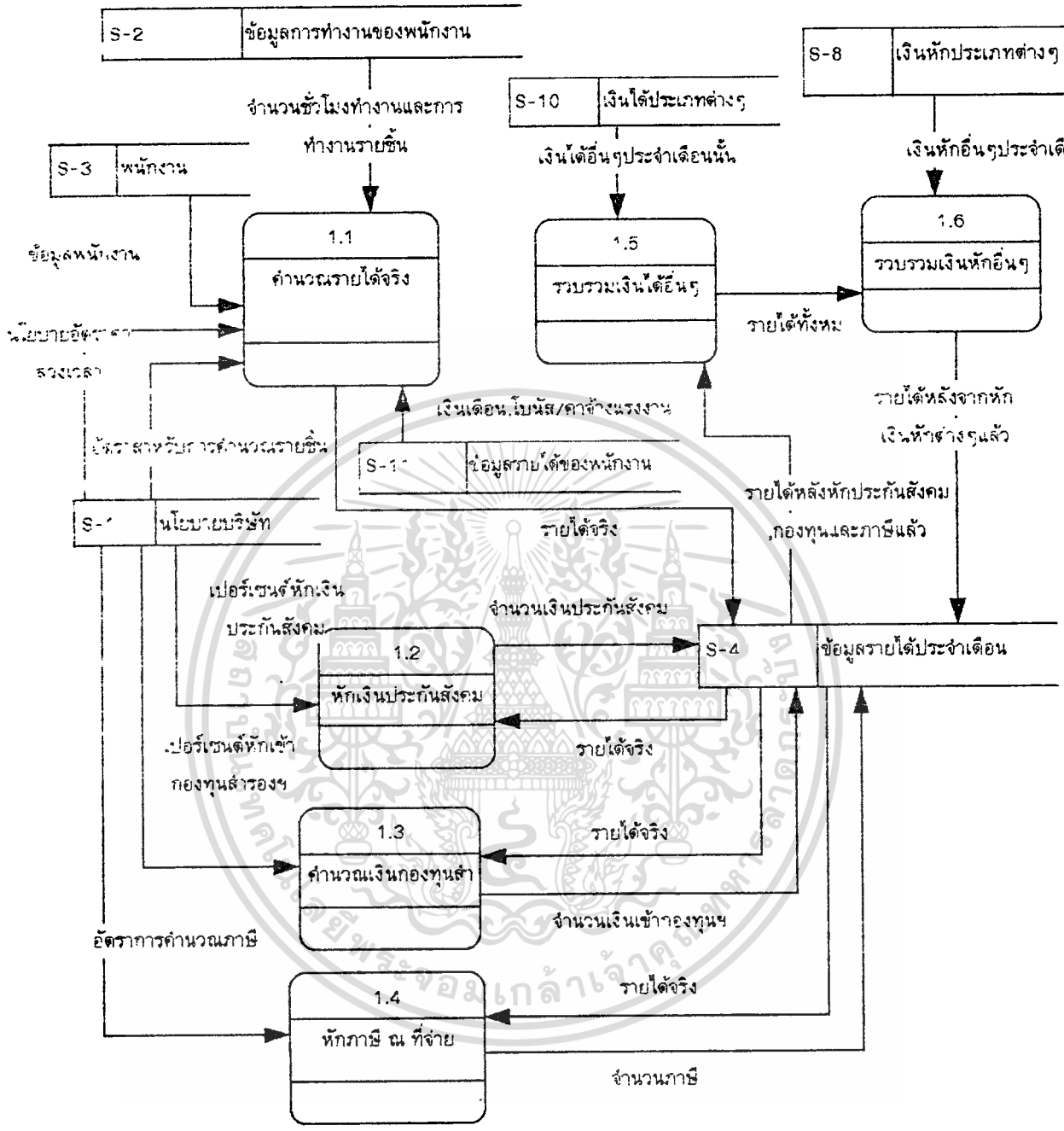

EE-1
ผู้ใช้
New Data
1.*
Maintain data store
Data item
Data item
S-1
Data Store


EE-1
ผู้ใช้
Data item to be added
Data item to be changed
Data item to be deleted
1.1
Add Data Item
1.2
Change Data Item
1.3
Delete Data Item
Changed data item
New data item
Old data item
Deleted data item
S-1
Data Store


S-1
TAM data
S-2
พนักงาน
ข้อมูลจากการรูดบัตร
ข้อมูลพนักงาน
S-6(1/3)
นโยบายบริษัท
นโยบายเกี่ยวกับขาด/ลา/มาสาย
1.*
คำนวณเวลาทำงาน
นโยบายเวลาทำงาน
S-7
การทำงานของพนักงา
S-6(2/3)
นโยบายบริษัท
ผลการคำนวณและ
จำนวนชั่วโมง
การลาล่วงหน้าของพนักงาน
นโยบายวันหยุด
S-3
ข้อมูลการลาล่วงหน้า
S-6(3/3)
นโยบายบริษัท


S-6(1/3) นโยบายบริษัท
S-1(1/3) TAM data
ข้อมูลจากการรูดบัตร
นโยบายขาด/ลา/มาสาย
S-6(2/3) นโยบายบริษัท
นโยบายวันหยุด
1.1
ตรวจสอบการลางานและขาดงาน
S-6(3/3) นโยบายบริษัท
การลาล่วงหน้าของพนักงาน
S-3(1/2) ข้อมูลการลาล่วงหน้า
S-1(3 TAM data
นโยบายเวลาทำงาน
สถานะเกี่ยวกับการทำงาน
ขาด/ลาหรือไม่
ข้อมูลจากการรูดบัตร
S-2 พนักงาน
1.3
คำนวณเวลาทำงาน
ถูกต้อง/ไม่ถูกต้อง
1.2
ตรวจสอบลำดับการรูดบัตร
ผลการคำนวณและจำนวนชั่วโมง
S-1(2/3 TAM data
ข้อมูลจากการรูดบัตร
การลาล่วงหน้าของพนักงาน
S-3(2/2) ข้อมูลการลาล่วงหน้า
S-7 การทำงานของพนักงาน


S-1
นโยบายบริษัท
นโยบายกองทุนประกั
นสังคมและอัตราภาษี
อัตราค่าลวงเวลารายชิ้น
นโยบายอัตราค่าล่วงเวล
S-2
ข้อมูลเวลาของพนักงาน
ข้อมูลพนักงาน
S-3
พนักงาน
คำนวณรายได้
S-4
ข้อมูลรายได้ประจำเดือน
รายได้ทั้งหมดภาษีประกันสังคม
กองทุน
เงินหักต่างๆประจำเดือนนั้น
เงินได้อื่นๆประจำเดือนนั้น
S-8
เงินหักประเภทต่างๆ
S-10
เงินได้ประเภทต่างๆ
S-11
ข้อมูลรายได้ของพนักงาน


S-2
ข้อมูลการทำงานของพนักงาน
จำนวนชั่วโมงทำงานและการทำงานรายชิ้น
S-3
พนักงาน
S-10
เงินได้ประเภทต่างๆ
เงินได้อื่นๆประจำเดือนนั้น
S-8
เงินหักประเภทต่างๆ
เงินหักอื่นๆประจำเดื
1.1
คำนวณรายได้จริง
1.5
รวบรวมเงินได้อื่นๆ
1.6
รวบรวมเงินหักอื่นๆ
ข้อมูลพนักงาน
รายได้ทั้งหม
เงินเดือน,โบนัส/ค่าจ้างแรงงาน
S-11
ข้อมูลรายได้ของพนักงาน
รายได้หลังจากหัก
เงินต่างๆแล้ว
S-1
นโยบายบริษัท
รายได้จริง
รายได้หลังหักประกันสังคม
,กองทุนและภาษีแล้ว
เปอร์เซนต์หักเงิน
ประกันสังคม
จำนวนเงินประกันสังคม
1.2
หักเงินประกันสังคม
S-4
ข้อมูลรายได้ประจำเดือน
เปอร์เซนต์หักเข้า
กองทุนสำรองฯ
รายได้จริง
1.3
คำนวณเงินกองทุนสำ
รายได้จริง
จำนวนเงินเข้ากองทุนฯ
อัตราการคำนวณภาษี
1.4
หักภาษี ณ ที่จ่าย
รายได้จริง
จำนวนภาษี

3 โมเดลทางออปเจค-โอเรียนเต็ด

เนื่องจากว่าในโครงงานนี้ เราเลือกที่จะใช้เทคนิคการเขียนโปรแกรมแบบออปเจค-โอเรียนเต็ดดังที่ได้กล่าวถึงหลักการไปแล้วข้างต้น ทำให้เราจะต้องหาโมเดลทางออปเจค-โอเรียนเต็ด ที่ใช้แทนระบบนี้ให้ได้เสียก่อน โดยในขั้นตอนนี้นั้นเราต้องนำเอาแนวความคิดทางด้านออปเจค-โอเรียนเต็ดมาประยุกต์ใช้กับโมเดล ER และผังการไหลของข้อมูลซึ่งได้ออกแบบไว้แล้ว เพื่อให้ได้ข้อสรุปที่ว่าในโปรแกรมของเราจะมีคลาสและซับคลาสอะไรบ้าง และในการถ่ายทอดวิธีการ (method) ของคลาสต่างๆไปยังคลาสย่อยนั้นมีการเปลี่ยนแปลงแก้ไขวิธีการนั้น หรือที่ศัพท์ทางด้านออปเจค-โอเรียนเต็ดเรียกว่า การโอเวอร์ไรด์ (override) หรือไม่อย่างไร คุณสมบัติด้านการถ่ายทอด และความสามารถในการเปลี่ยนแปลงแก้ไขวิธีการในคลาสย่อยที่ได้รับมาจากซุปเปอร์คลาสนี้เอง ก็ทำให้เกิดเป็นโพลีมอร์ฟิสซึ่ม นั่นคือ วิธีการนั้นมีชื่อเพียงชื่อเดียว แต่ทำงานรองรับออปเจคของคลาสต่างๆได้หลายคลาสแตกต่างกันไป

โดยหลักการของออปเจค-โอเรียนเต็ดนั้นคือการนำเอาข้อมูลและชุดของการทำงานมารวมเข้าไว้ด้วยกัน ซึ่งในการออกแบบระบบของเราโมเดลที่แสดงถึงข้อมูลต่างๆในระบบก็คือ โมเดล ER และส่วนที่แสดงถึงการทำงานก็คือผังการไหลของข้อมูลนั่นเอง เมื่อเราพิจารณาจากทั้งสองโมเดลแล้ว ก็ได้ความสัมพันธ์ระหว่างทั้งสองโมเดลดังนี้

โพรเซสคำนวณเวลาทำงาน

- โพรเซส1.1 (ตรวจสอบการลางานและขาดงาน) เกี่ยวข้องกับข้อมูลในโมเดล ER ดังต่อไปนี้
  - พนักงาน (Employee)
  - นโยบายวันหยุด (Company Holiday)
  - นโยบายการหยุดงานต่างๆ (Absent data)
  - บันทึกการลาล่วงหน้า (Advance absent record)
  - บันทึกการรูดบัตร (TAM record)
- โพรเซส 1.2 (ตรวจสอบลำดับการรูดบัตร) เกี่ยวข้องกับข้อมูลในโมเดล ER ดังต่อไปนี้
  - บันทึกการรูดบัตร (TAM record)
- โพรเซส 1.3 (คำนวณเวลาทำงาน) เกี่ยวข้องกับข้อมูลในโมเดล ER ดังต่อไปนี้
  - บันทึกการรูดบัตร (TAM record)
  - บันทึกการลาล่วงหน้าของพนักงาน (Advance absent record)
  - เวลาทำงานของบริษัท (Work time)
  - บันทึกการทำงานของพนักงาน (Work record)

- บันทึกการขาดงานของพนักงาน (Absent day)

โพรเซสคำนวณรายได้

- โพรเซส 1.1 (คำนวณรายได้จริง) เกี่ยวข้องกับข้อมูลในโมเดล ER ดังต่อไปนี้
  - ข้อมูลพนักงาน (Employee)
  - ประวัติรายได้ของพนักงาน (Salary History/Wage History,Bonus History)
  - บันทึกการทำงานของพนักงาน (Work record)
  - บันทึกการทำงานรายชิ้น (Piece working record)
  - อัตราการคำนวณรายได้รายชิ้น (Rate for piece working)
  - อัตราค่าล่วงเวลา (Rateต่างๆใน Hiring Type)
  - บันทึกรายได้ของพนักงาน (Income data)
- โพรเซส 1.2 (หักเงินประกันสังคม) เกี่ยวข้องกับข้อมูลในโมเดล ER ดังต่อไปนี้
  - บันทึกรายได้ของพนักงาน (Income data)
  - เปอร์เซนต์หักเงินประกันสังคม
- โพรเซส 1.3 (คำนวณเงินกองทุนสำรองเลี้ยงชีพ) เกี่ยวข้องกับข้อมูลในโมเดล ER ดังต่อไปนี้
  - บันทึกรายได้ของพนักงาน (Income data)
  - เปอร์เซนต์คำนวณเงินกองทุนสำรองเลี้ยงชีพ
- โพรเซส 1.4 (หักภาษี ณ ที่จ่าย) เกี่ยวข้องกับข้อมูลในโมเดล ER ดังต่อไปนี้
  - บันทึกรายได้ของพนักงาน (Income data)
  - อัตราภาษี (Tax Formula)
- โพรเซส 1.5 (รวบรวมเงินได้อื่นๆ) เกี่ยวข้องกับข้อมูลในโมเดล ER ดังต่อไปนี้
  - บันทึกเงินได้อื่นๆ (Other income data)
  - บันทึกรายได้ของพนักงาน (Income data)
- โพรเซส 1.6 (รวบรวมเงินหักอื่นๆ) เกี่ยวข้องกับข้อมูลในโมเดล ER ดังต่อไปนี้
  - บันทึกเงินหักอื่นๆ (Other deduction data)
  - บันทึกรายได้ของพนักงาน (Income data)

ต่อมาเราจะต้องพยายามมองว่าภายในระบบของเรานั้นมีคลาสและคลาสย่อยอะไรบ้าง และคลาสและคลาสย่อยเหล่านั้นมีความสัมพันธ์กันอย่างไร ซึ่งในส่วนนี้จะอธิบายเป็นขั้นตอนดังต่อไปนี้

1. เราจะพบว่าระบบของเรานั้นเป็นเรื่องเกี่ยวกับผู้สมัครงาน ,พนักงาน และบริษัท และสิ่งเหล่านี้คือ ออปเจคในระบบของเรานั่นเอง ซึ่งต่อมาเราก็จะได้กำหนดนิยามของออปเจคเหล่านี้ให้เป็นคลาสต่อไป

ผลลัพธ์ : ได้ออปเจคมา 3 ออปเจค คือ ผู้สมัครงาน, พนักงานและบริษัท และสร้างคลาสเพื่อเป็นนิยามของออปเจคต่อไป

2. ต่อจากนั้นเราก็จะพบว่า พนักงานนั้นมีความสัมพันธ์กับผู้สมัครงาน นั่นคือข้อมูลต่างๆของผู้สมัครงานก็ถ่ายทอดมาถึงพนักงานทั้งหมด นั่นคือ จริงๆแล้ว พนักงานเป็นคลาสย่อยของผู้สมัครงาน

ผลลัพธ์ : พบว่าพนักงานเป็นคลาสย่อยของผู้สมัครงาน

3. ค้นหาว่าคลาสทั้ง 3 ของเรานั้นจะมีคุณสมบัติและวิธีการอะไรบ้าง

ในส่วนของคุณสมบัติของคลาสนั้น ส่วนหนึ่งจะเห็นว่าเราได้จากโมเดล ER โดยตรง ซึ่งก็คือแอตตริบิวท์ของเอนติตี้ Applicant ,Emloyee และ Company นั่นเอง ต่อจากนั้นในการพิจารณาโมเดล ER ในส่วนอื่นๆ เราจะพบว่าพนักงานนั้นมีคุณสมบัติอื่นๆที่ไม่ได้อยู่ใน เอนติตี้ Employee โดยตรง โดยอยู่ในเอนติตี้อื่นๆ โดยมี Employee_id และ Company_id จากเอนติตี้ Employee ไปเป็นคีย์นอกในเอนติตี้เหล่านั้น เอนติตี้อื่นๆที่เก็บข้อมูลของพนักงานไว้ก็ได้แก่ Income data และ Work Record

ในกรณีของคลาสบริษัทก็เช่นเดียวกันคือ นอกจากจะมีคุณสมบัติตามแอตตริบิวท์ที่ปรากฏอยู่ในเอนติตี้ Company แล้วก็ยังมีส่วนอื่นๆที่ปรากฏอยู่ในเอนติตี้อื่นๆที่มี Company_id ไปเป็นคีย์นอกอยู่ด้วย เช่น งวดการจ่ายโบนัส(Bonus instalment), นโยบายการหยุดงาน (Absent data) , อัตราการคำนวณรายได้แบบรายชิ้น (Rate for piece working) ,จำนวนชิ้นที่น้อยที่สุด (Minimum number of piece)

ผลลัพธ์ : คุณสมบัติของคลาส พนักงาน มาจาก

1.ส่วนที่ได้รับการถ่ายทอดมาจาก คลาสผู้สมัครงาน

2.ส่วนที่อยู่ในเอนติตี้ Employee โดยตรง

3.ส่วนที่อยู่ในเอนติตี้อื่นๆซึ่งมี Employee_id และ Company_id จากเอนติตี้ Employee ไปเป็นคีย์นอก ได้แก่ Income Data และ Work record

คุณสมบัติของคลาส ผู้สมัครงาน นั้นก็ได้จากแอตตริบิวท์ของเอนติตี้Applicant

คุณสมบัติของคลาส บริษัท มาจาก

1.ส่วนที่เป็นแอตตริบิวของเอนติตี้ Company

2.ส่วนที่อยู่ในเอนติตี้อื่นๆซึ่งมี Company_id จากเอนติตี้ Company ไปเป็นคีย์นอกอยู่

4.จัดการกับคุณสมบัติที่อยู่ในเอนติตี้อื่นๆ

จากข้อ 3 เราจะเห็นว่าคุณสมบัติทั้งของคลาสพนักงานก็ดี คลาสบริษัทก็ดี มีส่วนหนึ่งที่อยู่ในเอนติตี้อื่นๆ ซึ่งก็คือว่าเอนติตี้พนักงาน หรือเอนติตี้บริษัท มีความสัมพันธ์เป็นแบบหนึ่งต่อกลุ่มกับเอนติตี้เหล่านั้น เช่นความสัมพันธ์ระหว่างพนักงาน (Employee) กับบันทึกการทำงาน

(Work record) หมายถึงว่า พนักงาน 1 คนก็มีบันทึกการทำงานได้หลายบันทึกตามจำนวนวันที่พนักงานคนนั้นมาทำงาน จึงเกิดคำถามที่ว่าแล้วในขณะใดขณะหนึ่งนั้นข้อมูลบันทึกการทำงานใดที่เป็นจะเป็นคุณสมบัติของออปเจคที่เราประกาศให้เป็นตัวแปรของคลาสนั้น หรืออาจยกตัวอย่างได้ว่า เมื่อเราสร้างออปเจค Employee1 ให้เป็นตัวแปรของคลาส พนักงาน แล้วเราจะเอาค่าใดของบันทึกการทำงานมาใส่เป็นคุณสมบัติของ Employee1 คำตอบก็คือว่าการที่เราจะเลือกเอาค่าของข้อมูลใดมาให้เป็นคุณสมบัติของออปเจคที่เราสร้างขึ้น ก็จะขึ้นอยู่กับบริบท (context) ของออปเจคนั้น เช่นในขณะที่เรากำลังคำนวณรายได้ของพนักงาน สำหรับเดือนมีนาคม ปี 2539 เราก็จะเลือกเอาบันทึกการทำงานของพนักงานในเดือนมกราคม ปี 2539 มาใส่เป็นค่าของคุณสมบัติที่เกี่ยวกับชั่วโมงทำงานของออปเจค Employee1

5. จากข้อ 3 และข้อ 4 ทำให้เราสามารถแก้ปัญหาที่ว่าคุณสมบัติของคลาสนั้นมีความสัมพันธ์แบบหนึ่งต่อกลุ่มอยู่ในโมเดล ER ได้ แต่ก็ยังมีจุดอื่นที่มีปัญหาอยู่ คือ ในกรณีของความสัมพันธ์ระหว่างเอนติตี้ บริษัท (Company) กับนโยบายการหยุดงาน (Absent data) จะเห็นว่าไม่เพียงแค่ความสัมพันธ์จะเป็นแบบหนึ่งต่อกลุ่มแล้ว นโยบายการหยุดงานยังถูกแบ่งเป็นสับไทป์ (subtype) อีก 2 สับไทป์ แล้วโครงสร้างข้อมูลของคุณสมบัติจะเป็นอย่างไร แนวทางในการแก้ปัญหานี้ อาจเป็นไปได้ดังนี้

- นำเอาแอตตริบิวท์ทุกตัวของ Absent data และสับไทป์ของมันมาเป็นคุณสมบัติของคลาสบริษัท และการแทนค่าต่างๆนั้นก็ใช้หลักการเดียวกับข้อ 4 ถ้าในครั้งใดที่ประเภทของการลาหยุดเป็นการลาประจำปี คุณสมบัติ salary indicator และ absent day ก็จะไม่มีค่า เช่นเดียวกับเมื่อประเภทของการหยุดงานเป็นการหยุดงานประเภทอื่นๆ คุณสมบัติ number of annual holiday และ year of work ก็จะไม่มีค่าไป วิธีการนี้สามารถแสดงในภาษาปาสคาลได้ดังนี้

```
Tcompany = class
        Company_id : string;
        . . .
        Absent_type_code : string;
        Absent_description : string;
        Absent_indicator : string;
        Number_of_annual_holiday : integer;
        Year_of_work : integer;
        Salary_indicator : real;
        Absent_day : integer;
        . . .
```

```
end;
```

- ใช้โครงสร้างข้อมูลแบบเรคคอร์ดในภาษาปาสคาล แทนประเภทของการหยุดงาน นั่นคือ มีโครงสร้างข้อมูลแบบเรคคอร์ดสำหรับการลาหยุดประเภทการลาประจำปี 1 เรคคอร์ด และสำหรับการลาหยุดประเภทอื่นๆอีก 1 เรคคอร์ด วิธีการนี้แสดงอยู่ในรูปของภาษาปาสคาลได้ดังนี้

```
Type
    Annual_holiday = record
        Absent_type_code : string;
        Number_of_annual_holiday : integer;
        Year_of_work : integer;
    end;
    Other_absent = record
        Absent_type_code : string;
        Salary_indicator : real;
        Absent_day : integer;
    end;
    Tcompany = class
        Company_id : string;
        . . .
        Annual_holiday_data : Annual_holiday;
        Other_absent_data : Other_absent;
        . . .
    end;
```

- ใช้โครงสร้างข้อมูลแบบ วาเรียนท์เรคคอร์ด (variant record) ซึ่งเป็นเรคคอร์ดที่มีโครงสร้างข้อมูลอนุญาตให้สมาชิกของเรคคอร์ดสามารถเปลี่ยนแปลงได้ เป็นโครงสร้างข้อมูลสำหรับข้อมูลนโยบายการหยุดงาน วิธีการนี้แสดงในภาษาปาสคาลได้ดังนี้

```
Type
Tabsent_data = record
        Absent_type_code : string;
        case Absent_indicator : Tabsent_type of
                Annual :
                        (Number_of_annual_holiday : integer;
                         Year_of_work : integer);
```

```
        Other :
                (Salary_indicator : real;
                 Absent_day : integer);
    end;
    Tcompany = class
        Company_id : string;
        . . .
        Absent_data : Tabsent_data;
        . . .
    end;
```

6. ในข้อ 5 นั้นปัญหาที่เกิดขึ้นเนื่องมาจากการที่คุณสมบัติของคลาสนั้นถูกแบ่งออกเป็นสับไทป์ แต่ว่ายังเหลือปัญหาอีกจุดหนึ่งคือ ข้อมูลรายได้นั้นถูกแบ่งออกไปตามประเภทของการจ้างงาน ซึ่งพิจารณาแล้วจะเห็นว่าการที่ข้อมูลรายได้ถูกแบ่งเป็นสับไทป์ เนื่องมาจากคลาสพนักงานที่เป็นเจ้าของคุณสมบัตินั้นถูกแบ่งออกเป็นประเภท ผิดกับในข้อ 5 ซึ่งตัวนโยบายการหยุดงานนั้นแบ่งเป็นประเภทด้วยตัวของมันเอง ไม่ใช่เนื่องมาจากบริษัทแบ่งออกเป็นประเภท ดังนั้นจึงต้องมีวิธีในการจัดการปัญหานี้ โดยการแบ่งพนักงานออกเป็นคลาสย่อยๆตามประเภทการจ้างงาน และข้อมูลรายได้ที่ถูกแบ่งเป็นสับไทป์นั้นก็เป็นคุณสมบัติของคลาสย่อยของพนักงานประเภทต่างๆ ซึ่งการที่แบ่งพนักงานออกเป็นคลาสย่อยนี้จะมีผลถึงวิธีการที่สัมพันธ์กับคลาสย่อยนั้นด้วย ดังจะกล่าวถึงในข้อต่อไป

ต่อจากนั้นเราจะเห็นว่าพนักงานรายวัน+รายชิ้นนั้นมีคุณสมบัติทางด้านข้อมูลรายได้เหมือนกับพนักงานรายวัน รวมกับพนักงานรายชิ้น แต่เนื่องจากภาษาปาสคาลไม่สนับสนุนการทำ multi-inheritance ได้ (หมายความว่า สับคลาสใดๆจะสามารถมีซุปเปอร์คลาสได้เพียงคลาสเดียวเท่านั้น) ดังนั้นเราจะเลือกให้พนักงานรายวัน+รายชิ้นนั้นเป็นคลาสย่อยของพนักงานรายวัน เนื่องจากว่า ถึงแม้ว่าพนักงานรายวัน+รายชิ้นจะมีคุณสมบัติในส่วนเกี่ยวกับการทำงานรายชิ้นเหมือนกับพนักงานรายชิ้นก็ตาม แต่ว่าขั้นตอนในการคำนวณรายได้ก็มีข้อแตกต่างกัน ดังนั้นถึงแม้ว่า เราจะให้พนักงานรายวัน+รายชิ้นเป็นคลาสย่อยของพนักงานรายชิ้น เมื่อพนักงานรายวัน+รายชิ้นได้รับถ่ายทอดวิธีการคำนวณรายได้จากพนักงานรายชิ้นไปก็จะต้องถูกโอเวอร์ไรด์ไปอยู่ดี ต่างจากเมื่อให้พนักงานรายวัน+รายชิ้นเป็นคลาสย่อยของพนักงานรายวัน ซึ่งสามารถจะถ่ายทอดวิธีการคำนวณรายได้จากค่าแรงรายวันไปได้โดยไม่ต้องเปลี่ยนแปลง

ผลลัพธ์ : คลาสพนักงานแบ่งออกเป็นคลาสย่อยดังนี้

- คลาสของพนักงานรายวัน
- คลาสของพนักงานรายชิ้น
- คลาสของพนักงานรายเดือน

และคลาสของพนักงานรายวัน มีคลาสของพนักงานรายวัน+รายชิ้นเป็นคลาสย่อยอีกคลาสหนึ่ง

7. ต่อจากการพิจารณาคุณสมบัติต่างๆของคลาสต่างๆ แล้วก็มาถึงการพิจารณาว่าคลาสต่างๆนั้นจะมีวิธีการอะไรบ้าง และในกรณีที่คลาสนั้นเป็นสับคลาสนั้นจะมีการเปลี่ยนแปลงแก้ไขวิธีการในสับคลาสที่ได้รับถ่ายทอดลงไปอย่างไรบ้าง

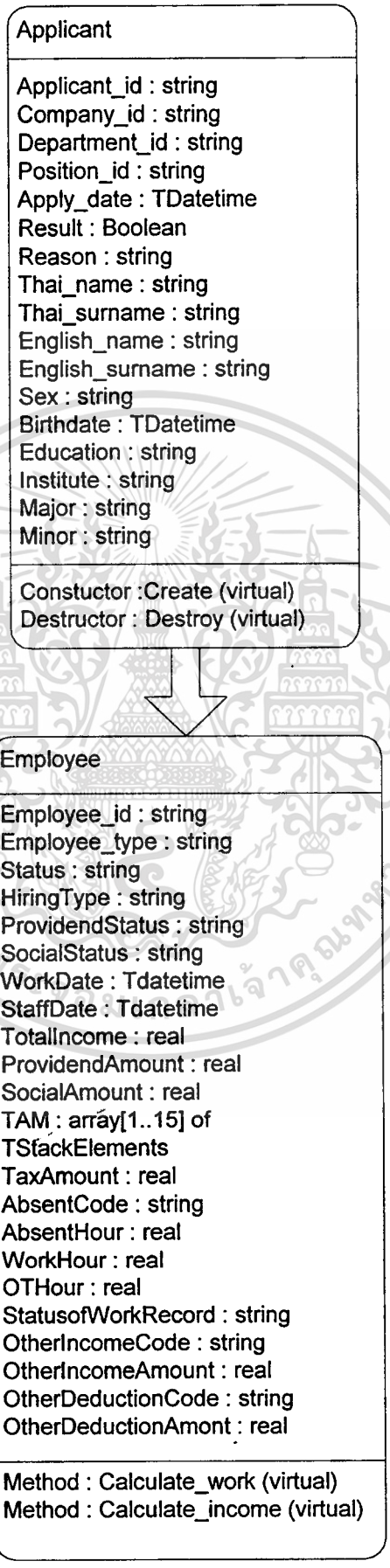
จากผังการไหลของข้อมูลจะพบว่าโพรเซสที่สำคัญของระบบคือ การคำนวณเวลาทำงานและการคำนวณรายได้ ซึ่งทั้งสองโพรเซสนี้เป็นโพรเซสที่เกี่ยวข้องกับพนักงานทั้งนั้น ดังนั้นเราจะสรุปว่า คลาสของพนักงานนั้นมีวิธีการคำนวณเวลาทำงาน และคำนวณรายได้ จากนั้นเราจะมาพิจารณาว่าเมื่อวิธีการทั้งสองนั้นถูกถ่ายทอดไปยังคลาสย่อยต่างๆแล้วมีการเปลี่ยนแปลงอย่างไรบ้าง

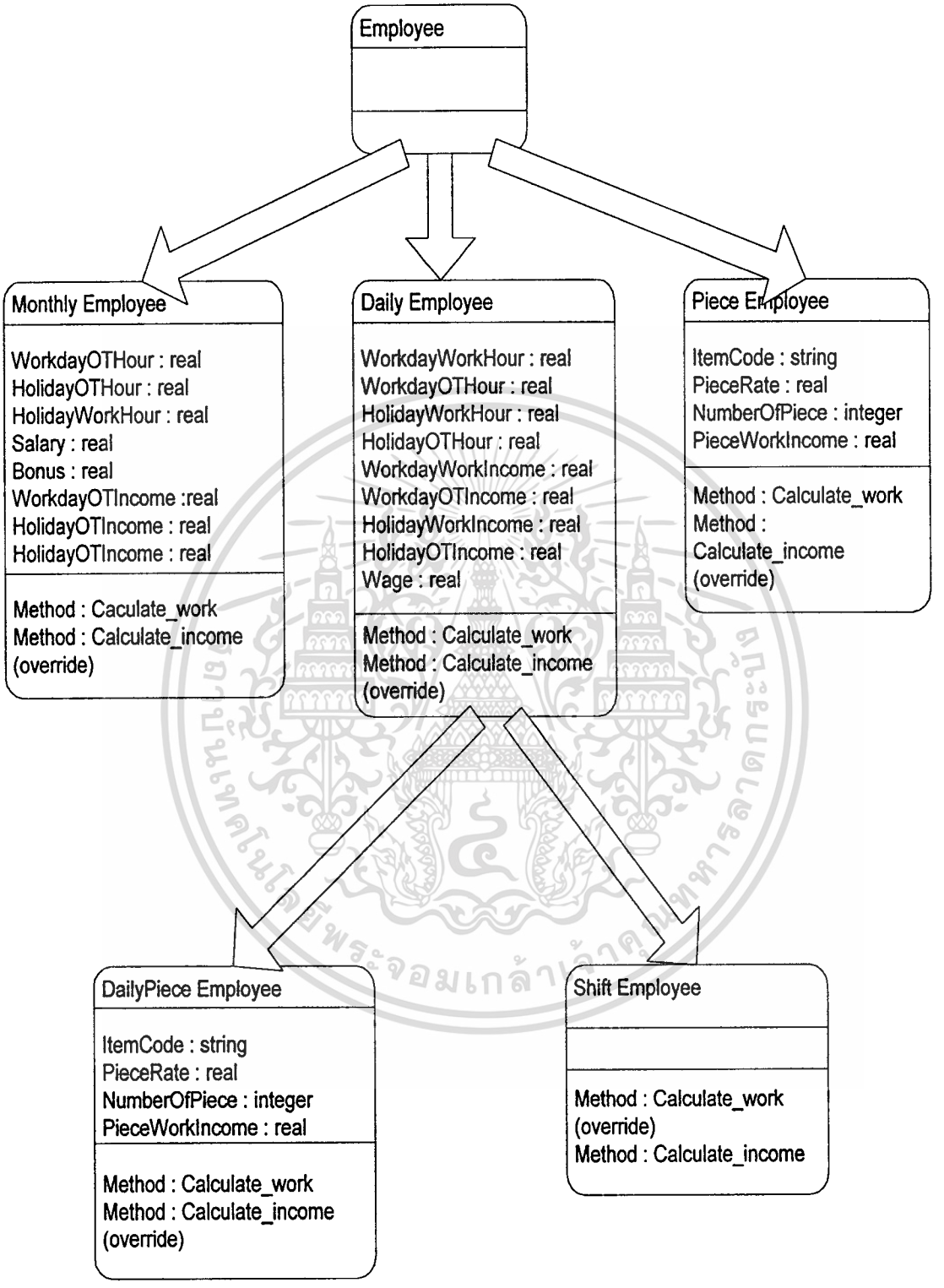
สำหรับวิธีการคำนวณเวลาทำงานนั้น พนักงานทุกประเภทจะคำนวณเวลาทำงานเหมือนกัน ยกเว้นสำหรับพนักงานกะ ซึ่งเป็นส่วนหนึ่งของพนักงานรายวันที่มีวิธีการคำนวณเวลาทำงานแตกต่างออกไป ดังนั้น เราจะได้ว่าคลาสของพนักงานรายวันนั้น จะมีแบ่งออกเป็นคลาสย่อยอีก 1 คลาส คือคลาสของพนักงานกะ ซึ่งจะมีขั้นตอนในการคำนวณเวลาทำงานแตกต่างออกไป หรือสามารถกล่าวในเชิงของออปเจค-โอเรียนเต็ดได้ว่า คลาสพนักงานกะนั้นโอเวอร์ไรด์วิธีการคำนวณเวลาทำงานที่ได้รับถ่ายทอดมาจากคลาสพนักงาน

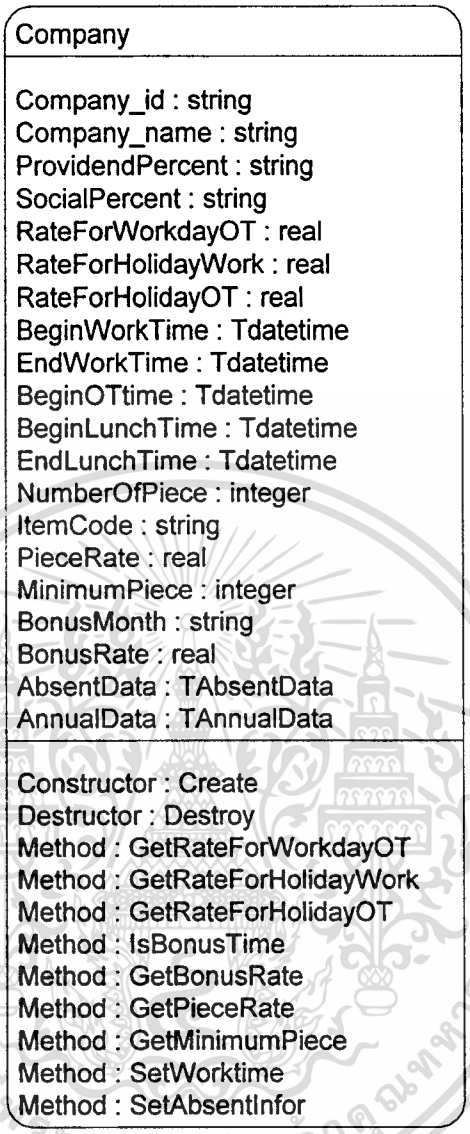
ในส่วนของวิธีการคำนวณรายได้ของพนักงานนั้นก็จะถูกโอเวอร์ไรด์ในคลาสย่อยของพนักงานประเภทต่างๆ ให้เป็นไปตามขั้นตอนการคำนวณรายได้ของพนักงานประเภทต่างๆดังที่ได้กล่าวมาแล้ว

8.ในส่วนของคลาสบริษัทนั้น วิธีการส่วนใหญ่จะเป็นการนำค่านโยบายต่างๆที่ใช้ในการทำงานขึ้นมา

จาก 8 ขั้นตอนที่กล่าวมาแล้ว สามารถวาดภาพการแตกสาขาของคลาส(Class Heirarchy)ได้ดังนี้


Applicant
Applicant_id : string
Company_id : string
Department_id : string
Position_id : string
Apply_date : TDatetime
Result : Boolean
Reason : string
Thai_name : string
Thai_surname : string
English_name : string
English_surname : string
Sex : string
Birthdate : TDatetime
Education : string
Institute : string
Major : string
Minor : string
Constuctor :Create (virtual)
Destructor : Destroy (virtual)
Employee
Employee_id : string
Employee_type : string
Status : string
HiringType : string
ProvidendStatus : string
SocialStatus : string
WorkDate : Tdatetime
StaffDate : Tdatetime
TotalIncome : real
ProvidendAmount : real
SocialAmount : real
TAM : arráy[1..15] of
TStackElements
TaxAmount : real
AbsentCode : string
AbsentHour : real
WorkHour : real
OTHour : real
StatusofWorkRecord : string
OtherIncomeCode : string
OtherIncomeAmount : real
OtherDeductionCode : string
OtherDeductionAmont : real
Method : Calculate_work (virtual)
Method : Calculate_income (virtual)


Employee
Monthly Employee
WorkdayOTHour : real
HolidayOTHour : real
HolidayWorkHour : real
Salary : real
Bonus : real
WorkdayOTIncome :real
HolidayOTIncome : real
HolidayOTIncome : real
Method : Caculate_work
Method : Calculate_income
(override)
Daily Employee
WorkdayWorkHour : real
WorkdayOTHour : real
HolidayWorkHour : real
HolidayOTHour : real
WorkdayWorkIncome : real
WorkdayOTIncome : real
HolidayWorkIncome : real
HolidayOTIncome : real
Wage : real
Method : Calculate_work
Method : Calculate_income
(override)
Piece Employee
ItemCode : string
PieceRate : real
NumberOfPiece : integer
PieceWorkIncome : real
Method : Calculate_work
Method :
Calculate_income
(override)
DailyPiece Employee
ItemCode : string
PieceRate : real
NumberOfPiece : integer
PieceWorkIncome : real
Method : Calculate_work
Method : Calculate_income
(override)
Shift Employee
Method : Calculate_work
(override)
Method : Calculate_income



รูปที่ 3-1 การแตกสาขาของคลาส

ความหมายของสัญลักษณ์ที่ใช้

1.กล่องสี่เหลี่ยมขอบมน หมายถึง คลาสหนึ่งคลาส

ส่วนบนสุด คือ ชื่อคลาส

ส่วนกลาง คือ คุณสมบัติของคลาส

ส่วนล่างสุด คือ วิธีการของคลาส

2. คำว่า override หมายถึงว่า วิธีการที่ได้รับถ่ายทอดมาจากซุปเปอร์คลาสนั้นถูกแก้ไขเปลี่ยนแปลงไปจากเดิม (เกิดโพลีมอร์ฟิสซึ่ม)

3. คุณสมบัติที่ถ่ายทอดมาจากซุปเปอร์คลาสจะไม่เขียนไว้ในคลาสย่อยอีก

บทที่ 4
การทดลองและผลการทดลอง

ในบทนี้จะเป็นการกล่าวถึงการทำงานของระบบ ซึ่งเป็นการนำเอาผลที่ได้จากการออกแบบในบทที่ 3 มาเขียนโปรแกรม โดยจะได้ทำการอธิบายไปพร้อมกับแสดงหน้าจอหลักบางส่วนเพื่อให้ได้เห็นภาพรวมของระบบมากขึ้น

การทำงานของระบบจัดการทรัพยากรมนุษย์นี้แบ่งได้ออกเป็น 5 ส่วนใหญ่ๆ โดยที่ 3 ส่วนแรกนั้นจะเกี่ยวข้องกับการแก้ไขเปลี่ยนแปลงข้อมูลของผู้สมัครงาน, พนักงาน และข้อมูลของบริษัท ตามลำดับ สำหรับส่วนที่ 4 นั้นเป็นส่วนที่เกี่ยวข้องกับการคำนวณเวลาทำงาน และส่วนที่ 5 เป็นส่วนที่เกี่ยวข้องกับการคำนวณรายได้ของพนักงาน ซึ่งจะได้อธิบายรายละเอียดของแต่ละส่วนดังต่อไปนี้

4.1 ส่วนข้อมูลผู้สมัครงาน

4.1.1ส่วนประกอบของข้อมูลผู้สมัครงาน

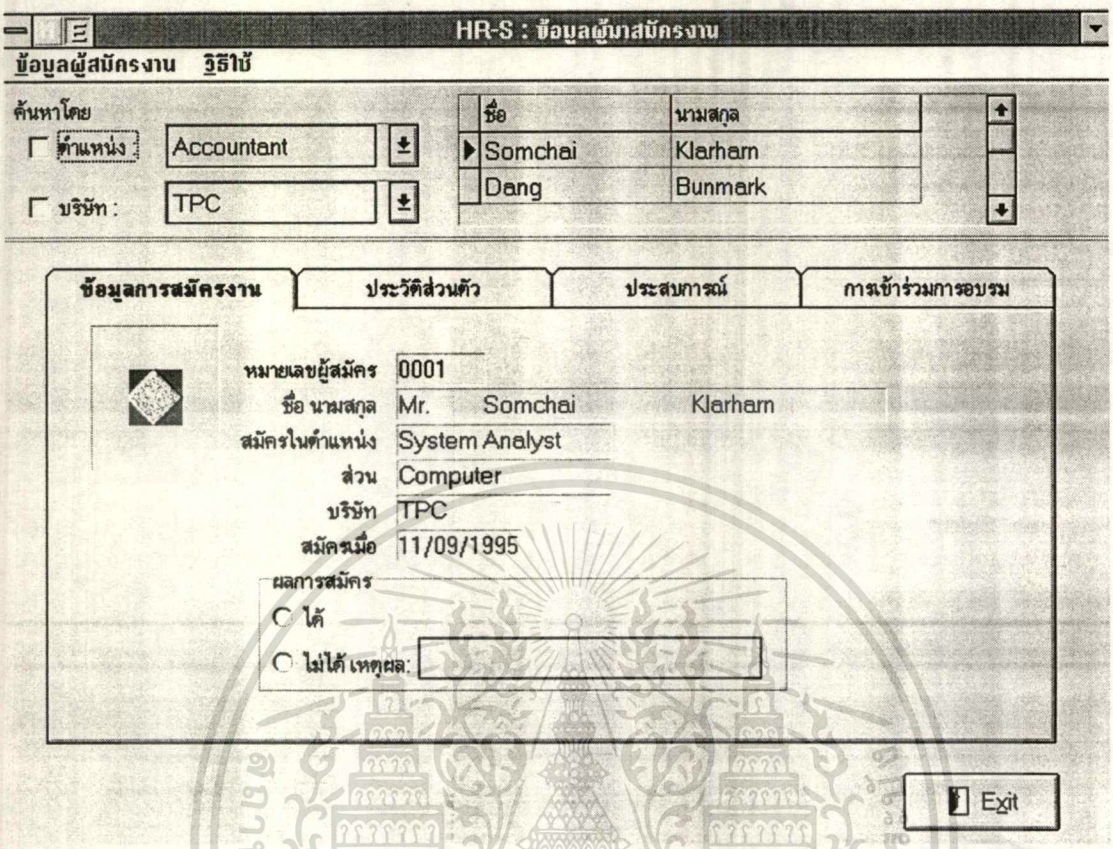
ข้อมูลของผู้สมัครงานประกอบไปด้วยข้อมูลการสมัครงาน เช่นว่ามาสมัครตำแหน่งอะไร เมื่อใด และผลการสมัครเป็นอย่างไร ,ข้อมูลส่วนตัว เช่น อายุ วุฒิการศึกษา เป็นต้น , ประสบการณ์การทำงาน และการเข้าร่วมการอบรมต่างๆของผู้สมัครงาน

4.1.2 วัตถุประสงค์ของส่วนข้อมูลผู้สมัครงาน

การเก็บข้อมูลของผู้สมัครงานไว้ มีจุดมุ่งหมายเพื่อที่จะสามารถนำมาพิจารณาได้ใหม่ เมื่อบริษัทต้องการรับพนักงานใหม่ จะได้ทำการตรวจสอบคุณสมบัติต่างๆ และความสนใจของผู้สมัครงานว่าเหมาะสมที่จะเรียกมาพบต่อไปหรือไม่

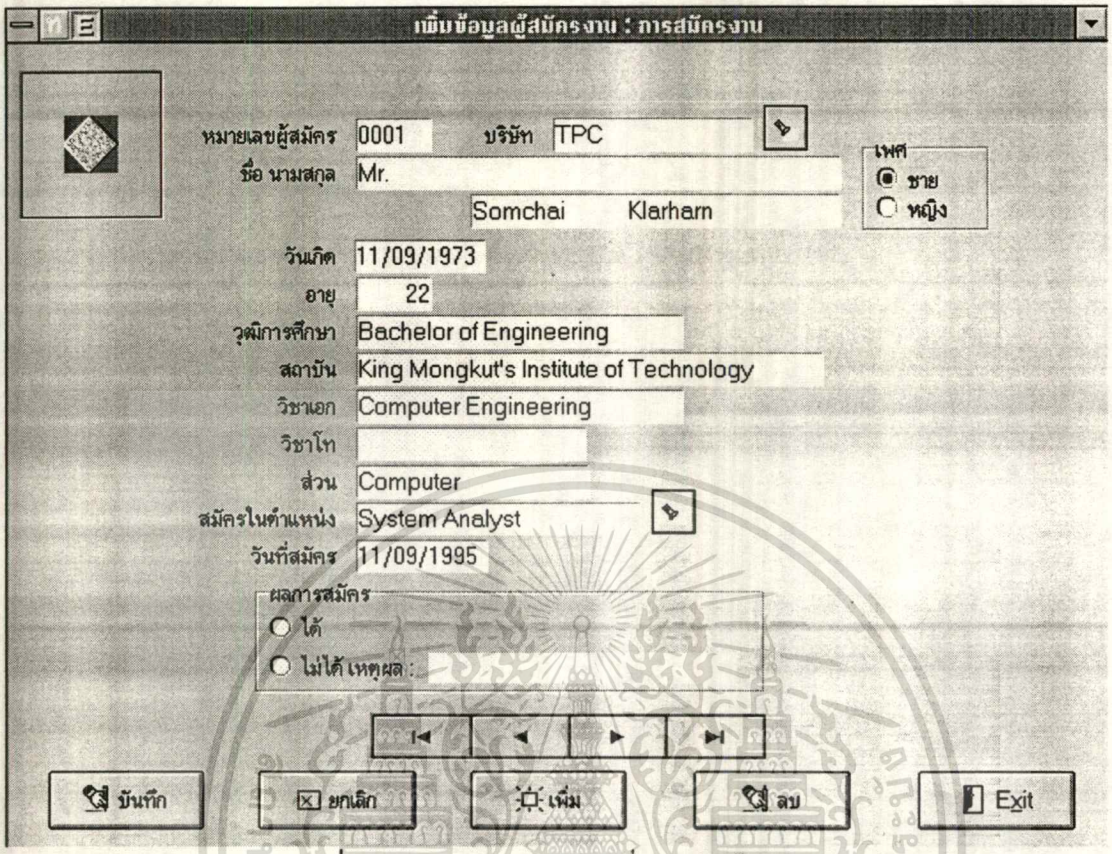
4.1.3 การทำงานของส่วนข้อมูลผู้สมัครงาน

หน้าที่ของส่วนนี้ก็คือเพื่อให้ผู้ใช้สามารถเรียกดู แก้ไข เปลี่ยนแปลงข้อมูลต่างๆของผู้สมัครงานได้ โดยมีหน้าจอหลักของส่วนนี้ ดังรูป



รูปที่4-1 หน้าจอหลักของส่วนข้อมูลผู้สมัครงาน

หน้าจอหลักนี้จะเป็นส่วนสำหรับเรียกดูข้อมูลของผู้สมัครงาน ซึ่งแบ่งออกเป็นส่วนๆ ตามที่ได้กล่าวมาแล้ว สำหรับการแก้ไขข้อมูลของผู้สมัครงานนั้นก็ทำได้โดยการเลือกเมนู เพิ่ม หรือแก้ไขและเลือกข้อมูลในส่วนที่ต้องการจะเพิ่มหรือแก้ไข สำหรับการเพิ่มข้อมูลใหม่นั้นถ้าเป็นข้อมูลของผู้สมัครงานที่ยังไม่ปรากฏอยู่ในระบบ ผู้ใช้จะต้องเพิ่มผู้สมัครงานใหม่เข้าไปเสียก่อน โดยเลือกเมนูเพิ่ม | ข้อมูลผู้สมัครงาน เสียก่อน ซึ่งจะมีหน้าจอในการทำงานดังนี้

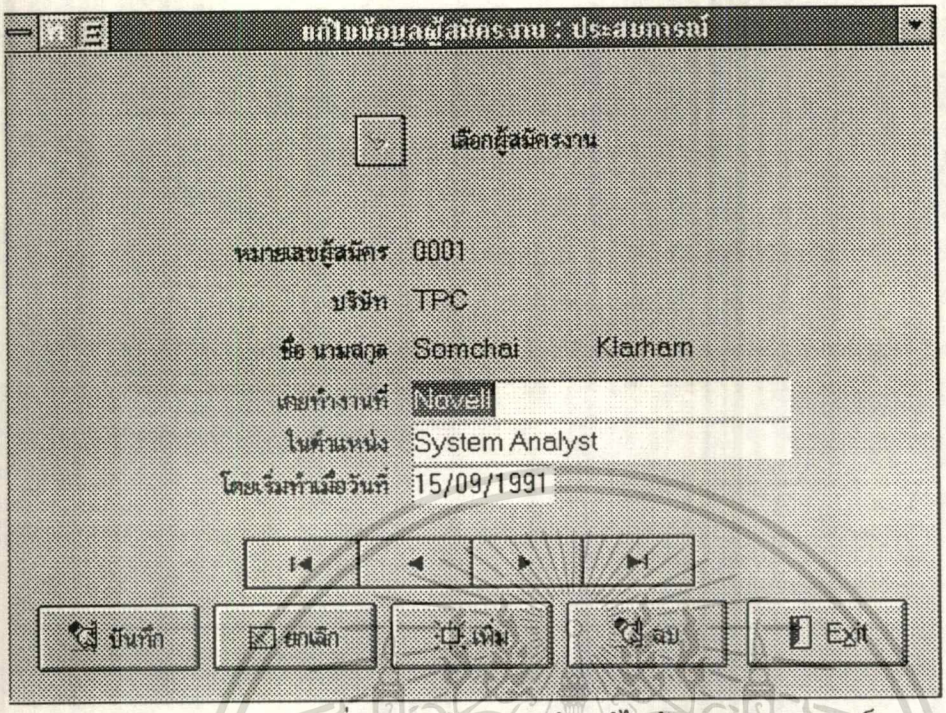


รูปที่ 4-2 หน้าจอสำหรับเพิ่มและแก้ไขข้อมูลผู้สมัครงาน

ในการเพิ่มข้อมูลผู้สมัครงาน ผู้ใช้จะต้องใส่ข้อมูล รหัสผู้สมัครและบริษัทที่ผู้สมัครมาสมัครงานก่อนถึงจะอนุญาตให้บันทึกข้อมูลได้

หลังจากที่เพิ่มผู้สมัครงานคนใหม่เข้าไปแล้ว จึงจะสามารถเพิ่มข้อมูลส่วนอื่นๆ เช่น ประสบการณ์การทำงานของผู้สมัครงานเข้าไปได้ เพราะในการเพิ่มข้อมูลส่วนอื่นๆของผู้สมัครงาน ผู้ใช้จะต้องทำการเลือกว่าข้อมูลที่จะเพิ่มเข้าไปนั้นเป็นของผู้สมัครงานคนใด ซึ่งถ้าหากว่าไม่ทำการเพิ่มผู้สมัครคนนั้นเข้าไปก่อนแล้ว ระบบก็จะไม่รู้จักผู้สมัครงานดังกล่าว ทำให้ไม่สามารถเพิ่มข้อมูลในส่วนอื่นๆได้

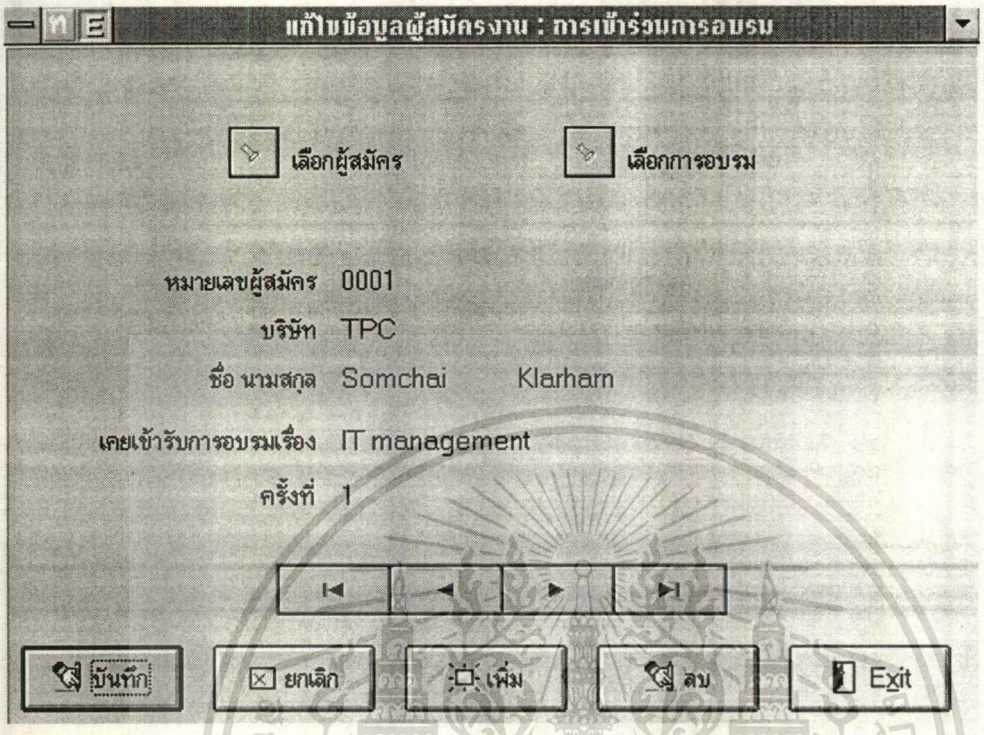
ตัวอย่างของการเพิ่มหรือแก้ไขข้อมูลอื่นๆของพนักงาน เช่น ข้อมูลประสบการณ์ของผู้สมัครงาน จะมีหน้าจอการทำงาน ดังต่อไปนี้



รูปที่4-3 หน้าจอสำหรับแก้ไขข้อมูลประสบการณ์

ในการเพิ่มข้อมูลใหม่ ผู้ใช้จะต้องเลือกผู้สมัครงานที่เป็นเจ้าของข้อมูลนั้น โดยการกดปุ่มเพื่อเลือกผู้สมัครงานจากรายชื่อผู้สมัครงานทั้งหมด การทำอย่างนี้เพื่อป้องกันข้อผิดพลาดในการใส่รหัสผู้สมัครหรือรหัสบริษัทของผู้ใช้ และป้องกันไม่ให้มีการใส่ข้อมูลของผู้สมัครงานที่ยังไม่ได้เพิ่มเข้าไปในระบบ

สำหรับหน้าจอสำหรับการแก้ไขข้อมูลส่วนอื่น ๆของผู้สมัครงานเป็นดังนี้



รูปที่ 4-4 หน้าจอสำหรับแก้ไขข้อมูลการอบรม

4.2 ส่วนข้อมูลพนักงาน

4.2.1 ส่วนประกอบของข้อมูลพนักงาน

ข้อมูลของพนักงานจะถูกแบ่งออกเป็นส่วนๆดังต่อไปนี้

- ข้อมูลการทำงาน
- ประวัติส่วนตัว
- ผู้ค้ำประกัน
- ประสบการณ์
- การเข้าร่วมการอบรม
- ประวัติการทำงาน
- ผลการประเมินการทำงาน

และนอกจากนี้ก็ยังมีข้อมูลเกี่ยวกับรายได้ของพนักงาน และบันทึกการหยุดงานต่างๆอีกด้วย ซึ่งรายละเอียดต่างๆนั้นได้อธิบายไว้แล้วในส่วนของโมเดล ER

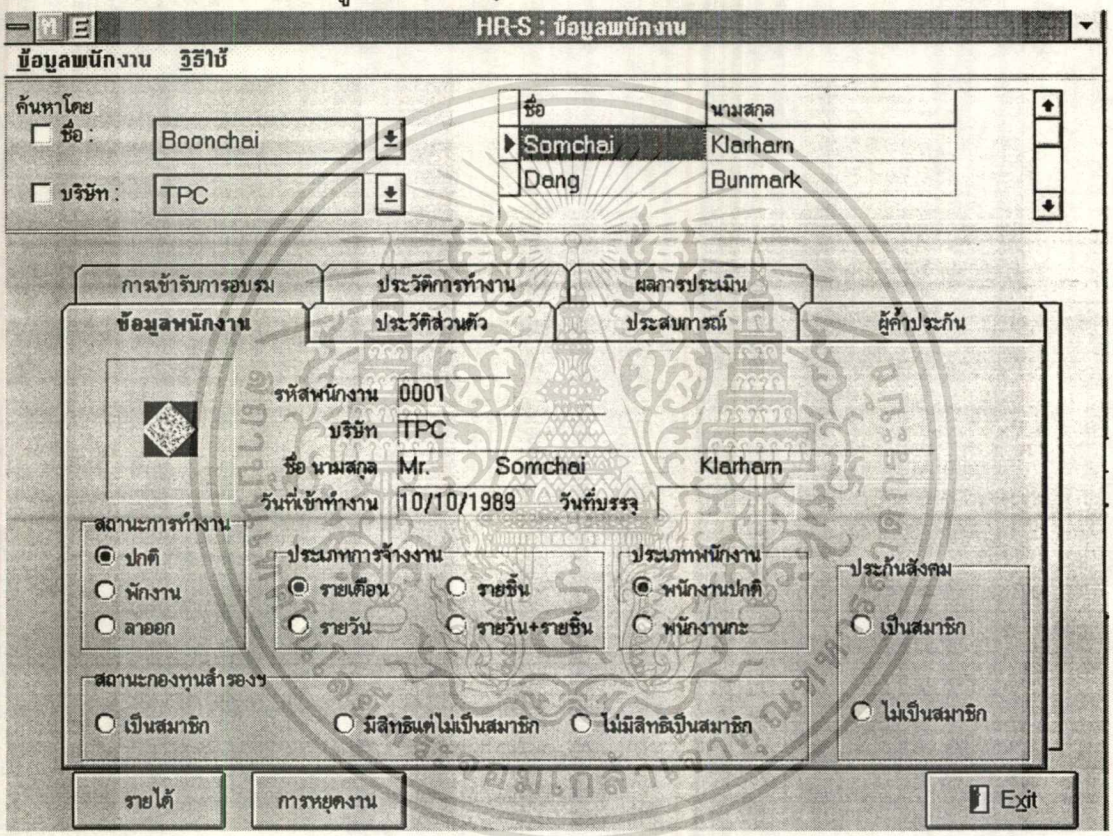
4.2.2 วัตถุประสงค์ของส่วนข้อมูลพนักงาน

การเก็บข้อมูลพนักงานในส่วนต่างๆที่ได้กล่าวมาแล้วนั้น นอกจากจะมีจุดมุ่งหมายเพื่อไว้เรียกดู หรือแก้ไขเปลี่ยนแปลงตามปกติแล้ว ก็ยังต้องเก็บไว้เพื่อใช้ประโยชน์ในการทำงานส่วนอื่นๆด้วย เช่น การเก็บข้อมูลรายได้ของพนักงานนั้น ก็เพื่อนำไปใช้ในการคำนวณรายได้

ของพนักงานด้วย

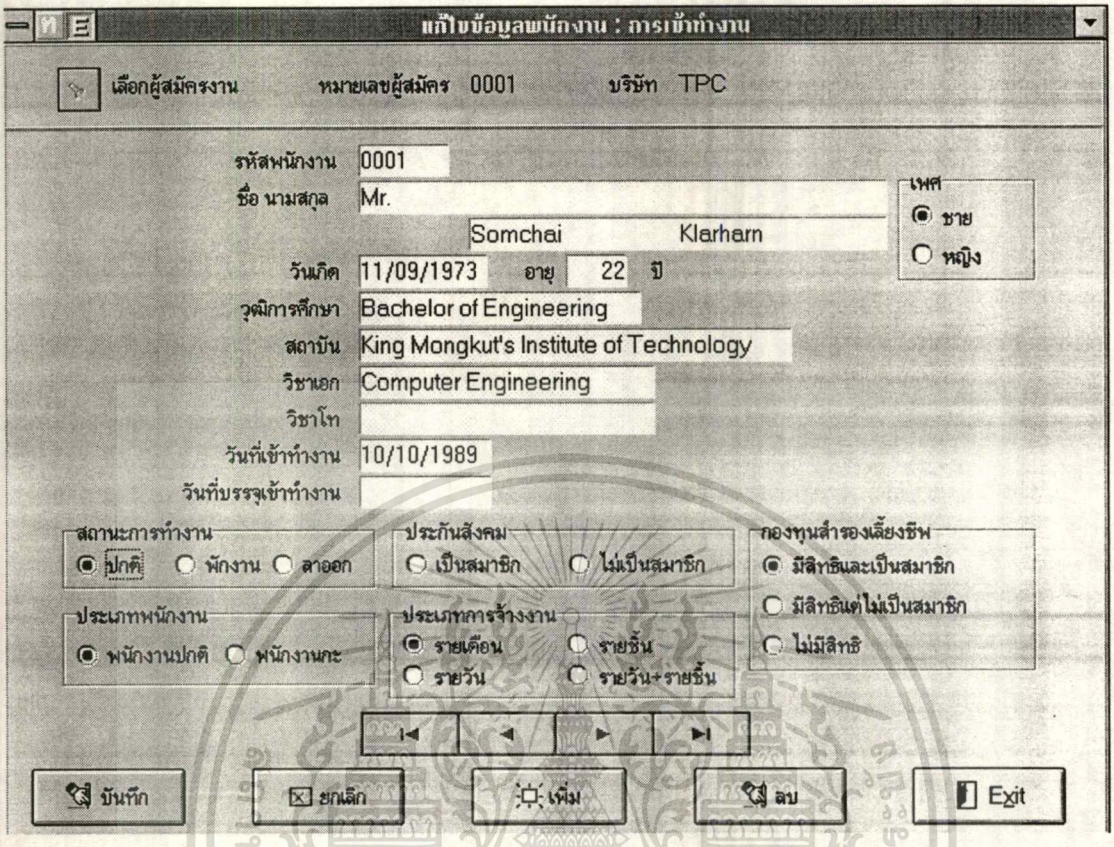
4.2.3 การทำงานของส่วนข้อมูลพนักงาน

ลักษณะการทำงานในส่วนนี้มีความใกล้เคียงกับการทำงานในส่วนข้อมูลผู้สมัครงานมาก เพียงแต่ว่าข้อมูลที่เกี่ยวข้องด้วยนั้นมีมากกว่า ดังรูปที่ เช่นเดียวกับการเพิ่มข้อมูลใหม่ลงไปในระบบ ผู้ใช้จะต้องระบุว่าเจ้าของข้อมูลนั้นเป็นใคร นั่นคือ ข้อมูลที่จะเพิ่มเข้าไปนั้นเป็นของพนักงานคนใด และถ้าข้อมูลนั้นเป็นของพนักงานคนใหม่ ผู้ใช้จะต้องเพิ่มพนักงานคนใหม่เข้าไปในระบบเสียก่อนจึงจะเพิ่มข้อมูลในส่วนอื่นๆ ได้



รูปที่ 4-5 หน้าจอหลักของส่วนข้อมูลพนักงาน

การเพิ่มข้อมูลพนักงานใหม่ รวมถึงการแก้ไขข้อมูลของพนักงานเดิมนั้นมีหน้าจอดังรูป



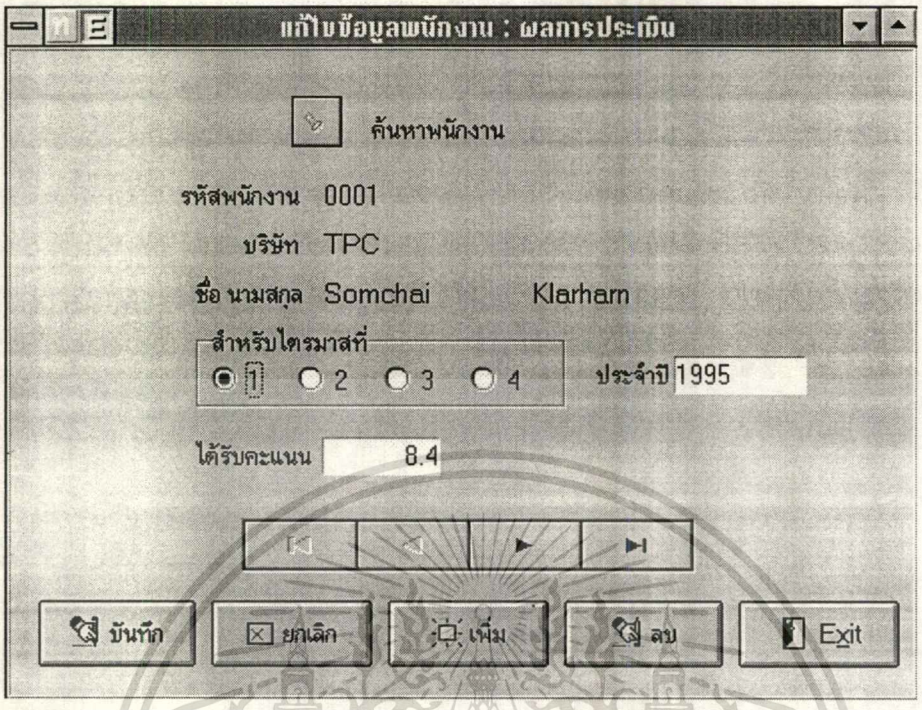
รูปที่ 4-6 หน้าจอแก้ไขข้อมูลพนักงาน

ในกรณีที่เป็นการเพิ่มพนักงานใหม่ ผู้ใช้จะต้องเลือกผู้เป็นพนักงานใหม่จากข้อมูลผู้สมัครที่มีอยู่เดิม ไม่ใช่เพิ่มเข้าไปใหม่ได้โดยอิสระ และถ้าผู้ใช้ยังไม่ได้เลือกข้อมูลผู้สมัครคนใดที่จะเป็นพนักงานใหม่ ระบบจะไม่ยอมให้บันทึกข้อมูลลงไป

หลังจากที่เพิ่มข้อมูลพนักงานใหม่ เข้าไปแล้ว ก็สามารถที่จะเพิ่มข้อมูลในส่วนอื่นๆของพนักงานคนนั้นเข้าไปได้ ซึ่งมีหน้าจอในการทำงานดังต่อไปนี้

รูปที่ 4-7 หน้าจอแก้ไขข้อมูลผู้ค้ำประกัน

รูปที่ 4-8 หน้าจอแก้ไขข้อมูลประวัติการทำงาน



รูปที่ 4-9 หน้าจอแก้ไขข้อมูลผลการประเมิน

4.3 ข้อมูลบริษัท

4.3.1 ส่วนประกอบของข้อมูลบริษัท

ข้อมูลบริษัทจะแบ่งได้กว้างๆ เป็น 2 ส่วน คือส่วนที่เกี่ยวกับโครงสร้างองค์กร และส่วนที่เกี่ยวกับนโยบายต่างๆ

ส่วนทีเกี่ยวกับโครงสร้างขององค์กรนั้นจะเป็นข้อมูลที่เกี่ยวกับว่า บริษัทนี้ประกอบไปด้วยฝ่ายใดบ้าง และฝ่ายต่างๆนั้นประกอบไปด้วยส่วนต่างๆอะไรบ้าง และยังรวมไปถึงตำแหน่งงานในบริษัท ฝ่ายและส่วนดังกล่าวด้วย

ส่วนที่เกี่ยวกับนโยบายบริษัทนั้น ก็ตัวอย่างเช่น งวดการจ่ายโบนัส ,นโยบายการลาหยุด ,อัตราการคำนวณรายชิ้น เป็นต้น ซึ่งรายละเอียดในส่วนนี้ได้กล่าวไว้แล้วในส่วนของโมเดล ER ของระบบ

4.3.2 วัตถุประสงค์ของส่วนของข้อมูลบริษัท

การจัดเก็บข้อมูลบริษัทในส่วนต่างๆนั้นนอกจากจะเพื่อความเป็นระเบียบ สามารถเรียกดู แก้ไขได้ง่ายแล้ว ก็ยังมีการนำไปใช้ในการทำงานส่วนอื่นของระบบด้วยเช่นเดียวกัน โดยเฉพาะในส่วนของนโยบายบริษัท ซึ่งจะถูกนำไปใช้มาก ดังที่ได้กล่าวแล้วในส่วนของผังการไหลของข้อมูลและโมเดล ER ของระบบ

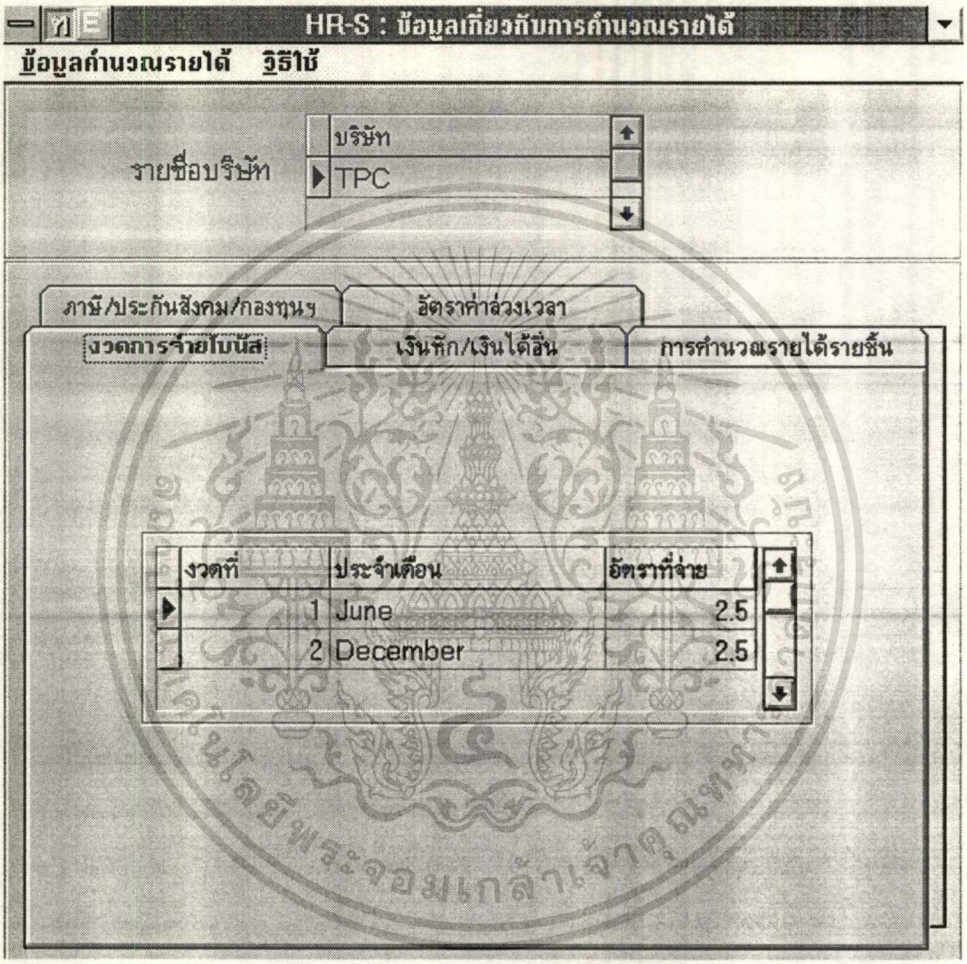
4.3.3 การทำงานของส่วนข้อมูลบริษัท

ในส่วนการทำงานของส่วนข้อมูลบริษัทนั้น จะแบ่งออกเป็น 5 หน้าจอ คือ

4.3.3.1.เป็นข้อมูลเกี่ยวกับการคำนวณรายได้

ประกอบด้วย งวดการจ่ายโบนัส, รหัสของเงินหักและเงินได้ต่างๆ , อัตราคำนวณรายชิ้น, จำนวนชิ้นที่น้อยที่สุด ,ชิ้นงาน ,นโยบายด้านกองทุนสะสม ประกันสังคม และอัตราภาษี ,อัตราค่าล่วงเวลา

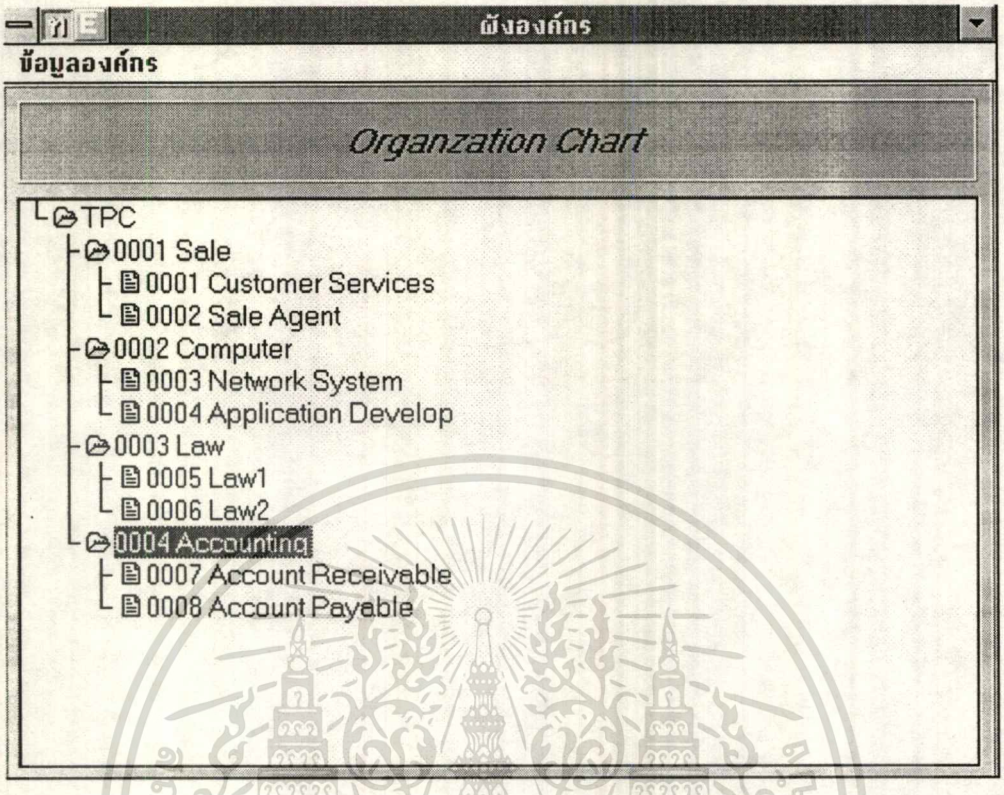
ซึ่งมีหน้าจอสำหรับทำงานดังนี้



รูปที่ 4-10 หน้าจอสำหรับส่วนข้อมูลคำนวณเกี่ยวกับการคำนวณรายได้

4.3.3.2 เป็นส่วนของนโยบายการหยุดงานและวันหยุด ได้แก่ นโยบายด้านวันหยุดของบริษัทและการลาหยุดประเภทต่างๆ

4.3.3.3 โครงสร้างองค์กร มีผังแสดงโครงสร้างของบริษัท และมีส่วนสำหรับแก้ไขข้อมูล ดังรูป



รูปที่ 4-11 หน้าจอสำหรับส่วนข้อมูลโครงสร้างองค์กร

4.3.3.4 วันเวลาทำงาน ใช้สำหรับเรียกดูและแก้ไขข้อมูลเวลาทำงานของพนักงานตามการจ้างงานแต่ละประเภท

4.4 ส่วนบันทึกเวลาทำงาน

4.4.1 ส่วนประกอบของส่วนบันทึกเวลาทำงาน

ส่วนบันทึกเวลาทำงานนี้จะเป็นส่วนทำงานเกี่ยวกับการคำนวณเวลาทำงาน ประกอบไปด้วยส่วนต่างๆ เช่น การแจ้งการทำงานล่วงเวลาล่วงหน้า , การแจ้งลางานล่วงหน้า ,การสั่งให้ระบบคำนวณเวลาทำงานของพนักงาน เป็นต้น การทำงานของส่วนนี้จะเทียบได้กับโพรเซสคำนวณเวลาทำงานในผังการไหลของข้อมูล และเกี่ยวพันกับข้อมูลในโมเดล ER เช่น บันทึกการลาล่วงหน้า (Advance absent record) , บันทึกการรูดบัตร (TAM record) และบันทึกการทำงาน (Work record) เป็นต้น

4.4.2 วัตถุประสงค์ของส่วนคำนวณเวลาทำงาน

การคำนวณเวลาทำงานทำเพื่อสรุปให้ผู้ใช้ได้รับทราบถึงสถิติการมาทำงานของพนักงาน รวมทั้งการขาด ลา มาสาย และยังนำผลที่ได้ไปใช้ในการคำนวณรายได้อีกด้วย

4.4.3 การทำงานของส่วนคำนวณเวลาทำงาน

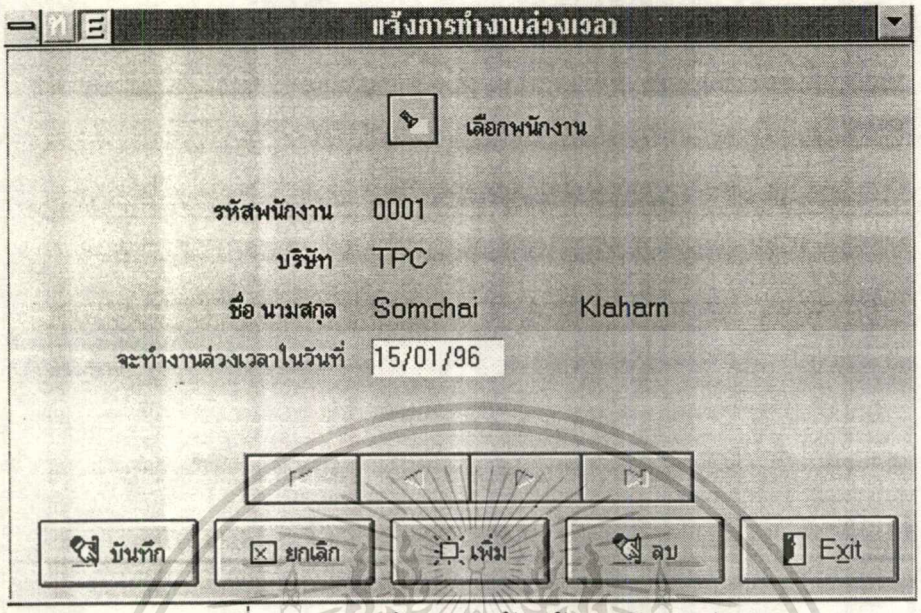
การทำงานในส่วนนี้จะมีส่วนเกี่ยวข้องกับโมเดลออปเจค-โอเรียนเต็ดที่เราได้สร้างขึ้นเพื่อการเขียนโปรแกรม ดังกล่าวในหัวข้อ 3.3 เพราะว่าในส่วนของโปรแกรมที่ทำการคำนวณ

เวลาทำงานนั้นก็คือวิธีการคำนวณเวลาทำงานของคลาสพนักงานนั่นเอง ดังนั้นการทำงานในส่วนนี้จะเกี่ยวพันกับทั้งการดูแลข้อมูลในฐานข้อมูลตามปกติ ดังเช่น ในส่วนที่ 1 ,2 และ 3 และทั้งการเขียนโปรแกรมแบบออปเจค-โอเรียนเต็ดด้วย

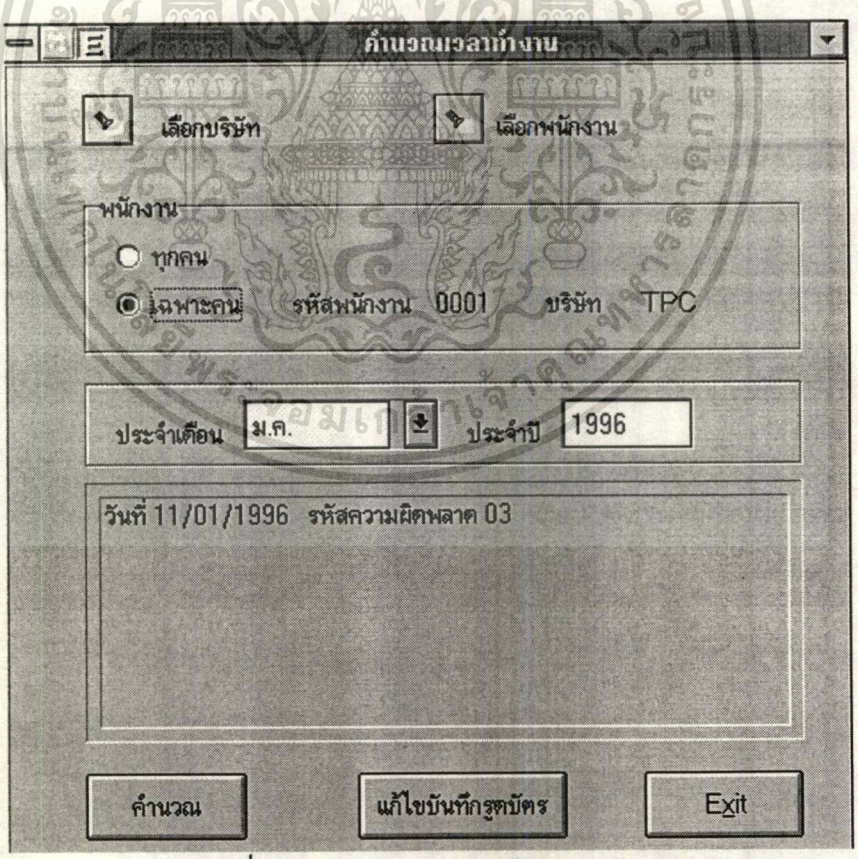
ในการคำนวณเวลาทำงานของพนักงานแต่ละคน ก็จะมีสร้างออปเจคของคลาสพนักงานขึ้นมาตามข้อมูลของพนักงานที่มีอยู่ เช่น เป็นพนักงานกะหรือไม่ ต่อจากนั้นก็เรียกใช้วิธีการคำนวณเวลาทำงานของคลาสพนักงานที่ได้เขียนไว้แล้ว

สำหรับหน้าจอในส่วนนี้ประกอบไปด้วยตัวอย่างดังนี้

รูปที่ 4-12 หน้าจอสำหรับแจ้งการลาล่วงหน้า



รูปที่ 4-13 หน้าจอสำหรับแจ้งการทำงานล่วงเวลา



รูปที่ 4-14 หน้าจอสำหรับการคำนวณเวลาทำงาน

4.5 ส่วนการคำนวณรายได้

4.5.1 ส่วนประกอบของส่วนคำนวณรายได้

การคำนวณรายได้นั้นนอกจากจะอาศัยข้อมูลจากบันทึกการทำงานที่เป็นผลสรุปของชั่วโมงทำงาน และชั่วโมงขาด ลา มาสาย ของพนักงานแล้ว ก็ยังมีข้อมูลส่วนอื่นๆที่ต้องนำมาพิจารณาด้วย ได้แก่ ส่วนบันทึกการทำงานรายชิ้น สำหรับพนักงานประเภทรายชิ้น และประเภทรายวัน+รายชิ้น , บันทึกเงินได้และเงินหักอื่นๆในเดือนนั้นๆ

4.5.2 วัตถุประสงค์ของส่วนคำนวณรายได้

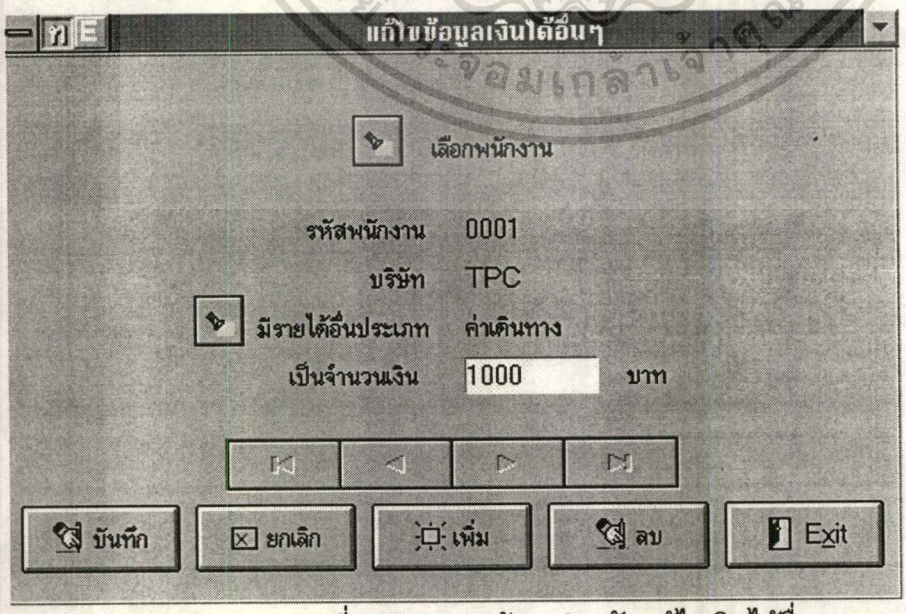
การทำงานในส่วนนี้ จะนำผลจากการคำนวณชั่วโมงทำงาน และส่วนอื่นๆที่กล่าวมาในข้อ 4.5.1 มาทำการคำนวณรายได้ให้กับพนักงานในแต่ละเดือน รวมทั้งคำนวณหักเงินประกันสังคม ,เงินกองทุนสำรองเลี้ยงชีพและ เงินภาษีหัก ณ ที่จ่าย ให้กับพนักงานรวมทั้งเก็บบันทึกไว้เพื่อตรวจสอบความถูกต้อง

4.5.3 การทำงานของส่วนคำนวณรายได้

การทำงานในส่วนนี้จะมีส่วนเกี่ยวข้องกับโมเดลออปเจค-โอเรียนเต็ดที่เราได้สร้างขึ้นเพื่อการเขียนโปรแกรม ดังกล่าวในหัวข้อ 3.3 เพราะว่าในส่วนของโปรแกรมที่ทำการคำนวณรายได้นั้นก็คือวิธีการคำนวณรายได้ของคลาสพนักงานนั่นเอง ดังนั้นการทำงานในส่วนนี้จะเกี่ยวพันกับทั้งการดูแลข้อมูลในฐานข้อมูลตามปกติ ดังเช่น ในส่วนที่ 1 ,2 และ 3 และทั้งการเขียนโปรแกรมแบบออปเจค-โอเรียนเต็ดด้วย

ในการคำนวณรายได้ของพนักงานแต่ละคน ก็จะมีสร้างออปเจคของคลาสพนักงานขึ้นมาตามข้อมูลของพนักงานที่มีอยู่ ซึ่งก็คือ ประเภทการจ้างงานของพนักงานเพื่อให้การเรียกใช้วิธีการนั้นถูกต้องเป็นไปตามที่โอเวอร์ไรด์ไว้ ต่อจากนั้นก็เรียกใช้วิธีการคำนวณเวลาทำงานของคลาสพนักงานประเภทการจ้างงานต่างๆที่ได้เขียนไว้แล้ว

สำหรับหน้าจอในส่วนนี้ประกอบไปด้วยตัวอย่างดังนี้



รูปที่ 4-15 หน้าจอสำหรับแก้ไขเงินได้อื่นๆ

บทที่ 5
วิจารณ์ สรุปและแนวทางในการพัฒนา

5.1 ความสามารถของระบบในโครงงานเมื่อเปรียบเทียบกับระบบที่ใช้งานอยู่จริง

เมื่อทำการเปรียบเทียบระหว่างระบบที่ทำการพัฒนาในโครงงานนี้กับระบบที่ใช้งานอยู่จริง หรือระบบที่มีขายอยู่ในท้องตลาด พบว่า ความสามารถพื้นฐานส่วนใหญ่นั้นสามารถทำได้ เช่น

- เก็บรายการพนักงานรายวัน รายเดือน รายชิ้น และรายวัน+รายชิ้นได้
- คิดค่าแรงรายวัน รายเดือน ชั่วโมงล่วงเวลา กะ การคำนวณค่าแรงต่อชิ้นงาน เงินกองทุนเลี้ยงชีพ เงินประกันสังคม และภาษีหัก ณ ที่จ่ายได้
- กำหนดการคิดล่วงเวลาของพนักงานแต่ละประเภทแยกออกจากกันได้

5.2 ข้อจำกัดของระบบ

การทำงานของระบบในโครงงานนี้ เป็นแบบเรียลไทม์ ซึ่งถ้าเปรียบเทียบกับการนำไปใช้งานจริงอาจจะทำให้มีอุปสรรคอยู่บ้าง เพราะถ้าข้อมูลมีปริมาณมาก ก็ควรจะมีการทำงานในลักษณะของแบตช์ด้วยเพื่อไม่ต้องให้ผู้ใช้ต้องรอการทำงานแบบเรียลไทม์ตลอดเวลา

นอกจากนี้การตรวจสอบลำดับการรูดบัตรของในโครงงานนี้ ใช้โครงสร้างข้อมูลที่รองรับบันทึกการรูดบัตรเป็นแบบอาร์เรย์ซึ่งมีขนาดจำกัดไว้ที่ 15 อัน ทำให้ถ้าหากว่าพนักงานมารูดบัตรในวันที่ต้องการคำนวณมากกว่า 15 ครั้งก็จะส่งผลให้การตรวจสอบลำดับการรูดบัตรเกิดข้อผิดพลาดได้

5.3 แนวทางในการพัฒนาระบบต่อไปในอนาคต

นอกจากข้อจำกัดที่ได้กล่าวไปแล้วในหัวข้อ 5.2 ซึ่งควรจะทำการแก้ไขให้ระบบมีความยืดหยุ่นมากขึ้น ส่วนนอกจากนี้ก็ควรจะปรับปรุงส่วนทั่วไป เช่น การแสดงข้อผิดพลาด (error message)ที่เกิดขึ้น ,การพิมพ์รายงานแสดงผลต่างๆที่จำเป็น

สำหรับในส่วนการทำงานของระบบ ก็ควรจะทำให้ระบบสามารถรองรับการคำนวณรายได้ของพนักงานได้หลากหลายมากขึ้น และอาจทำให้ระบบทำการคำนวณภาษีเงินได้บุคคลธรรมดาให้กับพนักงานในตอนสิ้นปีอีกด้วย ซึ่งจะทำให้ระบบสามารถใช้งานได้ครบวงจรมากขึ้น

ส่วนในแง่ของวิชาการ ถึงแม้ว่าส่วนหนึ่งของระบบโดยเฉพาะในส่วนของโพรเซสการคำนวณเวลาทำงานและคำนวณรายได้ จะถูกเขียนขึ้นด้วยเทคนิควิธีแบบออปเจค-โอเรียนเต็ดก็ตามแต่ก็ยังไม่สมบูรณ์ เนื่องด้วยเวลาในการศึกษาแนวความคิดทางด้านออปเจค-โอเรียนเต็ดนั้นไม่มากนัก ซึ่งยังคงมีทฤษฎีที่น่าสนใจอีกมาก รวมถึงเรื่องความสัมพันธ์ระหว่าง 2 โมเดลที่

ได้จากการวิเคราะห์ระบบแบบเอสเอสเอดีเอ็มกับโมเดลทางด้านออปเจค-โอเรียนเต็ดว่ามีความสัมพันธ์กันอย่างไร และความเข้ากันได้ระหว่างโมเดลทั้ง 2 แนวคิดนั้นเป็นอย่างไร

ภาคผนวก

เอสคิวแอล สคริปท์ไฟล์ ของฐานข้อมูลของระบบ

```
/* Extract Database C:\PROJECT\DB\INTERBAS\PROJECT.GDB */
CREATE DATABASE "C:\PROJECT\DB\INTERBAS\PROJECT.GDB" PAGE_SIZE 1024
;

/* Domain definitions */
CREATE DOMAIN ID AS CHAR(5) NOT NULL;

/* Table: AAR, Owner: PIMPUK */
CREATE TABLE AAR (EMPLOYEE_ID VARCHAR(5) NOT NULL,
      COMPANY_ID VARCHAR(5) NOT NULL,
      ABSENT_TYPE_CODE VARCHAR(5) NOT NULL,
      FROM_TIME DATE NOT NULL,
      TO_TIME DATE NOT NULL,
CONSTRAINT AAR_PK PRIMARY KEY (EMPLOYEE_ID, COMPANY_ID,
ABSENT_TYPE_CODE, FROM_TIME));

/* Table: ABSENT, Owner: PIMPUK */
CREATE TABLE ABSENT (COMPANY_ID VARCHAR(5) NOT NULL,
      ABSENT_TYPE_CODE VARCHAR(5) NOT NULL,
      ABSENT_TYPE_DESCRIPTION VARCHAR(40),
CONSTRAINT ABSENT_PK PRIMARY KEY (COMPANY_ID, ABSENT_TYPE_CODE));

/* Table: ABSENT_POLICY, Owner: PIMPUK */
CREATE TABLE ABSENT_POLICY (COMPANY_ID VARCHAR(5) NOT NULL,
      ABSENT_TYPE_CODE VARCHAR(5) NOT NULL,
      YEAR_OF_WORK SMALLINT NOT NULL,
      NUMBER_OF_HOLIDAY SMALLINT NOT NULL,
CONSTRAINT ABSENT_POLICY_PK PRIMARY KEY (COMPANY_ID,
ABSENT_TYPE_CODE, YEAR_OF_WORK));
```

```
/* Table: ABSENT_W, Owner: PIMPUK */
CREATE TABLE ABSENT_W (EMPLOYEE_ID VARCHAR(5) NOT NULL,
        COMPANY_ID VARCHAR(5) NOT NULL,
        ABSENT_TYPE_CODE VARCHAR(5) NOT NULL,
        WORK_DATE DATE NOT NULL,
        ABSENT_HOUR FLOAT);
/* Table: ANNNUAL, Owner: PIMPUK */
CREATE TABLE ANNNUAL (COMPANY_ID VARCHAR(5) NOT NULL,
        ABSENT_TYPE_CODE VARCHAR(5) NOT NULL,
        NUMBER_OF_HOLIDAY SMALLINT,
        YEAR_OF_WORK SMALLINT);

/* Table: APPLICAN, Owner: PIMPUK */
CREATE TABLE APPLICAN (TITLE VARCHAR(6),
        THAI_NAME VARCHAR(20),
        THAI_SURNAME VARCHAR(40),
        ENGLISH_NAME VARCHAR(15),
        ENGLISH_SURNAME VARCHAR(20),
        SEX VARCHAR(1),
        BIRTHDATE DATE,
        AGE SMALLINT,
        HIGHEST_EDUCATION VARCHAR(40),
        MAJOR VARCHAR(40),
        MINOR VARCHAR(40),
        INSTITUTE VARCHAR(50),
        POSITION_ID VARCHAR(5),
        APPLY_DATE DATE,
        RESULT_OF_APPLICATION VARCHAR(1),
        IMAGE BLOB SUB_TYPE 0 SEGMENT SIZE 80,
        REASON VARCHAR(50),
        APPLICANT_ID VARCHAR(5) NOT NULL,
        COMPANY_ID VARCHAR(5) NOT NULL,
CONSTRAINT APPLICANT_PK PRIMARY KEY (APPLICANT_ID, COMPANY_ID));
```

```
/* Table: APPLICANT, Owner: PIMPUK */
CREATE TABLE APPLICANT (COMPANY_ID VARCHAR(5),
        APPLICANT_ID VARCHAR(5),
        TITLE VARCHAR(4),
        THAI_NAME VARCHAR(20),
        THAI_SURNAME VARCHAR(40),
        ENGLISH_NAME VARCHAR(15),
        ENGLISH_SURNAME VARCHAR(20),
        SEX VARCHAR(1),
        BIRTHDATE DATE,
        AGE SMALLINT,
        HIGHEST_EDUCATION VARCHAR(40),
        MAJOR VARCHAR(40),
        MINOR VARCHAR(40),
        INSTITUTE VARCHAR(50),
        POSITION_ID VARCHAR(5),
        DEPARTMENT_ID VARCHAR(5),
        APPLY_DATE DATE,
        RESULT_OF_APPLICATION VARCHAR(20),
        IMAGE BLOB SUB_TYPE 0 SEGMENT SIZE 80,
        REASON VARCHAR(150));

/* Table: BONUS, Owner: PIMPUK */
CREATE TABLE BONUS (COMPANY_ID VARCHAR(5) NOT NULL,
        NUMBER_OF_INSTALMENT DOUBLE PRECISION NOT NULL,
        MONTH SMALLINT,
        BONUS_RATE DOUBLE PRECISION);

/* Table: BONUSHIS, Owner: PIMPUK */
CREATE TABLE BONUSHIS (EMPLOYEE_ID VARCHAR(5) NOT NULL,
        COMPANY_ID VARCHAR(5) NOT NULL,
        BONUS_CHANGED_DATE DATE NOT NULL,
        BONUS_AMOUNT DOUBLE PRECISION);
```

```
/* Table: CHOLIDAY, Owner: PIMPUK */
CREATE TABLE CHOLIDAY (HOLIDAY_DATE DATE NOT NULL,
        COMPANY_ID VARCHAR(5) NOT NULL);

/* Table: COMMON_I, Owner: PIMPUK */
CREATE TABLE COMMON_I (EMPLOYEE_ID VARCHAR(5) NOT NULL,
        COMPANY_ID VARCHAR(5) NOT NULL,
        RECEIVED_MONTH SMALLINT NOT NULL,
        RECEIVED_YEAR VARCHAR(4) NOT NULL,
        TOTAL_INCOME DOUBLE PRECISION,
        PROVIDEND_FUND DOUBLE PRECISION,
        SOCIAL_SECURITY DOUBLE PRECISION,
        TAX DOUBLE PRECISION,
        HIRING_TYPE_CODE VARCHAR(5) NOT NULL);

/* Table: COMPANY, Owner: PIMPUK */
CREATE TABLE COMPANY (COMPANY_ID VARCHAR(5) NOT NULL,
        PERCENT_FOR_PROVIDEND DOUBLE PRECISION,
        PERCENT_FOR_SECURITY DOUBLE PRECISION,
        COMPANY_NAME VARCHAR(30),
CONSTRAINT COMPANY_PK PRIMARY KEY (COMPANY_ID));

/* Table: DAILY_I, Owner: PIMPUK */
CREATE TABLE DAILY_I (EMPLOYEE_ID VARCHAR(5) NOT NULL,
        COMPANY_ID VARCHAR(5),
        RECEIVED_MONTH SMALLINT NOT NULL,
        WORKDAY_WORK_INCOME DOUBLE PRECISION,
        HOLIDAY_WORK_INCOME DOUBLE PRECISION,
        WORKDAY_OT_INCOME DOUBLE PRECISION,
        HOLIDAY_OT_INCOME DOUBLE PRECISION,
        RECEIVED_YEAR VARCHAR(4) NOT NULL);
```

```
/* Table: DEDUCT, Owner: PIMPUK */
CREATE TABLE DEDUCT (EMPLOYEE_ID VARCHAR(5) NOT NULL,
        COMPANY_ID VARCHAR(5) NOT NULL,
        RECEIVED_YEAR VARCHAR(4) NOT NULL,
        RECEIVED_MONTH SMALLINT NOT NULL,
        OTHER_DEDUCTION_CODE VARCHAR(5),
        DEDUCTION_AMOUNT DOUBLE PRECISION);

/* Table: DEPT, Owner: PIMPUK */
CREATE TABLE DEPT (DEPARTMENT_ID VARCHAR(5) NOT NULL,
        COMPANY_ID VARCHAR(5) NOT NULL,
        DEPARTMENT_NAME VARCHAR(30),
CONSTRAINT DEPT_PK PRIMARY KEY (COMPANY_ID, DEPARTMENT_ID));
/* Table: DIV, Owner: PIMPUK */
CREATE TABLE DIV (DIVISION_ID VARCHAR(5) NOT NULL,
        DEPARTMENT_ID VARCHAR(5) NOT NULL,
        COMPANY_ID VARCHAR(5) NOT NULL,
        DIVISION_NAME VARCHAR(30),
CONSTRAINT DIV_PK PRIMARY KEY (COMPANY_ID, DEPARTMENT_ID,
DIVISION_ID));

/* Table: EMPLOYEE, Owner: PIMPUK */
CREATE TABLE EMPLOYEE (EMPLOYEE_ID VARCHAR(5) NOT NULL,
        COMPANY_ID VARCHAR(5) NOT NULL,
        APPLICANT_ID VARCHAR(5),
        HIRING_TYPE_CODE VARCHAR(5),
        EMPLOYEE_STATUS_CODE VARCHAR(5),
        PROVIDEND_STATUS VARCHAR(5),
        SOCIAL_SECURITY_STATUS VARCHAR(1),
        WORK_DATE DATE,
        MARRIAGE_STATUS VARCHAR(5),
        EMPLOYEE_TYPE_CODE VARCHAR(5),
        STAFF_DATE DATE,
```

```
    RESIGN_DATE DATE,
CONSTRAINT EMPLOYEE_PK PRIMARY KEY (EMPLOYEE_ID, COMPANY_ID));

/* Table: ES, Owner: PIMPUK */
CREATE TABLE ES (EMPLOYEE_STATUS_CODE VARCHAR(5) NOT NULL,
    COMPANY_ID VARCHAR(5) NOT NULL,
    ES_DESCRIPTION VARCHAR(40));

/* Table: ET, Owner: PIMPUK */
CREATE TABLE ET (COMPANY_ID VARCHAR(5) NOT NULL,
    EMPLOYEE_TYPE_CODE VARCHAR(5) NOT NULL,
    ET_DESCRIPTION VARCHAR(40));

/* Table: EVALUATE, Owner: PIMPUK */
CREATE TABLE EVALUATE (EMPLOYEE_ID VARCHAR(5) NOT NULL,
    COMPANY_ID VARCHAR(5) NOT NULL,
    FOR_YEAR VARCHAR(4) NOT NULL,
    QUARTER_NUMBER VARCHAR(1) NOT NULL,
    POINTS DOUBLE PRECISION);

/* Table: EXP1, Owner: PIMPUK */
CREATE TABLE EXP1 (APPLICANT_ID VARCHAR(5) NOT NULL,
    COMPANY_ID VARCHAR(5) NOT NULL,
    ENTER_DATE DATE NOT NULL,
    EX_COMPANY VARCHAR(15),
    EX_POSITION VARCHAR(20));

/* Table: GENERAL_TAX, Owner: PIMPUK */
CREATE TABLE GENERAL_TAX (TAX_YEAR VARCHAR(4) NOT NULL,
    DISCOUNT_RATE FLOAT,
    MAXIMUM_DISCOUNT FLOAT,
    PRIVATE_EXPENSE FLOAT);
```

```
/* Table: GTEE, Owner: PIMPUK */
CREATE TABLE GTEE (EMPLOYEE_ID VARCHAR(5) NOT NULL,
        COMPANY_ID VARCHAR(5) NOT NULL,
        GUARUNTIST_NAME VARCHAR(20) NOT NULL,
        GUARUNTIST_SURNAME VARCHAR(35) NOT NULL,
        GUARUNTEE_AMOUNT DOUBLE PRECISION);

/* Table: GTIST, Owner: PIMPUK */
CREATE TABLE GTIST (GUARUNTIST_NAME VARCHAR(20) NOT NULL,
        GUARUNTIST_SURNAME VARCHAR(35) NOT NULL,
        GUARUNTIST_ADDRESS VARCHAR(100),
        GUARUNTIST_INCOME DOUBLE PRECISION);

/* Table: HT, Owner: PIMPUK */
CREATE TABLE HT (HIRING_TYPE_CODE VARCHAR(5) NOT NULL,
        HT_DESCRIPTION VARCHAR(30));

/* Table: INCOME, Owner: PIMPUK */
CREATE TABLE INCOME (EMPLOYEE_ID VARCHAR(5) NOT NULL,
        COMPANY_ID VARCHAR(5) NOT NULL,
        RECEIVED_MONTH SMALLINT NOT NULL,
        RECEIVED_YEAR VARCHAR(4) NOT NULL,
        OTHER_INCOME_CODE VARCHAR(5),
        INCOME_AMOUNT DOUBLE PRECISION);

/* Table: ITEM, Owner: PIMPUK */
CREATE TABLE ITEM (ITEM_CODE VARCHAR(5) NOT NULL,
        ITEM_DESCRIPTION VARCHAR(40),
        COMPANY_ID VARCHAR(5) NOT NULL);
```

```
/* Table: MINPIECE, Owner: PIMPUK */
CREATE TABLE MINPIECE (COMPANY_ID VARCHAR(5) NOT NULL,
        HIRING_TYPE_CODE VARCHAR(5) NOT NULL,
        ITEM_CODE VARCHAR(5) NOT NULL,
        MINIMUM_PIECE DOUBLE PRECISION);

/* Table: MONTH_I, Owner: PIMPUK */
CREATE TABLE MONTH_I (EMPLOYEE_ID VARCHAR(5) NOT NULL,
        COMPANY_ID VARCHAR(5) NOT NULL,
        RECEIVED_MONTH SMALLINT NOT NULL,
        RECEIVED_YEAR VARCHAR(4) NOT NULL,
        HOLIDAY_WORK_INCOME DOUBLE PRECISION,
        HOLIDAY_OT_INCOME DOUBLE PRECISION,
        WORKDAY_OT_INCOME DOUBLE PRECISION);

/* Table: OT, Owner: PIMPUK */
CREATE TABLE OT (EMPLOYEE_ID VARCHAR(5) NOT NULL,
        COMPANY_ID VARCHAR(5) NOT NULL,
        OT_DATE DATE NOT NULL);

/* Table: OTHERABS, Owner: PIMPUK */
CREATE TABLE OTHERABS (COMPANY_ID VARCHAR(5) NOT NULL,
        ABSENT_TYPE_CODE VARCHAR(5) NOT NULL,
        SALARY_INDICATOR DOUBLE PRECISION,
        ABSENTDAY1 SMALLINT,
        ABSENTDAY2 SMALLINT);

/* Table: OTHERDED, Owner: PIMPUK */
CREATE TABLE OTHERDED (COMPANY_ID VARCHAR(5) NOT NULL,
        OTHER_DEDUCTION_CODE VARCHAR(5) NOT NULL,
        DEDUCTION_DESCRIPTION VARCHAR(40));
```

```
/* Table: OTHERINC, Owner: PIMPUK */
CREATE TABLE OTHERINC (COMPANY_ID VARCHAR(5) NOT NULL,
        OTHER_INCOME_CODE VARCHAR(5) NOT NULL,
        INCOME_DESCRIPTION VARCHAR(40));

/* Table: OT_RATE, Owner: PIMPUK */
CREATE TABLE OT_RATE (COMPANY_ID VARCHAR(5) NOT NULL,
        HIRING_TYPE_CODE VARCHAR(5) NOT NULL,
        RATE_FOR_WORKDAY_OT FLOAT,
        RATE_FOR_HOLIDAY_WORK FLOAT,
        RATE_FOR_HOLIDAY_OT FLOAT);

/* Table: OVERALL_TAX, Owner: PIMPUK */
CREATE TABLE OVERALL_TAX (TAX_YEAR VARCHAR(4) NOT NULL,
        TAX_RATE DOUBLE PRECISION NOT NULL);

/* Table: PD_I, Owner: PIMPUK */
CREATE TABLE PD_I (EMPLOYEE_ID VARCHAR(5) NOT NULL,
        COMPANY_ID VARCHAR(5) NOT NULL,
        RECEIVED_MONTH SMALLINT NOT NULL,
        RECEIVED_YEAR VARCHAR(4) NOT NULL,
        WORKDAY_WORK_INCOME DOUBLE PRECISION,
        HOLIDAY_WORK_INCOME DOUBLE PRECISION,
        WORKDAY_OT_INCOME DOUBLE PRECISION,
        HOLIDAY_OT_INCOME DOUBLE PRECISION,
        PIECE_WORKING_INCOME FLOAT);

/* Table: PIECERAT, Owner: PIMPUK */
CREATE TABLE PIECERAT (COMPANY_ID VARCHAR(5) NOT NULL,
        HIRING_TYPE_CODE VARCHAR(5) NOT NULL,
        ITEM_CODE VARCHAR(5) NOT NULL,
        NUMBER_OF_PIECE DOUBLE PRECISION NOT NULL,
        PIECE_RATE DOUBLE PRECISION);
```

```
/* Table: PIECE_I, Owner: PIMPUK */
CREATE TABLE PIECE_I (EMPLOYEE_ID VARCHAR(5) NOT NULL,
      COMPANY_ID VARCHAR(5) NOT NULL,
      RECEIVED_MONTH SMALLINT NOT NULL,
      RECEIVED_YEAR VARCHAR(4) NOT NULL,
      PIECE_WORKING_INCOME DOUBLE PRECISION);

/* Table: PIECE_W, Owner: PIMPUK */
CREATE TABLE PIECE_W (EMPLOYEE_ID VARCHAR(5) NOT NULL,
      COMPANY_ID VARCHAR(5) NOT NULL,
      WORK_DATE DATE NOT NULL,
      ITEM_CODE VARCHAR(5) NOT NULL,
      NUMBER_OF_PIECE SMALLINT);

/* Table: POS, Owner: PIMPUK */
CREATE TABLE POS (POSITION_ID VARCHAR(5) NOT NULL,
      POSITION_NAME VARCHAR(15),
      COMPANY_ID VARCHAR(5),
      DEPARTMENT_ID VARCHAR(5),
      DIVISION_ID VARCHAR(5),
CONSTRAINT POS_PK PRIMARY KEY (POSITION_ID));

/* Table: POSHIST, Owner: PIMPUK */
CREATE TABLE POSHIST (EMPLOYEE_ID VARCHAR(5) NOT NULL,
      COMPANY_ID VARCHAR(5) NOT NULL,
      PROMOTED_DATE DATE NOT NULL,
      POSITION_ID VARCHAR(5),
CONSTRAINT POSHIST_PK PRIMARY KEY (EMPLOYEE_ID, COMPANY_ID,
PROMOTED_DATE));

/* Table: SALARY, Owner: PIMPUK */
CREATE TABLE SALARY (EMPLOYEE_ID VARCHAR(5) NOT NULL,
```

```
    COMPANY_ID VARCHAR(5) NOT NULL,
    SALARY_CHANGED_DATE DATE NOT NULL,
    SALARY_AMOUNT DOUBLE PRECISION);

/* Table: SHIFT, Owner: PIMPUK */
CREATE TABLE SHIFT (COMPANY_ID VARCHAR(5) NOT NULL,
    SHIFT_NUMBER SMALLINT NOT NULL,
    BEGIN_WORK_TIME DATE,
    END_WORK_TIME DATE,
    BEGIN_OT_TIME DATE,
    BEGIN_LUNCH_TIME DATE,
    END_LUNCH_TIME DATE);

/* Table: TAM, Owner: PIMPUK */
CREATE TABLE TAM (EMPLOYEE_ID VARCHAR(5) NOT NULL,
    COMPANY_ID VARCHAR(5) NOT NULL,
    TAM_DATETIME DATE NOT NULL,
    TAM_FUNCTION VARCHAR(7) NOT NULL,
    PROCESSED VARCHAR(5));

/* Table: TAX, Owner: PIMPUK */
CREATE TABLE TAX (TAX_YEAR VARCHAR(4) NOT NULL,
    TAX_RATE INTEGER NOT NULL,
    TO_INCOME FLOAT,
    FROM_INCOME FLOAT NOT NULL);

/* Table: TAX_RATE, Owner: PIMPUK */
CREATE TABLE TAX_RATE (TAX_YEAR VARCHAR(4) NOT NULL,
    TAX_RATE INTEGER NOT NULL,
    TO_INCOME FLOAT,
    FROM_INCOME FLOAT NOT NULL);
```

```
/* Table: TEMP, Owner: SYSDBA */
CREATE TABLE TEMP (EMPLOYEE_ID VARCHAR(5),
        COMPANY_ID VARCHAR(5));

/* Table: TEST, Owner: PIMPUK */
CREATE TABLE TEST (NAME VARCHAR(20),
        SURNAME VARCHAR(30));

/* Table: TEST1, Owner: PIMPUK */
CREATE TABLE TEST1 (TEST DATE);

/* Table: TRAINED, Owner: PIMPUK */
CREATE TABLE TRAINED (APPLICANT_ID VARCHAR(5) NOT NULL,
        COMPANY_ID VARCHAR(5) NOT NULL,
        SUBJECT VARCHAR(40) NOT NULL,
        RUNNING_NUMBER SMALLINT NOT NULL);

/* Table: TRAINING, Owner: PIMPUK */
CREATE TABLE TRAINING (SUBJECT VARCHAR(40) NOT NULL,
        RUNNING_NUMBER SMALLINT NOT NULL,
        LECTURE_BEGIN_DATE DATE,
        LECTURE_DAYS DOUBLE PRECISION,
        PLACE VARCHAR(60),
        FEE DOUBLE PRECISION,
        LECTURER VARCHAR(60),
        ORGANIZER VARCHAR(60));

/* Table: WAGE, Owner: PIMPUK */
CREATE TABLE WAGE (EMPLOYEE_ID VARCHAR(5) NOT NULL,
        COMPANY_ID VARCHAR(5) NOT NULL,
        WAGE_CHANGED_DATE DATE NOT NULL,
        WAGE_AMOUNT DOUBLE PRECISION);
```

```
/* Table: WEEK_HOLIDAY, Owner: PIMPUK */
CREATE TABLE WEEK_HOLIDAY (COMPANY_ID VARCHAR(5) NOT NULL,
        EMPLOYEE_TYPE_CODE VARCHAR(5) NOT NULL,
        MONDAY_IS_HOLIDAY VARCHAR(5),
        TUESDAY_IS_HOLIDAY VARCHAR(5),
        WEDNESDAY_IS_HOLIDAY VARCHAR(5),
        THURSDAY_IS_HOLIDAY VARCHAR(5),
        FRIDAY_IS_HOLIDAY VARCHAR(5),
        SATURDAY_IS_HOLIDAY VARCHAR(5),
        SUNDAY_IS_HOLIDAY VARCHAR(5));

/* Table: WORKREC, Owner: PIMPUK */
CREATE TABLE WORKREC (EMPLOYEE_ID VARCHAR(5) NOT NULL,
        COMPANY_ID VARCHAR(5) NOT NULL,
        WORK_DATE DATE NOT NULL,
        STATUS_CODE VARCHAR(3),
        WORK_HOUR DOUBLE PRECISION,
        OT_HOUR DOUBLE PRECISION);

/* Table: WORKTIME, Owner: PIMPUK */
CREATE TABLE WORKTIME (COMPANY_ID VARCHAR(5) NOT NULL,
        HIRING_TYPE_CODE VARCHAR(5) NOT NULL,
        SHIFT_NUMBER VARCHAR(2) NOT NULL,
        BEGIN_WORKDAY_WORK_TIME DATE,
        END_WORKDAY_WORK_TIME DATE,
        BEGIN_WORKDAY_OT_TIME DATE,
        BEGIN_LUNCH_TIME DATE,
        END_LUNCH_TIME DATE,
        BEGIN_HOLIDAY_WORK_TIME DATE,
        END_HOLIDAY_WORK_TIME DATE,
        BEGIN_HOLIDAY_OT_TIME DATE);
```

```
/* Table: WSTATUS, Owner: PIMPUK */
CREATE TABLE WSTATUS (STATUS_CODE VARCHAR(5) NOT NULL,
    STATUS_DESCRIPTION VARCHAR(40));

/*  Index definitions for all user tables */
CREATE INDEX TEST1_TEST ON TEST1(TEST);
ALTER TABLE POS ADD FOREIGN KEY (COMPANY_ID) REFERENCES COMPANY
(COMPANY_ID);
ALTER TABLE POS ADD FOREIGN KEY (COMPANY_ID, DEPARTMENT_ID)
REFERENCES DEPT(COMPANY_ID, DEPARTMENT_ID);
ALTER TABLE POS ADD FOREIGN KEY (COMPANY_ID, DEPARTMENT_ID,
DIVISION_ID) REFERENCES DIV(COMPANY_ID, DEPARTMENT_ID, DIVISION_ID);
ALTER TABLE POSHIST ADD FOREIGN KEY (POSITION_ID) REFERENCES POS
(POSITION_ID);
ALTER TABLE DEPT ADD FOREIGN KEY (COMPANY_ID) REFERENCES COMPANY
(COMPANY_ID);
ALTER TABLE DIV ADD FOREIGN KEY (COMPANY_ID, DEPARTMENT_ID)
REFERENCES DEPT(COMPANY_ID, DEPARTMENT_ID);


SET TERM ^ ;

/* Triggers only will work for SQL triggers */
CREATE TRIGGER DEPT_DELETE FOR DEPT
ACTIVE BEFORE DELETE POSITION 0
as
begin
 delete from div t1 where t1.Department_id = old.Department_id and t1.Company_id =
old.Company_id;
 delete from pos t2 where t2.Department_id = old.Department_id and
t2.Company_id = old.Company_id;
end
 ^
```

```
CREATE TRIGGER DIV_DELETE FOR DIV
ACTIVE BEFORE DELETE POSITION 0
as
begin
 delete from pos t1 where t1.Division_id = old.Division_id and t1.Department_id =
old.Department_id and t1.Company_id = old.Company_id;
end
 ^
COMMIT WORK ^
SET TERM ; ^

/* Grant permissions for this database */
```

ดรรชนี

กิตติกรรมประกาศ

โครงงานนี้คงไม่สามารถสำเร็จลุล่วงลงได้ ถ้าปราศจากบุคคลเหล่านี้ ผู้จัดทำโครงงานขอขอบคุณ

ดร. วรวัฒน์ ลิ้มโภคา ที่คอยเป็นที่ปรึกษา ให้คำแนะนำ และแก้ปัญหาให้ตลอดเวลา

อาจารย์ บุญธีร์ และอาจารย์กฤตวัน เครือตาชู สำหรับคำแนะนำ และความช่วยเหลือ

พี่ ๆ ที่บริษัทเทกซ์ไทล์ เพรสทีจ ที่คอยตอบคำถามให้อยู่เสมอ

พี่ ๆ ที่บริษัทสหอินโฟเทคโนโลยี ที่สละเครื่องให้เราใช้อยู่เสมอ

เพื่อน ๆ ภาควิชาวิศวกรรมคอมพิวเตอร์ที่คอยช่วยเหลือและช่วยคิดโปรเจ็คท์ด้วยในบางครั้ง

และขอขอบคุณทุกคนที่ไม่ได้กล่าวถึง

หนังสืออ้างอิง

- Borland International, "Database Application Developer's Guide",Borland International,1995.
- James A. Senn , "Analysis & Design of Information System",McGRAW-HILL PUBLISHING COMPANY,1989.
- Jeff Duntemann,Jim Mischel and Don Taylor, "DELPHI PROGRAMMING EXPLORER ",The Coriolis Group,1995
- Neil J. Rubenking, "DELPHI PROGRAMMING FOR DUMMIES",IDG Books Worldwide,1995.Blake Watson, "Delphi By Example",QUE Corporation,1995.
- Jon Matcho and group, "Special Edition Using Delphi",QUE Corporation,1995.
- บุญเลิศ เอี่ยมทัศนา , "โปรแกรมด้วย Delphi : เริ่มต้น",ไมโครคอมพิวเตอร์,ฉบับที่ 121,2538,หน้า 233-240.
- สิทธิพงศ์ จันทร์แสงอรุณ,"ไคล์เอ็นต์เซอร์ฟเวอร์ในระบบคอมพิวเตอร์" , ไมโครคอมพิวเตอร์,ฉบับที่ 112,2537,หน้า 227-233
- ยืน ภู่วรวรรณ ,"ไคล์เอ็นต์เซอร์ฟเวอร์ เทคโนโลยีการใช้คอมพิวเตอร์ในองค์กร", ไมโครคอมพิวเตอร์,ฉบับที่ 99,2538,หน้า 217-222.