HRE – DATABASE UPDATING - OVERVIEW

Revision history

2018-06-02	Robin Lamacraft	Original draft

SCOPE

There are a number of objectives when defining the way the HRE Project database is updated. Always:

1. Use complete record replacement rather than updating a field
2. Use a COMMIT to encapsulate a group of inter-related changes
3. Perform a validation test for each field value change
4. Perform a validation test for each record content change
5. Log each COMMIT with a unique identifier and link that identifier to the records that have been modified
6. Keep each now obsolete record in a manner that an audit trail exists
7. Be able to view that audit trail
8. Be able to undo (in sequence) a series of COMMITS
9. On receipt of a warning or error notification , be able to view the Java call stack
10. Be able to purge older parts of the audit trail.

SUGGESTED STRATEGY

Each record in the database has a unique (never reused) Persistent Identity (PID). This is separate from any user defined visible Object ID (VID). All internal operations use the PID. The record PID is a positive integer when it is the current record of the PID. When that record is made obsolete its PID is replaced by its negative value.

The HRE database tables **105 RECORD_VALID_DEFNS** and **115 FIELD_VALID_DEFNS** provide access to Jython Scripts that validate for that record type and field within record to determine whether the edit is valid or not. NOTE: The use of Jython scripts here means that those scripts in that database can be updated without modifying the base Java application.

HRE database tables **129 COMMIT LOGS** and **130 COMMIT ITEMS** are intended to provide the change audit trail. The incrementing value PID of the COMMIT LOGS record becomes the COMMIT identifier. Included here are the date stamp of the action and the identity of the user. Then that COMMIT LOG PID is stored in each COMMIT ITEMS record that relates to that COMMIT. The COMMIT ITEM record holds the TABLE_KEY and RECORD PID of this change. It also stores that previous COMMIT LOG PID (as field PREV_CMT_PID) that last modified that record. That gives a record-based audit trail that also links to changes made in the prior COMMITS.

This methodology should help when investigating unexpected outcomes for the user and for the development team in identifying the cause of a bug.