ENHANCING DATABASE INTEGRITY AND PROCESS AUTOMATION THROUGH SMART TRIGGERS

David H. Olsen
Department of Business Information Systems
Utah State University
3515 Old Main Hill
Logan, Utah 84322-3515
435-797-2349
E-mail: dolsen@cc.usu.edu

Olga Yatsenko
Utah State University
Department of Business Information Systems
3515 Old Main Hill
Logan, Utah 84322-3515
E-mail: olga@cc.usu.edu

Nicole Forsgren Meek
Department of Business Information Systems
3515 Old Main Hill
Logan, Utah 84322-3515
E-mail: nicoleforsgren@hotmail.com

ABSTRACT

In this paper, we theorize that integrating database triggers and constraints with specialized extensive analysis and design techniques will yield business systems with higher levels of integrity. In turn, these improved systems will allow accounting auditors to rely more heavily on such systems, which will improve audit quality and decrease costs.

We build a database prototype for a sample accounting system that includes the extensive use of constraints and database triggers to illustrate how these techniques facilitate higher levels of integrity. We also present an analysis and design technique named the TRAC method specifically designed for trigger and constraint design because trigger and constraint management continues to be a thorny issue. It has become so problematic that many database developers advise against using triggers because "unfettered triggers quickly make a database system less maintainable." Some purists state that the data should be maintained in a simple format and that adding triggers violates the notion that database systems should be loosely coupled.

We maintain that a standard online transaction processing system (OLTP) database that is used in typical business situations is well-suited for constraints and triggers provided that the constraints and triggers are well-documented. We maintain that any poorly documented system is difficult to use in the development environment. Finally, we make no claims of the usefulness

of triggers and constraints for data warehousing applications because the list of needs is very different.

Keywords: Data model, accounting information systems, business rules, database, trigger, constraint

I. INTRODUCTION

Data modeling is one of the most critical tasks in building an information system. Pair a well-developed and solid database design with extensive but appropriate integrated database triggers and constraints and the end result will be a database implementation that consistently yields accurate, timely information. Additionally, the database and its underlying data will realize greater data integrity with the reliance on external - and sometimes faulty - applications minimized, which will in turn contribute to higher audit quality and decreased business costs.

In this paper we present a database prototype for a sample accounting information system that makes extensive use of constraints and database triggers to contribute to the system's efficiency and data integrity. Though we use the SQL Server 2000 database system for this prototype, the principles are applicable for any database management systems (DBMS).

We also present the TRAC method which is an analysis and design technique well-suited for constraint and trigger design along with a defense of business rule implementation in the database itself. This paper follows with a literature review of business rules in Section II while Section III addresses the business rules presented in this example. The database structure and its business rule implementation are discussed in Section IV. Consequences of database constraint/trigger overuse or misuse are discussed in Section V. Section VI concludes the paper and provides direction for future research.

II. LITERATURE REVIEW

Business rules are defined as "a statement that defines or constraints some aspect of the business… that cannot be broken down or decomposed further." (Von Halle,1996). While formally-defined and program or database enforced business were not recognized in system analysis and design prior to the 1980s, there have been significant contributions in the past 20 years. Some alternative current data modeling approaches include the E/R model approach (Chen, 1976), semantic modeling approaches, and UML, but this paper will focus on the REA model, which is a specialized, accounting data modeling paradigm.

The four categories of business rules as specified in (Von Halle, 1996) are: 1) definitions, 2) facts, 3) constraints, and 4) derivations. Definitions are business rules that define entities and attributes, facts are relationships between entities, constraints are conditions about the data that must always be true, and derivations are rules that create new information from other information (which are often mathematical in nature). All four types of business rules are included in the prototype system presented here. The business rules should be a fundamental part of the database system stem and its many related applications – regardless of human resource changes or application development shortfalls.

The importance of implementing business rules from within the database structure itself has become increasingly apparent with the recent rise in popularity of object-oriented design and the massive re-engineering/restructuring that is taking place in the world today (Ross, 1999). Object orientation's focus on functional requirements has increased the need for data (process) collection that isn't captured in the typical database ERD. The massive reprogramming done to many applications to employ the 'latest and greatest' GUI and Internet technologies greatly increases the chances of business rules that were buried in the old code to become forgotten or lost.

In our example, as with most accounting information systems, business rules truly find their niche. While some data models and business situations do not call for a complex set of business rules built into the underlying databases, business rules are suited perfectly to applications that (among other things) have: many mathematical calculations, many constraints and logical choices and rules that must be shared across applications (Von Halle, 1997).

III. PAYROLL PROTYPE BUSINESS RULES EXPLAINED

This prototype system illustrates using active database implementing business rules. Figure 1 in Appendix A illustrates the ER diagram of the database structure. The company uses the following business rules:

- The company employs full time and part time workers. Full time employees may be paid a set salary or an hourly rate depending on the number of hours worked. If an hourly employee worked more than 40 hours a week, the employee is paid a special overtime rate for the extra hours.
- All employees are divided into three categories depending on their status for tax purposes. A person may be single, head of household or married filling jointly. A single person does not have any dependents. Head of household is an unmarried person who has one or more dependents. A married filling jointly person may or may not have any dependents. According to this status size and number of exemptions from taxable income are calculated.
- Federal tax deductions equal 15% (payment – exemptions).
- For a single person the exemption is $310 [($4550 standard deduction+ $2900 personal deduction)/24 periods].
- For head of household – $398 [($6650+$2900)/24] plus $120 ($2900/24) for each dependent.
- For married filing jointly – $558 [($7600+$2900*2)/24] plus $120 for each dependent.
- State tax is calculated as 5% of payment amount for any type of employees.
- All basic personal information is entered when a person is hired. At that time a choice is made what type of an employee the person is and what his or her status for tax is.
- For all full time employees two special deductions are calculated: contribution to 401K fund and insurance premium. Amount of the contribution is 5% of the amount earned. Insurance premium is determined as 50$ for each employee plus $25 for a spouse and each dependent.
- Payments are made twice a month, on the 1st and the 15th, equaling 24 payment periods a year.

IV. DATABASE STRUCTURE AND ITS

Several different constraints are placed on the Employees table with some of the many constraints diagrammed in Figure 2 and listed in Appendix B

- PhoneNumber constraint enforces a standard for the phone number format as (***) ***-****.
- SSNConstraint sets ***-**-**** as the format for entering social security number.
- TaxStatusConstraint ensures that only "S" for single, "HH" for head of household, or "MFJ" for married filling jointly is entered into StatusForTax column of the table.
- WorkStatusConstraint lets only "PTH" for part time hourly, "FTH" for full time hourly, or "FTS" for full time salary to be entered into WorkStatus column.

Several different triggers are used in this database and one that is attached to the employees table is listed in Appendix B. Because of space limitations, most of the triggers and constraints are not listed in the appendix but the authors would glad provide a listing on request.

- When a new person is entered into Employees table WorkStatusCheckHourly trigger makes sure that if the employee is entered as hourly his or her annual salary is set for NULL.
- WorkStatusCheckSalary sets NULL values for regular and overtime hourly rates for salary employees.
- TaxStatusCheckHH ensures that at least one dependent is included if the employee is marked as head of household.
- TaxStatusCheckS sets NULL value for the number of dependents if the employee is recorded as single for tax purposes.

As an example for two different ways to achieve the same result, TaxStatusCheck trigger is included into Employees table because this trigger accomplishes the same result as TaxStatusConstraint.

Each payment day SSN of each employee and date are entered into Payroll Hourly table if the employee is paid by hour, or into Payroll table if the employee has a salary. For hourly employees regular and overtime hours worked are also entered into Payroll Hourly table.

When information is inserted into Payroll Hourly table two triggers fire. One of them, HourlyCheck verifies that the SSN belongs to an hourly employee. The other one, PayCalc, calculates regular and overtime payments by multiplying the rates from Employees table and the number regular and overtime hours worked during the period. Also the second trigger inserts SSN and date into Payroll table. If the number of hours worked changes another trigger PayCalcUpdate fires. It calculated the new amount for regular and overtime payments and also updates Payroll table changing the total amount earned.

When a tuple is entered into Payroll table PaymentInsert trigger calculates amount earned as 1/24 of the annual salary amount from Employees table for salary employees or the sum of regular and overtime payments from Payroll Hourly table for hourly employees. It also finds the amount of federal and state income taxes according to the company business rules. If the employee works full time the amount of special deductions is calculated and inserted into Deductions table. Also this trigger determines total net payment amount by subtracting tax and special deductions from amount earned.

If any information in Payroll table is changed PaymentUpdate trigger updates all calculated data in Payroll and Deductions tables.

V. IMPORTANT DATABASE CONSIDERATIONS

While the implementation of business rules via database constraints and triggers is very powerful and beneficial, many database performance issues must be considered and are outlined in this section.

It is important to perform all referential integrity checks and data validations using constraints (foreign key and check constraints) rather than triggers, as they are faster. Triggers should be limited to tasks such as auditing, validations and custom tasks that cannot be performed using constraints. Constraints save time as well, as the code for these validations is usually much more simple which allows the DBMS to do all the work.

It is important to always access tables consistently in the same order in stored procedures and triggers. This helps in avoiding deadlocks. Other practices to avoid deadlocks include: keeping transactions as short as possible, touch as little data as possible during transactions, and never wait for user input in the middle of a transaction. Additionally, do not use higher level locking hints or restrictive isolation levels unless absolutely necessary and make the front-end applications deadlock-intelligent. That is, applications should be able to resubmit transactions if a prior transaction fails with an error 1205. It is also wise to process all the results returned by SQL Server immediately so that the locks on the processed rows are released.

To save processor time, function calls should not be made repeatedly within stored procedures, triggers, functions and batches. For example, the length of a string variable might be needed in many places in a procedure, but the LEN function should not be used whenever it's needed. For efficiency, call the LEN function once and store the result in a variable for later use.

Referential integrity should be used carefully. There are several pitfalls related to using all the of referential integrity (RI) bells and whistles within a given database management system (DBMS); always know the performance tradeoffs with RI. While foreign key constraints help data integrity, they have an associated cost on all insert, update and delete statements. Careful attention should be given regarding the use of constraints in a warehouse or an object database system (ODS) when one wishes to ensure data integrity and validation. While triggers are extremely powerful in standard database systems, they may slow mass inserts into very large databases (VLDB) considerably, as every row that is inserted will fire its corresponding trigger.

VI. CONCLUSION

Two schools of thought exist regarding implementing business rules in an information system. One would be a minimalist approach which would include few if any constraints and no triggers but would implement all the business logic in application programs. We subscribe to the second school of thought which implements as much business logic as possible using constraints and triggers as this approach leads to higher levels of data integrity and mandates that certain actions be taken when specific conditions exist. To this end, we developed the visual TRAC method for specifying in the design stage what tables are affected by various triggers and constraints. The TRAC method should help manage the myriad of constraints and triggers that are frequently developed.

Appendix A

Entity Relationship Diagram

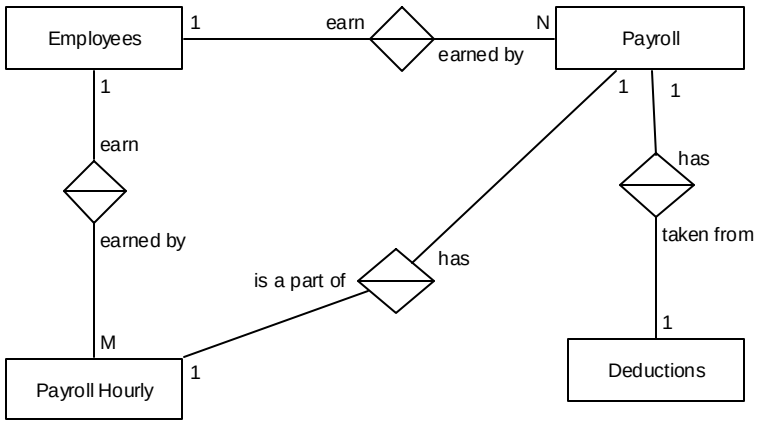

Figure 1

TRAC Diagram

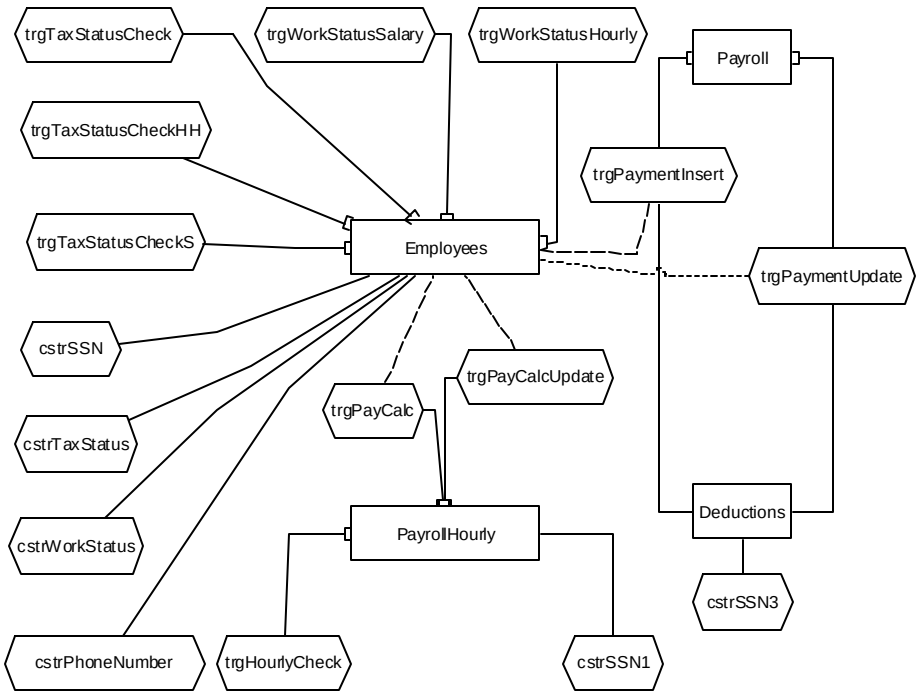

Figure 2

Appendix B

Trigger and constraints for the Employees table:

```
ALTER TABLE [dbo].[Employees] WITH NOCHECK ADD
	CONSTRAINT [DF_Employees_Dependents] DEFAULT (0) FOR [Dependents],
	CONSTRAINT [PK__EMPLOYEES__1B0907CE] PRIMARY KEY  CLUSTERED
	(
		[SSN]
	)  ON [PRIMARY] ,
	CONSTRAINT [cstrPhoneNumber] CHECK ([HomePhone] like '([0-9][0-9][0-9]) [0-9][0-9][0-9]-
[0-9][0-9][0-9][0-9]'),
	CONSTRAINT [cstrSSN] CHECK ([SSN] like '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),
	CONSTRAINT [cstrTaxStatus] CHECK ([StatusForTax] = 'S' or ([StatusForTax] = 'HH' or
[StatusForTax] = 'MFJ')),
	CONSTRAINT [cstrWorkStatus] CHECK ([WorkStatus] = 'PTH' or ([WorkStatus] = 'FTH' or
[WorkStatus] = 'FTS'))
GO


CREATE TRIGGER trgTaxStatusCheck ON dbo.Employees

FOR INSERT, UPDATE
AS
IF
	 (SELECT StatusForTax FROM Inserted) NOT IN ('S', 'HH', 'MFJ')
BEGIN
	RAISERROR ('Status for tax purposes must be: S- single, HH - head of household, or MFJ -
married filing jointly only',16,1)
	ROLLBACK TRAN
END

CREATE TRIGGER trgTaxStatusCheckS ON dbo.Employees
FOR INSERT, UPDATE
AS
IF 'S' =
	(SELECT StatusForTax FROM Inserted) AND
	(SELECT Dependents FROM Inserted)<>0
BEGIN
	RAISERROR ('If Status for Tax is Single the employee does not have any dependents',16,1)
	ROLLBACK TRAN
END
```

REFERENCES

Chen, P.P., "The Entity Relationship Model: Toward a Unified View of Data." *ACM Transactions on Database Systems* (January 1976): 9-36.

McCarthy, W.E.., "Resource-Event-Agent (REA) Model." *The Accounting Review* (July 1982): http://www.msu.edu/~mccarth4/McCarthy.pdf (February 2002).

Ross, R.G., "Exploring Business Rules." *Business Rule Book, Second Edition* (April 1999): http://www.brcommunity.com/cgi-local/x.pl/features/a439.html (February 2002).

Von Halle, B., Plotkin, D., "The Never-Ending Search for Knowledge" *Database Programming and Design*, 9(2): 15-18, February 1996.

Von Halle, B., "The Business Rule Roadmap" *Database Programming and Design Online*: http://www.dbpd.com/vault/9710arch.htm (October 1997).