Description the bug in retail\_modified\_item\_barcode\_list\_i trigger.

The purpose of this trigger is to created rows in the retail\_modified\_item\_barcode table based upon the SQL inserts into the retail\_modified\_item\_barcode\_list.  On certain conditions (mostly during item imports) the trigger will fail with a duplicate key insert error in the retail\_modified\_item\_barcode table.  This error appears to be caused by a defect in the trigger.

The logic of the trigger is:

Insert the values from the inserted trigger table into a table variable that contains an identity column:

insert into @barcode\_list (retail\_modified\_item\_id

              ,barcode\_id

              ,client\_id

              ,last\_modified\_user\_id

              ,last\_modified\_timestamp)

select   retail\_modified\_item\_id

        ,barcode\_id

        ,client\_id

        ,last\_modified\_user\_id

        ,last\_modified\_timestamp

      from inserted

Get the number of tickets needed based upon the insert statement:

select @cnt = @@rowcount

exec @ticket = plt\_get\_next\_named\_ticket 'Retail\_Modified\_Item\_Barcode','n',@cnt

Insert the rows for the primitive complete code into the retail\_modified\_item\_barcode table.  The key value is determined by subtracting the identity column value from the new key:

insert into retail\_modified\_item\_barcode (retail\_modified\_item\_id

                    ,barcode\_id

                    ,barcode\_type\_code

                    ,barcode\_number

                    ,client\_id

                    ,last\_modified\_user\_id

                    ,last\_modified\_timestamp)

  select i.retail\_modified\_item\_id

      ,@ticket-i.int\_id

      ,b.barcode\_type\_code

      ,b.primitive\_complete\_code

      ,i.client\_id

      ,i.last\_modified\_user\_id

      ,i.last\_modified\_timestamp

    from @barcode\_list i

    join barcode b

      on i.barcode\_id = b.barcode\_id

      and not exists (select 1 from Retail\_Modified\_Item\_Barcode bl

                where bl.retail\_modified\_item\_id = i.retail\_modified\_item\_id

                and bl.barcode\_type\_code = b.barcode\_type\_code

                and bl.barcode\_number = b.primitive\_complete\_code)

The code then performs a similar operation when the barcode primitive complete code does not match the primitive compressed code.  The key value is determined by subtracting the identity column value from the table variable:

/\* if the complete code is different from the compressed code

                            then insert record into Retail\_Modified\_Item\_Barcode \*/

select @compressed\_code\_cnt = ( select count(\*)

                               from barcode b

                               join @barcode\_list i

                                on i.barcode\_id = b.barcode\_id

                               where primitive\_compressed\_code <> primitive\_complete\_code )

if ( @compressed\_code\_cnt > 0 )

begin

  exec @ticket = plt\_get\_next\_named\_ticket 'Retail\_Modified\_Item\_Barcode','n',@compressed\_code\_cnt

  insert into retail\_modified\_item\_barcode (retail\_modified\_item\_id

                      ,barcode\_id

                      ,barcode\_type\_code

                      ,barcode\_number

                      ,client\_id

                      ,last\_modified\_user\_id

                      ,last\_modified\_timestamp)

    select i.retail\_modified\_item\_id

        ,@ticket-i.int\_id

        ,b.barcode\_type\_code

        ,b.primitive\_compressed\_code

        ,i.client\_id

        ,i.last\_modified\_user\_id

        ,i.last\_modified\_timestamp

      from @barcode\_list i

      join barcode b

        on i.barcode\_id = b.barcode\_id

        and not exists (select 1 from Retail\_Modified\_Item\_Barcode bl

                  where bl.retail\_modified\_item\_id = i.retail\_modified\_item\_id

                  and bl.barcode\_type\_code = b.barcode\_type\_code

                  and bl.barcode\_number = b.primitive\_compressed\_code)

This logic can result in a duplicate key error if a compressed barcode that is different than the complete barcode occurs far enough along in the list.  For example, if the barcode set of the retail item is this:

Complete Code           Compressed Code

1111111111              1111111111

2222222222              2222222222

3333333333              33333

Given a starting key of 1000001, the first insert statement would generate the key values of

Complete Code           Identity                Key Value

1111111111              1               1000001

2222222222              2               1000002

3333333333              3               1000003

When the count of compressed codes is determined, the count is 1, so the next key needed is 1000004.  However because the second statement uses the same identity values as the first statement, the key generated will produce a duplicate.

Compressed              Identity                Key Value

33333                   3               1000001 (A duplicate key, new key 1000004 – identify of 3)