A User-Oriented Approach To Data Modeling: A Blueprint For Generating Financial Statements And Other Accounting-Related Documents And Reports

Ting J. (TJ) Wang (E-mail: tjwang@uwm.edu), University of Wisconsin-Milwaukee
Hui Du (Email: huidu@bryant.edu), Bryant College
Hur-Li Lee (Email: hurli@uwm.edu), University of Wisconsin-Milwaukee

Abstract

In contrast to traditional and innovative data models in the database design process, such as the Entity-Relationship (E-R) and Resource Event Agent Location (REAL) models, respectively, a user-oriented approach to the data modeling of a relational database to satisfy users' information needs is presented. Although relational database management systems (RDBMS) are powerful for organizing, manipulating, and retrieving data, they are inadequate if the needed data elements are not captured and included, required relationships are not identified and implemented, or incorrect relationships are identified in the data model because the needs of information users were neglected or identified incorrectly during the data modeling. Using the case of an accounting information system (AIS) for a simple merchandising enterprise, this paper illustrates how user information needs can be met when a user-oriented approach is followed in the data modeling. Specifically, it provides a blueprint for generating financial statements and other accounting-related documents and reports from a relational database that utilizes user perspectives.

1. Introduction

The inability of accounting information systems (AIS) to provide non-financial statements information has been acknowledged for more than six decades (Goetz 1939; Firmin 1966; Hollander, Denna, and Cherrington 2000). During this period, and concurrent with advances in information technology, innovative models have been proposed (e.g., events accounting, database accounting, and REA accounting) to address the problem (Johnson 1970; Everest and Weber 1977; McCarthy 1982). As compared to earlier AIS, recent models concentrate more on modeling the broad conceptual essence underlying the business artifacts (business processes and events and their related agents, resources, and locations) (Denna, Jasperson, Fong, and Middleman 1994) and less on generating information artifacts such as accounting documents and reports. Consequently, to the detriment of many users, specific information in accounting documents and reports has become sketchy and ignored in the design process. Sometimes this lack of attention to user requirements can be overcome by the power of the information technology, but, in other instances, a system's ability to deliver specific information (artifacts) is limited or non-existent because required data was not captured in the first place (e.g., Perry and Schneider 2001)[1].

This problem in AIS can be addressed by a user-oriented approach to data modeling that supplements existing data models such as the Entity-Relationship (E-R) or Resource Event Agent Location (REAL) so that all information required by users is provided. For example, both E-R and REAL usually follow a traditional process of

Readers with comments or questions are encouraged to contact the authors via email.

system requirements analysis which results in a system that satisfies its database and business process designers. In contrast, a user-oriented approach to data modeling starts with the traditional system requirements analysis process but concludes with a "verification" process that utilizes the views of the system's users to arrive at their satisfaction with the final product. The particular needs of information users are specified, documented, and completely verified in the data modeling to insure the system's subsequent relevance and usefulness to users.

In the case chosen for this paper, the user-oriented approach will be applied to routine accounting information needs within an organization[2]. Every business has routine accounting procedures to handle general operations and meet the laws and regulations mandated by local, state, and federal agencies. Information needs of this type are constant and standard. On the other hand, businesses also need accounting information for managerial decisions that are dependent on internal and external situations. Information needs of this second type are highly variable and less predictable. While the user-oriented approach can be used with either information type, we chose routine accounting information needs for demonstration because the user information needs are pre-defined and can be verified in the data modeling. In addition, a blueprint (Fig. 2) is provided to design an accounting information system that incorporates specific user requirements for generating routine financial statements information and other accounting-related documents and reports which may be needed.

We begin by reviewing the architecture of a traditional AIS in Section I and the architecture of E-R based AIS in Section II to demonstrate the differences in viewpoint between accounting information users and database designers: a communication problem that is often reflected in the subsequent system design. In Section III, we discuss the problems in the system requirements analysis process. In Section IV, we present the case of an accounting information system for a simple merchandising enterprise designed using the verification process. Our summary is presented in Section IV.

2. Traditional AIS Architecture

In traditional AIS architecture, a chart of accounts and double-entry bookkeeping provide the primary scheme for organizing, classifying, and aggregating financial information (McCarthy, 1979). In the process, data about business activities is first identified and documented and then processed, based primarily on a firm's needs for financial statements. As a result, the traditional AIS architecture is biased toward one primary aim: financial statements (Hollander, Denna, and Cherrington, 2000). For example, after business activity data is captured and documented, it is recorded in one of the subsystems: general or specialized journals such as sales, purchase, cash receipt, and cash disbursement, which are designed according to the firm's needs for specific, summarized information about its business activities (see Fig. 1). Next, the data stored in the journals is re-organized through a periodic posting procedure and then re-stored in the general ledger arranged by the accounts in the chart of accounts. Some of the data are posted first to the accounts in the subsidiary ledgers created to show detailed information of the control accounts in the general ledger, such as A/R and A/P. Adjusting entries are recorded in the general journal and then also posted to the general ledger before the preparation of the financial statements. Finally, information about financial statements is gathered from the accounts stored in the general ledger. To end a specific accounting report period and prepare for a new one, closing entries are recorded in the general journal and posted to the general ledger. Adjusting entries are then reversed at the beginning of the next new period.

All data captured from business activities are documented and maintained in the traditional AIS architecture, but to be processed, data must be classifiable by the chart of accounts designed for the financial statements. In other words, data will not be processed if the existing chart of accounts cannot classify it or no new account can be created to classify it. Debits and credits are used to represent increases and decreases in the accounts in the process (double-entry system). In sum, the traditional AIS can be characterized as an account-oriented system that is biased toward financial statements; many events and corresponding data are not recorded and data entries are redundant.

3. Entity-Relationship – (E-R) Based AIS Architecture

E-R-based AIS architecture alleviates many problems associated with traditional AIS. In the conventional data models such as E-R, all of the potential information that will be needed by information users, including financial and non-financial information about business activities, are first identified through a system requirements analysis process (discussed in the Section III) and then organized using entities based on the business operating rules adapted in the organization (see Fig. 1). Entities can be defined as any person, place, object, event, or concept

(McFadden, Hoffer, and Prescott, 1999). For example, entities defined as a person may include, but are not limited to, customer, vendor, supplier, and employee. Next, relationships among entities are identified and established to link the data stored in the entities. Thus, data about business activities are organized by, and stored in, the entities. Information stored in two or more related entities, or related attributes in unrelated entities (or even unrelated attributes with compatible data types), can be retrieved using a query language such as SQL and QBE. A contrast between the traditional and E-R-based AIS architecture is illustrated in Figure 1.

The advantages of using a database system include minimizing data redundancy, preventing anomalies of inserting, updating, and deleting, and achieving task-data independence (Hall, 1998). To achieve these advantages, certain forms (properties or constraints) must be imposed on the entities (or tables in DBMS). The least restrictive constraint is called the first normal form, followed by the second, third, Boyce-Codd, fourth, fifth, and Domain/Key forms. Most accounting systems require use of the first three normal forms (Perry and Schneider, 2001). The first normal form imposes a very basic requirement on entities by restricting repeating attributes (or fields) and divisible data in an attribute; the second normal form requires dependency of the non-key attributes on the key attribute(s); and the third normal form eliminates transitive dependency, which means that non-key attributes cannot be dependent on any other non-key attributes (for further discussion, see Gelinas, Sutton, and Oram, 1999).

According to the first three normal forms, information about a specific business activity stored in the database system will be totally disaggregated, which makes setting up correct relationships between entities a crucial element in the data modeling. To establish relationships between entities, common attribute(s) must be included in the entities, that is, the same data fields must be created in the related tables)—and for this reason, data redundancy cannot be completely eliminated. Without the relationships between entities, for example, information about persons, places, objects, events, and concepts (as entities) are all isolated from each other: not only between the types of entities but also within the types of entities (e.g., purchase event and inventory, sales event and cash receipt event, respectively). Consequently, information stored in more than one entity can be extracted only if relationships are built between these entities[3]. For example, information about an "event" involving an "object" and carried out by a "person" at a "place" can be retrieved from the system only if relationships are created between these various entities.

All data gathered from business activities are stored in the entities. Relationships between entities are identified and established, based on the user information needs and business rules. In sum, financial or non-financial information can be provided by the E-R-based AIS architecture.

4. System Requirements Analysis Process

To ensure that user information needs are satisfied, database designers (or system analysts) must identify and determine the user information needs before the conception of a new database system takes place. The database design literature has long recognized the necessity of the system requirements analysis process as the very first step in the design process (e.g., Allen, 1996; Carlis and Maguire, 2001; Fidel, 1987; Hernandez, 1997). During this process, a number of tools are commonly used to identify users' needs, such as interviewing users, observing processes that involve users, analyzing existing systems and examining their utilization. In addition, a list of items is assembled including basics such as descriptions of all processes and data elements, data structure, copies of system inputs and outputs, and documentation (Romney and Steinbart, 2000).

Regardless of the tools used and items obtained, database designers may still encounter difficulties in identifying and determining user information needs. Some of the possible obstacles include: 1) users may have difficulty in expressing or articulating their needs or even correctly identifying them; (2) designers may have difficulty in understanding users' needs and/or business processes and rules because they lack necessary knowledge and experience; and (3) serious communication gaps may exist between users and designers due to their different backgrounds or other reasons (Krasner, Curtis, & Iscoe, 1987; Long et al., 1983, Sonnenwald, 1995). Ultimately, system requirements are usually defined by the database designers based on their own understanding of user information needs, existing systems, and business processes and rules, whether this understanding is correct or not. As a result, problems with the database can be discerned by users only after the system is built and they attempt to use it.

5. A Simple Case

We propose an effective database design approach that will guarantee to correctly and comprehensively capture both routine and *ad hoc* user information needs. Using a simple case, we will demonstrate how the user-oriented approach assures the inclusion of user information needs through a verification process at the end of the data modeling. We will avoid lengthy descriptions of the system requirements analysis process in order to stay focused on the verification process.

Briefly stated, the system designer (employing an E-R model) first uses specifications gathered from the system requirements analysis process to identify the entities and the relationships between entities. Next, the designer utilizes a verification process with users to confirm that: 1) the needed data elements for the information artifacts such as documents and reports are included in the entities or can be derived; 2) correct relationships between entities are established in the design; and 3) the required information (such as financial statements and other accounting related documents and reports) actually can be obtained from the database—this latter by constructing specific queries which are verified by the users.

To demonstrate the user-oriented approach, the data modeling of an accounting information system for a simple merchandising enterprise will serve as a case. Assume that the system designers have already defined the system requirements with their best knowledge, correct or not, based on the information acquired in the system requirements analysis process. Table 1 shows entities identified within specific business processes using the E-R (or REAL) model. We have simplified the case by limiting the types of business activities involved. In the Acquisition/Payment process, we include only: Events such as Purchase Ordering, Goods Receiving, Purchase Invoicing, and Purchase Payment Disbursing; Person (Agent) such as Vendor and Employee; Place (Location) such as Warehouse; and Object (Resource) such as Inventory and Cash. In the Sales/Collection process, we include: Events such as Sales Ordering, Product Shipping, Sales Invoicing, and Sales Payment Collecting; Person (Agent) such as Salesperson and Customer; Place (Location) such as Warehouse; Object (Resource) such as Inventory and Cash.

In Figure 2, we present a simplified E-R[4] diagram based on the entities recognized in Table 1 and the system designers' understanding of the user information needs and business rules. Although actual business policies/practices should be the basis upon which the relationships required are established, we set up the relationships based on standard business practices with less restrictive rules in order to avoid the coverage of specific business policies/practices. The E-R diagram is discussed later in the paper when the verification process is applied.

Table 2 presents some general types of information needed by the accounting information users which correspond to the business activities assumed in the case. They are the routine and standard types of documents and reports users need to carry out business activities in business processes under the traditional AIS architecture. For example, in the Acquisition/Payment process, users require documents such as Purchase Order, Receiving Report, and Check, and reports such as Purchase Journal, Cash Disbursement Journal, Inventory Subsidiary, and Accounts Payable Subsidiary. In the Sales/Collection process, users require documents such as Sales Order, Bill of Lading, Shipping Order, and Sales Invoice, and reports such as Sales Journal, Cash Receipt Journal, and Accounts Receivable Subsidiary. In addition, accounting information users must periodically generate financial statements. To simplify the case, we include only account information required for the balance sheet and income statement, such as Cash, Accounts Receivable, Inventory, Accounts Payable, Sales, and Cost of Goods Sold (see Table 3).

The verification process starts when the system designers complete the E-R diagram--the data model in Fig. 2. In the verification process, the system designers must ensure that data items in each of the documents and reports (in Table 2 and 3): 1) are included in the entities; 2) can be derived from other items in the related entities; or 3) can be generated from the system. This is a very important step because the system cannot provide the information if it has not been captured. In the case presented here, all of the documents listed in Table 2, e.g., Purchase Order or Receiving Report, are used to capture and carry out business activities in the events, such as Purchase Ordering and Goods Receiving, respectively, recognized by the E-R and REAL models in Table 1. Data in these documents must be stored and producible directly from the corresponding "event" entities in the database system as indicated in the "Entity" column in Tables 2 and 3. In our sample case, data in some of the reports are also directly based in specific event entities. Similarly, data in these reports also must be stored and producible from the corresponding event entities. For example, a Purchase Journal contains a list of purchasing records for activities that occurred in the Purchase Invoicing event, and a Cash Disbursement Journal in the Purchase Payment Disbursing event, as well as Sales

and Cash Receipt Journals. The system designers can ensure that they have included data from documents and reports by going through the documents and reports themselves and/or by involving users' assistance.

The system designers must verify with users that all the needed attributes (data fields) are included in the entities in order to generate specific reports, such as Accounts Payable Subsidiary, Accounts Receivable Subsidiary, and Inventory Subsidiary, as well as Balance Sheet accounts such as Cash, Accounts Receivable, Inventory, Accounts Payable, and Cost of Goods Sold (Fig. 3). Verification of attributes by the users themselves is crucial, especially if the system designers lack accounting knowledge.

For reports that are not directly based in any one specific business event, queries involving two or more entities must be constructed to ensure that all of the information needed on the reports can be obtained from the database. In our sample case, the system designers must ensure that relationships have been established correctly between related entities in the data model so that information for needed reports actually can be generated. In looking at the simplified E-R diagram in Figure 2, for example, we see that the Inventory Subsidiary report requires the involvement of entities such as Inventory, Goods Receiving, and Product Shipping. Given the many-to-many relationships between Inventory and Goods Receiving and between Inventory and Product Shipping, we know that information required for the Inventory Subsidiary report is obtainable. However, the database designers must construct an actual query for an Inventory Subsidiary report to guarantee that the report can be retrieved. (Sample detailed queries using SQL are included in the Appendix with the corresponding reference numbered in Tables 2 and 3).

If any required attributes are omitted or any incorrect relationships are set up between entities, the ability of the database system to meet users' needs will be limited. For example, accounts receivable by invoice number (or accounts payable by due date) cannot be generated without a relationship between Sales Invoicing (Purchase Invoicing) table and Sales Payment Collecting (Purchase Payment Disbursing) table linked by the sales invoice number (purchase invoice number) attribute (see Endnote 1).

5. Conclusion and Suggestions For Future Research

The proposed user-oriented approach containing a verification process makes a significant contribution to relational database design in AIS by articulating systematic steps for developing an accounting information system that meets users' needs. Explicit strategies in the approach are crucial for ensuring the inclusion of specific entities and attributes, the establishment of required relationships between related entities, and the production of needed information. Future research is needed to further evaluate the effectiveness of this approach and to expand the model to incorporate other considerations for *ad hoc* decision-making. 🕮

Endnotes

1. Based on their modeling of revenue cycle (p. 271), reports regarding accounts receivable can be constructed only through the attribute "customer account." As a result, this data model cannot generate an accounts receivable aging report by invoice--the most commonly used information for managing cash flow--because the invoice number was not captured in the Cash Receipt table.
2. The information needs in *ad hoc* decision making are unpredictable and should be a special consideration within the context of each decision, but they are applicable.
3. Some information may be retrievable without proper relationships being established between entities, but special cares need to be taken. These tasks usually are performed by database specialists who have advanced knowledge about the query language.
4. To simplify, we include only entities and cardinalities in the diagrams.

References

1. Allen, Bryce L. *Information Tasks: Toward a User-Centered Approach to Information Systems*. San Diego: Academic Press, 1996.
2. Carlis, John, and Joseph Maguire. *Mastering Data Modeling: A User-Driven Approach*. Boston: Addison-Wesley, 2001.
3. Denna, Eric L., John Jasperson, Kenny Fong, and David Middleman. "Modeling Conversion Process

Events." *Journal of Information Systems* 8, no. 1 (1994): 43-54.
4. Everest, Gordon C., and Ronald Weber. "A Relational Approach to Accounting Models." *Accounting Review* 52, no. 2 (1977): 340-359.
5. Fidel, Raya. *Database Design for Information Retrieval: A Conceptual Approach*. New York: John Wiley & Sons, 1987.
6. Firmin, Peter A. "The Potential of Accounting as a Management Information System." *Management International Review* (1966): 45-55.
7. Gelinas, Ulric J., Steve G. Sutton, and Allen E. Oram. *Accounting Information Systems*. 4th ed. Cincinnati, OH: Southwestern College Publishing, 1999.
8. Goetz, B.E. "What's Wrong with Accounting." *Advanced Management* (1939): 151-157.
9. Hall, James A. *Accounting Information Systems*. 2nd ed. Cincinnati, OH: Southwestern Publishing, 1998.
10. Hernandez, Michael J. *Database Design for Mere Mortals: A Hands-on Guide to Relational Database Design*. Reading, Mass.: Addison-Wesley Developers Press, 1997.
11. Hollander, Anita S., Eric L. Denna, and J. Owen Cherrington. *Accounting, Information Technology, and Business Solutions*. 2nd ed. Burr Ridge, IL: Irwin/McGraw-Hill, 2000.
12. Johnson, O. "Toward an "Events" Theory of Accounting." *Accounting Review* 45, no. 4 (1970): 641-652.
13. Krasner, H., B. Curtis, and N. Iscoe. "Communication Breakdowns and Boundary Spanning Activities on Large Programming Projects." In *Empirical Studies of Programmers: Second Workshop*, edited by G.M. Olson, S. Shepard and E. Soloway, 47-64. Norwood, NJ: Ablex, 1987.
14. Long, J., N. Hammond, P. Barnard, J. Morton, and I. Clark. "Introducing the Interactive Computer at Work: The Users' View." *Behaviour & Information Technology* 2, no. 1 (1983): 39-106.
15. McCarthy, William E. "An Entity-Relationship View of Accounting Models." *Accounting Review* 54, no. 4 (1979): 667-686.
16. ———. "The Rea Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment." *Accounting Review* 57, no. 3 (1982): 554-578.
17. McFadden, Fred R., Jeffrey A. Hoffer, and Mary B. Prescott. *Modern Database Management*. 5th ed. Reading, MA: Addison-Wesley, 1999.
18. Perry, James T., and Gary P. Schneider. *Building Accounting Systems Using Access 2000*. Cincinnati, OH: Southwestern College Publishing, 2001.
19. Romney, Marshall B., and Paul John Steinbart. *Accounting Information Systems*. 8th ed. Upper Saddle River, NJ: Prentice Hall, 2000.
20. Sonnenwald, Diane H. "Contested Collaboration: A Descriptive Model of Intergroup Communication in Information System Design." *Information Processing & Management* 31, no. 6 (1995): 859-877.

Figure 1
Traditional vs. E-R-based AIS architecture

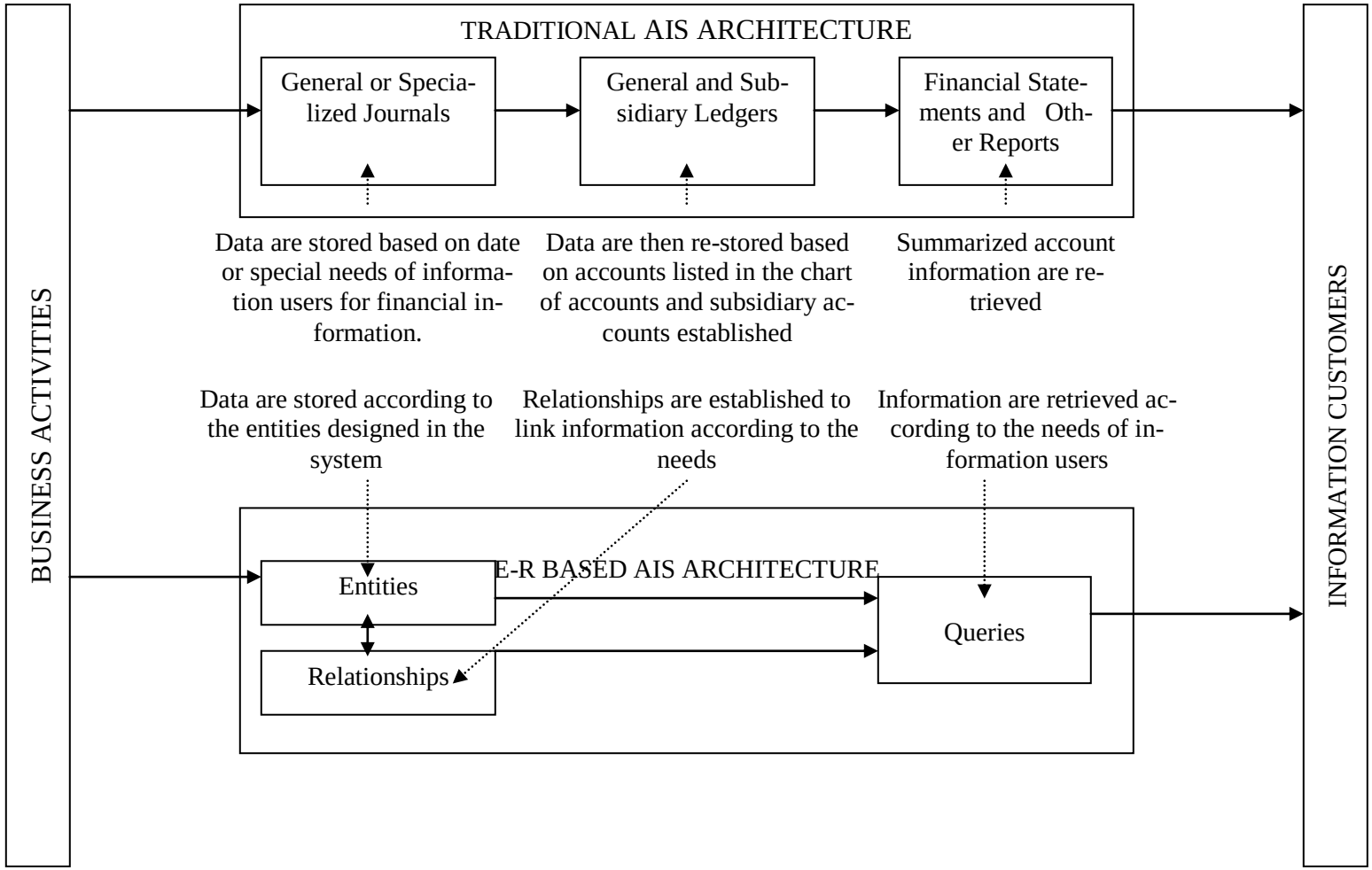
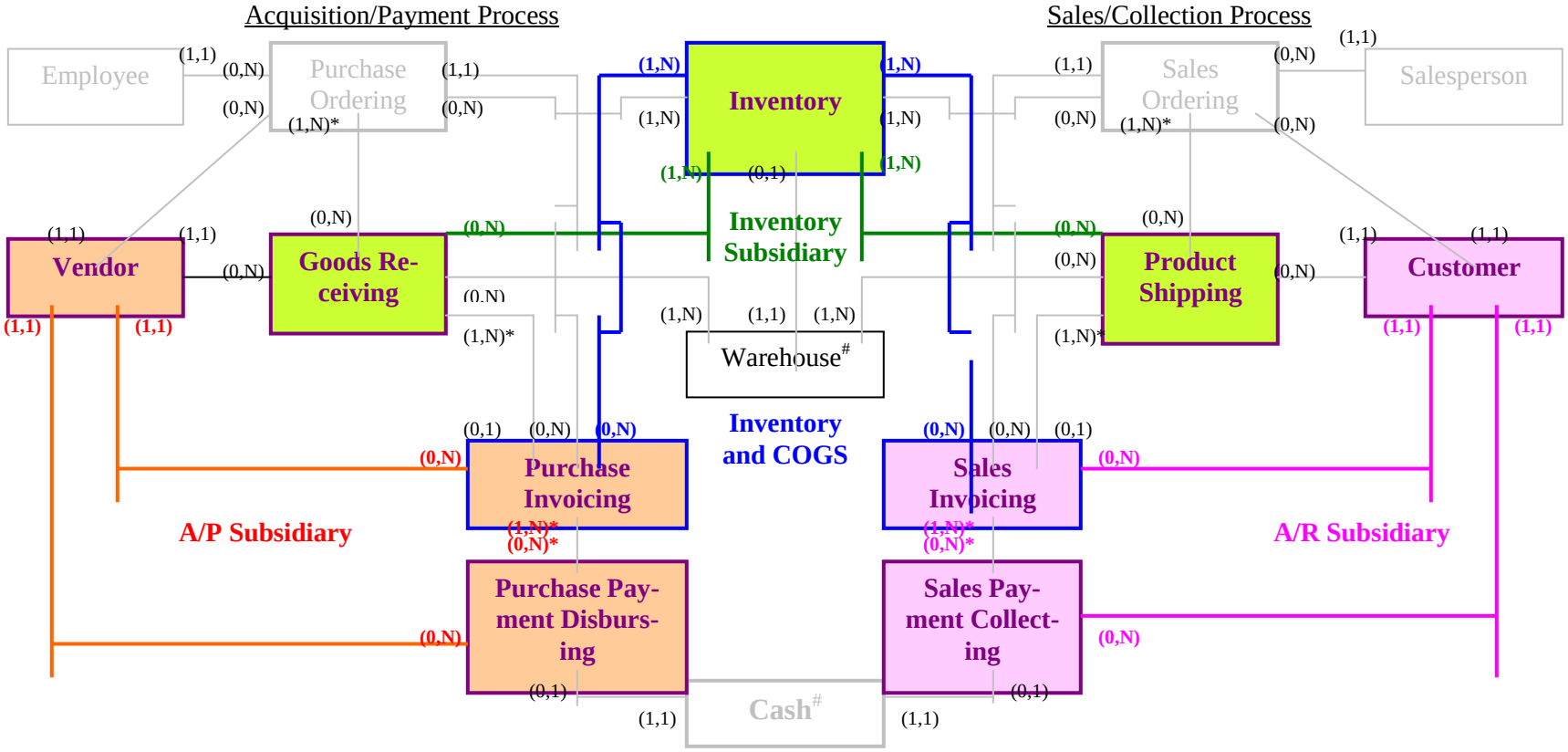


Figure 2
A simplified E-R diagram of a merchandising enterprise



* Less restrictive rules.
\# Optional.

Figure 3
Database structure of a merchandising enterprise

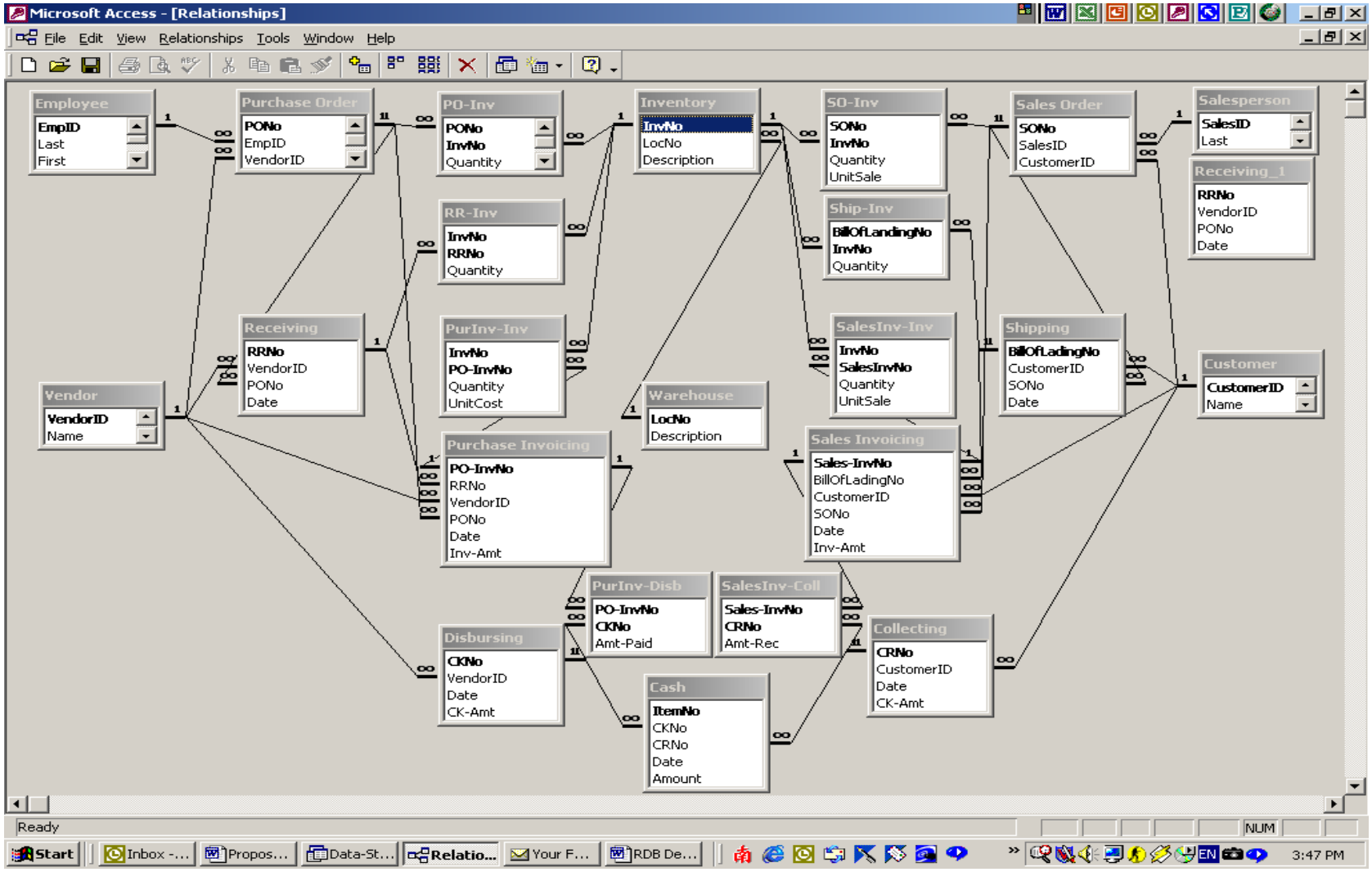


Table 1
Business Process vs. Entity Type For A Simple Merchandising Enterprise

Business Process	Entity Type Based on E-R (REAL) Model			
	Person (Agent)	Place (Location)	Object (Resource)	Event (Event)
Acquisition/ Payment	Vendor Employee	Warehouse	Inventory Cash	Purchase Ordering (Purchase Dept) Goods Receiving (Shipping & Receiving Dept) Purchase Invoicing* (Acctg Dept) Purchase Payment Disbursing (Acctg Dept)
Sales/Collection	Salesperson Customer	Warehouse	Inventory Cash	Sales Ordering (Sales Dept) Product Shipping (Shipping & Receiving Dept) Sales Invoicing (Acctg Dept) Sales Payment Collecting (Acctg Dept)

* Matching of purchase order, receiving report, and vendor's invoice, and recording of purchase are involved in this event.

Table 2
Business Process, Information Type, And Entity

Business Process	Information Type		Entity	SQL**
Acquisition/ Payment	Document	Purchase Order	Purchase Ordering*	
		Receiving Report	Goods Receiving*	
		Check	Purchase Payment Disbursing*	
	Report	Purchase Journal	Purchase Invoicing*	
		C/D Journal	Purchase Payment Disbursing*	
		Inventory Subsidiary	Inventory (●); Goods Receiving (+); Product Shipping (-)	Query #1
		A/P Subsidiary	Vendor (●); Purchase Invoicing (+); Purchase Payment Disbursing (-)	Query #2
Sales/Collection	Document	Sales Order	Sales Ordering*	
		Bill of Lading	Product Shipping*	
		Shipping Order	Product Shipping*	
		Sales Invoice	Sales Invoicing*	
	Report	Sales Journal	Sales Invoicing*	
		C/R Journal	Sales Payment Collecting*	
		A/R Subsidiary	Customer (●); Sales Invoicing (+); Sales Payment Collecting (-)	Query #3

* corresponding entity of the document or report.
** indicating query number in the Appendix.
(●) the key of the entity serves as a basis in the query with other entities;
(+) specific information in the entity is included in the query positively;
(-) specific information in the entity is included in the query negatively.

Table 3
Financial Statement Account And Entity

Financial Statement	Type	Account	Entity	SQL**
Balance Sheet	Assets	Cash	Cash, or Sales Payment Collecting (+) Purchase Payment Disbursing (-)	Query #4
		A/R	Sales Invoicing (+) Sales Payment Collecting (-)	Query #5
		Inventory	Inventory (●) Purchase Invoicing (+) Product Shipping (-)	Query #6
	Liabilities	A/P	Purchase invoicing (+) Purchase Payment Disbursing (-)	Query #7
Income Statement	Revenue	Sales	Sales Invoicing*	Query #8
	Expense	COGS	Inventory (●) Sales Invoicing (-) Purchase Invoicing (+)	Query #9

* corresponding entity of the document or report.
** indicating query number in the Appendix.
(●) the key of the entity serves as a basis in the query with other entities;
(+) specific information in the entity is included in the query positively;
(-) specific information in the entity is included in the query negatively.

Note: A "master" entity consisting of the beginning balances for balance sheet accounts may be needed if the business is involved in a large volume of activities. If so, this master entity must be included in the query to obtain correct information for balance sheet accounts.

Appendix - Queries For Generating Report Items Using SQL

Note: To fit the whole database structure on one page shown in Figure 3, we simplify some of the entity names; for example, from Goods Receiving to Receiving, from Product Shipping to Shipping, from Purchase Payment Disbursing to Disbursing, and from Sales Payment Collecting to Collecting. Please refer the table names below to Figure 3.

1. Subsidiary Reports

Query #1. Inventory
(*SELECT* 'SHIP', I.Description, SI.InvNo "INVNO", S.Date "DATE", SI.Quantity
FROM Ship-Inv SI, Shipping S, Inventory I
WHERE SI.InvNo = I.InvNo AND SI.BillOfLandingNo = S.BillOfLandingNo)
UNION
(*SELECT* 'RCV', I.Description, RI.InvNo, R.Date "DATE", RI.Quantity
FROM RR-Inv RI, Inventory I, Receiving R
WHERE RI.InvNo = I.InvNo AND RI.RRNo = R.RRNo)
ORDER BY INVNO, DATE

Note: Information about sales order and customer can also be retrieved by involving other attributes or other tables in the query.

Query #2. A/P
(*SELECT* 'PI', V.Name "VENDOR", V.VendorID, PI.Date "DATE", PI.Inv-Amt, PI.PO-InvNo
FROM Purchase Invoicing PI, Vendor V *WHERE* PI.VendorID = V.VendorID)
UNION
(*SELECT* 'PID', V.Name, V.VendorID "VENDOR", D.Date "DATE", PID.Amt-Paid, PID.PO-InvNo
FROM Disbursing D, Vendor V, PurInv-Disb PID
WHERE V.VendorID = D.VendorID AND D.CKNo = PID.CKNo)
ORDER BY VENDOR, DATE

Query #3. A/R
(*SELECT* 'SI', C.Name, C.CustomerID "CUSTOMER", SI.Date "DATE", SI.Inv-Amt, SI.Sales-InvNo
FROM Sales Invoicing SI, Customer C *WHERE* SI.CustomerID = C.CustomerID)
UNION
(*SELECT* 'SIC', C.Name "CUSTOMER", C.CustomerID, CO.Date "DATE", SIC.Amt-Rec, SIC.Sales-InvNo
FROM Collecting CO, Customer C, SalesInv-Coll SIC
WHERE C.CustomerID = CO.CustomerID AND CO.CRNo = SIC.CRNo)
ORDER BY CUSTOMER, DATE

2. Financial Statements Accounts

Query #4. Cash
SELECT SUM(Amount) *FROM* Cash, or

SELECT SUM(CK-Amt) "Coll" *INTO* #Total_Coll *FROM* Collecting
SELECT SUM(CK-Amt) "Disb" *INTO* #Total_Disb *FROM* Disbursing
SELECT Coll – Disb *FROM* #Total_Coll, #Total_Disb

Query #5. A/R
SELECT SUM(Amt-Rec) "T_Coll" *INTO* #Total_Coll *FROM* Collecting
SELECT SUM(Inv-Amt) "T_Inv" *INTO* #Total_SI *FROM* Sales Invoicing
SELECT T_Inv – T_Coll *FROM* #Total_Coll, #Total_SI

Query #6. Inventory

Calculation of the FIFO (LIFO) inventory will be performed as 'FIFO'('LIFO') in the procedure, Ending_Inventory_Proc. Procedure COGS_Per_Item_Proc and COGS_All_Item_Proc are auxiliary procedures for generating COGS per inventory item and all inventory items.

```
CREATE PROC Ending_Inventory_Proc (@METHOD CHAR(4)) AS

IF @METHOD != 'FIFO' AND @METHOD != 'LIFO'
   BEGIN
      PRINT  "Parameter must be either FIFO or LIFO"
      RETURN
   END

EXEC COGS_ALL_ITEM_PROC @METHOD

SELECT InvNo, SUM(Quantity * UnitCost) "Rcv_Amt" INTO #Rcv_Inventory FROM PurInv-Inv
GROUP BY InvNo

SELECT R.InvNo, R.Rcv_Amt – A.COGS FROM #All_Sold A, #Rcv_Inventory R WHERE A.InvNo = R.InvNo

RETURN
```

```
CREATE PROC COGS_All_Item_Proc (@METHOD CHAR(4)) AS

CREATE TABLE #All_Sold (InvNo CHAR(15), COGS DECIMAL(12,2))

DECLARE Quan_Sold_Curs CURSOR FOR
SELECT InvNo, SUM(Quantity) FROM SalesInv-Inv GROUP BY InvNo
FOR READ ONLY

DECLARE
    @InvNo CHAR(15),
    @Sold_Count INT,
    @COGS DECIMAL(12,2)

OPEN Quan_Sold_Curs
FETCH Quan_Sold_Curs INTO @InvNo, @Sold_Count

WHILE (@@SQLSTATUS = 0)
   BEGIN
      EXEC COGS_Per_Item_Proc @METHOD, @InvNo, @Sold_Count,
      @COGS OUTPUT
      INSERT #All_Sold VALUES(@InvNo, @COGS)
      FETCH Quan_Sold_Curs INTO @InvNo, @Sold_Count
   END

CLOSE Quan_Sold_Curs
DEALLOCATE CURSOR Quan_Sold_Curs

RETURN
```

```
CREATE PROC COGS_Per_Item_Proc (@METHOD CHAR(4), @InvNo CHAR(15), @Sold_Count INT, @COGS
DECIMAL(12,2) OUTPUT) AS

IF @METHOD = 'FIFO'
   BEGIN
```

```
      DECLARE Quan_Rcv_Curs CURSOR FOR
      SELECT PII.Quantity, PII.UnitCost FROM PurInv-Inv PII, Purchase Invoicing PI
      WHERE PII.InvNo = @InvNo AND PII.PO-InvNo = PI.PO-InvNo
      ORDER BY PI.Date ASC
      FOR READ ONLY
   END
ELSE
   BEGIN
      DECLARE Quan_Rcv_Curs CURSOR FOR
      SELECT PII.Quantity, PII.UnitCost FROM PurInv-Inv PII, Purchase Invoicing PI
      WHERE PII.InvNo = @InvNo AND PII.PO-InvNo = PI.PO-InvNo
      ORDER BY PI.Date DESC
      FOR READ ONLY
   END

DECLARE
   @Quantity INT,
   @UnitCost       DECIMAL(12,2),
   @OFFSET INT

OPEN Quan_Rcv_Curs
FETCH Quan_Rcv_Curs INTO @Quantity, @UnitCost

SELECT @OFFSET = @Sold_Count
WHILE (@@SQLSTATUS = 0)
   BEGIN
      IF (@OFFSET >= @Quantity)
         BEGIN
            SELECT @COGS = @COGS + @Quantity * @UnitCost
            SELECT @OFFSET = @OFFSET - @Quantity
         END
      ELSE
         BEGIN
            SELECT @COGS = @COGS + @OFFSET * @UnitCost
            BREAK
         END

      FETCH Quan_Rcv_Curs INTO @Quantity, @UnitCost
   END

CLOSE Quan_Rcv_Curs
DEALLOCATE CURSOR Quan_Rcv_Curs

RETURN
-------------------------------------------------------------------------------------------------
```

Query #7. A/P

```
SELECT SUM(Amt-Paid) "T_Disb" INTO #Total_Disb FROM PurInv-Disb
SELECT SUM(Inv-Amt) "T_Inv" INTO #Total_PI FROM Purchase Invoicing
SELECT T_Inv – T_Disb FROM #Total_Disb, #Total_PI
```

NOTE: #Total_Disb and #Total_PI are temporary tables, which do not need to be created.

Query #8. Sales

SELECT SUM(Inv-Amt) *FROM* Sales Invoicing SI WHERE SI.Date BETWEEN #1/1/01# AND #12/31/00#

NOTE: SI.Date must be specified.

Query #9. Cost of Goods Sold

Calculation of the FIFO (LIFO) cost of goods sold will be handled as 'FIFO' ('LIFO') in the procedure, COGS_Proc. Procedure COGS_Per_Item_Proc and COGS_All_Item_Proc are auxiliary procedures for calculating COGS per inventory item and all inventory items. They are also used in generating ending inventory reports.

```
CREATE PROC COGS_Proc
   (@METHOD CHAR(4)) AS

IF @METHOD != 'FIFO' AND @METHOD != 'LIFO'
   BEGIN
      PRINT  "Parameter must be either FIFO or LIFO"
      RETURN
   END

EXEC COGS_All_Item_Proc @METHOD

SELECT * FROM #All_Sold

RETURN
```

Notes