
# DEVELOPING INFORMATION SYSTEM OF SEED AND SEEDLING MANAGEMENT FOR SOIL AND WATER CONSERVATION : A CASE STUDY OF LAND DEVELOPMENT DEPARTMENT

WARUNEE PIANLERTPOJ

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE (TECHNOLOGY OF INFORMATION SYSTEM MANAGEMENT)
FACULTY OF GRADUATE STUDIES
MAHIDOL UNIVERSITY
2000

ISBN 974-664-862-4

Thesis
entitled

# DEVELOPING INFORMATION SYSTEM OF SEED AND SEEDLING MANAGEMENT FOR SOIL AND WATER CONSERVATION: A CASE STUDY OF LAND DEVELOPMENT DEPARTMENT

Warunee Pianlertpoj
Miss Warunee Pianlertpoj
Candidate

Lect.Bunlur Emaruchi, Ph.D.
Major-advisor

Lect.Thanakorn Uan-On, D.Engr.
Co-advisor

Piyada Chitchumnong
Mrs.Piyada Chitchumnong, M.Sc.
Co-advisor

Suttinant Nantachit
Lect.Suttinant Nantachit, M.S.
Co-advisor

Prof.Liangchai Limlomwongse, Ph.D.
Dean
Faculty of Graduate Studies

Lect.Thanakorn Uan-On, D.Engr.
Chairman
Master of Science Programme in Technology of Information System Management
Faculty of Engineering

Thesis
entitled

# DEVELOPING INFORMATION SYSTEM OF SEED AND SEEDLING MANAGEMENT FOR SOIL AND WATER CONSERVATION: A CASE STUDY OF LAND DEVELOPMENT DEPARTMENT

was submitted to the Faculty of Graduate Studies, Mahidol University for the degree of Master of Science (Technology of Information System Management)

on

October 13, 2000

Warunee Pianlertpoj
.....
Miss Warunee Pianlertpoj
Candidate

.....
Lect.Bunlur Emaruchi, Ph.D.
Chairman

Kasem Kulpradit
.....
Asst.Prof.Kasem Kulpradit, M.Sc.
Member

.....
Lect.Thanakorn Uan-On, D.Engr.
Member

Piyada Chitchumnong
.....
Mrs.Piyada Chitchumnong, M.Sc.
Member

Suttinant Nantachit
.....
Lect.Suttinant Nantachit, M.S.
Member

.....
Prof.Liangchai Limlomwongse, Ph.D.
Dean
Faculty of Graduate Studies
Mahidol University

.....
Lect.Thanakorn Uan-On, D.Engr.
Dean
Faculty of Engineering
Mahidol University

# ACKNOWLEDGEMENT

I would like to express my sincere gratitude and deep appreciation to Dr. Bunlur Emaruchi my advisor for his valuable advice and suggestion. I would particularly like to thanks Dr.Thanakorn Uan-on Mr. Suttinant Nantachit and Mrs. Piyada Chitchumnong my committee members for their extremely support and efforts in valuable data of Land Development Department.

I would like to thank the staff of Land Development Division1 (Phatumthani) for his helpful guidance and support with respect to the seed and seedling management operation of the data. Thanks are also extended to all my friends for their support and wonderful friendship.

In addition, I am indebted to my family for their love, everlasting supports and enthusiasm throughout the whole life.

Warunee Pianlertpoj

4037686 EGTI/M : MAJOR : TECHNOLOGY OF INFORMATION SYSTEM MANAGEMENT; M.Sc. (TECHNOLOGY OF INFORMATION SYSTEM MANAGEMENT)

KEY WORDS : SYSTEM ANALYSIS / SYSTEM DESIGN / INFORMATION SYSTEM DEVELOPMENT / SEED AND SEEDLING MANAGEMENT / CLIENT/SERVER DATABASE

WARUNEE PIANLERTPOJ: DEVELOPING INFORMATION SYSTEM OF SEED AND SEEDLING MANAGEMENT FOR SOIL AND WATER CONSERVATION: A CASE STUDY OF LAND DEVELOPMENT DEPARTMENT. THESIS ADVISORS: BUNLUR EMARUCHI, Ph.D., THANAKORN UAN-ON, D.Engr., SUTTINANT NANTTACHIT, M.S., PIYADA CHITCHUMNONG, M.Sc. 197 p. ISBN 974-664-862-4

As the amount of seed and seedling and stock transaction increases, it becomes more and more difficult to remember or approximate stock balances. An information system is needed to supplement the human mind. This research is aimed at introducing an information system of seed and seedling management for soil and water conservation, a case study of Land Development Department, which could reduce access time, increase accuracy and consistency of data. The information system was designed and developed using data flow analysis, relational database (Sybase SQL anywhere 5.0 as DBMS) and Object-Oriented programming (Powersoft Powerbuilder Enterprise Version 6.0).

The information system consists of three major processes: 1) Data manipulation: Manipulate relative data; 2) Transaction processing: Support operation management, which are distribution, transfer, purchase, production and receiving; 3) Query and report: Process pre-defined outputs and preplanned printed reports. Since the information system works on client/server network, users are divided into 3 levels: database administrator, operator and general user. Moreover, the information system used in many places, may not support an online system. For this reason, the data will be sent and executed at database center by electronic mail or file transfer process.

The information system solves and improves the inefficiency of seed and seedling management. Furthermore, the information system can develop functions for supporting electronic signature, applied to Internet/Intranet or OLAP application. In addition, the information system can be linked to another relative systems for a complete seed and seedling management system, which can be applied to EIS.

4037686 EGTI/M : สาขาวิชา : เทคโนโลยีการจัดการระบบสารสนเทศ;
วท.ม.( เทคโนโลยีการจัดการระบบสารสนเทศ)

วารุณี เพียรเลิศพจน์ : การพัฒนาระบบสารสนเทศการจัดการพันธุ์พืชเพื่ออนุรักษ์ดินและน้ำ กรณีศึกษาของกรมพัฒนาที่ดิน (DEVELOPING INFORMATION SYSTEM OF SEED AND SEEDLING MANAGEMENT FOR SOIL AND WATER CONSERVATION: A CASE STUDY OF LAND DEVELOPMENT DEPARTMENT). คณะกรรมการควบคุมวิทยานิพนธ์: บันลือ เอมะรุจิ,Ph.D., ธนากร อ้วนอ่อน,D.Engr., สุทธินันท์ นันทจิต M.S., ปิยดา จิตต์จำนงค์, M.Sc. 197 หน้า. ISBN 974-664-862-4

งานวิจัยนี้เสนอระบบสารเทศการจัดการพันธุ์พืชเพื่ออนุรักษ์ดินและน้ำ กรณีศึกษาของกรมพัฒนาที่ดิน ที่ช่วยลดเวลาการเข้าถึงข้อมูล เพิ่มความถูกต้องและสอดคล้องของข้อมูล ระบบสารสนเทศออกแบบและพัฒนาด้วยวิธีวิเคราะห์กระแสข้อมูล (Data flow analysis), ฐานข้อมูลเชิงสัมพันธ์ (relational database ใช้ Sybase SQL anywhere 5.0 เป็น DBMS) และการเขียนโปรแกรมเชิงวัตถุ (Object-Oriented programming ใช้ Powersoft Powerbuilder Enterprise Version 6.0).

ระบบการจัดการพันธุ์พืชเพื่ออนุรักษ์ดินและน้ำประกอบด้วย 3 ส่วนคือ 1) การจัดการฐานข้อมูล: จัดการข้อมูลพื้นฐานที่เกี่ยวข้องในระบบ 2) การประมวลผลรายการเปลี่ยนแปลง: สนับสนุนงานจัดการพันธุ์พืช ได้แก่ เบิก-จ่าย, โอน, จัดซื้อ, ผลิต และ ตรวจรับ 3) การสอบถามและรายงาน: ประมวลการสอบถามและรายงาน เนื่องจากระบบการจัดการพันธุ์พืชนี้ทำงานบนระบบไคเอนต์ เซิฟเวอร์ จึงแบ่งผู้ใช้ออกเป็น 3 ระดับคือ ผู้บริหารฐานข้อมูล, เจ้าหน้าที่ปฏิบัติงาน และ ผู้ใช้ทั่วไป และระบบสารสนเทศนี้ใช้ในหลายแห่ง ซึ่งบางแห่งไม่รองรับการทำงานแบบออนไลน์ทำให้การส่งโอนข้อมูลไปยังส่วนกลางทำได้โดยใช้วิธีส่งจดหมายอิเลคทรอนิค (E-mail) หรือ การส่งถ่ายแฟ้มข้อมูล (File transfer process) ระบบสารสนเทศนี้สามารถนำไปพัฒนาฟังก์ชั่นที่สนับสนุนการกำหนดสิทธิด้วยวิธีสร้างเป็นลายเซ็นอิเลคทรอนิค (Electronic signature) หรือประยุกต์เป็นโปรแกรมประยุกต์บนอินเตอร์เน็ตหรืออินทราเน็ต หรือนำไปเชื่อมต่อกับระบบที่เกี่ยวข้องเพื่อให้ได้ระบบจัดการพันธุ์พืชที่สมบูรณ์ ซึ่งระบบใหม่ที่รวมขึ้นยังสามารถพัฒนาต่อเป็นระบบสนับสนุนสารสนเทศสำหรับผู้บริหาร(EIS)

# CONTENTS

# CONTENTS (Continued)

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF FIGURES (Continued)

# LIST OF FIGURES (Continued)

# LIST OF FIGURES (Continued)

# CHAPTER I

# INTRODUCTION

## 1.1 Background and Statement of Problems

Nowadays, many computerized information systems are developed to reports containing information for decision makers. The information in these reports helps managers monitor and control business processed and operations. For example, reports that list number and quantity of inventory items in stock can be used to monitor inventory levels. In other words, information system is a collection of reports that are distributed to managers.

In stock management, for example, information systems are used to provide quantity of adding in stock, process user orders, control inventory levels, and so on. Thus, information system of seed and seedling management for soil and water conservation are developed for auditing quantity of seed and seedling in each stock, monitoring stock levels, provision number of seed and seedling whenever the number on hand is very low.

The main functions of seed and seedling management are distribution, transfer, purchase, production, receiving and reporting. As the amount of seed and seedling and stock transaction increases, it becomes more and more difficult to remember or approximate stock balances. Some form of record keeping is needed to supplement the

human mind, regardless of whether the records are operated by information system. The problems of inefficient operation are as follows:

1. Different format of data store.

Each section differently keeps data of seed and seedling; consequently seed and seedling management cannot retrieve entire department data.

2. Incorrect and inconsistent data.

Manipulation data is analog operation; officer must manipulate each file by himself so rate of incorrect and inconsistent data is higher than electronic operation. Updating data is the biggest factor of inconsistent data because officer may forget to update some files.

3. Difficult and slow data searching.

In case of a lot of data; officer takes his time with manual searching and may not discover data in time.

4. Difficult data checking.

Checking data of seed and seedling (such as quantity, location), employee (such as name, authority) and so on that officer must use ID (for example: plant_id, emp_id, etc.) to check the data. If he does not know the ID, this operation will spend a lot of time. Comparison with information system, he can use drill-down technique or entity relationship to discover the data.

5. Incorrect result.

There are many queries in stock control, some queries use arithmetic, searching and checking to calculate and summarize the result. So, it is possible to get incorrect result.

6. Inconsistent report.

It's the same query process, the process may produce inconsistent and incorrect reports.

The problems of inefficiency operation can be solved and improved by information system. The information system changes operation from analog to electronics, search from manual to digital and database from analog to digital. In addition, the information system introduces online operation, which reduces access time, increases accuracy and consistent data.

The Land Development Department was established on 23 May 1963 within the Ministry of National Development. Some years after their establishment, the government agencies were restructured, and as from 29 September 1972 the Land Development Department was transferred to the Ministry of Agriculture and Cooperatives. Land Development Department is responsible for:

- Soil survey, classification, and analysis and land use planning.
- Suggest policy and planning for setting appropriate land use, solve problems of soil using
- Conduct experiments and carry out research on various aspects of land development.
- Assist farmers in soil and water conservation and soil improvement, the production of seeds for cover crops and the production of soil improvement materials.
- Transfer knowledge of soil development and soil science for multiple purpose use. (1)

Land Development Department has compiled a long recorded data under his/her responsibility. At present, a seed and seedling management operation of Land Development Department has not information system to manage these data.

## 1.2 Objectives

To analyze and design information system of seed and seedling management for soil and water conservation.

To create database and implement information system of seed and seedling management for soil and water conservation for a case study of Land Development Department.

## 1.3 Scope of Work

The information system of seed and seedling management for soil and water conservation of Land Development Department is developed by Powersoft PowerBuilder Enterprise version 6.0. The information system works on client/server network, which supports operation management and has functions as follows:

- Distribution: This is a seed and seedling distribution control.
- Transfer: This is a seed and seedling transfer control.
- Purchase: It records data of purchasing.
- Production: It records data of production.
- Receiving: When transaction of distribution, transfer, purchase and production finished, officer will use this function to receive seed and seedling and use to update data into stock file.

- Inquiry and report: The outputs of this function are report and result for manger (head of section) such as seed and seedling details, quantity of seed in each stock and so on.

This information system has central relational database model that is created by Sybase SQL anywhere 5.0. The database consists of 2 parts as follows:

- Historical record of standard seed and seedling detail such as type, characteristic, etc.
- Historical records of seed and seedling management operation (e.g. distribute, transfer, purchase, produce, and receive).

## 1.4 Expected Results

The work should results information system of seed and seedling management for soil and water conservation, which consists of

1. Seed and seedling distribution module;
2. Seed and seedling transfer module;
3. Seed and seedling purchase module;
4. Seed and seedling production module;
5. Seed and seedling receiving module; and
6. Inquiry and report module.

Which have the capability of tracking seed and seedling management operation and improving inefficient operation.

# CHAPTER II

# LITERATURE SURVEY

## 2.1 Management Information Systems

A *management information system (MIS)* is an organized collection of people, procedures, databases, and devices used to provide routine information to managers and decision makers. The focus of and MIS is on operation efficiency. (10)

Management positions in a firm are often broken into three levels: upper management (the president and vice-presidents), middle management (the people responsible for anything between upper and lower management), and lower management (the people directly responsible for managing those who produce the firm's outputs). The roles of these levels of management are summarized in the pyramid in Figure 2.1

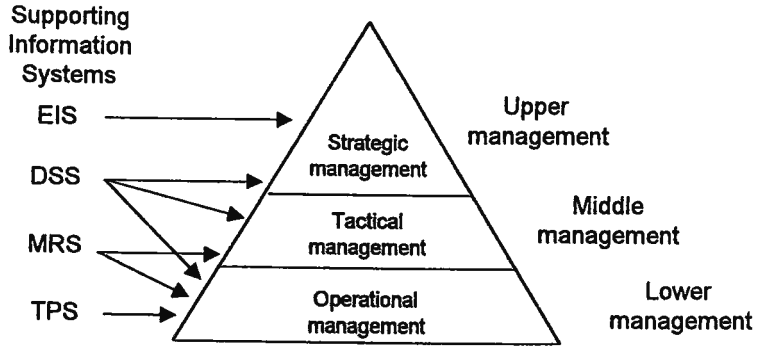



Figure 2.1 The three levels of management

Management information system (MIS) is any system that provides people with either data or information relating to an organization's operation. The systems provide routine information to managers and decision-maker. The focus of a MIS is on operation efficiency. (6,7)

The MIS effort in an organization as composed of the following four subsystems: TPS, MRS, DSS and OIS. (See Figure 2.2)

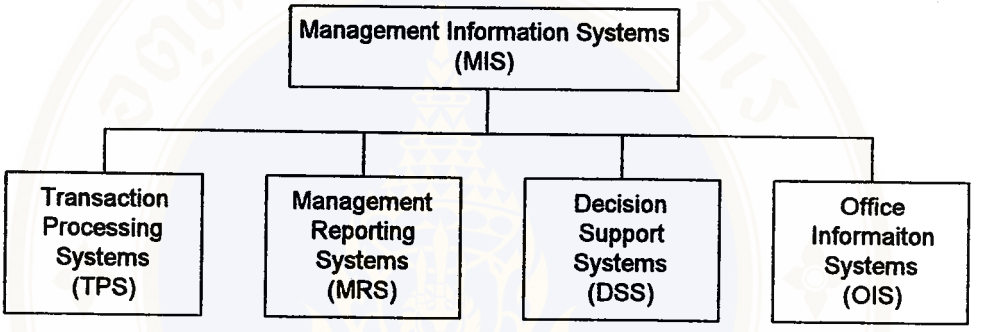



Figure 2.2 MIS subsystems

***Transaction Processing Systems***

A *transaction Processing Systems (TPS)* supports the processing of a firm's business transactions. For example, the TPS of a department store can record customer purchases, prepare billings to customers, and order merchandise for suppliers.

A *transaction Processing Systems (TPS)* keep an organization running smoothly by automating the processing of the voluminous amounts of paperwork that must be handled daily.

A *transaction* is any business-related exchange such as payments to employees, sales to customers, and payments to suppliers. A *transaction processing system (TPS)* is an organized collection of people, procedures, databases, and devices used to record completed business transactions, such as payroll.

***Management Report Systems***

*Management Report Systems (MRS)* is an information system that provides predefined types of information to management for relatively structured types of decision.

***Decision Support Systems***

A *decision Support Systems (DSS)* provides tools that enable managers to develop information in the manner that best suits the decisions they are currently trying to make.

*Decision Support Systems* as a system that provides tools to managers to assist them in solving semistructured and unstructured problems. A DSS is not intended to make decisions for managers, but rather to provide managers with a set of capabilities that enables them to generate the information they feel is needed to make decisions. In other words, a DSS supports the human decision-making process, rather than providing a means to replace it.

A *decision support system (DSS)* is an organized collection of people, procedures, database, and devices used to support problem-specific decision making. The focus of a DSS is on decision-making effectiveness.

***Office Information systems***

*Office Information systems (OIS)* include the use of such computer-based, office-oriented technologies as word processing, desktop publishing, electronic mail, video teleconferencing, and so on. (10,12)

## 2.2 Client/server technologies

### 2.2.1 Client technologies

The components of the client workstation are the basic workstation hardware, the operating system, the database connectivity software, applications, and a graphical-user interface (GUI).

***Client operating system.*** The primary purpose of the workstation operating system is to provide applications operating on that workstation with access to the hardware resources (memory, disk data storage, video and printer output, etc.) of that workstation and manage the interfaces between that workstation and devices external to that workstation (video, printer, etc.).

### 2.2.2 Server technologies

The server functions as the shared resource half of the client/server equation. As such it must be able to service multiple, often simultaneous, requests for data from the various application clients throughout the network. The technology of the server platform can range from basically equivalent to the client up to and including large mainframes.

***Server operation system.*** Like the client's operating system, the primary purpose of the server operating system is to provide applications operating on that server with access to the hardware resources (memory, disk data storage, video and printer output, etc.) of that server and manage the interfaces between that workstation and devices external to that workstation (video, printer, etc.).

Unlike the client, the nature of the server's mission as a shared resource for many clients requires greater sophistication and functionality beyond that needed by the typical client, and the facilities it provides closely resemble those provided by proprietary multi-user operating systems. (3)

## 2.3 Client/server characteristics

### 2.3.1 Client Attributes

The client process is proactive, issuing requests to the server. It typically is dedicated to its user's session and begins and ends with the session. A client may interact with a single server or with multiple servers to accomplish its work. However, at least one server process is always necessary.

At the application level, the client is responsible for maintaining and processing the entire dialog with the user. This typically includes performing all of the following.

- Screen handling
- Menu or command interpretation
- Data entry and validation
- Help processing
- Error recovery

In graphical applications, this also include all

- Window handling
- Mouse and keyboard entry
- Dialog box control
- Sound and video management (in multimedia application)

The client can usually handle all data preparation activities, single- and cross-field edits, and many table-driven data edits (such as validating codes used).

### 2.3.2 Server Attributes

The server process is reactive, triggered by the arrival of requests from its clients. A server process usually runs forever, providing services to many clients. These services may be provided either directly by the server process itself or indirectly by slave processes spawned for each client request. Figure 2.3 shows the use of master and slave server processes to provide a logical server for a client. For example, the Oracle database server process runs forever, accepting client request. When a request arrives, the Oracle server spawns a slave process dedicated to handling that request, allowing the master process to receive other requests immediately.

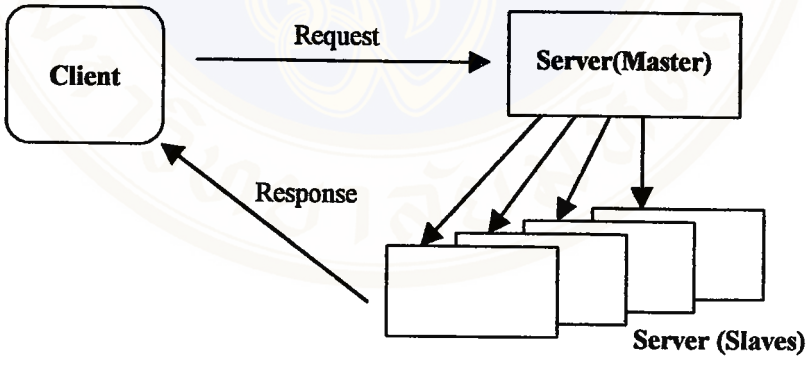



Figure 2.3 Master/slave server processes.

### 2.3.3 Communication Attributes

The style of communication between the client and server is transactional and cooperative. A characteristic of transaction-style communication is that the server sends back only the results relevant to the client's request.

For example, many early microcomputer networked applications were built using only filesharing technology. When a user started an application, the fileserver sent the entire executable across the network and loaded it into the desktop's memory. If the application used a database, the fileserver also sent the entire database executable to the desktop. When the database executable accessed the database, most of its data files had to be transmitted across the network to process the request. Even writing a few bytes of data required sending all the index files to the desktop so that they could be updated. When this application was replaced with a client/server implementation, the database executable remained and ran on the server. The server also sent much less data to the desktop, since index file I/O stayed at the server. Simple write requests generated network traffic consisting of just the data being written, the small return codes returned, and some packet header overhead. Overall, the network traffic is roughly equal to the amount of data written.

The attributes of client and server were explored in detail and are summarized in Table 2.1. The attributes of client/server communication were also explored. (2)

Table 2.1 Key client/server attributes

| Attribute | Client | Server |
|---|---|---|
| Mode | Proactive | Reactive |
| Execution | Fixed start and end | Runs forever |
| Primary purpose | Maintain user dialog | Provide functional service |
| | – Screen/window handing | – Application data sharing |
| | – Menu/command interpretation | – Communications sharing |
| | – Mouse/keyboard entry | – Filesharing |
| | – Data entry and validation | – Printer shaing |
| | – Help processing | – CPU sharing |
| | – Error recovery | – Display sharing |
| Transparency | Hides network and servers | Hides service implementation details |
| Includes | Communication with different servers | Communication with different clients |
| Excludes | No client-client communication | No server-server communication |

## 2.4 Client/server Database Model

In the true client/server database model, shown in Figure 2.4, the database again resides on a machine other than those that run the application processing components. But the database software is split between the client system that runs the application program and the server system that stores the database.

In this model, the application processing components on the client system make requests of the local database software. The local database software component in the client machine then communicates with the complementary database software running on the server. The server database software makes requests for accesses to the database and passes results back to the client machine.

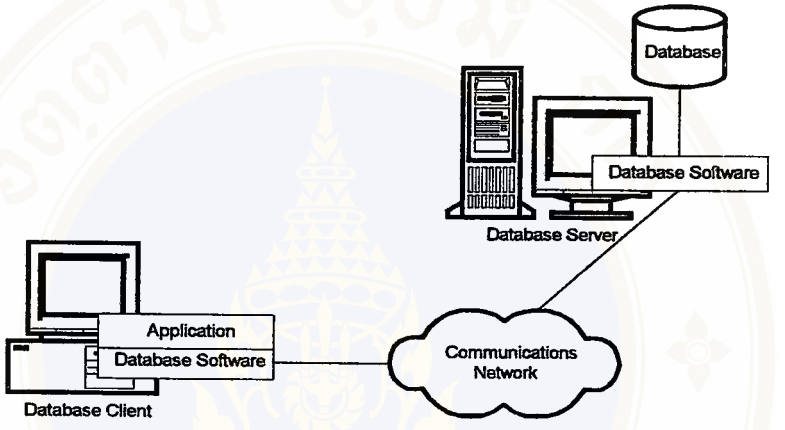



Figure 2.4 The Client/server database model

In the client/server database model, it is common to refer front-end software and back-end software. *Front-end software* typically runs on a personal computer or desktop workstation and serves the computing needs of a single individual. The front-end software typically plays the role of the client in a client/sever database application and performs functions that are oriented to the needs of the end user.

*Back-end software* consists of the client/server database software and network software that runs on a machine playing the role of database server. (4)

## 2.5 Distributed Database

Distributed database is a database that is distributed among a network of geographically separated locations.

Distributed database – a database in which the actual data may be spread across several smaller databases connected via telecommunications devices. Distributed databases give corporations more flexibility in now databases are organized and used.

Distributed databases allow more users direct access at different user site. To reduce the demand on telecommunications lines, some organizations will build a replicate database. A replicated database is a database that holds a duplicate set of frequently used data. At the beginning of the data, a company will send a copy of important data to each distributed processing location. At the end of the day, the different sites will send the changed data back to be stored in the main database. (5,6,7)

## 2.6 Relational Data Model

**Relational Model** represents data and relationships among data by a collection of tables, each of which has a number of columns with unique names (Figure 2.5) (8)

| *name* | *Street* | *city* | *number* |
|---|---|---|---|
| Lowery | Maple | Queens | 900 |
| Shiver | North | Bronx | 556 |
| Shiver | North | Bronx | 647 |
| Hodges | Sidehill | Brooklyn | 801 |
| Hodges | Sidehill | Brooklyn | 647 |

| *Number* | *balance* |
|---|---|
| 900 | 55 |
| 556 | 100000 |
| 647 | 105366 |
| 801 | 10533 |

Figure 2.5 A sample relational database

### 2.6.1 Normalization

Normalization is a process of simplifying the relationship between data elements in a record. Through normalization a collection of data in a record structure is

replaced by successive record structures that are simpler and more predictable and therefor more manageable. Normalization is carried out of four reasons:

- To structure the data so that any pertinent relationships between entities can be represented.
- To permit simple retrieval of data in response to query and report requests.
- To simplify the maintenance of the data through updates, insertions, and deletions
- To reduce the need to restructure or reorganize data when new application requirements arise. (9)

**First normal form:**

A relational table, by definition, is in first normal form. All values of the columns are atomic. That is, they contain no repeating values. Figure 2.6 shows the table FIRST in 1NF.

FIRST

| s# | status | city | p# | qty |
|---|---|---|---|---|
| s1 | 20 | London | p1 | 300 |
| s1 | 20 | London | p2 | 200 |
| s1 | 20 | London | p3 | 400 |
| s1 | 20 | London | p4 | 200 |
| s1 | 20 | London | p5 | 100 |
| s1 | 20 | London | p6 | 100 |
| s2 | 10 | Paris | p1 | 300 |
| s2 | 10 | Paris | p2 | 400 |
| s3 | 10 | Paris | p2 | 200 |
| s4 | 20 | London | p2 | 200 |
| s4 | 20 | London | p4 | 300 |
| s4 | 20 | London | p5 | 400 |

Figure 2.6 Table in 1NF

Although the table FIRST is in 1NF it contains redundant data. For example, information about the supplier's location and the location's status has to be repeated for every part supplied. Redundancy causes what are called update

**Second normal form:**

The definition of second normal form states that only tables with composite primary keys can be in 1NF but not in 2NF.

A relational table is in second normal form 2NF if it is in 1NF and every non-key column is fully dependent upon the primary key. That is, every non-key column must be dependent upon the entire primary key.

The process for transforming a 1NF table to 2NF is:

1. Identify any determinants, other the composite key, and the columns they determine.
2. Create and name a new table for each determinant and the unique columns it determines.
3. Move the determined columns from the original table to the new table. The determinate becomes the primary key of the new table.
4. Delete the columns you just moved from the original table except for the determinate, which will serve as a foreign key.
5. The original table may be renamed to maintain semantic meaning.

**SECOND**

| s# | status | city |
|---|---|---|
| s1 | 20 | London |
| s2 | 10 | Paris |
| s3 | 10 | Paris |
| s4 | 20 | London |
| s5 | 30 | Athens |

**PARTS**

| s# | p# | qty |
|---|---|---|
| s1 | p1 | 300 |
| s1 | p2 | 200 |
| s1 | p3 | 400 |
| s1 | p4 | 200 |
| s1 | p5 | 100 |
| s1 | p6 | 100 |
| s2 | p1 | 300 |
| s2 | p2 | 400 |
| s3 | p2 | 200 |
| s4 | p2 | 200 |
| s4 | p4 | 300 |
| s4 | p5 | 400 |

Figure 2.7 Table in 2NF

**Third normal form:**

The third normal form requires that all columns in a relational table are dependent only upon the primary key. A more formal definition is:

A relational table is in third normal form (3NF) if it is already in 2NF and every non-key column is non-transitively dependent upon its primary key. In other words, all non-key attributes are functionally dependent only upon the primary key.

The process of transforming a table into 3NF is:

1. Identify any determinants, other the primary key, and the columns they determine.
2. Create and name a new table for each determinant and the unique columns it determines.
3. Move the determined columns from the original table to the new table. The determinate becomes the primary key of the new table.
4. Delete the columns you just moved from the original table except for the determinate, which will serve as a foreign key.
5. The original table may be renamed to maintain semantic meaning. (11)

**SUPPLIER_CITY**

| s# | city |
|---|---|
| s1 | London |
| s2 | Paris |
| s3 | Paris |
| s4 | London |
| s5 | Athens |

**CITY_STATUS**

| city | status |
|---|---|
| London | 20 |
| Paris | 10 |
| Athens | 30 |
| Rome | 50 |

Figure 2.8 Table in 3NF

### 2.6.2 SQL

The SQL language has several parts:

- *Data definition language (DDL)*. The SQL DDL provides commands for defining relation schemes, deleting relations, creating indices, and modifying relation schemes.
- *Interactive data manipulation language (DML)*. The SQL DML includes a query language based on both the relational algebra and the tuple relational calculus. It includes also commands to insert, delete, and modify tuples in the database.
- *Embedded data manipulation language*. The embedded form of SQL is designed for use within general-purpose programming languages such as PL/I, Cobol, Pascal, Fortran, and C.
- *View definition*. The SQL DDL includes commands for defining views.
- *Authorization*. The SQL DDL includes commands for specifying access rights to relations and views.
- *Integrity*. The original System R Sequel language includes commands for specifying complex integrity constraints. Newer versions of SQL, including the ANSI standard, provide only a limited form of integrity checking. Future products and standards are likely to include enhanced features for integrity checking.
- *Transaction control*. SQL includes commands for specifying the beginning and ending of transactions. Several implementations, including IBM SAA-SQL, allow explicit locking of data for concurrency control. (8)

### 2.6.3 Mapping constraints

One important constraint is mapping cardinalities, which express the number of entities to which another entity can be associated via a relationship set.

For a binary relationship set R between entity sets A and B, the mapping cardinality must be one of the following:

- *One-to-one* : An entity in A is associated with at most one entity in B, and an entity in B is associated with at most on entity in A.
- *One-to-many* : An entity in A is associated with any number of entities in B. An entity in B, however, can be associated with at most one entity in A.
- *Many-to-one* : An entity in A is associated with at most one entity in B. An entity in B, however, can be associated with any number of entities in A.
- *Many-to-many* : An entity in A is associated with any number of entities in B, and an entity in B is associated with any number of entities in A. (8)

### 2.6.4 Keys

*Superkey* is a set of attributes that uniquely identifies each row in a relation.

*Key* is a minimal set of attributes that uniquely identifies each row in a relation. A key is a "minimal superkey"

*Composite key* is a key consisting of more than on attribute.

*Primary key* is the candidate key designated for principal use in uniquely identifying rows in a relation.

*Candidate key* is any set of attributes that could be chosen as a key of a relation.

*Foreign key* is a set of attributes in one relation that constitute a key in some other (or possibly the same) relation; used to indicate logical links between relations.

*Recursive foreign key* is a foreign key that references its own relation.

### 2.6.5 Integrity constraints

*Constraint* is a rule that restricts the values in a database:

- *Entity integrity rule*: No key attribute of any row in a relation may have a null value.
- *Referential integrity rule*: Every foreign key must either be null, or its value must be the actual value of a key in another relation.
- *Functional dependency*: The value of an attribute in a tuple determines the value of another attribute in the tuple. (5)

## 2.7 Data Flow Analysis

Data flow analysis examines the use of data to carry out specific business processes within the scope of a systems investigation. The components of data flow strategy span both requirements determination and system design. A prescribed notation facilitates the documentation of the current system and its analysis by all participants in the process of determining system requirement

**Tools of Data Flow strategy.**

1. Data Flow Diagram.

The graphic tool used to describe and analyze the movement of data through a system – manual or automated – including the processes, stores of data, and delays in the system.

2. Data Dictionary.

The logical characteristics of current systems data stores, including name, description, aliases, contents, and organization. Identifies processes where the data used and where immediate access to information is needed. Serves as the basic for identifying database requirements during system design.

3. Data Structure Diagram.

A pictorial description of the relation between entities (people, places, events, and things) in a system and the set of information about the entity. Does not deal with physical data store.

4. Structure Chart.

A design tool that pictorially shows the relation between processing modules in computer software. Describes the hierarchy of component modules and the data that are transmitted between them. Includes analysis of input-to-output transformations and analysis of transactions.

The first two are developed during requirements determination while the later two most useful during systems design. (9)

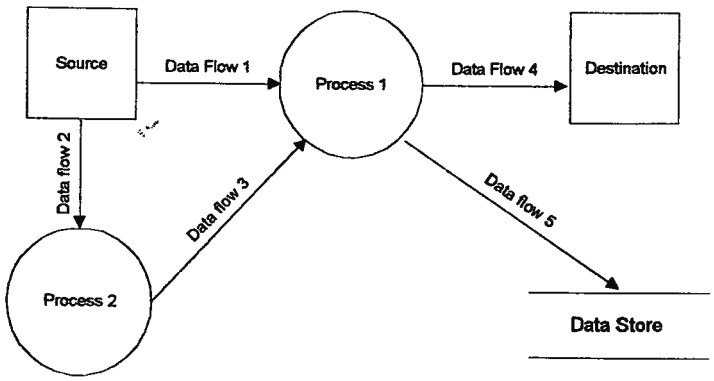



Figure 2.9 Data Flow Diagram using Yourdon notation

## 2.8 Testing

*Back-box testing* attempts to find errors in the following categories: 1) incorrect or missing functions, 2) interface errors, 3) errors in data structures or external database access, 4) performance errors, and 5) initialization and termination errors.

Back-box testing are designed to uncover errors in functional requirements without regard to the internal working of program.

*Boundary value analysis (BVA)* has been developed as a technique Boundary value analysis leads to a selection of test cases that exercise bounding values. Guidelines for BVA are as following

1. If an input condition specifies a range bounded by value a and b, test cases should be designed with values a and b, just above and just below a and b respectively.
2. If an input condition specifies a number of values, test cases should be developed that exercise the minimum and maximum numbers. Values just above and below minimum and maximum are also tested.
3. Guidelines 1 and 2 are applied to output conditions. For example, assume that a temperature vs. pressure table is required as output from an engineering analysis program. Test cased should be designed to create and output report that produces the maximum (and minimum) allowable number of table entries.

4. If internal program data structures have prescribed boundaries (e.g., an array has a defined limit of 100 entries), be certain to design a test case to exercise the data structure at its boundary. (13)

## 2.9 Software Development Tools

### 2.9.1 PowerBuilder

Client / Server Application Development Environment: PowerBuilder is a graphic PC-based client/server application development environment. It can develop front-end applications which access RDBMSs (Relational DataBase Management Systems) without coding in a 3GL (3rd Generation Language) such as C or C++.

4GL (4th Generation Language): In PowerBuilder you use screens, known as painters, to graphically put together the visual pieces of the application. Then it has code to this visual piece in a simple basic-like language, called PowerScript. Because it hides the complexities of a 3GL like this, it is known as a 4GL. (14)

***PowerBuilder's Technical Strengths***

*DataWindow:* It is the primary means by which a PowerBuilder application talks to the database. It has built-in features to format data for display, allow different edit-styles, validate data entered by user, generate appropriate SQL based on the changes made by a user and also the RDBMS it is talking to and scores of other such invaluable features.

*Object-Oriented (OO):* PowerBuilder is an object-oriented language. Though it is not a pure object-oriented language, it supports inheritance in most of the areas, permits encapsulation and enables polymorphism. Because of these reasons, it is

possible to architect your applications in such a way as to reuse code within and across applications. If you make use of OO features, it also makes it simpler to maintain that application.

*Native Drivers:* Though ODBC (Open Database Connectivity) is good for accessing multiple databases through a common gateway, it covers only the common minimum features of these databases. PowerBuilder provides native drivers for all the major RDBMSs, such as Oracle, Sybase, Informix, DB2, MS SQLServer, etc., so that you can take advantage of the power of these.

*Cross Platform:* You can write code once and run that application on all the flavors of Windows, namely Windows 3.1, Windows for workgroups, Win95 and Windows NT. You can also use the same code to run the application on Mac and Sun Solaris UNIX.

*Web-enabled:* With PowerBuilder 5.0 and the Internet add-ons, you can build an application, which can access data in an RDBMS through a browser, whether it is on the corporate intranet or on the internet. (15)

***Connecting to the database***

PB has two connections to the database.

1. While you are developing: You will notice that it takes longer when you first try to open the database, datawindow, query or pipeline painter. This is when PB is trying to connect to the database. The transaction object is populated (internally) with the values from the pb.ini file. Try to switch from one database to another ( for example, from the 'PowerSoft Demo DB' to the 'ABNC Main DB (v4)' ) and look closely at what is happening to

the values in the [database] section of your pb.ini file. One way of doing this switching is from File->Connect in the database painter.

2. While you run an application: In the application open event, you need code, prior to connect, which will specifically populate the transaction object from an application .ini file. (16)

### 2.9.2 Sybase

Sybase is an increasingly popular database. The databases can execute miscellaneous transaction processing logic. Data consistency and integrity checks don't have to be coded into each different client application. Instead it can be enforced centrally in the server. The feature of Sybase is follow:

- Run on a wide range of computer architectures such as Sun's Solaris, DEC's VMS, IBM's AIX, HP's HPUX, IBM's OS2, Novell's NLM and Windows NT.
- Support variety of networking protocols. These networking protocols such as TCP/IP, Decnet, Novell IPX, and LAN Manager.
- Programs supplied by independent software vendors, or user written programs.
- Support Client/Server architecture and exhibits a truly independent client/server architecture in its ability for each node of the network to support an independently functioning client and/or server which communicates with the other nodes in the network via SQL calls.
- Other Sybase produces called gateways to give clients and servers on line access to non-Sybase data.

- Transact-SQL is Sybase's extension to SQL. Transact-SQL permits procedural processing with control statements.
- Sybase servers also include the ability to communicate directly between each other without going through a client. The inter-server communication is implemented as an easy to write procedure call. RPC's can be called in stored procedure or triggers so that the client program doesn't have to know or care about other sources of data. (17,18,21)

### 2.9.3 Windows NT

Windows NT optimizations include:

- Support for multitasking, multithreading and multiprocessing.
- Has the potential of being the best of two words: a popular graphical environment for end users and a full-fledged 32-bit operating system.
- Using Microsoft's advanced operating system technologies, such as preemptive multitasking, to optimize resource utilization.
- Support connection to a variety of operating systems such as DOS, OS/2, Windows 3.X, Windows 95, Windows 98, UNIX and so on. (17,22,23)

## 2.10 Related Research

Usa Jugtrimongkol (19) did information system for government inventory management: A case study of Land Development Department. The information system consists of three major processes: 1) Data manipulation: Manipulate relative data; 2) Transaction processing: Support operation management, which are registration, distribution, borrowing, return, repair and disposal; 3) Query and report: Process pre-

defined outputs and preplanned printed reports, and works on client/server network. The information system was designed and developed using data flow analysis, relational database (Sybase SQL anywhere 5.0 as DBMS) and Object-Oriented programming (Powersoft Powerbuilder Enterprise Version 6.0).

Kristi Yuthas and Scott T. Young (20) researched material matters: assessing the effectiveness of materials management IS. This research is designed to enhance decision-making performance by lowering costs, increasing turnover, and improving service. The most direct approach is to assess the effects of the system on materials management performance outcomes such as inventory costs, turnover, and fill rates. The more common approach is to assess effectiveness via substitute measures, such as user perceptions and usage statistics. A laboratory experiment was conducted to examine the relationships between materials management performance, user satisfaction, and system usage.

# CHAPTER III

# MATERIALS AND METHODS

## 3.1 Materials

- Hardware
  - Computer (Minimum)

    Server: Intel Pentium III 667 MHz

    RAM 64 MB

    Hard disk 1.2 GB

    Client: Intel Pentium 166 MHz

    RAM 32 MB

    Hard disk 1.2 GB
  - Communication equipment: LAN card, Wire, Hub
  - Printer.
- Software

  Database : Sybase SQL anywhere 5.0

  Application development tools: Powersoft PowerBuilder Enterprise Ver. 6.0

  Operating System : Windows NT 4.0, Windows 95 or Windows 98

  Help authoring tools : HelpScribble 5.0.0

## 3.2 Methods

1. Study operation of stock management system

   1.1. Review data of seed and seedling management system from books and researches such as steps of operation, function and seed and seedling details.

   1.2. Identify problems of seed and seedling management system.

   1.3. Define problem solving by information technology.

2. Collect data and identify system requirement of Land Development Department

   2.1. Review existing system documents. General document should be reviewed first. General documentation includes management overviews, description of seed and seedling features, the organization's structure, and functional task. List of the existing system documents are as follows:

   - Organization chart
   - Division and section details
   - Seed and seedling document: name, kind of seed, quantity/unit, harvest age and so on
   - Distribution document
   - The money receiving form
   - Summary of seed distribution and receiving
   - Summary of seed production and providing

   2.2. Observe in operating environment, it can indicate how individuals participate in the operation of system. Interview users and related officers for the requirements and operation problems.

3. Analyze and design system

Use data flow analysis to analyze the system. The components of data flow strategy span both requirement determination and system design. Following the flow of data through processes tells how data are input, processed, stored, retrieved, used, changed and output.

3.1 Design of input and output

Design input/output screens. These screens are user-friendly with GUI (graphic user interface). The input screens must have input controls, which provides way to ensure that 1) only authorize users can access the system, 2) guarantee the transactions are acceptable, 3) the data are validated for accuracy, and 4) There is the determination of any necessary data have been omitted.

Design output reports, which support the activity of the operation and management. The outputs are generated in tubular form.

3.2 Design of procedure

Distinguish between manual and computer processes. The computer processes are to identify constraints such as authorization, volume, process timing, input method, output method and so on.

Design procedure and process specification. Formal statements using techniques and languages that enable analysts to describe important activities that make up the system

3.3 Design of database

Identify data used and information produced.

Select and identify data from collected data and system requirement, it gets related entities, attributes and constraints.

Identify relationships between those entities and design database with relational database concept.

4. Develop system

4.1. Create database. Transform designed data structure diagram to database.

4.2. Develop application to connect database and support operations. This step has unit testing.

4.3. Create help file.

5. Test software

Verify and validate software.

Check if the software conforms to its specification and if the software meets the expectation of the software requirement by black-box testing. Uncover errors and ensure that the software performs properly. Test cases are designed by boundary value analysis method.

6. Summarize and create document

This document includes detail, methodology, conclusion and recommendation of the research.

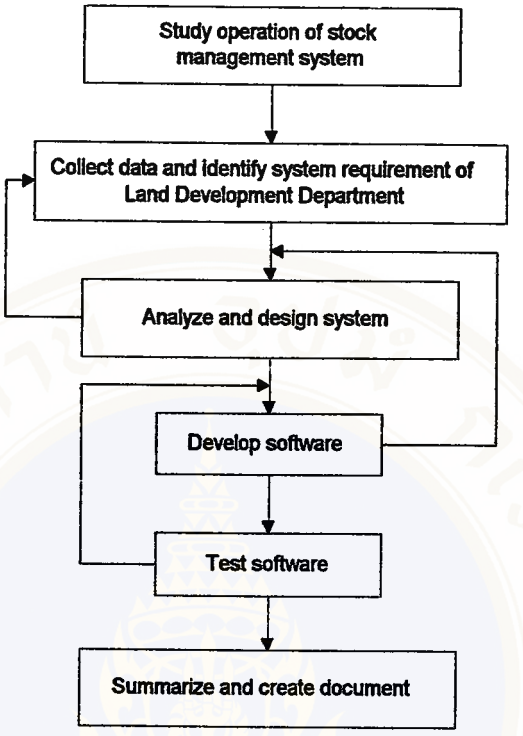



Figure 3.1 Steps and research methodology

# CHAPTER IV

# RESULTS

The result of study for this research is the information system of seed and seedling management for soil and water conservation: a case study of Land Development Department. The information system is designed and developed using data flow analysis, relational database and Object-Oriented programming.

## 4.1 Current System

The combination of step 1 and 2 in chapter 3 produces current seed and seedling management system in graph presentation style, which may be more effective to describing the system since the standard symbols, may limit communication.

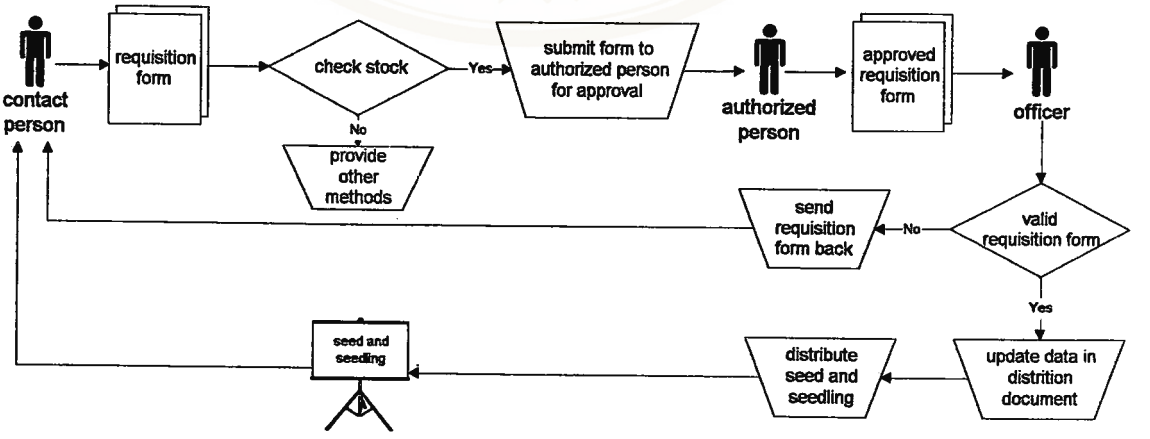



Figure 4.1 Graphic presentation of current distribution process

Figure 4.1 illustrates distribution process. Contact person fills him/her requirements into requisition form. Next, auditor will check stock. If stock has got

enough seed and seedling for distribution, requisition form is submitted to authorized person for approval. After requisition form has been approved and sent to officer, he/she will audit form, update data into distribution document, and distribute seed and seedling. But if numbers on hand are not enough, head of section will provide other methods for distribution.

Transfer process (Figure 4.2) is the same as distribution process but it has to contact destination stock for checking quantity of seed and seedling and update data in both stocks (start and destination stock).

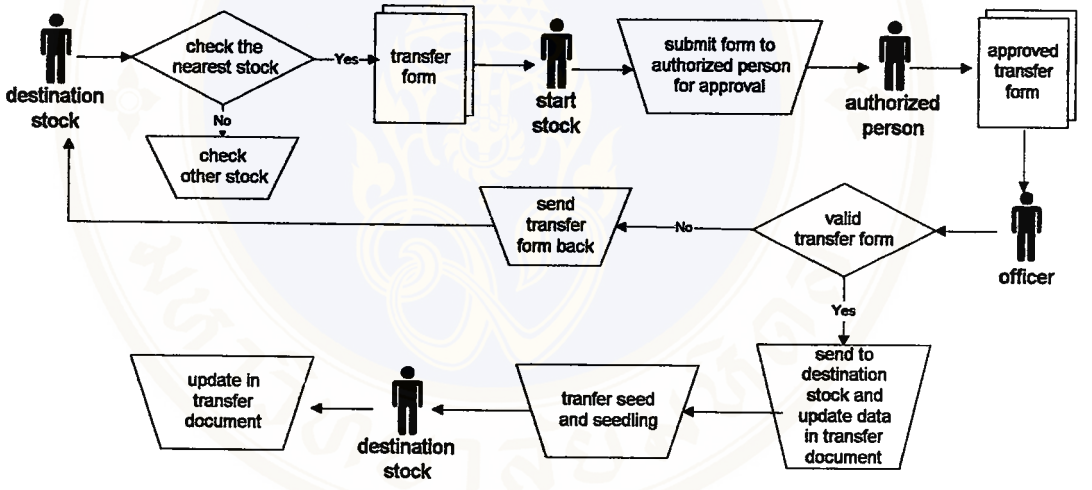



Figure 4.2 Graphic presentation of current transfer process

Figure 4.3 describes purchase process. Auditor checks quantity of seed and seedling in stock and generates report to purchasing department. If numbers on hand are very low, purchasing department will submit purchase form to authorized person for approval. Next, approval purchase form is sent to officer, he/she will audit form, update data in purchase document and send it to vendor. Vendor will provide and consign seed and seedling to consignee (person does duty for receiving seed and seedling).

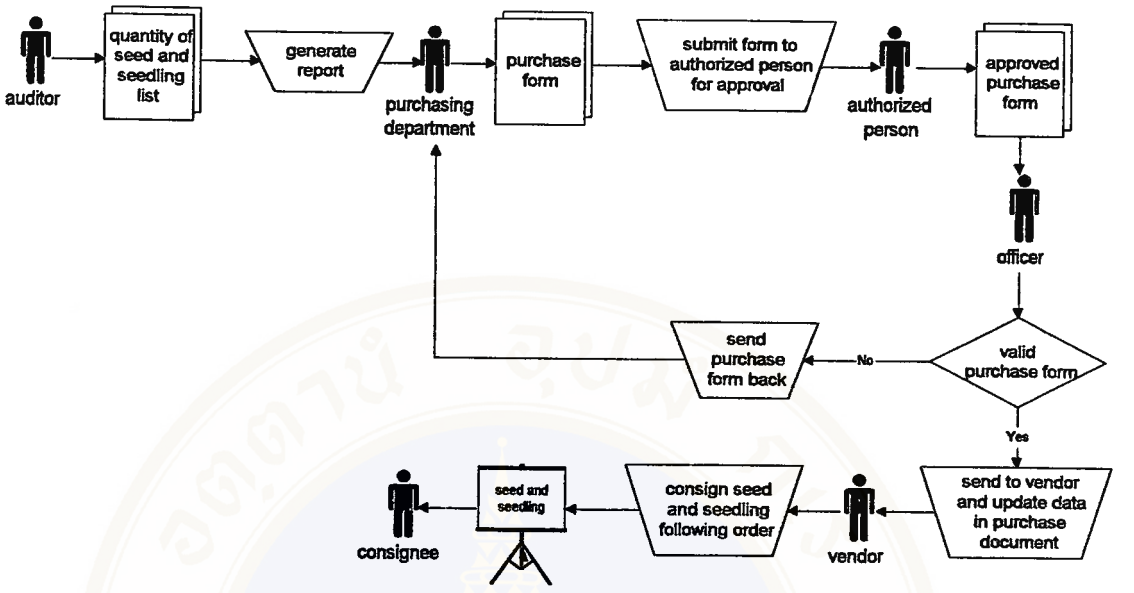



Figure 4.3 Graphic presentation of current purchase process

Figure 4.4 shows production process. Responsible officer will submit production form to authorized person for approval. Next, approval production form is sent to officer, he/she will audit form, update data in production document and distribute seed and seedling to agriculturist for growing.

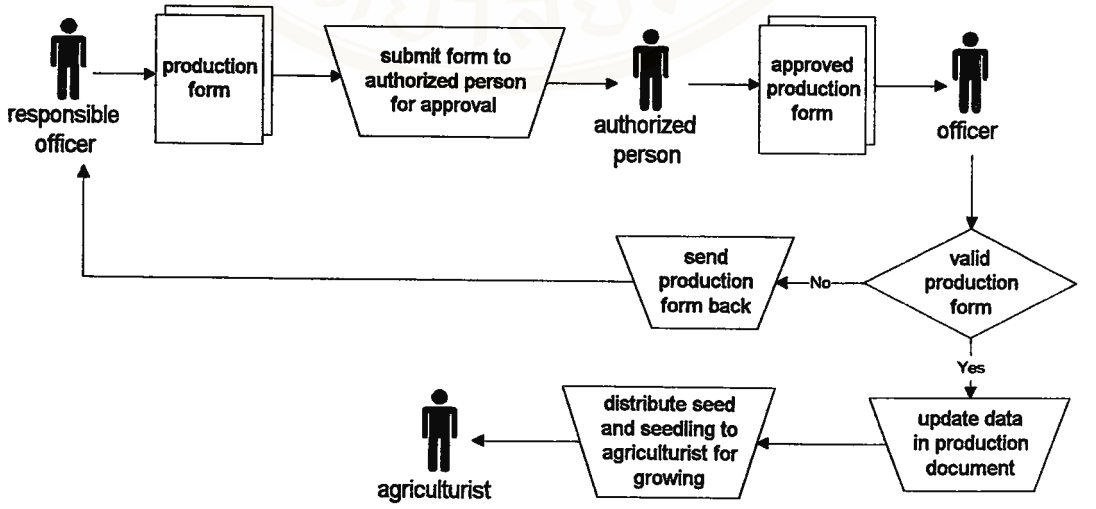



Figure 4.4 Graphic presentation of current production process

When consignor submits seed and seedling to organization, consignee will check those seed and seedling. If it valid with existent document, he/she will generate report to officer for updating data in receiving document and storing seed and seedling. But if it is not accurate, he/she will return it or make report about faults of consignment as shown in Figure 4.5

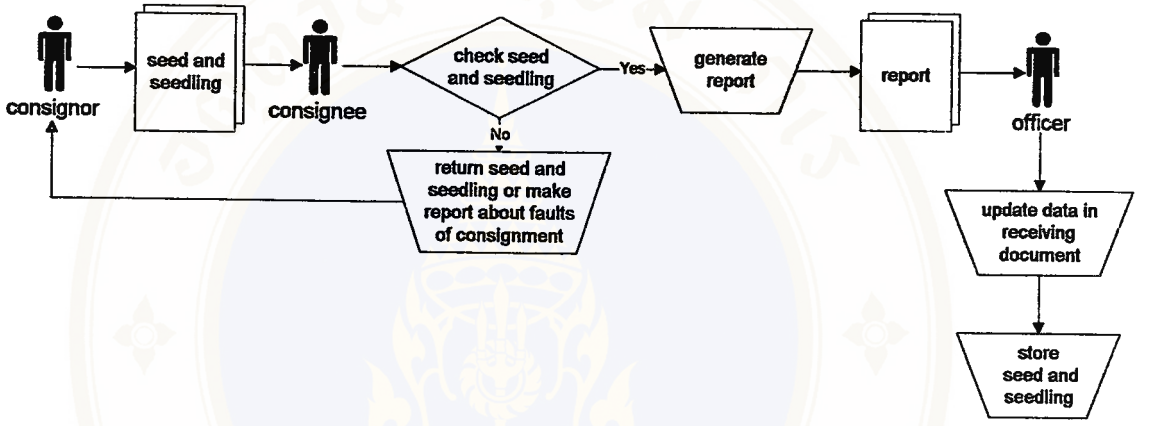



Figure 4.5 Graphic presentation of current receiving process

## 4.2 Results from System Analysis and Design

The results from system analysis and design consist of data flow diagram (describes and analyzes the movement of data through system), data dictionary (defines systems elements), data structure diagram (describes relation between entities), structure charts, input, output and program specification.

### 4.2.1 Data Flow Diagram

In Figure 4.6, The presentation graphs of current system are converted to new system, which consists of the following entities: section, head of section, officer, purchasing department

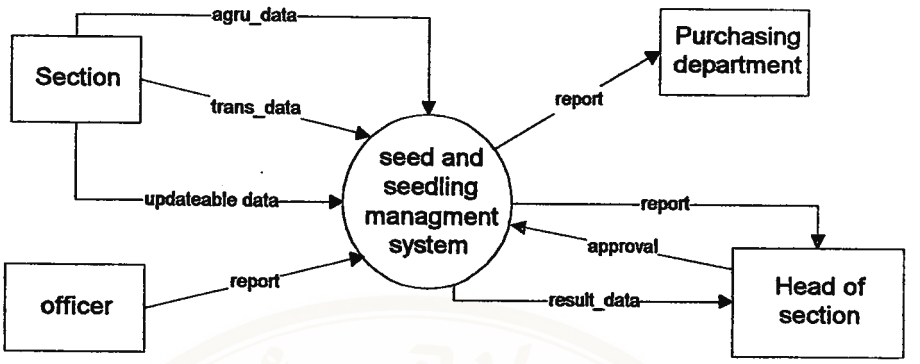



Figure 4.6 Context diagram of seed and seedling management system

The new system revealed 4 major processes: data manipulation, transaction process, queries process and report generation. The overall system, consisting of these processes, is shown in Figure 4.7

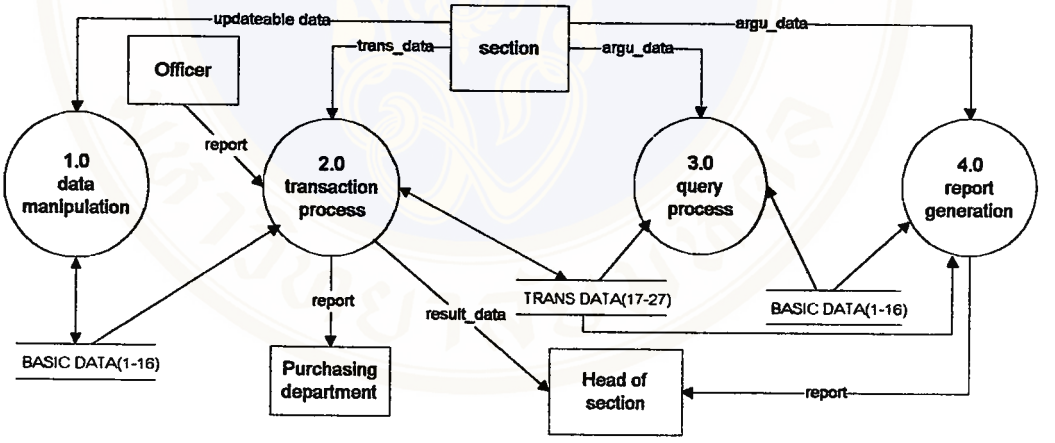



Figure 4.7 DFD Level 1: Seed and seedling management system

This data flow analysis also shows that the system use 27 data stores, which are divided into 2 categories:

1. "Basic data": It consists of 16 data stores, which are PLANT, KIND, DIVISION, SECTION, EMPLOYEE, PERSON, PERTIT, POSITION, STATUS, AUTHORITY, REGION, PROVINCE, AMPHOE, VENDOR,

GRUSER, and USERDATA. These data stores are not usually modified. The system fetch read-only data to display and reference.

2. "Trans data": It consists of 11 data stores, which are STOCK, STOCK_DETAIL, DISTRIBUTION, DISTRIBUTION_DETAIL, TRANSFER, TRANSFER_DETAIL, PURCHASING, PURCHASING_DETAIL, PRODUCTION, PRODUCTION_DETAIL and NET_QUANTITY. These data stores are modified in each transaction process( distribution, transfer, purchase, production and receiving process)

The details of "basic data" and "transaction data" data stores are described in section 4.2.4. Database Design. The system is explained in detail through the lower-level subsystem diagram, that developed to understand the activities associated with the system.

**Level 2: 1. Data Manipulation**

Data manipulation consists of the activities of adding, deleting and editing data. These activities manipulate data in "Basic data" data store. Some data store has one-to-many relationship with another one consequently; its activities have a little different detail.

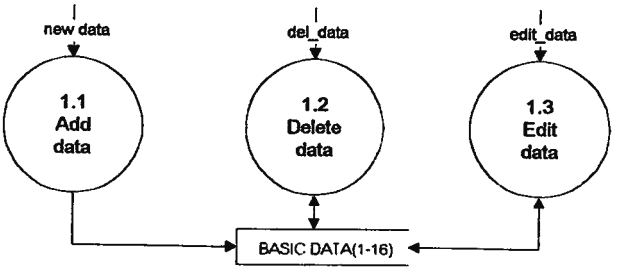



Figure 4.8 DFD Level 2: 1 Data manipulation

The downward explosion of the data manipulation process maintains the numbering scheme introduced at the system level 3 in Figure 4.9.

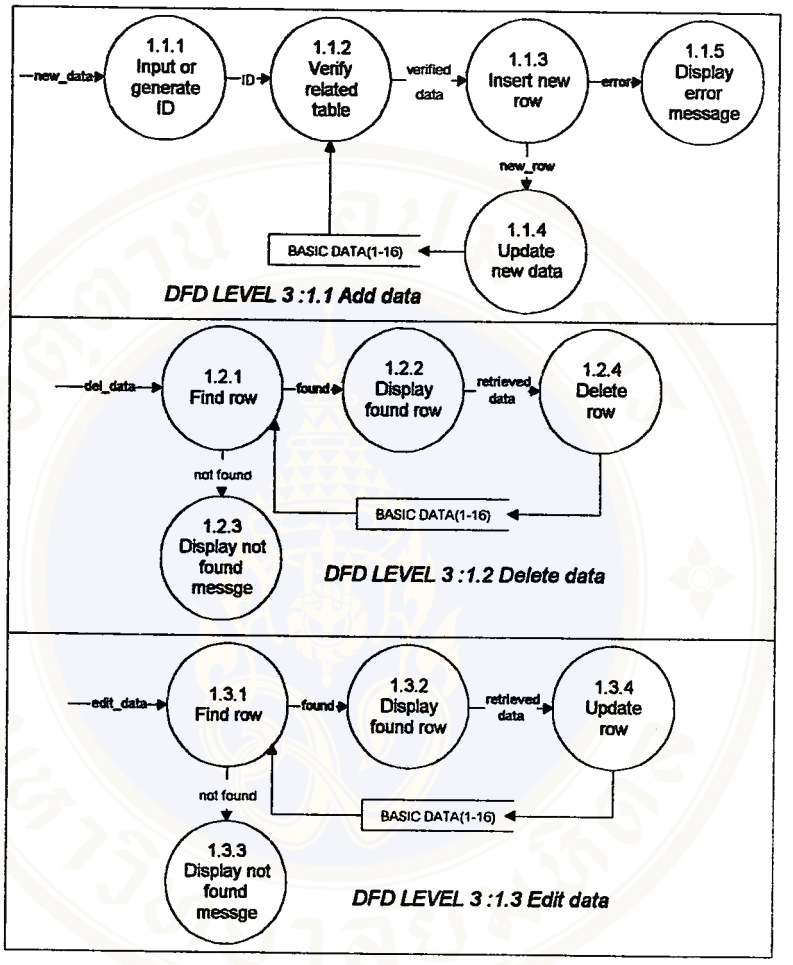



Figure 4.9 DFD Level 3: Data manipulation

**Level 2: 2. Transaction Process**

Transaction process consists of 5 activities: "distribute seed and seedling", "transfer seed and seedling", "purchase seed and seedling", "produce seed and seedling" and "receive seed and seedling", which are shown in Figure 4.10. In Figure 4.7 DFD Level 1 mentions that the activities use "Basic data" and "Trans data", but it does not explain. This level explains details of related data stores in each activity. In addition, DFD level 3 are show in Figure 4.11-4.12 to clarify the activities.

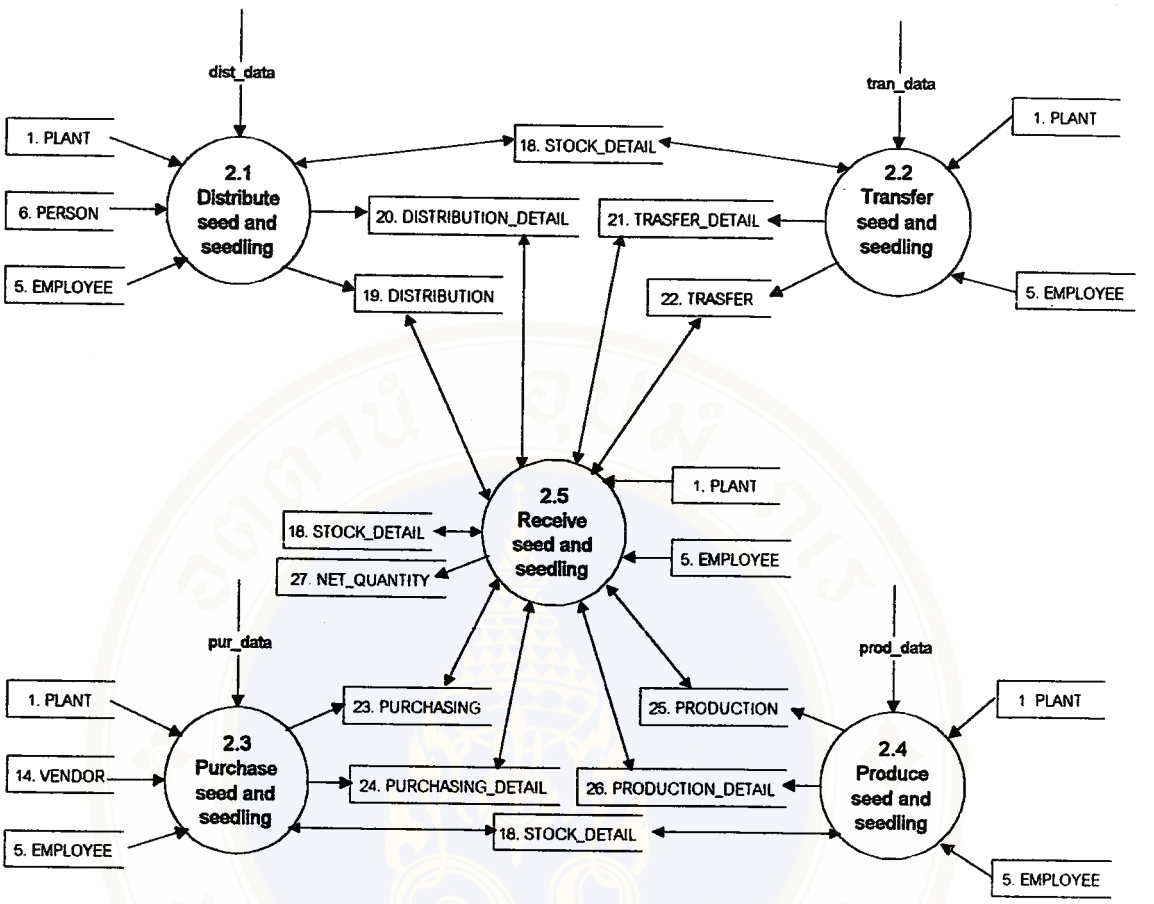



Figure 4.10 DFD Level 2: 2. Transaction process

"Distribute seed and seedling" activity involves 5 subactivities. "Enter distribution data" subactivity gets one of important data is PLANT_ID, which is verified in 2 subactivities, "verify PLANT_ID": to ensure that there is the PLANT_ID in data store and "verify distribute data": to ensure that the seed and seedling can be distributed (quantity of seed and seedling in stock are enough).

When the PLANT_ID pass the verification "update quantity" subactivity changes quantity value in STOCK_DETAIL to increasing or decreasing. "Generate distribution document" subactivity transform data into printed report. Finally, "add distribution data" subactivity stores distribution data into DISTRIBUTION and

DISTRIBUTION_DETAIL data store. Other activities in transaction process are similar to "distribute seed and seedling" activity, hence their description are skipped

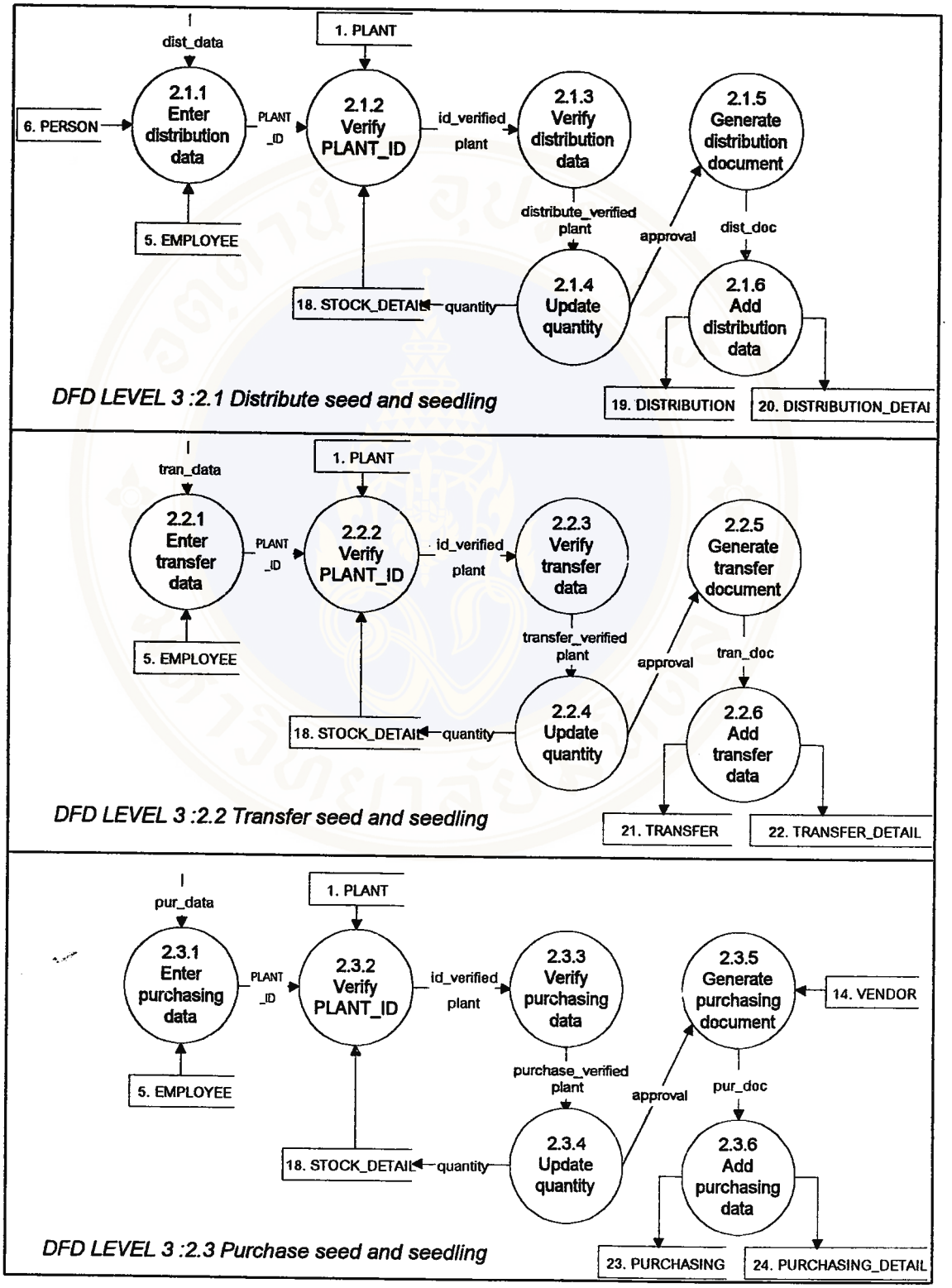



Figure 4.11 DFD Level 3: Transaction process

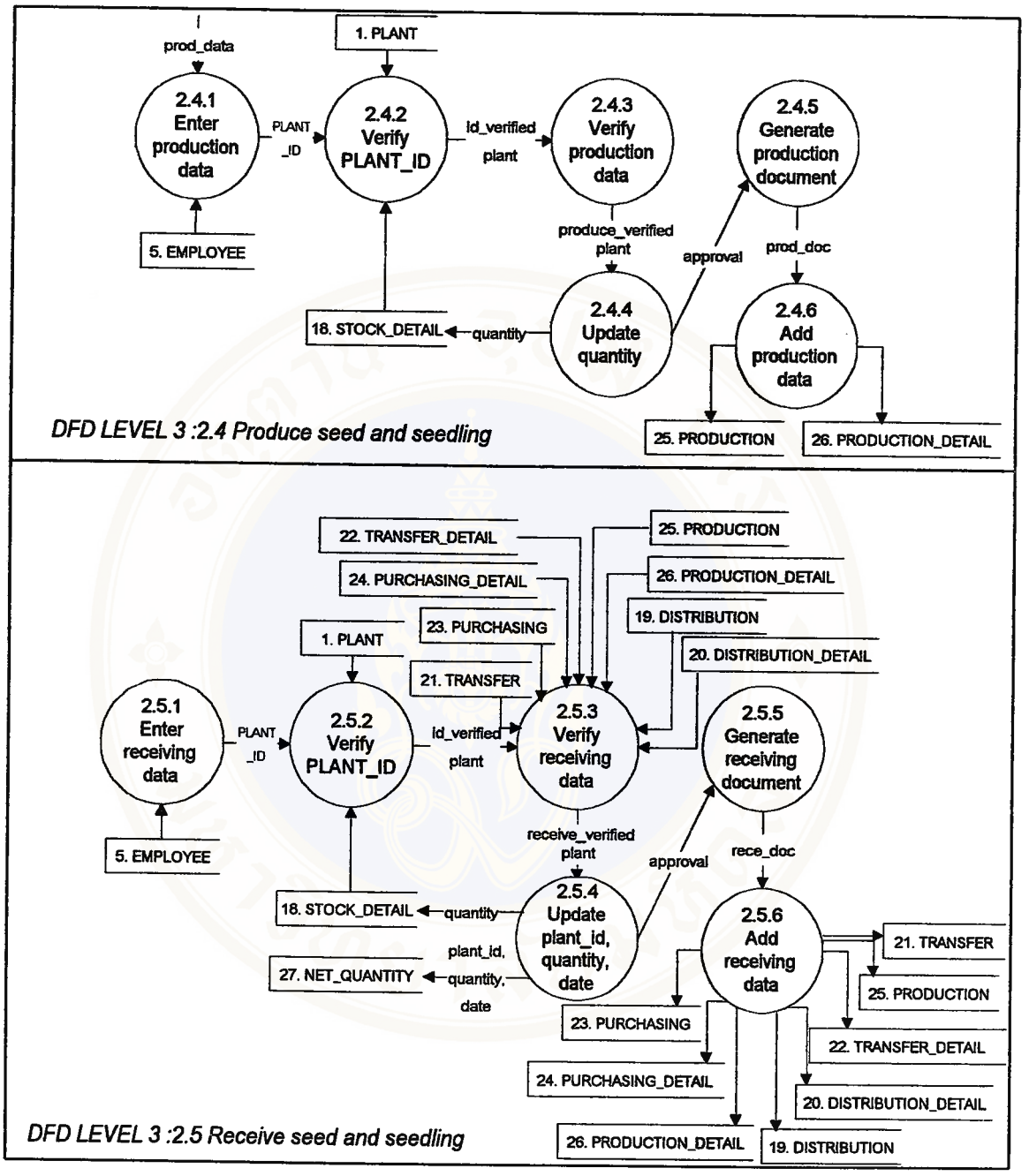



Figure 4.12 DFD Level 3: Transaction process (continued)

**Level 2: 3. Query process**

Query process consists of 4 activities, which use all data stores to process data. Result data has 2 types: tabular and graph, which have individual processing. See list of result data in section 4.2.2 data dictionary. "Process data" activity is calculation, summary and comparative analysis.

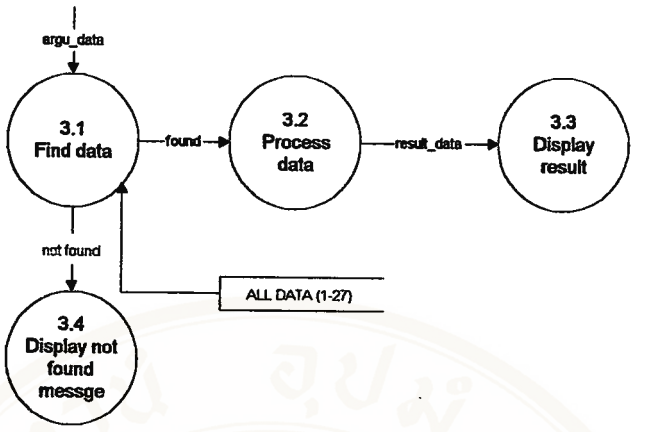



Figure 4.13 DFD Level 2: 3. Query Process

**Level 2: 4. Report generation**

Report generation consists of 4 activities, which use all data stores to process printable reports. See list of printable reports in 4.2.2 data dictionary. " Process data" activity is calculation and summary data.

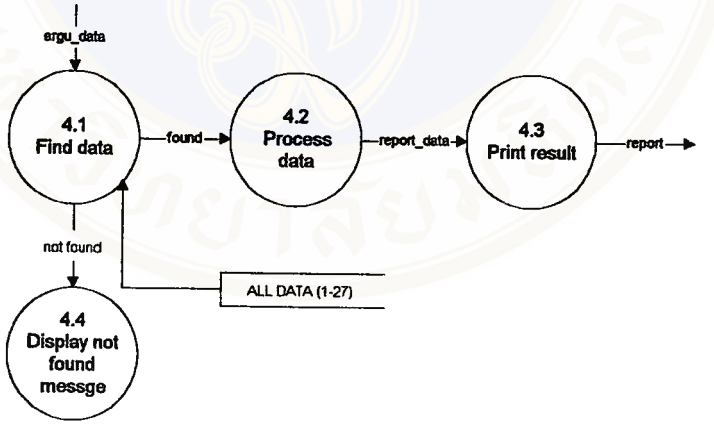



Figure 4.14 DFD Level 2: 4. Report generation

### 4.2.2 Data Dictionary

The data dictionary elements developed for seed and seedling management system consists of process, procedure, data flow and data structure. The process of

developing the data dictionary itself forces analysis to clarity their understanding of the data in the system

Figure 4.15 identifies each of the processes included in the DFD level 1. Each process is briefly described to clarify its purpose.

| | |
|---|---|
| **PROCESS NAME:** | 1. Data manipulation |
| **DESCRIPTION:** | Data of "BASIC DATA" is added, edited or deleted and stored in database. |
| **INBOUND DATA FLOWS:** | updateable_data |
| **OUTBOUND DATA FLOWS:** | data of "BASIC DATA" data store |
| **PROCESS NAME:** | 2.Transaction process |
| **DESCRIPTION:** | Distribution, transfer, purchase, production and receiving activities will be recorded in "TRANS DATA" data store. |
| **INBOUND DATA FLOWS:** | trans_data, report |
| **OUTBOUND DATA FLOWS:** | data of "TRANS DATA" data store, report and result_data |
| **PROCESS NAME:** | 3. Query process |
| **DESCRIPTION:** | User can select question to query by identifying argument, system will process result data in graph or tabular format of display screen. |
| **INBOUND DATA FLOWS:** | argu_data<br>data of "BASIC DATA" data store<br>data of "TRANS DATA" data store |
| **OUTBOUND DATA FLOWS:** | |
| **PROCESS NAME:** | 4.Report generation |
| **DESCRIPTION:** | User can select question to generate report by identifying argument, system will process printed report. |
| **INBOUND DATA FLOWS:** | argu_data<br>data of "BASIC DATA" data store<br>data of "TRANS DATA" data store |
| **OUTBOUND DATA FLOWS:** | report |

Figure 4.15 First level process descriptions

Figure 4.16 through 4.18 identify processes that take place in the DFD level 3 of transaction process. The processing logic is summarized. Some process has alias name and procedure. The logic summary of procedure is shown in Figure 4.19.

**PROCESS NAME:** 2.1.1 Enter distribution data
**LOGIC SUMMARY:**
SELECT max(convert(numeric,dist_id)) INTO :ls_main_id
FROM DISTRIBUTION WHERE (distribution.div_id =
div_id) and (distribution.sect_id = sect_id);
ii_main_id = integer(ls_main_id)+1
dist_id = string(ii_main_id)

READ date1 FROM edit box
READ stock_id, emp_id1 and per_id FROM list box
REPEAT
  READ plant_id FROM list box
  READ quan FROM edit box.
UNTIL END OF DISTRIBUTION

**PROCESS NAME:** 2.1.2 Verify PLANT_ID
**ALAIS:** 2.2.2
**DESCRIPTION:** To ensure that there is PLANT_ID in system
**LOGIC SUMMARY:**
READ quan FROM edit box

SELECT quan2, quan3, quan4, quan5 INTO
:ir_stock_quan2, :ir_stock_quan3, :ir_stock_quan4,
:ir_stock_quan5 FROM STOCK_DETAIL WHERE
(stock_detail.div_id = div_id) and (stock_detail.sect_id =
sect_id) and (stock_detail.stock_id = stock_id) and
(stock_detail.plant_id = plant_id);

gr_stock_quan5 = ir_stock_quan5 - ir_stock_quan2 -
ir_stock_quan3 - ir_stock_quan4

IF (quan <= gr_stock_quan5) THEN
  pass=TRUE
END IF

**PROCESS NAME:** 2.1.3 Verify distribution data
**ALAIS:** 2.2.3
**DESCRIPTION:** To ensure about the quantity of seed and seedling in stock
**LOGIC SUMMARY:**
IF (quan <= gr_stock_quan5) THEN
  gr_stock_quan5 = gr_stock_quan5 - quan
END IF

**PROCESS NAME:** 2.1.4 Update quantity
**LOGIC SUMMARY:**
UPDATE DISTRIBUTION_DETAIL SET chk = 'y'
WHERE (distribution_detail.dist_id = dist_id) and
(distribution_detail.div_id = div_id) and
(distribution_detail.sect_id = sect_id);

UPDATE STOCK_DETAIL SET quan5 = :gr_stock_quan5
+ :ir_stock_quan2 + :ir_stock_quan3 + :ir_stock_quan4
WHERE (stock_detail.div_id = div_id) and
(stock_detail.sect_id = sect_id) and (stock_detail.stock_id
= stock_id) and (stock_detail.plant_id = plant_id);

**PROCESS NAME:** 2.1.5 Generate distribution document
**LOGIC SUMMARY:**
SELECT * FROM DISTRIBUTION,
DISTRIBUTION_DETAIL WHERE
(distribution.div_id = div_id) and
(distribution.sect_id = sect_id) and
(distribution.dist_id = no);
Arrange data to distribution Form
Print

**PROCESS NAME:** 2.1.6 Add distribution data
**LOGIC SUMMARY:**
ADD NEW RECORD IN DISTRIBUTION AND
DISTRIBUTION_DETAIL

**PROCESS NAME:** 2.2.1 Enter transfer data
**LOGIC SUMMARY:**
SELECT max(convert(numeric,transfer_id)) INTO
:ls_main_id FROM TRANSFER WHERE
(transfer.div_id = div_id) and (transfer.sect_id = sect_id);
ii_main_id = integer(ls_main_id)+1
transfer_id = string(ii_main_id)

READ date1 FROM edit box
READ stock_id1, stock_id2 and emp_id1 FROM list box
REPEAT
  READ plant_id FROM list box
  READ quan FROM edit box.
UNTIL END OF TRANSFER

**PROCESS NAME:** 2.2.4 Update quantity
**LOGIC SUMMARY:**
UPDATE STOCK_DETAIL SET quan5 = :gr_stock_quan5
+ :ir_stock_quan2 + :ir_stock_quan3 + :ir_stock_quan4
WHERE (stock_detail.div_id = div_id) and
(stock_detail.sect_id = sect_id) and
(stock_detail.stock_id = stock_id) and
(stock_detail.plant_id = plant_id);

UPDATE TRANSFER_DETAIL SET chk = 'y'
WHERE (transfer_detail.transfer_id = transfer_id) and
(transfer_detail.div_id = div_id) and
(transfer_detail.sect_id = sect_id);

**PROCESS NAME:** 2.3.1 Enter purchasing data
**LOGIC SUMMARY:**
SELECT max(convert(numeric,purchase_id)) INTO
:ls_main_id FROM PURCHASING WHERE
(purchase.div_id = div_id) and (purchase.sect_id =
sect_id);
ii_main_id = integer(ls_main_id)+1
purchase_id = string(ii_main_id)
READ date1 FROM edit box
READ stock_id, vendor_id and emp_id1 FROM list box
REPEAT
  READ plant_id FROM list box OR *CALL* Plant_search
  READ quan and price_unit FROM edit box.
UNTIL END OF PURCHASE

**PROCESS NAME:** 2.2.5 Generate transfer document
**LOGIC SUMMARY:**
SELECT * FROM TRANSFER, TRANSFER_DETAIL
WHERE (transfer.div_id = div_id) and
(transfer.sect_id = sect_id) and (transfer.dist_id = no);
Arrange data to transfer Form
Print

**PROCESS NAME:** 2.2.6 Add transfer data
**LOGIC SUMMARY:**
ADD NEW RECORD IN TRANSFER AND
TRANSFER_DETAIL

Figure 4.16 Third level process descriptions

**PROCESS NAME:** 2.3.2 Verify PLANT_ID
**ALAIS:** 2.4.2
**DESCRIPTION:** To ensure that there is PLANT_ID in system
**LOGIC SUMMARY:**
SELECT quan1, quan2, quan3, quan4 INTO :ir_stock_quan1, :ir_stock_quan2, :ir_stock_quan3, :ir_stock_quan4 FROM STOCK_DETAIL WHERE (stock_detail.div_id = div_id) and (stock_detail.sect_id = sect_id) and (stock_detail.stock_id = stock_id) and (stock_detail.plant_id = plant_id);

IF (ir_stock_quan1 <= ir_stock_quan2 + ir_stock_quan3 + ir_stock_quan4) THEN
pass=TRUE
END IF

**PROCESS NAME:** 2.3.4 Update quantity
**LOGIC SUMMARY:**
UPDATE PURCHASING_DETAIL SET chk = 'y' WHERE (purchasing_detail.purchase_id = purchase_id) and (purchasing_detail.div_id = div_id) and (purchasing_detail.sect_id = sect_id);

**PROCESS NAME:** 2.4.1 Enter production data
**LOGIC SUMMARY:**
SELECT max(convert(numeric,produce_id)) INTO :ls_main_id FROM PRODUCTION WHERE (produce.div_id = div_id) and (produce.sect_id = sect_id);
ii_main_id = integer(ls_main_id)+1
produce_id = string(ii_main_id)

READ date1 FROM edit box
READ stock_id and emp_id1 FROM list box
REPEAT
READ plant_id FROM list box OR *CALL* Plant_search
READ quan FROM edit box.
UNTIL END OF PRODUCTION

**PROCESS NAME:** 2.4.6 Add production data
**LOGIC SUMMARY:**
ADD NEW RECORD IN PRODUCTION AND PRODUCTION_DETAIL

**PROCESS NAME:** 2.5.1 Enter receiving data
**LOGIC SUMMARY:**
READ no FROM value in w_trans_q_no
dw_1.Retrieve(no, div_id, sect_id)

READ date2 FROM edit box
READ emp_id2, emp_id3(receive by transfer) FROM list box
dw_2.Retrieve(no, div_id, sect_id)

**PROCESS NAME:** 2.5.3 Verify receiving data
**DESCRIPTION:** To ensure about the quantity of seed and seedling in stock
**LOGIC SUMMARY:**
READ plant_id and quan from each form (distribution, transfer, purchase and production)

ir_stock_quan1 = ir_stock_quan1 + quan (by purchasing and production)
ir_stock_quan1 = ir_stock_quan1 - quan (by distribution and transfer)

**PROCESS NAME:** 2.3.3 Verify purchasing data
**ALAIS:** 2.4.3
**DESCRIPTION:** To ensure about the quantity of seed and seedling in stock
**LOGIC SUMMARY:**
READ quan FROM edit box

ir_stock_quan1 = ir_stock_quan1 + quan

**PROCESS NAME:** 2.3.5 Generate purchasing document
**LOGIC SUMMARY:**
SELECT * FROM PURCHASING, PURCHASING_DETAIL WHERE (purchasing.div_id = div_id) and (purchasing.sect_id = sect_id) and (purchasing.purchase_id = no);
Arrange data to purchasing Form
Print

**PROCESS NAME:** 2.3.6 Add purchasing data
**LOGIC SUMMARY:**
ADD NEW RECORD IN PURCHASING AND PURCHASING_DETAIL

**PROCESS NAME:** 2.4.4 Update quantity
**LOGIC SUMMARY:**
UPDATE PRODUCTION_DETAIL SET chk = 'y' WHERE (production_detail.produce_id = production_id) and (production_detail.div_id = div_id) and (production_detail.sect_id = sect_id);

**PROCESS NAME:** 2.4.5 Generate production document
**LOGIC SUMMARY:**
SELECT * FROM PRODUCTION, PRODUCTION_DETAIL WHERE (production.div_id = div_id) and (production.sect_id = sect_id) and (production.produce_id = no);
Arrange data to production Form
Print

**PROCESS NAME:** 2.5.2 Verify PLANT_ID
**DESCRIPTION:** To ensure that there is PLANT_ID in system
**LOGIC SUMMARY:**
SELECT quan1 INTO :ir_stock_quan1 FROM STOCK_DETAIL WHERE (stock_detail.div_id = div_id) and (stock_detail.sect_id = sect_id) and (stock_detail.stock_id = stock_id) and (stock_detail.plant_id = plant_id);

**PROCESS NAME:** 2.5.4 Update plant_id, quantity, date
**LOGIC SUMMARY:**
UPDATE STOCK_DETAIL SET quan1 = :ir_stock_quan1 WHERE (stock_detail.div_id = div_id) and (stock_detail.sect_id = sect_id) and (stock_detail.stock_id = stock_id) and (stock_detail.plant_id = plant_id);

SELECT count(*) into :i_rc FROM net_quantity;

INSERT VALUES INTO NET_QUANTITY

UPDATE DISTRIBUTION_DETAIL, TRANSFER_DETAIL, PURCHASING_DETAIL, PRODUCTION_DETAIL SET chk = 'n'

Figure 4.17 Third level process description (continued)

| | |
|---|---|
| **PROCESS NAME:** 2.4.5 Generate receiving document<br>**LOGIC SUMMARY:**<br>Receive by distribution involve in process 2.1.5<br>Receive by transfer involve in process 2.2.5<br>Receive by purchase involve in process 2.3.5<br>Receive by production involve in process 2.4.5 | **PROCESS NAME:** 2.4.6 Add receiving data<br>**LOGIC SUMMARY:**<br>ADD NEW RECORD IN DISTRIBUTION, DISTRIBUTION_DETAIL, TRANSFER, TRANSFER_DETAIL, PURCHASING, PURCHASING_DETAIL, PRODUCTION AND PRODUCTION_DETAIL |

Figure 4.18 Third level process description (continued)

```
PROCEDURE:Plant_search
LOGIC SUMMARY:
SELECT local_name FROM PLANT

DISPLAY local_name  in list box
IF local_name in list box is clicked THEN
   READ selected_name FROM list box
   SELECT plant_id FROM PLANT WHERE
   plant.local_name = selected_name
   RETURN plant_id
END IF
```

Figure 4.19 Logic summary of procedure

The other processes in data manipulation, query process and report generation are not included in process description because the low-level of its DFD are sufficient explanation.

Figure 4.20 through 4.22 define 35 data flows that are included in the system. The processes that data flows originate from and into are also identified by number and name. Each data flow is also described by the data structures it contains. Unnamed data flows mean all data of data stores, which see detail in section 4.2.4 Database design.

**DATA FLOW NAME:** approval
**DESCRIPTION:** Result that there is no error after system has updated quantity in "STOCK_DETAIL" data store
**FROM PROCESS:**
2.1.4, 2.2.4, 2.3.4, 2.4.4, 2.5.4 Update quantity
**TO PROCESS:**
2.1.5 Generate distribution document
2.2.5 Generate transfer document
2.3.5 Generate purchasing document
2.4.5 Update production document
2.5.5 Generate receiving document
**DATA STRUCTURE:**
[dist_data, tran_data, pur_data, prod_data]
+ id_verified plant

**DATA FLOW NAME:** argu_data
**DESCRIPTION:** Retrieved argument of query process and report generation that is identified by user
**FROM PROCESS:**
**TO PROCESS:**
3.0 Query process
3.1 Find data
4.0 Report generation
4.1 Find data
**DATA STRUCTURE:**
(div_id) + (sect_id) + (plant_id) + (stock_id) +
([today, start date + stop date])

**DATA FLOW NAME:** del_data
**DESCRIPTION:** ID or name that can address row
**FROM PROCESS:**
**TO PROCESS:** 1.2 Delete data
1.2.1 Find row
**DATA STRUCTURE:**
[ID, name] *depend on each data store*

**DATA FLOW NAME:** dist_doc
**DESCRIPTION:** Distribution document for checking
**FROM PROCESS:** 2.1.5 Generate distribution document
**TO PROCESS:** 2.1.6 Add distribution data
**DATA STRUCTURE:**
*see report in appendix B*

**DATA FLOW NAME:** dist_data
**DESCRIPTION:** Data in approval requisition form
**FROM PROCESS:**
**TO PROCESS:** 2.1 Distribute seed and seedling
2.1.1 Enter distribution data
**DATA STRUCTURE:**
div_id + sect_id + dist_id + date1+ [emp_ID, name]
+ [vendor_id, name] + {plant_dist_list}

**DATA FLOW NAME:** distribute_verified plant
**DESCRIPTION:** System verifies that seed and seedling was distributed
**FROM PROCESS:** 2.1.3 Verify distribution data
**TO PROCESS:** 2.1.4 Update quantity
**DATA STRUCTURE:**
dist_id + div_id + sect_id + plant_id + stock_id

**DATA FLOW NAME:** edit_data
**DESCRIPTION:** ID or name that can address row and new data for updates the row
**FROM PROCESS:**
**TO PROCESS:** 1.3 Edit data
1.3.1 Find row
**DATA STRUCTURE:**
[ID, name]+ new_data

**DATA FLOW NAME:** error
**DESCRIPTION:** Error message from system that respond to user
**FROM PROCESS:** 1.1.3 Insert new row
**TO PROCESS:** 1.1.5 Display error message
**DATA STRUCTURE:**
*string of error statement*

**DATA FLOW NAME:** found
**DESCRIPTION:** Result that system can find requested row
**FROM PROCESS:** 3.1 Find data
4.1 Find data
1.2.1 Find row
1.3.1 Find row
**TO PROCESS:** 3.2 Process data
4.2 Process data
1.2.1 Display found row
1.3.2 Display found row
**DATA STRUCTURE:**
*row number of found row*

**DATA FLOW NAME:** ID
**DESCRIPTION:** Unique code that is primary key in data store and is generated by system
**FROM PROCESS:** 1.1.1 Input or generate ID
**TO PROCESS:** 1.1.2 Verify related table
**DATA STRUCTURE:**
*primary key in data store*

**DATA FLOW NAME:** id_verified h/w
**DESCRIPTION:** System verifies that quantity of seed and seedling are available in the stock
**FROM PROCESS:**
2.1.2, 2.2.2, 2.3.2, 2.4.2, 2.5.2 Verify PLANT_ID
**TO PROCESS:**
2.1.3 Verify distribution data
2.2.3 Verify transfer data
2.3.3 Verify purchasing data
2.4.3 Verify production data
2.5.3 Verify receiving data
**DATA STRUCTURE:**
*data in PLANT and STOCK_DETAIL data store*

**DATA FLOW NAME:** new_data
**DESCRIPTION:** Data of 16 data stores ("BASIC DATA") that pass validation rule
**FROM PROCESS:**
**TO PROCESS:** 1.1 Add data
1.1.1 Input or generate ID
**DATA STRUCTURE:**
*data of data store*

Figure 4.20 Data dictionary entries for data flows

**DATA FLOW NAME:** new_row
**DESCRIPTION:** Approval data that is inserted into data store
**FROM PROCESS:** 1.1.3 Insert new row
**TO PROCESS:** 1.1.4 Update new data
**DATA STRUCTURE:**
(ID) + new_data

**DATA FLOW NAME:** PLANT_ID
**DESCRIPTION:** Unique ID that represents seed and seedling and is generated by system to protect redundant data
**FROM PROCESS:** 2.1.1 Enter distribution data
2.2.1 Enter transfer data
2.3.1 Enter purchasing data
2.4.1 Enter production data
2.5.1 Enter receiving data
**TO PROCESS:**
2.1.2, 2.2.2, 2.3.2, 2.4.2, 2.5.2 Verify PLANT_ID
**DATA STRUCTURE:**
primary key in "PLANT" data store

**DATA FLOW NAME:** plant_id, quantity, date
**DESCRIPTION:** plant_id, quantity of seed and seedling, date that are updated in "NET_QUANTITY" data store
**FROM PROCESS:**
2.1.4, 2.2.4, 2.3.4, 2.4.4, 2.5.4 Update quantity
**TO PROCESS:**
**DATA STRUCTURE:**
plant_id, net_quan and date2 in "NET_QUANTITY" data store

**DATA FLOW NAME:** prod_data
**DESCRIPTION:** Data in approval production form
**FROM PROCESS:**
**TO PROCESS:** 2.4 Produce seed and seedling
2.4.1 Enter production data
**DATA STRUCTURE:**
div_id + sect_id + produce_id + date1+ [emp_ID, name] + {plant_produce_list}

**DATA FLOW NAME:** purchase_verified plant
**DESCRIPTION:** System verifies that seed and seedling was purchased
**FROM PROCESS:** 2.3.3 Verify purchasing data
**TO PROCESS:** 2.3.4 Update quantity
**DATA STRUCTURE:**
purchase_id + div_id + sect_id + plant_id + stock_id

**DATA FLOW NAME:** pur_doc
**DESCRIPTION:** Purchasing document for checking
**FROM PROCESS:** 2.3.5 Generate purchasing document
**TO PROCESS:** 2.3.6 Add purchasing data
**DATA STRUCTURE:**
*see report in appendix B*

**DATA FLOW NAME:** receive_verified plant
**DESCRIPTION:** System verifies that seed and seedling was received
**FROM PROCESS:** 2.5.3 Verify receiving data
**TO PROCESS:** 2.5.4 Update quantity
**DATA STRUCTURE:**
depend on what are methods of receiving

**DATA FLOW NAME:** not found
**DESCRIPTION:** Result that system cannot find requested row
**FROM PROCESS:** 3.1 Find data
4.1 Find data
1.2.1 Find row
1.3.1 Find row
**TO PROCESS:**
3.4 Display not found message
4.4 Display not found message
1.2.3 Display not found message
1.3.3 Display not found message
**DATA STRUCTURE:**
*0*

**DATA FLOW NAME:** produce_verified plant
**DESCRIPTION:** System verifies that seed and seedling was produced
**FROM PROCESS:** 2.4.3 Verify production data
**TO PROCESS:** 2.4.4 Update quantity
**DATA STRUCTURE:**
produce_id + div_id + sect_id + plant_id + stock_id

**DATA FLOW NAME:** prod_doc
**DESCRIPTION:** Production document for checking
**FROM PROCESS:** 2.4.5 Generate production document
**TO PROCESS:** 2.4.6 Add production data
**DATA STRUCTURE:**
*see report in appendix B*

**DATA FLOW NAME:** pur_data
**DESCRIPTION:** Data in approval purchase form
**FROM PROCESS:**
**TO PROCESS:** 2.3 Purchase seed and seedling
2.3.1 Enter purchasing data
**DATA STRUCTURE:**
div_id + sect_id + purchase_id + date1+ [emp_ID, name] + [per_ID, name]+ {plant_purchase_list}

**DATA FLOW NAME:** quantity
**DESCRIPTION:** quantity of seed and seedling that are updated in "STOCK_DETAIL" data store
**FROM PROCESS:**
2.1.4, 2.2.4, 2.3.4, 2.4.4, 2.5.4 Update quantity
**TO PROCESS:**
**DATA STRUCTURE:**
quan1 and quan5 in "STOCK_DETAIL" data store

**DATA FLOW NAME:** rece_doc
**DESCRIPTION:** Receiving document for checking
**FROM PROCESS:** 2.5.5 Generate receiving document
**TO PROCESS:** 2.5.6 Add receiving data
**DATA STRUCTURE:**
*see report in appendix B*

**DATA FLOW NAME:** report_data
**DESCRIPTION:** Data from report generation
**FROM PROCESS:** 4.2 Process data
**TO PROCESS:** 4.3 Print result
**DATA STRUCTURE:**
*same as the "report" but it is not printed output *

Figure 4.21 Data dictionary entries for data flows (continued)

**DATA FLOW NAME:** retrieved data
**DESCRIPTION:** Data that is retrieved from data store when system get row number
**FROM PROCESS:** 1.2.2 Display found row
1.3.2 Display found row
**TO PROCESS:** 1.2.4 Delete row
1.3.4 Update row
**DATA STRUCTURE:**
*data in data store*

**DATA FLOW NAME:** report
**DESCRIPTION:** Report is generated by system that has varies style
**FROM PROCESS:** 2.0 Transaction process
4.0 Report generation
4.3 Print result
**TO PROCESS:** 2.0 Transaction process
**DATA STRUCTURE:**
[seed and seedling report,
stock report,
vendor report,
contact person report,
employee report,
division and section report,
summary of seed and seedling quantity,
summary of seed and seedling quantity in section,
summary of seed and seedling quantity in section and requested range of time,
summary of seed and seedling quantity in stock and requested range of time,
summary of seed and seedling distribution,
summary of seed and seedling transfer,
summary of seed and seedling purchasing,
summary of seed and seedling production]

**DATA FLOW NAME:** tran_doc
**DESCRIPTION:** Transfer document for checking
**FROM PROCESS:** 2.2.5 Generate transfer document
**TO PROCESS:** 2.2.6 Add transfer data
**DATA STRUCTURE:**
*see report in appendix B*

**DATA FLOW NAME:** tran_data
**DESCRIPTION:** Data in approval transfer form
**FROM PROCESS:**
**TO PROCESS:** 2.2 Transfer seed and seedling
2.2.1 Enter transfer data
**DATA STRUCTURE:**
div_id + sect_id + transfer_id + date1+ [emp_ID, name] + {plant_transfer_list}

**DATA FLOW NAME:** trans_data
**DESCRIPTION:** Transaction data from section, that would like to distribute, transfer, purchase, produce or receive seed and seedling.
**FROM PROCESS:**
**TO PROCESS:** 2.0 transaction process
**DATA STRUCTURE:**
[dist_data, tran_data, pur_data, prod_data]

**DATA FLOW NAME:** result_data
**DESCRIPTION:** data from query process
**FROM PROCESS:** 2.0 Transaction process
3.2 process data
**TO PROCESS:** 3.3 display result
**DATA STRUCTURE:**
[seed and seedling data in each stock
stock that has each seed and seedling (all/specific)
list of all/specific seed and seedling quantity in stock and a period
list of all/specific seed and seedling distribution in a period
list of all/specific seed and seedling transfer in a period
list of all/specific seed and seedling purchasing in a period
list of all/specific seed and seedling production in a period
seed and seedling that provide in each stock (all/specific)
summary graph of seed and seedling quantity (sort out the stock or section)
summary graph of seed and seedling quantity in a period (sort out the stock or section)
summary graph of seed and seedling distribution in a period (sort out the stock or section)
summary graph of seed and seedling transfer in a period (sort out the stock or section)
summary graph of seed and seedling purchasing in a period (sort out the stock or section)
summary graph of seed and seedling production in a period (sort out the stock or section)]

**DATA FLOW NAME:** transfer_verified plant
**DESCRIPTION:** System verifies that seed and seedling was transfered
**FROM PROCESS:** 2.2.3 Verify transfer data
**TO PROCESS:** 2.2.4 Update quantity
**DATA STRUCTURE:**
transfer_id + div_id + sect_id + plant_id + stock_id

**DATA FLOW NAME:** Updateable data
**DESCRIPTION:** Data of 16 data stores, which can add, edit or delete. The data must pass validation rule.
**FROM PROCESS:**
**TO PROCESS:** 1.0 data manipulation
**DATA STRUCTURE:**
[new_data, del_data, edit_data]

**DATA FLOW NAME:** Verified data
**DESCRIPTION:** Data of basic data is verified with validation rule
**FROM PROCESS:** 1.1.2 verity related table
**TO PROCESS:** 1.1.3 insert new row
**DATA STRUCTURE:**
(ID) + new_data

Figure 4.22 Data dictionary entries for data flows (continued)

### 4.2.3 Input and Output Design

Output reports, which support the activity of the operation and management. The outputs are generated in tubular form. See designed output reports in appendix B.

Input and output screens are user-friendly with GUI (graphic user interface, they are MDI window. An MDI window is a frame window in which you can open multiple document windows (sheets) and move among the sheets. The input screens have input validation. A user can choose to access between a particular data, which is deep in detail, or just overall data, which is convenient for comparative analysis. See designed screens in Appendix C.

### 4.2.4 Database Design

Selecting and identifying data from collected data and system requirement, it gets related entities, attributes and constraints. These entities are identified relationships and designed with relational database concept. Through normalization a collection of data in a record structure is refined and shown in Figure 4.23 data structure diagram. The designed database is central database, which works as back office database. Descriptions of data stores are shown as following.

1. Table: DIVISION

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | | Div_ID | Char(3) | Division ID |
| | | Name1 | Char(68) | Division Name |
| | | Name2 | Char(40) | Division short name |

2. Table: SECTION

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | ✓ | Div_ID | Char(3) | Division ID |
| ✓ | | Sect_ID | Char(3) | Section ID |
| | | Name1 | Char(68) | Section name |
| | | Name2 | Char(40) | Section short name |

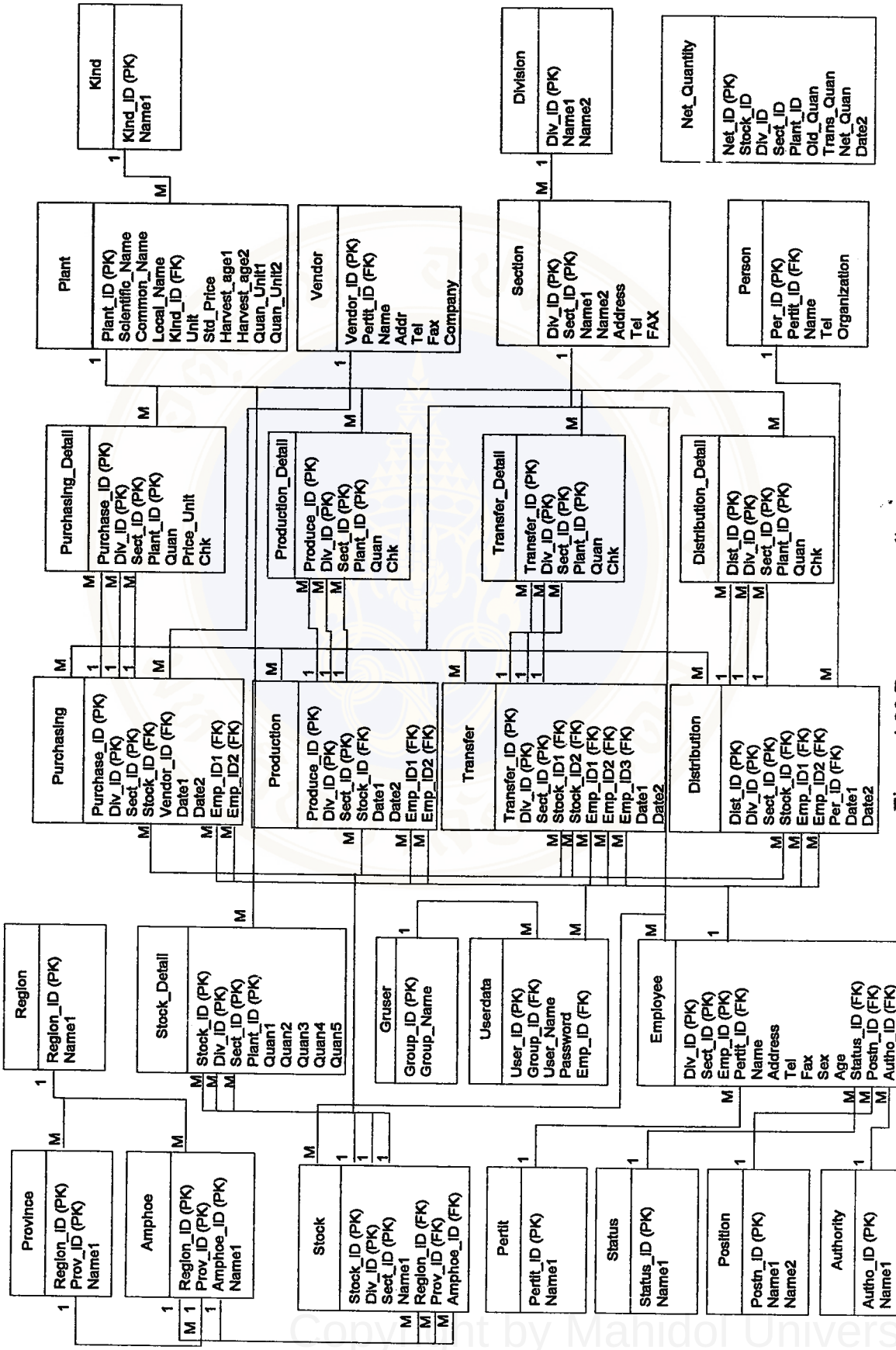



Figure 4.23 Data structure diagram

3. Table: PLANT

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | | Plant_ID | Char(3) | Plant ID |
| | | Scientific_Name | Char(120) | Scientific name |
| | | Common_Name | Char(100) | Common name |
| | | Local_Name | Char(120) | Hardware specification name |
| | ✓ | Kind_ID | Char(2) | Kinds of plant |
| | | Unit | Char(10) | Plant unit such as kg. |
| | | Std_Price | Real | Standard price |
| | | Harvest_Age1 | Number | Minimum harvest age (days) |
| | | Harvest_Age2 | Number | Maximum harvest age (days) |
| | | Quan_Unit1 | Number | Minimum Quantity per unit |
| | | Quan_Unit2 | Number | Maximum Quantity per unit |

4. Table: KIND

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | | Kind_ID | Char(2) | Kind of plant ID |
| | | Name1 | Char(50) | Kind of plant name |

5. Table: POSITION

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | | Postn_ID | Char(2) | Position ID |
| | | Name1 | Char(40) | Position name |
| | | Name2 | Char(24) | Position short name |

6. Table: STATUS

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | | Status_ID | Char(2) | Status ID |
| | | Name1 | Char(15) | Status name |

7. Table: AUTHORITY

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | | Autho_ID | Char(2) | Authority ID |
| | | Name1 | Char(40) | Authority name |

8. Table: PERTIT

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | | Pertit_id | Char(2) | Pertit ID |
| | | Name1 | Char(12) | Pertit name |

9. Table: EMPLOYEE

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | ✓ | Div_ID | Char(3) | Division ID |
| ✓ | ✓ | Sect_ID | Char(3) | Section ID |
| ✓ | | Emp_ID | Char(4) | Employee ID |
| | ✓ | Pertit_ID | Char(2) | Pertit ID |
| | | Name | Char(68) | Employee name |
| | | Address | Char(80) | Address |
| | | Tel | Char(20) | Telephone number |
| | | Fax | Char(20) | Facsimile number |
| | | Sex | Char(1) | Sex |
| | | Age | Number | Age |
| | ✓ | Status_id | Char(2) | Status ID |
| | ✓ | Postn_id | Char(2) | Position ID |
| | ✓ | Autho_id | Char(2) | Authority ID |

10. Table: PERSON

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | | Per_ID | Char(4) | Person ID |
| | ✓ | Pertit_id | Char(2) | Pertit ID |
| | | Name | Char(68) | Person name |
| | | Tel | Char(20) | Telephone number |
| | | Organization | Char(60) | Organization of person |

11. Table: REGION

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | | Region_ID | Char(1) | Region ID |
| | | Name1 | Char(30) | Region name |

12. Table: PROVINCE

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | ✓ | Region_ID | Char(1) | Region ID |
| ✓ | | Prov_ID | Char(3) | Province ID |
| | | Name1 | Char(30) | Province name |

13. Table: AMPHOE

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | ✓ | Region_ID | Char(1) | Region ID |
| ✓ | ✓ | Prov_ID | Char(3) | Province ID |
| ✓ | | Amphoe_ID | Char(2) | Amphoe ID |
| | | Name1 | Char(30) | Amphoe name |

14. Table: VENDOR

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | | Vendor_ID | Char(4) | Vendor ID |
| | ✓ | Pertit_ID | Char(2) | Pertit ID |
| | | Name | Char(68) | Vendor name |
| | | Addr | Char(100) | Address |
| | | Tel | Char(20) | Telephone number |
| | | Fax | Char(20) | Facsimile number |
| | | Company | Char(100) | Company of vendor |

15. Table: GRUSER

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | | Group_ID | Char(1) | Group ID |
| | | Group_Name | Char(20) | Group name or authorization (1 = DBA/System administrator, 2 = Operator or staff, 3 = general user |

16. Table: USERDATA

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | | User_ID | Char(4) | User ID |
| | ✓ | Group_ID | Char(1) | Group ID |
| | | User_Name | Char(10) | User name for enter system |
| | | Password | Char(10) | Password |
| | ✓ | Emp_ID | Char(4) | Employee ID |

17. Table: STOCK

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | | Stock_ID | Char(2) | Stock ID |
| ✓ | ✓ | Div_ID | Char(3) | Division ID |
| ✓ | ✓ | Sect_ID | Char(3) | Section ID |
| | | Name1 | Char(30) | Stock name |
| | ✓ | Region_ID | Char(1) | Region ID |
| | ✓ | Prov_ID | Char(3) | Province ID |
| | ✓ | Amphoe_ID | Char(2) | Amphoe ID |

18. Table: STOCK_DETAIL

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | ✓ | Stock_ID | Char(2) | Stock ID |
| ✓ | ✓ | Div_ID | Char(3) | Division ID |
| ✓ | ✓ | Sect_ID | Char(3) | Section ID |
| ✓ | ✓ | Plant_ID | Char(3) | Plant ID |
| | | Quan1 | Number | Quantity in stock |
| | | Quan2 | Number | Quantity that keep for making ancestry of plant |

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| | | Quan3 | Number | Quantity for using in demonstration plot |
| | | Quan4 | Number | Quantity for support jobs of organization |
| | | Quan5 | Number | Quantity that can distribute, transfer |

19. Table: DISTRIBUTION

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | | Dist_ID | Char(4) | Running ID of DISTRIBUTION table |
| ✓ | ✓ | Div_ID | Char(3) | Division ID |
| ✓ | ✓ | Sect_ID | Char(3) | Section ID |
| | ✓ | Stock_ID | Char(2) | Stock ID |
| | ✓ | Emp_ID1 | Char(4) | Emp_ID (Employee has responsibility about requisition of distribution) |
| | ✓ | Emp_ID2 | Char(4) | Distributor ID (Emp_ID) |
| | ✓ | Per_ID | Char(4) | Requisition person ID (Per_ID) |
| | | Date1 | Date | Requisition of distribution date |
| | | Date2 | Date | Distribution date |

20. Table: DISTRIBUTION_DETAIL

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | ✓ | Dist_ID | Char(4) | Running ID of DISTRIBUTION table |
| ✓ | ✓ | Div_ID | Char(3) | Division ID |
| ✓ | ✓ | Sect_ID | Char(3) | Section ID |
| ✓ | ✓ | Plant_ID | Char(3) | Plant ID |
| | | Quan | Number | Quantity of distribution |
| | | Chk | Char(1) | Record data checking |

21. Table: TRANSFER

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | | Transfer_ID | Char(4) | Running ID of TRANSFER table |
| ✓ | ✓ | Div_ID | Char(3) | Division ID |
| ✓ | ✓ | Sect_ID | Char(3) | Section ID |
| | ✓ | Stock_ID1 | Char(2) | Start stock ID |
| | ✓ | Stock_ID2 | Char(2) | Destination stock ID |
| | ✓ | Emp_ID1 | Char(4) | Emp_ID (Employee has responsibility about requisition of transfer) |
| | ✓ | Emp_ID2 | Char(4) | Emp_ID (Employee has responsibility about seed and seedling transfer) |
| | ✓ | Emp_ID3 | Char(4) | Recipient ID (Emp_ID) |
| | | Date1 | Date | Requisition of transfer date |
| | | Date2 | Date | Receipt date |

22. Table: TRANSFER_DETAIL

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | ✓ | Transfer_ID | Char(4) | Running ID of TRANSFER table |
| ✓ | ✓ | Div_ID | Char(3) | Division ID |
| ✓ | ✓ | Sect_ID | Char(3) | Section ID |
| ✓ | ✓ | Plant_ID | Char(3) | Plant ID |
| | | Quan | Number | Quantity of transfer |
| | | Chk | Char(1) | Record data checking |

23. Table: PURCHASING

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | | Purchase_ID | Char(4) | Running ID of PURCHASING table |
| ✓ | ✓ | Div_ID | Char(3) | Division ID |
| ✓ | ✓ | Sect_ID | Char(3) | Section ID |
| | ✓ | Stock_ID | Char(2) | Stock ID |
| | ✓ | Vendor_ID | Char(4) | Vendor ID |
| | | Date1 | Date | Requisition of purchasing date |
| | | Date2 | Date | Receipt date |
| | ✓ | Emp_ID1 | Char(4) | Emp_ID (Employee has responsibility about requisition of purchasing) |
| | ✓ | Emp_ID2 | Char(4) | Recipient ID (Emp_ID) |

24. Table: PURCHASING_DETAIL

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | ✓ | Purchase_ID | Char(4) | Running ID of PURCHASING table |
| ✓ | ✓ | Div_ID | Char(3) | Division ID |
| ✓ | ✓ | Sect_ID | Char(3) | Section ID |
| ✓ | ✓ | Plant_ID | Char(3) | Plant ID |
| | | Quan | Number | Quantity of purchasing |
| | | Price_unit | Real | Price per unit |
| | | Chk | Char(1) | Record data checking |

25. Table: PRODUCTION

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | | Produce_ID | Char(4) | Running ID of PRODUCTION table |
| ✓ | ✓ | Div_ID | Char(3) | Division ID |
| ✓ | ✓ | Sect_ID | Char(3) | Section ID |
| | ✓ | Stock_ID | Char(2) | Stock ID |
| | | Date1 | Date | Requisition of production date |
| | | Date2 | Date | Receipt date |
| | ✓ | Emp_ID1 | Char(4) | Emp_ID (Employee has responsibility about requisition of production) |
| | ✓ | Emp_ID2 | Char(4) | Recipient ID (Emp_ID) |

26. Table: PRODUCTION_DETAIL

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | ✓ | Produce_ID | Char(4) | Running ID of PRODUCTION table |
| ✓ | ✓ | Div_ID | Char(3) | Division ID |
| ✓ | ✓ | Sect_ID | Char(3) | Section ID |
| ✓ | ✓ | Plant_ID | Char(3) | Plant ID |
| | | Quan | Number | Quantity of production |
| | | Chk | Char(1) | Record data checking |

27. Table: NET_QUANTITY

| PK | FK | Field Name | Data Type | Description |
|---|---|---|---|---|
| ✓ | | Net_ID | Char(4) | Running ID of NET_QUANTITY table |
| | | Stock_ID | Char(2) | Stock ID |
| | | Div_ID | Char(3) | Division ID |
| | | Sect_ID | Char(3) | Section ID |
| | | Plant_ID | Char(3) | Plant ID |
| | | Old_Quan | Number | Exist quantity in the stock |
| | | Trans_Quan | Number | Quantity is involved by transactions |
| | | Net_Quan | Number | Net quantity in the stock |

### 4.2.5 Procedure Design

PowerBuilder is object- oriented programming, which each created menu or window with PowerBuilder is a self-contained module called an object. The basic building blocks of a PowerBuilder application are the created objects. Each object contains the particular characteristics and behaviors (properties, events, and functions) that are appropriate to it. By taking advantage of object-oriented programming techniques such as encapsulation, inheritance, and polymorphism, the processes in DFD are refine and adjusted as following:

1. Transform the processes in DFD into structure chart, which is easier to maintain and reduce complexity.

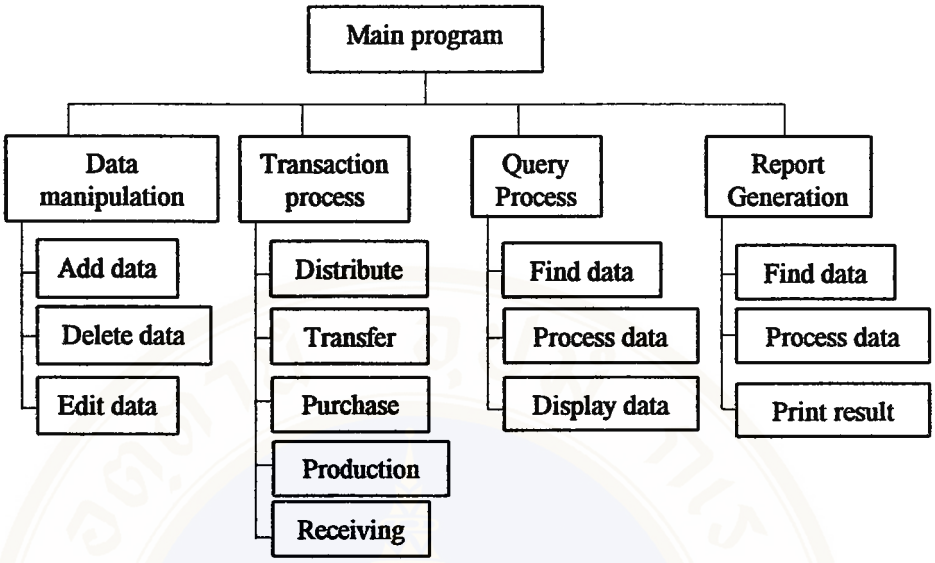



Figure 4.24 Structure Chart

2. Boxes in structure chart are grouped and designed to window objects. These window objects have some common functions and events, which can design to ancestor window objects for inheritance and reuse.

Table 4.1 Characteristic of process

| Process | Characteristic |
|---|---|
| Data manipulation | - All windows can inherit |
| Transaction process | - Distribution, transfer, purchase, production windows inherit from the same window but use different data.<br>- Receiving window is inherited separate from the others.<br>- Stock window is the separate window. |
| Query process | - Distinguish tabular and graph output window<br>- Tabular windows use the same window but use different display data (data window object).<br>- Graph window can inherit |
| Report generation | - All windows use the same window but use different display data (data window object) |

3. Design additional events to ease of use in window operations style and convenient operation such as open, close and so on.

Table 4.2 Ancestor window objects

| Process | Event group | Event |
|---|---|---|
| Data manipulation | Window | Open<br>Close |
| | Navigation | First<br>Previous<br>Next<br>Last<br>Search |
| | Data | Add<br>Delete<br>Update<br>Undo<br>Undelete |
| Transaction process | Window | Open<br>Close |
| | Data | Insert master<br>Update master<br>Insert detail<br>Update detail<br>Search<br>Print |
| Query process (graph) | Window | Open<br>Close |
| | Data | Set argument<br>Process data |
| | Convenient | Set graph type<br>Set graph spacing |
| Report generation | Window | Open<br>Close |
| | Data | Set argument<br>Process data |
| | Convenient | Zoom<br>Page<br>Save as<br>Printer setup<br>Print |

4. Consider shared functions to distribute and create new window object.

Table 4.3 Shared window object

| Process | Function | Description |
|---|---|---|
| Data manipulation | Search | Search data by name |
| Transaction process | Search | Search plant_id by name |
| | Print | Print output document |
| Query process | Display Graph | Set graph type<br>Set graph spacing |
| Query process and report generation | Set argument | Set division and section<br>Set plant<br>Set date<br>Set stock |
| Share | About | About the system |

5. Design program specification of each window that is shown in Appendix D.

## 4.3 Information System of Seed and Seedling Management for Soil and Water Conservation

Using the Database painter of Sybase SQL anywhere 5.0 to create database. Powersoft PowerBuilder Enterprise Version 6.0 is graphical application development tools and HelpScribble 5.0.0 is help authoring tools for seed and seedling management system, which implements under microcomputer Pentium III 667 MHz CPU, Windows98 operating system. The information system consists of 3 major processes: 1) Data Manipulation 2) Transaction Processing, 3) Query and Report. The capability of processes is show in Table 4.4

Table 4.4 Capability of process

| Process | Capability |
|---|---|
| Data manipulation | Manipulate data in these data store<br>1. Plant<br>2. Kind<br>3. Division<br>4. Section<br>5. Employee<br>6. Person |

| Process | Capability |
|---|---|
| | 7. Pertit<br>8. Position<br>9. Status<br>10. Authority<br>11. Region<br>12. Province<br>13. Amphoe<br>14. Vendor<br>15. Gruser<br>16. Userdata |
| Transaction process | 1. Distribute<br>2. Transfer<br>3. Purchase<br>4. Produce<br>5. Receive<br>6. Net_Quantity |
| Query and report | *A. Tabular result*<br>1. Seed and seedling data in each stock<br>2. Stock that has each seed and seedling (all/specific)<br>3. List of all/specific seed and seedling quantity in stock and a period<br>4. List of all/specific seed and seedling distribution in a period<br>5. List of all/specific seed and seedling transfer in a period<br>6. List of all/specific seed and seedling purchasing in a period<br>7. List of all/specific seed and seedling production in a period<br>8. Seed and seedling that provide in each stock (all/specific)<br>*B. Graph result*<br>1. Summary graph of seed and seedling quantity (sort out the stock or section)<br>2. Summary graph of seed and seedling quantity in a period (sort out the stock or section)<br>3. Summary graph of seed and seedling distribution in a period (sort out the stock or section)<br>4. Summary graph of seed and seedling transfer in a period (sort out the stock or section)<br>5. Summary graph of seed and seedling purchasing in a period (sort out the stock or section)<br>6. Summary graph of seed and seedling production in a period (sort out the stock or section) |

| Process | Capability |
|---|---|
| | *C. Report*<br>1. Seed and seedling report<br>2. Stock report<br>3. Vendor report<br>4. Contact person report<br>5. Employee report<br>6. Division and section report<br>7. Summary of seed and seedling quantity<br>8. Summary of seed and seedling quantity in section<br>9. Summary of seed and seedling quantity in section and requested range of time<br>10. Summary of seed and seedling quantity in stock and requested range of time<br>11. Summary of seed and seedling distribution<br>12. Summary of seed and seedling transfer<br>13. Summary of seed and seedling purchasing<br>14. Summary of seed and seedling production |

Since the information system works on client/server network, many users can use it. The users are divided into 3 levels: Database Administrator, operator and general user. The privilege of each user group is shown in Table 4.5.

Table 4.5 Privilege of users

| Process / User | Data Manipulation | Transaction Processing | Query And report |
|---|---|---|---|
| DBA | ✓ | | ✓ |
| Operator | | ✓ | ✓ |
| General user | | | ✓ |

Users access the information system through login window, which verifies user name, password and accessible level. In addition, operator can access data of transaction processing in division and section that user is under.

By the way, the information system has an additional process for data access of branch office, which may not support online system, communicate database server across network. For this reason, the data is sent to database center by electronic mail or file transfer process.

# CHAPTER V

# DISCUSSION

The information system of seed and seedling management for soil and water conservation for a case study of Land Development Department works on client/server network. The system consists of three major processes as follows:

- Data manipulation: Manipulate relative data such as plant, vendor, employee, division, section and so on.
- Transaction processing: Support operation management, which are distribution, transfer, purchasing, production and receiving.
- Query and report: Process pre-defined outputs and preplanned printed reports. A user can choose to access between a particular data, which is deep in detail, or just overall data, which is convenient for comparative analysis.

Users of the information system are divided into 3 levels: Database administrator, operator and general user. All levels can access query and report menu, but data manipulation menu can only be accessed by database administrator level, and transaction processing menu can only be accessed by operator level. They access the information system through login window, which verifies user name, password and accessible level. In addition, operator can access data of transaction processing in division and section in which user is under.

Moreover, the information system uses in many places; somewhere may not support online system, and communicate database server across network. For this reason, the data will be sent to and be executed at database center by electronic mail or file transfer process.

The information system is designed and developed using data flow analysis, relational database and Object-Oriented programming. It uses the Database painter of Sybase SQL anywhere 5.0 to create database, Powersoft PowerBuilder Enterprise Version 6.0 to develop application and HelpScribble 5.0.0 to create help file.

The advantages of the information system of seed and seedling management for soil and water conservation for a case study of Land Development Department are as follows:

- The problems of inefficiency seed and seedling management can be solved and improved. The information system changes operation from analog to electronics, searching from manual to digital and database from analog to digital. The information system introduces online operation and standard database, which reduces access time, increases accuracy and provides consistency of data.
- There is database and information system of seed and seedling management for soil and water conservation.
- Can transfer this application to other related organizations.

# CHAPTER VI

# CONCLUSION

Nowadays the information system is becoming more important in every field. Consequently, Land Development Department's computer network system can take advantage of seed and seedling management information system for soil and water conservation. Not only Land Development Department but also other organizations can use the information system to improve their ineffective operation of seed and seedling management or similar systems.

This information system was developed by PowerBuilder because it is object-oriented programming tool. The benefits of object-oriented as follows:

- Object-oriented programs can be more easily maintained and adapted than procedural languages.
- More code can be reused from project to project when the code is object-oriented, rather than procedural.
- The key ingredients of Object-oriented program are inheritance, Encapsulation and polymorphism.

Data flow analysis was used in this research, it was a valuable for performing process analysis although it had to be adjusted and refined in design process. Beside, as data flow analysis have been used for a long time, this research may be guideline for developer who would like to move his old data flow diagram and structure

programming to object-oriented programming. But the objects in this research focus on replacing process in data flow diagram with window object.

By the way, data in the information system was designed by relational database model. Relational database is adequate for database model in this research because information needs are two-dimensional and not complex data, which can be normalized.

## Recommendation

1. Develop functions for supporting electronic signature.
2. Develop this information system to OLAP (Online Analytical Processing) system that is designed for live data access and analysis. In this information system, most results in query process are summary and comparative analysis model. Developer can add statistical analysis model for viewing trends and required historical data. The OLAP system is used by analysts and managers who frequently want a higher-level aggregated view of the data.
3. Develop and link this information system with another relative systems for developing to EIS.
4. Apply this information system to Internet/ Intranet application. PowerBuilder version 7.0 offers significant productivity enhancements and broad support for Web-based component standards, web.pb (access Distributed PowerBuilder DPB objects as part of an HTTP application. And PowerBuilder 8.0 beta includes several new features for developers in the client/server environment and those who are migrating to heterogeneous, distributed Web enterprise, workspace and targets which manage and deploy HTML files, images, and other assets that comprise a Web site.

# REFERENCES

1. Land Development Department (1996). Introduction to Land Development Department. Available from: http://www.ldd.go.th. [Accessed 2000 Aug 25].
2. Renaud, PE. Introduction to client/server systems: A practical guide for systems professionals. USA: John Wiley & Sons; 1993.
3. Vaughn, LT. Client/Server system design & implementation. n.p.: Donnelley & Sons; 1994.
4. Martin J. Client/server database : enterprise computing / James Martin, Joe Leben. USA: Prentice Hall; 1995.
5. Hansen GW, James V. Database management and design. USA: Prentice-Hall; 1992.
6. Parker C. Management information systems: strategy and action. 2nd ed. Singapore: McGraw-Hill; 1993.
7. Ralph M. Stair. Principles of information systems: a managerial approach. 2nd ed. USA: boyd&fraser; 1996.
8. Korth HF, Silberschatz A. Database system concepts. 2nd ed. Singapore: McGraw-Hill; 1991.
9. James AS. Analysis and design of information systems. 2nd ed. Singapore: McGraw-Hill; 1989.

10. Stair RM. Principles of Information systems: A managerial approach 2nd ed. USA: Boyd & fraser; 1996.

11. The University of Texas at Austin(1996). Normalization. Available from: http://www.utexas.edu/cc/dbms/utinfo/relmode/normal1.html. [Accessed 2000 Aug 25].

12. Parker C, Thomas C. Management information systems second edition. Singapore: McGraw-Hill; 1993.

13. Roger SP. Software engineering a practitioner's approach. 4th ed. Singapore: McGraw-Hill; 1997.

14. Ashok Ramachandran (1998). What is PowerBuilder?. Available from: http://www.ashok.pair.com/whatispb.htm. [Accessed 2000 Aug 25].

15. Ashok Ramachandran (1998). PowerBuilder's Technical Strengths. Available from: http://www.ashok.pair.com/technica.htm. [Accessed 2000 Aug 25].

16. Ashok Ramachandran (1998). Connecting to the database. Available from: http://www.ashok.pair.com/connecti.htm. [Accessed 2000 Aug 25].

17. Informix Software Inc.(1999). Informix. Available from: http://www.informix.com. [Accessed 2000 Aug 25].

18. Sybase Inc. (1999). Sybase Available from: http://www.sybase.com. [Accessed 2000 Apr 25].

19. Jugtrimongkol U. Information system for government inventory management: A case study of Land Development Department [M.S. Thesis in Technology of Information System Management]. Bangkok: Factory of Graduate Studies, Mahidol University; 1999.

20. Yuthas K, Young ST. Material matters: Assessing the effectiveness of materials management IS. Information & Management 1998;33:115-124.

21. Kirkwood J. Sybase Architecture and Administration. New York: Ellis Horwood; 1993.

22. Branchek B, Eidson R, Kindel C, et al. Using Windows NT. Indianapolis (IN): Que; 1993.

23. Eckel G, Houlette F, Stoddard J, Wagner R. Inside Windows NT. Carmel (IN): New Riders; 1993.

24. กรมพัฒนาที่ดิน กระทรวงเกษตรและ สหกรณ์. บันทึกสรุปการจัดทำแผนอัตรากำลัง 3 ปี 2538-2540. ฝ่ายการพิมพ์ กองแผนที่และการพิมพ์ สำนักพัฒนาโครงสร้าง ส่วนราชการและอัตรากำลัง สำนักงานก.พ.

25. ชุมพล คนศิลป์, วิทูร ชินพันธุ์, วิชัย สุวรรณเกิด, และคณะ. พืชตระกูลถั่วเพื่อการปรับปรุงบำรุงดิน. กรุงเทพมหานคร: ห้างหุ้นส่วนจำกัด เฟิร์สเพรส.

# APPENDIX A

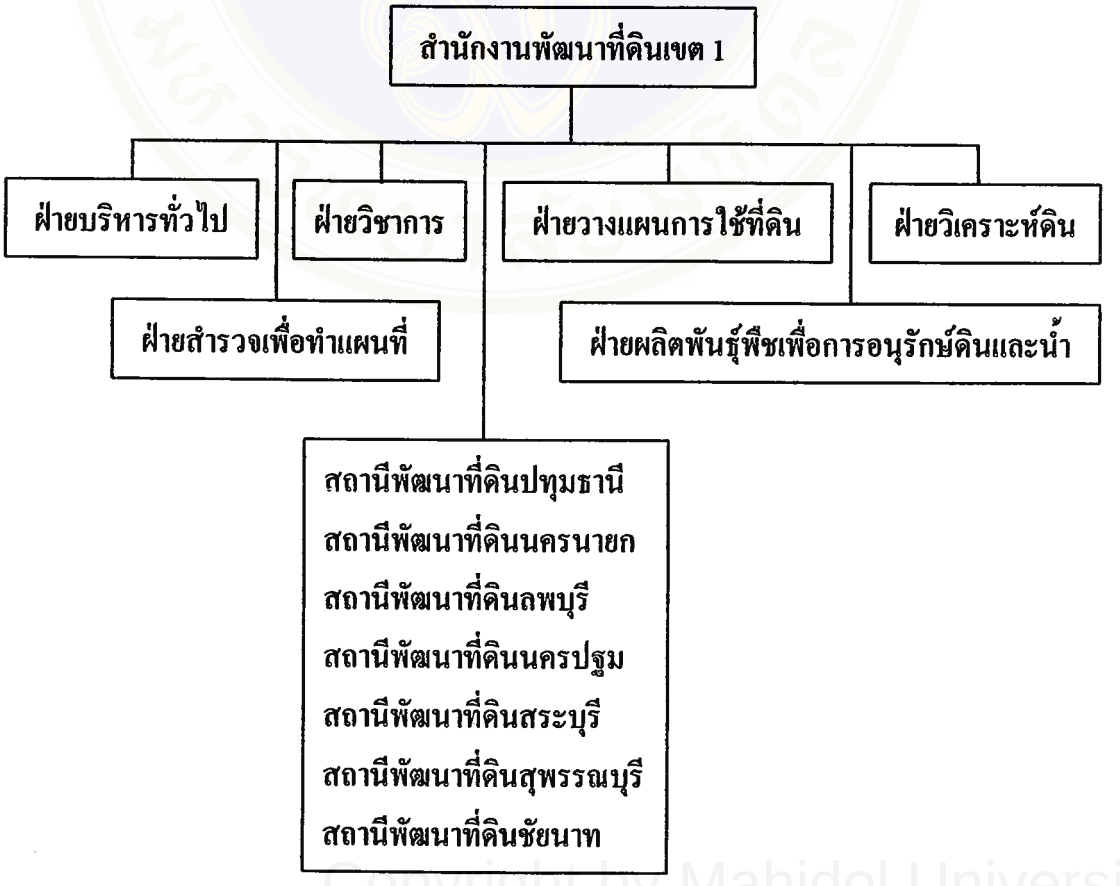
# EXSISTING REPORT

**พื้นที่รับผิดชอบของสำนักงานพัฒนาที่ดินเขต**

1. *สำนักงานพัฒนาที่ดินเขต 1* รับผิดชอบในเขตพื้นที่จังหวัด ปทุมธานี อยุธยา นครปฐม สุพรรณบุรี นครนายก สระบุรี สมุทรปราการ นนทบุรี ลพบุรี สิงห์บุรี อ่างทอง ชัยนาท และ กรุงเทพฯ
2. *สำนักงานพัฒนาที่ดินเขต 2* รับผิดชอบในเขตพื้นที่จังหวัด ชลบุรี สระแก้ว ฉะเชิงเทรา ระยอง จันทบุรี ตราด และปราจีนบุรี
3. *สำนักงานพัฒนาที่ดินเขต 3* รับผิดชอบในเขตพื้นที่จังหวัด นครราชสีมา บุรีรัมย์ ชัยภูมิ และสุรินทร์
4. *สำนักงานพัฒนาที่ดินเขต 4* รับผิดชอบในเขตพื้นที่จังหวัด อุบลราชธานี นครพนม ร้อยเอ็ด ยโสธร ศรีษะเกษ มุกดาหาร และอำนาจเจริญ
5. *สำนักงานพัฒนาที่ดินเขต 5* รับผิดชอบในเขตพื้นที่จังหวัด ขอนแก่น อุดรธานี มหาสารคาม หนองคาย กาฬสินธุ์ สกลนคร และหนองบัวลำพู
6. *สำนักงานพัฒนาที่ดินเขต 6* รับผิดชอบในเขตพื้นที่จังหวัด เชียงใหม่ แม่ฮ่องสอน ลำพูน และลำปาง
7. *สำนักงานพัฒนาที่ดินเขต 7* รับผิดชอบในเขตพื้นที่จังหวัด น่าน เชียงราย แพร่ และพะเยา
8. *สำนักงานพัฒนาที่ดินเขต 8* รับผิดชอบในเขตพื้นที่จังหวัด พิษณุโลก เพชรบูรณ์ อุตรดิตถ์ เลย และพิจิตร
9. *สำนักงานพัฒนาที่ดินเขต 9* รับผิดชอบในเขตพื้นที่จังหวัด นครสวรรค์ ตาก กำแพงเพชร สุโขทัย และอุทัยธานี
10. *สำนักงานพัฒนาที่ดินเขต 10* รับผิดชอบในเขตพื้นที่จังหวัด ราชบุรี กาญจนบุรี เพชรบุรี สมุทรสาคร ประจวบคีรีขันธ์ และสมุทรสงคราม
11. *สำนักงานพัฒนาที่ดินเขต 11* รับผิดชอบในเขตพื้นที่จังหวัด สุราษฎร์ธานี ระนอง พังงา นครศรีธรรมราช ชุมพร กระบี่ และภูเก็ต
12. *สำนักงานพัฒนาที่ดินเขต 12* รับผิดชอบในเขตพื้นที่จังหวัด สงขลา สตูล ปัตตานี ยะลา พัทลุง นราธิวาส และตรัง

**สำนักงานพัฒนาที่ดินเขต 1 ประกอบด้วย**

1. ฝ่ายบริหารทั่วไป
2. ฝ่ายวิชาการ
3. สถานีพัฒนาที่ดินฝ่ายวางแผนการใช้ที่ดิน
4. ฝ่ายสำรวจเพื่อทำแผนที่
5. ฝ่ายวิเคราะห์ดิน
6. ฝ่ายผลิตพันธุ์พืชเพื่อการอนุรักษ์ดินและน้ำ
7. สถานีพัฒนาที่ดินปทุมธานี
8. สถานีพัฒนาที่ดินนครนายก
9. สถานีพัฒนาที่ดินลพบุรี
10. สถานีพัฒนาที่ดินนครปฐม
11. สถานีพัฒนาที่ดินสระบุรี
12. สถานีพัฒนาที่ดินสุพรรณบุรี
13. สถานีพัฒนาที่ดินชัยนาท



## รายชื่อเมล็ดพันธุ์พืชเพื่อการอนุรักษ์ดินและน้ำและการปรับปรุงบำรุงดิน

| เมล็ดพันธุ์พืช | ชื่อวิทยาศาสตร์ | ชื่อสามัญ | ชื่ออื่นๆ | ประเภทพันธุ์พืช |
|---|---|---|---|---|
| กระถิน | *Leucaena leucocephala (Lam de Wit)* | White popinae, Lead tree, Wild tamarind | ภาคกลาง – กระถินบ้าน<br>ราชบุรี – กะเส็ดโคก กะเส็ดบก<br>ภาคใต้ – สะตอเบา สะตอเทศ<br>เชียงใหม่ – ผักก้านดิน<br>ภาคเหนือ - ผักหนองบก | ไม้พุ่ม |
| แคฝรั่ง | *Gliricidia sepium (jacguin) Kunth ex Walp.* | Nicaraguan coffee shade; madre de cacao | - | ไม้ยืนต้นขนาดเล็ก |
| คราม่ป่า | *Tephrosia candida (Roxb.) D.C.* | - | - | ไม้พุ่มตระกูลถั่ว ขนาดเล็ก |
| ถั่วเขียวธรรมดา | *Phaseolus aureus* | Mung bean | - | พืชล้มลุก |
| ถั่วเขียวผิวดำ | *Phaseolus mungo* | Black gram | - | พืชล้มลุก |
| ถั่วเขียวเมล็ดแดง | *Phaseolus radiatus* | Green gram | - | พืชล้มลุก |
| ถั่วคาโลโปโกเนียม | *Calopogonium mucunoides* | - | ถั่วคาโลโป | พืชตระกูลถั่ว แบบเถาเลื้อย อายุ 1 ปี |

| เมล็ดพันธุ์พืช | ชื่อวิทยาศาสตร์ | ชื่อสามัญ | ชื่ออื่นๆ | ประเภทพันธุ์พืช |
|---|---|---|---|---|
| ถั่วคุดซู | *Pueraria phaseoloides* | Kudzu | เพอโร, เพอราเรีย, ถั่วเลื้อยป่า | พืชตระกูลถั่ว แบบเถาเลื้อย |
| ถั่วไซราโตรซีราโตร, เซอราโตร | *Macroptilium atropurpureum* | Sirato | ซีราโตร, เซอราโตร | พืชตระกูลถั่ว แบบเถาเลื้อย อายุหลายปี |
| ถั่วนิ้วนางแดง | *Phaseolus calcaratus* | Rice bean | ถั่วแดงซีลอน, ถั่วข้าว | พืชข้ามปีอายุสั้น |
| ถั่วแปบ | *Lablab purpureus (L.) Sweet* | - | - | พืชตระกูลถั่ว แบบเถาเลื้อย |
| ถั่วพุ่ม | *Vigna spp.* | Cowpea | - | พืชล้มลุก |
| ถั่วพร้าเมล็ดขาว | *Canavalia ensiformis* | Jack bean | - | พืชล้มลุก |
| ถั่วพร้าเมล็ดแดง | *Canavalia gladiata* | Sword bean | - | พืชล้มลุก |
| ถั่วมะแฮะ | *Cajanus cajan (L.) Millsp* | Pigeon pea | Kadios | ไม้พุ่มขนาดเล็ก |
| ถั่วมะแฮะนก | *Flemingia macrophylla (Willd) merr or Flemingia congesta* | Flemingia | - | ไม้พุ่ม (shrub) |
| ถั่วลาย | *Centrosema pubescens* | Centrosema | - | พืชตระกูลถั่ว อายุหลายปี |

| เมล็ดพันธุ์พืช | ชื่อวิทยาศาสตร์ | ชื่อสามัญ | ชื่ออื่นๆ | ประเภทพันธุ์พืช |
|---|---|---|---|---|
| ถั่วลิสง | *Arachis hypogaea L.* | Groundnut | - | พืชตระกูลถั่วฤดูเดียวหรือข้ามปี |
| ถั่วหรั่งหรือถั่วปันหยี | *Voandzeia subterranea (L.) Thouars* | Bambara Groundnut | ถั่วไทร, ถั่วเมล็ดเดียว, การแจโป | พืชตระกูลถั่ว |
| ถั่วเหลือง | *Glycine max* | Soybeans | - | พืชล้มลุกตระกูลถั่ว |
| ถั่วเวอราโน | *Stylosanthes hamata* | แคริบเบี้ยนสไตโล | ถั่วฮามาต้า | ไม้พุ่มเตี้ย |
| ปอเทือง | *Crotalaria juncea* | - | - | พืชตระกูลถั่วฤดูเดียว |
| โสนคางคก | *Sesbania aculeata* | - | - | พืชตระกูลถั่วอายุ 1ปี |
| โสนจีนแดง | *Sesbania Cannabina* | - | - | พืชตระกูลถั่วอายุ 1ปี |
| โสนอัฟริกัน | *Sesbania rostrata* | - | - | พืชตระกูลถั่วอายุ 1ปี |
| โสนอินเดีย | *Sesbania speciosa* | - | - | พืชตระกูลถั่วอายุ 1ปี |

ใบขอเลขที่.....

วันที่.....เดือน.....พ.ศ.....

เรื่อง ขอเมล็ดพันธุ์

เรียน หัวหน้าฝ่ายผลิตฯ

ข้าพเจ้า.....อยู่บ้านเลขที่.....หมู่ที่.....
ตำบล.....อำเภอ.....จังหวัด.....
มีความประสงค์จะขอเมล็ดพันธุ์.....จำนวน.....
เพื่อนำไปใช้ปลูกปรับปรุงดินในพื้นที่จำนวน.....ไร่ ซึ่งตั้งอยู่บ้านเลขที่.....
หมู่.....ตำบล.....อำเภอ.....จังหวัด.....
จึงเรียนมาเพื่อโปรดพิจารณา

ลงชื่อ.....ผู้ขอ

อนุมัติ

| เมล็ดพันธุ์ | จำนวน (กก.) | อัตราปลูก (กก./ไร่) | ใช้ปลูกในพื้นที่ (ไร่) |
|---|---|---|---|
| ..... | ..... | ..... | ..... |
| ..... | ..... | ..... | ..... |

ลงชื่อ.....ผู้อนุมัติ
(.....)
หัวหน้าฝ่ายผลิตฯ

ได้รับเมล็ดพันธุ์ไปเรียบร้อยแล้ว เมื่อวันที่.....

ลงชื่อ.....ผู้รับ
ลงชื่อ.....ผู้จ่าย

ลงบัญชีควบคุมเมล็ดพันธุ์เรียบร้อยแล้ว เมื่อวันที่.....

| งาน/โครงการ | ชื่อเมล็ดพันธุ์ | จำนวน (กก.) |
|---|---|---|
| งานอนุรักษ์ดินและน้ำ | ..... | ..... |
| โครงการปุ๋ยหมัก | ..... | ..... |
| โครงการดินเค็ม | ..... | ..... |

ลงชื่อ.....ผู้ลงบัญชี

# ใบตรวจรับพัสดุ

สำนักงานพัฒนาที่ดินเขต 1

วันที่.....เดือน.....พ.ศ.....

ชื่อผู้ส่งพัสดุ.....

ได้รับสิ่งของตามรายการข้างล่างนี้ให้แก่ สำนักงานพัฒนาที่ดินเขต 1 หมู่ 2 ต.ลำผักกูด อ.ธัญบุรี จ. ปทุมธานี

| ลำดับที่ | รายการ | จำนวนหน่วย | จำนวนเงิน | |
|---|---|---|---|---|
| | | | บาท | สตางค์ |
| | | | | |

รวม.....รายการ เป็นเงินรวมทั้งสิ้น.....บาท (.....)

คณะกรรมการฯ ได้รับตรวจรับพัสดุ ซึ่ง.....ได้นำมาส่งมอบ ณ สำนักงานพัฒนาที่ดินเขต 1 ปรากฏว่าถูกต้องครบถ้วนตามหลักฐานที่ได้ตกลงกันไว้ จึงได้รับของไว้ และได้มอบให้แก่สำนักงานพัฒนาที่ดินเขต 1 และได้ลงชื่อไว้เป็นหลักฐาน พร้อมทั้งได้มอบใบตรวจรับพัสดุให้แก่เจ้าหน้าที่พัสดุของสำนักงานพัฒนาที่ดินเขต 1 และผู้ขายไว้เป็นหลักฐานคนละ 1 ฉบับ

(ลงชื่อ).....ประธานกรรมการ
(.....)
ตำแหน่ง.....

(ลงชื่อ).....กรรมการ
(.....)
ตำแหน่ง.....

(ลงชื่อ).....กรรมการ
(.....)
ตำแหน่ง.....

(พ.ด. ด. 6)

## ใบสำคัญรับเงิน

เขียนที่.....

วันที่.....เดือน.....พ.ศ.....

ข้าพเจ้า.....ชื่อสกุล.....ตำบล.....

อำเภอ.....จังหวัด.....ได้รับเงินจาก.....

เป็นเงิน.....บาท.....สตางค์ ตัวอักษร.....

เป็นค่า.....

.....

ไปเป็นการถูกต้องแล้ว ตั้งแต่วันที่.....เดือน.....พ.ศ.....

ลงชื่อ.....ผู้รับเงิน

ลงชื่อ.....พยาน

ลงชื่อ.....พยาน

ได้จ่ายเงินจำนวนนี้ไปถูกต้องแล้ว ตรวจแล้ว

..... .....

# ใบสำคัญรับเงิน

ที่.....

ข้าพเจ้า.....อยู่บ้านเลขที่.....หมู่ที่.....

ตำบล.....อำเภอ.....จังหวัด.....

ได้รับงานจาก.....ดังรายการต่อไปนี้

| ลำดับที่ | รายการ | จำนวนเงิน | |
|---|---|---|---|
| | | บาท | สตางค์ |
| | | | |
| | รวม | | |

จำนวนเงิน.....

(ตัวอักษร)

ข้าพเจ้า.....ขอมอบให้.....

เป็นผู้รับเงินตามใบสำคัญแทนข้าพเจ้า

ลงชื่อ.....ผู้มอบฉันทะ

ลงชื่อ.....ผู้รับเงิน

ลงชื่อ.....ผู้จ่ายเงิน

บัญชีการรับ-จ่ายพันธุ์พืช

(ชื่อพันธุ์พืช)

| วัน เดือน ปี | รายการ | เลขที่เอกสาร | รับ | | ก้นไว้ | | จ่าย | | เหลือ | | หมายเหตุ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | ผลิต | ซื้อ | ผลิต | ซื้อ | ผลิต | ซื้อ | ผลิต | ซื้อ | |
| | | | | | | | | | | | |

สรุปผลการดำเนินงานผลิตและจัดหาเมล็ดพันธุ์

ของ ฝ่ายผลิตพันธุ์พืชฯ สำนักงานพัฒนาที่ดินเขต 1

ณ วันที่.....เดือน.....พ.ศ.....

| งาน/โครงการ (ชนิดพันธุ์พืช) | แผนงาน (กก.) | | งบประมาณที่ได้รับ (บาท) | | ผลการดำเนินงาน (สะสม) | | การใช้จ่ายงบประมาณ (สะสม) | | การใช้จ่ายเมล็ดพันธุ์พืช (กก.) | | | | คงเหลือ (กก.) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ผลิต | จัดหา | ผลิต | จัดหา | ผลิต | จัดหา | ผลิต | จัดหา | คงเหลือจากปี..... | ทำพันธุ์ | ใช้สาธิตส่งเสริม | แจกจ่าย | |
| | | | | | | | | | | | | | |
| รวม | | | | | | | | | | | | | |

**รายงานผลการรับจ่ายเมล็ดพันธุ์**

**ของ ฝ่ายผลิตพันธุ์พืชฯ สำนักงานพัฒนาที่ดินเขต 1**

**เดือน.....พ.ศ.....**

| งาน/โครงการ (ชนิดพันธุ์พืช) | คงเหลือยกมาจากเดือนก่อน | จำนวนเมล็ดพันธุ์ที่ผลิต/จัดหา | การใช้จ่ายเมล็ดพันธุ์ | | | | คงเหลือ | |
|---|---|---|---|---|---|---|---|---|
| | | | เก็บไว้ทำพันธุ์ | ใช้ในแปลงสาธิตส่งเสริมของ สพด. | ให้หมอดินและเกษตรกรทั่วไป | ให้หน่วยงานอื่นๆระบุ | หมายเหตุ | |
| | | | | | | | | |

แผนการใช้จ่ายเมล็ดพันธุ์

ของ ฝ่ายผลิตพันธุ์พืชฯ สำนักงานพัฒนาที่ดินเขต 1

ปีงบประมาณ.....

| งาน/โครงการ (ชนิดพันธุ์พืช) | คงเหลือยกมาจากปีก่อน | จำนวนเมล็ดพันธุ์ที่ผลิต/จัดหา | เก็บไว้ทำพันธุ์ | ใช้ในแปลงสาธิตและส่งเสริม | สนับสนุนหน่วยงานในกรมฯ | ให้หมอดินอาสาและเกษตรกรทั่วไป | | ให้หน่วยงานอื่นๆ (ระบุ) กรมส่งเสริมการเกษตร | | คงเหลือ |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | ขยายพันธุ์ | ปรับปรุงดิน | ขยายพันธุ์ | ปรับปรุงดิน | |
| | | | | | | | | | | |

แผนการผลิตและจัดหาเมล็ดพันธุ์

ของ ฝ่ายผลิตพันธุ์พืชฯ สำนักงานพัฒนาที่ดินเขต 1

ปีงบประมาณ.....

| งาน/โครงการ (ชนิดพันธุ์พืช) | ผลิต | | | | จัดหา | | | |
|---|---|---|---|---|---|---|---|---|
| | เป้าหมายที่ได้รับ (กก.) | สถานที่ดำเนินการ | พื้นที่ ดำเนินการ (ไร่) | ผลผลิตที่ได้รับ (กก.) | เป้าหมายที่ได้รับ (กก.) | สถานที่ดำเนินการ | พื้นที่ ดำเนินการ (ไร่) | ผลผลิตที่ได้รับ (กก.) |
| | | | | | | | | |

# APPENDIX B

# DESIGNED REPORT

## รายงานพันธุ์พืช

| รหัสพันธุ์พืช / ประเภทพันธุ์พืช / อายุเก็บเกี่ยวน้อยสุด (วัน) | ชื่อวิทยาศาสตร์ / หน่วยที่ใช้จัดเก็บ / อายุเก็บเกี่ยวมากสุด (วัน) | ชื่อสามัญ / ราคากลางต่อหน่วย (บาท) / ปริมาณน้อยสุดที่ใช้ต่อไร่ | ชื่อพื้นเมือง / ปริมาณมากสุดที่ใช้ต่อไร่ |
|---|---|---|---|
| 001 | Leucaena leucocephala (Lam de Wit) | White popinac, Lead tree, Wild tamarind | กระถินบ้าน, กระเจ็ดโคก, กระเจ็ดบก, สะตอบก, สะตอเทศ, มักกาดดิน, มักหนองบก |
| ไม้พุ่ม | กิโลกรัม | 100.00 | |
| 002 | Glincidia sepium (Jacquin) Kunth ex Walp. | Nicaraguan coffee shade, madre de | แคฝรั่ง |
| ไม้ยืนต้นขนาดเล็ก | กิโลกรัม | 100.00 | |
| 003 | Tephrosia candida (Roxb.) D.C. | | คราม(ป่า) |
| ไม้พุ่มตระกูลถั่วขนาดเล็ก | กิโลกรัม | 100.00<br>2.00 | 3.00 |

Plant report

## รายงานคลังเก็บพันธุ์พืช

| รหัสคลัง | คลังเก็บพันธุ์พืช | ภาค | จังหวัด | อำเภอ |
|---|---|---|---|---|
| กอง/สำนักงาน : 000 ส่วนกลาง | | กลุ่ม/ฝ่าย/สถานีฯ : 000 ส่วนกลาง | | |
| 1 | คลัง1 | ภาคกลาง | กรุงเทพมหานคร | บางกอกน้อย |
| 2 | คลัง2 | ภาคกลาง | กรุงเทพมหานคร | บางกอกใหญ่ |
| 3 | คลัง3 | ภาคกลาง | กรุงเทพมหานคร | บางพลัด |
| กอง/สำนักงาน : 111 สำนักงานพัฒนาที่ดินเขต 1 | | กลุ่ม/ฝ่าย/สถานีฯ : 001 ฝ่ายบริหารทั่วไป | | |
| 1 | aaa | ภาคใต้ | กระบี่ | เมือง |
| 2 | bbb | ภาคใต้ | กระบี่ | คลองท่อม |
| 3 | ccc | ภาคใต้ | ตรัง | เมือง |
| 4 | ddd | ภาคใต้ | ชุมพร | ทะโต๊ะ |

Stock report

## รายงานตัวแทนจำหน่าย

| รหัสตัวแทนจำหน่าย<br>โทรศัพท์ | คำนำหน้า | ชื่อ-สกุล<br>โทรสาร | ที่อยู่<br>บริษัทหรือองค์กรที่สังกัด |
|---|---|---|---|
| 0001<br>2445881 | นางสาว | จันทนา ช้างนงษ์ | ห้วยขวาง |
| 0002<br>4481142 | นางสาว | สันทณีย์ รัตติธรรมกุล | ม.สินพัฒนาธานี พุทธมณฑลสาย 2<br>บ.มอนชานโต้ |
| 0003<br>8665702 | นางสาว | แขอิตา ยันศิลชนชาติ | ซ.วัดยางสุทธาราม<br>บ.ไอโซเนท |
| 0004<br>9714705 | นางสาว | รัตดาว อนันตชัยยง | ถ.รามอินทรา<br>บ.มอนชานโต้ |
| 0005<br>5945064 | นางสาว | อรพินทร์ พิริยะสิริศิลป์ | ม.บ้านบัวทอง<br>บ.เอกชำรง |
| 0006<br>8665491 | นางสาว | วารุณา เพียรสิทธจน์ | ม.พระปิ่น 3 บางใหญ่ นนทบุรี<br>Power Buy |

รวมทั้งหมด 6 คน

Vendor report

## รายงานบุคคลภายนอกที่มาติดต่อ

| รหัสบุคคลภายนอก | คำนำหน้า | ชื่อ-สกุล | โทรศัพท์ | องค์กรที่สังกัด |
|---|---|---|---|---|
| 0001 | นาย | สุวรรณ กัลยา | 2158799 | กระทรวงเกษตรและสหกรณ์ |
| 0002 | นางสาว | อมรฤดี ทวีศักดิ์สกุล | 2782802 | กระทรวงเกษตรและสหกรณ์ |
| 0003 | นาย | ทรงพร อำพริ | 5834786 | สหกรณ์การเกษตร |
| 0004 | นางสาว | อุษา จักรีมงคล | 9328892 | กรมพัฒนาที่ดิน |
| 0005 | นาย | ศึกฤทธิ์ อสัมภินพงษ์ | 4652382 | ธนาคารเพื่อการเกษตร |
| 0006 | นาย | เอนก กุมาร | 01-2545211 | กรมพัฒนาที่ดินจังหวัดพะเยา |
| 0007 | นาย | คมสรรค์ ปาละวัฒน์ | 01-6391244 | ธนาคารเพื่อการเกษตร |
| 0008 | นาง | จันทร เกษมสุข | 5987523 | กระทรวงเกษตรและสหกรณ์ |

รวมทั้งหมด 8 คน

Contact person report

## รายงานบุคคลในองค์กรจำแนกตามหน่วยงาน

| รหัสพนักงาน คำนำหน้า | ชื่อ-สกุล | ที่อยู่ | เพศ | อายุ |
|---|---|---|---|---|
| โทรศัพท์ | โทรสาร สถานภาพการสมรส | ตำแหน่ง | ตำแหน่งที่ | |

**กอง/สำนักงาน :** 111 สำนักงานพัฒนาที่ดินเขต 1

**กลุ่ม/ฝ่าย/สถานี :** 001 ฝ่ายบริหารทั่วไป **มีจำนวนบุคลากร** 6 คน

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0003 | นาง | จิตินณ เสอนดี | | กลอง 7 ปทุมธานี | | F | 40 |
| 5698747 | | | แต่งงาน | เจ้าหน้าที่บริหารงานทั่วไป | ผู้มีอำนาจจ่ายพันธุ์พืช | | |
| 0004 | นาง | เพ็ญศรี กลางทัพ | | กลอง 6 ปทุมธานี | | F | 36 |
| 6931245 | | | แต่งงาน | เจ้าหน้าที่บริหารงานธุรการ | ผู้ทำการจัดซื้อ | | |
| 0005 | นาง | บุษดี มีใย | | กรุงเทพฯ | | F | 40 |
| 3259874 | | | แต่งงาน | เจ้าหน้าที่ธุรการ | ผู้มีอำนาจทำเรื่องโอนพันธุ์พืช | | |
| 0006 | นาย | สมจิตร์ ภาคสุขล | | นนทบุรี | | M | 32 |
| 5487912 | | | แต่งงาน | เจ้าหน้าที่การเงินและบัญชี | ผู้ทำการโอนพันธุ์พืช | | |
| 0007 | นาย | ศาสล ณ ฤทธิ์ | | ปทุมธานี | | M | 38 |
| 5632147 | | | แต่งงาน | นักวิชาการเกษตร | ผู้มีอำนาจตรวจรับพันธุ์พืช | | |
| 0011 | นางสาว | แก้วตา แดงอ่อน | | อ.ดิวานนท์ นนทบุรี | | F | 25 |
| 5884334 | | | โสด | เจ้าหน้าที่บันทึกข้อมูล | ผู้มีอำนาจทำเรื่องเบิกพันธุ์พืช | | |

Employee report

## รายงานกอง/สำนักงานพัฒนาที่ดิน

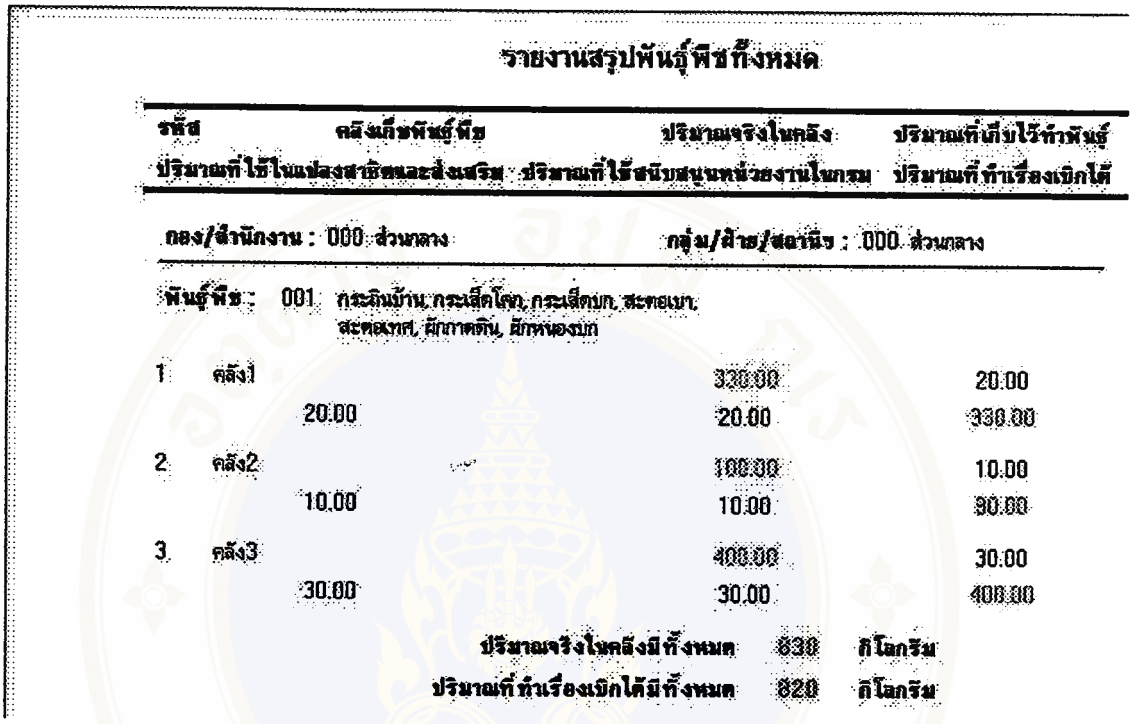
| รหัสกอง/สำนักงาน | ชื่อกอง/สำนักงาน | ชื่อย่อ |
|---|---|---|
| 000 | ส่วนกลาง | ส่วนกลาง |
| 101 | สำนักงานเลขานุการกรม | สลก. |
| 102 | กองคลัง | กค. |
| 103 | กองการเจ้าหน้าที่ | กกจ. |
| 104 | กองช่าง | กช. |
| 105 | กองแผนงาน | กผง. |
| 106 | กองแผนที่และการพิมพ์ | กผพ. |
| 107 | กองวางแผนการใช้ที่ดิน | กวผ. |
| 108 | กองวิเคราะห์ดิน | กวด. |
| 109 | กองสำรวจและจำแนกดิน | กสด. |
| 110 | กองอนุรักษ์ดินและน้ำ | กอน. |

Division report

## รายงานกลุ่ม/ฝ่าย/สถานีพัฒนาที่ดินจำแนกตามกอง/สำนักงานพัฒนาที่ดิน

| รหัสกลุ่ม/ฝ่าย/สถานีฯ | ชื่อกลุ่ม/ฝ่าย/สถานีฯ | ชื่อย่อ |
|---|---|---|
| **กอง/สำนักงาน : 000** | **ส่วนกลาง** | |
| 000 | ส่วนกลาง | ส่วนกลาง |
| 001 | หน่วยงานตรวจสอบภายใน | หน่วยงานตรวจสอบภายใน |
| **กอง/สำนักงาน : 101** | **สำนักงานเลขานุการกรม** | |
| 000 | สำนักงานเลขานุการกรม | สลก. |
| 001 | ฝ่ายสารบรรณ | ฝ่ายสารบรรณ |
| 002 | ฝ่ายนิติการ | ฝ่ายนิติการ |
| 003 | ฝ่ายช่วยอำนวยการและประสานราชการ | ฝ่ายช่วยอำนวยการฯ |
| 004 | ฝ่ายเผยแพร่และประชาสัมพันธ์ | ฝ่ายเผยแพร่ฯ |

Section report



รายงานสรุปพันธุ์พืชทั้งหมด

| รหัส | คลังเก็บพันธุ์พืช | ปริมาณจริงในคลัง | ปริมาณที่เก็บไว้ทำพันธุ์ |
|---|---|---|---|
| ปริมาณที่ใช้ในแปลงสาธิตและสิ่งแสดง | ปริมาณที่ใช้สนับสนุนหน่วยงานในกรม | ปริมาณที่ทำเรื่องเบิกได้ | |

กลุ่ม/สำนักงาน : 000 ส่วนกลาง กลุ่ม/ฝ่าย/สถานีฯ : 000 ส่วนกลาง

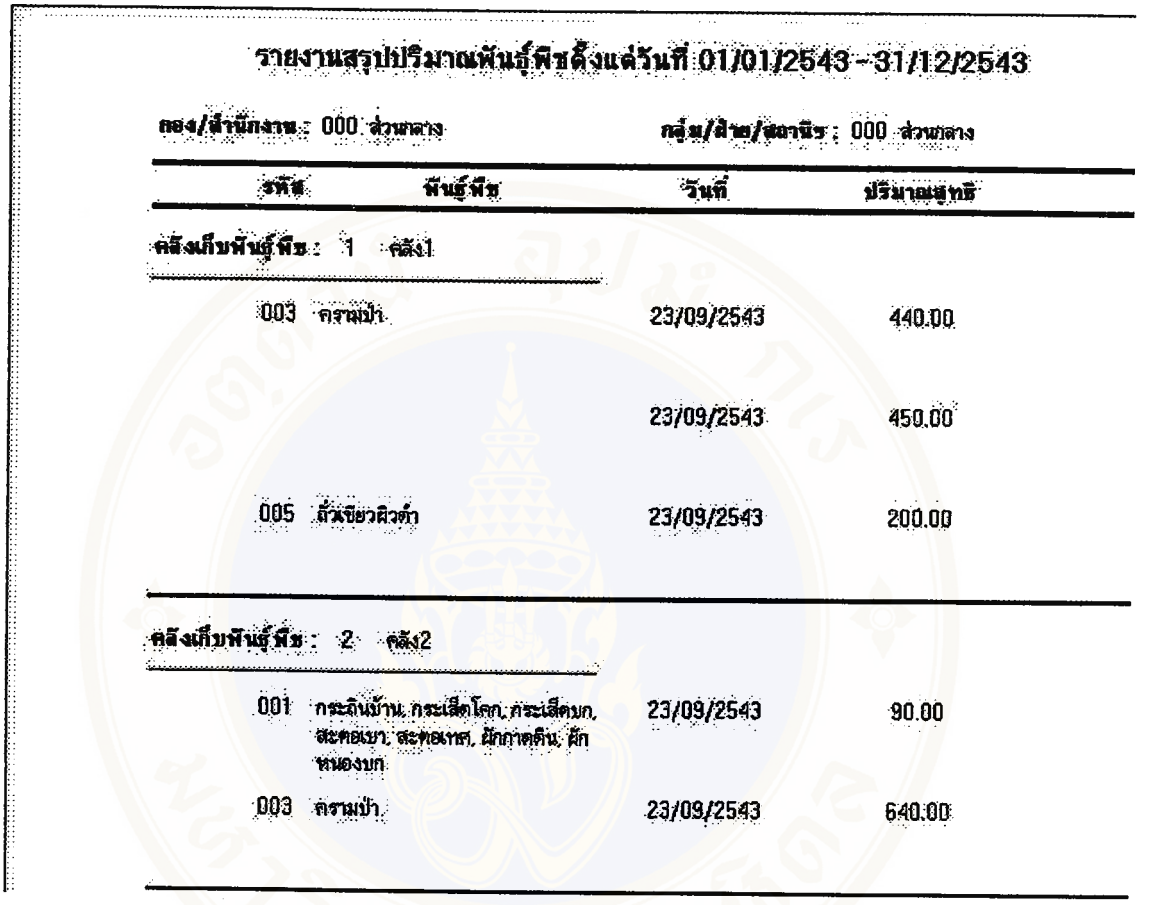
พันธุ์พืช : 001 กระถินบ้าน, กระเจี๊ยบโคก, กระเจี๊ยบบก, สะตอเบา, สะตอมาก, ผักกาดดิน, ผักหนองบก

| | | | | | |
|---|---|---|---|---|---|
| 1 | คลัง1 | | 320.00 | | 20.00 |
| | | 20.00 | 20.00 | | 330.00 |
| 2 | คลัง2 | | 100.00 | | 10.00 |
| | | 10.00 | 10.00 | | 90.00 |
| 3 | คลัง3 | | 400.00 | | 30.00 |
| | | 30.00 | 30.00 | | 400.00 |

ปริมาณจริงในคลังมีทั้งหมด 820 กิโลกรัม

ปริมาณที่ทำเรื่องเบิกได้มีทั้งหมด 820 กิโลกรัม

Summary of seed and seedling quantity

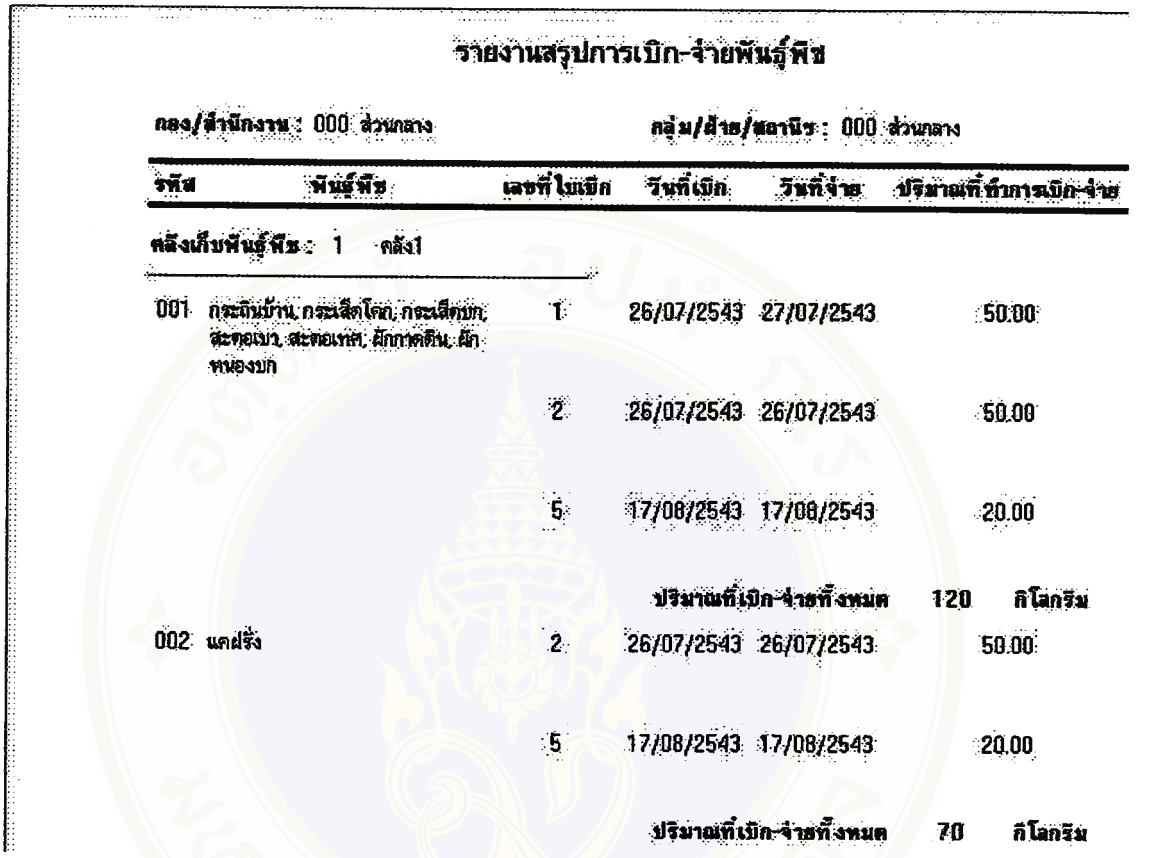


รายงานสรุปปริมาณพันธุ์พืชตั้งแต่วันที่ 01/01/2543 - 31/12/2543

กอง/สำนักงาน : 000 ส่วนกลาง กลุ่ม/ฝ่าย/สถานี : 000 ส่วนกลาง

| รหัส | พันธุ์พืช | วันที่ | ปริมาณสุทธิ |
|---|---|---|---|
| คลังเก็บพันธุ์พืช : 1 คลัง1 | | | |
| 003 | ตรามเปา | 23/09/2543 | 440.00 |
| | | 23/09/2543 | 450.00 |
| 005 | ถั่วเขียวผิวดำ | 23/09/2543 | 200.00 |
| คลังเก็บพันธุ์พืช : 2 คลัง2 | | | |
| 001 | กระถินบ้าน, กระเจี๊ยบโคก, กระเจี๊ยบบก, สะตอเขา, สะตอเทศ, ผักกาดดิน, ผักหนองบก | 23/09/2543 | 90.00 |
| 003 | ตรามเปา | 23/09/2543 | 640.00 |

Summary of seed and seedling quantity in requested range of time



รายงานสรุปการเบิก-จ่ายพันธุ์พืช

กลอง/สำนักงาน : 000 ส่วนกลาง กลุ่ม/ฝ่าย/สถานี : 000 ส่วนกลาง

| รหัส | พันธุ์พืช | เลขที่ใบเบิก | วันที่เบิก | วันที่จ่าย | ปริมาณที่ทำการเบิก-จ่าย |
|---|---|---|---|---|---|
| คลังเก็บพันธุ์พืช : 1 คลัง1 | | | | | |
| 001 | กระถินบ้าน, กระเจี๊ยบโคก, กระเจี๊ยบบก, สะทอเบา, สะตอเทศ, ผักกาดดิน, ผักหนองบก | 1 | 26/07/2543 | 27/07/2543 | 50.00 |
| | | 2 | 26/07/2543 | 26/07/2543 | 50.00 |
| | | 5 | 17/08/2543 | 17/08/2543 | 20.00 |
| | | | ปริมาณที่เบิก-จ่ายทั้งหมด | | 120 กิโลกรัม |
| 002 | แคฝรั่ง | 2 | 26/07/2543 | 26/07/2543 | 50.00 |
| | | 5 | 17/08/2543 | 17/08/2543 | 20.00 |
| | | | ปริมาณที่เบิก-จ่ายทั้งหมด | | 70 กิโลกรัม |

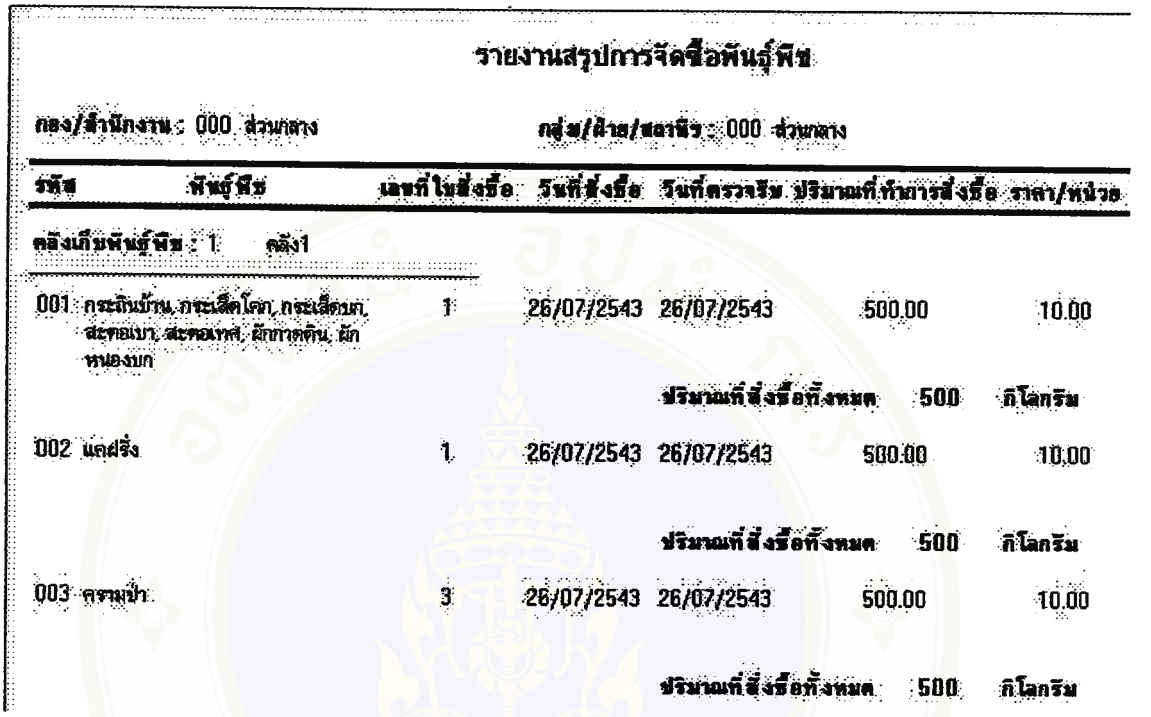
Summary of seed and seedling distribution

## รายงานสรุปการโอนพันธุ์พืช

กอง/สำนักงาน : 000 ส่วนกลาง กลุ่ม/ฝ่าย/สถาบัน : 000 ส่วนกลาง

| รหัส | พันธุ์พืช | เลขที่ใบโอน | วันที่ทำเรื่องโอน | วันที่รับโอน | ปริมาณที่ทำการโอน |
|---|---|---|---|---|---|
| คลังเก็บพันธุ์พืชต้นทาง : 1 คลัง1 | | | | | |
| คลังเก็บพันธุ์พืชปลายทาง : 2 คลัง2 | | | | | |
| 001 | กระถินบ้าน, กระเจี๊ยบโคก, กระเจี๊ยบเขา, สะตอเบา, สะตอเทศ, ผักกาดต้น, ผักหนองบก | 1 | 26/07/2543 | 26/07/2543 | 50.00 |
| | | | ปริมาณที่โอนทั้งหมด | 50 | กิโลกรัม |
| 002 | แคฝรั่ง | 1 | 26/07/2543 | 26/07/2543 | 50.00 |
| | | | ปริมาณที่โอนทั้งหมด | 50 | กิโลกรัม |
| 003 | ตะขบป่า | 1 | 26/07/2543 | 26/07/2543 | 50.00 |
| | | | ปริมาณที่โอนทั้งหมด | 50 | กิโลกรัม |

Summary of seed and seedling transfer



รายงานสรุปการจัดซื้อพันธุ์พืช

กอง/สำนักงาน : 000 ส่วนกลาง กลุ่ม/ฝ่าย/แผนก : 000 ส่วนกลาง

| รหัส | พันธุ์พืช | เลขที่ใบสั่งซื้อ | วันที่สั่งซื้อ | วันที่ตรวจรับ | ปริมาณที่ทำการสั่งซื้อ | ราคา/หน่วย |
|---|---|---|---|---|---|---|
| คลังเก็บพันธุ์พืช : 1 คลัง1 | | | | | | |
| 001 | กระถินบ้าน, กระเจี๊ยบโคก, กระเจี๊ยบแดง, สะเดาเทศ, สะเดาไทย, ผักกาดดอง, ผักหนองบก | 1 | 26/07/2543 | 26/07/2543 | 500.00 | 10.00 |
| | | | | ปริมาณที่สั่งซื้อทั้งหมด | 500 | กิโลกรัม |
| 002 | แคฝรั่ง | 1 | 26/07/2543 | 26/07/2543 | 500.00 | 10.00 |
| | | | | ปริมาณที่สั่งซื้อทั้งหมด | 500 | กิโลกรัม |
| 003 | ตราชูป่า | 3 | 26/07/2543 | 26/07/2543 | 500.00 | 10.00 |
| | | | | ปริมาณที่สั่งซื้อทั้งหมด | 500 | กิโลกรัม |

Summary of seed and seedling purchasing

## รายงานสรุปการผลิตพันธุ์พืช

กอง/สำนักงาน : 000 ส่วนกลาง  กลุ่ม/ฝ่าย/สถานี : 000 ส่วนกลาง

| รหัส | พันธุ์พืช | เลขที่การผลิต | วันที่ทำเรื่องผลิต | วันที่ตรวจรับ | ปริมาณที่ทำการผลิต |
|---|---|---|---|---|---|
| คลังเก็บพันธุ์พืช : 2 คลัง2 | | | | | |
| 005 | ตั้วเขียวผิวดำ | 4 | 17/08/2543 | 17/08/2543 | 200.00 |
| | | | ปริมาณที่ผลิตทั้งหมด | 200 | กิโลกรัม |
| คลังเก็บพันธุ์พืช : 3 คลัง3 | | | | | |
| 001 | กระถินบ้าน, กระเจี๊ยบโคก, กระเจี๊ยบบก, สะตอเบา, สะตอเทศ, ผักกาดดิน, ผักหนองบก | 1 | 26/07/2543 | 26/07/2543 | 500.00 |
| | | | ปริมาณที่ผลิตทั้งหมด | 500 | กิโลกรัม |
| 002 | แคฝรั่ง | 1 | 26/07/2543 | 26/07/2543 | 500.00 |
| | | | ปริมาณที่ผลิตทั้งหมด | 500 | กิโลกรัม |
| 003 | ตามป่า | 2 | 26/07/2543 | 26/07/2543 | 500.00 |

Summary of seed and seedling production

# APPENDIX C

# DESIGNED SCREEN

## 1. Data manipulation

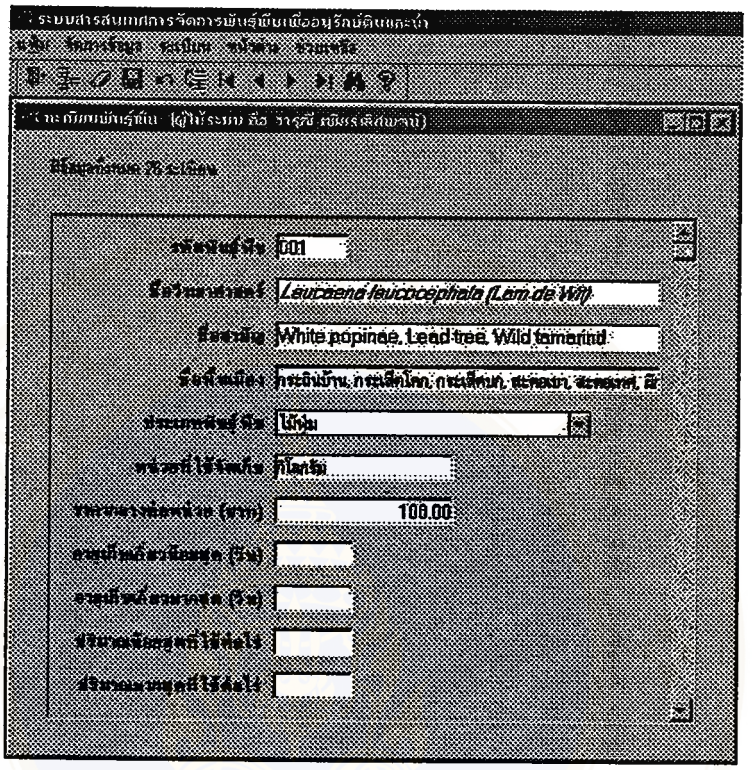



Figure C.1 w_plant

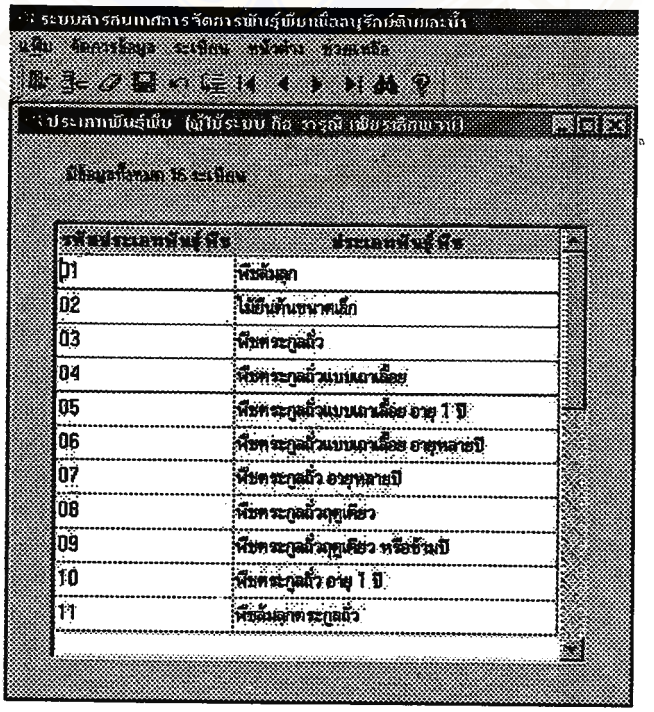



Figure C.2 w_kind

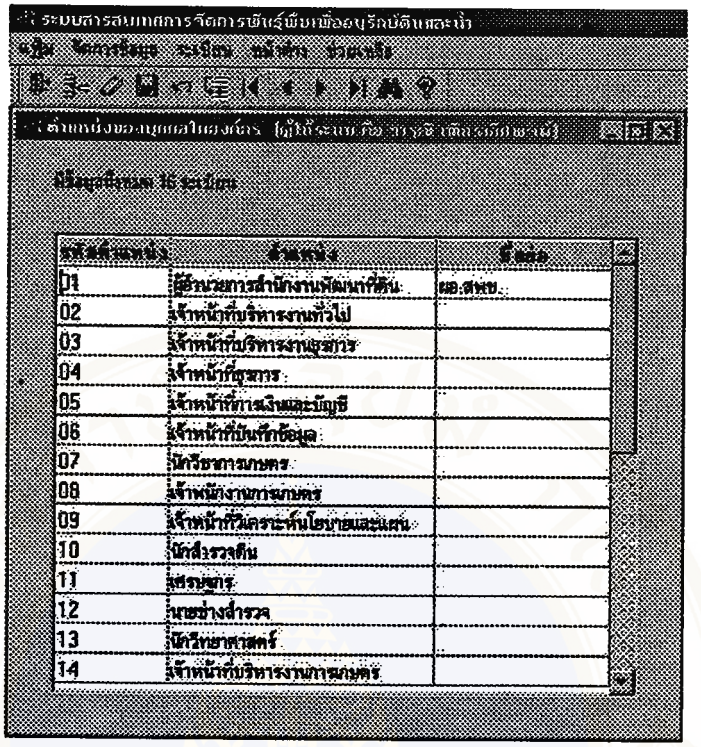


Figure C.3 w_position

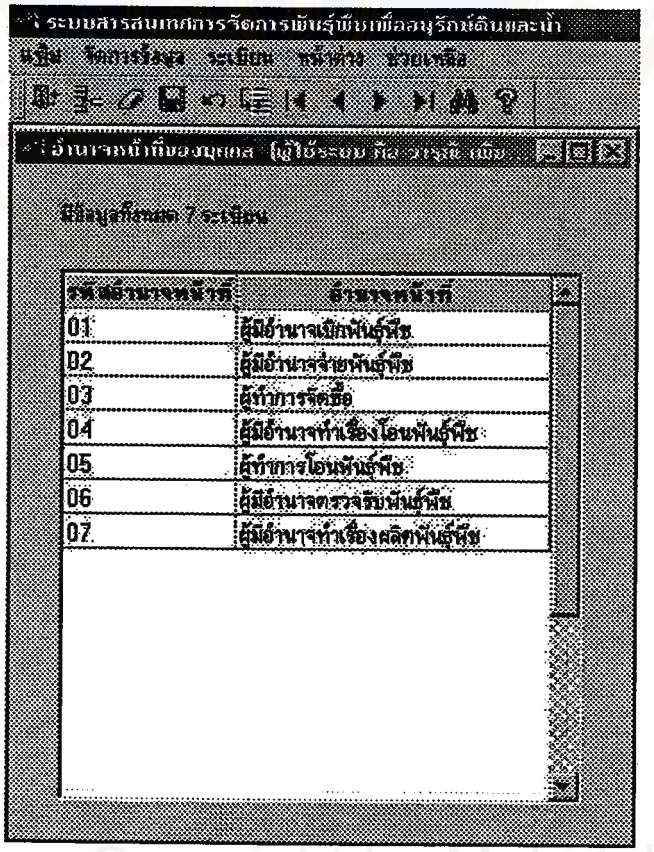


Figure C.4 w_authority

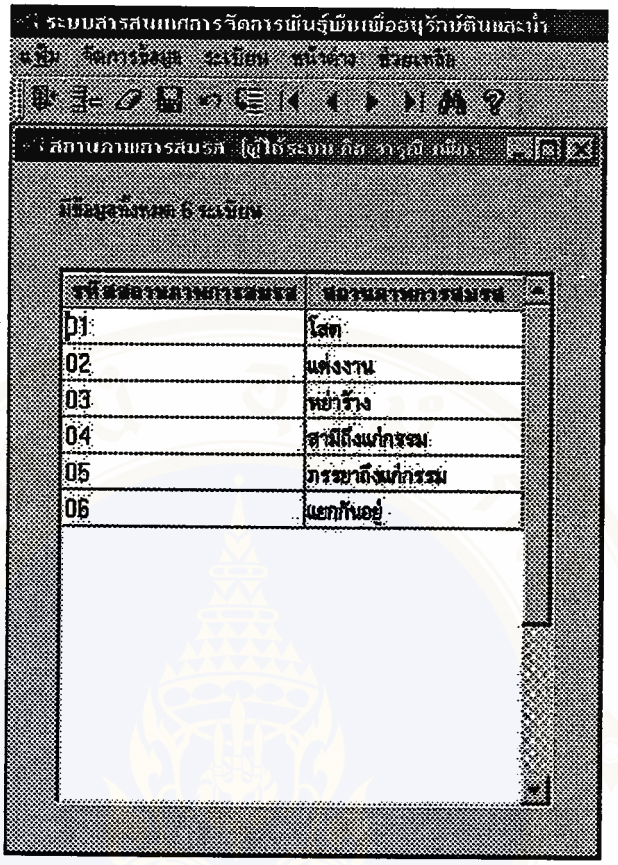



Figure C.5 w_status

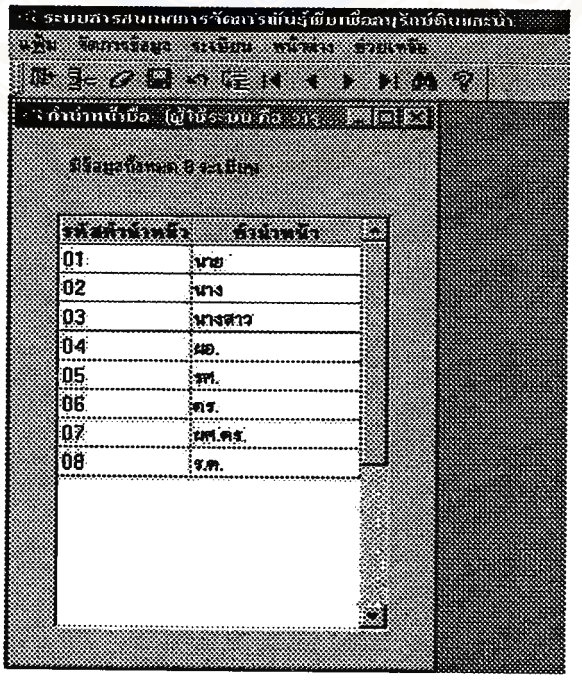



Figure C.6 w_pertit

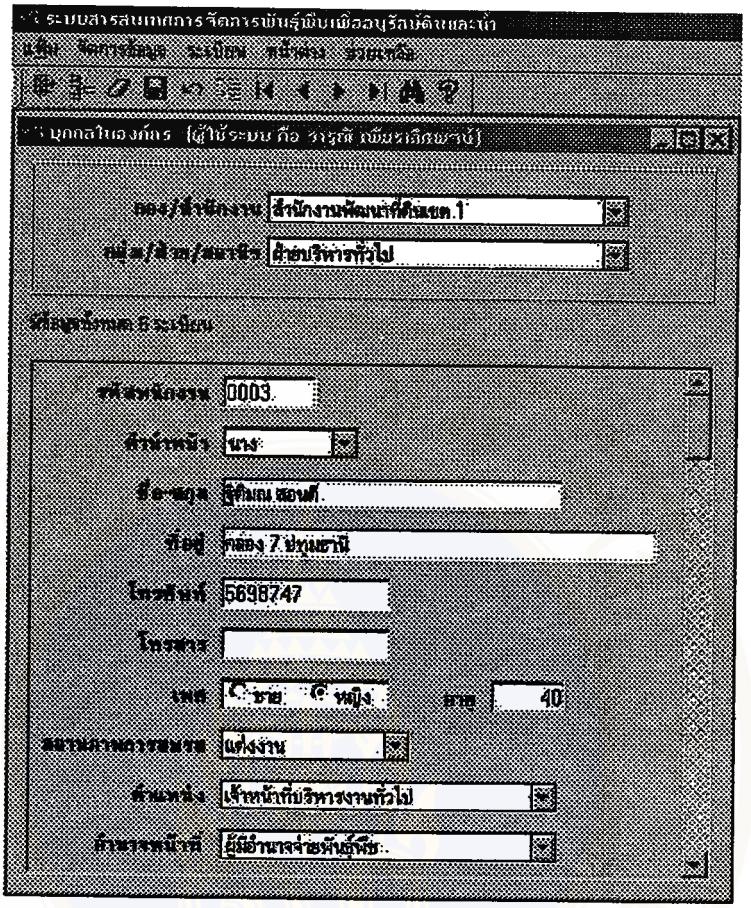



Figure C.7 w_employee

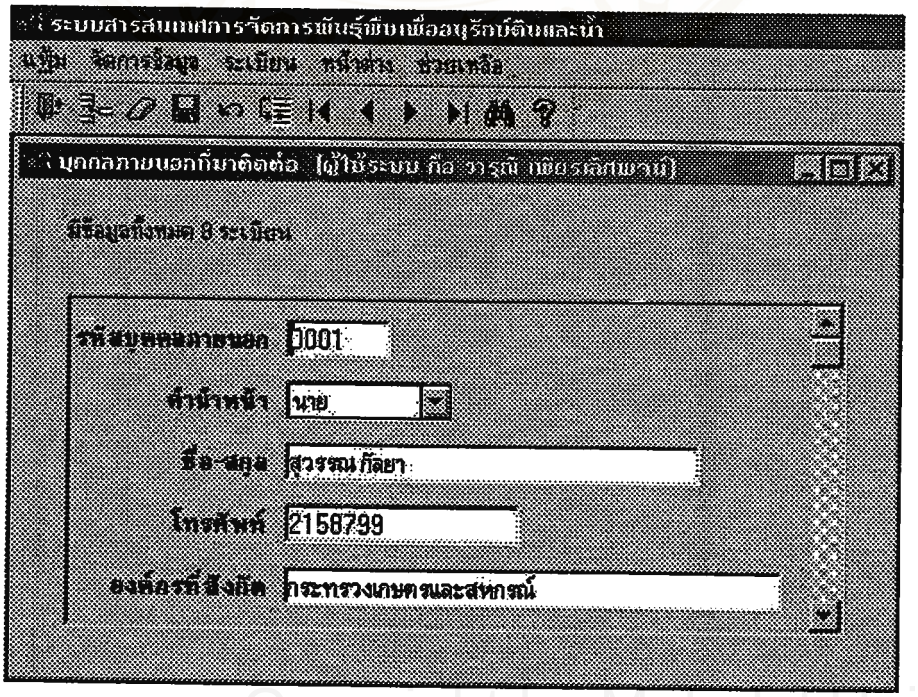



Figure C.8 w_person

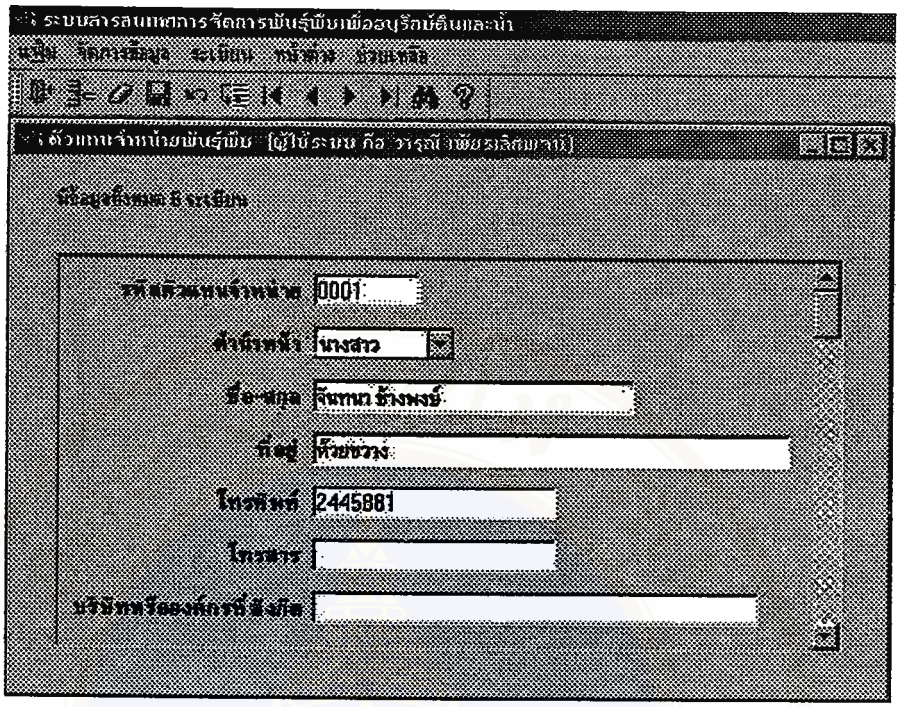



Figure C.9 w_vendor

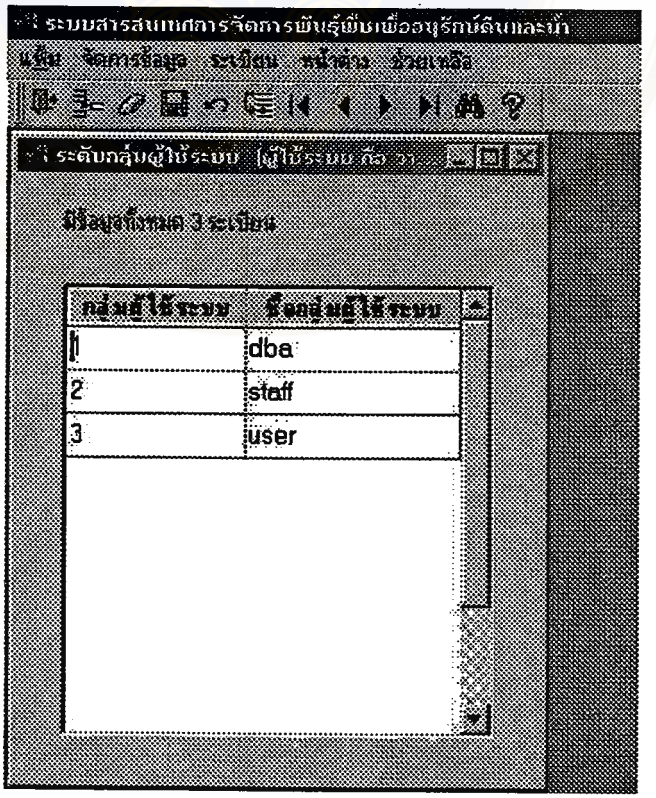



Figure C.10 w_gruser

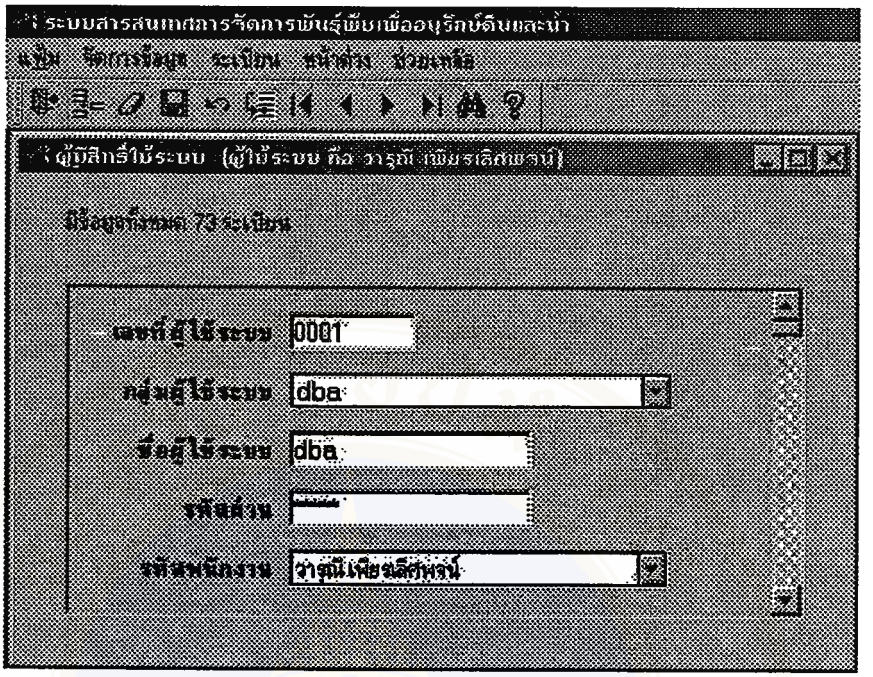



Figure C.11 w_user

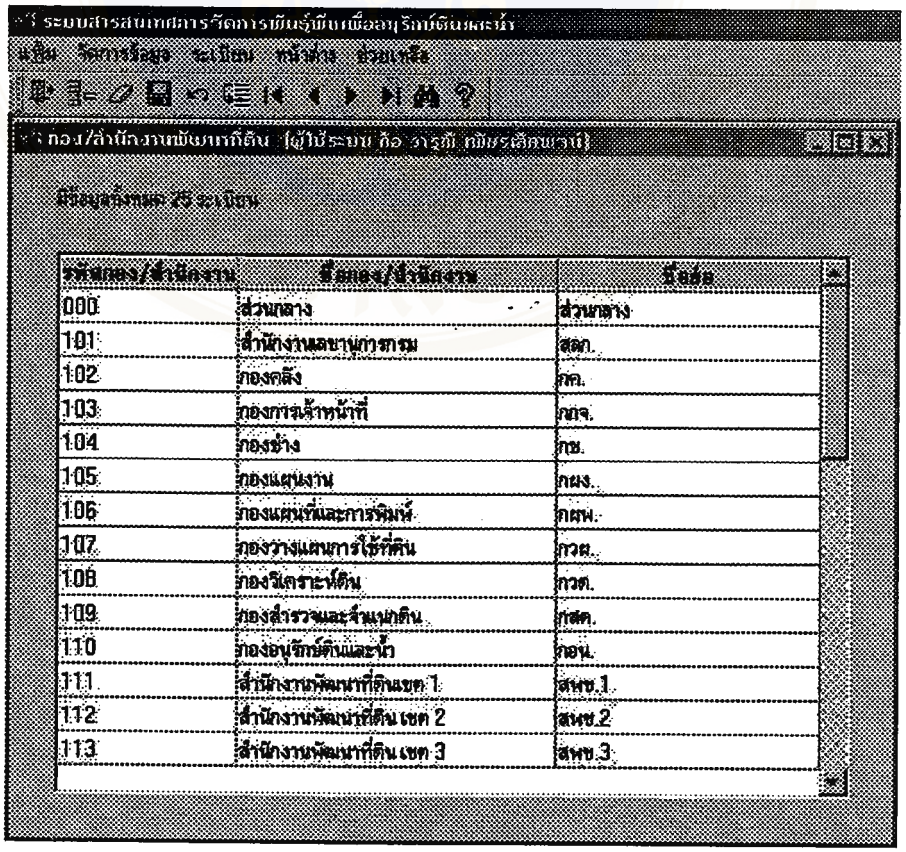



Figure C.12 w_division

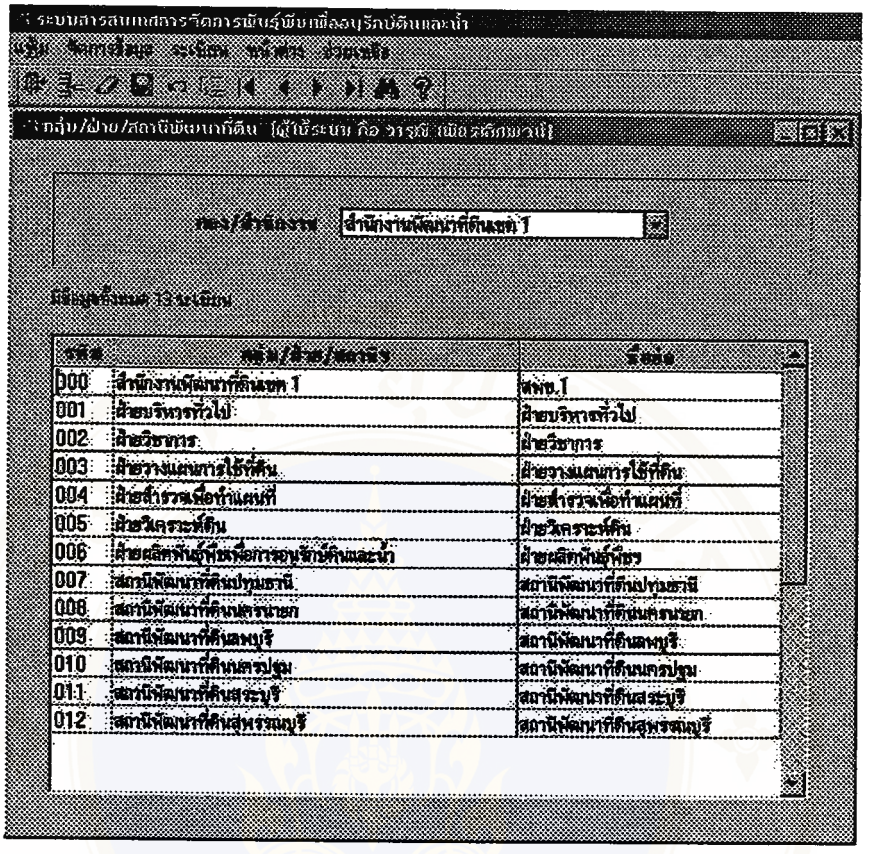



Figure C.13 w_section

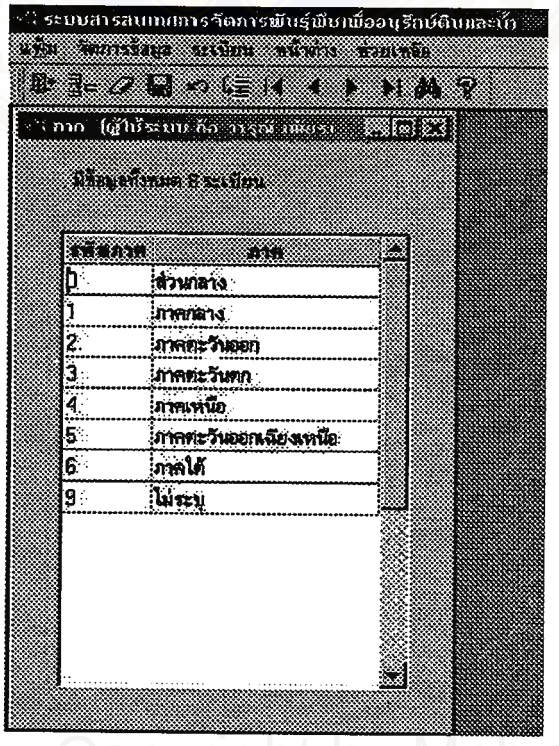



Figure C.14 w_region

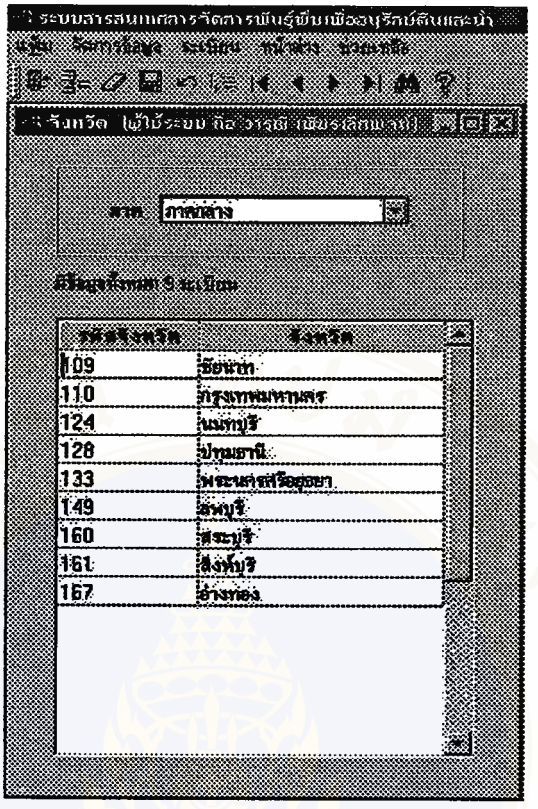



Figure C.15 w_province

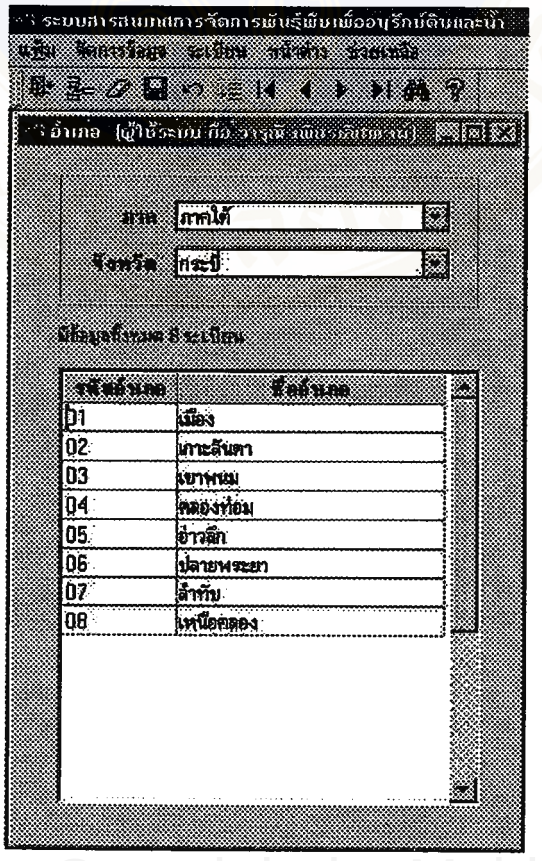



Figure C.16 w_amphoe

## 2. Transaction

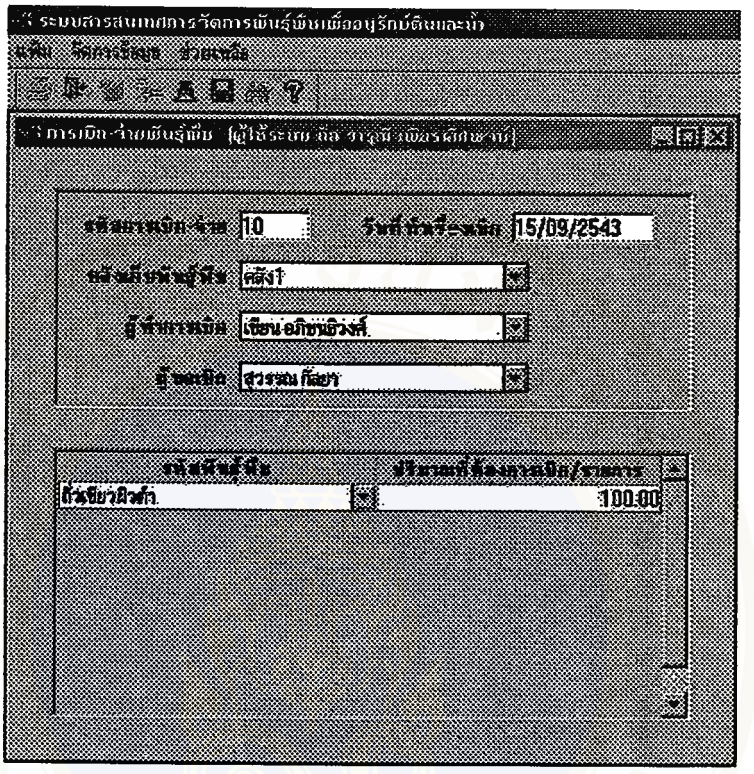



Figure C.17 w_trans_distribute

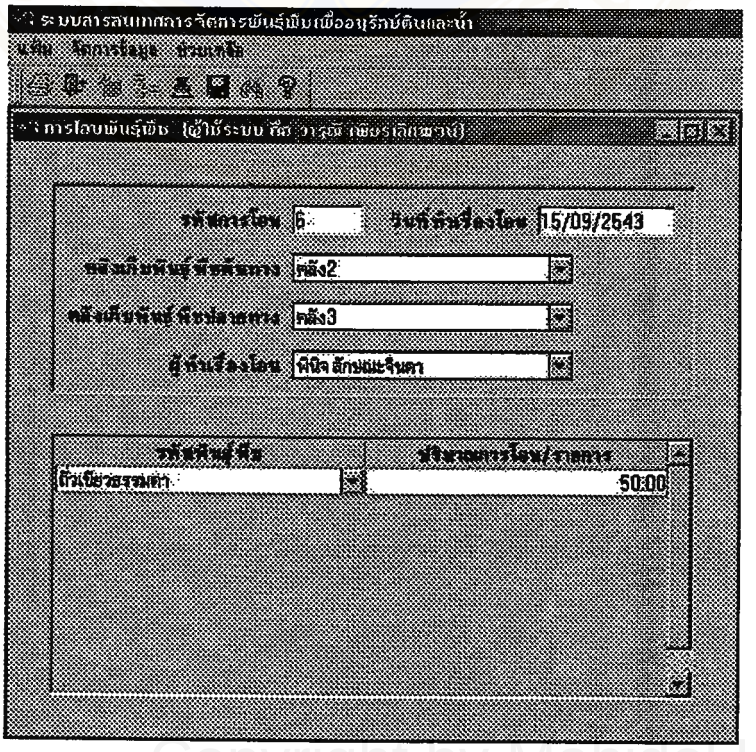



Figure C.18 w_trans_transfer

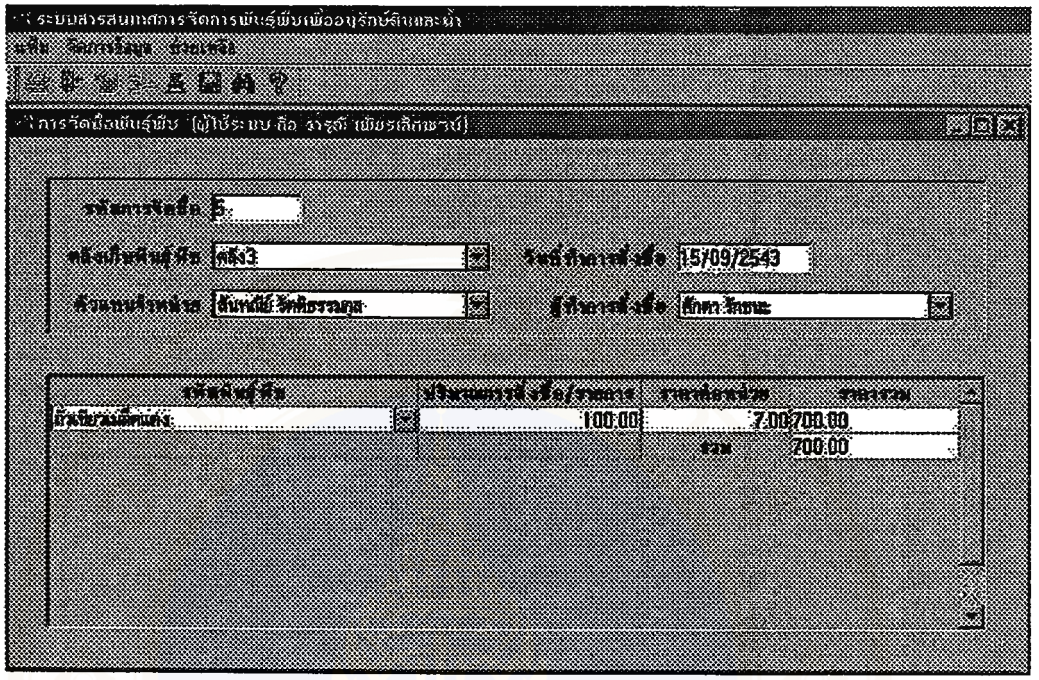



Figure C.19 w_trans_purchase

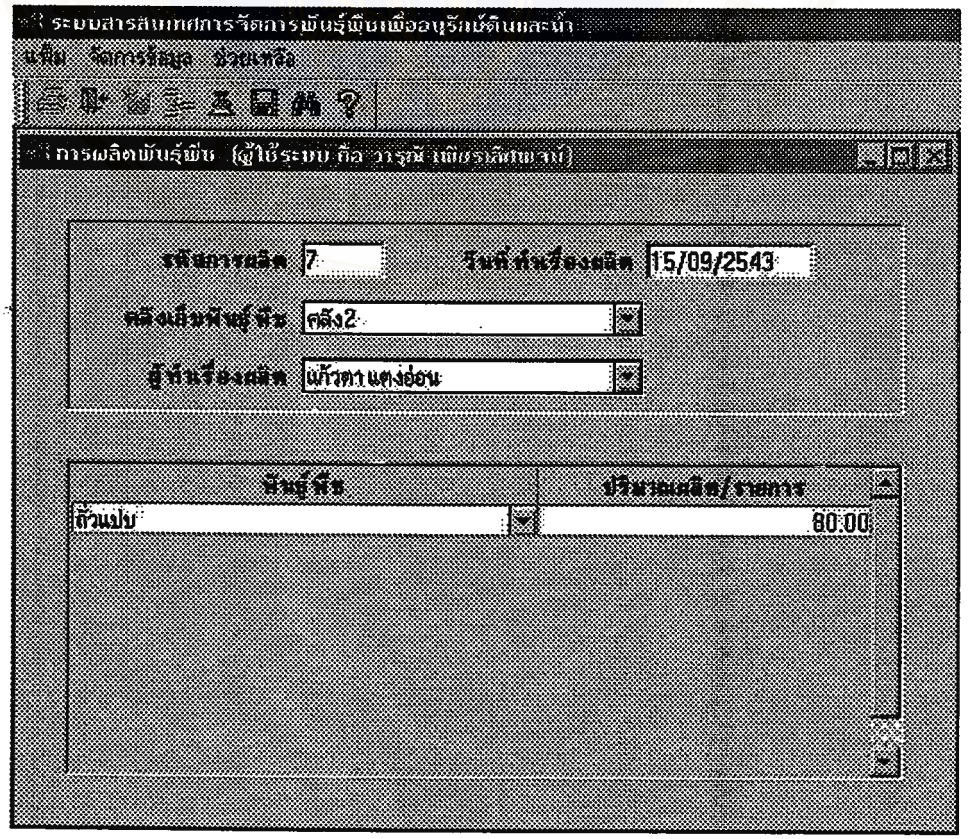



Figure C.20 w_trans_produce

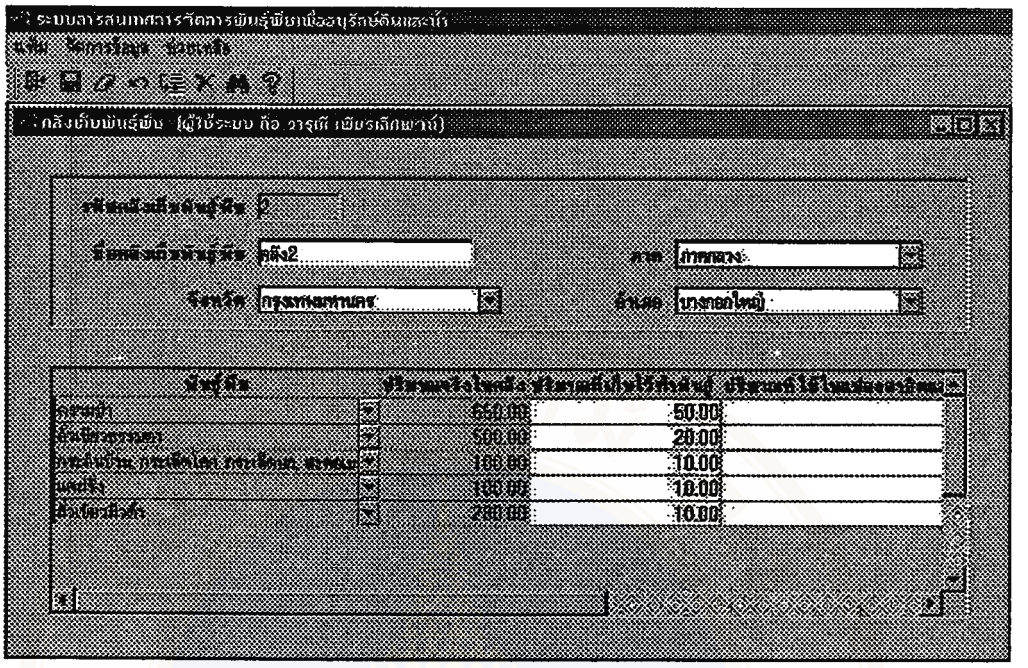


Figure C.21 w_trans_edit_stock

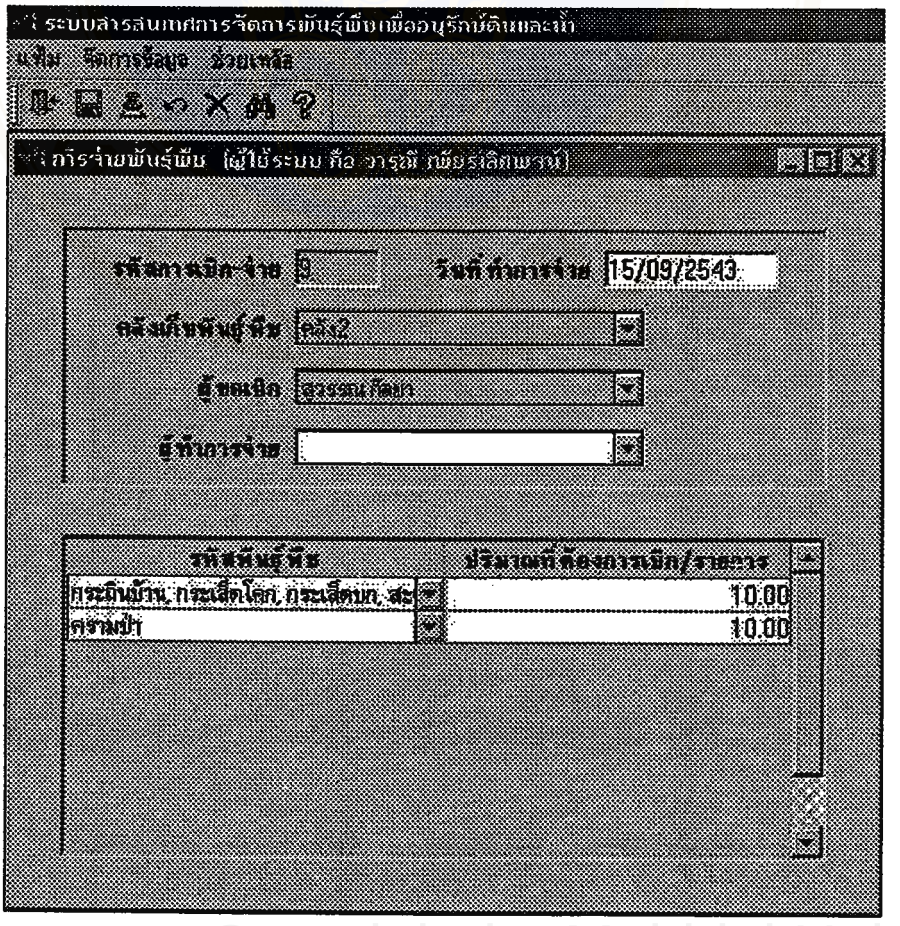


Figure C.22 w_trans_ctrl_dist

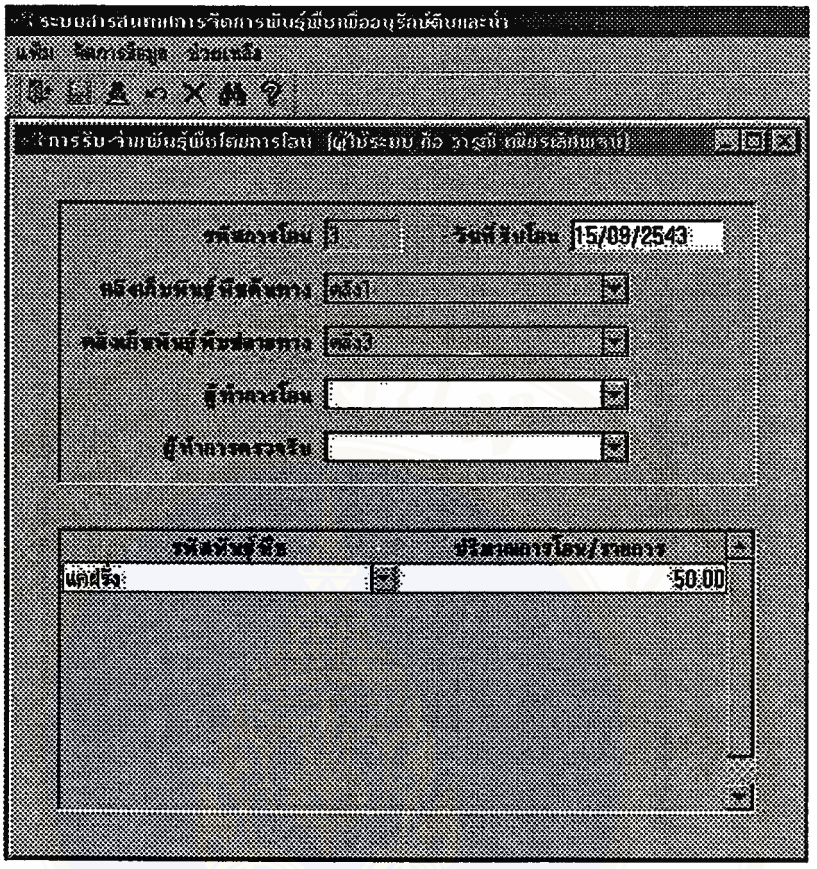



Figure C.23 w_trans_ctrl_transfer

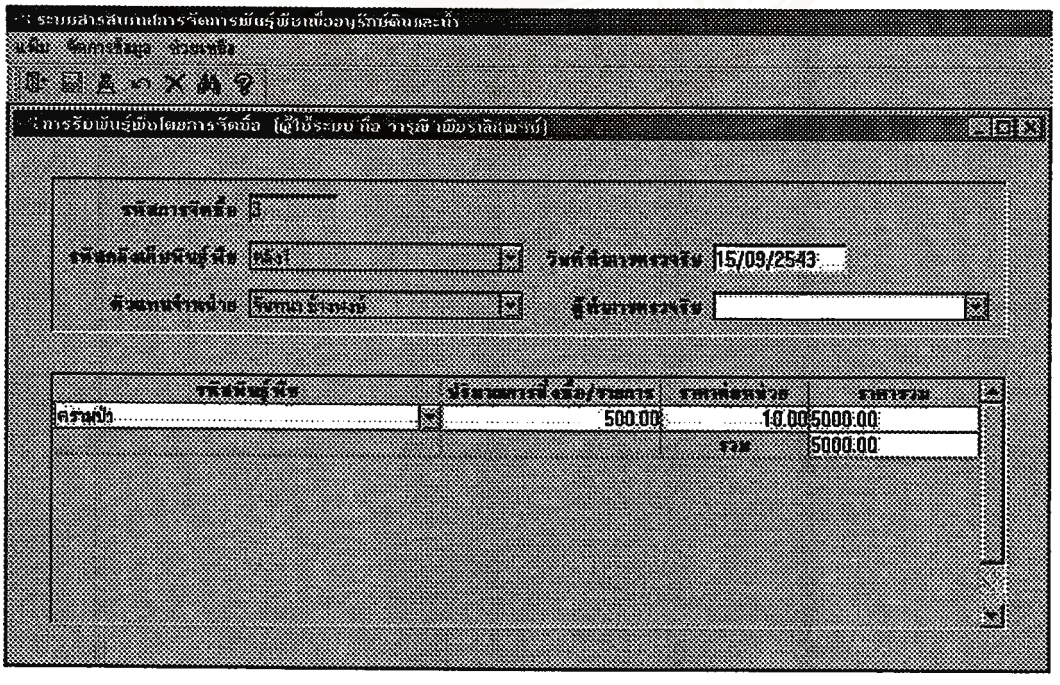



Figure C.24 w_trans_ctrl_purchase

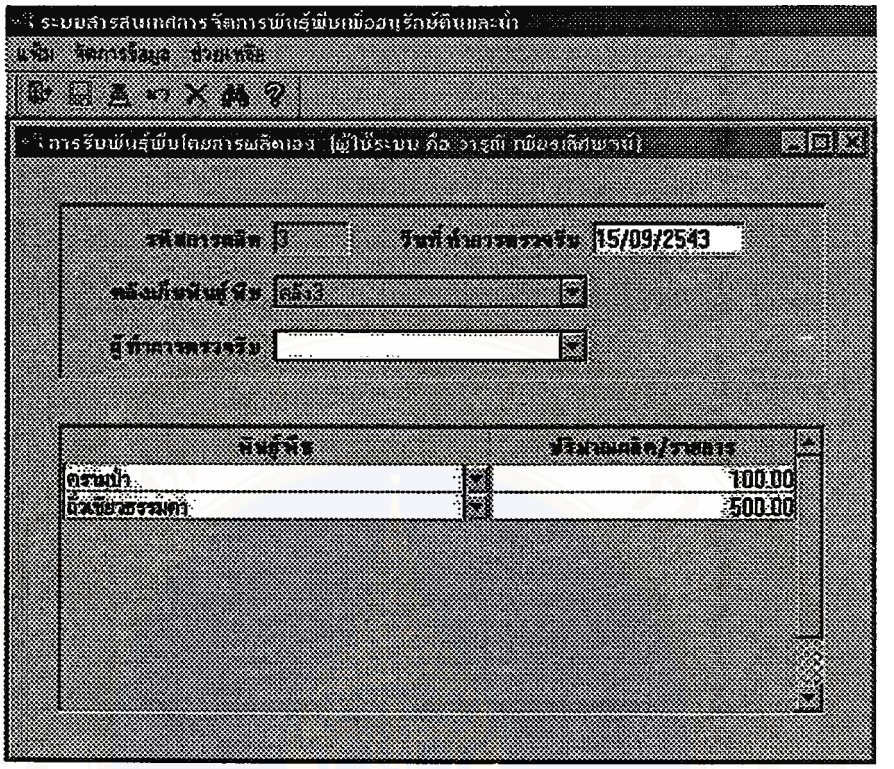



Figure C.25 w_trans_ctrl_produce

## 3. Query and Report

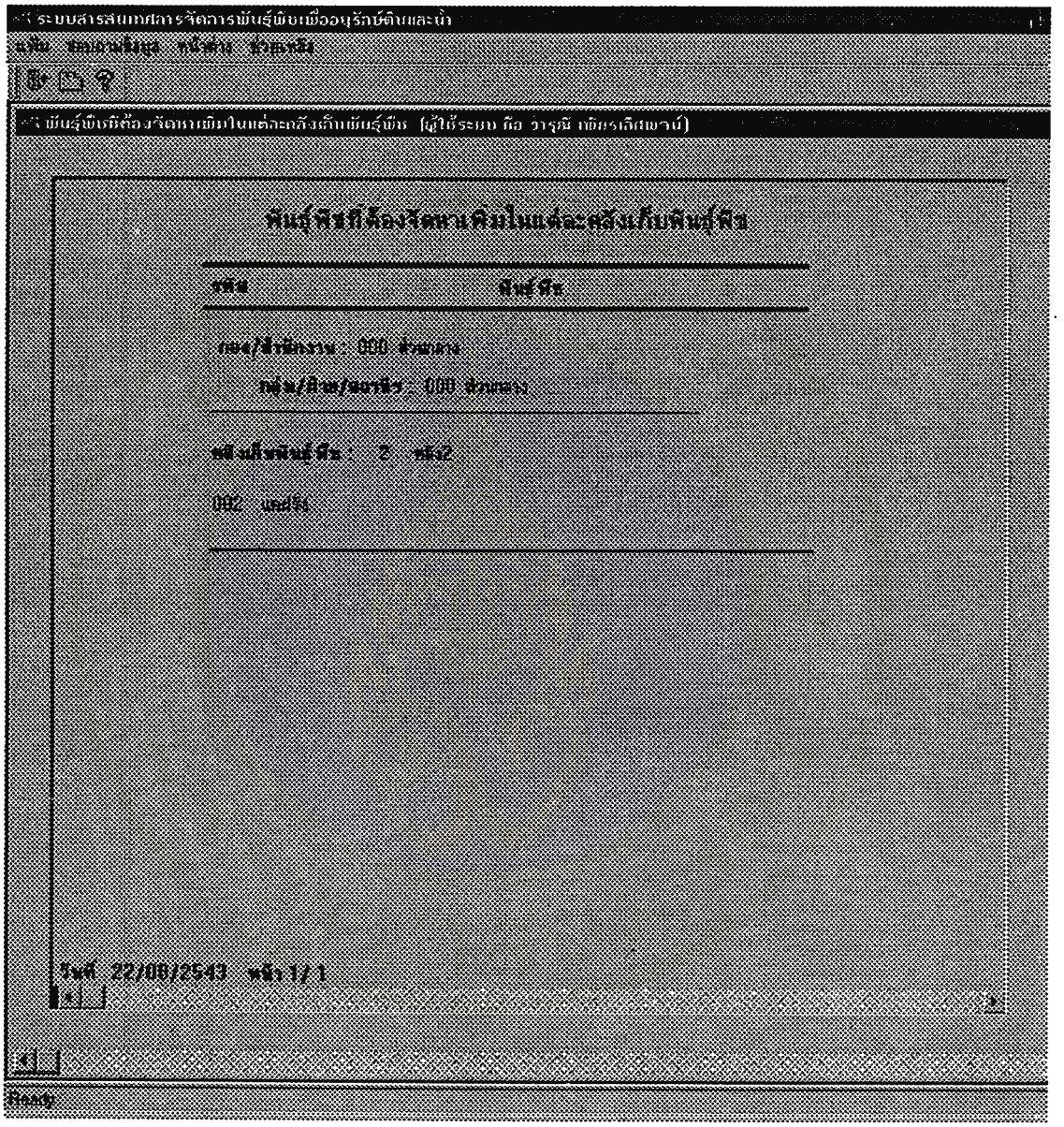


Figure C.26 w_query

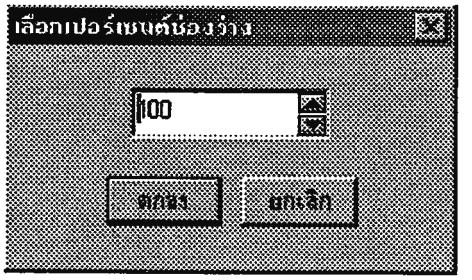


Figure C.27 w_graph_spacing

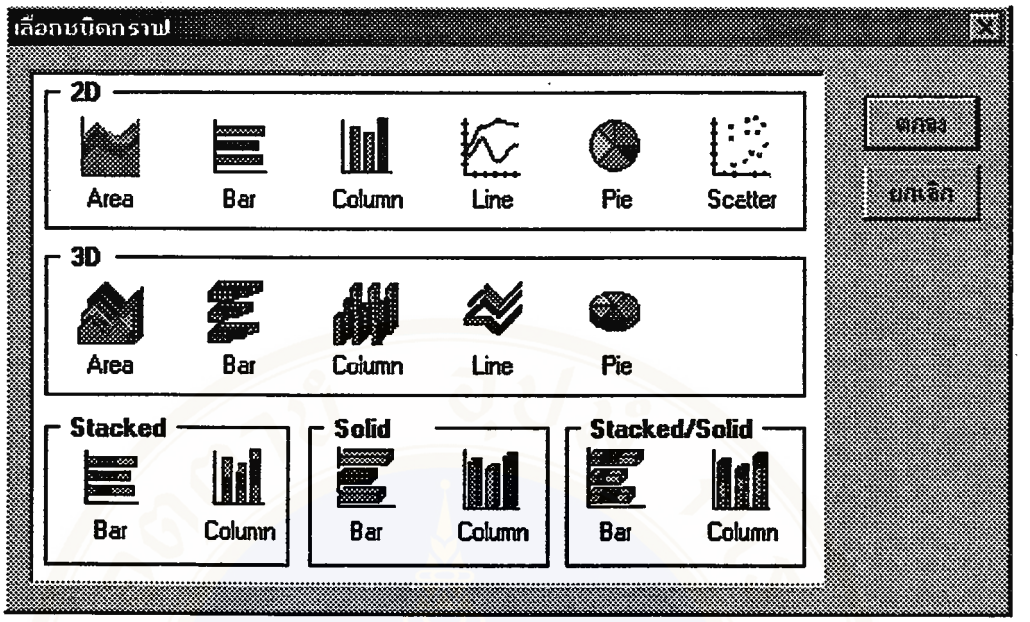



Figure C.28 w_graph_type

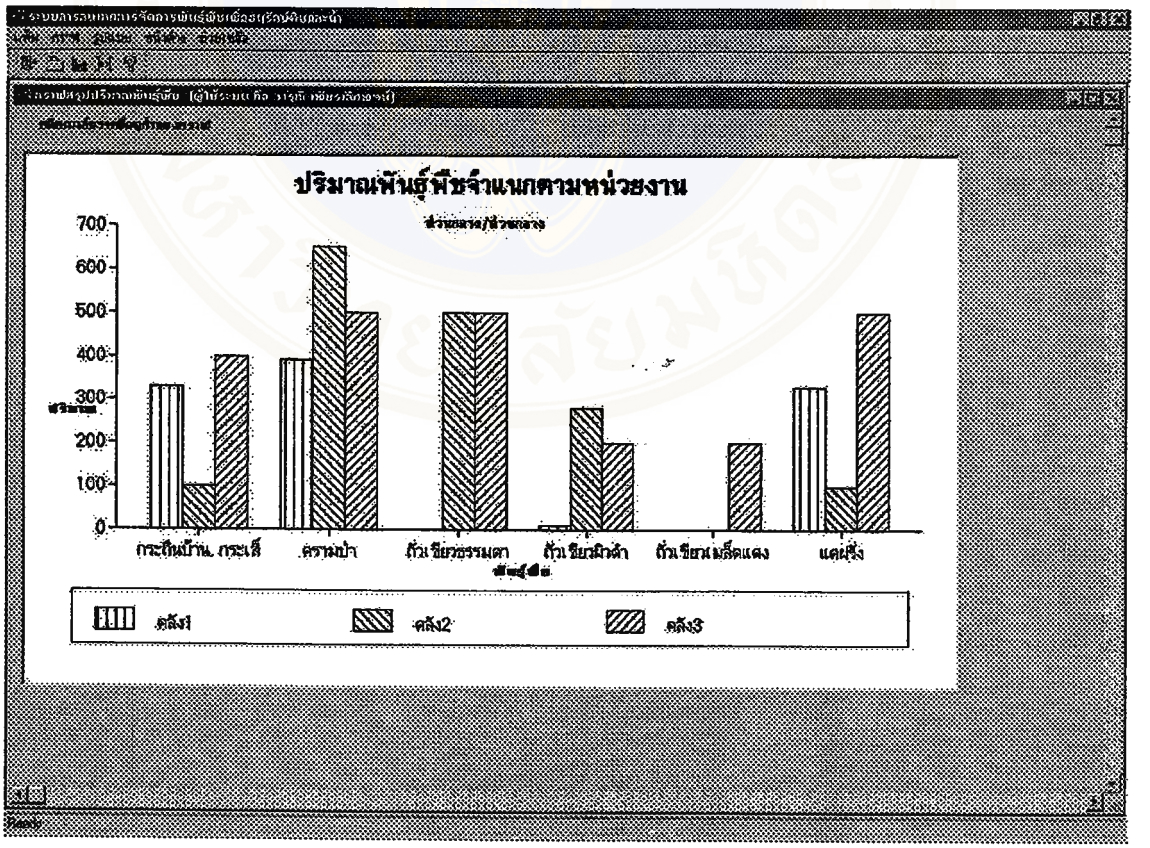



Figure C.29 w_graph

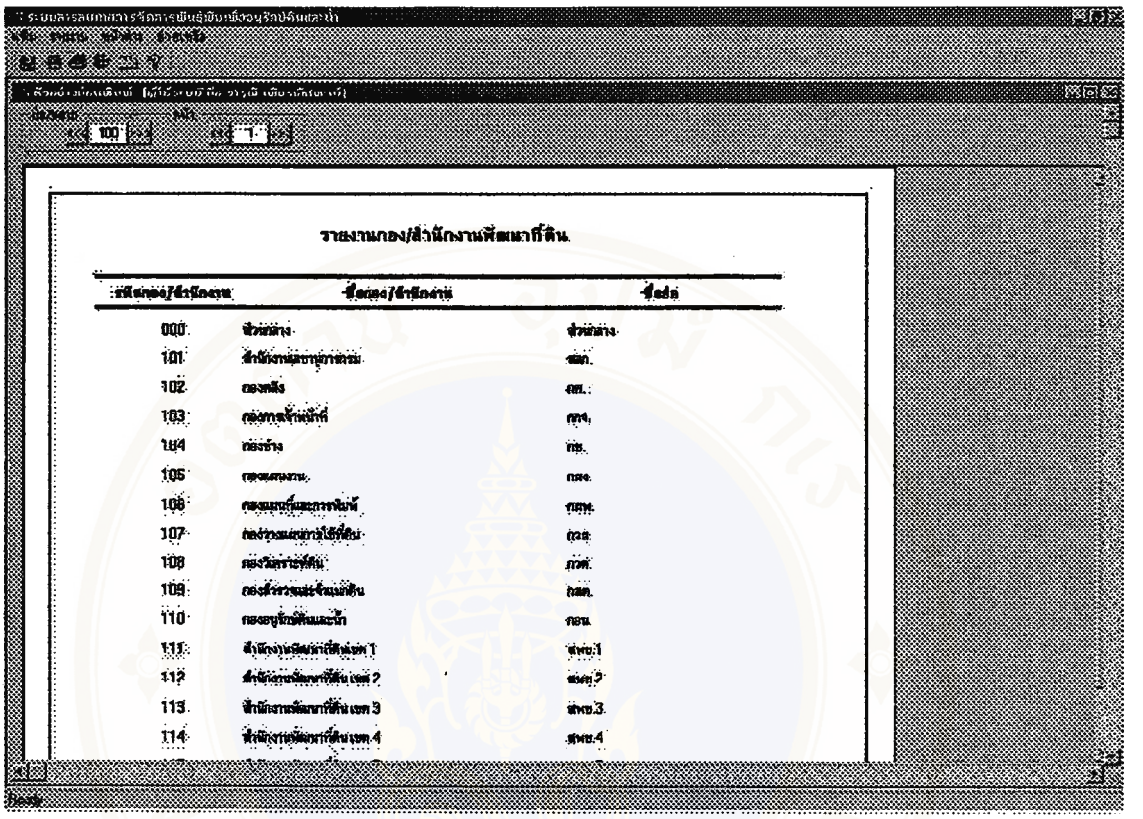



Figure C.30 w_preview

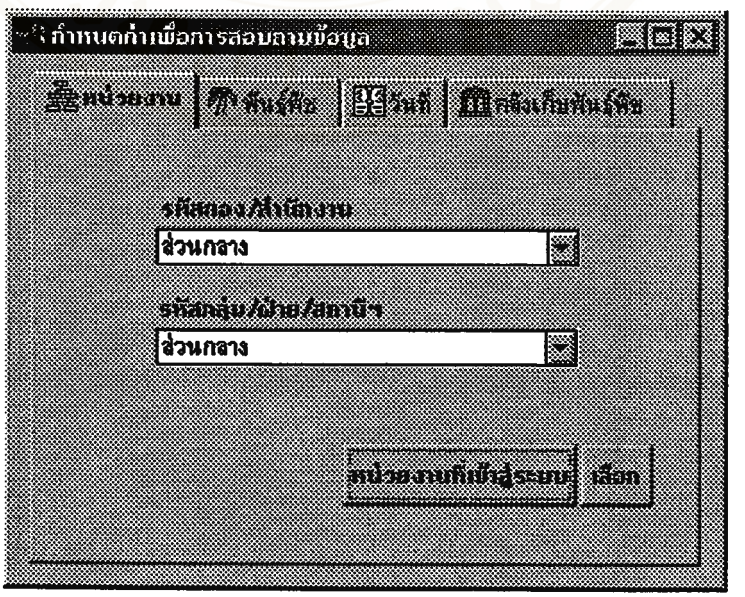



Figure C.31 w_set_argument

## 4. Shared Window

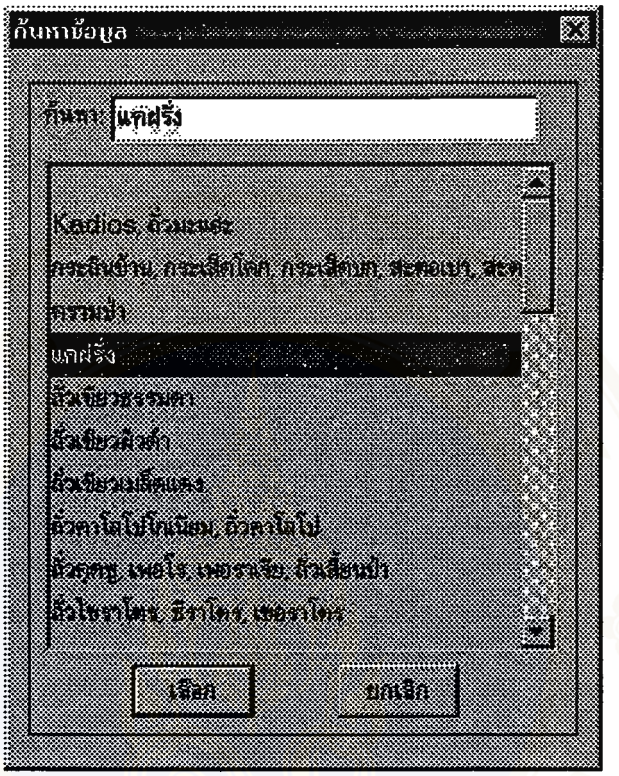



Figure C.32 w_search

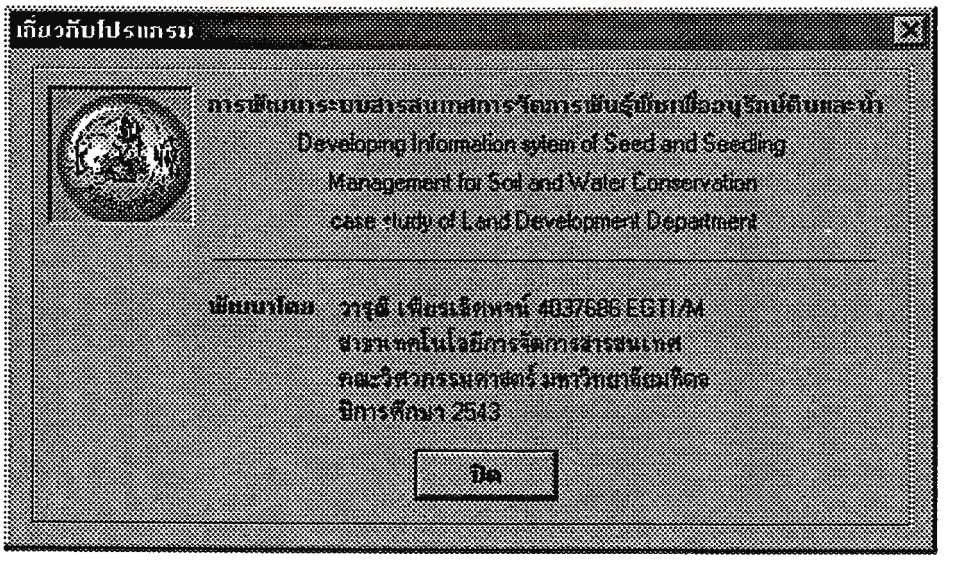



Figure C.33 w_about

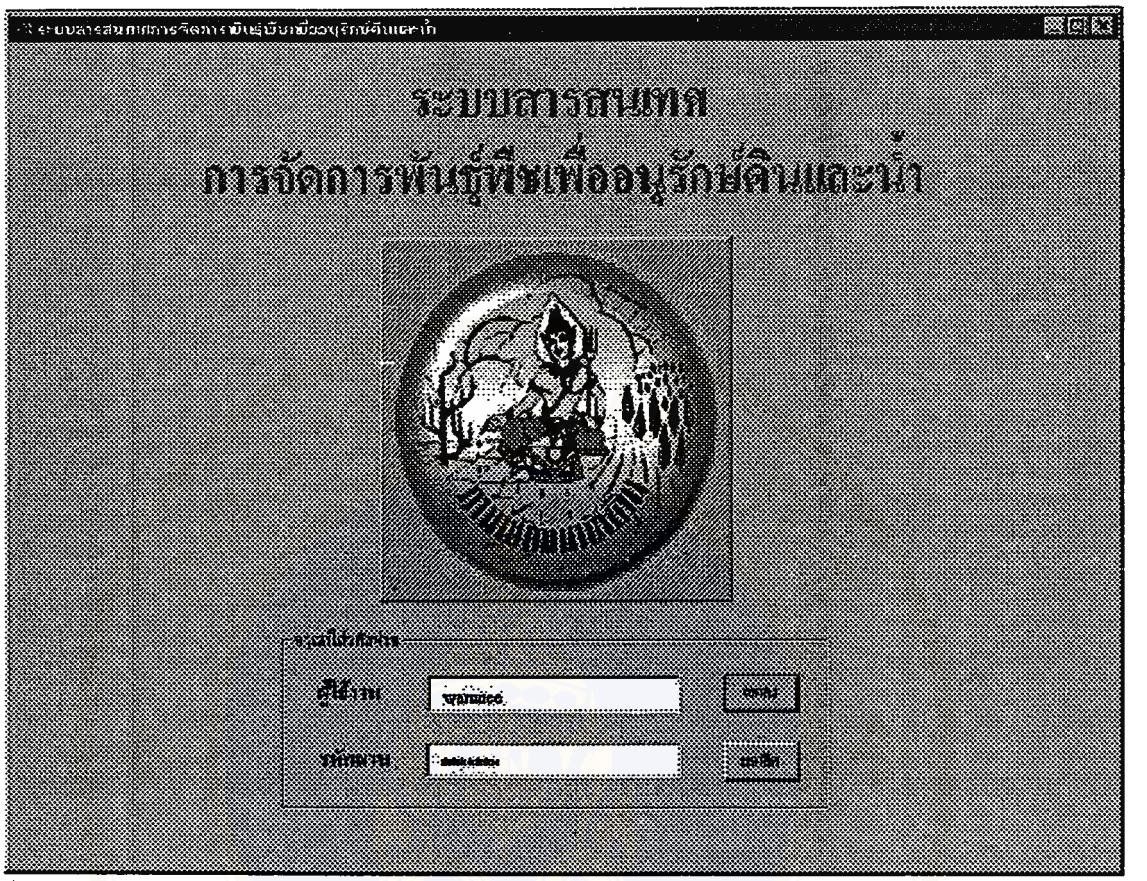



Figure C.34 w_login

# APPENDIX D

# PROGRAM SPECIFICATION

## 1. Data manipulation

These windows manipulate data in "BASIC DATA" data store. It has 2 ancestor windows, 16 descendent windows and a shared window.

| **[Window] w_seed from Window**<br>**[Description] ancestor window for data manipulation (descendent window that has 1 data window)** |
|---|

| **Controls** | *Control Name* | *Remarks* |
|---|---|---|
| Window | <w_seed> | Menu="m_code_base"; |
| Datawindow | dw_data1 | Tab=10; |
| Statictext | st_record | |

| **Declaration** | |
|---|---|
| Instance variable | boolean i_b_edit = false<br>int i_i_row_delete<br>long i_l_row, i_l_row_deleted<br>string i_s_max_id, is_emp_name |

| **Script** | *Type* | *Name* |
|---|---|---|
| w_seed | Event | Close<br>Closequery<br>Open<br>ue_delete ( )<br>ue_first ( )<br>ue_insert ( )<br>ue_last ( )<br>ue_next ( )<br>ue_open ( )<br>ue_previous ( )<br>ue_record ( string a_s_record )<br>ue_undelete ( )<br>ue_undo ( )<br>ue_update ( ) |
| dw_data1 | ControlEvent | Dberror<br>Retrieveend |

*Event: w_seed. Close*

```
Close(This)
```

*Event: w_seed. Closequery*

```
integer l_i_ret_code
IF dw_data1.DeletedCount() > 0 OR dw_data1.ModifiedCount() > 0 OR i_b_edit THEN
        l_i_ret_code = Messagebox("Ask for save", Question!, YesNoCancel!,1)
        CHOOSE CASE l_i_ret_code
        CASE 1 // Update
                TriggerEvent('ue_update')
                Return
        CASE 3 // Cancel
                return 1
```

```
        CASE ELSE // Don't want to update
                return
        END CHOOSE
ELSE
        return
END IF
```

*Event: w_seed. Open*

```
This.PostEvent ("ue_open")
```

*Event: w_seed. ue_delete ()*

```
integer l_i_ret_code
IF dw_data1.RowCount() > 0 then
        l_i_ret_code = MessageBox("Confirm to delete",Question!,YesNo!,2)
        IF l_i_ret_code = 1 THEN
                i_l_row_deleted = dw_data1.getrow()
                i_i_row_delete = dw_data1.deleterow(i_l_row_deleted)
                IF i_i_row_delete <> 1 THEN
                        Messagebox("Error","Can't delete")
                END IF
                dw_data1.setfocus()
                i_b_edit = true
        ELSE
                return
        END IF
END IF
```

*Event: w_seed. ue_first ()*

```
dw_data1.scrolltorow(1)
dw_data1.setfocus()
```

*Event: w_seed. ue_insert ()*

```
i_l_row = dw_data1.insertrow(dw_data1.rowcount()+1)
IF i_l_row > 0 then
        dw_data1.setrow(i_l_row)
        dw_data1.scrolltorow(i_l_row)
        dw_data1.setcolumn(1)
        dw_data1.setfocus()
        i_b_edit = true
ELSE
        MessageBox("Cannot insert row")
END IF
```

*Event: w_seed. ue_last ()*

```
dw_data1.scrolltorow(dw_data1.rowcount())
dw_data1.setfocus()
```

*Event: w_seed. ue_next ()*

```
dw_data1.scrollnextrow()
dw_data1.setfocus()
```

*Event: w_seed. ue_open ()*

```
dw_data1.settransobject(sqlca)
dw_data1.retrieve()
```

| *Event: w_seed. ue_previous ()* |
|---|
| dw_data1.scrollpriorrow()<br>dw_data1.setfocus() |
| *Event: w_seed. ue_record ( string a_s_record )* |
| st_record.text = a_s_record |
| *Event: w_seed. Ue_undelete ()* |
| Get the last row in the deleted buffer<br>Move the last row in the deleted buffer to the primary buffer<br>set focus to the row that was "restored" |
| *Event: w_seed. Ue_undo ()* |
| Get the original value of this row/column<br>Reset it to the original value<br>Reset the modified flag for this row |
| *Event: w_seed. Ue_update ()* |
| if dw_data1.update() = 1 then<br>commit;<br>messagebox("save")<br>i_b_edit = false<br>else<br>rollback;<br>messagebox("error")<br>end if |
| *ControlEvent: dw_data1. Dberror* |
| set sqldbcode, sqlerrtext, sqlsyntax, row, dw_name to error_data<br>OpenWithParm(w_db_error,error_data)<br>Return 1 |
| *ControlEvent: dw_data1. Retrieveeend* |
| Parent.Post Event ue_record("retrieved data "+string(This.RowCount())+" rows ") |

| **[Window] w_authority from w_seed** | Normal |
|---|---|
| [Description] manipulate data in AUTHORITY data store | |

| **Controls** | *Control Name* | *Remarks* |
|---|---|---|
| w_seed | <w_authority> | Title="Authority"; |
| w_seed`dw_data1 | Dw_data1 | Tab=10; DataObject="d_autho_grid"; |

| **Declaration** | |
|---|---|
| Instance variable | integer i_i_autho_id |

| **Script** | *Type* | *Name* |
|---|---|---|
| w_authority | Event | Open<br>ue_calc_id<br>ue_insert<br>ue_open<br>ue_search |

*Event: w_authority. Open*

```
Select user's name is using system.
this.title = "Authority" + "(user's name)"
```

*Event: w_authority. ue_calc_id*

```
select max(authority.autho_id) Into :i_s_max_id from authority;
if isnull(i_s_max_id) then
        i_s_max_id = '0'
end if
i_i_autho_id = Integer(i_s_max_id)
```

*Event: w_authority. ue_insert*

```
Call super::ue_insert
string   l_s_autho_id
long     l_l_len, l_l_i, l_l_chk

// Set '0' of id in the new row
i_i_autho_id++
i_s_max_id = String(i_i_autho_id)
l_l_len = len(i_s_max_id)
i_s_max_id = ''
if l_l_len <> 2 then
        i_s_max_id = ''
        l_l_len = 2 - l_l_len
        for l_l_i = 1 to l_l_len step 1
                i_s_max_id = i_s_max_id + '0'
        next
end if

// Generate Auto id
l_s_autho_id = i_s_max_id + string(i_i_autho_id)
l_l_chk = len(l_s_autho_id)

// Check orror of id
if l_l_chk > 2 then
        l_l_chk -= 2
        l_s_autho_id = Right(l_s_autho_id, 2)
        i_s_max_id = Left(i_s_max_id, len(i_s_max_id) - l_l_chk)
end if
dw_data1.setitem(i_l_row, "autho_id", l_s_autho_id)
dw_data1.setcolumn(2)
```

*Event: w_authority. ue_open*

```
Call super::ue_open
This.Post Event ue_calc_id()
```

*Event: w_authority. ue_search*

```
string   ls_sql, ls_search_name
long     ll_found

ls_sql = 'select name1 from authority'
OpenWithParm(w_search,ls_sql)
ls_search_name = Message.StringParm
if ls_search_name <> "" then
```

```
        ll_found = dw_data1.Find("name1 = '"+ls_search_name+"'", 1, dw_data1.RowCount())
        if ll_found > 0 THEN
                dw_data1.ScrollToRow(ll_found)
        else
                MessageBox("search", "not found")
        end if
end if
```

| [Window] w_division from w_seed | Normal |
|---|---|
| [Description] manipulate data in DIVISION data store | |

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| w_seed | <w_division> | Title="Division"; |
| w_seed`dw_data1 | Dw_data1 | Tab=10; DataObject="d_division_grid"; |

| Declaration | |
|---|---|
| Instance variable | integer i_i_div_id |

| Script | *Type* | *Name* |
|---|---|---|
| w_division | Event | Open<br>ue_calc_id<br>ue_insert<br>ue_open<br>ue_search |

*Event: w_division. Open*

```
Select user's name is using system.
this.title = "Division" + "(user's name)"
```

*Event: w_division. ue_calc_id*

```
select max(division.div_id) Into :i_s_max_id from division;
if isnull(i_s_max_id) then
        i_s_max_id = '0'
end if
i_i_div_id = Integer(i_s_max_id)
```

*Event: w_division. ue_insert*

```
Call super::ue_insert
string    l_s_div_id
long      l_l_len, l_l_i, l_l_chk

// Set '0' of id in the new row
i_i_div_id++
i_s_max_id = string(i_i_div_id)
l_l_len = len(i_s_max_id)
i_s_max_id = ''
if l_l_len <> 3 then
        i_s_max_id = ''
        l_l_len = 3 - l_l_len

        for l_l_i = 1 to l_l_len step 1
```

```
                i_s_max_id = i_s_max_id + '0'
        next
end if
// Generate Auto id
l_s_div_id = i_s_max_id + string(i_i_div_id)
l_l_chk = len(l_s_div_id)

// Check orror of id
if l_l_chk > 3 then
        l_l_chk -= 3
        l_s_div_id = right(l_s_div_id, 3)
        i_s_max_id = left(i_s_max_id, len(i_s_max_id) - l_l_chk)
end if
dw_data1.setitem(i_l_row, "div_id", l_s_div_id)
dw_data1.setcolumn(2)
```

*Event: w_division. ue_open*

```
Call super::ue_open
This.Post Event ue_calc_id()
```

*Event: w_division. ue_search*

```
string    ls_sql, ls_search_name
long      ll_found

ls_sql = 'select name1 from division'
OpenWithParm(w_search, ls_sql)
ls_search_name = Message.StringParm

if ls_search_name <> "" then
        ll_found = dw_data1.Find("name1 = '"+ls_search_name+"'", 1, dw_data1.RowCount())
        if ll_found > 0 THEN
                dw_data1.ScrollToRow(ll_found)
        else
                MessageBox("search", "not found")
        end if
end if
```

**[Window] w_gruser from w_seed** Normal
[Description] manipulate data in GRUSER data store

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| w_seed | <w_gruser> | Title="User group"; |
| w_seed`dw_data1 | Dw_data1 | Tab=10; DataObject="d_gruser_grid"; |

| Declaration | |
|---|---|
| Instance variable | integer i_i_group_id |

| Script | *Type* | *Name* |
|---|---|---|
| w_gruser | Event | Open<br>ue_calc_id<br>ue_insert |

| Script | *Type* | *Name* |
|---|---|---|
| | | ue_open<br>ue_search |

*Event: w_gruser. Open*

```
Select user's name is using system.
this.title = "User group" + "(user's name)"
```

*Event: w_gruser. ue_calc_id*

```
select max(gruser.group_id) Into :i_s_max_id from gruser;
if isnull(i_s_max_id) then
        i_s_max_id = '0'
end if
i_i_group_id = Integer(i_s_max_id)
```

*Event: w_gruser. ue_insert*

```
Call super::ue_insert
string    l_s_group_id
long      l_l_len, l_l_i, l_l_chk

// Set '0' of id in the new row
i_i_group_id++
i_s_max_id = String(i_i_group_id)
l_l_len = len(i_s_max_id)
i_s_max_id = ''
if l_l_len <> 1 then
        i_s_max_id = ''
        l_l_len = 1 - l_l_len
        for l_l_i = 1 to l_l_len step 1
                i_s_max_id = i_s_max_id + '0'
        next
end if

// Generate Auto id
l_s_group_id = i_s_max_id + string(i_i_group_id)
l_l_chk = len(l_s_group_id)

// Check orror of id
if l_l_chk > 1 then
        l_l_chk -= 1
        l_s_group_id = Right(l_s_group_id, 1)
        i_s_max_id = Left(i_s_max_id, len(i_s_max_id) - l_l_chk)
end if
dw_data1.setitem(i_l_row, "group_id", l_s_group_id)
dw_data1.setcolumn(2)
```

*Event: w_gruser. ue_open*

```
Call super::ue_open
This.Post Event ue_calc_id()
```

*Event: w_gruser. ue_search*

```
string    ls_sql, ls_search_name
long      ll_found
```

```
ls_sql = 'select group_name from gruser'
OpenWithParm(w_search,ls_sql)
ls_search_name = Message.StringParm

if ls_search_name <> "" then
        ll_found = dw_data1.Find("group_name = '"+ls_search_name+"'", 1, dw_data1.RowCount())
        if ll_found > 0 THEN
                dw_data1.ScrollToRow(ll_found)
        else
                MessageBox("search", "not found")
        end if
end if
```

**[Window] w_kind from w_seed** Normal
[Description] manipulate data in KIND data store

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| w_seed | <w_kind> | Title="Kind of plant"; |
| w_seed`dw_data1 | Dw_data1 | Tab=10; DataObject="d_kind_grid"; |

| Declaration | |
|---|---|
| Instance variable | integer i_i_kind_id |

| Script | *Type* | *Name* |
|---|---|---|
| w_kind | Event | Open<br>ue_calc_id<br>ue_insert<br>ue_open<br>ue_search |

*Event: w_kind. Open*

```
Select user's name is using system.
this.title = "Kind of plant" + "(user's name)"
```

*Event: w_kind. ue_calc_id*

```
select max(kind.kind_id) Into :i_s_max_id from kind;
if isnull(i_s_max_id) then
        i_s_max_id = '0'
end if
i_i_kind_id = Integer(i_s_max_id)
```

*Event: w_kind. ue_insert*

```
Call super::ue_insert
string    l_s_kind_id
long      l_l_len, l_l_i, l_l_chk

// Set '0' of id in the new row
i_i_kind_id++
i_s_max_id = string(i_i_kind_id)
l_l_len = len(i_s_max_id)
i_s_max_id = ''
```

```
if l_l_len <> 2 then
        i_s_max_id = ''
        l_l_len = 2 - l_l_len
        for l_l_i = 1 to l_l_len step 1
                i_s_max_id =  i_s_max_id + '0'
        next
end if

// Generate Auto id
l_s_kind_id = i_s_max_id + string(i_i_kind_id)
l_l_chk = len(l_s_kind_id)

// Check orror of id
if l_l_chk > 2 then
        l_l_chk -= 2
        l_s_kind_id = right(l_s_kind_id, 2)
        i_s_max_id = left(i_s_max_id, len(i_s_max_id)-l_l_chk)
end if
dw_data1.setitem(i_l_row, "kind_id", l_s_kind_id)
dw_data1.setcolumn(2)
```

*Event: w_kind. ue_open*

```
Call super::ue_open
This.Post Event ue_calc_id()
```

*Event: w_kind. ue_search*

```
string    ls_sql, ls_search_name
long      ll_found

ls_sql = 'select name1 from kind'
OpenWithParm(w_search,ls_sql)
ls_search_name = Message.StringParm

if ls_search_name <> "" then
        ll_found = dw_data1.Find("name1 = '"+ls_search_name+"'", 1, dw_data1.RowCount())
        if ll_found > 0 THEN
                dw_data1.ScrollToRow(ll_found)
        else
                MessageBox("search", "not found")
        end if
end if
```

**[Window] w_person from w_seed** Normal
[Description] manipulate data in PERSON data store

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| w_seed | <w_person> | Title="Contact person"; |
| w_seed`dw_data1 | Dw_data1 | Tab=10; DataObject="d_person_ff"; |

| Declaration | |
|---|---|
| Instance variable | integer i_i_per_id |

| Script | *Type* | *Name* |
|---|---|---|
| w_person | Event | Open<br>ue_calc_id<br>ue_insert<br>ue_open<br>ue_search |

*Event: w_person. Open*

```
Select user's name is using system.
this.title = "Contact person" + "(user's name)"
```

*Event: w_person. ue_calc_id*

```
select max(person.per_id) Into :i_s_max_id from person;
if isnull(i_s_max_id) then
        i_s_max_id = '0'
end if
i_i_per_id = Integer(i_s_max_id)
```

*Event: w_person. ue_insert*

```
Call super::ue_insert
string    l_s_per_id
long      l_l_len, l_l_i, l_l_chk

// Set '0' of id in the new row
i_i_per_id++
i_s_max_id = string(i_i_per_id)
l_l_len = len(i_s_max_id)
i_s_max_id = ''

if l_l_len <> 4 then
        i_s_max_id = ''
        l_l_len = 4 - l_l_len
        for l_l_i = 1 to l_l_len step 1
                i_s_max_id = i_s_max_id + '0'
        next
end if

// Generate Auto id
l_s_per_id = i_s_max_id + string(i_i_per_id)
l_l_chk = len(l_s_per_id)

// Check orror of id
if l_l_chk > 4 then
        l_l_chk -= 4
        l_s_per_id = right(l_s_per_id, 4)
        i_s_max_id = left(i_s_max_id, len(i_s_max_id) - l_l_chk)
end if

dw_data1.setitem(i_l_row, "per_id", l_s_per_id)
dw_data1.setcolumn(2)
```

*Event: w_person. ue_open*

```
Call super::ue_open
This.Post Event ue_calc_id()
```

*Event: w_person. ue_search*

```
string    ls_sql, ls_search_name
long      ll_found

ls_sql = 'select name from person'
OpenWithParm(w_search,ls_sql)
ls_search_name = Message.StringParm

if ls_search_name <> "" then
        ll_found = dw_data1.Find("name = '"+ls_search_name+"'", 1, dw_data1.RowCount())
        if ll_found > 0 THEN
                dw_data1.ScrollToRow(ll_found)
        else
                MessageBox("search", "not found")
        end if
end if
```

**[Window] w_pertit from w_seed** Normal
[Description] manipulate data in PERTIT data store

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| w_seed | <w_pertit> | Title="Pertit"; |
| w_seed`dw_data1 | Dw_data1 | Tab=10; DataObject="d_pertit_grid"; |

| Declaration | |
|---|---|
| Instance variable | integer i_i_pertit_id |

| Script | *Type* | *Name* |
|---|---|---|
| w_pertit | Event | Open<br>ue_calc_id<br>ue_insert<br>ue_open<br>ue_search |

*Event: w_pertit. Open*

```
Select user's name is using system.
this.title = "Pertit" + "(user's name)"
```

*Event: w_pertit. ue_calc_id*

```
select max(pertit.pertit_id) Into :i_s_max_id from pertit;

if isnull(i_s_max_id) then
        i_s_max_id = '0'
end if

i_i_pertit_id = Integer(i_s_max_id)
```

*Event: w_pertit. ue_insert*

```
Call super::ue_insert
string    l_s_pertit_id
long      l_l_len, l_l_i, l_l_chk
// Set '0' of id in the new row
i_i_pertit_id++
i_s_max_id = String(i_i_pertit_id)
l_l_len = len(i_s_max_id)
i_s_max_id = ''
if l_l_len <> 2 then
        i_s_max_id = ''
        l_l_len = 2 - l_l_len
        for l_l_i = 1 to l_l_len step 1
                i_s_max_id = i_s_max_id + '0'
        next
end if
// Generate Auto id
l_s_pertit_id = i_s_max_id + string(i_i_pertit_id)
l_l_chk = len(l_s_pertit_id)

// Check orror of id
if l_l_chk > 2 then
        l_l_chk -= 2
        l_s_pertit_id = Right(l_s_pertit_id, 2)
        i_s_max_id = Left(i_s_max_id, len(i_s_max_id) - l_l_chk)
end if

dw_data1.setitem(i_l_row, "pertit_id", l_s_pertit_id)
dw_data1.setcolumn(2)
```

*Event: w_pertit. ue_open*

```
Call super::ue_open
This.Post Event ue_calc_id()
```

*Event: w_pertit. ue_search*

```
string    ls_sql, ls_search_name
long      ll_found

ls_sql = 'select name1 from pertit'
OpenWithParm(w_search,ls_sql)
ls_search_name = Message.StringParm

if ls_search_name <> "" then
        ll_found = dw_data1.Find("name1 = '"+ls_search_name+"'", 1, dw_data1.RowCount())
        if ll_found > 0 THEN
                dw_data1.ScrollToRow(ll_found)
        else
                MessageBox("search", "not found")
        end if
end if
```

| [Window] w_plant from w_seed | Normal |
|---|---|
| [Description] manipulate data in PLANT data store | |

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| w_seed | <w_plant> | Title="Plant"; |
| w_seed`dw_data1 | Dw_data1 | Tab=10; DataObject="d_plant_ff"; |

| Declaration | |
|---|---|
| Instance variable | integer i_i_plant_id |

| Script | *Type* | *Name* |
|---|---|---|
| w_plant | Event | Open<br>ue_calc_id<br>ue_insert<br>ue_open<br>ue_search |

*Event: w_plant. Open*

```
Select user's name is using system.
this.title = "Plant" + "(user's name)"
```

*Event: w_plant. ue_calc_id*

```
select max(plant.plant_id) Into :i_s_max_id from plant;
if isnull(i_s_max_id) then
        i_s_max_id = '0'
end if
i_i_plant_id = Integer(i_s_max_id)
```

*Event: w_plant. ue_insert*

```
Call super::ue_insert
string    l_s_plant_id
long      l_l_len, l_l_i, l_l_chk

// Set '0' of id in the new row
i_i_plant_id++
i_s_max_id = String(i_i_plant_id)
l_l_len = len(i_s_max_id)
i_s_max_id = ''

if l_l_len <> 3 then
        i_s_max_id = ''
        l_l_len = 3 - l_l_len
        for l_l_i = 1 to l_l_len step 1
                i_s_max_id = i_s_max_id + '0'
        next
end if

// Generate Auto id
l_s_plant_id = i_s_max_id + string(i_i_plant_id)
l_l_chk = len(l_s_plant_id)

// Check orror of id
if l_l_chk > 3 then
        l_l_chk -= 3
```

```
        l_s_plant_id = Right(l_s_plant_id, 3)
        i_s_max_id = Left(i_s_max_id, len(i_s_max_id) - l_l_chk)
end if
dw_data1.setitem(i_l_row, "plant_id", l_s_plant_id)
dw_data1.setcolumn(2)
```

*Event: w_plant. ue_open*

```
Call super::ue_open
This.Post Event ue_calc_id()
```

*Event: w_plant. ue_search*

```
string    ls_sql, ls_search_name
long      ll_found

ls_sql = 'select local_name from plant'
OpenWithParm(w_search,ls_sql)
ls_search_name = Message.StringParm

if ls_search_name <> "" then
        ll_found = dw_data1.Find("local_name = '"+ls_search_name+"'", 1, dw_data1.RowCount())
        if ll_found > 0 THEN
                dw_data1.ScrollToRow(ll_found)
        else
                MessageBox("search", "not found")
        end if
end if
```

**[Window] w_position from w_seed** Normal
[Description] manipulate data in POSITION data store

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| w_seed | <w_position> | Title="Position"; |
| w_seed`dw_data1 | Dw_data1 | Tab=10; DataObject="d_postn_gird"; |

| Declaration | |
|---|---|
| Instance variable | integer i_i_postn_id |

| Script | *Type* | *Name* |
|---|---|---|
| w_position | Event | Open<br>ue_calc_id<br>ue_insert<br>ue_open<br>ue_search |

*Event: w_position. Open*

```
Select user's name is using system.
this.title = "Position" + "(user's name)"
```

*Event: w_position. ue_calc_id*

```
select max(position.postn_id) Into :i_s_max_id from position;
if isnull(i_s_max_id) then
        i_s_max_id = '0'
end if
i_i_postn_id = Integer(i_s_max_id)
```

*Event: w_position. ue_insert*

```
Call super::ue_insert
string   l_s_postn_id
long     l_l_len, l_l_i, l_l_chk

// Set '0' of id in the new row
i_i_postn_id++
i_s_max_id = String(i_i_postn_id)
l_l_len = len(i_s_max_id)
i_s_max_id = ''

if l_l_len <> 2 then
        i_s_max_id = ''
        l_l_len = 2 - l_l_len
        for l_l_i = 1 to l_l_len step 1
                i_s_max_id = i_s_max_id + '0'
        next
end if
// Generate Auto id
l_s_postn_id = i_s_max_id + string(i_i_postn_id)
l_l_chk = len(l_s_postn_id)

// Check orror of id
if l_l_chk > 2 then
        l_l_chk -= 2
        l_s_postn_id = Right(l_s_postn_id, 2)
        i_s_max_id = Left(i_s_max_id, len(i_s_max_id) - l_l_chk)
end if

dw_data1.setitem(i_l_row, "postn_id", l_s_postn_id)
dw_data1.setcolumn(2)
```

*Event: w_position. ue_open*

```
Call super::ue_open
This.Post Event ue_calc_id()
```

*Event: w_position. ue_search*

```
string   ls_sql, ls_search_name
long     ll_found

ls_sql = 'select name1 from position'
OpenWithParm(w_search,ls_sql)
ls_search_name = Message.StringParm

if ls_search_name <> "" then
        ll_found = dw_data1.Find("name1 = '"+ls_search_name+"'", 1, dw_data1.RowCount())
        if ll_found > 0 THEN
                dw_data1.ScrollToRow(ll_found)
```

```
        else
                MessageBox("search", "not found")
        end if
end if
```

**[Window] w_region from w_seed** Normal
[Description] manipulate data in REGION data store

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| w_seed | <w_region> | Title="Region"; |
| w_seed`dw_data1 | Dw_data1 | Tab=10; DataObject="d_region_grid"; |

| Declaration | |
|---|---|
| Instance variable | integer i_i_region_id |

| Script | *Type* | *Name* |
|---|---|---|
| w_region | Event | Open<br>ue_calc_id<br>ue_insert<br>ue_open<br>ue_search |

*Event: w_region. Open*

```
Select user's name is using system.
this.title = "Region" + "(user's name)"
```

*Event: w_region. ue_calc_id*

```
select max(region.region_id) Into :i_s_max_id from region;
if isnull(i_s_max_id) then
        i_s_max_id = '0'
end if
i_i_region_id = Integer(i_s_max_id)
```

*Event: w_region. ue_insert*

```
Call super::ue_insert
string    l_s_region_id
long      l_l_len, l_l_i, l_l_chk

// Set '0' of id in the new row
i_i_region_id++
i_s_max_id = String(i_i_region_id)
l_l_len = len(i_s_max_id)
i_s_max_id = ''
if l_l_len <> 1 then
        i_s_max_id = ''
        l_l_len = 1 - l_l_len
        for l_l_i = 1 to l_l_len step 1
                i_s_max_id = i_s_max_id + '0'
        next
end if
```

```
// Generate Auto id
l_s_region_id = i_s_max_id + string(i_i_region_id)
l_l_chk = len(l_s_region_id)

// Check orror of id
if l_l_chk > 1 then
        l_l_chk -= 1
        l_s_region_id = Right(l_s_region_id, 1)
        i_s_max_id = Left(i_s_max_id, len(i_s_max_id) - l_l_chk)
end if
dw_data1.setitem(i_l_row, "region_id", l_s_region_id)
dw_data1.setcolumn(2)
```

*Event: w_region. ue_open*

```
Call super::ue_open
This.Post Event ue_calc_id()
```

*Event: w_region. ue_search*

```
string    ls_sql, ls_search_name
long      ll_found

ls_sql = 'select name1 from region'
OpenWithParm(w_search,ls_sql)
ls_search_name = Message.StringParm

if ls_search_name <> "" then
        ll_found = dw_data1.Find("name1 = '"+ls_search_name+"'", 1, dw_data1.RowCount())
        if ll_found > 0 THEN
                dw_data1.ScrollToRow(ll_found)
        else
                MessageBox("search", "not found")
        end if
end if
```

**[Window] w_status from w_seed** Normal
[Description] manipulate data in STATUS data store

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| w_seed | <w_status> | Title="Status"; |
| w_seed`dw_data1 | Dw_data1 | Tab=10; DataObject="d_status_grid"; |

| Declaration | |
|---|---|
| Instance variable | integer i_i_status_id |

| Script | *Type* | *Name* |
|---|---|---|
| w_status | Event | Open<br>ue_calc_id<br>ue_insert<br>ue_open<br>ue_search |

*Event: w_status. Open*

```
Select user's name is using system.
this.title = "Status" + "(user's name)"
```

*Event: w_status. ue_calc_id*

```
select max(status.status_id) Into :i_s_max_id from status;

if isnull(i_s_max_id) then
        i_s_max_id = '0'
end if
i_i_status_id = Integer(i_s_max_id)
```

*Event: w_status. ue_insert*

```
Call super::ue_insert
string   l_s_status_id
long     l_l_len, l_l_i, l_l_chk

// Set '0' of id in the new row
i_i_status_id++
i_s_max_id = String(i_i_status_id)
l_l_len = len(i_s_max_id)
i_s_max_id = ''
if l_l_len <> 2 then
        i_s_max_id = ''
        l_l_len = 2 - l_l_len
        for l_l_i = 1 to l_l_len step 1
                i_s_max_id = i_s_max_id + '0'
        next
end if
// Generate Auto id
l_s_status_id = i_s_max_id + string(i_i_status_id)
l_l_chk = len(l_s_status_id)

// Check orror of id
if l_l_chk > 2 then
        l_l_chk -= 2
        l_s_status_id = Right(l_s_status_id, 2)
        i_s_max_id = Left(i_s_max_id, len(i_s_max_id) - l_l_chk)
end if
dw_data1.setitem(i_l_row, "status_id", l_s_status_id)
dw_data1.setcolumn(2)
```

*Event: w_status. ue_open*

```
Call super::ue_open
This.Post Event ue_calc_id()
```

*Event: w_status. ue_search*

```
string   ls_sql, ls_search_name
long     ll_found

ls_sql = 'select name1 from status'
OpenWithParm(w_search,ls_sql)
ls_search_name = Message.StringParm
```

```
if ls_search_name <> "" then
        ll_found = dw_data1.Find("name1 = '"+ls_search_name+"'", 1, dw_data1.RowCount())
        if ll_found > 0 THEN
                dw_data1.ScrollToRow(ll_found)
        else
                MessageBox("search", "not found")
        end if
end if
```

**[Window] w_user from w_seed** Normal
[Description] manipulate data in USERDATA data store

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| w_seed | <w_user> | Title="User"; |
| w_seed`dw_data1 | Dw_data1 | Tab=10; DataObject="d_user_ff"; |

| Declaration | |
|---|---|
| Instance variable | integer i_i_user_id |

| Script | *Type* | *Name* |
|---|---|---|
| w_user | Event | Open<br>ue_calc_id<br>ue_insert<br>ue_open<br>ue_search |

*Event: w_user. Open*

```
Select user's name is using system.
this.title = "User" + "(user's name)"
```

*Event: w_user. ue_calc_id*

```
select max(userdata.user_id) Into :i_s_max_id from userdata;
if isnull(i_s_max_id) then
        i_s_max_id = '0'
end if
i_i_user_id = Integer(i_s_max_id)
```

*Event: w_user. ue_insert*

```
Call super::ue_insert
string    l_s_user_id
long      l_l_len, l_l_i, l_l_chk

// Set '0' of id in the new row
i_i_user_id++
i_s_max_id = String(i_i_user_id)
l_l_len = len(i_s_max_id)
i_s_max_id = ''
if l_l_len <> 4 then
        i_s_max_id = ''
        l_l_len = 4 - l_l_len
        for l_l_i = 1 to l_l_len step 1
                i_s_max_id = i_s_max_id + '0'
```

```
        next
end if
// Generate Auto id
l_s_user_id = i_s_max_id + string(i_i_user_id)
l_l_chk = len(l_s_user_id)

// Check orror of id
if l_l_chk > 4 then
        l_l_chk -= 4
        l_s_user_id = Right(l_s_user_id, 4)
        i_s_max_id = Left(i_s_max_id, len(i_s_max_id) - l_l_chk)
end if
dw_data1.setitem(i_l_row, "user_id", l_s_user_id)
dw_data1.setcolumn(2)
```

*Event: w_user. ue_open*

```
Call super::ue_open
This.Post Event ue_calc_id()
```

*Event: w_user. ue_search*

```
string    ls_sql, ls_search_name
long      ll_found

ls_sql = 'select user_name from userdata'
OpenWithParm(w_search,ls_sql)
ls_search_name = Message.StringParm
if ls_search_name <> "" then
        ll_found = dw_data1.Find("user_name = '"+ls_search_name+"'", 1, dw_data1.RowCount())
        if ll_found > 0 THEN
                dw_data1.ScrollToRow(ll_found)
        else
                MessageBox("search", "not found")
        end if
end if
```

**[Window] w_vendor from w_seed** Normal
[Description] manipulate data in VENDOR data store

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| w_seed | <w_vendor> | Title="Vendor"; |
| w_seed`dw_data1 | Dw_data1 | Tab=10; DataObject="d_vendor_ff"; |

| Declaration | |
|---|---|
| Instance variable | integer i_i_vendor_id |

| Script | *Type* | *Name* |
|---|---|---|
| w_vendor | Event | Open<br>ue_calc_id<br>ue_insert<br>ue_open<br>ue_search |

*Event: w_vendor. Open*

```
Select user's name is using system.
this.title = "Vendor" + "(user's name)"
```

*Event: w_vendor. ue_calc_id*

```
select max(vendor.vendor_id) Into :i_s_max_id from vendor;
if isnull(i_s_max_id) then

        i_s_max_id = '0'
end if
i_i_vendor_id = Integer(i_s_max_id)
```

*Event: w_vendor. ue_insert*

```
Call super::ue_insert
string    l_s_vendor_id
long      l_l_len, l_l_i, l_l_chk

// Set '0' of id in the new row
i_i_vendor_id++
i_s_max_id = string(i_i_vendor_id)
l_l_len = len(i_s_max_id)
i_s_max_id = ''
if l_l_len <> 4 then
        i_s_max_id = ''
        l_l_len = 4 - l_l_len
        for l_l_i = 1 to l_l_len step 1
                i_s_max_id = i_s_max_id + '0'
        next
end if
// Generate Auto id
l_s_vendor_id = i_s_max_id + string(i_i_vendor_id)
l_l_chk = len(l_s_vendor_id)

// Check orror of id
if l_l_chk > 4 then
        l_l_chk -= 4
        l_s_vendor_id = right(l_s_vendor_id, 4)
        i_s_max_id = left(i_s_max_id, len(i_s_max_id) - l_l_chk)
end if
dw_data1.setitem(i_l_row, "vendor_id", l_s_vendor_id)
dw_data1.setcolumn(2)
```

*Event: w_vendor. ue_open*

```
Call super::ue_open
This.Post Event ue_calc_id()
```

*Event: w_vendor. ue_search*

```
string    ls_sql, ls_search_name
long      ll_found

ls_sql = 'select name from vendor'
OpenWithParm(w_search,ls_sql)
ls_search_name = Message.StringParm
```

```
if ls_search_name <> "" then
        ll_found = dw_data1.Find("name = '"+ls_search_name+"'", 1, dw_data1.RowCount())
        if ll_found > 0 THEN
                dw_data1.ScrollToRow(ll_found)
        else
                MessageBox("search", "not found")
        end if
end if
```

**[Window] w_code_seed from Window**
[Description] ancestor window for data manipulation (descendent window that has 2 data windows)

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| Window | <w_code_seed> | Menu="m_code_base"; |
| Datawindow | dw_data1 | Tab=10; |
| Datawindow | dw_data2 | Tab=30; |
| Statictext | st_record | |

| Declaration | |
|---|---|
| Instance variable | boolean i_b_edit = false<br>long i_l_row, i_l_row_deleted<br>int i_i_row_delete<br>string i_s_max_id, is_emp_name |

| Script | *Type* | *Name* |
|---|---|---|
| w_code_seed | Event | Close<br>Closequery<br>Open<br>ue_delete ( )<br>ue_first ( )<br>ue_insert ( )<br>ue_last ( )<br>ue_next ( )<br>ue_open ( )<br>ue_previous ( )<br>ue_record ( string as_record)<br>ue_undelete ( )<br>ue_undo ( )<br>ue_update ( ) |
| dw_data2 | ControlEvent | Dberror<br>Itemchanged<br>Retrieveend |

*Event: w_code_seed. Close*

```
Close(This)
```

*Event: w_code_seed. Closequery*

```
integer l_i_ret_code

IF dw_data2.DeletedCount() > 0 OR dw_data2.ModifiedCount() > 0 OR i_b_edit THEN
        l_i_ret_code = Messagebox("Ask for save", Question!, YesNoCancel!,1)
```

```
        CHOOSE CASE l_i_ret_code
        CASE 1 // Update
                TriggerEvent('ue_update')
                return
        CASE 3 // Cancel
                return 1
        CASE ELSE // Don't want to update
                return
        END CHOOSE
ELSE
        return
END IF
```

*Event: w_code_seed. Open*

```
This.Post Event ue_open()
```

*Event: w_code_seed. ue_delete ()*

```
integer l_i_ret_code

if dw_data2.RowCount() > 0 then
        l_i_ret_code = MessageBox("Confirm to delete",Question!,YesNo!,2)
        IF l_i_ret_code = 1 THEN
                i_l_row_deleted = dw_data2.getrow()
                i_i_row_delete = dw_data2.deleterow(i_l_row_deleted)

                IF i_i_row_delete <> 1 THEN
                        Messagebox("Error","Can't delete")
                END IF
                dw_data2.setfocus()
                dw_data1.enabled = false
                i_b_edit = true
                m_code_base.m_modi.m_undelete.enabled = true
        ELSE
                return
        END IF
end if
```

*Event: w_code_seed. ue_first ()*

```
dw_data2.ScrollToRow(1)
dw_data2.SetFocus()
```

*Event: w_code_seed. ue_insert ()*

```
dw_data2.SetRedraw(false)
i_l_row = dw_data2.InsertRow(dw_data2.rowcount()+1)

if i_l_row > 0 then
        dw_data2.SetRow(i_l_row)
        dw_data1.enabled = false
        dw_data2.SetRedraw(true)
        i_b_edit = true
else
        MessageBox("Cannot insert row")
end if
```

| *Event: w_code_seed. ue_last ( )* |
|---|
| dw_data2.ScrollToRow(dw_data2.RowCount())<br>dw_data2.SetFocus() |
| *Event: w_code_seed. ue_next ( )* |
| dw_data2.ScrollNextRow()<br>dw_data2.SetFocus() |
| *Event: w_code_seed. ue_open ( )* |
| Disable undelete button |
| *Event: w_code_seed. ue_previous ( )* |
| dw_data2.ScrollPriorRow()<br>dw_data2.SetFocus() |
| *Event: w_code_seed. ue_record ( string as_record )* |
| st_record.text = as_record |
| *Event: w_code_seed. Ue_undelete ( )* |
| Get the last row in the deleted buffer<br>Move the last row in the deleted buffer to the primary buffer<br>set focus to the row that was "restored" |
| *Event: w_code_seed. Ue_undo ( )* |
| Get the original value of this row/column<br>Reset it to the original value<br>Reset the modified flag for this row |
| *Event: w_code_seed. Ue_update ( )* |
| if dw_data2.Update() = 1 then<br>commit;<br>messagebox("save")<br>dw_data1.enabled = true<br>i_b_edit = false<br>else<br>rollback;<br>messagebox("error")<br>end if |
| *ControlEvent: dw_data2. Dberror* |
| set sqldbcode, sqlerrtext, sqlsyntax, row, dw_name to error_data<br>OpenWithParm(w_db_error,error_data)<br>Return 1 |
| *ControlEvent: dw_data2. Itemchanged* |
| i_b_edit = True |
| *ControlEvent: dw_data2. Retrieveeend* |
| Parent.Post Event ue_record("retrieved data "+string(This.RowCount())+" rows ") |

| **[Window] w_province from w_code_seed** Master-detail<br>[Description] Manipulate data in PROVINCE data store |
|---|

| **Controls** | *Control Name* | *Remarks* |
|---|---|---|
| w_code_seed | <w_province> | Title="Province"; |
| w_code_seed`dw_data1 | Dw_data1 | Tab=10; DataObject="dw_region_list"; |
| w_code_seed`dw_data2 | Dw_data2 | Tab=30; DataObject="d_province_input"; |

| **Declaration** | |
|---|---|
| Instance Variable | int ii_prov_id<br>string is_region_id<br>datawindowchild dwc_1 |

| **Script** | *Type* | *Name* |
|---|---|---|
| w_province | Event | Open<br>ue_calc_id<br>ue_insert<br>ue_open<br>ue_retrieve<br>ue_search |
| dw_data1 | ControlEvent | itemchanged<br>Rowfocuschanged |
| dw_data2 | ControlEvent | Rowfocuschanged |

*Event: w_province. Open*

```
Select user's name is using system.
this.title = "Province" + "(user's name)"
```

*Event: w_province. Ue_calc_id*

```
select max(province.prov_id) into :i_s_max_id from province
where province.region_id = :is_region_id;
if isnull(i_s_max_id) then
        i_s_max_id = '0'
end if
ii_prov_id = Integer(i_s_max_id)
```

*Event: w_province. ue_insert*

```
Call super::ue_insert
string    l_s_prov_id
long      l_l_len, l_l_i, l_l_chk

// Set '0' of id in the new row
ii_prov_id++
i_s_max_id = string(ii_prov_id)
l_l_len = len(i_s_max_id)
i_s_max_id = ''
if l_l_len <> 3 then
        i_s_max_id = ''
        l_l_len = 3 - l_l_len
        FOR l_l_i = 1 TO l_l_len
                i_s_max_id = i_s_max_id + '0'
        NEXT
end if
```

```
// Generate Auto id
if i_l_row > 0 then
        dw_data2.setItem(i_l_row,"region_id",is_region_id)
        dw_data2.setrow(i_l_row)
        l_s_prov_id = i_s_max_id + string(ii_prov_id)
        l_l_chk = len(l_s_prov_id)

        // Check orror of id
        if l_l_chk > 3 then
                l_l_chk -= 3
                l_s_prov_id = Right(l_s_prov_id, 3)
                i_s_max_id = Left(i_s_max_id, len(i_s_max_id) - l_l_chk)
        end if
        dw_data2.setitem(i_l_row, "prov_id", l_s_prov_id)
        dw_data2.setfocus()
        dw_data2.ScrollToRow(i_l_row)
        dw_data2.setcolumn("name1")
end if
```

*Event: w_province. ue_open*

```
Call super::ue_open
int i_rc

i_rc = dw_data1.getChild('region_id',dwc_1) // successful = 1
if i_rc <> 1 Then
        MessageBox("can't retrieve")
Else
  dwc_1.SetTransObject(SQLCA)
  dwc_1.Retrieve()
end If
dw_data1.SetTransObject(SQLCA)
dw_data1.Retrieve()
dw_data2.SetTransObject(SQLCA)
```

*Event: w_province. Ue_retrieve*

```
dw_data2.Retrieve(is_region_id)
This. Post Event ue_calc_id()
```

*Event: w_province. ue_search*

```
string    ls_sql, ls_search, ls_search_name
long      ll_found

ls_search = dw_data1.gettext()
CHOOSE CASE ls_search
        CASE "0"
                ls_sql = 'select name1 from province where province.region_id = 0'
        CASE "1"
                ls_sql = 'select name1 from province where province.region_id = 1'
        CASE "2"
                ls_sql = 'select name1 from province where province.region_id = 2'
        CASE "3"
                ls_sql = 'select name1 from province where province.region_id = 3'
        CASE "4"
                ls_sql = 'select name1 from province where province.region_id = 4'
        CASE "5"
```

```
                ls_sql = 'select name1 from province where province.region_id = 5'
        CASE "6"
                ls_sql = 'select name1 from province where province.region_id = 6'
        CASE "9"
                ls_sql = 'select name1 from province where province.region_id = 9'
        CASE ELSE
                return
END CHOOSE

OpenWithParm(w_search,ls_sql)
ls_search_name = Message.StringParm

if ls_search_name <> "" then
        ll_found = dw_data2.Find("name1 = '"+ls_search_name+"'", 1, dw_data2.RowCount())
        if ll_found > 0 THEN
                dw_data2.setfocus()
                dw_data2.ScrollToRow(ll_found)
        else
                MessageBox("search", "not found")
        end if
end if
```

*ControlEvent: dw_data1. Itemchanged*

```
string    ls_column

ls_column = dwo.name
IF ls_column = 'region_id' THEN
        This.Post Event rowFocuschanged(This.getrow())
END IF
```

*ControlEvent: dw_data1. Rowfocuschanged*

```
is_region_id = This.GetItemString(This.GetRow(),'region_id')
Parent.Post event ue_retrieve()
```

*ControlEvent: dw_data2.RowFocusChange*

```
IF i_b_edit then
        This.ScrollToRow(i_l_row)
end if
```

**[Window] w_section from w_code_seed** Master-detail
[Description] Manipulate data in SECTION data store

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| w_code_seed | <w_section> | Title="Section"; |
| w_code_seed`dw_data1 | Dw_data1 | Tab=10; DataObject="dw_division_list "; |
| w_code_seed`dw_data2 | Dw_data2 | Tab=30; DataObject="d_section_input "; |

| Declaration | |
|---|---|
| Instance Variable | int ii_sect_id<br>string is_div_id<br>datawindowchild dwc_1 |

| Script | *Type* | *Name* |
|---|---|---|
| w_section | Event | Open<br>ue_calc_id<br>ue_insert<br>ue_open<br>ue_retrieve<br>ue_search |
| dw_data1 | ControlEvent | itemchanged<br>Rowfocuschanged |
| dw_data2 | ControlEvent | Rowfocuschanged |

*Event: w_section. Open*

```
Select user's name is using system.
this.title = "Section" + "(user's name)"
```

*Event: w_section. Ue_calc_id*

```
select max(section.sect_id) into :i_s_max_id
from section where section.div_id = :is_div_id;

if isnull(i_s_max_id) then
        i_s_max_id = '0'
end if
ii_sect_id = Integer(i_s_max_id)
```

*Event: w_section. ue_insert*

```
Call super::ue_insert
string   l_s_sect_id
long     l_l_len, l_l_i, l_l_chk

// Set '0' of id in the new row
ii_sect_id++
i_s_max_id = string(ii_sect_id)
l_l_len = len(i_s_max_id)
i_s_max_id = ''
if l_l_len <> 3 then
        i_s_max_id = ''
        l_l_len = 3 - l_l_len
        FOR l_l_i = 1 TO l_l_len
                i_s_max_id = i_s_max_id + '0'
        NEXT
end if
// Generate Auto id
if i_l_row > 0 then
        dw_data2.setItem(i_l_row,"div_id",is_div_id)
        dw_data2.setrow(i_l_row)
        l_s_sect_id = i_s_max_id + string(ii_sect_id)
        l_l_chk = len(l_s_sect_id)

        // Check orror of id
        if l_l_chk > 3 then
                l_l_chk -= 3
                l_s_sect_id = Right(l_s_sect_id, 3)
                i_s_max_id = Left(i_s_max_id, len(i_s_max_id) - l_l_chk)
        end if
```

```
        dw_data2.setitem(i_l_row, "sect_id", l_s_sect_id)
        dw_data2.setfocus()
        dw_data2.ScrollToRow(i_l_row)
        dw_data2.setcolumn("name1")
end if
```

*Event: w_section. ue_open*

```
Call super::ue_open
int i_rc

i_rc = dw_data1.getChild('div_id',dwc_1) // successful = 1
if i_rc <> 1 Then
        MessageBox("can't retrieve")
Else
  dwc_1.SetTransObject(SQLCA)
  dwc_1.Retrieve()
end If

dw_data1.SetTransObject(SQLCA)
dw_data1.Retrieve()
dw_data2.SetTransObject(SQLCA)
```

*Event: w_section. Ue_retrieve*

```
dw_data2.Retrieve(is_div_id)
This. Post Event ue_calc_id()
```

*Event: w_section. ue_search*

```
string    ls_sql, ls_search_name
long      ll_found

ls_sql = 'select name1 from division'
OpenWithParm(w_search,ls_sql)

ls_search_name = Message.StringParm

if ls_search_name <> "" then

        Select div_id into :is_div_id
        From division
        Where name1 = :ls_search_name;
        ll_found = dw_data1.Find("div_id = '"+is_div_id+"'", 1, dw_data1.RowCount())

        IF ll_found > 0 THEN
                dw_data1.ScrollToRow(ll_found)
                dw_data1.SetColumn('div_id')
                dw_data1.setItem(ll_found,"div_id",is_div_id)
        ELSE
                MessageBox("search", "not found")
        end if
END IF
```

*ControlEvent: dw_data1. Itemchanged*

```
string    ls_column
```

```
ls_column = dwo.name
IF ls_column = 'div_id' THEN
        This.Post Event rowFocuschanged(This.getrow())
END IF
```

*ControlEvent: dw_data1. Rowfocuschanged*

```
is_div_id = This.GetItemString(This.GetRow(),'div_id')
Parent.Post event ue_retrieve()
```

*ControlEvent: dw_data2.RowFocusChange*

```
IF i_b_edit then
        This.ScrollToRow(i_l_row)
END IF
```

**[Window] w_amphoe from w_code_seed** Master-detail
[Description] Manipulate data in AMPHOE data store

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| w_code_seed | <w_amphoe> | Title="Amphoe"; |
| w_code_seed`dw_data1 | Dw_data1 | Tab=10; DataObject="dw_prov_list"; |
| w_code_seed`dw_data2 | Dw_data2 | Tab=30; DataObject="d_amphoe_input"; |

| Declaration | |
|---|---|
| Instance Variable | string is_region_id, is_prov_id<br>datawindowchild dwc_1,dwc_2<br>int i_i_amphoe_id |

| Script | *Type* | *Name* |
|---|---|---|
| w_amphoe | Event | Open<br>ue_calc_id<br>ue_insert<br>ue_open<br>ue_retrieve<br>ue_search |
| dw_data1 | ControlEvent | Itemchanged<br>Itemfocuschanged<br>Rowfocuschanged |
| dw_data2 | ControlEvent | Rowfocuschanged |

*Event: w_amphoe. Open*

```
Select user's name is using system.
this.title = "Amphoe" + "(user's name)"
```

*Event: w_amphoe. Ue_calc_id*

```
select max(amphoe.amphoe_id) into :i_s_max_id from amphoe
where amphoe.region_id = :is_region_id and amphoe.prov_id = :is_prov_id;

if isnull(i_s_max_id) then
        i_s_max_id = '0'
end if
i_i_amphoe_id = Integer(i_s_max_id)
```

*Event: w_amphoe. ue_insert*

```
Call super::ue_insert
string    l_s_amphoe_id
long      l_l_len, l_l_i, l_l_chk

// Set '0' of id in the new row.
i_i_amphoe_id++
i_s_max_id = string(i_i_amphoe_id)
l_l_len = len(i_s_max_id)
i_s_max_id = ''
if l_l_len <> 2 then
        i_s_max_id = ''
        l_l_len = 2 - l_l_len
        FOR l_l_i = 1 TO l_l_len
                i_s_max_id = i_s_max_id + '0'
        NEXT
end if

// Generate Auto id
if i_l_row > 0 then
        dw_data2.setItem(i_l_row,"region_id",is_region_id)
        dw_data2.setItem(i_l_row,"prov_id",is_prov_id)
        dw_data2.setrow(i_l_row)
        l_s_amphoe_id = i_s_max_id + string(i_i_amphoe_id)
        l_l_chk = len(l_s_amphoe_id)

                // Check orror of id
        if l_l_chk > 2 then
                l_l_chk -= 2
                l_s_amphoe_id = Right(l_s_amphoe_id, 2)
                i_s_max_id = Left(i_s_max_id, len(i_s_max_id) - l_l_chk)
        end if

        dw_data2.setitem(i_l_row, "amphoe_id", l_s_amphoe_id)
        dw_data2.setfocus()
        dw_data2.ScrollToRow(i_l_row)
        dw_data2.setcolumn("name1")
end if
```

*Event: w_amphoe. ue_open*

```
Call super::ue_open
int i_rc

i_rc = dw_data1.getChild('region_id',dwc_1) // successful = 1

if i_rc <> 1 Then
        MessageBox("can't retrieve")
Else
  dwc_1.SetTransObject(SQLCA)
  dwc_1.Retrieve()
end If

i_rc = dw_data1.getChild('prov_id',dwc_2) // successful = 1
if i_rc <> 1 Then
        MessageBox("can't retrieve")
```

```
Else
  dwc_2.SetTransObject(SQLCA)
  dwc_2.Retrieve()
end If

dw_data1.SetTransObject(SQLCA)
dw_data1.Retrieve()
dw_data2.SetTransObject(SQLCA)
```

*Event: w_amphoe. Ue_retrieve*

```
dw_data2.Retrieve(is_region_id,is_prov_id)
This. Post Event ue_calc_id()
```

*Event: w_amphoe. ue_search*

```
string    ls_sql, ls_search_name
long      ll_found

ls_sql = 'select name1 from province'
OpenWithParm(w_search,ls_sql)
ls_search_name = Message.StringParm

if ls_search_name <> "" then

        Select region_id,prov_id into :is_region_id, :is_prov_id
        From province
        Where name1 = :ls_search_name;
        ll_found = dw_data1.Find("region_id = '"+is_region_id+"' and prov_id = '"+is_prov_id+"'",
1, dw_data1.RowCount())

        IF ll_found > 0 THEN
                dw_data1.ScrollToRow(ll_found)
                dw_data1.SetColumn('region_id')
                dw_data1.setItem(ll_found,"region_id",is_region_id)
                dw_data1.SetColumn('prov_id')
                dw_data1.setItem(ll_found,"prov_id",is_prov_id)
        ELSE
                MessageBox("search", "not found")
        end if
END IF
```

*ControlEvent: dw_data1. Itemchanged*

```
string    ls_column

ls_column = dwo.name
IF ls_column = 'prov_id' THEN
        This.Post Event rowFocuschanged(This.getrow())
END IF
```

*ControlEvent: dw_data1. Itemfocuschanged*

```
string    ls_column, ls_expression
int       i_rc

ls_column = dwo.name
IF ls_column = 'prov_id' THEN
```

```
        is_region_id = This.GetItemString(This.GetRow(),'region_id')
        IF isNull (is_region_id) then
                MessageBox("error")
                i_rc = SetColumn('region_id')
                return
        end if
                ls_expression = "region_id = '"+is_region_id+"'"
                i_rc = dwc_2.SetFilter(ls_expression)
                if i_rc <> 1 then
                        MessageBox("error")
                else
                        dwc_2.filter()
                end if
END IF
```

*ControlEvent: dw_data1. Rowfocuschanged*

```
is_prov_id = This.GetItemString(This.GetRow(),'prov_id')
Parent.Post event ue_retrieve()
```

*ControlEvent: dw_data2.RowFocusChange*

```
IF i_b_edit then
        This.ScrollToRow(i_l_row)
END IF
```

**[Window] w_employee from w_code_seed** Master-detail
[Description] Manipulate data in EMPLOYEE data store

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| w_code_seed | <w_employee> | Title="Employee"; |
| w_code_seed`dw_data1 | Dw_data1 | Tab=10; DataObject="dw_sect_list"; |
| w_code_seed`dw_data2 | Dw_data2 | Tab=30; DataObject="d_employee_input"; |

| Declaration | |
|---|---|
| Instance Variable | string is_div_id, is_sect_id, is_emp_id<br>datawindowchild dwc_1,dwc_2<br>integer i_i_emp_id |

| Script | *Type* | *Name* |
|---|---|---|
| w_employee | Event | Open<br>ue_calc_id<br>ue_insert<br>ue_open<br>ue_retrieve<br>ue_search |
| dw_data1 | ControlEvent | Itemchanged<br>Itemfocuschanged<br>Rowfocuschanged |
| dw_data2 | ControlEvent | Rowfocuschanged |

*Event: w_employee. Open*

```
Select user's name is using system.
this.title = "Employee" + "(user's name)"
```

*Event: w_employee. Ue_calc_id*

```
select max(employee.emp_id) Into :i_s_max_id from employee;
if isnull(i_s_max_id) then
        i_s_max_id = '0'
end if
i_i_emp_id = Integer(i_s_max_id)
```

*Event: w_employee. ue_insert*

```
Call super::ue_insert
string   l_s_emp_id
long     l_l_len, l_l_i, l_l_chk

// Set '0' of id in the new row
i_i_emp_id++
i_s_max_id = string(i_i_emp_id)
l_l_len = len(i_s_max_id)
i_s_max_id = ''
if l_l_len <> 4 then
        i_s_max_id = ''
        l_l_len = 4 - l_l_len
        FOR l_l_i = 1 TO l_l_len
                i_s_max_id = i_s_max_id + '0'
        NEXT
end if

// Generate Auto id
if i_l_row > 0 then
        dw_data2.setItem(i_l_row,"div_id",is_div_id)
        dw_data2.setItem(i_l_row,"sect_id",is_sect_id)
        dw_data2.setrow(i_l_row)
        l_s_emp_id = i_s_max_id + string(i_i_emp_id)
        l_l_chk = len(l_s_emp_id)

        // Check orror of id
        if l_l_chk > 4 then
                l_l_chk -= 4
                l_s_emp_id = Right(l_s_emp_id, 4)
                i_s_max_id = Left(i_s_max_id, len(i_s_max_id) - l_l_chk)
        end if

        dw_data2.setitem(i_l_row, "emp_id", l_s_emp_id)
        dw_data2.setfocus()
        dw_data2.ScrollToRow(i_l_row)
        dw_data2.setcolumn("pertit_id")
end if
```

*Event: w_employee. ue_open*

```
Call super::ue_open
int i_rc

i_rc = dw_data1.getChild('div_id',dwc_1) // successful = 1
```

```
if i_rc <> 1 Then
        MessageBox("can't retrieve")
Else
  dwc_1.SetTransObject(SQLCA)
  dwc_1.Retrieve()
end If

i_rc = dw_data1.getChild('sect_id',dwc_2) // successful = 1
if i_rc <> 1 Then
        MessageBox("can't retrieve")
Else
  dwc_2.SetTransObject(SQLCA)
  dwc_2.Retrieve()
end If

dw_data1.SetTransObject(SQLCA)
dw_data1.Retrieve()
dw_data2.SetTransObject(sqlca)
```

*Event: w_employee. Ue_retrieve*

```
dw_data2.Retrieve(is_div_id,is_sect_id)
This. Post Event ue_calc_id()
```

*Event: w_employee. ue_search*

```
string    ls_sql, ls_search_name
long      ll_found

ls_sql = 'select name from employee'
OpenWithParm(w_search,ls_sql)
ls_search_name = Message.StringParm

IF ls_search_name <> "" then

        Select div_id,sect_id,emp_id into :is_div_id, :is_sect_id, :is_emp_id
        From employee
        Where name = :ls_search_name;

        ll_found = dw_data2.Find("div_id = '"+is_div_id+"' and sect_id = '"+is_sect_id+"' and
emp_id = '" +is_emp_id+"'",  1, dw_data2.RowCount())
        IF ll_found > 0 THEN
                dw_data2.ScrollToRow(ll_found)
                dw_data2.setfocus()
        ELSE
                MessageBox("search", "not found")
        END IF
END IF
```

*ControlEvent: dw_data1. Itemchanged*

```
string ls_column

ls_column = dwo.name
IF ls_column = 'sect_id' THEN
        This.Post Event rowFocuschanged(This.getrow())
END IF
```

*ControlEvent: dw_data1. Itemfocuschanged*

```
string ls_column, ls_expression
int i_rc

ls_column = dwo.name
IF ls_column = 'sect_id' THEN
        is_div_id = This.GetItemString(This.GetRow(),'div_id')
        IF isNull(is_div_id) then
                MessageBox("error")
                i_rc = SetColumn('div_id')
                return
        end if
                ls_expression = "div_id = '"+is_div_id+"'"
                i_rc = dwc_2.SetFilter(ls_expression)
                if i_rc <> 1 then
                        MessageBox("error")
                else
                        dwc_2.filter()
                end if
END IF
```

*ControlEvent: dw_data1. Rowfocuschanged*

```
is_sect_id = This.GetItemString(This.GetRow(),'sect_id')
Parent.Post event ue_retrieve()
```

*ControlEvent: dw_data2.RowFocusChange*

```
IF i_b_edit then
        This.ScrollToRow(i_l_row)
END IF
```

**[Window] w_search from Window** Shared window

[Description] receive SQL statement to create datawindow and return selected name

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| Window | <w_search> | Type=response!; Title="Search data"; |
| u_selection_list | uo_1 | |

| Declaration | |
|---|---|
| Instance Variable | string is_name |

| Script | *Type* | *Name* |
|---|---|---|
| w_search | Event | Close<br>Open |
| uo_1 | ControlEvent | ue_entry_chosen |

*Event: w_search.close*

```
CloseWithReturn(This, is_name)
```

Event: w_search.open

```
string ls_sql
ls_sql = Message.StringParm
```

| uo_1.create_datawindow(sqlca, ls_sql) |
|---|
| *ControlEvent: uo_1.ue_entry_chosen* |
| is_name = return_selected ()<br>Close(Parent) |

## 2. Transaction process

These windows manipulate data in "TRANS DATA" data store. It divides into 2 major processes: adding and editing. Adding process has 2 ancestor windows, 8 descendent windows and a shared window. Editing process has 2 ancestor windows, 5 descendent windows and a shared window.

### Adding process

| **[Window] w_trans_base1 from Window**<br>[Description] window ancestor for transaction adding process |
|---|

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| Window | <w_trans_base1> | Menu=" m_code_trans_add"; Title="Transaction"; |
| Datawindow | dw_1 | Tab=10; |
| Datawindow | dw_2 | Tab=20; |

| Declaration | |
|---|---|
| Shared Variable | string ss_list |
| Instance Variable | boolean i_b_edit = false<br>long il_row_1, il_row_2, il_row<br>integer ii_main_id<br>datawindowchild dwc_1, dwc_2, dwc_3<br>string is_autho_id, is_emp_name, is_plant_id, is_plant_id1, is_id<br>string is_stock_id, is_stock_id1, is_check, is_dataobject |

| Script | *Type* | *Name* |
|---|---|---|
| w_trans_base1 | Event | Close<br>Closequery<br>Open<br>ue_insert_dw_1 ( )<br>ue_insert_dw_2 ( )<br>ue_new_dw_1( )<br>ue_open ( )<br>ue_print ( )<br>ue_update ( )<br>ue_update_dw_1 ( ) |
| dw_1 | ControlEvent | Dberror<br>Itemchanged |
| dw_2 | ControlEvent | Dberror<br>Itemchanged |

*Event: w_trans_base1. Close*

```
close(this)
```

*Event: w_trans_base1.* Closequery

```
int i_rc

IF (dw_1.ModifiedCount() > 0 OR dw_2.ModifiedCount() > 0 ) and (i_b_edit) Then
        i_rc = Messagebox ("Ask for save", Question!, YesNoCancel!,1)
        CHOOSE CASE i_rc
        CASE 1 // Update
                This.Trigger Event ue_update()
                return
        CASE 3 // Cancel
                return 1
        CASE ELSE // Don't want to update
                return
        END CHOOSE
ELSE
        return
END IF
```

*Event: w_trans_base1.* Open

```
This.Post Event ue_open()
```

*Event: w_trans_base1.* ue_insert_dw_1 ( )

```
il_row_1 = dw_1.insertrow(0)

if il_row_1 <= 0 then
        MessageBox("Cannot insert row")
        return
elseif il_row_1 > 0 then
        dw_1.ScrollToRow(il_row_1)
        i_b_edit = TRUE
        m_code_trans_add.m_modi.m_insert_dw_1.enabled = false
        dw_1.object.div_id[il_row_1] = gs_div_id
        dw_1.object.sect_id[il_row_1] = gs_sect_id
end if
```

*Event: w_trans_base1.* ue_insert_dw_2 ( )

```
dw_2.visible = true
il_row_2 = dw_2.insertrow(0)

if il_row_2 <= 0 then
        MessageBox("Cannot insert row")
        return
elseif il_row_2 > 0 then
        dw_2.scrollToRow(il_row_2)
        i_b_edit = TRUE
        m_code_trans_add.m_modi.m_insert_dw_1.enabled = false
        m_code_trans_add.m_modi.m_insert_dw_2.enabled = false
        m_code_trans_add.m_modi.m_update.enabled = true
        dw_1.enabled = false
end if
```

*Event: w_trans_base1.* ue_new_dw_1 ( )

```
m_code_trans_add.m_file.m_print.enabled = false
dw_1.reset()
dw_1.enabled = true
trigger event ue_calc_id()
this.trigger event ue_insert_dw_1()
dw_1.setrow(il_row_1)
dw_2.reset()
```

*Event: w_trans_base1* ue_open ( )

```
dw_2.visible = false
dw_1.SetTransObject(SQLCA)
dw_1.retrieve()
dw_2.SetTransObject(SQLCA)

m_code_trans_add.m_file.m_print.enabled = false
m_code_trans_add.m_modi.m_update.enabled = false
```

*Event: w_trans_base1.* ue_print ( )

```
str_trans_print lstr_data

w_trans_print l_w_instance
lstr_data.s_dataobject = is_dataobject
lstr_data.s_no = string(ii_main_id)
OpenSheetWithParm (l_w_instance,lstr_data,g_w_mdi,0,Layered!)
```

*Event: w_trans_base1. ue_update ( )*

```
if dw_1.update(true,false) > 0 then
        commit using sqlca;
        if dw_2.update(true,false) > 0 then
                This.Trigger event ue_update_dw_2()
                commit using sqlca;
                messagebox("save")
                m_code_trans_add.m_modi.m_update.enabled = false
                m_code_trans_add.m_modi.m_insert_dw_1.enabled = true
                m_code_trans_add.m_modi.m_insert_dw_2.enabled = true
                m_code_trans_add.m_file.m_print.enabled = true
                dw_1.enabled = false
                i_b_edit = false
                dw_1.resetupdate()
                dw_2.resetupdate()
        else
                rollback using sqlca;
                messagebox("error")
                return
        end if
else
        rollback using sqlca;
        messagebox("error")
        return
end if
```

*Event: w_trans_base1. ue_update_dw_1 ( ).*

```
IF dw_1.Update() = 1 Then
```

```
        commit using sqlca;
        messagebox("save")
        m_code_trans_add.m_modi.m_update_dw_1.enabled = false
else
        rollback using sqlca;
        messagebox("error")
        return
end if
```

*ControlEvent: dw_1. Dberror*

See in w_code_seed.dw_data2.dberror

*ControlEvent: dw_1. Itemchanged*

m_code_trans_add.m_modi.m_update_dw_1.enabled = true
i_b_edit = True

*ControlEvent: dw_2. Dberror*

See in w_code_seed.dw_data2.dberror

*ControlEvent: dw_2. Itemchanged*

m_code_trans_add.m_modi.m_update.enabled = true
i_b_edit = True

**[Window] w_trans_distribute from w_trans_base1**
[Description] Distribution

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| w_trans_base1 | <w_trans_distribute> | Title="Distribute"; |
| w_trans_base1`dw_1 | dw_1 | DataObject="d_dist_master "; |
| w_trans_base1`dw_2 | dw_2 | DataObject="d_dist_detail "; |

| Declaration | |
|---|---|
| Instance Variable | string is_dist_id<br>real ir_stock_quan2, ir_stock_quan3, ir_stock_quan4, ir_stock_quan5 |

| Script | *Type* | *Name* |
|---|---|---|
| w_trans_distribute | Event | Open<br>ue_calc_id<br>ue_insert_dw_1<br>ue_insert_dw_2<br>ue_open<br>ue_update_dw_2<br>ue_update_fn |
| dw_1 | ControlEvent | Itemchanged |
| dw_2 | ControlEvent | Itemchanged |

*Event: w_trans_distribute.Open*

Select user's name is using system.
this.title = "Distribute" + "(user's name)"

*Event: w_trans_distribute. ue_calc_id*

```
string ls_main_id
select max(convert(numeric, distribution.dist_id)) into :ls_main_id from distribution
where distribution.div_id = :gs_div_id and distribution.sect_id = :gs_sect_id;

if isnull(ls_main_id) then
        ii_main_id = 1
else
        ii_main_id = integer(ls_main_id)+1
end if
```

*Event: w_trans_distribute. ue_insert_dw_1*

```
Call super::ue_insert_dw_1
dw_1.object.dist_id[il_row_1] = string(ii_main_id)
dw_1.object.date1[il_row_1] = gd_today
```

*Event: w_trans_distribute. ue_insert_dw_2*

```
Call super:: ue_insert_dw_2
dw_2.object.div_id[il_row_2] = gs_div_id
dw_2.object.sect_id[il_row_2] = gs_sect_id
dw_2.object.dist_id[il_row_2] = string(ii_main_id)
dw_2.object.chk[il_row_2] = 'y'
dw_2.setcolumn("plant_id")
```

*Event: w_trans_distribute. ue_open*

```
Call super::ue_open
int i_rc

i_rc = dw_1.getChild('stock_id',dwc_1) // successful = 1
if i_rc <> 1 Then
        MessageBox("error")
Else
  dwc_1.SetTransObject(SQLCA)
  dwc_1.Retrieve()
end If
i_rc = dw_1.getChild('emp_id1',dwc_2) // successful = 1
if i_rc <> 1 Then
        MessageBox("error")
Else
  dwc_2.SetTransObject(SQLCA)
  dwc_2.Retrieve()
end If
i_rc = dw_2.getChild('plant_id',dwc_3) // successful = 1
if i_rc <> 1 Then
        MessageBox("error")
Else
  dwc_3.SetTransObject(SQLCA)
  dwc_3.Retrieve()
end If

is_dataobject = 'd_report_dist'
m_code_trans_add.m_modi.m_search.enabled = false
this.post event ue_calc_id()
this.post event ue_insert_dw_1()
```

*Event: w_trans_distribute. ue_update_dw_2*

```
long ll_row

ll_row = dw_2.RowCount()

if ll_row > 0 then
        For il_row_2 = 1 to ll_row
                is_check = dw_2.object.chk[il_row_2]
                if is_check = 'y' then
                        This.Trigger event ue_update_fn()
                end if
        Next
end if

update distribution_detail set chk = 'y'
where dist_id = :is_dist_id and div_id = :gs_div_id and sect_id = :gs_sect_id;
```

*Event: w_trans_distribute. ue_update_fn*

```
is_dist_id        = dw_1.GetItemString(dw_1.GetRow(),'dist_id')
is_stock_id = dw_1.GetItemString(dw_1.GetRow(),'stock_id')
is_plant_id = dw_2.object.plant_id[il_row_2]
gr_dist_quan = dw_2.object.quan[il_row_2]

IF isNull (is_stock_id) or isNull (is_plant_id) or isNull (gs_div_id) or isNull (gs_sect_id) then
        MessageBox("error")
        return
END IF

if isnull(gr_dist_quan) then
        gr_dist_quan = 0
end if

select quan2 into :ir_stock_quan2 from stock_detail
where div_id = :gs_div_id and sect_id = :gs_sect_id and
stock_id = :is_stock_id and plant_id = :is_plant_id;

select quan3 into :ir_stock_quan3 from stock_detail
where div_id = :gs_div_id and sect_id = :gs_sect_id and
stock_id = :is_stock_id and plant_id = :is_plant_id;

select quan4 into :ir_stock_quan4 from stock_detail
where div_id = :gs_div_id and sect_id = :gs_sect_id and
stock_id = :is_stock_id and plant_id = :is_plant_id;

select quan5 into :ir_stock_quan5 from stock_detail
where div_id = :gs_div_id and sect_id = :gs_sect_id and
stock_id = :is_stock_id and plant_id = :is_plant_id;

if isnull(ir_stock_quan2) then
        ir_stock_quan2 = 0
end if

if isnull(ir_stock_quan3) then
        ir_stock_quan3 = 0
end if
```

```
if isnull(ir_stock_quan4) then
        ir_stock_quan4 = 0
end if

if isnull(ir_stock_quan5) then
        ir_stock_quan5 = 0
end if

gr_stock_quan5 = ir_stock_quan5 - ir_stock_quan2 - ir_stock_quan3 - ir_stock_quan4

if gr_dist_quan > gr_stock_quan5 then
        MessageBox("error", "This plant can't distribute more than gr_stock_quan5")
        m_code_trans_add.m_modi.m_update.enabled = false
        m_code_trans_add.m_modi.m_insert_dw_2.enabled = false
else
        gr_stock_quan5 = gr_stock_quan5 - gr_dist_quan

        UPDATE stock_detail SET quan5 = :gr_stock_quan5 + :ir_stock_quan2 + :ir_stock_quan3 +
:ir_stock_quan4
        WHERE div_id = :gs_div_id and sect_id = :gs_sect_id and stock_id = :is_stock_id and
plant_id = :is_plant_id;

        dw_2.object.chk[il_row_2] = 'n'

        m_code_trans_add.m_modi.m_update.enabled = true
        m_code_trans_add.m_modi.m_insert_dw_2.enabled = true
end if
```

*ControlEvent: dw_1. Itemfocuschanged*

```
string  ls_column, ls_expression, ls_expression1
int     i_rc

ls_column = dwo.name

IF ls_column = 'stock_id' THEN
        IF isNull (gs_div_id) then
                MessageBox("error")
                return
        end if
        IF isNull (gs_sect_id) then
                MessageBox("error")
                return
        end if
        ls_expression = "div_id = '"+gs_div_id+"' and sect_id = '"+gs_sect_id+"'"
        i_rc = dwc_1.SetFilter(ls_expression)
        if i_rc <> 1 then
                MessageBox("error")
        Else
                dwc_1.filter()
        end if
elseif ls_column = 'emp_id1' then
        select autho_id into :is_autho_id from employee
        where autho_id = 01;

        ls_expression1 = "autho_id = '"+is_autho_id+"'"
```

```
        i_rc = dwc_2.SetFilter(ls_expression1)
        if i_rc <> 1 then
                MessageBox("error")
        else
                dwc_2.filter()
        end if
END IF
```

*ControlEvent: dw_2. Itemfocuschanged*

```
string  ls_column, ls_expression
int     i_rc

ls_column = dwo.name
IF ls_column = 'quan'  or ls_column = 'plant_id' THEN
        is_stock_id = dw_1.GetItemString(dw_1.GetRow(),'stock_id')
        IF isNull (is_stock_id) then
                MessageBox("error")
                return
        end if
        IF isNull (gs_div_id) then
                MessageBox("error")
                return
        end if
        IF isNull (gs_sect_id) then
                MessageBox("error")
                return
        end if

        ls_expression = "stock_detail_stock_id = '"+is_stock_id+"' and stock_detail_div_id =
'"+gs_div_id+"' and stock_detail_sect_id = '"+gs_sect_id+"'"
        i_rc = dwc_3.SetFilter(ls_expression)
        if i_rc <> 1 then
                MessageBox("error")
        else
                dwc_3.filter()
        end if
end if
```

Other windows that inherit from w_trans_base1 (w_trans_transfer, w_trans_purchase and w_trans_produce) are similar to "w_trans_distribute", hence their program specification are skipped.

**[Window] w_trans_print from w_print_base**
[Description] Print transaction report

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| w_print_base | <w_trans_print> | Menu="m_trans_print"; |

| Script | *Type* | *Name* |
|---|---|---|
| w_trans_print | Event | ue_open |

| *Event w_trans_print.ue_open()* |
|---|
| str_trans_print str_x<br>string ls_rc<br><br>str_x = Message.PowerObjectParm<br>dw_1.dataobject = str_x.s_dataobject<br>dw_1.SetTransObject(SQLCA)<br>dw_1.Retrieve(gs_div_id,gs_sect_id,gs_year,str_x.s_no)<br>This.Post event Ue_set_page()<br>dw_1.modify("DataWindow.Print.Preview = yes") |

**[Window] w_trans_ctrl_base from Window**
[Description] window ancestor for transaction adding process (methods of receiving)

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| Window | <w_trans_ctrl_base> | Menu=" m_code_trans_ctrl"; |
| Datawindow | dw_1 | Tab=10; |
| Datawindow | dw_2 | Tab=20; |

| Declaration | |
|---|---|
| Shared Variable | string ss_list |
| Instance Variable | boolean i_b_edit = false<br>long il_row_1, il_row_2, il_row<br>integer ii_main_id<br>datawindowchild dwc_1, dwc_2, dwc_3<br>string is_autho_id, is_emp_name, is_plant_id, is_plant_id1<br>string is_stock_id, is_stock_id1, is_no, is_check |

| Script | *Type* | *Name* |
|---|---|---|
| w_trans_ctrl_base | Event | Close<br>Closequery<br>Open<br>ue_cancel<br>ue_open ( )<br>ue_retrieve ( )<br>ue_search ( )<br>ue_undo ( )<br>ue_update ( )<br>ue_update_dw_1 ( ) |
| | PrivateFunc | wf_undo (datawindow adw_x) return integer |
| dw_1 | ControlEvent | Dberror<br>Itemchanged |
| dw_2 | ControlEvent | Dberror<br>Itemchanged |

| *Event: w_trans_ctrl_base. Close* |
|---|
| close(this) |
| *Event: w_trans_ctrl_base.* Closequery |
| int i_rc |

```
IF (dw_1.ModifiedCount() > 0 OR dw_2.ModifiedCount() > 0 ) and (i_b_edit) Then
        i_rc = Messagebox ("Ask for save", Question!, YesNoCancel!,1)
        CHOOSE CASE i_rc
        CASE 1 // Update
                This.Trigger Event ue_update()
                return
        CASE 3 // Cancel
                return 1
        CASE ELSE // Don't want to update
                return
        END CHOOSE
ELSE
        return
END IF
```

*Event: w_trans_ctrl_base.* Open

```
This.Post Event ue_open()
```

*Event: w_trans_ctrl_base.* ue_cancel ( )

```
int i_rc

i_rc = MessageBox("Ask for cancel",Question!,YesNo!,2)
if i_rc = 1 then
        This.Post Event ue_open()
end if
```

*Event: w_trans_ctrl_base* ue_open ( )

```
dw_1.visible = false
dw_2.visible = false
This.Trigger Event ue_search()
```

*Event: w_trans_ctrl_base.* ue_retrieve ( )

```
long l_rc

if is_no <> 'n' then
        dw_1.SetTransObject(SQLCA)
        l_rc = dw_1.Retrieve(is_no,gs_div_id,gs_sect_id)
        if l_rc > 0 then
                dw_1.visible = true
                dw_1.enabled = true
                il_row_1 = dw_1.GetRow()
                m_code_trans_ctrl.m_modi.m_update.enabled = true
                m_code_trans_ctrl.m_modi.m_update_dw_1.enabled = true
                m_code_trans_ctrl.m_modi.m_undo.enabled = true
                m_code_trans_ctrl.m_modi.m_cancel.enabled = true
                This.Trigger event ue_retrieve_dw_2()
        Else
                dw_1.visible = false
                dw_2.visible = false
                MessageBox('search','not found')
                m_code_trans_ctrl.m_modi.m_update.enabled = false
                m_code_trans_ctrl.m_modi.m_update_dw_1.enabled = false
                m_code_trans_ctrl.m_modi.m_undo.enabled = false
                m_code_trans_ctrl.m_modi.m_cancel.enabled = false
```

```
                m_code_trans_ctrl.m_modi.m_search.enabled = true
        end if
else
        m_code_trans_ctrl.m_modi.m_update.enabled = false
        m_code_trans_ctrl.m_modi.m_update_dw_1.enabled = false
        m_code_trans_ctrl.m_modi.m_undo.enabled = false
        m_code_trans_ctrl.m_modi.m_cancel.enabled = false
        m_code_trans_ctrl.m_modi.m_search.enabled = true
end if
```

*Event: w_trans_ctrl_base* ue_search ( )

```
open(w_trans_q_no,This)
is_no = Message.StringParm
This.Trigger Event ue_retrieve()
```

*Event: w_trans_ctrl_base* ue_undo ( )

```
if wf_undo(dw_1) = 0 then
        wf_undo(dw_2)
end if
```

*Event: w_trans_ctrl_base. ue_update ( ).*

```
IF dw_1.Update() = 1 Then
        This.Trigger Event ue_update_dw_2()
        messagebox("save")
        m_code_trans_ctrl.m_modi.m_update.enabled = false
        m_code_trans_ctrl.m_modi.m_search.enabled = true
        i_b_edit = false
Else
        dw_1.setFocus()
end if
```

*Event: w_trans_ctrl_base. ue_update_dw_1 ( ).*

```
IF dw_1.Update() = 1 Then
        commit using sqlca;
        messagebox("save")
        m_code_trans_ctrl.m_modi.m_update_dw_1.enabled = false
        m_code_trans_ctrl.m_modi.m_search.enabled = true
else
        rollback using sqlca;
        messagebox("error")
        return
end if
```

*PrivateFunction: w_trans_edit_base.* wf_undo (datawindow adw_x) return integer

```
Get the original value of  adw_x row/column
Reset it to the original value
Reset the modified flag for adw_x row
```

*ControlEvent: dw_1. Dberror*

```
See in w_code_seed.dw_data2.dberror
```

*ControlEvent: dw_1. Itemchanged*

```
m_code_trans_ctrl.m_modi.m_update_dw_1.enabled = true
m_code_trans_ctrl.m_modi.m_search.enabled = false
```

| |
|---|
| i_b_edit = True |
| *ControlEvent: dw_2. Dberror* |
| See in w_code_seed.dw_data2.dberror |
| *ControlEvent: dw_2. Itemchanged* |
| m_code_trans_ctrl.m_modi.m_search.enabled = false<br>i_b_edit = True |

| |
|---|
| **[Window] w_trans_ctrl_purchase from w_trans_ctrl_base**<br>[Description] Receive by purchasing method. |

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| w_trans_base1 | <w_trans_ctrl_purchase> | Title="Receive by purchasing method"; |
| w_trans_base1`dw_1 | dw_1 | DataObject="d_pur_ctrl "; |
| w_trans_base1`dw_2 | dw_2 | DataObject="d_pur_ctrl_detail "; |

| Declaration | |
|---|---|
| Instance Variable | string is_purchase_id<br>real ir_price_unit, ir_stock_quan5 |

| Script | *Type* | *Name* |
|---|---|---|
| w_trans_ctrl_purchase | Event | Open<br>ue_open<br>ue_retrieve_dw_2<br>ue_update_dw_2<br>ue_update_fn |
| dw_1 | ControlEvent | Itemfocuschanged |

| *Event: w_trans_ctrl_purchase.Open* |
|---|
| Select user's name is using system.<br>this.title = " Receive by purchasing method " + "(user's name)" |
| *Event: w_trans_ctrl_purchase. ue_open* |
| Call super::ue_open<br>int i_rc<br><br>i_rc = dw_1.getChild('emp_id2',dwc_1) // successful = 1<br>if i_rc <> 1 Then<br>MessageBox("error")<br>Else<br>dwc_1.SetTransObject(SQLCA)<br>dwc_1.Retrieve()<br>end If<br>dw_1.setcolumn('emp_id2') |
| *Event: w_trans_ctrl_purchase. ue_retrieve_dw_2* |
| if is_no <> 'n' then<br>dw_1.object.date2[il_row_1] = gd_today<br>end if |

```
dw_2.SetTransObject(SQLCA)
dw_2.Retrieve(is_no,gs_div_id,gs_sect_id)
dw_2.visible = true

long    ll_row
int     li_chk

ll_row = dw_2.RowCount()
li_chk = 0
For il_row_2 = 1 to ll_row
        is_check = dw_2.object.chk[il_row_2]
        if is_check = 'n' then
                m_code_trans_ctrl.m_modi.m_update.enabled = false
                li_chk = li_chk + 1
        end if
Next

if li_chk > 0 then
        messagebox("checking")
end if
```

*Event: w_trans_ctrl_purchase. ue_update_dw_2*

```
long ll_row

ll_row = dw_2.RowCount()

if ll_row > 0 then
        For il_row_2 = 1 to ll_row
                is_check = dw_2.object.chk[il_row_2]
                if is_check = 'y' then
                        This.Trigger event ue_update_fn()
                end if
        Next
end if

update purchasing_detail set chk = 'n'
where purchase_id = :is_purchase_id and div_id = :gs_div_id and sect_id = :gs_sect_id;
```

*Event: w_trans_ctrl_purchase. ue_update_fn*

```
integer i_rc

is_purchase_id = dw_1.GetItemString(dw_1.GetRow(),'purchase_id')
is_stock_id = dw_1.GetItemString(dw_1.GetRow(),'stock_id')
id_date = dw_1.GetItemDate(dw_1.GetRow(),'date2')
is_plant_id = dw_2.object.plant_id[il_row_2]
ir_price_unit = dw_2.object.price_unit[il_row_2]
gr_pur_quan = dw_2.object.quan[il_row_2]

IF isNull (is_stock_id) or isNull (is_plant_id) or isNull (gs_div_id) or isNull (gs_sect_id) then
        MessageBox("error")
        return
end if
if isnull(gr_pur_quan) then
        gr_pur_quan = 0
end if
```

```
if isnull(ir_price_unit) then
        ir_price_unit = 0
end if

//checking in the stock has this plant
select plant_id into :is_plant_id1 from stock_detail
where div_id = :gs_div_id and sect_id = :gs_sect_id and stock_id = :is_stock_id and plant_id =
:is_plant_id;

if is_plant_id1 = "" then
        INSERT INTO stock_detail (stock_id,div_id,sect_id,plant_id,quan1,quan5)
        VALUES (:is_stock_id,:gs_div_id,:gs_sect_id,:is_plant_id,:gr_pur_quan,:gr_pur_quan);
else
        //checking quantity of plant in stock
        select quan1 into :gr_stock_quan from stock_detail
        where div_id = :gs_div_id and sect_id = :gs_sect_id and stock_id = :is_stock_id and
plant_id = :is_plant_id;

        select quan5 into :ir_stock_quan5 from stock_detail
        where div_id = :gs_div_id and sect_id = :gs_sect_id and stock_id = :is_stock_id and
plant_id = :is_plant_id;

        if isnull(gr_stock_quan) then
                gr_stock_quan = 0
        end if

        if isnull(ir_stock_quan5) then
                ir_stock_quan5 = 0
        end if

        UPDATE stock_detail SET quan1 = :gr_stock_quan + :gr_pur_quan
        WHERE div_id = :gs_div_id and sect_id = :gs_sect_id and stock_id = :is_stock_id and
plant_id = :is_plant_id;

        UPDATE stock_detail SET quan5 = :ir_stock_quan5 + :gr_pro_quan
        WHERE div_id = :gs_div_id and sect_id = :gs_sect_id and stock_id = :is_stock_id and
plant_id = :is_plant_id;
end if

UPDATE purchasing_detail SET price_unit = :ir_price_unit
WHERE purchase_id = :is_purchase_id and div_id = :gs_div_id and sect_id = :gs_sect_id and
plant_id = :is_plant_id;

gr_net_quan = gr_stock_quan + gr_pur_quan

select count(*) into :i_rc from net_quantity;

insert into "net_quantity"("net_id","stock_id","div_id","sect_id","plant_id","old_quan","trans_quan",
"net_quan","date2")
values(:i_rc,:is_stock_id,:gs_div_id,:gs_sect_id,:is_plant_id,:gr_stock_quan,:gr_pur_quan,
:gr_net_quan,:id_date);

is_plant_id = ""
is_plant_id1 = ""
```

```
dw_2.object.chk[il_row_2] = 'n'

m_code_trans_ctrl.m_modi.m_update.enabled = false
```

*ControlEvent: dw_1. Itemfocuschanged*

```
string   ls_column, ls_expression, ls_expression1
int      i_rc

ls_column = dwo.name
if ls_column = 'emp_id2' then

        select autho_id into :is_autho_id from employee
        where autho_id = 06;
        ls_expression = "autho_id = '"+is_autho_id+"'"
        i_rc = dwc_1.SetFilter(ls_expression)
        if i_rc <> 1 then
                MessageBox("error")
        else
                dwc_1.filter()
        end if
END IF
```

Other windows that inherit from w_trans_ctrl_base (w_trans_ctrl_dist, w_trans_ctrl_transfer and w_trans_ctrl_produce) are similar to "w_trans_ctrl_purchase", hence their program specification are skipped.

## Editing process

**[Window] w_trans_edit_base from Window**
[Description] ancestor window of transaction editing process

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| Window | <w_trans_edit_base> | Menu="m_code_trans_edit "; |
| datawindow | dw_1 | Tab=20; |
| datawindow | dw_2 | Tab=10; |

| Declaration | |
|---|---|
| Instance Variable | boolean i_b_edit = false<br>long il_row_1, il_row_2, il_row<br>integer ii_main_id, ii_row_delete<br>datawindowchild dwc_1, dwc_2, dwc_3<br>string is_autho_id, is_emp_name, is_plant_id, is_plant_id1<br>string is_stock_id, is_stock_id1, is_no, is_check<br>real ir_quan, ir_stock_quan5 |

| Script | *Type* | *Name* |
|---|---|---|
| w_trans_edit_base | Event | Close<br>Closequery<br>Open |

| Script | *Type* | *Name* |
|---|---|---|
| | | ue_cancel ( )<br>ue_delete_dw_2 ( )<br>ue_open ( )<br>ue_retrieve ( )<br>ue_search ( )<br>ue_undelete ( )<br>ue_undo ( )<br>ue_update ( ) |
| | PrivateFunc | wf_undo (datawindow adw_x) return integer |
| Dw_1 | ControlEvent | Dberror<br>Itemchanged |
| Dw_2 | ControlEvent | dberror<br>itemchanged |

*Event: w_trans_edit_base.Close*

```
close(this)
```

*Event: w_trans_edit_base.Closequery*

```
int i_rc

IF (dw_1.ModifiedCount() > 0 OR dw_2.ModifiedCount() > 0 ) and (i_b_edit) Then
        i_rc = Messagebox ("Ask for save", Question!, YesNoCancel!,1)
        CHOOSE CASE i_rc
        CASE 1 // Update
                This.Trigger Event ue_update()
                return
        CASE 3 // Cancel
                return 1
        CASE ELSE // Don't want to update
                return
        END CHOOSE
ELSE
        return
END IF
```

*Event: w_trans_edit_base.Open*

```
This.Post Event ue_open()
```

*Event: w_trans_edit_base.ue_cancel()*

```
See in w_trans_ctrl_base.ue_cancel
```

*Event: w_trans_edit_base.ue_delete_dw_2()*

```
int i_rc

if dw_2.RowCount() > 0 then
        i_rc = MessageBox("Ask for delete",Question!,YesNo!,2)

        IF i_rc = 1 THEN
                il_row = dw_2.getrow()
                ir_quan = dw_2.object.quan[il_row]
                is_plant_id        = dw_2.object.plant_id[il_row]
                is_stock_id = dw_1.GetItemString(dw_1.GetRow(),'stock_id')
```

```
                select quan5 into :ir_stock_quan5 from stock_detail
                where div_id = :gs_div_id and sect_id = :gs_sect_id and stock_id = :is_stock_id and
plant_id = :is_plant_id;

                ii_row_delete = dw_2.deleterow(il_row)

                IF ii_row_delete <> 1 THEN
                        Messagebox("error","can't delete")
                ELSE
                        UPDATE stock_detail SET quan5 = :ir_stock_quan5 + :ir_quan
                        WHERE div_id = :gs_div_id and sect_id = :gs_sect_id and stock_id =
:is_stock_id and plant_id = :is_plant_id;
                END IF
                dw_2.setfocus()
                i_b_edit = true
        ELSE
                return
        END IF
end if
```

*Event: w_trans_edit_base.ue_open()*

```
dw_1.visible = false
dw_2.visible = false
This.Trigger Event ue_search()
```

*Event: w_trans_edit_base.ue_retrieve()*

```
long l_rc

if is_no <> 'n' then
        dw_1.SetTransObject(SQLCA)
        l_rc = dw_1.Retrieve(is_no,gs_div_id,gs_sect_id)
        if l_rc > 0 then
                dw_1.visible = true
                dw_1.enabled = true
                il_row_1 = dw_1.GetRow()
                m_code_trans_edit.m_modi.m_update.enabled = true
                m_code_trans_edit.m_modi.m_undo.enabled = true
                m_code_trans_edit.m_modi.m_cancel.enabled = true
                This.Trigger event ue_retrieve_dw_2()
        else
                dw_1.visible = false
                dw_2.visible = false
                MessageBox('search','not found')
                m_code_trans_edit.m_modi.m_update.enabled = false
                m_code_trans_edit.m_modi.m_undo.enabled = false
                m_code_trans_edit.m_modi.m_cancel.enabled = false
                m_code_trans_edit.m_modi.m_search.enabled = true
        end if
else
        m_code_trans_edit.m_modi.m_update.enabled = false
        m_code_trans_edit.m_modi.m_undo.enabled = false
        m_code_trans_edit.m_modi.m_cancel.enabled = false
        m_code_trans_edit.m_modi.m_search.enabled = true
end if
```

*Event:w_trans_edit_base.ue_search*

```
open(w_trans_q_no,This)
is_no = Message.StringParm

This.Trigger Event ue_retrieve()
```

*Event:w_trans_edit_base.ue_undelete*

```
long ll_row

// Get the last row in the deleted buffer
ll_row = dw_2.DeletedCount()
//Move the last row in the deleted buffer to the primary buffer
dw_2.SetRedraw(false)

if dw_2.RowsMove(ll_row,ll_row,delete!,dw_2, ii_row_delete,primary!) = -1 then
        MessageBox("can't undelete")
else
        //set focus to the row that was "restored"
        dw_2.SetFocus()
        il_row = ii_row_delete
        dw_2.ScrollToRow(il_row)
end if
dw_2.SetReDraw(True)
if il_row <> 0 then
        ir_quan = dw_2.object.quan[il_row]
        is_plant_id = dw_2.object.plant_id[il_row]
        is_stock_id = dw_1.GetItemString(dw_1.GetRow(),'stock_id')

        select quan5 into :ir_stock_quan5 from stock_detail
        where div_id = :gs_div_id and sect_id = :gs_sect_id and stock_id = :is_stock_id and plant_id
= :is_plant_id;

        UPDATE stock_detail SET quan5 = :ir_stock_quan5 - :ir_quan
        WHERE div_id = :gs_div_id and sect_id = :gs_sect_id and stock_id = :is_stock_id and
plant_id = :is_plant_id;
end if
```

*Event:w_trans_edit_base.ue_undo*

```
if wf_undo(dw_1) = 0 then
        wf_undo(dw_2)
end if
```

*Event: w_trans_edit_base.ue_update()*

```
IF dw_1.Update() = 1 Then
        commit using sqlca;
        if dw_2.Update() = 1 then
                commit using sqlca;
                messagebox("save")
                m_code_trans_edit.m_modi.m_search.enabled = true
                i_b_edit = false
        else
                rollback using sqlca;
                messagebox("error")
                return
        end if
```

```
else
        rollback using sqlca;
        messagebox("error")
        return
end if
```

*PrivateFunction: w_trans_edit_base.* wf_undo (datawindow adw_x) return integer

Get the original value of adw_x row/column
Reset it to the original value
Reset the modified flag for adw_x row

*ControlEvent: dw_1.dberror*

See in w_code_seed.dw_data2.dberror

*ControlEvent: dw_1.Itemchanged*

m_code_trans_edit.m_modi.m_search.enabled = false
i_b_edit = True

*ControlEvent: dw_2.dberror*

See in w_code_seed.dw_data2.dberror

*ControlEvent: dw_2.itemchanged*

m_code_trans_edit.m_modi.m_search.enabled = false
i_b_edit = True

**[Window] w_trans_edit_dist from w_trans_edit_base**
[Description] Edit distribution data

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| w_trans_edit_base | <w_trans_edit_dist> | Title="Edit distribution"; |
| w_trans_edit_base`dw_1 | dw_1 | DataObject="d_dist_edit "; |
| w_trans_edit_base`dw_2 | dw_2 | DataObject="d_dist_detail_edit "; |

| Script | *Type* | *Name* |
|---|---|---|
| w_trans_edit_dist | Event | Open<br>ue_open<br>ue_retrieve_dw_2 |
| dw_1 | ControlEvent | Itemfocuschanged |

*Event: w_trans_edit_dist.Open*

Select user's name is using system.
this.title = "Edit distribution" + "(user's name)"

*Event: w_trans_edit_dist.ue_open*

```
Call super::ue_open
int i_rc

i_rc = dw_1.getChild('emp_id1',dwc_1) // successful = 1
if i_rc <> 1 Then
        MessageBox("error")
Else
```

```
  dwc_1.SetTransObject(SQLCA)
  dwc_1.Retrieve()
end if
```

*Event: w_trans_edit_dist.ue_retrieve_dw_2*

```
dw_2.SetTransObject(SQLCA)
dw_2.Retrieve(is_no,gs_div_id,gs_sect_id)
dw_2.visible = true

long     ll_row
int      li_chk

ll_row = dw_2.RowCount()
li_chk = 0

For il_row_2 = 1 to ll_row
        is_check = dw_2.object.chk[il_row_2]
        if is_check = 'n' then
                li_chk = li_chk + 1
        end if
Next
if li_chk = ll_row then
        messagebox("checking")
        This.Post Event ue_open()
end if
```

*ControlEvent: dw_1.Itemfocuschanged*

```
string   ls_column, ls_expression
int      i_rc

ls_column = dwo.name
if ls_column = 'emp_id1' then
        select autho_id into :is_autho_id from employee
        where autho_id = 01;

        ls_expression = "autho_id = '"+is_autho_id+"'"
        i_rc = dwc_1.SetFilter(ls_expression)
        if i_rc <> 1 then
                MessageBox("error")
        else
                dwc_1.filter()
        end if
END IF
```

Other windows that inherit from w_trans_edit_base (w_trans_edit_transfer, w_trans_edit_purchase and w_trans_edit_produce) are similar to "w_trans_edit_dist", hence their program specification are skipped.

| **[Window] w_trans_edit_base1 from Window**<br>[Description] ancestor window of transaction editing process (Only w_trans_edit_stock ) |
|---|

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| Window | <w_trans_edit_base1> | Menu="m_code_trans_edit "; |
| datawindow | dw_1 | Tab=10; |
| datawindow | dw_2 | Tab=20; |

| Declaration | |
|---|---|
| Instance Variable | boolean i_b_edit = false<br>long il_row_1, il_row_2, il_row<br>integer ii_main_id<br>datawindowchild dwc_1, dwc_2, dwc_3<br>string is_autho_id, is_emp_name, is_plant_id, is_plant_id1<br>string is_stock_id, is_stock_id1, is_no, is_check |

| Script | *Type* | *Name* |
|---|---|---|
| w_trans_edit_base1 | Event | Close<br>Closequery<br>Open<br>ue_cancel ( )<br>ue_open ( )<br>ue_retrieve ( )<br>ue_search ( )<br>ue_undo ( )<br>ue_update ( ) |
| | PrivateFunc | wf_undo (datawindow adw_x) return integer |
| Dw_1 | ControlEvent | Dberror<br>Itemchanged |
| Dw_2 | ControlEvent | dberror<br>itemchanged |

*Event: w_trans_edit_base1.Close*

```
close(this)
```

*Event: w_trans_edit_base1.Closequery*

```
int i_rc

IF (dw_1.ModifiedCount() > 0 OR dw_2.ModifiedCount() > 0 ) and (i_b_edit) Then

        i_rc = Messagebox ("Ask for save ", Question!, YesNoCancel!,1)
        CHOOSE CASE i_rc
        CASE 1 // Update
                This.Trigger Event ue_update()
                return
        CASE 3 // Cancel
                return 1
        CASE ELSE // Don't want to update
                return
        END CHOOSE
ELSE
        return
END IF
```

*Event: w_trans_edit_base1.Open*

```
This.Post Event ue_open()
```

*Event: w_trans_edit_base1.ue_cancel()*

```
See in w_trans_ctrl_base.ue_cancel
```

*Event: w_trans_edit_base1.ue_open()*

```
dw_1.visible = false
dw_2.visible = false
This.Trigger Event ue_search()
```

*Event: w_trans_edit_base1.ue_retrieve()*

```
long l_rc

if is_no <> 'n' then
        dw_1.SetTransObject(SQLCA)
        l_rc = dw_1.Retrieve(is_no,gs_div_id,gs_sect_id)
        if l_rc > 0 then
                dw_1.visible = true
                il_row_1 = dw_1.GetRow()
                m_code_trans_edit.m_modi.m_update.enabled = true
                m_code_trans_edit.m_modi.m_undo.enabled = true
                m_code_trans_edit.m_modi.m_cancel.enabled = true
                This.Trigger event ue_retrieve_dw_2()
        else
                dw_1.visible = false
                dw_2.visible = false
                MessageBox('search','not found')
                m_code_trans_edit.m_modi.m_update.enabled = false
                m_code_trans_edit.m_modi.m_undo.enabled = false
                m_code_trans_edit.m_modi.m_cancel.enabled = false
                m_code_trans_edit.m_modi.m_search.enabled = true

        end if
else
        m_code_trans_edit.m_modi.m_update.enabled = false
        m_code_trans_edit.m_modi.m_undo.enabled = false
        m_code_trans_edit.m_modi.m_cancel.enabled = false
        m_code_trans_edit.m_modi.m_search.enabled = true
end if
```

*Event:w_trans_edit_base1.ue_search*

```
open(w_trans_q_no,This)
is_no = Message.StringParm

m_code_trans_edit.m_modi.m_update.enabled = true
dw_1.enabled = true
This.Trigger Event ue_retrieve()
```

*Event:w_trans_edit_base1.ue_undo*

```
if wf_undo(dw_1) = 0 then
        wf_undo(dw_2)
end if
```

*Event: w_trans_edit_base1.ue_update()*

```
IF dw_1.Update() = 1 Then
        commit using sqlca;
        if dw_2.Update() = 1 then
                commit using sqlca;
                messagebox("save")
                m_code_trans_edit.m_modi.m_search.enabled = true
                i_b_edit = false
        else
                rollback using sqlca;
                messagebox("error")
                return
        end if
else
        rollback using sqlca;
        messagebox("error")
        return
end if
```

*PrivateFunction: w_trans_edit_base1.* wf_undo (datawindow adw_x) return integer

Get the original value of adw_x row/column
Reset it to the original value
Reset the modified flag for adw_x row

*ControlEvent: dw_1.dberror*

See in w_code_seed.dw_data2.dberror

*ControlEvent: dw_1.Itemchanged*

m_code_trans_edit.m_modi.m_search.enabled = false
i_b_edit = True

*ControlEvent: dw_2.dberror*

See in w_code_seed.dw_data2.dberror

*ControlEvent: dw_2.itemchanged*

m_code_trans_edit.m_modi.m_search.enabled = false
i_b_edit = True

**[Window] w_trans_edit_stock from w_trans_edit_base1**
[Description] Edit stock data

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| w_trans_edit_base1 | <w_trans_edit_stock> | Title="Edit stock"; |
| w_trans_edit_base1`dw_1 | dw_1 | DataObject="d_stock_edit "; |
| w_trans_edit_base1`dw_2 | dw_2 | DataObject="d_stock_detail_edit "; |

| Script | *Type* | *Name* |
|---|---|---|
| w_trans_edit_stock | Event | Open<br>ue_open<br>ue_retrieve_dw_2 |
| dw_1 | ControlEvent | Itemfocuschanged |

*Event: w_trans_edit_stock.Open*

```
Select user's name is using system.
this.title = "Edit stock" + "(user's name)"
```

*Event: w_trans_edit_stock.ue_open*

```
Call super::ue_open
int i_rc

i_rc = dw_1.getChild('region_id',dwc_1) // successful = 1
if i_rc <> 1 Then
        MessageBox("error")
Else
  dwc_1.SetTransObject(SQLCA)
  dwc_1.Retrieve()
end If

i_rc = dw_1.getChild('prov_id',dwc_2) // successful = 1
if i_rc <> 1 Then
        MessageBox("error")
Else
  dwc_2.SetTransObject(SQLCA)
  dwc_2.Retrieve()
end If
i_rc = dw_1.getChild('amphoe_id',dwc_3) // successful = 1
if i_rc <> 1 Then
        MessageBox("error")
Else
  dwc_3.SetTransObject(SQLCA)
  dwc_3.Retrieve()
end If
```

*Event: w_trans_edit_stock.ue_retrieve_dw_2*

```
dw_2.SetTransObject(SQLCA)
dw_2.Retrieve(is_no,gs_div_id,gs_sect_id)
dw_2.visible = true
```

*ControlEvent: dw_1.Itemfocuschanged*

```
string   ls_column, ls_expression, ls_expression1
int      i_rc

ls_column = dwo.name
IF ls_column = 'prov_id' THEN
        is_region_id = This.GetItemString(This.GetRow(),'region_id')
        IF isNull (is_region_id) then
                MessageBox("error")
                return
        end if
        ls_expression = "region_id = '"+is_region_id+"'"
        i_rc = dwc_2.SetFilter(ls_expression)
        if i_rc <> 1 then
                MessageBox("error")
        else
                dwc_2.filter()
        end if
```

*Event: w_trans_edit_stock.Open*

```
elseif ls_column = 'amphoe_id' then
        is_prov_id = This.GetItemString(This.GetRow(),'prov_id')
        IF isNull (is_prov_id) then
                MessageBox("error")
                return
        end if
        ls_expression1 = "region_id = '"+is_region_id+"' and prov_id = '"+is_prov_id+"'"
        i_rc = dwc_3.SetFilter(ls_expression1)
        if i_rc <> 1 then
                MessageBox("error")
        else
                dwc_3.filter()
        end if
END IF
```

**[Window] w_trans_q_no from Window**
[Description] Get editing number of transaction and display the data to edit

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| Window | <w_trans_q_no> | Type=response!; Title="Get number"; |
| Commandbutton | cb_cancel | Tab=40; Text="Cancel"; |
| commandbutton | cb_ok | Tab=30; Text="OK"; |
| editmask | em_no | Tab=10; |
| groupbox | gb_1 | Tab=50; Text="Enter requested number"; |

| Declaration | |
|---|---|
| Instance Variable | string is_no = 'n' |

| Script | *Type* | *Name* |
|---|---|---|
| w_trans_q_no | Event | Close |
| cb_cancel | ControlEvent | Clicked |
| cb_ok | ControlEvent | Clicked |

*Event: w_trans_q_no.Close*

```
CloseWithReturn(This,is_no)
```

*ControlEvent:cb_ok.clicked*

```
IF isNull(em_no.text) then
        MessageBox(Warn to enter number)
Else
        is_no = em_no.text
        CloseWithReturn(Parent,is_no)
end if
```

*ControlEvent: cb_cancel.clicked*

```
is_no = 'n'
CloseWithReturn(Parent,is_no)
```

## 3. Query and Report

These windows create pre-define results and reports. These outputs devide into 3 style: tabular, graph and report. Tabular has a windows, which display overall or deep in detail data. Graph has an ancestor window, a descendent window and 2 shared window. Report has a window to preview and print. All 3 styles have to call w_set_agr window that is described in section 4. Shared window.

### Tabular result

| **[Window] w_query from Window**<br>[Description] Display inquiry window |
|---|

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| Window | <w_query> | Menu=" m_code_query"; |
| datawindow | dw_1 | Tab=10; |

| Declaration | |
|---|---|
| Instance Variable | str_q_plant istr_x<br>string is_emp_name, is_local_name, is_stock_name |

| Script | *Type* | *Name* |
|---|---|---|
| w_query | Event | Open<br>ue_open ( )<br>ue_retrieve_all ( )<br>ue_retrieve_specity ( )<br>ue_select_name ( ) |

*Event: w_query.open*

```
select employee.name into :is_emp_name from employee
where employee.emp_id = :gs_emp_id;

Post Event ue_open()
```

*Event: w_query.ue_open()*

```
istr_x = Message.PowerObjectParm

//ค่า istr_data.s_status เป็นตัวตรวจสอบว่าจะเปิด datawindow ตัวใด โดยมีค่า ดังนี้

CHOOSE CASE istr_x.s_status
        CASE '1' //1 - เป็นการสอบถามคลังเก็บพันธุ์พืชที่มีพันธุ์พืชแต่ละชนิดอยู่ (ทุกชนิดพันธุ์พืช)
                set title and user's name
                dw_1.dataobject = 'dq_gp_plant_stock_all'
                dw_1.SetTransObject(SQLCA)
                dw_1.Retrieve(istr_x.s_div_id,istr_x.s_sect_id)

        CASE '2' //2 - เป็นการสอบถามคลังเก็บพันธุ์พืชที่มีพันธุ์พืชแต่ละชนิดอยู่ (ระบุชนิดพันธุ์พืช)
                this.trigger event ue_select_name()
```

```
            set title and user's name
            dw_1.dataobject = 'dq_gp_plant_stock_specify'
            dw_1.SetTransObject(SQLCA)
            dw_1.Retrieve(istr_x.s_div_id,istr_x.s_sect_id,istr_x.s_plant_id)

CASE '3' //3 - เป็นการสอบถามพันธุ์พืชแยกตามคลังเก็บพันธุ์พืชในแต่หน่วยงาน
            set title and user's name
            dw_1.dataobject = 'dq_gp_stock_plant'
            dw_1.SetTransObject(SQLCA)
            dw_1.Retrieve(istr_x.s_div_id,istr_x.s_sect_id,istr_x.s_stock_id)

CASE '4' //4 - เป็นการสอบถามพันธุ์พืชที่มีการเบิก-จ่ายในแต่ละช่วงเวลา (ทุกชนิดพันธุ์พืช)
            set title and user's name
            dw_1.dataobject = 'dq_gp_dist_all'
            this.trigger event ue_retrieve_all()

CASE '5' //5 - เป็นการสอบถามพันธุ์พืชที่มีการเบิก-จ่ายในแต่ละช่วงเวลา (ระบุชนิดพันธุ์พืช)
            this.trigger event ue_select_name()
            set title and user's name
            dw_1.dataobject = 'dq_gp_dist_specify'
            this.trigger event ue_retrieve_specify()

CASE '6' //6 - เป็นการสอบถามพันธุ์พืชที่มีการโอนในแต่ละช่วงเวลา (ทุกชนิดพันธุ์พืช)
            set title and user's name
            dw_1.dataobject = 'dq_gp_tran_all'
            this.trigger event ue_retrieve_all()

CASE '7' //7 - เป็นการสอบถามพันธุ์พืชที่มีการโอนในแต่ละช่วงเวลา (ระบุชนิดพันธุ์พืช)
            this.trigger event ue_select_name()
            set title and user's name
            dw_1.dataobject = 'dq_gp_tran_specify'
            this.trigger event ue_retrieve_specify()

CASE '8' //8 - เป็นการสอบถามพันธุ์พืชที่มีการจัดซื้อในแต่ละช่วงเวลา (ทุกชนิดพันธุ์พืช)
            set title and user's name
            dw_1.dataobject = 'dq_gp_pur_all'
            this.trigger event ue_retrieve_all()

CASE '9' //9 - เป็นการสอบถามพันธุ์พืชที่มีการจัดซื้อในแต่ละช่วงเวลา (ระบุชนิดพันธุ์พืช)
            this.trigger event ue_select_name()
            set title and user's name
            dw_1.dataobject = 'dq_gp_pur_specify'
            this.trigger event ue_retrieve_specify()

CASE '10' //10 - เป็นการสอบถามพันธุ์พืชที่มีการผลิตในแต่ละช่วงเวลา (ทุกชนิดพันธุ์พืช)
            set title and user's name
            dw_1.dataobject = 'dq_gp_prod_all'
            this.trigger event ue_retrieve_all()

CASE '11' //11 - เป็นการสอบถามพันธุ์พืชที่มีการผลิตในแต่ละช่วงเวลา (ระบุชนิดพันธุ์พืช)
            this.trigger event ue_select_name()
```

```
            set title and user's name
            dw_1.dataobject = 'dq_gp_prod_specify'
            this.trigger event ue_retrieve_specify()

        CASE '12' //12 - เป็นการสอบถามพันธุ์พืชที่ต้องจัดหาเพิ่มในแต่ละคลังเก็บพันธุ์พืช (ทุกชนิดคลังเก็บพันธุ์พืช)
            set title and user's name
            dw_1.dataobject = 'dq_gp_add_plant_all'
            dw_1.SetTransObject(SQLCA)
            dw_1.Retrieve(istr_x.s_div_id,istr_x.s_sect_id)

        CASE '13' //13 - เป็นการสอบถามพันธุ์พืชที่ต้องจัดหาเพิ่มในแต่ละคลังเก็บพันธุ์พืช (ระบุชนิดคลังเก็บพันธุ์พืช)
            select name1 into :is_stock_name
            from stock where stock_id = :istr_x.s_stock_id;

            set title and user's name
            dw_1.dataobject = 'dq_gp_add_plant_specify'
            dw_1.SetTransObject(SQLCA)
            dw_1.Retrieve(istr_x.s_div_id,istr_x.s_sect_id,istr_x.s_stock_id)

        CASE '14' //14 - เป็นการสอบถามพันธุ์พืชที่มีอยู่ในคลังในแต่ละช่วงเวลา (ทุกคลังเก็บพันธุ์พืช)
            set title and user's name
            dw_1.dataobject = 'dq_gp_net_div_sect'
            this.trigger event ue_retrieve_all()

        CASE '15' //15 - เป็นการสอบถามพันธุ์พืชที่มีอยู่ในคลังในแต่ละช่วงเวลา (ระบุคลังเก็บพันธุ์พืช)
            select name1 into :is_stock_name
            from stock where stock_id = :istr_x.s_stock_id;

            set title and user's name
            dw_1.dataobject = 'dq_gp_net_stock'
            dw_1.SetTransObject(SQLCA)
            dw_1.Retrieve(istr_x.s_div_id,istr_x.s_sect_id,istr_x.s_stock_id,
            istr_x.d_startdate,istr_x.d_stopdate)
END CHOOSE
```

*Event: w_query.ue_retrieve_all*

```
dw_1.SetTransObject(SQLCA)
dw_1.Retrieve(istr_x.s_div_id,istr_x.s_sect_id,istr_x.d_startdate,istr_x.d_stopdate)
```

*Event: w_query.ue_retrieve_specity*

```
dw_1.SetTransObject(SQLCA)
dw_1.Retrieve(istr_x.s_div_id,istr_x.s_sect_id,istr_x.s_plant_id,istr_x.d_startdate,istr_x.d_stopdate)
```

*Event: w_query.ue_select_name*

```
select local_name into :is_local_name
from plant where plant_id = :istr_x.s_plant_id;
```

## Graph result

| **[Window] w_graph_base from Window**<br>[Description] ancestor window for graph result in query process |
|---|

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| Window | <w_graph_base> | Menu="m_code_graph"; |
| datawindow | dw_1 | Tab=10; |
| statictext | st_1 | Text="click right button to display graph value"; |
| statictext | st_popup | Invisible; Text="Popup"; |

| Script | *Type* | *Name* |
|---|---|---|
| w_graph_base | Event | Close<br>Open<br>ue_graph_type<br>ue_spacing |
| dw_1 | Event | ue_rbuttonup pbm_rbuttonup |
| | ControlEvent | Rbuttondown |

*Event: w_graph_base.Close*

```
Close(This)
```

*Event: w_graph_base.Open*

```
Post event ue_open()
```

*Event: w_graph_base.ue_graph_type*

```
OpenWithParm (w_graph_type, dw_1)
```

*Event: w_graph_base.ue_spacing*

```
OpenWithParm (w_graph_spacing, dw_1)
```

*Event: dw_1.ue_rbuttonup pbm_rbuttonup*

```
st_popup.visible = false
```

*ControlEvent: dw_1.Rbuttondown*

```
grObjectType    ClickedObject
string          ls_grgraphname="gr_1"
int             li_series, li_category
ClickedObject = this.ObjectAtPointer (ls_grgraphname, li_series, li_category)
If ClickedObject = TypeData! Then
        st_popup.text = string(this.GetData(ls_grgraphname, li_series, li_category)) + " units"
        st_popup.x = parent.PointerX()
        st_popup.y = parent.PointerY() - 65
        st_popup.visible = true
End If
```

**[Windoww_graph from w_graph_base**
[Description] Display graph window

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| w_graph_base | <w_graph> | Menu="m_code_graph"; |
| w_graph_base`dw_1 | dw_1 | Tab=10; |

| Declaration | |
|---|---|
| Instance Variable | string is_emp_name |

| Script | *Type* | *Name* |
|---|---|---|
| w_graph | Event | Open<br>ue_open |

*Event: w_graph.Open*

```
Select user's name is using system.
this.title = "Summary seed and seedling quantity graph" + "(user's name)"
```

*Event: w_graph.ue_open*

```
str_report str_x
string ls_rc

str_x = Message.PowerObjectParm
dw_1.dataobject = str_x.s_dataobject
dw_1.SetTransObject(SQLCA)

CHOOSE CASE str_x.i_style
        CASE 1
                dw_1.Retrieve(gstr_data.s_div_id,gstr_data.s_sect_id)
        CASE 2
                dw_1.Retrieve(gstr_data.s_div_id,gstr_data.s_sect_id, gstr_data.s_stock_id)
        CASE 3
                dw_1.Retrieve(gstr_data.s_div_id,gstr_data.s_sect_id,gstr_data.d_startdate,
gstr_data.d_stopdate)
        CASE 4
                dw_1.Retrieve(gstr_data.s_div_id,gstr_data.s_sect_id,gstr_data.s_stock_id,
gstr_data.d_startdate, gstr_data.d_stopdate)
END CHOOSE
```

**[Window] w_graph_type from Window**
[Description] General response window to modify a graph type

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| Window | <w_graph_type> | Type=response!; Title="Graph Type"; |
| u_graph_gallery | uo_1 | |

| Declaration | |
|---|---|
| Instance Variable | graph igr_parm<br>datawindow idw_parm<br>object io_passed |

| Script | *Type* | *Name* |
|---|---|---|
| w_graph_type | Event | open |
| uo_1 | ControlEvent | gallery_cancel gallery_ok |

*Event: w_graph_type.open*

```
// Receive and remember in the igr_parm or idw_parm instance variable
Graphicobject lgro_hold
Lgro_hold = message.powerobjectparm
If lgro_hold.TypeOf() = Graph! Then
        io_passed = Graph! And   igr_parm = message.powerobjectparm
Elseif lgro_hold.TypeOf() = Datawindow! Then
        io_passed = Datawindow and idw_parm = message.powerobjectparm
End If
```

*ControlEvent:uo_1.gallery_cancel gallery_ok*

```
Get the graph type from the graph gallery user object.
Set the type in the passed graph object.
Close (parent)
```

**[Window] w_graph_spacing from Window**
[Description] General response window to set graph spacing

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| Window | <w_graph_spacing> | Type=response!; Title=" graph spacing"; |
| commandbutton | cb_cancel | Tab=20; Text= "Cancel"; |
| commandbutton | cb_ok | Tab=30; Text="OK"; |
| editmask | em_spacing | Tab=10; |

| Declaration | |
|---|---|
| Instance Variable | object io_passed<br>graph igr_parm<br>datawindow idw_parm<br>int ii_original_spacing |

| Script | *Type* | *Name* |
|---|---|---|
| w_graph_spacing | Event | open |
| cb_cancel | ControlEvent | Clicked |
| cb_ok | ControlEvent | Clicked |
| em_spacing | Event | ue_enchange pbm_enchange |

*Event: w_graph_spacing.open*

```
Graphicobject lgro_hold
lgro_hold = Message.PowerObjectParm
If lgro_hold.TypeOf() = Graph! Then
        io_passed = Graph! And   igr_parm = Message.PowerObjectParm
        em_spacing.text = string(igr_parm.spacing) and ii_original_spacing = igr_parm.spacing
Elseif lgro_hold.TypeOf() = Datawindow! Then
        io_passed = Datawindow! And  idw_parm = Message.PowerObjectParm
        em_spacing.text = idw_parm.Object.gr_1.spacing
        ii_original_spacing = Integer(em_spacing.text)
End If
```

| *Event: em_spacing.ue_enchange pbm_enchange* |
|---|
| If io_passed = Graph! Then<br>igr_parm.spacing = integer (em_spacing.text)<br>Elseif io_passed = Datawindow! Then<br>idw_parm.Object.gr_1.spacing = em_spacing.text<br>End If |
| *ControlEvent: cb_cancel.Clicked* |
| If io_passed = Graph! Then<br>igr_parm.spacing = ii_original_spacing<br>Elseif io_passed = Datawindow! Then<br>idw_parm.Object.gr_1.spacing= string(ii_original_spacing)<br>End If<br>Close (parent) |
| *ControlEvent: cb_ok.Clicked* |
| Close (parent) |

## 4. Shared windows

These windows derive from shared functions in above-mentioned section. Shared windows are w_db_error, w_login, w_print_base, w_set_argument and w_about.

| **[Window] w_db_error from Window**<br>[Description] Display error of data window |
|---|

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| Window | <w_db_error> | Type=response!; Title="Error"; |
| commandbutton | cb_1 | Tab=30; Text="Close"; |
| multilineedit | mle_error_detail | Tab=10; |
| multilineedit | mle_fix_detail | Tab=20; |
| statictext | st_1 | Text="Table"; |
| statictext | st_2 | Text="Row"; |
| statictext | st_3 | Text="Error detail"; |
| Statictext | st_5 | Text="Fix detail"; |
| Statictext | st_row | |
| Statictext | st_table | |

| Script | *Type* | *Name* |
|---|---|---|
| w_db_error | Event | Open |
| cb_1 | ControlEvent | Clicked |

| *Event: w_db_error.open* |
|---|
| str_db_error error_FromDW<br>error_FromDW = Message.PowerObjectParm<br>Convert error_FromDW to thai string<br>Display thai string |

| *ControlEvent: cb_1.Clicked* |
|---|
| Close(Parent) |

| **[Window] w_login from Window** |
|---|
| [Description] Validate access level |

| **Controls** | *Control Name* | *Remarks* |
|---|---|---|
| Window | <w_login> | Title="Login to seed and seedling management system"; |
| commandbutton | cb_close | Tab=40; Text="Cancel"; |
| commandbutton | cb_ok | Tab=30; Text="OK"; |
| singlelineedit | sle_password | Tab=20; |
| singlelineedit | sle_userid | Tab=10; |
| statictext | st_userid | Text="User name:"; |
| statictext | st_password | Text="Password:"; |

| **Declaration** | |
|---|---|
| Instance Variable | integer i<br>string is_group_id,is_user_id<br>string is_login_state = "NO" |

| **Script** | *Type* | *Name* |
|---|---|---|
| w_login | Event | close |
| cb_close | ControlEvent | clicked |
| cb_ok | ControlEvent | clicked |

| *Event: w_login.Close* |
|---|
| CloseWithReturn(This, is_login_state) |
| *ControlEvent.cb_close.Clicked* |
| close(w_login) |
| *ControlEvent.cb_ok.clicked* |

```
string ls_passwd, ls_emp_id, ls_level

i = i +1
if i <= 3 then
        If sle_userid.text <> "" Then
                if sle_password.text <> "" then

                        Select password,emp_id,group_id into :ls_passwd, :ls_emp_id, :ls_level
From userdata Where user_name = :sle_userid.text;

                        if ls_passwd > " " then
                                if ls_passwd = sle_password.text then
                                        is_login_state = "YES"
                                        select div_id, sect_id into :gs_div_id, :gs_sect_id From
employee where emp_id = :ls_emp_id;
                                        if isNumber(gs_div_id) and Len(gs_div_id) = 3 and &
                                                isNumber(gs_sect_id) and Len(gs_sect_id) = 3
then
```

```
                                        gs_level = ls_level
                                        gs_emp_id = ls_emp_id
                                        OPEN ( g_w_mdi, g_s_mdi )
                                        close(parent)
                                else
                                        MessageBox("error")
                                end if
                        else
                                sle_password.setfocus()
                                MessageBox("error")
                        end if
                else
                        sle_userid.setfocus()
                        MessageBox("error")
                end if
        else
                sle_password.setfocus()
                Messagebox("error")
        end if
else
        sle_userid.setfocus()
        MessageBox("error")
end if
else
        MessageBox("error")
        halt close
end if
```

**[Window] w_set_argument from Window**
[Description] Setting section, hardware specification, date, year argument for query and report

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| Window | <w_set_argument> | Title=" Set argument value"; |
| commandbutton | cb_log_sect | Text=" login section"; |
| datawindow | dw_sect | DataObject="d_ff_sect_select"; |
| datawindow | dw_plant | DataObject="d_ff_plant_select"; |
| datawindow | dw_stock | DataObject="d_ff_stock_select"; |
| editmask | em_startdate | Disable; |
| editmask | em_stopdate | Disable |
| Picturebutton | pb_date_ok | |
| Picturebutton | pb_plant_ok | |
| Picturebutton | pb_sect_ok | |
| Picturebutton | pb_stock_ok | |
| radiobutton | rb_range | Text="Date"; |
| radiobutton | rb_today | Text="Today"; |
| statictext | st_date | Text="to"; |
| tab | tab_query | Tab=10; |
| userobject | tabpage_date | Text="Date"; |
| userobject | tabpage_plant | Text="Plant"; |
| userobject | tabpage_sect | Text="section"; |
| userobject | tabpage_stock | Text="Stock"; |

| Declaration | |
|---|---|
| Instance Variable | datawindowchild dwc, dwc_1,dwc_2<br>boolean ib_today |

| Script | *Type* | *Name* |
|---|---|---|
| w_set_argument | Event | Open<br>ue_open ( ) |
| cb_log_sect | ControlEvent | Clicked |
| dw_plant | Event | ue_get_plant_id ( ) |
| dw_sect | Event | ue_find ( string as_div_id, string as_sect_id )<br>ue_get_sect_id ( ) |
| | ControlEvent | Itemfocuschanged |
| dw_stock | Event | ue_get_stock_id ( ) |
| | ControlEvent | Getfocus |
| em_stopdate | ControlEvent | Losefocus |
| pb_date_ok | ControlEvent | Clicked |
| pb_plant_ok | ControlEvent | Clicked |
| pb_sect_ok | ControlEvent | Clicked |
| pb_stock_ok | ControlEvent | Clicked |
| rb_range | ControlEvent | Clicked |
| rb_today | ControlEvent | Clicked |
| tab_query | Event | ue_date_enable ( boolean ab_flag ) |

*Event:w_set_argument.open*

```
This.Post Event ue_open()
```

*Event:w_set_argument.ue_open()*

```
int i_rc
string ls_parm

ls_parm = message.stringParm

if ls_parm = 'report' then
        This.ChangeMenu(m_code_report)
        Tab_query.tabpage_plant.visible = false
        Tab_query.tabpage_stock.visible = false
elseif ls_parm = 'query' then
        This.ChangeMenu(m_code_query)
elseif ls_parm = 'graph' then
        This.ChangeMenu(m_code_graph)
        Tab_query.tabpage_plant.visible = false
end if

//set data of tabpage_sect
i_rc = tab_query.tabpage_sect.dw_sect.getChild('sect_id',dwc)
if i_rc <> 1 then
        MessageBox("error")
Else
        dwc.SetTransObject(SQLCA)
        dwc.Retrieve()
end If

tab_query.tabpage_sect.dw_sect.SetTransObject(SQLCA)
```

```
tab_query.tabpage_sect.dw_sect.Retrieve()
tab_query.tabpage_sect.dw_sect.Post event ue_find(gstr_data.s_div_id,gstr_data.s_sect_id)

//set data of tabpage_stock
i_rc = tab_query.tabpage_stock.dw_stock.getChild('stock_id',dwc_1)
if i_rc <> 1 then
        MessageBox("error")
Else
        dwc_1.SetTransObject(SQLCA)
        dwc_1.Retrieve()
end If

tab_query.tabpage_stock.dw_stock.SetTransObject(SQLCA)
tab_query.tabpage_stock.dw_stock.Retrieve()

//set data of tabpage_plant
i_rc = tab_query.tabpage_plant.dw_plant.getChild('plant_id',dwc_2)
if i_rc <> 1 then
        MessageBox("error")
Else
        dwc_2.SetTransObject(SQLCA)
        dwc_2.Retrieve()
end If

tab_query.tabpage_plant.dw_plant.SetTransObject(SQLCA)
tab_query.tabpage_plant.dw_plant.Retrieve()
```

*ControlEvent: cb_log_sect.clicked*

```
gstr_data.s_div_id = gs_div_id
gstr_data.s_sect_id = gs_sect_id
dw_sect.Trigger Event ue_find(gs_div_id, gs_sect_id)
```

*Event: dw_plant.ue_get_plant_id()*

```
long ll_row

ll_row = This.getRow()
gstr_data.s_plant_id = This.GetItemString(ll_row, "plant_id")
if (gstr_data.s_plant_id <> ' ') then
        return
else
        MessageBox("warning")
        This.SetFocus()
end if
```

*Event: dw_sect.ue_find (string as_div, string as_sect_id)*

```
ll_row = This.Find("div_id = '"+as_div_id+"'and sect_id = '"+as_sect_id+"'",1,This.RowCount())
if ll_row > 0 then
        set as_div_id , as_sect_id to ddw_sect
        This.ScrollToRow(ll_row)
end if
```

*Event: dw_sect.ue_get_sect_id()*

```
ll_row = This.getRow()
gstr_data.s_div_id = This.GetItemString(ll_row, "div_id")
```

```
gstr_data.s_sect_id = This.GetItemString(ll_row, "sect_id")
```

*ControlEvent: dw_sect.Itemfocuschanged*

```
ll_row = This.GetRow()
IF dwo.name = 'sect_id' THEN
        ls_div_id = This.object.div_id[ll_row]
        dwc.SetFilter("div_id = '"+ls_div_id+"'") and filter
end if
```

*Event: dw_sect.ue_get_stock_id()*

```
long ll_row

ll_row = This.getRow()
gstr_data.s_stock_id = This.GetItemString(ll_row, "stock_id")

if (gstr_data.s_stock_id <> ' ') then
        return
else
        MessageBox("warning")
        This.SetFocus()
end if
```

*ControlEvent: dw_stock.Getfocus*

```
long ll_row
string ls_div_id, ls_sect_id, ls_expression
int i_rc

ls_div_id = gstr_data.s_div_id
ls_sect_id = gstr_data.s_sect_id

IF isNull (ls_div_id) then
        MessageBox("error")
        Return
end if

IF isNull (ls_sect_id) then
        MessageBox("error")
        Return
end if
ls_expression = "div_id = '"+ls_div_id+"' and sect_id = '"+ls_sect_id+"'"
i_rc = dwc_1.SetFilter(ls_expression)
if i_rc <> 1 then
        MessageBox("error")
else
        dwc_1.filter()
end if
```

*ControlEvent: em_stopdate. Losefocus*

```
if Date(em_stopdate.text) < date(em_startdate.text) then
        ls_temp = em_stopdate.text
        em_stopdate.text = em_startDate.text
        em_startDate.text = ls_temp
end if
```

| *ControlEvent: pb_date_ok. Clicked* |
|---|
| if ib_today then<br>gstr_data.d_startdate = gd_today<br>gstr_data.d_stopdate = gd_today<br>else<br>gstr_data.d_startdate = Date(em_startdate.text)<br>gstr_data.d_stopdate = Date(em_stopdate.text)<br>end if |
| *ControlEvent: pb_plant_ok. Clicked* |
| dw_plant.Post Event ue_get_plant_id() |
| *ControlEvent: pb_sect_ok. Clicked* |
| dw_sect.Post Event ue_get_sect_id() |
| *ControlEvent: pb_stock_ok.clicked* |
| dw_stock.Post Event ue_get_stock_id() |
| *ControlEvent: rb_range .Clicked* |
| ib_today = false<br>tab_query.Post event ue_date_enable(True)<br>em_startdate.SetFocus() |
| *ControlEvent: rb_today.Clicked* |
| ib_today = true<br>tab_query.Post event ue_date_enable(false) |
| *Event: tab_query.ue_date_enable(boolean ab_flag)* |
| tab_query.tabpage_date.em_startdate.enabled = ab_flag<br>tab_query.tabpage_date.em_stopdate.enabled = ab_flag |

**[Window] w_print_base from Window**
[Description] ancestor window for print output from dw_1

| Controls | *Control Name* | *Remarks* |
|---|---|---|
| Window | <w_print_base> | Menu="m_reports_main"; Title="Preview; |
| commandbutton | cb_1 | Tab=60; Text="<<<"; |
| commandbutton | cb_2 | Tab=50; Text=">>>"; |
| commandbutton | cb_last | Tab=40; Text="<<<"; |
| commandbutton | cb_next | Tab=70; Text=">>>"; |
| datawindow | dw_1 | Tab=90; |
| editmask | em_1 | Tab=80; Text="100"; |
| editmask | em_page | Tab=30; |
| groupbox | gb_1 | Tab=20; Text="Zoom:"; |
| groupbox | gb_rows | Tab=10; Text="Page:"; |

| Declaration | |
|---|---|
| Instance Variable | int ii_pagecount |

| Script | *Type* | *Name* |
|---|---|---|
| w_print_base | Event | Open<br>Resize<br>ue_print ( )<br>ue_save_as ( )<br>ue_set_page ( )<br>ue_zoom ( integer ai_size ) |
| cb_1 | ControlEvent | Clicked |
| cb_2 | ControlEvent | Clicked |
| cb_last | Event | clicked pbm_bnclicked |
| cb_next | Event | clicked pbm_bnclicked |
| em_1 | ControlEvent | Modified |

| *Event: w_print_base.open* |
|---|
| This.Post Event ue_open() |
| *Event: w_print_base.resize* |
| dw_1.width = this.width - 100<br>dw_1.height = this.height – 400 |
| *Event: w_print_base.ue_print()* |
| dw_1.print() |
| *Event: w_print_base.ue_save_as().* |
| dw_1.saveas() |
| *Event: w_print_base.ue_set_page()* |
| em_page.text = string(1)<br>ii_pagecount = integer(dw_1.describe("evaluate('pagecount()',1)"))<br>em_page.minmax = "1-"+string(ii_pagecount) |
| *Event: w_print_base.ue_zoom(integer as_size)* |
| dw_1.modify('datawindow.print.preview.zoom = ' + String(ai_size)) |
| *ControlEvent:cb_1.Clicked* |
| if integer(em_1.text) > 10 THEN<br>li_size = integer(em_1.text) -10<br>parent.event trigger ue_zoom(li_size)<br>em_1.text = string(li_size)<br>end if |
| *ControlEvent:cb_2.Clicked* |
| li_size = integer(em_1.text) +10<br>parent.event trigger ue_zoom(li_size)<br>em_1.text = string(li_size) |
| *Event:cb_last.Cliked pbm_bnclicked* |
| if integer(em_page.text) >1 THEN<br>em_page.text = string(integer(em_page.text) - 1)<br>dw_1.scrollpriorpage()<br>end if |
| *Event:cb_next.Clicked pbm_bnclicked* |
| if integer(em_page.text) < ii_pagecount THEN |

```
        em_page.text = string(integer(em_page.text) + 1)
        dw_1.scrollnextpage()
end if
```

*ControlEvent: em_1.Modified*

```
parent.event trigger ue_zoom(integer(this.text))
```

# BIOGRAPHY

| | |
|---|---|
| **NAME** | Miss Warunee Pianlertpoj |
| **DATE OF BIRTH** | 3 September 1975 |
| **PLACE OT BIRTH** | Bangkok, Thailand |
| **INSTITUTIONS ATTENDED** | Rajaphat Institute Suan Sunandha, 1993-1997: Bachelor of Science (Computer Science) Mahidol University, 1997-2000: Master of Science (Technology of Information system management) |