IAC Accounting Data Model: Eliminating Redundancies for a Consistent Accounting System

Ricardo Salim
Cautus Networks Co.
rsalim@cautus.net

Carlos Ferran
Penn State Great Valley
cferran@psu.edu

ABSTRACT

Accounting is the heart of all Enterprise Resource Planning Systems (ERP). While this metaphor reminds us that accounting is the center all business information; it is also misleading. The heart is an independent organ that pumps blood to the whole body; on the other hand, accounting is (1) totally dependent on the subsystems –auxiliary ledgers- and (2) the data it process comes from those subsystems. This paper shows how accounting is a summarization that replicates the data contained in the other subsystems; a duplication completely necessary in a manual information system but totally redundant in a computerized one. Thereupon it demonstrates how part of the subsystems' data is also redundant in itself. It concludes with an accounting data model (IAC) that provides a better data structure for a computerized accounting information system.

Keywords

Accounting, accounting information systems, enterprise resource planning systems, ERP, accounting model, data model, database management systems.

INTRODUCTION

The first attempts made to computerize accounting did not question the traditional data structure of this discipline. In fact, the first accounting databases kept the model described by Pacioli during the renaissance (Lee, Bishop, & Parker, 1996; Macve, 1996) almost intact. That could have been because a large part of that structure was theoretically grounded in concepts that are still valid today. On the other hand, another large part of that model was based in operational requirements that have lost validity with the advent of computerized accounting systems. However, even the most well-known and up-to-date Accounting Information Systems (AIS) and Enterprise Resource Planning Systems (ERP) like SAP, ORACLE Financials, and PeopleSoft still keep some of them; the ones that are less obvious and more complex to solve and explain.

This paper highlights the parts of the traditional model that are no longer necessary and proposes a radically new accounting database model that eliminates them. Maintaining the status-quo is the easiest, safest, and most conservative position; and conservatism is highly valued in the accounting profession. Therefore, a change to an established model used for over half a millennium is not going to be easily accepted by the community unless it is very clear and quite useful. A prior attempt was made by McCarthy (1982) with his REA model. While his model is widely accepted as accurate and interesting we found no practical implementations and while the major elements of the model are found in most ERPs they differ semantically in several critical areas because the model is too general (Weber, 1986).

We will start our presentation with a very simple example that has been already corrected by the majority of the modern ERPs and AISs. Then, we will move towards more complex problems that are still present in most implementations.

The traditional model includes the use of Balance books where the entries are summarizations of entries contained in other books (or ledgers). These Balance books are meant to avoid the need to recalculate the totals every time they are required as long as the primary data has not changed. An example of these Balance books is the Account Trial Balance.

Figure 1 shows a General Ledger and an Account Trial Balance. The Account Trial Balance is calculated by totaling all the transactions for each account. This calculation is quite fast and easy with only five transactions but the General Ledger usually has many more accounts and transactions making this process quite long and making the Account Trial Balance a necessity for practical purposes. Therefore, for practical operational purposes, the traditional accounting data structure includes not only the General Ledger but also the Account Trial Balance. The Account Trial Balance is updated every time the General Ledger is updated.

General Ledger

Date	Account	Number	Amount
10-10-01	A	A-001	15.000
10-10-01	B	B-001	14.000
10-10-01	B	B-001	2.000
10-10-01	C	C-001	1.000
10-10-01	C	C-002	4.000
		Total	36.000

Account Trial Balance

Account	Amount
A	15.000
B	16.000
C	5.000
Total	36.000

Figure 1

Conversely, computers make calculations and recalculation quick and easy; in fact, that is one of their main advantages (Oz, 2004). Therefore, the Account Trial Balance is an unnecessary artifact in computerized systems. Furthermore, the Account Trial Balance is not only unnecessary but its redundancy becomes a potential problem instead of an aid. It is not uncommon to find inconsistency between the primary information (General Ledger in this case) and the summarizations (Account Trial Balance). Every time a transaction is added, deleted, or modified the corresponding account total needs to be updated. These updates occur in different books and potentially at different times allowing for potential involuntary errors. In a typical accounting the General Ledger is under constant update. Each update requires an update of the Account Trial Balance; nonetheless, this could be so burdensome that the system (or accountant) may opt to execute many updates in the General Journal before updating the Account Trial Balance. However, the more the Account Trial Balance update is postponed, the higher the possibility of making that update incorrectly. In conclusion, the Account Trial Balance should a separate physical table in computerized systems but totals should be recalculated every time they are needed.

The Account Trial Balance is a part of the traditional accounting data structure that modern computerized systems have generally eliminated. Its redundancy and the potential problems it brings to accounting are so evident that both IS professionals and accountants have agreed in eliminating them. But this is not the only redundancy present and a more methodic examination of the traditional accounting data structure is needed. Below we use database theory to examine the generally accepted accounting data structure.

REDUNDANCIES

Relational database theory states that we need to eliminate redundancy whenever possible (Codd, 1970). Normalization is the process that takes a given data structure and eliminates all of its redundancies and hidden dependencies (Chen, 1976; Codd, 1971). A basic tenet of normalization is that while the final data structure may be different we are always able to recombine it in a manner that the original (and maybe redundant) representation of the data may be restored (Date, 2000).

Relational database technology includes a facility known as views. A database view is an alternate mode of presenting the data (or a subset of the data) in a format that may be different to that in which it is physically stored (Date, 2000). Views can be used to create static (screen or printed) reports but -under certain conditions- they can also be presented on the screen as dynamic reports where the user can make changes and the system updates the underlying data accordingly. The original data structures are still accessible through views.

A caveat to normalization is that the views required to present some of the original data structures may require a long time and a larger amount of computer power (Codd, 1970). Therefore, it is a generally accepted practice that some systems may use data structures that are not completely normalized (Date, 2000). System analysts have to choose between to evils: slow response time versus redundancies that may cause inconsistencies. Initially this problem was thought as deciding between using additional processing resources versus storage resources and since the cost of storage has dropped considerably most analyst opted towards implementing redundancies that would speed up the processes. However, time has shown that the main problem with redundancies is not the need for additional storage resources but the data inconsistencies that appear through time. Inconsistencies may occur due to bugs, systems failure, procedural failures, or user commissions and omissions. Even in the most robust systems, users find that inconsistencies occur as they use the system. The more we use a system, the more inconsistencies we will find (Salim Koussa & Ferran-Urdaneta, 2001). Most systems have maintenance modules that check inconsistencies and try to correct them; however, some inconsistencies cannot be repaired automatically since the system cannot know which of the inconsistent values is the correct one.

The prior accounting example is a typical case of redundancy. The totals and subtotals contained in the Account Trial Balance can be calculated by adding the appropriate transactions contained in the General Ledger. Individual transactions and their corresponding totals are redundant; totals should not be stored. Normalizing the traditional accounting data structure eliminates the Account Trial Balance from the stored data (physically) but it still can be recreated. The first computerized AIS kept this redundancy present in the manual systems; however, current systems are implemented using more powerful hardware and software developers have eliminated it. Even though the underlying data structure has been altered, the user still sees all the information the traditional way. It is through views that the accounting user can still *see* the traditional data structure even though the data structure has been partially normalized and therefore it is stored in a different format.

Redundancies in Integrated Management and Accounting Systems

Redundancies in the traditional accounting data structure are not just exceptions. In fact, almost the entire accounting system is a pyramid made up of different levels of aggregations. They are the aggregation of the auxiliary ledgers that register the basic business transactions; like the inventory journal and the cash journal. These journals are also known as the administration journals. Therefore, we can say that the traditional accounting data structure is a redundancy of the administration journals.

The Concept of Double-Entry Is Not a Redundancy

While we just argued the pervasive existence of redundancies in the traditional accounting data structure, we will also argue that what may seem to be a redundancy, the traditional double-entry method, is not such. Double-entry -the most basic accounting principle- seems to be a redundancy even from its own name (because of the word double); however, it is not. Double-entry means that we need to simultaneously register the reception or delivery of a product or service with the corresponding and counterbalancing delivery or reception of its value in cash or credit. There is no redundancy here; they are different but complementing elements.

Cox Benita (2003) recently wrote an article titled “Accountability lost: the rise and fall of double entry”. While the title would seem to forecast the demise of the double-entry method, the article in itself does not hold that hypothesis. The article presents the current problems of the *datalogic* accounting methods and argues that *infologic* methods are the way of future accounting; however, there are solutions to those problems that do not involve the elimination of the double-entry method. And the double-entry method is critical to assure accountability. Furthermore, double-entry does not hamper information usefulness which is Benita’s main concern.

The information that a product has been exchanged cannot be obtained from the information that there has been a cash exchange between those parties. Without double-entry we could record the funds received for merchandise sold without making any reference to the merchandise in itself and vice versa. In fact, that is what generally occurs in the two separate administration journals (inventory journal and cash journal). It is the method of double-entry that forces us to relate the two records so that we keep a balance between the products delivered, the cash received, and the gain or losses that occur in each exchange. And we are forced to do it because we could not obtain one from the other. What could be a redundancy is the physical storage of both transactions as separate records (McCarthy, 1979).

The Physical Storage of the Separate Double-Entry Is a Redundancy

The exchange of assets between people or organizations (or even between parts of the same organization) requires the use of documents such as receipts and reception acknowledgments. Otherwise one of the parties could always argue that they never received the assets even though they did. These documents become proof of the exchange and they are the primary source of data for both management and accounting. These documents describe the asset, identify the parties, establish an agreed value, and set the date of the exchange. Accounting extracts the amounts and generates the transactions to the corresponding accounts while management uses them to manage the appropriate function. For example, with the customer’s reception acknowledgment of a given merchandise, accounting deducts its cost from the inventory account, increases accounts receivable for that customers, and registers the difference between the two in the Capital Gains (or Losses) account. With the same document, management starts the accounts receivable process for that customer and the purchasing or production process for restocking the merchandise sold.

Proper business administration requires the use of several detailed journals to control each aspect of the process. For example, the inventory journal keeps track of purchases, sales, and transfers for each item separately. The accounts receivable journal keeps track of all receivables owned by the company and the corresponding payments made by each customer without describing how the receivables were generated or where the funds credited were applied. The banking journal lists the cash flows to each bank account but it does not fully explain the origin or application of the funds. On the other hand, accounting registers all of the transactions mentioned earlier (in a detailed or summary form at the accountant’s choice) but making sure to explain where incoming funds are placed or where outgoing funds come from. In other words, that the sum of all the credits is equal to the sum of all the debits. This way accounting assures the proper interrelationship between auxiliary journals which is what is called double-entry bookkeeping. From this point of view, accounting could seem to be indispensable and related but separate from the auxiliary journals. However, this duplicity is a necessary artifact only for keeping control in a system manually operated.

It is very difficult for an individual to assure that all that is registered in one journal is properly counter registered in another journal since all of these journals are physically separate (and probably managed by different people). This is even more difficult when more than two journals are involved (for example a sale that is partly paid in cash and partly paid in credit affects the bank, account receivable, inventory, and sales journals). Registering and duplicating all of these transactions in a single journal (or general journal) brings them together in a way in which the accountant can make sure that the debits are equivalent to the credits. However, this physical duplication that brings them together is unnecessary in a computerized system. Computerized database systems easily allow the presentation of the general journal as a "view" of the auxiliary journals making the physical storage of the general journal an unnecessary redundancy.

The redundancy between the accounting and the administration information is only one of several redundancies present in the data structure of current AISs and ERPs. These modern systems also hold similar redundancies between the journals of different sub-modules. Following parts will discuss the two most common *inter auxiliary journal redundancies*.

Redundancy between Invoice Header and Customer/Supplier Statement of Account

This redundancy generally occurs in modular designs where the invoicing module (sales) stores the data separately from that of the customer or the supplier module (accounts receivable or accounts payable respectively). To explain this redundancy lets use the invoice shown in figure 2.

James Smith (C-001) Invoice # 001
206 Main St. Date: 01-25-04
Philadelphia, PA 19756

Description	Amount (US$)
Product "P"	15,000.00
2 years Maintenance agreement	1,500.00
Subtotal	16,500.00
Tax	1,650.00
Grand total	18,150.00

Figure 2

The invoicing module data structure would be made up of two tables: invoice header and invoice detail. Invoice detail would include the invoice number (to establish the relationship with the invoice header), the item description (or maybe an ID or SKU that relates it to yet another table) and the amount (additional information like quantity, unit price, and discount could also be included but we omit them for sake of simplicity). The invoice header includes the invoice number, date, customer ID, and a few others. It could also include subtotal, tax, and grand total although these are an added redundancy. In addition, the accounts receivable module has a table with the document number, date, type (invoice or credit for example), the customer ID, the invoice grand total, a description, and a few others. Figure 3 shows the data structure using table notation.

InvoiceHeader(<u>InvNum</u>, Date, CustID, …)
InvoiceDetail(<u>DetailID</u>, InvNum, ItemDesc, Amount, …)
Receivable(<u>DocNum</u>, Date, Type, CustID, Descript, Amount, …)
BankDetail(AccNum, Date, Type, <u>TransID</u>, CustID, Descrip, Amount, …)

Figure 3

The redundancy is obvious. Most (if not all) of the Invoice Header information is also in the Receivable table. Furthermore, there is a one to one relationship between these two tables.

Redundancy between Accounts Receivable / Accounts Payable and Cash Ledger

This redundancy tends to occur in modular designs where the customer or the supplier module (accounts receivable or accounts payable respectively) stores the data separately from that of the Cash and Bank module. To explain this redundancy let us use the invoice shown in figure 4.

James Smith (C-001)
206 Main St.
Philadelphia, PA 19756

Invoice # 001
Date: 01-25-04

Description	Amount (US$)
Product "P"	15,000.00
2 years Maintenance agreement	1,500.00
Subtotal	16,500.00
Tax	1,650.00
Grand total	18,150.00
Cash	14,000.00
Amount due	4,150.00

Figure 4

In this case the receivables table would have a second record that corresponds to the US$14,000.00 paid in cash and the bank transaction table would also be used. Figure 5 shows the data in the receivables table as well as the data in the bank transaction table.

Receivables

CustID	Date	Type	DocNum	Description	Amount
C-001	01-25-04	Invoice	F-001	Invoice F-001	18,150.00
C-001	01-25-04	Cash	Cb-001	Partial payment F-001	-14,000.00
				Total	4,150.00

Bank Journal

AccNum	Date	Type	TransID	CustID	Description	Amount
001	01-25-04	Deposit	0102991	C-001	Partial payment F-001	14,000.00
					Total	14,000.00

Figure 5

There is clearly a redundancy between the date and amount in DocNum Cb-001 in the Receivables table and the date and amount in TransID 0102991 of the Bank journal table.

THE IAC MODEL

This section describes a model with an alternate accounting data structure that radically eliminates the most pervasive and potentially inconsistent redundancies of the traditional administrative-accounting model. The model is called IAC and makes up the data structure of the SecureAccounting system (both property of Cautus Network Corporation, Miami, FL.). The system (and in consequence the underlying data structure –IAC-) has been in place and under intensive use in over fifty companies in the past two years. The name IAC comes from the initials of Items, Agents, and Cash.

Items, Agents, and Cash

The traditional accounting model contains five types of entities. They are Assets, Liabilities, Capital, Revenues, and Expenses. The three main entities of IAC, the model we propose, are Items, Agents, and Cash. They reclassify and consolidate all of the entities of the traditional model. The IAC classification is a more normalized data structure that eliminates many of the redundancies present in the traditional model.

The traditional model subdivides the main accounts (entities) in sub-accounts. Many of the sub-accounts are kept in separate books due to their different nature and structure. Additionally, some sub-accounts that have very similar structure are stored separately because they originate from different main accounts. For example, Accounts receivable is an Asset and Accounts payable is a Liability; they are kept separate although their structure is almost the same. There probably is a third group of sub-accounts called Employees (used for payroll purposes at a minimum), which is also kept separately. The IAC model consolidates customers, suppliers, employees, and stockholders into one entity: Agents. In general, Agents are all people and institutions that exchange cash, goods, and/or services with the company. They all have many attributes in common (like name, address, and balance) but what is even more important, they all have the same type of database relationships with the other two main entities (Items and Cash).

The entity Item consolidates fixed assets, inventory, and all the goods and services that the company buys or sells including shares (stock). They are generally kept in separate books although they have many attributes in common (e.g. name, description, measurement unit, quantity, cost, and price). Furthermore, they all maintain the same type of database relationships with the other two main entities (Agents and Cash).

Finally the entity Cash consolidates bank accounts, petty cash, and any other account that reflects the flow of money.

Administrative and accounting systems register the economic activity of a company. In general, this economic activity is made up of the exchange of goods or services for money (or in certain cases for other goods or services). The Agent entity responds to the question of "Who" (surrenders or receives the goods or services); Item responds to the question of "What" (goods or services); and Cash to the "How much" (money it is given or received in exchange for the goods or services). This way the three IAC entities cover all the entities involved in economic activity and therefore consolidates all of the accounts of an administrative or accounting system.

However, the consolidation of all sub-accounts into three main entities is a necessary but not sufficient condition to eliminate the redundancies present in the traditional model. Further work is needed to eradicate them.

The following three sections mirror (but in reverse order) the last three sections of the previous section. They explain how IAC eliminates each of those redundancies.

Eliminating the Accounts Receivable / Accounts Payable – Cash Ledger Redundancy

Previously we showed -in Figure 5- how the date and amount in DocNum Cb-001 in the Receivables table and the date and amount in TransID 0102991 of the Bank journal table were redundant (as well as some other attributes). To solve this redundancy it would be enough to add the non-common attributes (like bank account, type, number, and transactionID) of the Bank Journal table to the accounts receivable and eliminate the whole Bank Journal table.

Now then, we have created a single table (or entity) called AgentTransactions which is the fusion of account receivable transactions and the bank journal table. Therefore, it replaces in the IAC model the two above mentioned tables of the traditional model. Figure 6 shows the new table and the data in it based in our last example.

CustID	Date	Type	DocNum	Description	AccNum	Type	TransID	Amount
C-001	01-25-04	Invoice	F-001	Invoice F-001				18,150.00
C-001	01-25-04	Cash	Cb-001	Partial payment F-001	001	Deposit	0102991	-14,000.00
					Total Accounts Receivable			4,150.00
					Total Cash			14,000.00

Figure 6

Total accounts receivable is calculated by adding up all of the transaction amounts. Total cash is calculated reversing the sign on the amount of all records for which AccNum is not null and adding them together.

Note that this radical solution entitles the elimination of the entire physical table for Bank (and Petty Cash) Journal and to calculate it and present it based on views of the Agent transaction table. Therefore, while Cash is one of the three main entities of the IAC model it exists only in a logical form; not as a physical table.

Eliminating the Invoice Header – Customer/Supplier Statement of Account Redundancy

In a previous section we described two redundancies: (1) the invoice total can be calculated by adding the corresponding details; however, systems often physically record the total in the invoice header too, (2) most of the data included in the invoice header is also found in the corresponding account receivable transaction.

To solve the redundancy between the invoice header and the receivables detail we simply eliminate the invoice header (or purchase order header table) from the physical database. We selected to eliminate the invoice header and not the receivables transaction since all invoice headers will be reflected in the receivables transactions while not all the receivable transactions may be reflected in the invoice headers (all invoices generate a receivable –even a "dummy" or "hidden" receivable with an amount of cero if fully paid in cash- but not all the receivables are generated from invoices).

In the IAC data structure the invoice information is stored partly under Agents and partly under Items. Under Agents we have the invoice header information and under Item we have the invoice detail information. The header describes the party involved, the date, and the conditions. The detail describes the what, the how many, and the how much is being transacted.

The conventional solution to solving the redundancy of physically storing a total is to calculate it every time is needed. However, we (like most major software developers) have chosen -for the sake of operating speed- to keep the redundancy and store the total, at least for the initial practical implementation of the IAC model (SecureAccounting). Showing the total of an invoice is an option very frequently used on any administrative/accounting system; therefore, calculating it every time instead of reading the result from a physical location substantially increases response time. This exception is a process of denormalization based on operating requirements.

Yet another reason not to eliminate this redundancy is that the receivable detail table includes records that come from invoices but also records that are later payments to the account. When the record comes directly from an invoice then the amount recorded is the sum of the items in the invoice (clearly a redundancy); but when it comes from later payments it cannot be calculated, it has a meaning of its own, and it is not a redundancy. We could still place a null value for the former and a number value for the latter to avoid the redundancy but this would make the summation of records in this table a very complex (and slow) process that would not be able to make use of the standard (and quick) summation features found in current database management systems.

Eliminating the Redundancy between the Accounting and the Subsidiary Ledgers

As discussed earlier the accounting ledgers are meant for assuring the proper balance of the double-entry method. They are not needed for making the double-entry, only to assure that the double-entry was made properly. For example, in the case of a sale, the accounting ledger physically brings together the total charged to a customer with the total subtracted from inventory and the difference applied to gain or losses. While these three are already stored in separate subsidiary ledgers, the traditional model -as well as most current ERP systems- still stores them in a separate set of accounting ledgers -or tables-.

Under the IAC model the accounting ledgers are virtual ledgers. The double-entry is assured by bringing the records together in a logical -not a physical- manner. The records are stored as Item, Cash, and Agents -as the primary data- and the general journal entry is a view extracted from them. In this way, IAC radically eliminates the redundancy between the accounting and the subsidiary ledgers.

Even though all accounting entries should come from the subsidiary ledgers, many accountants still either modify the entry directly in the accounting ledger and not in the subsidiary ledger, or create it in accounting without registering them in the subsidiary ledger. A system that would not allow them to work directly with the accounting ledger would not be welcome by the accounting community. However, one that would allow them to do so and still maintain the consistency with the subsidiary ledgers would be of a great help.

The IAC model allows direct manipulation of general journal entries without loosing consistency with the subsidiary ledgers. This can be easily done with simple and consistent DBMS procedures because all the information is stored in two physical tables (Agent and Item) and one logical table (Cash). In turn this also allows for very fast response time when generating or regenerating views.

Converting the accounting ledgers into simple views provides increased data integrity while not slowing down reports considerably. Generating a balance sheet or an income and losses statement is not noticeably faster under the traditional model than under the IAC model. A test of the integrity between subsidiary ledgers and the accounting ledger is not necessary under the IAC model because they are one and the same data. Furthermore, a fundamental integrity test (ex. check if the fundamental accounting equation holds) under the IAC model is not just about accounting but a comprehensive check of the entire data (accounting and subsidiary ledgers). And they may be done so rapidly that they can be included at the end of each database update without noticeable performance degradation.

The Fundamental Accounting Equation under IAC

Assets are the resources that a company owns. When a company is started, the only assets it holds are those that the stockholders provide as their initial investment in exchange for shares. The company may also acquire additional assets by incurring in debt. Therefore, total assets will be equal to the sum of the assets provided by the stockholders and the sum of the assets provided by the creditors. This is what is called the fundamental accounting equation and it is generally written as follows:

$$\text{Assets} = \text{Liabilities} + \text{Capital}$$

The fundamental accounting equation must be held true at all times. Each side of the equation may be calculated separately for comparison and thus checking that the equation is maintained. Therefore if we add all the assets we will get the same number than if we add together all of the debt provided by creditors and the stockholder's equity.

As the accounting cycle advances, two more entities are used: Revenues and Expenses. However, at any time we may add all the Revenues and subtract all the Expenses and we would get the operating result which is part of the Capital. Summarizing: the traditional accounting model contains five types of entities. They are Assets, Liabilities, Capital, Revenues, and Expenses. Revenues and Expenses are operating entities that show the results of the company's activity during a given period; they are cleared at the end of the period and the difference between them is added (or subtracted) from the Capital account.

The traditional accounting model classifies information based on ownership (assets are owned by the company, liabilities are owned by creditors, and capital is owned by stockholders). On the other hand the IAC accounting model classifies information based on data structure similarities which not only facilitates the use of information technology to process the data but also helps in eliminating redundancies. Moreover, although the classification is different the model still allows easy testing of the fundamental accounting equation; in fact, it makes the process much faster and easier.

We can apply the logic of the fundamental equation in different terms. For example, a company acquires assets in exchange for cash or debt from stockholders or third parties. It can also sell goods (or services) in exchange for cash or loans given to third parties. Therefore, following a similar methodology to the one applied earlier to obtain the fundamental accounting equation we get that:

$$\text{Account of Goods (Items)} = \text{Accounts of People (Agents)} + \text{Accounts of Money (Cash)}$$

where each side of the equality may be calculated independently of the other as required by the double-entry bookkeeping method.

The process required under the IAC model to verify the fundamental equation is very basic and simple making it even more reliable than the traditional one. All we have to verify is that:

$$\sum \text{Items(Amount)} = \sum \text{Agents(Amount)} + \sum \text{Cash(Amount)}$$

and since all of the data is contained in only two tables very little programming is required.

To guarantee that the fundamental equation holds for the whole, we need to assure that it holds for each part. In other words, every time an economic event (i.e. purchase, sale, etc) is registered, the set of records entered need to comply with the equation. However, this is quite easy and intuitive. Every good or amount of cash received or surrendered comes or goes to a person or institution (even if it is to the same company being administered or to a part or department of the same). We can establish that the amount of every record in Agents will be the same to the related records in Cash and Items. In more technical terms, let AgentTransID be the attribute in Cash and/or Items that contains the ID of the related transaction in Agents (in other words, AgentTransID is the foreign key in Cash and Items that relates the records to Agents whose primary key is ID). We then have that:

$$\sum \text{Agent(ID, Amount)} = \sum \text{Item(AgentTransID=ID, Amount)} + \sum \text{Cash(AgentTransID=ID, Amount)}$$

CONCLUSIONS

We have shown that both the traditional model and the IAC model are equally effective in controlling the required balance in bookkeeping (verifying the fundamental accounting equation); however, we also showed how the IAC model is easier to implement in computerized information systems. The IAC model eliminates most of the redundancies present in the traditional classification therefore increasing data integrity. Furthermore, we can use the IAC model for physical storage and still use the traditional data structure for data presentation and manipulation at the user level. Finally, the IAC model is not a theoretical model looking for an application but a theoretically grounded model that has been tested and is implemented in accounting software that has been in use in over fifty companies for the past two years.

ACKNOWLEDGMENTS

We thank Cautus Network Corporation of Miami, FL for their permission to publish this paper on its radically new accounting data model.

REFERENCES

Benita, C. (2003). Accountability lost: The rise and fall of double entry. *Omega The International Journal of Management Science, 31*((2003)), 303 –310.
Chen, P. P.-S. (1976). The Entity-Relationship Model - Toward a Unified View of Data. *ACM Transactions on Database Systems, 1*(1), 9-36.
Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM, 13*(6), 377-387.
Codd, E. F. (1971). *Normalized data base structure: A brief tutorial.* Paper presented at the ACM-SIGFIDET 1971 Workshop: data processing, access and control, San Diego, CA.
Date, C. J. (2000). *An introduction to database systems* (7th ed.). Reading, Mass.: Addison-Wesley.
Lee, T. A., Bishop, A. C., & Parker, R. H. (1996). *Accounting history from the Renaissance to the present: A remembrance of Luca Pacioli (New works in accounting history)*. New York: Garland Pub.
Macve, R. H. (1996). Pacioli's Legacy. In T. A. Lee & A. C. Bishop & R. H. Parker (Eds.), *Accounting history from the Renaissance to the present: a remembrance of Luca Pacioli (New works in accounting history)* (pp. xxiv, 290). New York: Garland Pub.
McCarthy, W. E. (1979). An Entity-Relationship View of Accounting Models. *The Accounting Review, 54*(4), 667.
McCarthy, W. E. (1982). The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment. *The Accounting Review, 57*(3), 554.
Oz, E. (2004). *Management Information Systems* (4th ed.). Boston, MA: Course Technology.
Salim Koussa, R., & Ferran-Urdaneta, C. (2001). *Neither a perpetuum mobile nor a perfect software: Sincerity in the relationship between the manufacturer and the client with respect to software defects.* Paper presented at the 5th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI'01/ISAS'01), Orlando, FL.
Weber, R. (1986). Data Models Research in Accounting: An Evaluation of Wholesale Distribution Software. *The Accounting Review, 61*(3), 498-518.