BR_EntityLink – Entity Link Services

Revision history

2017-03-24	Robin Lamacraft	Original draft
2018-04-11	John Lucas	Revision of Action items #1 and 4
2018-06-03	Don Ferguson	Added email dialogue on this issue from RL's email of 14 April

SCOPE

This module must be used to allocate, disable and retrieve the persistent (non-reusable) unique HRE_ID values that control the type and inter-relationship of all records in all HRE databases.
NOTE: All records in all HRE tables have an HRE-ID identity, whether they represent an Entity or a Link between 2 Entities.

This module also controls the access to a database record (or its equivalent memory-based object). It compares the rights of the current user with the rights associated with that data base record to confirm or deny an action on that record.

This module also assigns the ownership and rights for each new database record created.

ACTIONS

The fundamental operations are:

1. Create unique persistent HRE-ID. This construction is dependent on the HRE data type that is being requested.
   PROPOSAL: An HRE_ID is a 64bit integer value that has 4 parts with following suggested partitions:
   - The HRE-Entity Group (maximum 254 groups = 8 bits as 127 for user data and 127 for application data)
   - The HRE-Entity Pattern Index within a Group (maximum 65,536 = 16 bits)
   - The persistent index of this instance within that HRE- Entity Group Pattern Type (maximum 4,294,967,295 = 32 bits). This becomes the maximum number of records that have been created in a table (including the disabled and purged records over time)
   - Whether this is the most current record or not – a negative HRE-ID is a disabled version of the positive value. There can be only one positive but there may be several negative values existing the database at any one time.

   The encoding and the decoding of the HRE_ID must be extremely simple. The proposed encoding is very quick to decode for various purposes. When a record is superseded by a new update, reverse the sign on the disabled copy. Using the absolute value function ABS() in SQL will enable recovery of all transactions on that record that are still remaining in that table. Note that database transactions can simultaneously update several tables. The database would become corrupted if not all changes were recognized for any sequence of undo actions.
   a. Each HRE database record must have its own unique HRE-ID. Each record must have 3 Transaction ID fields - This Transaction, Previous Transaction and Next Transaction
   b. At the creation of a record, This Transaction ID is set, both Previous Transaction ID and Next Transaction ID are zero
   c. On subsequent update, the replacement record is created with a copy of the old record's HRE_ID, the original This Transaction ID is copied to the new record's Previous Transaction ID field

d. On the disabled record, rewrite it with the new record's Transaction ID copied into the Next Transaction ID field and rewrite the HRE-ID as its negative value.

NB: Table **104 Table_Defns** is the table that has a record for each HRE database table. The LAST_RECORD_PID field is the field in a record in that table that holds the highest PID value ever used for that corresponding table.

2. Allocate new HRE-ID
3. Decode an HRE ID
4. Retrieve the identity, ownership , access rights, timestamp and transaction ID of a database record
   NB: there are several places that interact to ascertain a user's rights to each table. Table **104 Table_Defns** has one record for each combination of table types. Table **131 Users** is used in association with **132 User_Group_Defns** to determine the rights of a particular user to view, perform operation on all recorded data or whether that user is explicitly precluded from seeing records for administration or cultural reasons.
5. Perform searches for a change log of transactions
6. Perform database change undo
7. Perform purge of transaction records before a particular date/time
8. For a given HRE-ID, provide a list of the non-identity fields in all HRE tables that could hold a reference to that HRE-ID.

   PROPOSAL: If we take a Citation as an example:
   (a) SOURCE SIDE: It always has a pair of fields Sub-Type-Key and a PID reference. The Entity type is always Contain so does not need recording here, but the sub-type is the Source type and the PID is the record that holds the details of that source
   (b) WHAT IS CITING THE SOURCE SIDE: The other entity link in the Citation record now has 3 fields to define which record of which entity type and which entity sub-type of that entity and which PID in that entity pertains to this Citation
   That is, both the references from a Citation to another record in the database are conditional on either one or two other fields as to which table and sub-type that the PID value actually refers to. The alternative to this (making each case a new field for each possible entity type, entity sub-type for (a) and (b) implies that the database table would have several sets of fields where in each set there would one real value stored. The other consequence of this separate field solution is that the schema has to be rebuilt every time a plug-in needs a new sub-type of the existing entity types.
   Hence, a table to hold "where is this database record possibly referenced" needs to be more complex, allowing for the same referencing field to be repeated in that 'where referenced' table with one record for each possible case. There are some cases where a reference linkage is unique, a tag instance can only reference one Tag definition. So this list is a list of possible references. It is not a list of all records that reference this record, rather it is map of the types of records in various identified tables where specific fields could reference the record in focus. For some cases, it would be appropriate to have database maintained index files to speed up this searching. At this moment the HRE schema does not hold that this table/field may/always references another table information.

   It would be essential for many modules, particularly those making database record edits and validating data, to have a standard method to list all the places where that record's PID resided in the database.

As the number of records required in that table is not 1:1 with any existing table, the only method to support this data is to add another table to the schema.

USED BY:
Any data management process.

DATA UNDER CONTROL OF THIS MODULE

1. The database Field Definition table
2. HRE_ID last used ID database
3. HRE-ID where referenced database.

REQUIRED EXTERNAL DATA UNDER CONTROL OF ANOTHER MODULE

1. Transaction Data Base.

REQUIRED SERVICES

1. Needs details

APPLICATION PROGRAMMING INTERFACE (API)

1. Needs details

EVENT ACTIONS

1. Need details of event.

WARNING CONDITIONS

1. Need details of the condition that raised the warning, example message and possible next steps.

ERROR CONDITIONS

1. Need to record the condition that raised the error, example message and possible next steps.