บทที่ 5

การพัฒนาระบบฐานข้อมูลระบบบุคลากรมหาวิทยาลัย

สภาพแวดล้อมในการพัฒนา

ในการพัฒนาระบบฐานข้อมูลนี้ ได้พัฒนาบนเครื่องพีซี (Personnel Computer) ซึ่งมีซีพียู 80386 ภายใต้ระบบปฏิบัติการดอส รุ่นที่ 5 (DOS V.5.0) โดยใช้ระบบจัดการฐานข้อมูลแบบเชิงสัมพันธ์ของออราเคิล (ORACLE RDBMS V6.0.27.9.2 with transaction processing option and PL/SQL V1.0.28.1.0) และเพื่อให้โปรแกรมอรรถประโยชน์ของออราเคิล และสามารถใช้ภาษาไทยได้ จึงได้ทำการดัดแปลงแฟ้ม .CRT ซึ่งเป็นโปรแกรมที่ใช้ในการแปลงเครื่องปลายทาง (terminal) ให้ใช้ภาษาไทยได้กับเครื่องปลายทางชนิดต่าง ๆ

เครื่องมือที่ใช้ในการพัฒนา

ระบบฐานข้อมูลได้ถูกสร้างขึ้น โดยการใช้โปรแกรมอรรถประโยชน์ต่าง ๆ ของออราเคิล ดังนี้

1. SQL*Plus V3.0.6.5.1 ประโยคที่ใช้แบ่งออกเป็น 4 ประเภท คือ

1.1 ประโยคสอบถาม (Queries Statements) เพื่อเรียกดูข้อมูลที่ต้องการ โดยมีรูปแบบคำสั่งดังนี้

```
SELECT [ ALL | DISTINCT ]  { * | [user.]select-column | expr | literal} [alias]
FROM [user. ]table-name  [t_alias]  [ , [user.]table-name [t_alias]].....
[WHERE clause] [AND clause] [OR clause]
[GROUP BY clause [ HAVING clause]]
[ORDER BY clause [ASC | DESC]]
```

1.2 ประโยคจัดการข้อมูล (Data Manipulation Statements) ใช้เปลี่ยนแปลงข้อมูล มีดังนี้

- การเพิ่มแถวใหม่ของข้อมูลในตาราง มีรูปแบบคำสั่ง ดังนี้

```
INSERT INTO [user.]table-name [(column-name [,column_name.....])]
    VALUES (value [,value ....]) | query_block
```

- การแก้ไขค่าของคอลัมน์ ในแถวที่มีอยู่ มีรูปแบบคำสั่ง ดังนี้

```
UPDATE [user.][table-name
SET  { [column-name = value
      [,column-name = value, ....]] |
      column-name[,column-name]...] = (query_block[, ....])}
      [WHERE  clause]
```

- การลบแถวของข้อมูลจากตาราง มีรูปแบบคำสั่ง ดังนี้

```
DELETE FROM [user.]table-name [alias]
    [WHERE  clause]
```

นอกจากนี้ คำสั่ง COMMIT [WORK] และ ROLLBACK [WORK] ก็เป็นประโยคจัดการข้อมูลที่ใช้ด้วยเหมือนกัน

1.3 ประโยคจำกัดความข้อมูล (Data Definition Statements) ประกอบด้วยคำสั่งในการสร้าง คำสั่งที่ใช้ ได้แก่

- คำสั่งในการสร้างตาราง มีรูปแบบคำสั่ง ดังนี้

```
CREATE TABLE [user.]table-name
        (column-name  data-type | table-constraint]
        [column-name  data-type | table-constraint]....]
```

- คำสั่งในการสร้างวิว มีรูปแบบคำสั่ง ดังนี้

```
CREATE VIEW [user.]view-name[ (alias[, alias]...)]
  AS query-block
  [WITH CHECK  OPTION [CONSTRAINT constraint]]
```

การใส่ WITH CHECK OPTION เป็นวิธีหนึ่งในการตรวจสอบความถูกต้องของข้อมูลในการเพิ่ม หรือแก้ไขข้อมูลผ่านวิวได้ และยังสามารถบังคับให้เป็นไปตามกฎความเป็นบูรณภาพของข้อมูลได้ด้วย

- คำสั่งในการสร้างเลขลำดับ มีรูปแบบคำสั่งดังนี้

```
CREATE SEQUENCE [user.]sequence
  INCREASE BY { 1 | integer }
  [ START WITH integer ]
```

คอลัมน์ในบางตาราง เช่น รหัสผลงานวิชาการ เป็นต้น อาจจะมีการกำหนดให้มีค่าเป็นตัวเลขเรียงลำดับก็ได้

- คำสั่งในการสร้างดัชนี มีรูปแบบคำสั่ง ดังนี้

```
CREATE [ UNIQUE ] INDEX index-name
  { ON table-name (column-name [ ASC | DESC]
  [ , (column-name [ASC | DESC]) ...] }
```

คำสั่งนี้มักใช้สร้างดัชนีบนคอลัมน์ที่เป็นคีย์หลักของตารางที่สำคัญ

- คำสั่งในการสร้างคำเหมือน (synonym) มีรูปแบบคำสั่ง ดังนี้

```
CREATE [PUBLIC] SYNONYM [user.]synonym-name
  FOR [user.]table-name | view-name
```

เนื่องจากใช้ตารางซึ่งได้รับอนุญาตมาจากระบบทะเบียนนักศึกษา และระบบอาคารสถานที่ จึงต้องมีการสร้างคำเหมือนเพื่อสะดวกในการอ้างถึง

- คำสั่งในการแก้ไขโครงสร้าง ของตาราง มีรูปแบบคำสั่ง ดังนี้

```
ALTER TABLE [user.]table-name
  [ADD ({column-name  data-type | table-constraint}
     [ , ({column-name  data-type | tablr-constraint},....]]
  [MODIFY (column-name data-type
        [ ,column-name  data-type,....]]
```

- คำสั่งในการ drop ตาราง วิว หรือดัชนี มีรูปแบบคำสั่ง ดังนี้

```
DROP [ INDEX  [user.]index-name] |
     [ TABLE [user.]table-name] |
     [ VIEW [user.]view-name]
```

1.4 ประโยคควบคุมข้อมูล (Data Control Statements)

- ควบคุมการเข้าถึงฐานข้อมูล (Database Level Security) ได้แก่ คำสั่งในการอนุญาต มีรูปแบบคำสั่ง ดังนี้

```
GRANT  { CONNECT | RESOURCE | DBA }
  TO  userid1 [ , userid2, .... ]
  IDENTIFIED BY password1 [ , password2 , ....]
```

ซึ่งการกำหนดสิทธิให้ CONNECT หมายถึงยอมให้มีการเข้าถึง และอ่านตาราง รวมทั้งสามารถใช้คำสั่งดีเอ็มแอลในการทำงานกับตารางที่ตนมีสิทธิ และยอมให้สามารถสร้างวิว และคำเหมือนได้ แต่ไม่สามารถสร้างตาราง คลัสเตอร์ และดัชนี แต่ถ้าได้รับสิทธิให้ใช้เนื้อที่เก็บข้อมูล (RESOURCE) จะสามารถสร้างตาราง ดัชนี และคลัสเตอร์ รวมทั้งอนุญาตผู้อื่นในการใช้ตารางของตน และสามารถตรวจสอบผู้ปฏิบัติงานนั้นได้ (auditability) สำหรับการได้รับสิทธิเป็นผู้บริหารฐานข้อมูล (DBA) นั้นจะสามารถใช้สิทธิในการเป็นผู้บริหารฐานข้อมูล

ส่วนคำสั่งถอนการอนุญาต มีรูปแบบคำสั่ง ดังนี้

```
REVOKE { CONNECT | RESOURCE | DBA }
  FROM userid
```

- ควบคุมการเข้าถึงข้อมูลในฐานข้อมูล (Table Level Security) ได้แก่ คำสั่งในการอนุญาต มีรูปแบบคำสั่ง ดังนี้

```
GRANT { privilege | ALL }
  ON { table-name | view-name }
  TO { userid | PUBLIC [ , user ] ... }
  [ WITH GRANT OPTION ]
```

privilege := SELECT, INSERT, DELETE, UPDATE[(column-name)], ALTER, INDEX

โดยที่ WITH GRANT OPTION เป็นการยอมให้ผู้อื่นเห็น แก้ไข และใช้ข้อมูลในตาราง รวมทั้งสามารถส่งผ่านสิทธิให้ผู้อื่นต่อไปได้อีก

ส่วนคำสั่งถอนการอนุญาต มีรูปแบบคำสั่ง ดังนี้

```
REVOKE { privilege | ALL }
  ON { table-name | view-name }
  FROM { userid | PUBLIC }
```

ในที่นี้มีการอนุญาตให้ระบบอื่น คือ ระบบทะเบียนนักศึกษา ระบบอาคารสถานที่ และระบบการเงิน ใช้ตารางร่วมกับระบบบุคลากรด้วย ตารางร่วมได้แก่ ภาค คณะ บุคลากร ระดับการศึกษา เชื้อชาติ สัญชาติ ศาสนา ประเทศ อาชีพ ตำแหน่ง ชื่อตำแหน่ง และ ระดับตำแหน่ง นอกจากนี้ระบบบุคลากรได้รับอนุญาตในการใช้ตารางของระบบทะเบียนนักศึกษา คือ ตารางหลักสูตร และใช้ตารางของระบบอาคารสถานที่ คือ ตารางสถานที่ทำงาน (ห้อง)

2. SQL*Forms V3.0.1.6 เป็นเครื่องมือสำหรับพัฒนาระบบงานประยุกต์ ซึ่งมีรูปแบบในลักษณะเชิงโต้ตอบแสดงผลทางจอภาพ โดยสามารถออกแบบฟอร์มที่จะเพิ่ม ลบ หรือแก้ไข และสอบถามข้อมูล ในการทำงานนั้นจะประกอบด้วยฟอร์มหลาย ๆ ฟอร์ม แต่ละฟอร์มจะถูกออกแบบสำหรับแต่ละฟังก์ชันการทำงาน ซึ่งแสดงให้เห็นข้อมูลมากกว่า 1 หน้าจอก็ได้ ฟอร์มหนึ่ง ๆ จะประกอบด้วย "บล็อก" หรือเป็นกลุ่มข้อมูล หรือคอลัมน์ซึ่งได้จากตารางที่เก็บในฐานข้อมูล หรืออาจมาจากวิวที่สร้างไว้ โดยปกติข้อมูลจาก 1 ตารางจะเป็น 1 บล็อก ซึ่งการทำการใด ๆ กับตารางจะถูกจัดการโดยคำสั่งเอสคิวแอลที่ถูกสร้างขึ้นโดยอัตโนมัติ และคำสั่งเอสคิวแอลนี้ยังสามารถถูกกำหนดเพิ่มเติมเองในการทำงานได้ โดยจะระบุไว้ในทริกเกอร์ (Trigger) นอกจากนี้ยังมีบล็อกที่เป็นบล็อกในการควบคุม (Control Block) ซึ่งมีคอลัมน์ที่ไม่ได้สัมพันธ์กับตารางใดในฐานข้อมูล เราสามารถกำหนดคุณลักษณะของเขตข้อมูล หรือคอลัมน์ต่าง ๆ ในการทำงาน และมีการตรวจสอบความถูกต้องของข้อมูลในการป้อนข้อมูลโดยใช้ฟอร์ม

บล็อกที่ออกแบบไว้ในที่นี้ มีแบบต่าง ๆ โดยสรุป ดังนี้

- แสดงผล และดูแลรักษาข้อมูลของตาราง 1 ตาราง ใน 1 บล็อก ซึ่งมีการตรวจสอบความถูกต้อง รวมทั้งข้อจำกัดต่าง ๆ ของข้อมูลด้วย
- แสดงผลของอีกตารางเป็นบล็อกที่ 2 (ตารางลูก) ซึ่งขึ้นกับบล็อกแรก (ตารางแม่) กล่าวคือ เป็นลักษณะ Mater/Detail
- แสดงตารางแม่ในบล็อกเพื่อใช้ในการอ้างอิง หรือดูค่าที่เป็นไปได้สำหรับฟอร์เรนจ์คีย์ ซึ่งเป็นคอลัมน์ในตารางลูก

นอกจากนี้ยังใช้ SQL*Forms ในการบังคับรูปแบบต่าง ๆ ของแอตตริบิวต์ ได้แก่

- ไม่ยอมให้แก้ไขคีย์หลัก
- ใช้ทริกเกอร์ในการป้องกันการลบแถวข้อมูลของตาราง ในกรณีที่คีย์หลักของตารางนั้น (ตารางแม่) ปรากฏตรงกับฟอร์เรนจ์คีย์ซึ่งยังมีอยู่ในอีกตารางหนึ่ง (ตารางลูก)
- คอลัมน์ที่เป็นฟอร์เรนจ์คีย์สามารถเรียกดูได้ว่า ค่าที่เป็นได้มีอะไรบ้าง (list of values) และมีการตรวจสอบว่า ค่าที่ใส่ฟอร์เรนจ์คีย์มีอยู่จริงหรือไม่ด้วย

3. SQL*Menu V5.0.11.3.3 เป็นเครื่องมือที่ใช้ในการทำรายการเลือก ให้กับผู้ใช้งานตามสิทธิที่ได้รับ โดยประกอบด้วยคำสั่งต่าง ๆ ดังนี้

- คำสั่งชนิดที่ 1 เป็นการเรียกใช้รายการเลือกย่อยที่สร้างขึ้นมา (Invoke Submenu)
- คำสั่งชนิดที่ 2 เป็นการเรียกใช้คำสั่ง หรือโปรแกรมของระบบปฏิบัติการ (Execute an operation system command)
- คำสั่งชนิดที่ 3 เป็นการเรียกใช้คำสั่ง หรือโปรแกรมของระบบปฏิบัติการ แต่ให้หยุดรอการตอบรับจากผู้ใช้ก่อน จึงจะกลับเข้ามาทำงานที่รายการเลือกต่อ (Execute an operating system command, then puase for the operator's response)
- คำสั่งชนิดที่ 4 เป็นการเรียกใช้ฟอร์ม (Invoke SQL*Forms [Run Form])
- คำสั่งชนิดที่ 5 เป็นการเรียกใช้เอสคิวแอล (Invoke SQL*Plus)

- คำสั่งชนิดที่ 6 เป็นการเรียกใช้แมโครของรายการเลือกเอง(Execute a SQL*Menu macro command)

- คำสั่งชนิดที่ 7 เป็นการเรียกใช้ PL/SQL (Execute a PL/SQL command)

นอกจากนี้ในการทำรายการเลือกต้องกำหนดว่า จะให้รายการเลือกใดกับผู้ใช้กลุ่ม (user group) ใด มิฉะนั้นรายการเลือกจะไม่สามารถทราบได้ว่า รายการเลือกที่สร้างขึ้นมานี้ สร้างขึ้นมาเพื่อให้ใครใช้งาน (Grant Role Access)

4. EXP (Export Data) ใช้ในการสำรองข้อมูลไว้ ซึ่งสามารถสำรองข้อมูลได้เฉพาะตาราง หรือ จะสำรองข้อมูลของระบบบุคลากรทั้งระบบก็ได้

5. IMP (Import Data) ใช้ในการเรียกคืนข้อมูล ซึ่งการใช้งานนั้นก่อนที่จะเรียกข้อมูลคืน ต้องทำการล้างข้อมูลเก่าในตารางทิ้งเสียก่อน มิฉะนั้นข้อมูลที่เรียกคืนจะเข้าไปเพิ่มในตาราง ทำให้เกิดข้อมูลซ้ำซ้อนขึ้น

ขั้นตอนในการพัฒนา

1. การกำหนดโครงสร้างของตาราง

โครงสร้างของตารางจะมีรูปแบบเช่นเดียวกับ การออกแบบกฎธุรกิจของแอตตริบิวต์ เพียงแต่แอตตริบิวต์ใดที่เป็นวันที่นั้นจะทำการแปลงให้เป็นอักขระ ซึ่งมีความยาวของข้อมูล คือ 8 อักขระ และมีรูปแบบของวันที่ คือ 'DDMMYYYY'

2. การออกแบบจอภาพ และโปรแกรม เพื่อการใช้งาน

ขั้นตอนนี้เป็นการออกแบบจอภาพ โปรแกรม พร้อมทั้งรายการเลือกต่าง ๆ เพื่อให้ผู้ใช้ได้ใช้งานในโปรแกรมต่าง ๆ ตามสิทธิที่ตนได้รับ ซึ่งมีรายละเอียดดังนี้

รายการเลือกหลัก

MAIN

- **PS_ENTRY** (Personnel Entry Data) เป็นรายการเลือกที่ใช้สำหรับบันทึกข้อมูลต่าง ๆ ของระบบบุคลากร ซึ่งรายการเลือกประเภทนี้จะให้ใช้งานได้กับผู้ใช้ที่ได้รับสิทธิเป็น "PS_OPR" หรือ "SP_OPR" เท่านั้น
- **PS_QUERY** (Personnel Query Data) เป็นรายการเลือกที่ใช้สำหรับสอบถามข้อมูลต่าง ๆ ของระบบบุคลากร ซึ่งรายการเลือกประเภทนี้จะให้ใช้งานได้กับผู้ใช้ที่ได้รับสิทธิเป็น "PS_GUS" "PS_USR" "PS_MGR" หรือ "SP_MGR" เท่านั้น
- **UTILITY** เป็นรายการเลือกของเครื่องมือที่จัดเตรียมไว้ให้กับผู้ดูแลระบบทั้งหมด (System/manager) ทั้งนี้เพื่อทำให้การทำงานบางอย่างสะดวกขึ้น
- **EXIT** เลิกการทำงาน

จากนั้น จึงแยกรายการเลือกย่อยของแต่ละรายการเลือกออกมาได้อีก ดังนี้

รายการเลือกย่อยระดับที่ 1

PS_ENTRY

- HISTORY เป็นรายการเลือกสำหรับบันทึกประวัติต่าง ๆ ของบุคลากร
- COMMAND เป็นรายการเลือกสำหรับบันทึกคำสั่งต่าง ๆ
- ABSENT เป็นรายการเลือกสำหรับบันทึกประวัติการลา ของบุคลากร
- SPECIAL JOB เป็นรายการเลือกสำหรับบันทึกผลงานพิเศษต่าง ๆ
- WELFARE เป็นรายการเลือกสำหรับบันทึกการกู้เงินเคหะสงเคราะห์
- REWARD เป็นรายการเลือกสำหรับบันทึกการได้รับพระราชทานเครื่องราชอิสริยาภรณ์
- REF TABS เป็นรายการเลือกสำหรับบันทึกตารางอ้างอิง
- OTHERS เป็นรายการเลือกสำหรับรายการอื่น ๆ
- QUIT เป็นรายการเลือกสำหรับเลิกการทำงาน

PS_QUERY

- HISTORY เป็นรายการเลือกสำหรับสอบถามประวัติส่วนตัว ของบุคลากร
- COMMAND เป็นรายการเลือกสำหรับสอบถาม คำสั่งต่าง ๆ
- ABSENT เป็นรายการเลือกสำหรับสอบถาม การลาประจำปี
- SPECIAL JOB เป็นรายการเลือกสำหรับสอบถาม ผลงานพิเศษต่าง ๆ
- WELFARE เป็นรายการเลือกสำหรับสอบถาม การกู้เงินเคหะสงเคราะห์
- REWARD เป็นรายการเลือกสำหรับสอบถาม การได้รับพระราชทานเครื่องราชอิสริยาภรณ์
- SQL*Plus เป็นรายการเลือกสำหรับสอบถาม ข้อมูลโดยใช้ SQL
- QUIT เป็นรายการเลือกเลิกการทำงาน

UTILITY

- BACK UP เป็นรายการเลือกสำหรับ สำรองข้อมูลของระบบบุคลากร
- RESTORE เป็นรายการเลือกสำหรับ เรียกคืนข้อมูลของระบบบุคลากร
- GRANT USER เป็นรายการเลือกสำหรับ สร้างผู้ใช้งาน ตามสิทธิที่ได้รับ
- REVOKE USER เป็นรายการเลือกสำหรับ ถอนสิทธผู้ใช้งานคืน
- QUIT เป็นรายการเลือก เลิกการทำงาน

นอกจากนั้นยังมีการแยกเป็น <u>รายการเลือกย่อยระดับที่ 2</u> โดยรายการเลือกระดับนี้จะมีการเรียกใช้ฟอร์ม หรือโปรแกรมที่สร้างขึ้นมาเพื่อใช้งานต่อไป ดังนี้

PS_ENTRY

- **HISTORY**
  - PERSON.FRM (ฟอร์ม การบันทึกประวัติส่วนตัวของบุคลากร)
  - ADDRESS.FRM (ฟอร์ม การบันทึกที่อยู่ของบุคลากร)
  - CONTACT.FRM (ฟอร์ม การบันทึกผู้ที่ติดต่อกับบุคลากรได้)
  - PARENT.FRM (ฟอร์ม การบันทึกประวัติบิดา/มารดา ของบุคลากร)
  - SPOUSE.FRM (ฟอร์ม การบันทึกประวัติการสมรส)
  - CHILD.FRM (ฟอร์ม การบันทึกประวัติบุตร/ธิดา)
  - OLD_NAME.FRM (ฟอร์ม การบันทึกการเปลี่ยน ชื่อ - สกุล)
  - U_OLDNAM.FRM (ฟอร์ม การแก้ไขชื่อ - สกุลที่เปลี่ยนไปแล้ว)
- **COMMAND**
  - PER_POS.FRM (ฟอร์ม การบันทึกคำสั่งที่เกี่ยวกับ ตำแหน่งของบุคลากร)
  - PER_SAL..FRM (ฟอร์ม การบันทึกคำสั่งที่เกี่ยวกับการเลื่อนขั้นเงินเดือน)
  - ABS_EDU.FRM (ฟอร์ม การบันทึกคำสั่ง การลาราชการเพื่อศึกษาต่อ หรืออบรม)
  - PER_PUNT.FRM (ฟอร์ม การบันทึกคำสั่งที่เกี่ยวกับการลงโทษทางวินัย)
  - PER_TERM.FRM (ฟอร์ม การบันทึกคำสั่งการออกจากราชการ)

PS_ENTRY (ต่อ)

- **ABSENT**
  - ABS_QUAT.FRM (ฟอร์ม การบันทึกโควต้าของการลาแต่ละประเภทของบุคลากร)
  - ABS_ANUA.FRM (ฟอร์ม การบันทึกการลาประจำปีของบุคลากร)
- **SPECIAL JOB**
  - PROJECT.FRM (ฟอร์ม การบันทึกรายละเอียดของผลงานทางวิชาการ)
  - PER_PROJ.FRM (ฟอร์ม การบันทึกการทำผลงานทางวิชาการ โดยระบุบุคลากร)
  - PER_PRJ1.FRM (ฟอร์ม การบันทึกการทำผลงานทางวิชาการ โดยระบุรหัสผลงาน)
  - RESEARCH.FRM (ฟอร์ม การบันทึกรายละเอียดของผลงานวิจัย)
  - PER_RES.FRM (ฟอร์ม การบันทึกการทำผลงานวิจัย โดยระบุบุคลากร)
  - PER_RES1.FRM (ฟอร์ม การบันทึกการทำผลงานวิจัย โดยระบุรหัสผลงาน)
  - RESPONSE.FRM (ฟอร์ม การบันทึกภาระความรับผิดชอบของบุคลากร)
  - EXT_POST.FRM (ฟอร์ม การบันทึกการดำรงตำแหน่งนอกมหาวิทยาลัยของบุคลากร)
- **REWARD**
  - RCPT_REW.FRM (ฟอร์ม การบันทึกการได้รับพระราชทานเครื่องราชฯ โดยระบุบุคลากร)
  - RCPT_RW1.FRM (ฟอร์มการบันทึกการได้รับพระราชทานเครื่องราชฯโดยระบุเครื่องราชฯ)
- **WELFARE**
  - WELFARE.FRM (ฟอร์ม การบันทึกการกู้เงินเคหะสงเคราะห์ของบุคลากร
- **REF TABS**
  - PROVINCE.FRM (ฟอร์ม การบันทึกตารางอ้างอิงถึงจังหวัด)
  - COUNTRY.FRM (ฟอร์ม การบันทึกตารางอ้างอิงถึงประเทศ)
  - ฯลฯ เป็นต้น
- **OTHERS**
  - PS_USER.FRM (ฟอร์ม การบันทึกตารางผู้ใช้งาน และสิทธิการใช้งาน)
  - BACK UP (เป็นรายการเลือกสำหรับการสำรองข้อมูลของระบบบุคลากร)
    - EXP โดย parfile=PER_HIST.EXP (สำรองข้อมูลของแฟ้มประวัติส่วนตัวต่าง ๆ)
    - EXP โดย parfile=COMMAND.EXP (สำรองข้อมูลของแฟ้มคำสั่งต่าง ๆ)
    - EXP โดย parfile=SPC_JOB.EXP (สำรองข้อมูลของแฟ้มผลงานต่าง ๆ)
    - EXP โดย parfile=ABS_ANU.EXP (สำรองข้อมูลของแฟ้มการลาประจำปี)
    - EXP โดย parfile=WELFARE.EXP (สำรองข้อมูลของการกู้เงินเคหะสงเคราะห์)
    - EXP โดย parfile=POSITION.EXP (สำรองข้อมูลของตำแหน่งต่าง ๆ)
    - EXP โดย parfile=REWARD.EXP (สำรองข้อมูลของการได้รับเครื่องราชฯ)
    - EXP โดย parfile=REF_TABS.EXP (สำรองข้อมูลของตารางอ้างอิงต่าง ๆ)
    - EXP โดย parfile=PS_USER.EXP (สำรองข้อมูลของตารางผู้ใช้งาน)
    - EXP โดย parfile=HISTORY.EXP สำรองข้อมูลของแฟ้มประวัติศาสตร์)
  - CLR_PERS.FRM (ฟอร์มการล้างข้อมูลของบุคลากรที่ไม่ใช้งาน)
  - SQL*Plus (รายการเลือกการสอบถามข้อมูล โดยใช้ SQL)

PS_QUERY

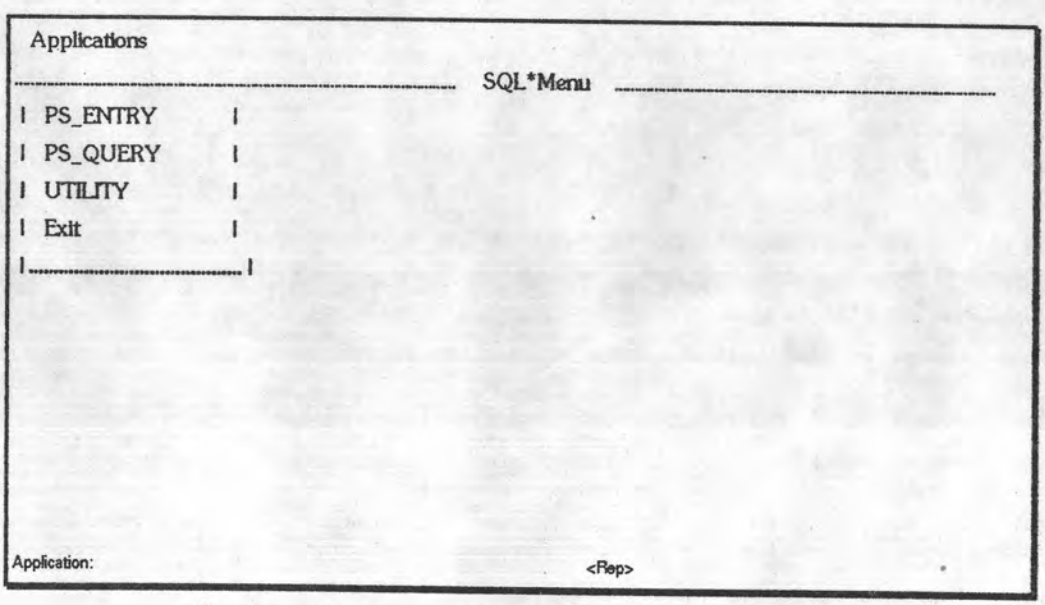
- **HISTORY**
  - Q_PERSON.FRM (ฟอร์ม การสอบถามประวัติส่วนตัวของบุคลากร)
  - Q_ADDRES.FRM (ฟอร์ม การสอบถามที่อยู่ของบุคลากร)
  - Q_CONTAC.FRM (ฟอร์ม การสอบถามผู้ที่สามารถติดต่อกับบุคลากรได้)
  - Q_PARENT.FRM (ฟอร์ม การสอบถามประวัติบิดา/มารดา)
  - Q_SPOUSE.FRM (ฟอร์ม การสอบถามประวัติการสมรส)
  - Q_CHILD.FRM (ฟอร์ม การสอบถามประวัติบุตร/ธิดา)
  - Q_OLDNAM.FRM (ฟอร์ม การสอบถามประวัติการเปลี่ยน ชื่อ - สกุล)
- **COMMAND**
  - Q_PERPOS.FRM (ฟอร์ม การสอบถามคำสั่งที่เกี่ยวกับตำแหน่งของบุคลากร)
  - Q_PERSAL.FRM (ฟอร์ม การสอบถามคำสั่งที่เกี่ยวกับขั้นเงินของบุคลากร)
  - Q_ABSEDU.FRM (ฟอร์ม การสอบถามคำสั่งการลาราชการเพื่อศึกษาต่อ)
  - Q_PUNISH.FRM (ฟอร์ม การสอบถามคำสั่งการได้รับโทษทางวินัยของบุคลากร)
  - Q_TERM.FRM (ฟอร์ม การสอบถามคำสั่งการออกจากราชการ โดยระบุบุคลากร)
  - Q_TERM1.FRM (ฟอร์ม การสอบถามคำสั่งการออกจากราชการโดยระบุเหตุผลการออก)
- **ABSENT**
  - Q_ABSANU.FRM (ฟอร์ม การสอบถามการลาประจำปี โดยระบุบุคลากร)
  - Q_ABSAN1.FRM (ฟอร์ม การสอบถามการลาประจำปี โดยระบุประเภทการลา)
  - Q_ABSQUT.FRM (ฟอร์ม การสอบถามโควต้าของการลาประจำปี โดยระบุบุคลากร)
  - Q_ABSQU1.FRM (ฟอร์ม การสอบถามโควต้าของการลาประจำปี โดยระบุประเภทการลา)
- **SPECIAL JOB**
  - Q_PROJEC.FRM (ฟอร์ม การสอบถามการทำผลงานทางวิชาการ โดยระบุบุคลากร)
  - Q_PRJ1.FRM (ฟอร์ม การสอบถามการทำผลงานทางวิชาการ โดยระบุผลงาน)
  - Q_RESEAR.FRM (ฟอร์ม การสอบถามการทำผลงานวิจัย โดยระบุบุคลากร)
  - Q_RES1.FRM (ฟอร์ม การสอบถามการทำผลงานวิจัย โดยระบุผลงาน)
  - Q_RESPON.FRM (ฟอร์ม การสอบถามภาระความรับผิดชอบ โดยระบุบุคลากร)
  - Q_RESP1.FRM (ฟอร์ม การสอบถามภาระความรับผิดชอบ โดยระบุประเภทภาระ)
  - Q_EXTPOS.FRM (ฟอร์ม การสอบถามการดำรงตำแหน่งนอกมหาวิทยาลัยของบุคลากร)
- **REWARD**
  - Q_REWARD.FRM (ฟอร์ม การสอบถามการได้รับเครื่องราชฯ โดยระบุบุคลากร)
  - Q_REW1.FRM (ฟอร์ม การสอบถามการได้รับเครื่องราชฯ โดยระบุประเภทเครื่องราชฯ)
- **WELFARE**
  - Q_WELFAR.FRM (ฟอร์ม การสอบถามการกู้เงินเคหะสงเคราะห์ โดยระบุบุคลากร)
  - Q_WELF1.FRM (ฟอร์ม การสอบถามการกู้เงินเคหะสงเคราะห์โดยระบุวัตถุประสงค์การกู้)

UTILITY

- **BACK UP** มีรายการเลือกย่อยเช่นเดียวกับการสำรองข้อมูลของ PS_ENTRY
- **RESTORE** เป็นรายการเลือกสำหรับการเรียกคืนข้อมูล
  - IMP โดย parfile=PER_HIST.IMP (เรียกคืนข้อมูลของแฟ้มประวัติส่วนตัวต่าง ๆ)
  - IMP โดย parfile=COMMAND.IMP (เรียกคืนข้อมูลของแฟ้มคำสั่งต่าง ๆ)
  - IMP โดย parfile=SPC_JOB.IMP (เรียกคืนข้อมูลของแฟ้มผลงานต่าง ๆ)
  - IMP โดย parfile=ABS_ANU.IMP (เรียกคืนข้อมูลของแฟ้มการลาประจำปี)
  - IMP โดย parfile=WELFARE.IMP (เรียกคืนข้อมูลของการกู้เงินเคหะสงเคราะห์)
  - IMP โดย parfile=POSITION.IMP (เรียกคืนข้อมูลของตำแหน่งต่าง ๆ)
  - IMP โดย parfile=REWARD.IMP (เรียกคืนข้อมูลของการได้รับเครื่องราชฯ)
  - IMP โดย parfile=REF_TABS.IMP (เรียกคืนข้อมูลของตารางอ้างอิงต่าง ๆ)
  - IMP โดย parfile=PS_USER.IMP (เรียกคืนข้อมูลของตารางผู้ใช้งาน)
  - IMP โดย parfile=HISTORY.IMP (เรียกคืนข้อมูลของแฟ้มประวัติศาสตร์)
- **GRANT USER**
  - PS_GRANA.FRM (ฟอร์ม การสร้างผู้ใช้งานของระบบบุคลากรทั้งมหาวิทยาลัย)
  - PS_GRANF.FRM (ฟอร์ม การสร้างผู้ใช้งานของระบบบุคลากรโดยระบุคณะ)
  - PS_GRANU.FRM (ฟอร์ม การสร้างผู้ใช้งานของระบบบุคลากรโดยระบุ USER ID)
- **REVOKE USER**
  - PS_REVOK.FRM (ฟอร์ม การถอนสิทธิผู้ใช้งานของระบบบุคลากร)

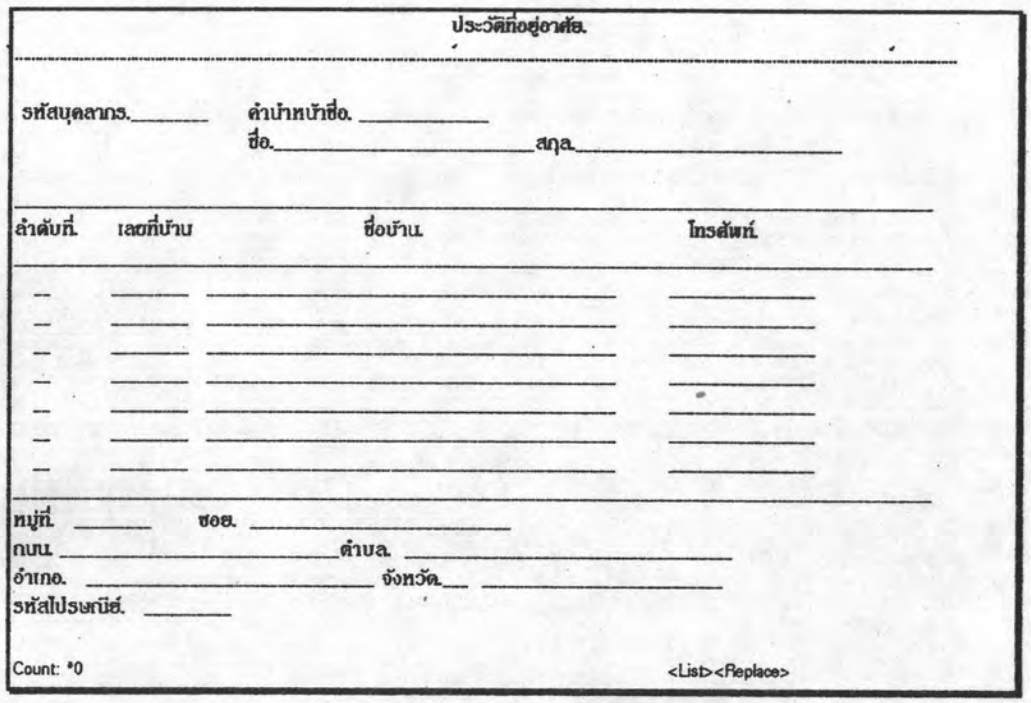
ซึ่งในที่นี้ได้ทำการออกแบบจอภาพที่ใช้งานไว้ 2 ลักษณะ คือ

1. จอภาพที่ใช้สำหรับเลือกรายการเลือกต่าง ๆ ได้ออกแบบให้เป็นจอภาพที่มีลักษณะเป็นแบบรายการเลือกแบบดึงลง (Pull down menu) ตัวอย่างเช่น



รูปที่ 5.1 แสดงตัวอย่างจอภาพของรายการเลือกหลักของระบบงานบุคลากร

2. จอภาพที่ใช้สำหรับการนำข้อมูลเข้าฐานข้อมูล หรือการสอบถามข้อมูลต่าง ๆ ของระบบบุคลากร ได้ออกแบบให้เป็นจอภาพที่มีลักษณะการทำงานเป็นแบบใช้แป้นกำหนดหน้าที่ และจะมีเขตข้อมูลต่าง ๆ รองรับข้อมูลของบุคลากรเพื่อนำไปเก็บไว้ในฐานข้อมูล หรือนำจากฐานข้อมูลมาแสดงผล ตัวอย่างเช่น



รูปที่ 5.2 แสดงตัวอย่างจอภาพของฟอร์มการบันทึกประวัติที่อยู่อาศัยของบุคลากร

การพัฒนาโปรแกรม

ในการพัฒนาโปรแกรมนั้นเริ่มจากการสร้างตารางต่างๆ ไว้ใช้งานในรหัสผู้ใช้งานของออราเคิล ที่ชื่อว่า "SU" ซึ่งมีรหัสผ่าน (password) ว่า "SU" เช่นเดียวกัน ส่วนคำสั่งที่ใช้ในการสร้างตารางต่าง ๆ นั้นได้เก็บไว้ในแฟ้มที่มีตระกูลว่า ".str" และสร้างตารางโดยใช้ SQL*Plus

จากนั้นจึงสร้างฟอร์มเพื่อใช้งานในการบันทึกข้อมูลต่าง ๆ ของระบบ โดยใช้ SQL*Forms ซึ่งจะทำให้ได้แฟ้ม 2 ตระกูลด้วยกันคือ ".frm" และ ".inp" ซึ่ง ".frm" ก็คือ แฟ้มที่จะนำไปให้ผู้ใช้งานของระบบบุคลากรได้ใช้งาน ส่วน ".inp" เป็นแฟ้มที่เก็บคำสั่งต่าง ๆ ของฟอร์มที่สามารถนำมาแก้ไขได้ แล้วจึงทำการประมวลผล (generate) ใหม่ ก็จะทำให้ได้ ".frm" ตัวใหม่ได้ ฉะนั้นแฟ้มตระกูลนี้ไม่ควรจะนำไปให้ผู้ใช้งานของระบบใช้ แต่ควรจะเก็บไว้เฉพาะที่ผู้ที่มีหน้าที่ในการพัฒนาโปรแกรมเท่านั้น

ส่วนหลักการของการสร้างฟอร์มเพื่อใช้ในการบันทึกนั้น ได้ออกแบบให้มีบล็อกต่าง ๆ คือ มีบล็อกแรกชื่อว่า "KEY" เป็นบล็อกที่คอยรับข้อมูลที่จะเป็นกุญแจนำไปสู่การเพิ่ม หรือการแก้ไขข้อมูล

เมื่อได้ข้อมูลจากบล็อกแรกแล้ว จะทำการตรวจสอบตามตารางของบล็อกที่ 2 ถ้าพบข้อมูลดังกล่าว ก็จะนำขึ้นมาให้แก้ไข แต่ถ้าไม่พบก็จะเป็นการให้ผู้ใช้เพิ่มเติมข้อมูลเข้าไป โดยคำสั่งที่จะทำงานดังกล่าวจะอยู่ในทริกเกอร์ของ "KEY-NXTFLD" ของเขตข้อมูลสุดท้ายของบล็อกแรก นอกจากนั้นในขณะที่ใส่ข้อมูลของตารางตามบล็อกที่ 2 โปรแกรมก็จะมีการตรวจสอบค่าของข้อมูลที่เป็นรหัสอ้างอิงว่า ถูกต้องตามที่พบอยู่ในตารางอ้างอิงหรือไม่ โดยคำสั่งจะอยู่ในทริกเกอร์ของ "POST-FILED" ของเขตข้อมูลนั้นๆ หลังจากใส่ข้อมูลเรียบร้อยแล้ว ก็ต้องจัดเก็บข้อมูลโดยคำสั่งการจัดเก็บข้อมูลอยู่ในทริกเกอร์ของ "KEY-COMMIT" แต่อย่างไรก็ตามฟอร์มที่สร้างขึ้นมามีความซับซ้อนต่างกันขึ้นกับจุดประสงค์ของการทำงาน ในที่นี้จะขอยกตัวอย่างฟอร์มที่มีการทำงานซับซ้อนมาก คือ ฟอร์มที่ใช้ถอนสิทธิของผู้ใช้งานคืน (PS_REVOK.FRM) และ ฟอร์มการสร้างผู้ใช้ระบบงาน (PS_GRANA.FRM, PS_GRANAF.FRM, PS_GRANU.FRM)

ในการสร้างฟอร์มถอนสิทธิผู้ใช้งานคืน (PS_REVOK.FRM) ผู้ที่สามารถสร้าง และใช้ฟอร์มนี้ได้ คือ ผู้บริหารฐานข้อมูล (DBA) ในที่นี้คือ system/manager เท่านั้น เพราะผู้บริหารฐานข้อมูลเป็นผู้ที่สามารถเรียกใช้ตารางของระบบได้ ซึ่งผู้ใช้ระบบทั่วไปถ้าไม่ใช่ผู้บริหารแล้วจะไม่สามารถเรียกใช้ตารางของระบบได้ การทำงานของฟอร์มถอนสิทธิผู้ใช้งานคืนนั้นเริ่มจาก

1. การให้ผู้บริหารระบุรหัสผ่านของตนเอง 2 ครั้ง ทั้งนี้เพราะว่ารหัสผ่านของผู้ใช้งานทุกคนจะถูกออราเคิลเข้ารหัสไว้ ไม่สามารถอ่านเพื่อนำมาใช้งานได้

2. จากนั้นฟอร์มจะค้นหารหัสผู้ใช้ที่ถูกถอนสิทธิการใช้ข้อมูลของระบบบุคลากร ซึ่งก็คือรหัสผู้ใช้ที่เคยสร้างขึ้นมาแต่ขณะนี้ไม่มีอยู่ในตารางผู้ใช้งาน (PS_USER) คำสั่งในการทำงานดังกล่าวอยู่ในกระบวนคำสั่ง (procedure) ที่ชื่อว่า "REVOKE_USER" (สิ่งพิมพ์รายการแสดง (listing) รูปที่ 5.3) จะเห็นได้ว่ามีการเรียกใช้โปรแกรม "N_REVOKE.BAT" (สิ่งพิมพ์รายการแสดงรูปที่ 5.8) เพื่อบันทึกคำสั่งที่ใช้ถอนสิทธิคืนจากผู้ใช้ดังกล่าวไว้ในแฟ้มที่มีชื่อเดียวกับรหัสผู้ใช้ (user_id) และมีนามสกุลคือ .RVU (Revoke User)

3. แล้วจึงเรียกใช้กระบวนคำสั่ง "REVOKE_FILE" (สิ่งพิมพ์รายการแสดงรูปที่ 5.4) และมีการเรียกใช้โปรแกรม "A_REVOKE.BAT" และ "E_REVOKE.BAT" (สิ่งพิมพ์รายการแสดงรูปที่ 5.9 และ 5.10 ตามลำดับ) เพื่อบันทึกคำสั่งที่ใช้เรียกข้อมูลคืนจากผู้ใช้ดังกล่าว แล้วเก็บไว้ในแฟ้มที่มีชื่อเดียวกันกับรหัสผู้ใช้ แต่จะมีนามสกุลของแฟ้ม คือ .RVF (Revoke Files) เช่น ในกรณีที่ผู้ใช้ที่มี user id คือ A220016 ถูกถอนสิทธิในการใช้ข้อมูลของระบบบุคลากร ก็จะทำให้เกิดแฟ้มที่บันทึกคำสั่งขึ้น 2 แฟ้มคือ A220016.RVU และ A220016.RVF เป็นต้น

4. โปรแกรมจะรวบรวมแฟ้มของคำสั่งทั้งหมด แล้วบันทึกไว้ในแฟ้มที่ชื่อว่า "PS_GRANT.SQL"

5. ย้อนกลับไปทำข้อ 2-4 กับผู้ที่ถูกถอนสิทธิทุกคน แล้วฟอร์มนี้จะมาเรียกใช้โปรแกรม "PS_GRANT.BAT" (สิ่งพิมพ์รายการแสดงรูปที่5.11) เพื่อทำการประมวลผลตามคำสั่งของโปรแกรม PS_GRANT.SQL

6. เมื่อจบสิ้นการทำงานของฟอร์มแล้ว ฟอร์มดังกล่าวจะเลิกการทำงานโดยอัตโนมัติ

ส่วนการสร้างผู้ใช้ ทั้ง 3 ฟอร์ม คือ

- ฟอร์มการสร้างผู้ใช้ทั้งมหาวิทยาลัย (PS_GRANA.FRM)
- ฟอร์มการสร้างผู้ใช้เฉพาะหน่วยงาน (คณะ) (PS_GRANF.FRM)
- ฟอร์มการสร้างผู้ใช้เฉพาะบุคคล (PS_GRANU.FRM)

จะมีวิธีการทำงานเหมือนกันเพียงแต่ขอบเขตของข้อมูลที่จะนำมาสร้างผู้ใช้ต่างกันเท่านั้น และมีวิธีการดังนี้

1. ตรวจสอบการถอนสิทธิคืนจากผู้ใช้ที่ไม่มีสิทธิ ซึ่งการทำงานเหมือนกับ ฟอร์มการถอนสิทธิคืนที่กล่าวไปแล้วข้างต้น

2. จากนั้นฟอร์มจะหารหัสของผู้ใช้ที่จะนำมาสร้าง โดยถ้าเป็นการสร้างผู้ใช้ทั้งมหาวิทยาลัย ก็จะนำรหัสผู้ใช้มาทั้งตารางผู้ใช้ (ps_user table) แต่ถ้าเป็นการสร้างผู้ใช้เฉพาะหน่วยงาน (คณะ) ก็จะเลือกรหัสผู้ใช้มาจากตารางผู้ใช้ (ps_user table) ที่สังกัดเฉพาะคณะที่ระบุมา ส่วนถ้าเป็นการระบุรหัสผู้ใช้มาก็จะนำรหัสผู้ใช้นั้นมาสร้างเลย

3. เมื่อเลือกผู้ใช้มาได้แล้วฟอร์มจะเรียกใช้กระบวนคำสั่ง "GRANT_USER" เพื่อสร้างผู้ใช้ให้ได้ใช้ข้อมูลของระบบบุคลากรตามสิทธิที่ผู้ใช้นั้น ๆ ได้รับมา โดยมีการเรียกใช้โปรแกรมต่าง ๆ ตามสิทธิของผู้ใช้ดังนี้

- gu_grant.bat	เก็บคำสั่งมอบข้อมูล และสร้างคำเหมือน ให้กับผู้ที่ได้รับสิทธิเป็น	"PS_GUS"
- pu_grant.bat	"	"PS_USR"
- pm_grant.bat	"	"PS_MGR"
- po_grant.bat	"	"PS_OPR"
-sm_grant.bat	"	"SP_MGR"
- so_grant.bat	"	"SP_OPR"

เพื่อที่จะบันทึกคำสั่งของการสร้างผู้ใช้ไว้ในแฟ้มชื่อเดียวกับรหัสผู้ใช้ (user_id) .GRN (Grant User)

4. จากนั้นจึงรวบรวมแฟ้มของคำสั่งดังกล่าวไปไว้ในแฟ้ม PS_GRANT.SQL เช่นเดียวกับฟอร์มการถอนสิทธิ นอกจากนี้ในการสร้างผู้ใช้ของระบบโปรแกรม .bat ข้างต้น จะบันทึกคำสั่ง GENMENU ของผู้ใช้ดังกล่าวไว้ในแฟ้มชื่อ "MENU_USR.BAT"

5. ทำเช่นนี้กับผู้ใช้ทุกคนที่เลือกมาสร้าง จากนั้นฟอร์มจะเลือกใช้โปรแกรม PS_GRANT.BAT เพื่อทำงานตามคำสั่งของโปรแกรม PS_GRANT.SQL และทำคำสั่ง GENMENU เพื่อให้ผู้ใช้สามารถใช้รายการเลือกต่าง ๆ ได้ตามสิทธิของตน

ส่วนการสร้างฟอร์มสำหรับการสอบถามนั้นต้องทำการสร้างวิวต่าง ๆ ก่อน ตามการออกแบบวิวให้กับผู้ใช้งานในบทที่ 4 (หัวข้อการออกแบบวิวให้กับผู้ใช้งาน) โดยแฟ้มที่บรรจุคำสั่งในการสร้างวิวเป็นแฟ้มที่มีนามสกุล ".vew" จากนั้นจึงสร้างรหัสผู้ใช้งานของออราเคิล ขึ้นมาอีกหนึ่งรหัส แล้วมอบสิทธิ (grant) การใช้วิวที่สร้างขึ้นมาให้กับผู้ใช้งานใหม่ที่สร้างขึ้นมา เพื่อสร้างฟอร์มการสอบถามโดยใช้รหัสของผู้ใช้งานใหม่นี้ ไม่ใช่รหัสผู้ใช้ "SU" และได้สร้างโปรแกรมคำสั่งที่ใช้ในการมอบวิว และสร้างคำเหมือนให้กับผู้ที่ทำหน้าที่สร้างฟอร์มการสอบถามไว้ในแฟ้มที่ชื่อว่า "GRAN_VEW.SQL" และ มีวิธีการเรียกใช้ดังนี้

```
SQL> @gran_vew  user_id1  password1  user_id2  password2
```

โดย user_id1 คือ รหัสผู้ใช้ที่เป็นผู้สร้างวิว
password1 คือ รหัสผ่านของ user_id1
user_id2 คือ รหัสผ่านของผู้ที่ทำหน้าที่สร้างฟอร์มการสอบถาม
password2 คือ รหัสผ่านของ user_id2

และถ้าต้องการถอนสิทธิการใช้วิวคืนจากผู้สร้างฟอร์มการสอบถาม ก็ได้สร้างโปรแกรมคำสั่งไว้ในแฟ้มที่ชื่อว่า "REVK_VEW.SQL" ซึ่งมีวิธีการเรียกใช้เหมือนกับวิธีการเรียกโปรแกรมการมอบวิว

```
DEFINE PROCEDURE

    NAME = revoke_user
    DEFINITION = <<<
    procedure revoke_user is
    begin
      Declare
    /*  revoke user from system PERSONNAL*/
          CURSOR revoke_user is
            select distinct user_name
            from menu_v_user, dba_users
            where menu_v_user.user_name = dba_users.username
            and   group_name in ('PS_GUS','PS_MGR','PS_USR','PS_OPR',
                                 'SP_OPR','SP_MGR')
            and   user_name not in (select user_id from su.ps_user);
          revoke_rec revoke_user%ROWTYPE;
        Begin
          :GLOBAL.USER := USER;
          OPEN revoke_user;
          LOOP
            FETCH revoke_user INTO revoke_rec;
            if revoke_user%FOUND then
              /*  revoke file  */
              host('n_revoke ' || revoke_rec.user_name,NO_SCREEN);
              revoke_file(revoke_rec.user_name);
            else
              exit;
            end if;
          end loop;
          close revoke_user;
        end;
    end;
    >>>

ENDDEFINE PROCEDURE
```

รูปที่ 5.3 แสดงสิ่งพิมพ์รายการแสดงกระบวนคำสั่ง REVOKE_USER

```
DEFINE PROCEDURE

  NAME = revoke_file
  DEFINITION = <<<
  procedure revoke_file( user_name in char) is
  begin
    declare
      cursor revoke_file is select synonym_name, table_name
                            from dba_synonyms
                            where owner = user_name
                            and table_owner = 'SU';
      table_name          char(30);
      synonym_name        char(30);
      flag_revoke         char(1);
      begin
        open revoke_file;
        flag_revoke := 'N';
        loop
          fetch revoke_file into synonym_name, table_name;
          if revoke_file%FOUND then
            flag_revoke := 'Y';
            host('a_revoke '||user_name||' '||table_name||
                 ' '||synonym_name,NO_SCREEN);
          else
            exit;
          end if;
        end loop;
        close revoke_file;
        if flag_revoke = 'Y' then
          host('e_revoke '||:GLOBAL.USER||' '||:PASSWORD||
               ' SU '||'SU '||user_name,NO_SCREEN);
        end if;
      end;
  end;
  >>>

ENDDEFINE PROCEDURE
```

รูปที่ 5.4 แสดงสิ่งพิมพ์รายการแสดงกระบวนคำสั่ง REVOKKE_FILE

```
DEFINE PROCEDURE

  NAME = grant_user
  DEFINITION = <<<
  procedure grant_user is
  begin
    Declare
  /*  grant user from system PERSONNAL*/
        CURSOR grant_user is
          select user_id,priority  from su.ps_user;
        grant_rec grant_user%ROWTYPE;
      Begin
        open grant_user;
        loop
          fetch grant_user into grant_rec;
          if grant_user%FOUND then
            revoke_file(grant_rec.user_id);
            if grant_rec.priority = 'PS_GUS' then
              host('gu_grant ' || grant_rec.user_id ||
    ' ' || grant_rec.priority || ' ' || :global.user ||
    ' ' || :password || ' SU ' || 'SU',NO_SCREEN);
            end if;
            if grant_rec.priority = 'PS_USR' then
              host('pu_grant ' || grant_rec.user_id ||
    ' ' || grant_rec.priority || ' ' || :global.user ||
    ' ' || :password || ' SU ' || 'SU',NO_SCREEN);
            end if;
            if grant_rec.priority = 'PS_MGR' then
              host('pm_grant ' || grant_rec.user_id ||
    ' ' || grant_rec.priority || ' ' || :global.user ||
    ' ' || :password || ' SU ' || 'SU',NO_SCREEN);
            end if;
            if grant_rec.priority = 'PS_OPR' then
              host('po_grant ' || grant_rec.user_id ||
    ' ' || grant_rec.priority || ' ' || :global.user ||
    ' ' || :password || ' SU ' || 'SU',NO_SCREEN);
            end if;
```

รูปที่ 5.5 แสดงสิ่งพิมพ์รายการแสดงกระบวนคำสั่ง GRANT_USER (*ทั้งมหาวิทยาลัย [PS_GRANA.INP]*)

```
            if grant_rec.priority = 'SP_MGR' then
              host('sm_grant ' || grant_rec.user_id ||
    ' ' || grant_rec.priority || ' ' || :global.user ||
    ' ' || :password || ' SU ' || 'SU',NO_SCREEN);
            end if;
            if grant_rec.priority = 'SP_OPR' then
              host('so_grant ' || grant_rec.user_id ||
    ' ' || grant_rec.priority || ' ' || :global.user ||
    ' ' || :password || ' SU ' || 'SU',NO_SCREEN);
            end if;
          else
            exit;
          end if;
        END LOOP;
        CLOSE grant_user;
        host('ps_grant ' || :global.user || ' ' || :password);
        ERASE ('GLOBAL.USER');
        EXIT_FORM;
      end;
    end;
>>>

ENDDEFINE PROCEDURE
```

รูปที่ 5.5 (ต่อ)

```
DEFINE PROCEDURE

  NAME = grant_user
  DEFINITION = <<<
  procedure grant_user is
  begin
    Declare
  /*  grant user from system PERSONNAL*/
        CURSOR grant_user is
          select user_id,priority
          from su.ps_user
          where fac_code = :key.fac_code;
        grant_rec grant_user%ROWTYPE;
      Begin
        open grant_user;
        loop
          fetch grant_user into grant_rec;
          if grant_user%FOUND then
            revoke_file(grant_rec.user_id);
            if grant_rec.priority = 'PS_GUS' then
              host('gu_grant ' || grant_rec.user_id ||
    ' ' || grant_rec.priority || ' ' || :global.user ||
    ' ' || :password || ' SU ' || 'SU',NO_SCREEN);
            end if;
                  ' '
                  ' '
                  ' '
          else
            exit;
          end if;
        END LOOP;
        CLOSE grant_user;
        host('ps_grant ' || :global.user || ' ' || :password);
        ERASE ('GLOBAL.USER');
        EXIT_FORM;
      end;
    end;
  >>>

ENDDEFINE PROCEDURE
```

รูปที่ 5.6 แสดงสิ่งพิมพ์รายการแสดงกระบวนคำสั่ง GRANT_USER (*เฉพาะหน่วยงานคณะ [PS_GRANF.INPJ*)

```
DEFINE PROCEDURE

  NAME = grant_user
  DEFINITION = <<<
  procedure grant_user is
  begin
    Declare
  /*  grant user from system PERSONNAL */
      Begin
        revoke_file(:key.user_id);
            if :key.priority = 'PS_GUS' then
              host('gu_grant ' || :key.user_id ||
      ' ' || :key.priority || ' ' || :global.user ||
      ' ' || :password || ' SU ' || 'SU',NO_SCREEN);
            end if;
            if :key.priority = 'PS_USR' then
              host('pu_grant ' || :key.user_id ||
      ' ' || :key.priority || ' ' || :global.user ||
      ' ' || :password || ' SU ' || 'SU',NO_SCREEN);
            end if;
                  ''
                  ''
                  ''
                  ''
        host('ps_grant ' || :global.user || ' ' || :password);
        ERASE ('GLOBAL.USER');
      end;
    end;
  >>>

ENDDEFINE PROCEDURE
```

รูปที่ 5.7 แสดงสิ่งพิมพ์รายการแสดงกระบวนคำสั่ง GRANT_USER (เฉพาะบุคคล *[PS_GRANU.INP]*)

```
cls
@echo on
rem Delete user (%1) from Personnal System
@echo off
echo delete MENU_V_USER where user_name = '%1' > %1.rvu
echo and group_name in ('PS_GUS','PS_MGR','PS_USR','PS_OPR','SP_OPR','SP_MGR'); >> %1.rvu
echo @%1.rvu >> ps_grant.sql
```

รูปที่ 5.8 แสดงสิ่งพิมพ์รายการแสดงโปรแกรม *N_REVOKE.BAT*

```
@echo on
rem Revoke file %2 from %1.
@echo off
echo drop synonym %1.%3; >> d_revoke.tmp
echo revoke all privileges on %2 from %1; >> r_revoke.tmp
```

รูปที่ 5.9 แสดงสิ่งพิมพ์รายการแสดงโปรแกรม *A_REVOKE.BAT*

```
@echo on
rem Revoke old file of Personnal System from user (%5)
@echo off
echo connect %1/%2 > %5.rvf
type d_revoke.tmp >> %5.rvf
echo connect %3/%4 >> %5.rvf
type r_revoke.tmp >> %5.rvf
echo commit; >> %5.rvf
echo connect %1/%2 >> %5.rvf
echo @%5.rvf >> ps_grant.sql
rem echo exit;
rem sqlplus %1/%2 @revoke
rem copy revoke.sql revoke.log
rem del revoke.sql
del d_revoke.tmp
del r_revoke.tmp
```

รูปที่ 5.10 แสดงสิ่งพิมพ์รายการแสดงโปรแกรม *E_REVOKE.BAT*

```
echo on
echo NOW! program is granting user.
echo off
echo exit; >> ps_grant.sql
sqlplus %1/%2 @ps_grant.sql
IF EXIST MENU_USR.BAT call menu_usr
IF EXIST *.GRN del *.grn
IF EXIST *.RVU del *.rvu
IF EXIST *.RVF del *.rvf
IF EXIST PS_GRANT.SQL del ps_grant.sql
IF EXIST MENU_USR.BAT del menu_usr.bat
```

รูปที่ 5.11 แสดงสิ่งพิมพ์รายการแสดงโปรแกรม *PS_GRANT.BAT*

นอกจากนี้ยังมีการทำโปรแกรมที่ใช้ในการสำรอง และเรียกคืนข้อมูลนั้นได้เรียกใช้โปรแกรมอรรถประโยชน์ของออราเคิลคือ EXP และ IMP แต่จะมีการกำหนดตัวแปรต่าง ๆ (parfile) ไว้ในแฟ้มตระกูล ".exp" และ ".imp" ซึ่งสามารถแก้ไข แล้วกำหนดตัวแปรใหม่ได้ (ข้อควรระวัง ในการเรียกข้อมูลคืนต้องล้างข้อมูลเก่าก่อน มิฉะนั้นข้อมูลที่เรียกคืนจะเข้าไปซ้ำซ้อนกับข้อมูลเดิม โดยแฟ้มที่เก็บคำสั่งในการล้างข้อมูลของแต่ละรายการเลือกสำหรับการเรียกข้อมูลคืนจะอยู่แฟ้มที่มีนามสกุล ".drp") ซึ่งมีตัวอย่างรายละเอียดของแฟ้มกำหนดตัวแปรต่าง ๆ เช่น

```
FILE=C:\personbk\ref_tabs.dmp
COMPRESS=Y
GRANTS=Y
INDEXES=Y
TABLES=(ABS_TYPE,CMD_TYPE,COUNTRY,CURRICULUM,DEPARTMENT,DIVISION,DUTY,
EDUCATION_LEVEL,FACULTY,FUND,LOAN,MARITAL_STATUS,POS_LEVEL,POSITION_STATUS,
PROFESSION,PROJECT_TYPE,PROVINCE,PUNISH_TYPE,ORGANIZATION,RACE,NATION,REWARD,
RELATION,RELIGION,RES_FINISH_STATUS,SPOUSE_STATUS,TERM_REASON,WORK_TYPE)
```

รูปที่ 5.12 สิ่งพิมพ์รายการแสดงของ REF_TABS.EXP

```
FILE=C:\personbk\ref_tabs.dmp
IGNORE=N
GRANTS=Y
INDEXES=Y
TABLES=(ABS_TYPE,CMD_TYPE,COUNTRY,CURRICULUM,DEPARTMENT,DIVISION,DUTY,
EDUCATION_LEVEL,FACULTY,FUND,LOAN,MARITAL_STATUS,POS_LEVEL,POSITION_STATUS,
PROFESSION,PROJECT_TYPE,PROVINCE,PUNISH_TYPE,ORGANIZATION,RACE,NATION,REWARD,
RELATION,RELIGION,RES_FINISH_STATUS,SPOUSE_STATUS,TERM_REASON,WORK_TYPE)
```

รูปที่ 5.13 สิ่งพิมพ์รายการแสดงของ REF_TABS.IMP

สุดท้ายก็เป็นการสร้างรายการเลือกต่าง ๆ โดยใช้ SQL*Menu และได้สร้างแฟ้มของรายการเลือก 3 แฟ้ม ด้วยกันคือ PS_ENTRY PS_QUERY และ UTILITY ในการสร้างรายการเลือกนี้ จะทำให้ได้แฟ้มตระกูล ".dmm"

การทดสอบโปรแกรม

จุดประสงค์ของการทดสอบโปรแกรมก็เพื่อค้นหาข้อผิดพลาดของโปรแกรม ซึ่งอาจจะเป็นข้อผิดพลาดทางไวยกรณ์ หรือการทำงานของโปรแกรมที่ไม่ตรงตามวัตถุประสงค์ที่ตั้งไว้ เป็นต้น ดังนั้นขั้นตอนในการทดสอบโปรแกรมในขั้นนี้จะแบ่งการทดสอบจริง ๆ เป็น 2 ขั้นตอน คือ

1. Syntax Program Testing คือการเอาสิ่งพิมพ์รายการแสดงของโปรแกรม (listing program) ที่ได้จากการแปลโปรแกรม (compile) มาตรวจสอบดู เป็นการตรวจสอบความถูกต้องของรูปแบบ โดยตัวแปลโปรแกรม (compiler) จะเป็นผู้ทำหน้าที่ตรวจสอบวากยสัมพันธ์ของโปรแกรมเอง และรายงานความผิดพลาดออกมาให้ทราบ หากไม่มีข้อผิดพลาดใด ๆ ปรากฏก็จะต้องตรวจสอบความครบถ้วนของโปรแกรมต่อไป โดยการนำเอาสิ่งพิมพ์รายการแสดงของโปรแกรมที่ได้มาตรวจว่าตรงกับรหัสต้นฉบับ (source code) ที่ป้อนเข้าไปหรือไม่ มีคำสั่งเกินมา หรือขาดหายไปอย่างไร

2. Logic Program Testing มีรูปแบบการตรวจสอบที่สำคัญแบ่งเป็น 2 แบบคือ

2.1 Data Driven Testing ในลักษณะนี้จะมองรูปแบบของโปรแกรมเป็นเสมือนกล่องดำ (Black box) ทดสอบโดยอาศัยป้อนข้อมูลที่สมมุติขึ้นเข้าไปโดยไม่สนใจว่าโปรแกรมจะมีตรรกะ (logic) อย่างไร และนำผลลัพธ์ออกมาตรวจสอบกับผลลัพธ์ที่เราทราบอยู่แล้ว วิธีนี้เป็นวิธีที่ง่าย และเปิดโอกาสให้ผู้ใช้งานมีส่วนร่วมในการทดสอบด้วย แต่สิ่งที่ต้องระวังคือค่าข้อมูลที่นำมาทดสอบกับโปรแกรมจะต้องเป็นข้อมูลที่เป็นตัวแทนครอบคลุมข้อมูลทั้งหมดของงานที่จะทำจริงในระบบงาน และรวมไปจนถึงการตรวจสอบด้วยค่าข้อมูลที่ไม่เป็นจริงแล้ว โปรแกรมจะต้องจบการทำงานในลักษณะสิ้นสุดปกติ (Normal End)

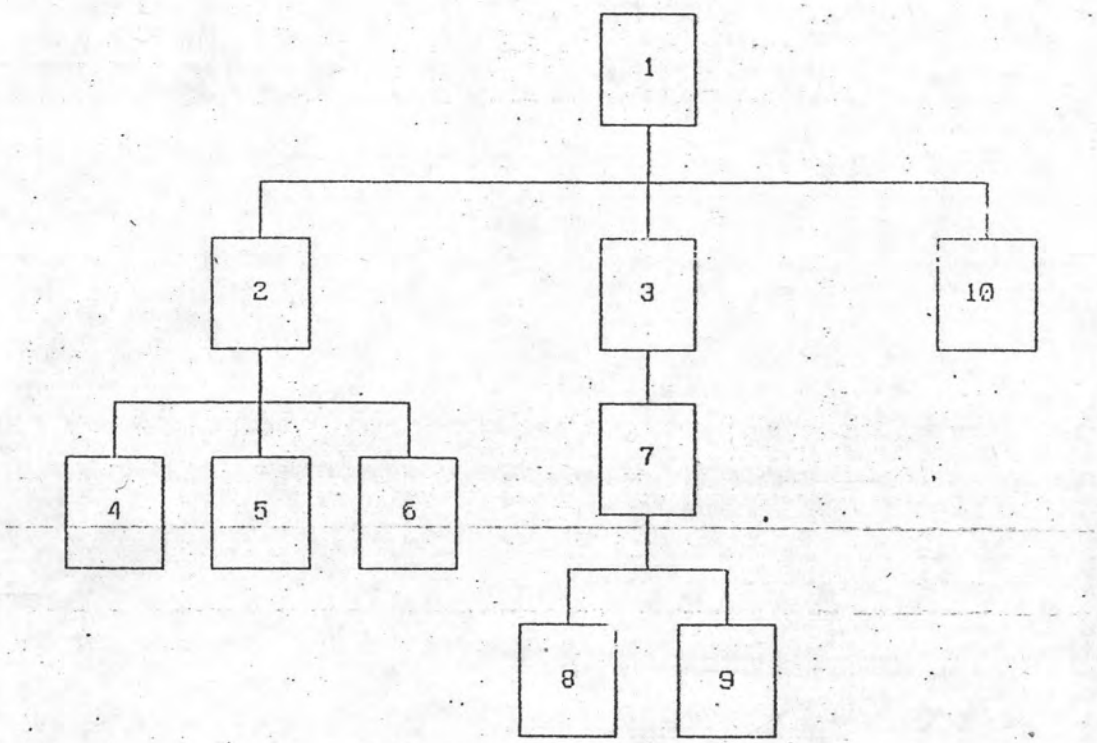
2.2 Logic Driven Testing จากการพิจารณาตรรกะของโปรแกรม จะสมมุติข้อมูลขึ้นมาแล้วป้อนเข้าไปในโปรแกรม ซึ่งในลักษณะนี้จะมองโปรแกรมเป็นเสมือนกล่องขาว (White box) และจะติดตามการทำงานของข้อมูลในแต่ละส่วนของโปรแกรมจนกระทั่งได้ผลลัพธ์ออกมา วิธีนี้จำเป็นต้องใช้ผู้มีความสามารถด้านโปรแกรม อาจเป็นโปรแกรมเมอร์ หรือผู้ที่ได้รับการแต่งตั้ง การทดสอบอาจจะทำแบบ desk checking หรือใช้คอมพิวเตอร์โดยอาศัยโปรแกรมอรรถประโยชน์บางตัวช่วยในการตรวจสอบการทำงานของข้อมูลในโปรแกรม

ในกรณีที่โปรแกรมถูกออกแบบโดยแบ่งเป็นมอดูลย่อย เราไม่จำเป็นที่จะต้องทดสอบทีเดียวทั้งโปรแกรมอาจจะทดสอบโปรแกรมในลักษณะที่เรียกว่า Structured Testing ซึ่งมีเทคนิคใน การทดสอบเป็น 2 ลักษณะ คือ

- TOP-DOWN Testing เป็นการทดสอบจากโปรแกรมหลัก ไปสู่โปรแกรมย่อย โดยการทดสอบจะเริ่มจากมอดูล 1 ก่อน ไม่สนใจมอดูลอื่น ๆ โดยตัดในส่วนที่มีความสัมพันธ์กับมอดูลย่อย (Sub module) อื่นออกก่อน แล้วทดสอบเฉพาะมอดูล 1 ล้วน ๆ ข้อเสียคือส่วนที่เรียกมอดูลย่อยจะไม่ทำงาน

จึงต้องทำการสร้าง stub module คือส่วนที่ทำหน้าที่เป็นมอดูลหลอก ๆ ที่มีแต่ชื่อของโปรแกรมย่อย (sub program) และ ส่งค่ากลับ (return) หากมีการส่งค่าระหว่างมอดูลก็ให้สมมติค่าแทนขึ้น

ลักษณะการทดสอบ จะสมมติ stub module เป็นขั้นตอนดังนี้

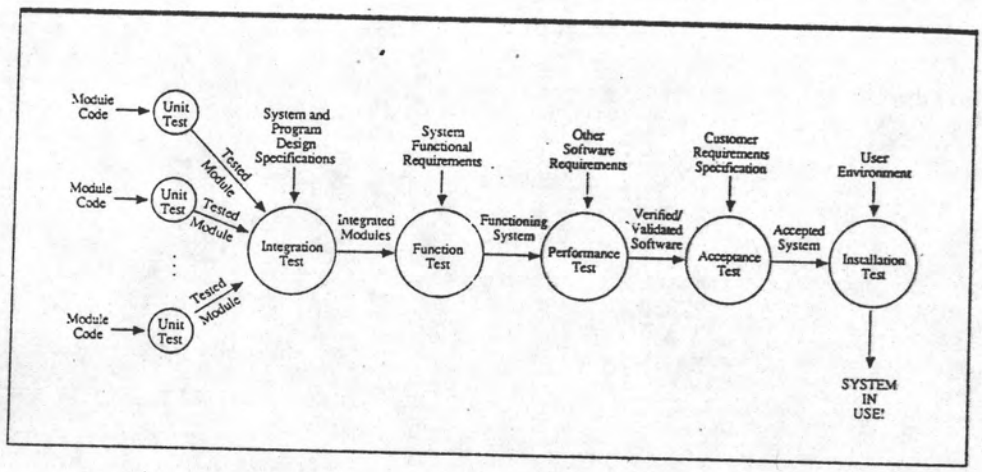


Test Module	Stub Module
1	2, 3, 10
1, 2	3, 10, 4, 5, 6
1, 3	2, 7, 10
1, 10	2, 3
1, 2, 3, 10, 4, 5, 6, 7	8, 9

- BOTTOM-UP Testing เป็นการทดสอบจากโปรแกรมย่อยแต่ละมอดูลอย่างอิสระ แล้วค่อยนำมารวมไปหาโปรแกรมหลักทีหลัง การทดสอบโปรแกรมย่อยมีวิธีการสำคัญ 2 วิธี เนื่องจากโปรแกรมย่อยไม่สามารถทำงานได้ด้วยตนเอง โดย

- สร้าง test dirver module เพื่อ perform หรือ call test module ได้
- สร้าง test module ให้เป็นโปรแกรมที่ทำงานเป็นอิสระได้ เมื่อทดสอบเสร็จแล้วจึงค่อยเปลี่ยนกลับ แต่ไม่นิยมทำกันเนื่องจากต้องเปลี่ยนรูปแบบของมอดูล

ในทางปฏิบัติอาจสามารถทำ TOP-DOWN และ BOTTOM-UP ไปพร้อม ๆ กันได้ แล้วแต่เหตุการณ์ในการพัฒนาโปรแกรมว่า ส่วนใดจะออกแบบแล้วเสร็จก่อน และถ้าเป็นระบบใหญ่ที่จะนำไปใช้งานจริงนั้น จะมีขั้นตอนในการทดสอบมากมาย ดังรูปที่ 5.14 [6]



รูปที่ 5.14 แสดงขั้นตอนของการทดสอบระบบ (Stages of Testing)