A Reusable Database Engine for Accounting Arithmetic

The author is Principal Scientist, CEC Services, LLC, 1613 Morning Drive, Loveland, CO 80538-4410, cjames@www.cec-services.com, 970.622.0466 (970.622.0177 facsimile).

Abstract

Report Accounts [RA] is a reusable, core, database engine for accounting arithmetic.

Recent advances in business information systems are:

1. How to map the attributes, objects, and classes of accounting arithmetic directly into relational columns, rows, and tables.

2. How to encode user-defined transaction logic into a single table; and

3. How to coerce nonprocedural SQL to perform the procedural processes of accounting arithmetic.

Introduction

This paper is divided into seven progressive parts: 1. Requirements; 2. Analysis; 3. Design; 4. Implementation; 5. Testing; and 6. Maintenance.

1. Requirements

Requirements must be present in the problem domain in order for it to be decomposed effectively. Problem domain requirements include completeness, reliability, scalability, performance, and methodology.

1.1. Completeness

The requirement of completeness means that the problem domain must contain complete information for the assessment of the expert field, which here is accounting arithmetic.

Traditional accounting uses double-entry bookkeeping with the atomic parts of debits [dr] and credits [cr]. Momentum accounting of Ijiri 1989 uses triple-entry bookkeeping with the additional atomic part of trebits [tr]. This allows for time-rate of change measurements that produce momenta (dr), impulse (cr), wealth (dr), income (cr), action (tr), and a three-dimensional structure of accounting. Derivatives of these measurements produce aggregation matrix reports that benefit management. The business value of momentum accounting is that while traditional accounting reports would show a business to be solvent, momentum accounting reports could show the business to be insolvent, and exactly where the business is insolvent. (The author's company obtained permission to implement momentum accounting per Polhemus 1995.)

According to generally accepted accounting principles [GAAP] of AICPA 1994, a valid accounting system must also not modify any prior transaction and must afford a complete audit trail for accountability.

1.2. Reliability

The requirement of reliability means that the problem domain must include enough information to assess security and stability.

Security includes the encryption of data records and the integrity of data records obtained from across a multi-user, networked environment. Data encryption is addressed by the non-invertive random number generator of James 1993. Data integrity is addressed by checksums such as CRC-CCITT and transaction commit heuristics of an industrial strength, backend database platform.

1.3. Scalability

The requirement of scalability means that the problem domain must contain information that is capable of describing the largest size of such a core accounting system as RA.

For example, consider accounting for High Definition Television [HDTV] as a potential subset of the problem domain. The maximum number of potential users in a perfect world would be the population of about 4.3 billion subjects. Given a set-top box for each such subject and with 256 separately addressable central processing units [CPUs] per box, RA would have to accommodate a fine enough time slice for record commits to differentiate between all CPUs committing at about the same instant in time. Furthermore, the core database for accounting arithmetic would have to accommodate an arbitrarily large enough number of separate accounts such as 16,384. Given such scalability, it is estimated that the accounting database needs for IBM Corporation could be processed and addressed from a single such set-top box.

1.4. Performance

The requirement of performance means that the problem domain must provide information as to the acceptable speed and throughput to be deemed useful and of strategic business value.

A standard real-time benchmark in the business community is that more than three-seconds for a complete computer transaction to commit is not acceptable. That less complex systems usually perform faster implies that RA necessarily should be kept less complex for the best possible performance. Another way to ensure high performance in database throughput is to make use of recent advances in sorting and indexing, such as that available from James 1995.

1.5. Methodology

The requirement of methodology means that the problem domain must contain enough information to be assessed in an abstract and logical notation capable of describing the entire life cycle of the development effort.

Of the 100 methodologies in use today, the only one based on third-order predicate logic is Business Object Notation [BON] of Walden/Nerson 1995.

2. Analysis

For the accounts, logic, and transactions subsystem of RA, the analysis in BON diagrams appears in Figure 1. Attributes of objects have indented labels in italic font. Objects have outdented labels in normal font. Classes of objects are ellipses with centered labels in bold font. Arrows show inheritance and point from the child class to the parent class. Constraints or business rules are not listed, but apply to single or multiple objects at the same level as do attributes.

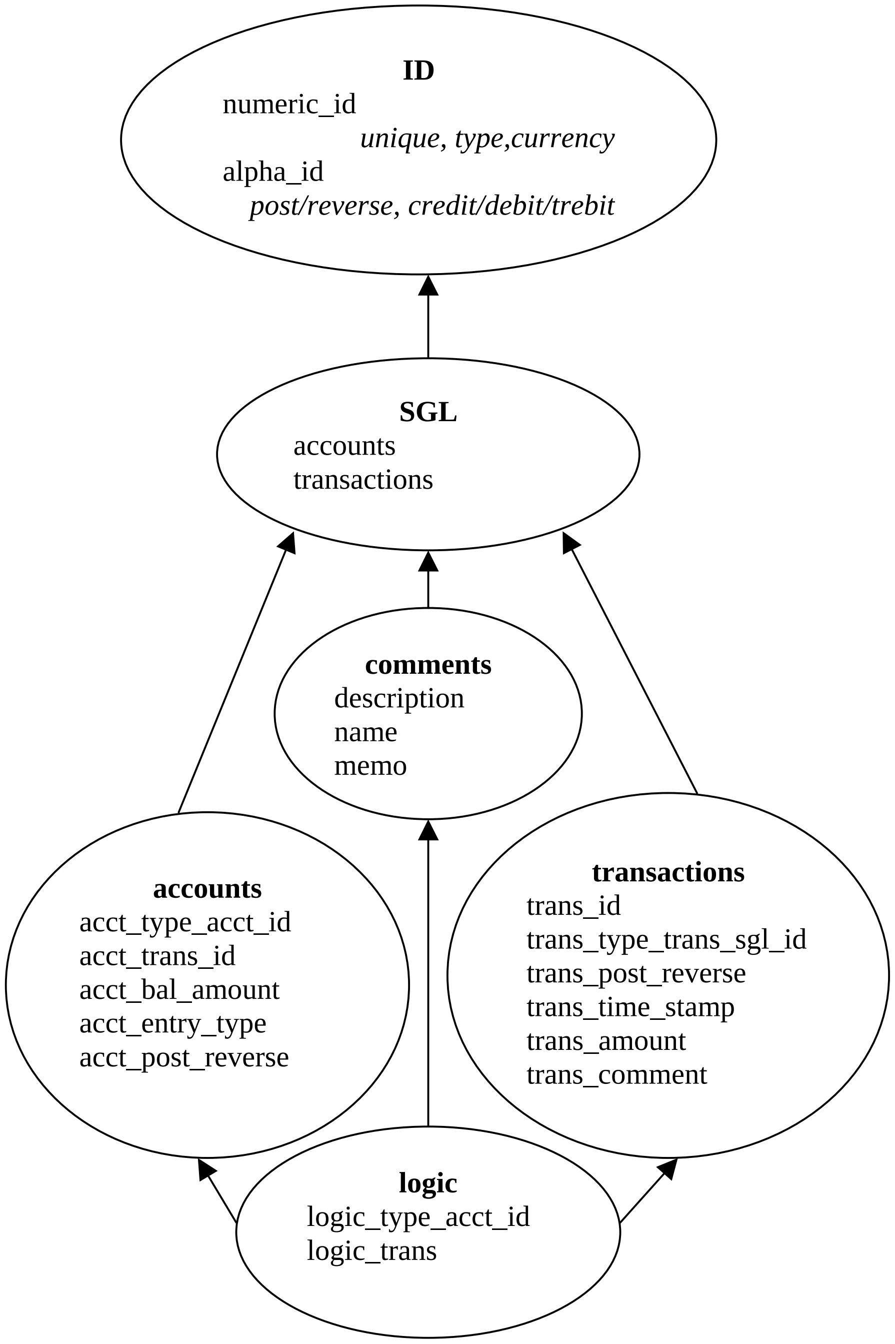

Figure 1: BON Diagrams

3. Design

The design of RA was undertaken using relational database tables, resulting in eight tables and 26 columns.

A recent advance in business information systems was the mapping of BON diagrams directly into relational database tables on a one-to-one basis. The classes for transactions, logic, and accounts in Figure 1 map to and match the same respective tables in Figure 2.

transactions

TRANS_ID		LONGINT AUTONUM
TRANS_TYPE_TRANS_SGL_ID	(0..9999 trans)	INT
TRANS_POST_REVERSE	(Post/Reverse)	TEXT(1)
TRANS_TIME_STAMP		DATE
TRANS_AMOUNT		CURRENCY
TRANS_COMMENT		TEXT(255)

logic

LOGIC_TYPE_ACCT_ID	(0..9999 accounts)	INT
LOGIC_TRANS		MEMO(9999)

accounts

ACCT_TYPE_ACCT_ID	(0..9999 accounts)	INT
ACCT_TRANS_ID		LONGINT
ACCT_BAL_AMOUNT		CURRENCY
ACCT_ENTRY_TYPE	(Cr/Dr/Tr)	TEXT(1)
ACCT_POST_REVERSE	(Post/Reverse)	TEXT(1)

Figure 2: RA Relational Table Subsystem

4. Implementation

Other recent advances in business information systems were discovered during the development of RA. They are table-loaded logic and using SQL92 as a procedural language.

Certain security factors were discussed above in section 1.2. Reliability. In the initial design, the assumption was that the level of security, including data encryption and data integrity across a network, was to be handled externally by the database and its environment. However, those factors are easily implemented for tertiary security levels by including these respective columns in each table: commit switch; type of encryption; and checksum value.

4.1. Table-loaded logic

The logic for the accounting arithmetic is stored in the logic table in Figure 2. There are arbitrarily 16,384-transaction types and 16,384-account types. A given transaction type operates on from 2- to 34-accounts, as specified by accounts in the U.S. Government Standard General Ledger [SGL 1996]. Exactly which accounts are updated for a given transaction is also contained there. For example:

4135. Receipt of other cash. [*transaction 4135*]

Proprietary Accounts

Dr.	1190	Other cash [*debit account 1190*]	
	Cr.	2990	Other Liabilities [*credit account 2990*]

The SGL 1996 specifies accounts (and transactions) in the range of 1000 to 9999. RA arbitrarily has designated transactions in the same ranges. RA also designates transactions in the range of 1 to 99 for internal use and maintenance of tables such as updating and deleting logic, type of transaction, type of account, and the respectively affected balances. RA designates reports in the transaction range of 100 through 999, that is 900 transactions for reporting use.

The process flow of RA is as follows. The user inputs a type of transaction, monetary amount, and post / reverse switch to the transactions table of Figure 2. RA looks up the transaction type in the logic table to determine which accounts are to be updated and by which process of cr, dr, or tr. Those respective account numbers are: looked up in the balances table for the current balance; inserted into the accounts table with the updated, new current balance; and updated in the balances table with the new running balance. Upon completion of the respective account number transactions; the time stamp on the original transaction is updated and set as a commit switch. Another purpose of the balances table is for fast reporting because it is easier to look up an indexed, unique account number for the current running balance than to search the accounts table for the most recent entry.

For internal RA transactions such as updating descriptions of the accounts in the type of account table and updating descriptions of the transactions in the type of transaction table the same process flow above is used except that the balance accessed always renders a zero balance. Similarly, for external RA reporting transactions the same process flow is used.

Complete audibility, accountability, and rollback is also available due to the design and implementation of the accounting system of RA.

4.2. SQL92 as a procedural language

To effect the table-loaded logic above, what was originally implemented was a look up table of T-transaction rows, each row containing switches for A-account columns, ie, a T-by-A table.

The practical problem was the sheer number of A-account columns needed, totaling 16,384. Most vendors of relational databases do not support that many columns. The A-columns could be collapsed into one column of bit-switches, internally grouped by 2-bits per switch for the 4-states of interest of cr, dr, tr, and none. Such a column would become a string of 32,768-bits extending over 4,096-bytes, the length of which most vendors would allow. But testing the respective bit groups had the additional problem of doing bit manipulation by progressing through the bits which is computationally more intensive than searching for switches as bytes. In other words, while the Oracle SUBSTR command works on a single query basis, SUBSTR is not practical in a series of 16,384 single SELECT statements.

The problem was submitted to the generalization mode of thinking learned from using BON for object-oriented analysis. The resulting abstract solution was to swap the row and column designation of the T-by-A table into an A-by-T table of A-account rows and T-transaction columns. Thus the SUBSTR command could be used on a string of T-transactions to retrieve only those A-account rows having the switch states of interest.

This solution made use of the obvious fact that the querying power of SQL92 is in returning rows and not columns which is sometimes contrary to the way one would ordinarily view matrices or tables. Hence the A-by-T table has only one column which is constrained in such a way as to produce quickly a number of rows, rather than the unwieldy T-by-A table which would have effectively required the A-columns to be queried sequentially by many separate queries. The logic table of RA thus effectively coerces non procedural SQL92 to perform procedural tasks and thereby avoids embedding SQL in procedural languages.

RA uses three sets of compact SQL code for updating accounts, balances, and transactions. The SQL code totals less than 50-lines. (A line of SQL code is defined as beginning with a SQL keyword such as AND, AS, and IN.)

5. Testing

Because RA contains less than 50 lines of SQL code, test suites were relatively simple with a total of 28 tests as follows. The code set for updating accounts code had nine input boundary tests and 15 intermediate and final query tests. The code set for updating balances had two tests. The code set for updating transactions had two tests. Because the test suite was modest, it was not automated, and regression testing was performed by hand.

6. Maintenance

Due to SQL92, RA code is portable and extensible to more platforms than other accounting database engines.

Due to the logic table design, it is easy to modify RA for accounting transactions and reporting functions. It is also easy to adapt RA for other business purposes such as inventory where the inventory data is contained in multiple accounts.

Conclusion

This paper shows the nearly automatic process of exactly: how an object-oriented methodology BON was used to analyze the problem domain and how the resulting object-oriented design was mapped directly on a one-to-one basis into relational database tables.

Recent advances in automated software engineering are: how to load accounting transaction logic directly into a relational database table; and how to access and process that logic in an automatic and sequential manner from within the non procedural language of SQL92 and without embedding it in a procedural language.

Lessons learned were: how important automatic documentation is for the development environment and the end user production environment; and how necessary some integrated reporting tool is to business rules automation and the business rules repository.

More research and practical experience needs to be concentrated on the significant process of automating object-oriented analysis with relational database table design and business rules implementation. It is through such study that the once disparate fields of object-oriented analysis and business rules automation can be reconciled and unified into a generic process and useful engineering tools which are seamless and reversible.

Acknowledgments

This acknowledges with gratitude helpful comments of and discussions with: Brian Campbell, CPA, Campbell, Irvine and Rosenbaum; and Yuji Ijiri, Robert M Trueblood University Professor of Accounting and Economics, Carnegie Mellon University.

References

AICPA 1994

American Institute of Certified Public Accountants, Inc. Improving Business Reporting – A Customer Focus. Comprehensive Report of the Special Committee on Financial Reporting. 1994.

Ijiri 1989

Yuji Ijiri. Momentum Accounting and Triple-Entry Bookkeeping: Exploring the Dynamic Structure of Accounting Measurements. 1989. American Accounting Association -- Studies in Accounting Research #31.

James 1993

Colin James III. United States Patent Number 5, 251,165. Two Phase Random Number Generator. October 5, 1993.

James 1995

Colin James III. United States Patent Pending. The Binary Sort Access Method [BSAM].

Polhemus 1995

Craig E Polhemus, Executive Director, American Accounting Association. Letter, December 6, 1995.

SGL 1996

Financial Management Service [FMS]. Department of the Treasury. U.S. Government Standard General Ledger, 6/01/89; Attachment 1, 5/3/94; and Section III. Accounting Transactions, 7/21/86.

Walden/Nerson 1995

Kim Walden and Jean-Marc Nerson. Seamless Object-Oriented Software Architecture: Analysis and Design of Reliable Systems. 1995. Prentice Hall UK.