

# **A Practical Design of Network Structured Files and Data Access Methods of Accounts in Accounting Information Systems: Via Application of Triply-Linked Tree Algorithm**

**Huan-Chen Lin, Department of Banking and Finance, Takming University of Science and Technology**

**Yu-Je Lee, Department of Marketing Management, Takming University of Science and Technology**

**Lung-Yu Chang, Dept. of Insurance and Financial Management, Takming University of Science and Technology**

## **ABSTRACT**

*The main purpose of this study is to provide big enterprises or publicly owned institutions with an architecture of an accounting information system which possesses internal audit functions; to fulfill this purpose, this study proposes tree-structured file algorithms which can be used in the industry or publicly owned institutions for developing account subsystems. To deal with the accounts in an accounting system, the design and data access methods - based on a triply-linked tree structure, which has to maintain correctness of the tree indices and file integrity - have been explored in detail. The results show that the file structure and data access methods of the accounts in an accounting information system based on a tree structure linked in a triple manner, should be of practical value.*

**Keywords:** *Triply-Linked Tree Algorithm, Accounting Information Systems (AIS), Auditing*

## **RESEARCH MOTIVATION AND PURPOSES**

American Production and Inventory Control Society (APICS) defines “enterprise resource planning” (ERP) in its eighth edition dictionary published in 1995 as “an accounting-oriented information system used to identify and plan the required overall enterprise resources and make integrated planning for manufacturing, delivery and dealing with customer orders.” However, in developing modular AIS software of account management subsystem, the design of accounting files is a very important subject (Luan & Luo, 2006).

Today, the financial status and operating results of an enterprise are aggregated in an accounting information system, thus the computerized accounting account has been a major backbone of enterprise resource planning (Yuan, 2004). The foundation of account manipulation in AIS lies in the establishment of the account codes. Beginning with transcribing vouchers, every debit/credit ledger entry has to be endowed with a specific account code; from then on, this code will substitute for the traditional account name and form the basis of journalizing and posting. In other words, inside computer operation, the most important track to keep is account codes.

The foundation of today's accounting information system lies in the establishment of computerized account codes, and in big enterprises or publicly-owned institutions, the account codes (mainly according to the budget account code which is made and published by Directorate-General of Budget, Accounting and Statistics, Executive Yuan, R.O.C. for publicly-owned business institutions) often adopt mainly classification numbering, and consist of accounts in general and subsidiary ledgers; the former are encompassing accounts while the latter are subordinate accounts.

In order to show the relationship between the encompassing and the subordinate accounts, there is a need to link up these various accounts. Not only account records of same levels, but account records of different levels are linked with pointers; and these pointers are required to have both downward (child) reference ability to find low-level accounts from high-level accounts, and upward (father) reference ability to find high-level accounts from low-level accounts. Moreover, accounts of the same level can also be easily identified one by one by rightward reference (sibling). Therefore, three pointers are needed to link with in order to indicate the data structure relationship of level precedence between any accounting account and those in the same level or different levels. In order to preserve the correctness and integrity of overall accounting account data in the process of journaling and posting various vouchers, the amounts in accounting accounts of each level must be vertically and horizontally consistent. The triple tree structure contains

tree indexes (Father, Sibling, Child) and hence can satisfy the above requirements. In other words, triple tree structure design has the following three advantages. (1) When input data is entered based on the lowest level account, it will automatically be aggregated into a ledger account. The subtotal account can be obtained from the Father pointer of the lowest level account, then via the Father pointer of the subtotal account, the accounts can be aggregated and updated level by level so as to keep the account amounts in each level vertically and horizontally consistent and preserve the integrity of the master accounting file. (2) To process online query or audit the master accounting file, we need only to know the encompassing account code of the account to be queried or audited; based on the pointers of the same level and the subordinate level and applying depth-first-search method, the needed data can be obtained one by one. The processing sequence using this method is very clear and the end users can have good sense at a glance. (3) When the Indexed Sequential Access File (ISAM) is adopted for the master accounting file, the accounts can be connected by the network structure without constructing many index files, thereby obtaining relevant data (Lin, 1996).

The above cognitions motivated this study, and the main purpose of this study is to apply the design of triple tree structure and the associated tree operations (i.e. account addition, deletion, renaming, querying, and auditing) to efficiently and correctly build an architecture for accounting information systems that possesses internal auditing functions.

## LITERATURE REVIEW

Concerning internal audit literature, many scholars have previously engaged in this area of research; most of them explored manual accounting system processes, and for the study of applying computer-assisted audit techniques, Chen (1995) indicated whether the internal control mechanism can effectively and continuously operate depends on how the internal audit function works; in that study seven variables, which comprised computer-assisted audit techniques, auditing staff's involvement in system development, role conflict, role ambiguity, management's support, objectivity and competence, were set as independent variables whereas the internal audit quality was the dependent variable and seven research hypotheses were established respectively. The study results showed that the use of computer-assisted audit techniques, audit staff's involvement in system development, as well as audit staff's competence had positive influence on the quality of internal audit; the above results inspired this study significantly.

Wu (1991) believed that internal audit is particularly important for high-level management; it can help high-level management to find serious deficiencies which were deliberately neglected by other superintendents, and the internal auditors can substitute high-level management to communicate with organization members as well. In order to maintain a more independent, detached standing, the internal auditors should be responsible to high-level management, whereas high-level management also has to assess the effectiveness of the internal audit to see whether resources should be allocated to it. The study tried to explore what affecting factors should be considered in assessing the efficacy of internal audit in views of the opinions of high-level managers. Wu's study suggested concrete thinking directions for the design of network structured files and data access methods of accounts in accounting information systems for this study.

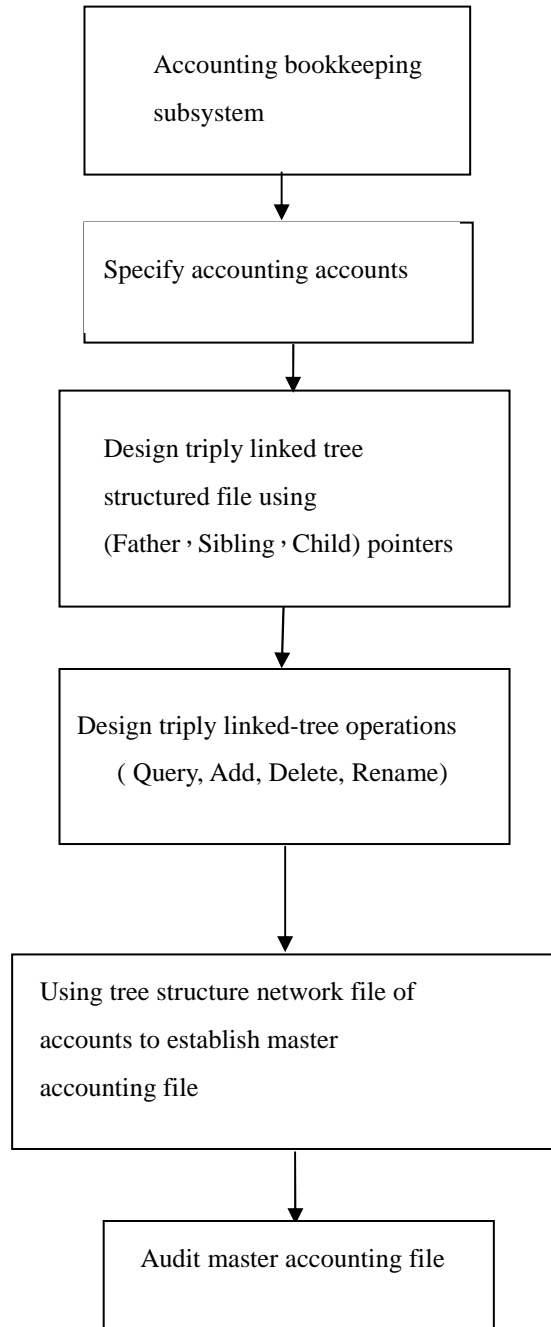
Huang (2001) explored computerization of accounting and internal control of small and medium-sized enterprises; he deemed that when business in the environment of manual operation has a sound internal control mechanism, computerization would not change the original operating procedures. The benefit of computerization is only to make processing more efficient. In contrast, when the business in the environment of manual operation does not have a sound internal control mechanism, computerization will cause the enterprises to standardize or rationalize the operation processes, in the meantime to establish or strengthen the internal control mechanism. The results of Huang's study helped to illuminate the direction of research for this study.

Chen, Q. (2002), having examined corporate governance in the United States, Germany, Japan, and Taiwan, realized that internal control is the cornerstone of corporate governance, and internal audit has to be effectively implemented. After having collected relevant data and analyzed/summarized such data, he came up with concrete proposals as the basis for the proper implementation of an internal audit in order to strengthen the function of corporate governance. The content of Q. Chen's has important influence in forming the purpose of this study.

Xu (2003) applied Hay's (1996) Data Model Patterns to find out the possible deficiencies of the REA accounting model: he then used these patterns to strengthen the REA accounting model (McCathy, 1982), so that the REA accounting model can more adequately reflect the business processes of the organization; but the research did not make in-depth study in designing internal audit mechanism for modernization of accounting information system. However, Xu's study formed the basis of this research and provided a deeper thought direction.

## RESEARCH METHODS

According to the above literature review, in this study, triply-linked tree algorithms are proposed to design network structured files and data access methods of accounts in an accounting information system and the design process is shown in Figure 1.



**Figure 1: Design Process of Triply-linked Tree**

### Applying Triply-Linked Tree to the Design of Accounting Accounts

In order to show the tree data structure relationship of level precedence between any accounting account and those it encompasses and is subordinated to, three data structures, namely, list, tree (with left child and right sibling pointers), and threaded tree (Horowitz & Sahni, 1993), can be employed in file design. In the application of the first two data structures, there exists the ability of downward reference; besides, the accounts at the same level can be easily accessed one by one with the ability of rightward reference. However, they lack the ability of upward reference. Threaded tree, while having the ability of upward reference, cannot directly provide the location of the father to any leaf node; thus, there is a need to design a third pointer (Father) that can refer upward to the father node.

There are two kinds of the most common organizations for tree-structured files: Index Sequential Access Method (ISAM) and Virtual Storage Access Method (VSAM); either organization can be employed in designing master file for accounting accounts. When querying the accounting master file or needing to find the data in encompassing lower level accounts from higher level accounts, we need only to know the account code of the encompassing account to be queried. Using stack and last-in-first-out (LIFO) operation and the pointers at the same account level and in the lower account level, data can be accessed one at a time; the processing sequence using this method is very clear and the end users can have good sense at a glance (Chen, H. 1998).

### The data structure

There are 5 items in a record of account file: department code, account code, credit/debit bit (D/C), addition or deduction (+/-), and account level. Account code is assigned according to classification numbering; it can be up to 10 bits and mainly consists of general ledger accounts (4 classes) and subsidiary ledger accounts. The usage account code within the account code is assigned general ledger account code to show their affiliations. For the first bit of account code, 1 means asset, 2 means liability, 3 means owner's equity, 4 means revenue, and 5 means expense. Level denotes the layer sequencing of the account. As an example, when data contents of the input account file are ordered by departments, accounting code, and level, it can be shown as in table 1.

**Table 1: Accounts of asset class**

| Department | Account Code | Credit/Debit | +/- | Level |
|------------|--------------|--------------|-----|-------|
| 01         | 1            | D            | +   | 1     |
|            | 11-12        | D            | +   | 2     |
|            | 110          | D            | +   | 3     |
|            | 1101         | D            | +   | 4     |
|            | 110190       | D            | +   | 5     |
|            | 1102         | D            | +   | 4     |
|            | 110210       | D            | +   | 5     |
|            | 110220       | D            | +   | 5     |
|            | 1105         | D            | +   | 4     |
|            | 110510       | D            | +   | 5     |
|            | 110520       | D            | +   | 5     |
|            | 110590       | D            | +   | 5     |

### Triply-Linked Tree Network Structured File of Accounting Accounts

The account file, having been audited to be error-free, is ranked according to department, account code, and level. In the beginning, account tables of level 1 to level 10 are established, the table can be regarded as one array and the processed accounts are linked together in such a way that every account record contains three fields of upper level link, same level link, and lower level link and has department code and account code as its primary key to construct network structured file of accounts; according to this structured file, accounting master file can be built using ISAM file structure, the record format of accounting account file is shown in Table 2.

**Table 2: Data Structure of Accounting Account File**

| Field Name | Data Type | Length | Description                |
|------------|-----------|--------|----------------------------|
| Dept       | Char      | 2      | Department Code            |
| Actno      | Char      | 10     | Account Code               |
| DC         | Char      | 1      | Debit/Credit               |
| AS         | Char      | 1      | Addition/Subtraction(+/-↓) |
| Level      | Char      | 1      | Account Level              |
| Father     | Char      | 10     | Upper level pointer        |
| Sibling    | Char      | 10     | Same level pointer         |
| Child      | Char      | 10     | Lower level pointer        |

The algorithm to build a triply-linked tree is as follows:

/\* T is Triply-Link Tree, Input: Acn, Deg, Output: Actno, Father, Sibling, Child

I is New input Account No Level, K is Old input Account No Level \*/

PROCEDURE BUILDT

DIM ACT (10), FAT (10), SBL (10), CHD (10);

INPUT ACN, DEG;

FAT (1) = '\*'; Q = ACN; K = DEG; ACT (DEG) = ACN;

REPEAT

INPUT ACN, DEG;

P = ACN; I = DEG;

CASE

I = K: CHD (K) = '\*'; SBL (K) = P;

ACTNO = ACT (K);

FATHER = FAT (K);

SIBLING = SBL (K);

CHILD = CHD (K);

WRITE T; ACT (I) = P;

FAT (I) = FAT (K); K = I;

I > K: CHD (K) = P;

ACT (I) = P; FAT (I) = Q;

K = I;

I < K: CHD (K) = '\*';

SBL (K) = '\*';

ACTNO = ACT (K);

FATHER = FAT (K);

WRITE T;

IF K-I > 1 THEN

FOR J = K-1

TO I+1 STEP -1

SBL (J) = '\*';

ACTNO = ACT (J);

FATHER = FAT (J);

SIBLING = SBL (J);

CHILD = CHD (J);

WRITE T;

NEXT

ELSE

CHD (I) = P;

```

ACTNO = ACT (I);
FATHER = FAT (I);
SIBLING = SBL (I);
CHILD = CHD (I);
WRITE T;
TEMP = FAT (I);
END IF
ACT (I) = P; FAT (I) = TEMP;
K = I;
END CASE
UNTIL ACN = '*'
END BUILDT

```

An example account file content of the network-structured file using the above algorithm to build a triply-linked tree is shown in Table 3.

**Table 3: Network structured file of asset account**

| Department code | Account code | D/C | +/- | level | Upper level | Same level | Lower level |
|-----------------|--------------|-----|-----|-------|-------------|------------|-------------|
| 01              | 1            | D   | +   | 1     | *           | *          | 11-12       |
|                 | 11-12        | D   | +   | 2     | 1           | *          | 110         |
|                 | 110          | D   | +   | 3     | 11-12       | *          | 1101        |
|                 | 1101         | D   | +   | 4     | 110         | 1102       | 110190      |
|                 | 110190       | D   | +   | 5     | 1101        | *          | *           |
|                 | 1102         | D   | +   | 4     | 110         | 1105       | 110210      |
|                 | 110210       | D   | +   | 5     | 1102        | 110220     | *           |
|                 | 110220       | D   | +   | 5     | 1102        | *          | *           |
|                 | 1105         | D   | +   | 4     | 110         | *          | 110510      |
|                 | 110510       | D   | +   | 5     | 1105        | 110520     | *           |
|                 | 110520       | D   | +   | 5     | 1105        | 110590     | *           |
|                 | 110590       | D   | +   | 5     | 1105        | *          | *           |

### Accounting Account Operation of Triply-Linked Tree and Design of File Auditing

#### 1. Query of Accounting Account files

Traversal of tree structure can be categorized into three processing types: depth-first, symmetric, and bottom-up. For big enterprises or public-owned institutions, the account codes mainly adopt classification numbering, which, when represented in tree structure, is suitable for depth-first traversal algorithm; As an example, when querying data with department code No. 20 (Pingchen Town Factory, Taoyuan Hsien, Taiwan), account code 514 (direct production expense), the data that can be accessed within reach of its branched network account can be exhaustively found using recursive depth-first-search algorithm (Horowitz, E. & Sahni, S., 1993) .

Querying algorithm using triply-linked tree is shown below.

```

/* T is Triply-Link Tree */
PROCEDURE INQUIRET (DEPT, ACTNO)
TKEY = DEPT + ACTNO; READ T;
PUT DEPT, ACTNO, FATHER, SIBLING, CHILD;
INQUIRET (DEPT, CHILD);
INQUIRET (DEPT, SIBLING);
END INQUIRET

```

As an example, a network-structured file as a result of querying direct production expense accounts using the above algorithm is shown in Table 4.

**Table 4: Network structured file output of querying direct production expense accounts**

| Department code | Account code | D/C | +/- | level | Upper level account | Same level account | Lower level account |
|-----------------|--------------|-----|-----|-------|---------------------|--------------------|---------------------|
| 20              | 514          | D   | +   | 3     | 51                  | 570                | 5142                |
|                 | 5142         | D   | +   | 4     | 514                 | 5143               | 514220              |
|                 | 514220       | D   | +   | 5     | 5142                | *                  | *                   |
|                 | 5143         | D   | +   | 4     | 514                 | *                  | 514320              |
|                 | 514320       | D   | +   | 4     | 5143                | *                  | *                   |
|                 | 570          | D   | +   | 3     | 57                  | *                  | 5702                |
|                 | 5702         | D   | +   | 4     | 570                 | 5703               | 570220              |
|                 | 570220       | D   | +   | 5     | 5702                | *                  | *                   |
|                 | 5703         | D   | +   | 4     | 570                 | 5704               | 570310              |
|                 | 570310       | D   | +   | 5     | 5703                | 570320             | *                   |
|                 | 570320       | D   | +   | 5     | 5703                | 570330             | *                   |
|                 | 570330       | D   | +   | 5     | 5703                | *                  | 57033010            |
|                 | 57033010     | D   | +   | 6     | 570330              | 57033020           | *                   |
|                 | 57033020     | D   | +   | 6     | 570330              | *                  | *                   |
|                 | 5704         | D   | +   | 4     | 570                 | 5705               | 570420              |
|                 | 570420       | D   | +   | 5     | 5704                | 570520             | *                   |
|                 | 5705         | D   | +   | 4     | 570                 | *                  | 570520              |
|                 | 570520       | D   |     | 5     | 5705                | *                  | *                   |

2. Adding accounting account

To add new accounts, the data structure of the input account file needs to be designed as

| Department | New Account code | Upper account | Modification Type (A) |
|------------|------------------|---------------|-----------------------|
|------------|------------------|---------------|-----------------------|

To add a new node with account code ACN, there are 4 main steps to consider: (1) If the tree is empty, the new node becomes the root. (2) If  $ACN = T_i$  (ACTNO), addition has failed, because the account already exists in the tree. (3) If the  $ACN > T_i$  (ACTNO), we should search the right sub-tree of  $T_i$  for the appropriate place for the new node and modify all relevant linking indexes. (4) If the  $ACN < T_i$  (ACTNO), we should search the left sub-tree of  $T_i$  for the appropriate place for the new node and modify all relevant linking indexes.

To exemplify, let the new accounts with codes 1103 and 110201 be added into the asset class accounts as in Table 1: their upper level account codes are 110 and 1102. By applying the addition algorithm, the following intermediate files can be obtained (Lee, Chang, Tseng & Tsai, 1999).

I. Partial content of original file before adding 1103 account:

| Department Code | Account code | D/C | +/- | Level | Upper Level Account | Same Level Account | Lower Level Account |
|-----------------|--------------|-----|-----|-------|---------------------|--------------------|---------------------|
| 01              | 1101         | D   | +   | 4     | 110                 | 1102               | 110190              |
|                 | 1102         | D   | +   | 4     | 110                 | 1105               | 110210              |

II. File contents change as a result of adding 1103 account:

| Department Code | Account code | D/C | +/- | Level | Upper Level Account | Same Level Account | Lower Level Account |
|-----------------|--------------|-----|-----|-------|---------------------|--------------------|---------------------|
| 01              | 1101         | D   | +   | 4     | 110                 | 1102               | 110190              |
|                 | 1102         | D   | +   | 4     | 110                 | 1103               | 110210              |
|                 | 1103         | D   | +   | 4     | 110                 | 1105               | *                   |

III. Partial content of original file before adding 110201 account:

| Department Code | Account code | D/C | +/- | Level | Upper Level Account | Same Level Account | Lower Level Account |
|-----------------|--------------|-----|-----|-------|---------------------|--------------------|---------------------|
| 01              | 1102         | D   | +   | 4     | 110                 | 1103               | 110210              |
|                 | 110210       | D   | +   | 5     | 1102                | 110220             | *                   |

IV. File content change as a result of adding 1103 account:

| Department Code | Account code | D/C | +/- | Level | Upper Level Account | Same Level Account | Lower Level Account |
|-----------------|--------------|-----|-----|-------|---------------------|--------------------|---------------------|
| 01              | 1102         | D   | +   | 4     | 110                 | 1103               | 110201              |
|                 | 110201       | D   | +   | 5     | 1102                | 110210             | *                   |

The algorithm using triply-linked tree to add new accounting account is as follows:

PROCEDURE ADDT (DEPT, ACN, NFATHER)

SW = 1;

REPEAT

CASE

NFATHER = '\*': TKEY = 1; READ T;

DO WHILE SIBLING < ACN OR SIBLING = '\*'

TKEY = DEPT + SIBLING; TEMP = ACTNO;

READ T;

LOOP

TKEY = DEPT + TEMP; READ T;

SIBLING = ACN; REWRITE T;

ACTNO = ACN; FATHER = '\*';

SIBLING = '\*'; CHILD = '\*'; WRITE T;

NFATHER <> '\*': TKEY = DEPT + NFATHER;

READ T; TKEY = DEPT + CHILD;

READ T;

IF ACN = ACTNO THEN INSERT

ERROR

DO

DO WHILE ACN > ACTNO

SW = 0; TKEY = DEPT + SIBLING;

TEMP = ACTNO; READ T;

LOOP

IF SW = 0 THEN

TKEY = DEPT + TEMP; READ T;

TEMP = SIBLING;

SIBLING = ACN; REWRITE T;

ACTNO = ACN;

FATHER = NFATHER;

SIBLING = TEMP;

CHILD = '\*'; WRITE T;

SW = 1; EXIT DO;

ELSE

EXIT DO;

END IF

LOOP

IF ACN < ACTNO THEN

TKEY = DEPT + NFATHER;

READ T; TEMP = CHILD;

CHILD = ACN; REWRITE T;

ACTNO = ACN;

FATHER = NFATHER;

SIBLING = TEMP;  
 CHILD = '\*'; WRITE T;  
 END IF  
 END CASE  
 UNTIL ACN = '\*'  
 END ADĐT

3. Deletion of accounts from accounting file

To delete an accounting account, the data structure of the input account file is designed as

|            |                  |  |                       |
|------------|------------------|--|-----------------------|
| Department | New Account code |  | Modification Type (D) |
|------------|------------------|--|-----------------------|

To delete a node with account code ACN, we can regard this node as a root and delete all traversed sub-tree from the node to its leaves and modify all relevant linking indexes. As an example, suppose the account with code 5703 is to be deleted from the Table 4 direct expense accounts; by applying the deletion algorithm, the following intermediate files can be obtained.

I. Partial content of original file before deleting account 5703:

| Department Code | Account code | D/C | +/- | Level | Upper Level Account | Same Level Account | Lower Level Account |
|-----------------|--------------|-----|-----|-------|---------------------|--------------------|---------------------|
| 20              | 570          | D   | +   | 3     | 57                  | *                  | 5702                |
|                 | 5702         | D   | +   | 4     | 570                 | 5703               | 570220              |
|                 | 570220       | D   | +   | 5     | 5702                | *                  | *                   |
|                 | 5703         | D   | +   | 4     | 570                 | 5704               | 570310              |
|                 | 570310       | D   | +   | 5     | 5703                | 570320             | *                   |
|                 | 570320       | D   | +   | 5     | 5703                | 570330             | *                   |
|                 | 570330       | D   | +   | 5     | 5703                | *                  | 57033010            |
|                 | 57033010     | D   | +   | 6     | 570330              | 57033020           | *                   |
|                 | 57033020     | D   | +   | 6     | 570330              | *                  | *                   |

II. After deleting account 5703, file content turns out to be:

| Department Code | Account code | D/C | +/- | Level | Upper Level Account | Same Level Account | Lower Level Account |   |
|-----------------|--------------|-----|-----|-------|---------------------|--------------------|---------------------|---|
| 20              | 570          | D   | +   | 3     | 57                  | *                  | 5702                |   |
|                 | 5702         | D   | +   | 4     | 570                 | 5704               | 570220              |   |
|                 | 570220       | D   | +   | 5     | 5702                | *                  | *                   |   |
|                 | 5703         | D   | +   | 4     | 570                 | 5704               | 570310              | D |
|                 | 570310       | D   | +   | 5     | 5703                | 570320             | *                   | D |
|                 | 570320       | D   | +   | 5     | 5703                | 570330             | *                   | D |
|                 | 570330       | D   | +   | 5     | 5703                | *                  | 57033010            | D |
|                 | 57033010     | D   | +   | 6     | 570330              | 57033020           | *                   | D |
|                 | 57033020     | D   | +   | 6     | 570330              | *                  | *                   | D |

III. Partial content of original file before deleting account 570420:

| Department Code | Account code | D/C | +/- | Level | Upper Level Account | Same Level Account | Lower Level Account |
|-----------------|--------------|-----|-----|-------|---------------------|--------------------|---------------------|
| 20              | 5704         | D   | +   | 4     | 570                 | 5705               | 570420              |
|                 | 570420       | D   | +   | 5     | 5704                | 570520             | *                   |

IV. File content changes as a result of deleting 570420 account:

| Department Code | Account code | D/C | +/- | Level | Upper Level Account | Same Level Account | Lower Level Account |   |
|-----------------|--------------|-----|-----|-------|---------------------|--------------------|---------------------|---|
| 20              | 5704         | D   | +   | 4     | 570                 | 5705               | *                   |   |
|                 | 570420       | D   | +   | 5     | 5704                | 570520             | *                   | D |

The algorithm using triply-linked tree to delete an accounting account is as follows:

PROCEDURE DELETET (DEPT, ACN)

REPEAT

TKEY = DEPT + ACN;

READ T; TEMP1 = SIBLING;

TEMP2 = FATHER;

CASE

FATHER = '\*':

DFSDLT (DEPT, ACTNO);

TKEY = 1; READ T;

WHILE ACN <> SIBLING

TKEY = DEPT + SIBLING;

READ T;

WEND

SIBLING = TEMP1;

REWRITE T;

FATHER NOT = '\*':

DFSDLT (DEPT, ACTNO);

TKEY = DEPT + TEMP2;

READ T;

IF ACN = CHILD THEN

CHILD = '\*'; REWRITE T;

ELSE

TKEY = DEPT + CHILD;

READ T;

DO WHILE ACN <> SIBLING

TKEY = DEPT + SIBLING;

READ T;

LOOP

SIBLING = TEMP1;

REWRITE T;

END IF

END CASE

UNTIL ACN = '\*'

END DELETET

PROCEDURE DFSDLT (SDEPT, SACN)

TKEY = SDEPT + SACN; READ T;

REMOVE T;

DFSDLT (SDEPT, CHILD);

DFSDLT (SDEPT, SIBLING);

END DFSDLT

#### 4. Renaming an Accounting Account

To rename an accounting account, the data structure of the input account file is designed as

| Department | Old Account code | New Account code |  | Modification Type (N) |
|------------|------------------|------------------|--|-----------------------|
|            |                  |                  |  |                       |

To rename an account ACN as a new node, there are 3 main steps to consider: (1) If  $ACN = Ti(AC\text{TNO})$ , renaming has failed, because the account already exists in the tree. (2) If  $ACN = Ti(CHILD)$ , indicating that the place of the account to be renamed has been found; all relevant linking indexes with the account need to be modified. (3) If the  $ACN \neq Ti(CHILD)$ , we should search the sub-tree of the same level to the right of  $Ti$  until the place of the old account has been identified, and modify all linking indexes related with this account. For example, assume the content of the accounts in original master file is as follows:

|    |         |   |   |   |         |         |         |
|----|---------|---|---|---|---------|---------|---------|
| 01 | 214-217 | C | + | 3 | 21-22   | 225-226 | 2145    |
|    | 2145    | C | + | 4 | 214-217 | 2147    | 214501  |
|    | 2147    | C | + | 4 | 214-217 | 2151    | 214710* |
|    | 2151    | C | + | 4 | 214-217 | 2162    | 215110  |
|    | 215110  | C | + | 5 | 2151    | 215120  | *       |
|    | 215120  | C | + | 5 | 2151    | *       | *       |

I. After renaming account 2151 to 2149, file content turns out to be:

| Department Code | Account code | D/C | +/- | Level | Upper Level Account | Same Level Account | Lower Level Account |
|-----------------|--------------|-----|-----|-------|---------------------|--------------------|---------------------|
| 01              | 214-217      | C   | +   | 3     | 21-22               | 225-226            | 2145                |
|                 | 2145         | C   | +   | 4     | 214-217             | 2147               | 214501              |
|                 | 2147         | C   | +   | 4     | 214-217             | 2149               | 214710*             |
|                 | 2149         | C   | +   | 4     | 214-217             | 2162               | 215110              |
|                 | 215110       | C   | +   | 5     | 2149                | 215120             | *                   |
|                 | 215120       | C   | +   | 5     | 2149                | *                  | *                   |

II. After renaming account 215110 to 214910, 215120 to 214920, file content turns out to be:

| Department Code | Account code | D/C | +/- | Level | Upper Level Account | Same Level Account | Lower Level Account |
|-----------------|--------------|-----|-----|-------|---------------------|--------------------|---------------------|
| 01              | 2149         | C   | +   | 4     | 214-217             | 2162               | 214910              |
|                 | 214910       | C   | +   | 5     | 2149                | 214920             | *                   |
|                 | 215120       | C   | +   | 5     | 2149                | *                  | *                   |

The algorithm using triply-linked tree to rename an accounting account is as follows:

PROCEDURE RENAMET (DEPT, ACT, ACN)

DO

REPEAT

TKEY = DEPT + ACN;

READ T NOT INVALID GOTO L1

DISPLAY "RENAME ERROR";

EXIT DO;

L1: TKEY = DEPT + ACT; READ T;

TFATHER = FATHER;

TSIBLING = SIBLING;

TCHILD = CHILD;

TKEY = DEPT + FATHER; READ T;

IF ACT = CHILD THEN

CHILD = ACN; REWRITE T;

ACTNO = ACN; FATHER = TFATHER;

SIBLING = TSIBLING;

CHILD = TCHILD;

WRITE T;

ELSE

```

TKEY = DEPT + CHILD; READ T;
WHILE ACT <> SIBLING
    TKEY = DEPT + SIBLING;
    READ T;
WEND
SIBLING = ACN; REWRITE T;
ACTNO = ACN;
FATHER = TFATHER;
SIBLING = TSIBLING;
CHILD = TCHILD; WRITE T;
END IF
UNTIL ACT = '*'
LOOP
END RENAMET

```

### **Auditing Master Accounting File**

In big private enterprises or public-owned institutions, each voucher transcribed by the accountants of various departments is audited by management of different levels and certified by the chiefs; it then can be inputted into the computer for editing. The items to be audited are as follows: (1) Code check: including checking on department codes, certification number, and account codes. (2) Checking on voucher number to see if it is numbered continuously and by date order. (3) Account code should be the lowest level subsidiary ledger or general ledger account, otherwise it is not processed (4) Subsidiary ledger should be consistent with general ledger. (5) Amount in subsidiary ledger should be consistent with that in general ledger. (6) Check for balancing debit and credit. (7) Check for total amount. (8) The beginning and ending numbers of various vouchers should be consistent with those recorded in delivery lists (Wu, J. 2005).

Utilizing the error-free modification data file, one can modify the master accounting file, post, adjust spreadsheet, and close account; moreover, before making the income statements or balance sheets, etc., one can check on the master accounting file for any abnormal account balance (e.g. credit balance in asset accounts, debit balance in debt accounts, or deficit occurred in expense accounts, etc.), on consistence of accounts in each level (e.g. balance in first level accounts should be equal to the sum of balances in encompassing second level accounts, etc. to the lowest level accounts), for purposes of internal control (Wu, J. 1997).

The auditing of triply-linked tree network file can use level-by-level method to check on the master accounting file: this method is classified as level sequencing auditing and is suitable for depth-first-search or breadth-first-search processing (Cormen, Leiserson & Rivest, 1992). In this study depth-first-search algorithm is adopted: first find the very node to be audited (e.g. abnormal account balance, inconsistent account balances) and treat this node as a starting position (i.e. root); along the search path from the root to the leaves (the last level accounts), read out the contents of each node when traversing, and continue to look for all the branch nodes until all nodes have been traversed. Reading account balances from the bottom up, aggregated sum of the balances of the last level should be equal to those of corresponding accounts in the previous level; and the check can be recursively applied until the very node to be audited is checked.

The algorithm using triply-linked tree to audit an account is as follows:

```

PROCEDURE AUDITT (DEPT, ACN)
TKEY = DEPT + ACN; READ T;
PUT DEPT, ACTNO, FATHER, SIBLING, CHILD;
AUDITT (DEPT, CHILD);
AUDITT (DEPT, SIBLING);
END AUDITT

```

## CONCLUSION

To summarize, the triply-linked tree algorithm proposed in this study can provide any leaf node account with its parent node position in the tree data structure of accounting accounts; for this reason, there is a need to design a third link that has upward referencing ability, hence the triply-linked tree structure is formed. Each node in this tree has Father, Sibling, and Child three pointers, designed in this way, any account has the ability of upward, rightward, and downward references; therefore the accounting accounts can be linked in a network structure style, rendering the account data in each level correct and complete.

Especially for the account management of an accounting information system, the triply-linked tree network structure and the associated data access method are applied in processing accounting accounts and are explored in depth in this study. For example, "How to establish level sequencing network structure for accounting accounts?", "Modification of tree indexes for account operations (Addition, Deletion, and Renaming)", "How to keep correctness of the indexes?", and "In the process of internally auditing the file, how to check if the balances in each account level are consistent?" have been thoroughly explained. The practical value of this tree structure design and the associated data access method is also exemplified with concrete examples.

## REFERENCES

- Chen, H.C., (October 1998), "Construction and Recognition of Effective Accounting Information System (I)", *Accounting Research Monthly*, Vol. 155.
- Chen, H.C., (November 1998), "Construction and Recognition of Effective Accounting Information System (II)", *Accounting Research Monthly*, Vol. 156.
- Chen, Q. C., (2002 ), "A Study on Corporate Governance and Internal Audit in Taiwan", Master's Thesis, Institute of Accounting, Chung Yuan Christian University.
- Chen, Z. Z., (1994), "A Study of influential factors on the quality of internal audit of Taiwan's enterprises", Master's Thesis, Institute of Accounting, National Cheng Kung University.
- Cormen, T.H., Leiserson C.E., and Rivest R.L.,(1992), "Introduction to Algorithms", MIT Press, pp.469-470, pp.477-479
- Hay, David C., (1996), "Data Model Patterns: Conventions of Thought", Dorset House Publishing Company.
- Horowitz, E. & Sahni, S., (1993), "Fundamentals of Data Structures", Silicon Pr., pp.189-191, pp.201-202, pp.213-215.
- Huang, Z. C.,(2001), "A Study of Computerization and Internal Control of Small and Medium-Sized Enterprises" Master's Thesis, Institute of Accounting, Tamkang University.
- Lee, R.C.T., Chang R.C., Tseng S.S., Tsai Y.T.,(1999), "Introduction to the Design and Analysis of Algorithms", McGraw-Hill Education.
- Lin, H. M. ,(1996), "Data Structures and File Processing", Song-gang Bookstore, pp. 506-517.
- Luan, B., and Luo K. Y., (2006), " Electronic Commerce", Chan-Hai Bookstore, pp. 224-225.
- McCarthy W.E., (1982)," The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment", *The Accounting Review*, Vol. 57, No.3, pp.554-578.
- Wu, J. F., *Accounting Information Systems and Computer Audit*, Ji-Sen Culture Co., Ltd., 1997, page 310-315,332-333,631-632.
- Wu, J.F., (2005), " The Financial and Accounting Information Systems", Culture Co., Ltd..
- Wu, Y. S.,(1992), "A Study of Assessing Factors on the Effectiveness of Internal Audit", Master's Thesis, Institute of Management Sciences, National Chiao-Tung University.
- Xu, R. N,( 2003), "Using Data Model Template to Strengthen the Study of REA Accounting Model", Master's Thesis, Department of Accounting Information, Da-Yeh University.
- Yuan, F.C., (2004), *ERP Accounting Information Systems and Practice*, Chi-Fong Bookstore.