Implementing The Information Processes Underlying The Sales/Collection Cycle Using Access

Laurence R. Paquette (E-mail: lpaquette1@aol.com), Westfield State College

Abstract

The objective of this paper is to provide the reader with some exposure as to how accounting information is recorded and processed when a strict relational database approach is implemented. The application described is from the sales/collection cycle and it should provide the information customer with a meaningful Access database exercise that will allow the user to implement the concepts set forth in good database design, as well as the event-driven approach to business technology solutions. The text of this article is excerpted in the form of a student exercise and is included as an Appendix, which can be used as an assignment by the AIS professor.

Introduction

The goal of this paper is to demonstrate how Access can be used to implement some of the information processes involved in the Sales/Collection cycle as outlined by Hollander, Denna, and Cherington (2000). Managers obtain information from an information system to help them manage the business processes. They identify three business processes that exist, regardless of the type of good or service provided, as follows: acquisition/payment, conversion, and sales/collection. The focus in this paper will be on the sales/collection process. They further describe the underlying information processes as consisting of the following activities:

Recording Business Activity Data - the specific features used to capture business data.

Storing and Maintaining Data - the maintenance procedures for keeping the enterprise-wide database relevant and up to date.

Reporting processes - the procedures used to generate useful information required by those who execute, control, and evaluate business processes.

The Access exercise presented in this paper will touch on all three activities that comprise an information process as outlined above and will attempt to take full advantage of the software's capabilities.

Because information customers (managers) are increasingly being given the responsibility for extracting their own information (developing their own views of the data), they must be aware of how to use the query tools provided with a database. A general objective of this paper is to provide the reader with some training in the use of query tools provided by relational databases

Readers with comments or questions are encouraged to contact the author via e-mail.

and should prove to be especially useful for students in an AIS course, as well as the broader audience of all information customers in an organization.

Specifically the process of querying the database to determine the balance for Accounts Receivable for all customers, as well as for an individual customer when an order is being placed, will be addressed. Knowing a customer's balance at the time an order is taken is a critical piece of information, in that it is needed to determine whether or not a customer has sufficient credit available to process the transaction. Extracting this information when relational database conventions are strictly adhered to is not especially straightforward and is a specific objective of the paper.

The information processes to be implemented in Access will consist of the following:

Data Capture - the form used to capture a sale and provide information to the underlying sales table.

Storing and Maintaining Data - the tables used in the sales/collection cycle.

Reporting Processes - the use of query tools provided with the database to extract the desired information.

In the following section of the paper the relational and traditional approaches of tracking accounts receivable are compared. Next the tables and relationships to be used are outlined. The process of defining total queries for sales and payments is then illustrated. Another total query is designed to incorporate the results of the two prior queries to generate a balance due for each customer. Resolution of null values that occur in this summary process is discussed and resolved. Parameter queries are discussed next so that one can find the balance for a particular customer rather than for all customers. Finally, this information will be integrated on a form that is used to record sales which in turn, provides values to the table containing sales information. The event of entering a customer identification number on the form will cause the customers credit limit as well as their current balance and available credit to be calculated and displayed on the form. This approach provides the data entry clerk with information that supports the decision to fill the order in a routine fashion.

Relational and Traditional Approaches Compared

Perhaps the most popular implementation of a relational approach is the Resources, Events, Agents, and Locations (REAL) model. This approach to maintaining accounting records is the redesign of traditional accounting processes to achieve improvements in performance via the elimination of multiple recording processes. Proponents of the REAL approach argue that there is no reason to rerecord data once all the essential aspects of the resources, events, agents, and locations are captured (Denna, Cherington, Andros, and Hollander, 1993, Walker and Denna, 1997). This approach is consistent with good database design and the concepts of normalization and elimination of redundancies and also requires the abandonment of calculated fields from the records that are maintained in the tables. Arbitrary aggregations, representing a particular user view, are therefore avoided. An example of this is the elimination of the concept of posting to subsidiary ledgers.

Hall (1998) suggests that the abandonment of traditional accounting records and the preparation of financial statements from millions of transactions can be problematic, resulting in the need to periodically summarize transactions from their detail database into a traditional chart-of-accounts structure that better facilitates financial reporting. Since the transaction details remain intact and uncorrupted by this summarization, the objectives of the REAL approach are not compromised. Even proponents of good database design (Hernandez, 1997) suggest that in

some instances one might consider relaxing those principles. Those instances are when an analytical database is being maintained to store and track historical and time dependent data and when processing performance needs to be improved.

Clearly, at some level of activity, it is desirable to summarize activity and improve performance at the expense of duplicate fields and redundant data as suggested by Hall (1998). Paquette (1998) describes the implementation of this approach using real time updates in Access to a subsidiary table that represents the accounts receivable subsidiary ledger. Paquette and Roohani (1998) extended this approach to include batch updates.

A strict database implementation of the REAL approach to derive Accounts Receivable is described by Levitan (1999). As he suggests, this approach is neither straightforward nor trivial. It requires queries, queries of queries and finally an Accounts Receivable report can be generated. Implementation in Access can pose some problems that will be addressed in this paper. Perhaps it is these problems that account for the paucity of information on implementing this approach. An interesting extension is also considered. A form is used to capture the data required to build the sales table, which in turn is a partial source of information required on the form, available credit for that customer, where available credit is the difference between the customer's credit limit and their current account balance.

Tables and Relationships

In order to develop the REAL approach to Accounts Receivable, the tables shown in Figure 1 will be used. In practice there would be many more attributes associated with each table, however they have not been included here in order to allow one to focus on the issue at hand.

See Figure 1

A traditional approach to the sales/collection cycle might include a field for both total sales made to each customer and total payments made by each customer, which would in turn be updated as a result of a transaction to record either a sale to or a cash receipt from that customer. Because these are calculated fields, they are omitted in this approach and are not included in the customer table shown in Figure 1. We see that we have 8 customers in this example, but to get an idea about the activity or balance for any customer one would have to refer to either the Sales or Cash Receipts tables. These tables contain the detail needed to construct the desired reports or respond to a query, i.e. convert the data to information. The goal of this paper is to examine how an information customer can employ the tools available in a relational database to view business events stored in the tables.

In order to take advantage of the benefits afforded by a relational database, one must define relations that exist between the tables. A relationship exists when the tables are

See Figure 2

connected or linked via a Primary key in one table and a foreign key in another table as shown in Figure 2, or are linked by a linking or junction table. For example, a logical relationship exists between the data in the customer table and the sales table, as well as between the customer table and the cash receipts table. Both of these relationships are characterized as being one to many relations as indicated by the lines drawn in Figure 2 to show the relationship. These relationships called join lines will serve a critical role when designing queries in the next section of the paper. Access supports three types of joins called inner joins, left outer joins, and right outer joins and they will be discussed when they are defined later. The tables are joined via the common field CustomerId. CustomerId is a primary field in the Customer table and a foreign key in the Sales and CashReceipts table.

Figure 1
Datasheet View of Tables Used

Customer table

CustomerID	CompanyName	CreditLimit
1000	Paquette	$1,100.00
2000	Rienzo	$1,200.00
3000	Ferris	$1,300.00
4000	Knipes	$900.00
5000	Rathay	$2,000.00
6000	Merlo	$2,100.00
7000	Trump	$4,000.00
8000	Brunelle	$800.00

Sales table

InvoiceNo	Date	SaleAmt	CustomerId
00001	1/1/99	$100.00	1000
00002	1/1/99	$500.00	2000
00003	1/1/99	$600.00	3000
00004	1/1/99	$100.00	4000
00005	1/1/99	$1,500.00	5000
00006	1/1/99	$400.00	6000
00007	1/1/99	$600.00	7000
00008	1/10/99	$50.00	1000
00009	1/10/99	$100.00	2000
00010	1/10/99	$200.00	5000
00011	1/10/99	$300.00	6000
00013	1/10/99	$200.00	7000
00014	1/20/99	$50.00	1000
00015	1/20/99	$300.00	2000
00016	1/20/99	$300.00	5000

Cash Receipts Table

RemitNo	Date	PaymentAmt	CustCheckNo	InvoiceNo	CustomerId
00001	1/5/99	$100.00	4566	00001	1000
00002	1/5/99	$500.00	39099	00002	2000
00003	1/6/99	$600.00	77889	00003	3000
00004	1/6/99	$100.00	7805	00004	4000
00005	1/6/99	$1,500.00	64922	00005	5000
00006	1/7/99	$600.00	55554	00006	7000
00007	1/16/99	$200.00	65004	00010	5000
00008	1/17/99	$100.00	39453	00009	2000
00009	1/17/99	$50.00	4607	00008	1000

Figure 2

Relationships

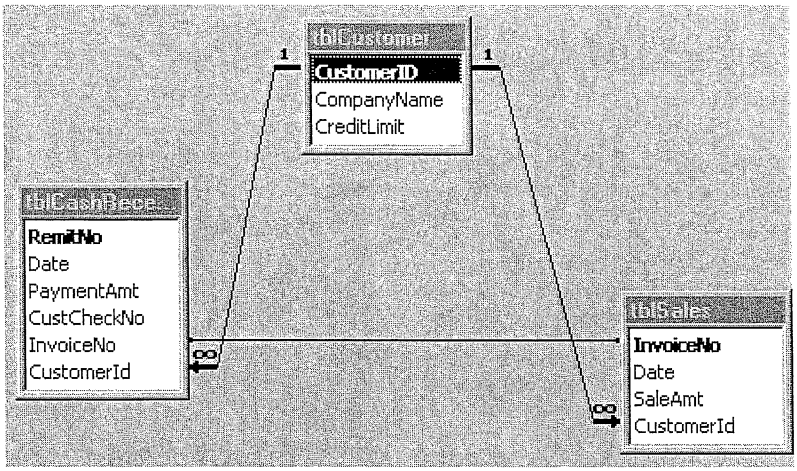


Figure 3

Accounts Receivable—Subsidiary Ledger

	PAQUETTE		RIENZO		FERRIS		KNIPES		RATHAY	
	100	100	500	500	600	600	100	100	1500	1500
	50	50	100	100					200	200
	50		300						300	
Totals	200	150	900	600	600	600	100	100	2000	1700
Balance	50		300		0		0		300	

	MERLO		TRUMP		BRUNELLE	
	400		600	600		
	300		200			
Totals	700		800	600	0	0
Balance	700		200		0	

A one to one relationship is specified between the sales and cash receipts tables via the common field InvoiceNo. In fact this relationship is better described as being a many to many relationship, however it is desired to not introduce the additional concerns that result with this type of specification. One might want to extend this problem in the future by relaxing this assumption.

Before proceeding with the relational approach, it may prove useful to look at the postings that would occur to the subsidiary ledger in a traditional system using the events found in the tables presented in Figure 1. This exercise will also generate the desired results for the queries that will be constructed in the following section. The Accounts Receivable subsidiary ledger is shown in Figure 3.

See Figure 3

Although this traditional approach may involve some redundancies, it does have information content in terms of the number of transactions, debit and credit totals, and current balances. In order to extract similar information content from the Access tables, one might consider the construction of queries.

Total Queries

In this section the focus will be first on performing a total query for sales, and then for payments. It should be noted that a total query is not a separate kind of query; rather, it extends the flexibility of select queries. Designing the query for total sales by customer begins by opening the New Query dialog box and adding the Customer and Sales tables. Note that the Customer table is also being used as part of this query for two reasons. The first is that it is desired to include the field CompanyName in the resulting dynaset, and that field is not included in the Sales table. Secondly, it is desired to include in the dynaset those customers to whom no sales were made. Without joining the Sales table to the Customer table, one would not be aware that no sales were made to customer 8000 -Brunelle. This type of information could be a source of concern to management. A left outer join (option 2 in Access) is used to join the two tables. This ensures that the resulting dynaset will include all records from the table Customer, and only those records from the table Sales where the joined fields are equal. The fields CustomerId and CompanyName from the table Customer and SaleAmt from the table Sales are added to the design grid. Next the totals button on the Query Design toolbar is clicked causing a row labeled "Total" to be added to the design grid between the Table

See Figure 4

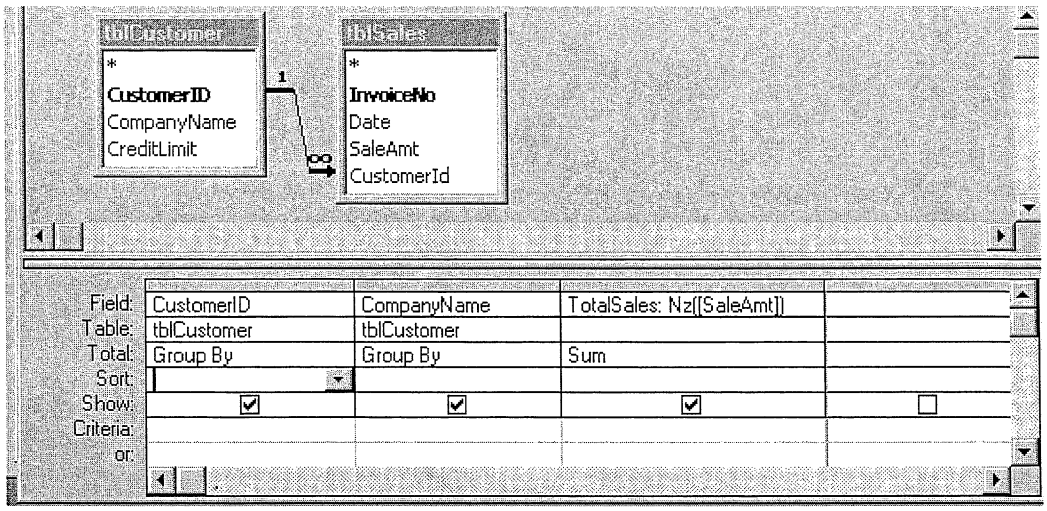
and Sort rows as shown in Figure 4. The term Group By will appear in all three columns of the design grid. Position the cursor in the third column of the Total row and specify the Sum function to calculate the total of all the SaleAmt fields for each customer. Return to the datasheet view to see the resulting dynaset.

Access will automatically assign a datasheet column name of SumOfSaleAmt for the field being totaled. Also note that customer 8000 Brunelle is included in the dynaset and that the SumOfSaleAmt field is zero. A null value was initially returned as a value for this field because no sales had been made to this customer. A null value is the absence of a value, it is not blank, nor zero, nor any other value. One has to test for null values when doing arithmetic with fields and this test can be accomplished with the use of the Nz function in Access. This function allows one to convert a null value to another value and prevent it from propagating through an expression. By default, the null value is converted to a value of zero when using this function and is an alternative to using the Immediate If (IIf) and Is Null functions that some readers may be familiar with.

To obtain the query result shown in the datasheet view of Figure 4 the following modifica-

Figure 4

Select Query for Total Sales by Customer



Design View

CustomerID	CompanyName	TotalSales
1000	Paquette	$200.00
2000	Rienzo	$900.00
3000	Ferris	$600.00
4000	Knipes	$100.00
5000	Rathay	$2,000.00
6000	Merlo	$700.00
7000	Trump	$800.00
8000	Brunelle	$0.00

Datasheet View

tions were made in the design view for this query as shown in Figure 4.

The name TotalSales: was entered before the field name SaleAmt. This will control the naming of this column when viewing the dynaset. Next, the field name SaleAmt is made to appear as the argument to the Nz function. The design grid for this field should now contain the following expression: TotalSales: Nz([SaleAmt])

When the IIf and Is Null functions are used to resolve null values the expression would appear as follows: TotalSales: IIf(IsNull([SaleAmt]),0,[SaleAmt]). This expression will test to see if the sum of the SaleAmt field for any customer is a null value. If it is, then it will be replaced with a zero, if not, then it retains its current value. Finally, the format property of this field should be changed to currency.

The select query to find the total payments made by each customer is accomplished in a similar fashion to the query just described. The tables Customer and CashReceipts are used and the field PaymentAmt is summed as shown in the design view of the query in Figure 5.

See Figure 5

Note that two null values have been replaced with zeros in the resulting dynaset. The customers Brunelle and Merlo have not made any payments in this time frame and the Nz function was used to generate the values of zero. Now that these two total queries have been written, a select query can be designed to calculate a balance for each customer.

Calculating Balances for Accounts Receivable

The query that will determine the balance for each account is similar to the two preceding total queries, in that it also is a select query. The balance query is constructed using the two total queries just constructed. Note, that in the design view of the query shown in Figure 6, that the two queries are linked using an inner join(option 1 in Access). This is done because outer joins were used to construct the two prior total queries, resulting in dynasets that have an equal number of rows (one for each customer).

See Figure 6

The first two fields included in the design grid can be selected from either of the Total queries. In this case they are selected from the TotalSales query along with the field TotalSales. The field TotalReceipts is the only field selected from the TotalReceipts query. A fifth field called Balance is calculated as follows:

Balance: TotalSales-TotalReceipts

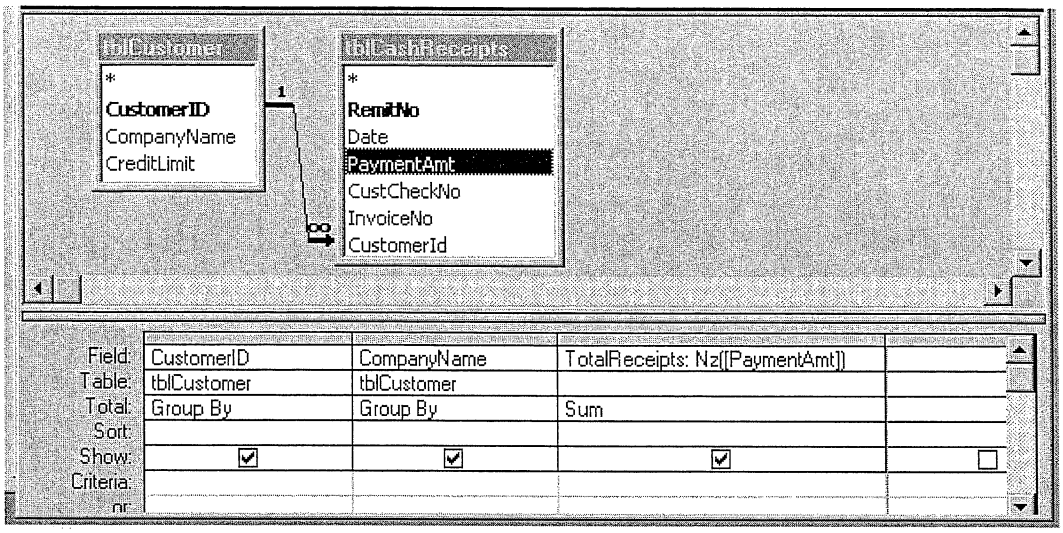
The balance for each customer is simply the difference between the resulting totals for sales and payments. The result of the Balance query is shown in the datasheet view of the query in Figure 6. The reader should note the correspondence between the balances shown here and those in Figure 3. Because the possibility of null values was tested for and resolved in the two prior queries none of the balances should have a null value. If this testing did not take place, then the customer Merlo who has a balance of $700.00 would have had a null value for the Balance field, which is not the correct value. This type of erroneous reporting can lead to an incorrect decision. If it is desired to produce a formatted report the resulting query dynaset can be used as the input source for a report.

Parameter Queries

In the prior two sections the focus was on finding total sales, total receipts, and balances for all customers. One may in fact be interested in obtaining this same information for only one particular customer at a given point in time. This would clearly be the case in deciding on whether to fill an order for a particular customer. This type of query is referred to as a parameter

Figure 5

Select Query for Total Receipts by Customer



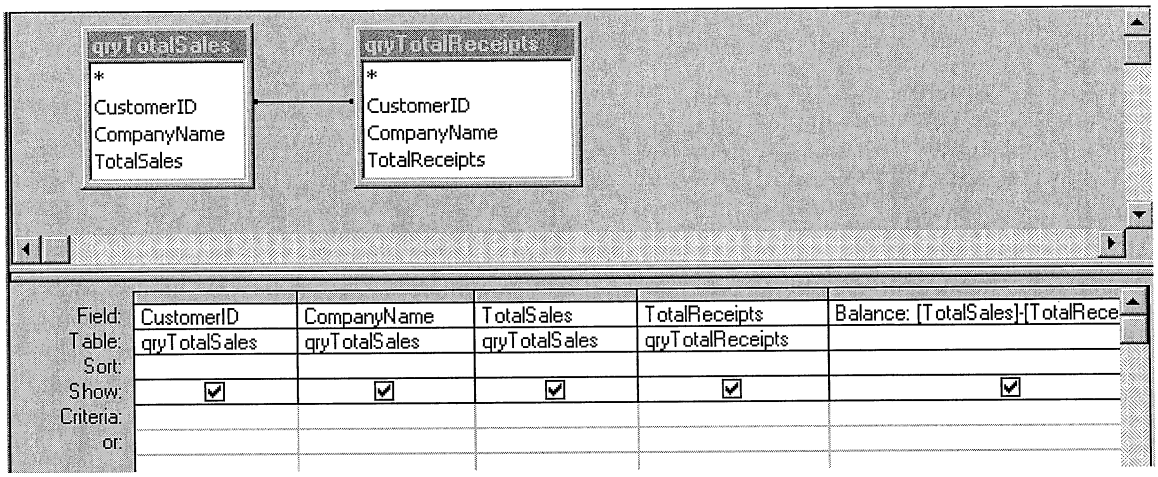
Design View

CustomerID	CompanyName	TotalReceipts
1000	Paquette	$150.00
2000	Rienzo	$600.00
3000	Ferris	$600.00
4000	Knipes	$100.00
5000	Rathay	$1,700.00
6000	Merlo	$0.00
7000	Trump	$600.00
8000	Brunelle	$0.00

Datasheet View

Figure 6

Select Query for Customer Balance



Design View

CustomerID	CompanyName	TotalSales	TotalReceipts	Balance
1000	Paquette	$200.00	$150.00	$50.00
2000	Rienzo	$900.00	$600.00	$300.00
3000	Ferris	$600.00	$600.00	$0.00
4000	Knipes	$100.00	$100.00	$0.00
5000	Rathay	$2,000.00	$1,700.00	$300.00
6000	Merlo	$700.00	$0.00	$700.00
7000	Trump	$800.00	$600.00	$200.00
8000	Brunelle	$0.00	$0.00	$0.00

Datasheet View

query. A parameter query is one that prompts the user for information when the query is run. In this case we would want to create a query that prompts the user to enter a value for the CustomerID field for the customer of interest. To create the prompt, all that is required is to enter the desired text to be displayed in the prompt, in the criteria text box for the CustomerID field. This is the only change that needs to be made to the balance query that was designed in the prior section. The text of the prompt must be enclosed in brackets as shown in the design view of Figure 7.

See Figure 7

When the query is run, the parameter value dialog box shown in Figure 7 will prompt you for the CustomerID. To see the balance for customer 2000 enter that value in the dialog box and the resulting dynaset would appear as shown in Figure 7. An alternate approach to generating this information is presented in the section that follows.

Using a Calculated Control to Calculate an A/R Balance

At an early stage in the sales/cash receipts cycle it is necessary to determine that a customer has sufficient credit available when an order is being placed. This can be accomplished by comparing the customer's credit limit to their existing A/R balance. Assume that the abridged order form shown in figure 8 is used to gather information for

See Figure 8

recording information in the Sales table. The process of completing the form would be as follows. Upon entering a value for CustomerId the corresponding values for the Company Name and Credit Limit fields would appear as well as the two calculated values for the Current Customer Balance and Available Credit. As in the prior sections our focus has been on the calculation of a customer's balance when this value must be calculated rather than being readily available as in a traditional approach. To find the current balance for the customer placing the order the DSum function is used in a calculated control on the form. In this particular example the expression to be entered in the text box for Current Customer Balance while in design mode is as follows:

=(DSum("SaleAmt", "tblSales", "CustomerId = """ & [Forms]![frmOrderEntry]![CustomerId] & """")-DSum("PaymentAmt", "tblCashReceipts", "CustomerId= """ & [Forms]![frmOrderEntry]![CustomerId] & """"))

The Dsum function has two required arguments. The first is an expression that identifies the field whose values are to be totaled. I.e. the SaleAmt field in the table called tblSales and the PaymentAmt field in the table called tblCashReceipts. The second is a string expression identifying the set of records that constitutes the domain. I.e. the sum to be calculated is restricted to the customer identified by CustomerId on the form titled frmOrderEntry.

The following expression is contained in the text box for Available Credit:

=[Forms]![frmOrderEntry]![CreditLimit]-Forms]![frmOrderEntry]![CurrentBalance]

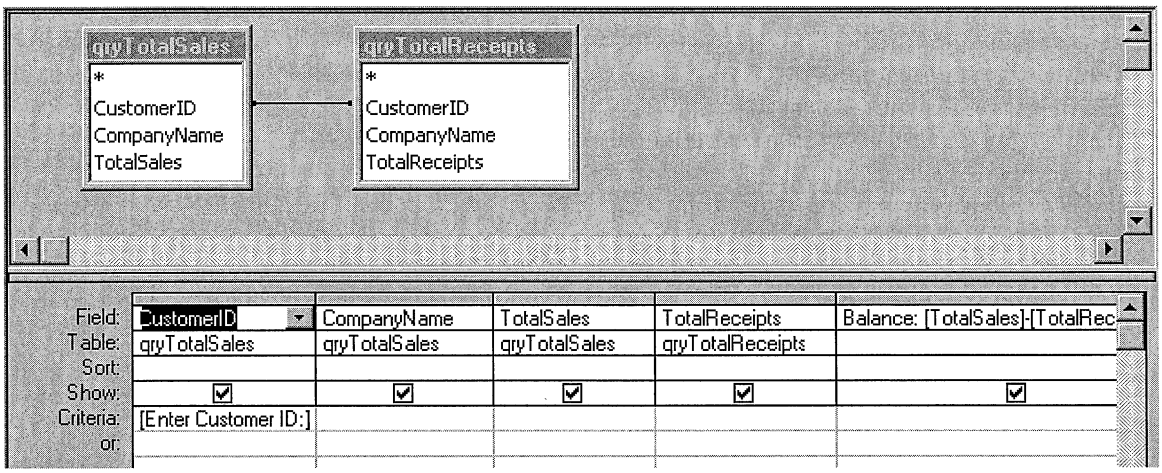
This calculated control simply returns a value for the expression that subtracts a customer's current balance from their credit limit.

Summary

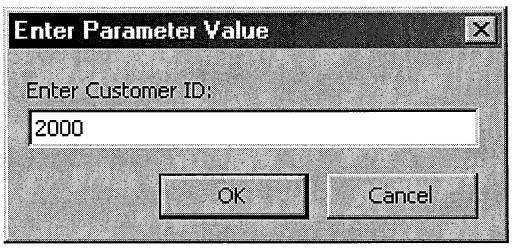
This paper focused on the underlying information processes associated with a relational implementation of the sales/collection cycle. One objective of this paper was to provide future information customers with an exposure to the use of query tools provided by relational databases. The relational and traditional approaches of tracking accounts receivable was compared and

Figure 7

Parameter Query for Customer Balance



Design View



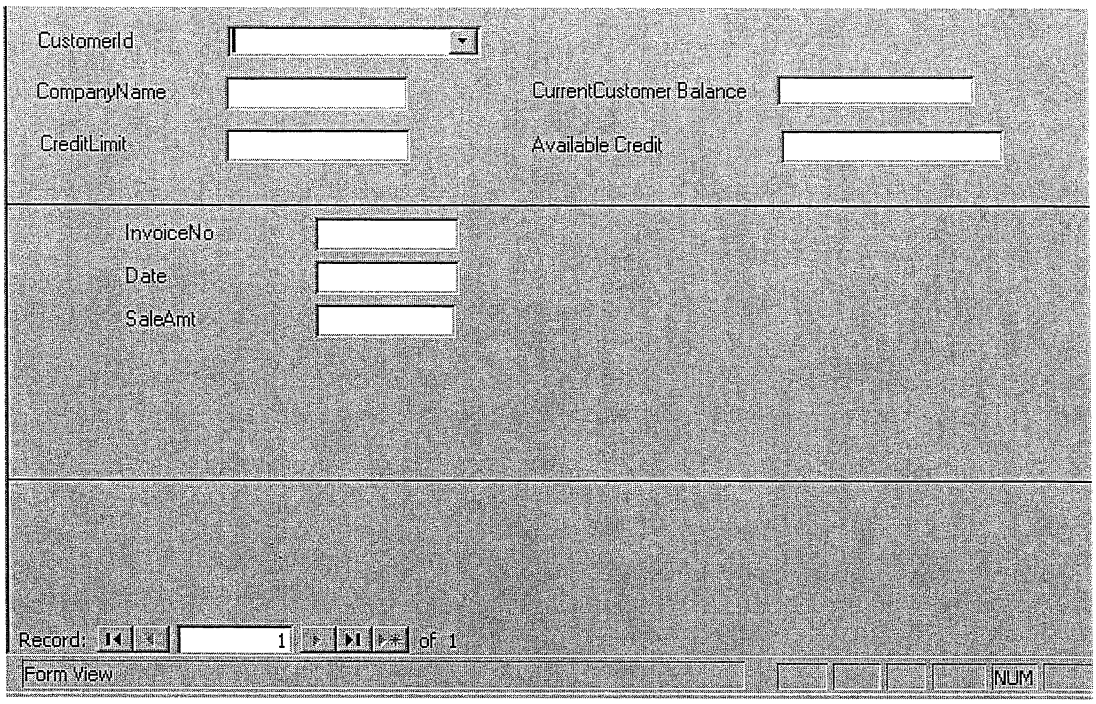
Parameter Value Dialog Box

Customer ID	CompanyName	TotalSales	TotalReceipts	Balance
2000	Rienzo	$900.00	$600.00	$300.00

Result Of Parameter Query

Figure 8

Order Entry Form



some difficulties that might be encountered while trying to implement the REAL modeling approach in an event-driven system were presented. Deriving balances for Accounts Receivable for all customers, as well as a specific customer, was illustrated using Access. Access topics addressed include total queries, inner and left outer joins, Immediate IF, IsNull, Nz, and DSum functions. Because of the limited amount of instructional material that is available on this topic, an exercise has been excerpted from the paper for use as an assignment, and is included in the Appendix that follows.

Appendix

The objective of this assignment is to provide the information user with an opportunity to implement the concepts of the REAL approach. Assume that one would like to generate information on customer balances given that a relational database approach is used to maintain the sales and cash receipt information and that no subsidiary tables are maintained in an effort to eliminate redundant recording of transaction information. The abridged information that you have to work with is contained in the three tables given in Figure 1.

Using the information in the tables given in Figure 1, prepare an accounts receivable subsidiary ledger that one might find in a traditional manual accounting system.

Show the relations that exist between the tables and identify primary and foreign keys.

Using Access prepare a query that will find the total sales made to each customer. Be sure that the resulting dynaset includes those

customers to whom no sales were made during the period, in that this would be valuable information to management. Hint: Consider using a left outer join with the Customer table. Also be sure to identify and resolve resulting null values. Consider the use of the Nz or the IIf and Is Null functions.

Using Access prepare a query that will find the total payments made by each customer. Be sure that the resulting dynaset includes those customers from whom no payments were received during the period, in that this would be valuable information to management. Hint: Same as part c.

Prepare a query that utilizes the results of the two previous queries to find the outstanding balance for each customer. The resulting dynaset should correspond to the results obtained in Part a. Use the query as input to an Accounts Receivable report.

Design a parameter query that will provide information on the total sales, total cash receipts and the account receivable balance for a particular customer. Hint: All that is required here is a modification of the query developed in Part e.

Design a sales order form that will capture information to be added to the Sales table. As part of the order process the customer attributes, credit limit, current account receivable balance, and available credit should be made available on the form. Hint: The DSum function may prove helpful as a calculated control on the form.

Note to the instructor: Attach Figure 1 from the text of the article as part of the exercise.

References

1. Denna, Eric L., J. Owen Cherrington, David P. Andros, and Anita S. Hollander, *Event-Driven Business Solutions*, Business One Irwin, Homewood, IL, 1993.
2. Hall, James A. "REA: An Evolutionary Approach," *The Review of Accounting Information Systems*. Vol. 2, No. 3. pp. 1-10, 1998.
3. Hernandez, Michael J., *Database Design for Mere Mortals*, Addison-Wesley, Reading, MA, 1997.
4. Hollander, Anita S., Eric L. Denna, and J. Owen Cherrington, *Accounting, Information Technology, and Business Solutions*, Irwin McGraw-Hill, Chicago, Illinois, 2000.
5. Levitan, Alan S. "Deriving Accounts Receivable In An Events-Based, Relational Database System," *The Review of Accounting Information Systems*. Vol. 3, No. 3, pp.47-52, 1999.
6. Paquette, Laurence R. "Database Updates in Access,". *The Review of Accounting Information Systems*. Vol. 2, No. 4, pp.45-50, 1998.
7. Paquette, Laurence R. and Saeed Roohani, "Dynamic Databases: Real Time and Batch Updates," *Proceedings of the Northeast Regional Decision Sciences Institute Meeting*, (Spring) 1998.
8. Walker, Kenton B. and Eric L. Denna, "Arrivederci, Pacioli? A New Accounting System Is Emerging," *Management Accounting*. (July) pp.22-30, 1997.