Static and Dynamic Driver Triggers

Colin James III, Principal Scientist, CEC Services, LLC, 1613 Morning Dr, Loveland CO 80538-4410, psum@cec-services.com, Mobile 719.210.9534 , Office / Fax 970.593.1350

Keywords

Logic Table Technology, LTT, subject-oriented software, disruptive technology, "Report Accounts", RA, pSUM, "portable, Scalable, Useable, Maintainable", dynamic triggers, static triggers, stacked logic tables, folded logic tables, compressed logic tables, self-modifying logic tables, logic switches, real-time, coercing procedural processing, Structured Query Language, SQL, accounting arithmetic, accounts, transactions, type of account, type of transaction, balance, type of balance, entrys, processes, post, reverse, debit, dr, credit, cr

Abstract

Static driver triggers support accounting arithmetic engines when the number of types of accounts and types of transactions are known at the time of implementation. Static driver triggers are suited for logic tables of a known size and that are stacked, folded, compressed, and self-modifying. Dynamic driver triggers support accounting arithmetic engines when the number of types accounts and types of transactions are not known at the time of implementation. Dynamic driver triggers are suited for logic tables that grow without bound. Dynamic driver triggers require slightly more lines of code to implement than do static driver triggers.

Introduction

The RA system name means "Report Accounts" and is pronounced "Ra" after the Egyptian sun god because the initial design was made in Cairo. RA is an accounting arithmetic engine with a size designed to remain static with a constant, upper limit number of accounts. The account types are numbered from 0000 to 9999 such that numeric ranges decode to general functions. For example, system maintenance account types range from 0000 to 0999, and report account types range from 9000 to 9999. In the tables type account, accounts, and balances, the common column relating the tables is based on type of account ID as account type account ID and balance type account ID, all of data type integer.

In RA, transaction types map directly into logic switch number IDs in the range from 0000 to 9999 as follows and with the order of tables processed:

Name	ID range	Tables processed in order [*with look up tables noted*]
Logic	0000 – 0049	[type transaction], transactions, [type account], type account logic, transactions
Transaction	0050 – 0074	[type transaction], transactions, [type account logic], type transaction, type transactions
Account	0075 – 0099	[type transaction], transactions, [type account logic], type account, type transactions
Report	0100 – 0999	[type transaction], transactions, [type account logic], [type account balance], transactions
Transaction	1000 – 9999	[type transaction], transactions, [type account logic], [type account balance], accounts, type account balance, transactions

Therefore, transaction types that apply to changing logic switches have an ID in the numeric range of 0000 to 0049. Also, transaction types that apply to transactions have an ID in the numeric range of 1000 to 9999.

The pSUM system name means "portable, Scalable, Useable, Maintainable" and is pronounced "Sum" with a silent "p". pSUM is an accounting arithmetic engine with a size designed to expand dynamically to an unbound number of accounts. In pSUM, by contrast to RA, the accounts are numbered uniquely, such as with a date time combination that is system unique, and thus may go by any system dependent numeric name. In the tables type account, accounts, and balances, the common column relating the tables is based on the account type ID in the table type account. That column is renamed as such in the table accounts as account type account ID and in the table balance as balance type account ID. In IBM UDB DB2, the account type ID is implemented as the data type of time stamp in 14-bytes. The reason for using unique IDs is to avoid any counting and increment tables for the ID number. An alternative scheme is to specify the unique ID as data type of float. For example, an account type ID within the numeric range of 2000 to 2999 may have many decimal values such as 2000.0000, 2000.0001, 2000.0002, … , 2998.9998, 2998.9999, 2999.0000, with each value representing a distinct account ID number.

Static Driver Triggers

For RA, the static driver trigger has the following logic in pseudo code.

> In table accounts, insert respective values from tables logic, transactions, entrys, processes, and balances with the constraints that: the transaction time stamp is null with the logic switch as a recognized switch; the same account ID is present in the logic and balance tables; and the post / reverse switch is recognized.

In table balances, update the balance for the account ID of interest with the constraints that: the transaction time stamp is null and the same account ID is present in the transactions and accounts tables; and the same account ID is present in the account type and balances tables.

In table transactions, update the transaction time stamp from null to the current time stamp, thereby setting the switch to show the transaction is completed.

Preliminary table population requires the following.

In table entrys, insert type and sign values: for debit (dr) of d, -1 and D, -1; and for credit (cr) of c, +1 and C, +1.

In table processes, insert type and sign values: for post of p, +1 and P, +1; and for reverse of r, -1 and R, -1.

In table transactions, insert an initial sentinel record 1 as follows.

In table input buffer, insert values

From table input buffer, insert values into table transactions, to avoid the static driver trigger from firing.

In table type account, insert records for each of the account type IDs.

In table balances, insert records for each of the account type IDs as balance type account ID with a starting balance of zero.

In table logic, insert records for each of the account type IDs, as logic type account ID and logic transaction switches.

Thus, the static driver trigger(s) is pre-specified to process new records which are preloaded. Data is also preloaded as values for: table entrys; table processes; table input buffer; table transactions in an initial sentinel record; table type account; table balances; and table logic.

Dynamic Driver Triggers

The constraints for dynamic driver triggers may be abstracted so that general rules become apparent. The rules are that at least one logic switch must exist for each account type ID. If all or some account type IDs share the same single logic switch, then only one logic switch applies to those account type IDs. This means that the same logic transaction column value would be repeated for each row of account type ID. The actual value contents of an account type ID therefore becomes irrelevant so long as the same

account type ID is referenced by the balance account type ID, account type account ID, and logic account type ID. Only one balance amount and one logic string is associated with the same account type ID. Therefore, the table balances and the table logic may be combined with the table account type.

In the trigger to update an account, a constraint is that the account type ID references a logic string and a balance amount. An alternative to checking for the existence of such a account type ID is checking for the nonexistence of such a account type ID. If the account type ID does not exist, then the trigger is to insert dynamically such a account type ID in the account type table. Consistent with a unique account type ID is also a unique social security number (SSN) associated with it. Hence a test of the non-existence of a account type ID is equivalent to the test of the non-existence of a specific SSN. Therefore a better test plan is for the user to supply a SSN which if not existing causes the insertion of the respective columns tables for a new account type ID in table account type. All that is required to insert such a row is the SSN. This logic in pseudo code appears below.

> In table account type, insert a new row with populated columns for account type ID as a time stamp, balance amount as zero, SSN as supplied by the user, and logic string, usually the most recent used, where there does not exist a known account type SSN equal to the account type SSN to be inserted.

As implemented in SQL code, the table account type requires only one insert statement and test. In the sample SQL code below, the semi-colon is the termination character used internally within the section of begin-atomic-end to separate command blocks. Because the entire trigger command requires a termination character different from the semi-colon, the at-symbol "@" was arbitrarily chosen.

```
CREATE TRIGGER DB2ADMIN.tran_acct_bal
AFTER INSERT ON TRANSACTIONS
FOR EACH ROW MODE DB2SQL
BEGIN ATOMIC

    INSERT INTO
        type_account
            (type_acct_id,
            type_acct_logic,
            type_acct_ssn,
            type_acct_bal_amt)
        SELECT
            current timestamp,
            MAX( Instant.type_acct_logic),
            SUM( type_acct_ssn),
            SUM( Instant.type_acct_bal_amt)
        FROM
            type_account AS Instant
```

```
            WHERE NOT EXISTS
                ( SELECT
                    Old.type_acct_ssn
                FROM
                    type_account as Old
                WHERE
                    Old.type_acct_ssn = Instant.type_acct_ssn) ;

INSERT INTO
    accounts
        (acct_type_acct_id,
        acct_trans_id,
        acct_entry_type,
        acct_post_reverse,
        acct_bal_amount)
    SELECT
        type_acct_id,
        trans_id,
        E.proc_entry_type,
        trans_post_reverse,
        type_acct_bal_amt + (trans_amount  * E.proc_sign * P.proc_sign)
    FROM
        type_account,
        transactions,
        process as E,
        process as P
    WHERE
        ( trans_time_stamp IS NULL
        AND
        Substr( type_acct_logic, trans_type_trans_id, 1) = E.proc_entry_type )
    AND
        trans_post_reverse = P.proc_type  ;

UPDATE
    type_account
    SET
        ( type_acct_bal_amt)   =
        ( SELECT
            acct_bal_amount
        FROM
            accounts,
            transactions,
            type_account
        WHERE
            ( trans_time_stamp IS NULL
            AND
```

```
                accounts.acct_trans_id = transactions.trans_id )
            AND
                type_account.type_acct_id = accounts.acct_type_acct_id)  ;

    UPDATE
        transactions
        SET
            trans_time_stamp  =  CURRENT TIMESTAMP
            WHERE
                trans_time_stamp IS NULL ;

END @
```

If the account type ID exists already in the table account type, then the constraint is true that an account type SSN exists. The trigger then proceeds to insert a new row in the table accounts, update the balance amount in the table account type, and update the transaction time stamp in the table transactions.

The dynamic driver trigger(s) is prespecified, as the static driver trigger(s), but also inserts new records automatically on the fly as needed.

Conclusion

The static driver trigger is based on the five tables for accounts, account type, balances, logic, and transactions. The static driver trigger does not insert new records automatically as needed. The dynamic driver trigger combines the two tables for balances and logic into the table account type. The dynamic driver trigger is based on the three tables for accounts, account type, and transactions. The dynamic driver trigger inserts new records automatically as needed.

Acknowledgments

Thanks are due for helpful discussions to Larry Cagg, Cagg Enterprises, Westminster CO.

References

James, C. 2002, "Additional Information", unpublished, CEC Services, LLC, Loveland CO.

James, C. 2001, "Report Accounts [RA] v 1.2 Inventory / Point of Sale", unpublished, CEC Services, LLC, Loveland CO.

James, C., 2001, "Report Accounts [RA] v 1.2", unpublished, CEC Services, LLC, Loveland CO.

James, C., 1998, "Competency test for CEC Services, LLC", unpublished, CEC Services, LLC, Loveland CO.

James, C., 1998, "Ticket Reservations [TR] ver 1.1", unpublished, CEC Services, LLC, Loveland CO.

James, C., 1998, "Multiple and Self-Modifying Logic Tables with Queries", unpublished, CEC Services, LLC, Loveland CO.

James, C., 1997, "User Documentation", unpublished, CEC Services, LLC, Loveland CO.

James, C., 1997, "Logic Table Design for Reports in RA", unpublished, CEC Services, LLC, Loveland CO.