HRE – COMMIT LOGGING - OVERVIEW

Revision history

2018-08-03	Robin Lamacraft	Original draft
2018-08-08	Don Ferguson	Tidy-up edits throughout

SCOPE

As an HRE Project Database is edited, HRE can record the sequence of changes in a form that:

- The user can view to understand who made what change when. These records are bundled together on the basis of the group of record changes that are made within one database Commit action
- These commit logging records are separate tables in the HRE database that will need to be managed to avoid their size getting too large. Some tools are provided to purge log records on different criteria
- These commit logs are likely to be very useful in understanding why the user ended up with an unexpected result and hence will be an aid to understanding and removing bugs
- These log records will allow the user to undo actions in reverse sequence to recover a corrupted database when possible.

THE COMMIT LOGGING DATABASE STRUCTURE

There are 2 versions of each database table:

1. The tables that hold the active application control and user-entered data (as has been described in many HRE specification documents). These tables are complete in themselves and are used when retrieving data values or composing output
2. The second set of tables with names prefixed with CL_ (for Commit Logging) have
    a. a structure that has a common set of fields for all Commit Logging tables
    b. and an additional set of fields which is then an exact copy of the fields that are in the table being logged.

EDITING A GROUP OF TABLES TO CONSISTENTLY UPDATE THE DATABASE

Each proposed change to the database, which may involve making complementary changes to several database tables, has to be considered as the only method by which the database is modified:

1. The list of active tables that are involved in this update must be determined and the updating process must gain exclusive access to these records while the update proceeds
2. The whole process must be controlled by a singular SQL COMMIT action. That Commit action assigns a unique ID, the user requesting the change and time stamp of the start of the change
3. Once the record to represent this Commit action is created, then a series of records that contain the old data values of the records about to be overwritten are copied into equivalent CL_named tables of the same name as the original(s)
4. On completion of the old value copy operation, the revised values are stored in the live record
5. When this is successfully completed, the action completion time is saved in the Commit Identified record in the commit log
6. Finally, the records which were set with exclusive access are released.

USING THE COMMIT LOG

- A view of the most recent Commit action is presented at the top of a list of actions.
- Each Commit can be undone

- A range of Commit action logs can be selected and printed or saved to a file.