7/3/2014

|  |  |
| --- | --- |
| **Database Table** | **Description/Usage** |
| Abstract | Main table for Abstracts storage. Data is loaded into this table by DBA. |
| AbstractStatusChangeHistory | The first record for each Abstract will be inserted into this table by DBA when the Abstract is loaded into the database. AbstractStatusCode will be set to “0”. In addition, CreatedDate and CreatedBy value will be filled in. CreatedBy column will contain *System Admin* account’s UserId.  Application code will insert new records for the same Abstract each time the Abstract’s Status is changed”. In addition, CreatedDate and CreatedBy value will be filled in. CreatedBy column will contain user’s identity (UserId from dbo. aspnet\_Users table).  **No update or delete operations for this table. Inserts only! Insert needs to be the last step in developer data entry process.** |
| Evaluation | Evaluation is an entity used to model the process. When evaluation process starts, a row is inserted into this table with appropriate EvaluationTypeId. IsCompete column is set to “true” when Consensus data is submitted by Coders (1B) OR Comparison is done by ODP Staff Members (2C). In addition, DateTimeEnded value is inserted when Abstract Status is changed to 1B or 2C). |
| EvaluationType | 2 different Evaluation Types exist: *Coder Evaluation (1 > 1B), ODP Staff Members Evaluation (2 > 2C).* |
| AbstractCategory | Shows what category each Abstract belongs to. Combination of *CategoryID* and *Sorting* columns has to be unique. This combination identifies a group the next Abstract is picked from. This design allows DB administrator to rearrange groups by modifying *Sorting* column values for Abstracts with the same *CategoryID.* |
| Submission | Main table for submitted evaluation data |
| SubmissionType | Submission Type is used to differentiate between submissions for different evaluation processes stored in the same table. 5 submission types exist: *Coder Evaluation (1 > 1A), Coder Consensus (1A > 1B), ODP Staff Members Evaluation (2 > 2A), ODP Staff Member Consensus (2A > 2B),* and *ODP Staff Member Comparison (2B > 2C).* |
| Status | StatusID column is used in multiple tables as Foreign Key (FK) to mark items *active*, *inactive*, or *deleted* instead of physical row deletion. If StatusID exists in any table as a FK, developers have to check the Status making sure the item is *Active*. |
| A\_StudyFocus | *Sort* column should be used for display order on Evaluation Form.  A1\_IsEnabled, A2\_IsEnabled, A3\_IsEnabled are used to mark not available columns in Abstract Coding form. |
| Team | At midnight SQL job will set the status of all teams to *Deleted*. When supervisor overrides existing Team he also sets Team’s Status to *Deleted* instead of physical row deletion in Team table. In addition, the value of “UpdatedBy” column in Team table is updated by developers and by database job.  When a new Team is created, application code will set CreatedBy and CreatedDateTime values.  Team Type is used to differentiate between *Coders* and *ODP Staff Members* teams stored in this table. |
| AbstractReviewList | At each moment not more than one Review List can exist. The list could be modified any time through the user interface. Review List is a subset of items from Abstract table with Status “1N”. It’s used for display purposes. |
| Protocol | Contains the most current definition/description for the Taxonomy form |
| ProtocolHistory | History record of protocol when definitions are updated |
| ProtocolVersionControl | Version control for Protocol |
| KappaBaseData | Base data used in Kappa Calculation |
| KapppaData | Result of Kappa Calculation. Shows the K1-K9 scores for each abstract. |
| KappaType | Lookup table of the Kappa Type and its description |
| KappaUserIdentify | Identify which user on a team is Coder A, Coder B, Coder C, and ODP Coder A, ODP Coder B, and ODP Coder C. |
|  |  |

# Data Cleaning/ Canceling Evaluation Process

Supervisor of each group (Coders or ODP Members) can stop (override) evaluation process. This could be done BEFORE the Abstract under evaluation gets the Status 1B or 2B. Overriding functionality will require doing Abstract Cleaning (see details below). Team Cleaning could be done separately through the user interface.

SQL job will run at midnight to perform data cleaning – both Abstract Cleaning and Team Cleaning.

## Abstract Cleaning Steps

1. All Abstracts with Abstract Status Code “1” or “1A” should be changed to Abstract Status Code = “0” and all Abstracts with Abstract Status Code “2” or “2A” should be changed to Abstract Status Code = “1N”. New records will be inserted into *AbstractStatusChangeHistory* table for that. The following values will be used:

AbstractStatusChangeHistoryID – identity column

AbstractID - FK

AbstractStatusID = 1 (reset from “1” and “1A”to “0”) OR AbstractStatusID = 6 (reset from “2” and “2A”to “1N”)

CreatedDate = current date and time

CreatedBy = User’s Id (**application code** update) OR CreatedBy = System Admin Id (**nightly job** update),

EvaluationId = FK (Used original EvaluationID)

* For “1” or “1A” to “0”, the Evaluation ID will be null
* For “2” and “2A”to “1N”, the Evaluation ID will be the previous Evaluation ID for “1N”

1. Update row with appropriate EvaluationId in *Evaluation* table. Set the following value:

* IsStopped = 1,
* StoppedBy = User’s Id (**application code** update) OR StoppedBy = System Admin Id (**nightly job** update),
* StoppedDateTime = current date and time

1. Update row for appropriate EvaluationId in *Submission* table and set the following value:

* StatusID = 3 (Deleted),
* UpdatedBy= User’s Id (**application code** update) OR UpdatedBy = System Admin Id (**nightly job** update),
* UpdateDate= current date and time

## Coder Team Cleaning

All Coder Teams with Status *Active* will get Status *Deleted* – rows will be updated in Team table.

# Evaluation Process Modelling

(Based on Abstract Workflow Diagram)

|  |  |
| --- | --- |
| **Abstract Status Code** | **Actions** |
| 0 | DBA loads data into *Abstract* table and inserts a row for each Abstract into *AbstractStatusChangeHistory* table with the following column values:  AbstractStatusChangeHistoryID – identity column  AbstractID - FK  AbstractStatusID = 1  CreatedDate = set current date  CreatedBy = System Admin Id from *aspnet\_Users* table  EvaluationId = NULL  Abstract Status could be changed from 1 and 1A to 0 by nightly job or as a result of process overriding/cancelling. See Abstract Cleaning Steps above for details. |
| 1 | When Coder Process starts, Abstract’s Status Code is changed from 0 to 1. A new row is inserted into *AbstractStatusChangeHistory* table with the following column values:  AbstractStatusChangeHistoryID – identity column  AbstractID - FK  AbstractStatusID = 2  CreatedDate = current time  CreatedBy = UserId from *aspnet\_Users* table  EvaluationId - FK  Also, a new row is inserted into *Evaluation* table (new EvaluationId) with the following values:  EvaluationTypeId = 1 (Coder Evaluation)  ConsensusStartedBy = NULL  TeamID – insert current team ID  AbstractID - insert current abstract ID  IsComplete = 0  DateTimeStarted – insert current time  DateTimeEnded = NULL  IsStopped = 0  StoppedBy = NULL  StoppedDateTime = NULL  Individual Coder’s evaluation data will be saved to *Submission* table with the following value:  SubmissionID - identity column  SubmissionTypeId = 1 (Coder Evaluation)  EvaluationId = FK  UserId = UserId from *aspnet\_Users* table  StatusID = 1 (*Active*)  UpdatedBy = NULL  UpdatedDate = NULL  SubmissionDateTime = current time  Notes – user’s data from application Form  UnableToCode – user’s data from application Form  Additional tables exist in the database for storing answers with *SubmissionID* as a Foreign Key. |
| 1A | After all 3 coders saved data into *Submission* table for the same EvaluationId; Abstract’s Status Code will be changed to 1A. Application code will insert a new row into *AbstractStatusChangeHistory* table with  AbstractStatusID = 3 |
| 1B | On Coders’ Consensus starting application code will update row for the current EvaluationId and set ConsensusStartedBy = UserId.  On Coders’ Consensus submission application code will do the following:  Validate that UserId is the same as ConsensusStartedBy value in *Evaluation* table – only the user who started consensus is allowed to submit consensus data;  Consensus data is saved into *Submission* table for the same EvaluationId and SubmissionTypeId = 2 (Coder Consensus);  Update row for the same EvaluationId in *Evaluation* table, set IsComplete = 1, DateTimeEnded = current time;  Insert a new row into *AbstractStatusChangeHistory* table with AbstractStatusID = 4  Call store procedure: KappaBaseData\_Insert\_ByAbs\_EvlID with the following parameters: @AbstractID int, @EvaluationId int, @AbstractStatusID int. Will insert or modify data on tables:   * KappaBaseData * KappaData\_Abs\_Evl\_Team * KappaDataTracking * KappaBaseLog |
| 1N | When Coder Supervisor uploads Scanned evaluation forms, application code will do the following:   1. Insert row into *AbstractScan* table   AbstractScanID – identity column  EvaluationId – FK  FileName – Naming convention: concatenate AbstractID, current date, and add suffix “\_coder”  UpdatedBy = UserId  UpdatedDateTime = current time  **Details: AbstractScan is related to Evaluation. On PDF file uploading application code will check if there is a row in AbstractScan table for EvaluationId. If NOT, a new row will be inserted into AbstractScan table and PDF file will be saved on the hard drive. If the record exists, the file name will be retrieved from it (FileName column) and PDF file will be overwritten on hard drive.**  The same scan could be uploaded again (overwritten on hard drive);   1. Insert a new row into *AbstractStatusChangeHistory* table with AbstractStatusID = 6   Abstract Status could be changed from 2 and 2A to 1N by nightly job or as a result of process overriding/cancelling. See Abstract Cleaning Steps above for details. |
| 2 | When ODP Staff Process starts, Abstract’s Status Code is changed from 1N to 2. A new row is inserted into *AbstractStatusChangeHistory* table with the following column values:  AbstractStatusChangeHistoryID – identity column  AbstractID - FK  AbstractStatusID = 7  CreatedDate = current time  CreatedBy = UserId from *aspnet\_Users* table  EvaluationId - FK  Also, a new row is inserted into *Evaluation* table (new EvaluationId) with the following values:  EvaluationTypeId = 2 (ODP Staff Members Evaluation)  ConsensusStartedBy = NULL  TeamID – insert current team ID  AbstractID - insert current abstract ID  IsComplete = 0  DateTimeStarted – insert current time  DateTimeEnded = NULL  IsStopped = 0  StoppedBy = NULL  StoppedDateTime = NULL  Individual ODP Staff Member’s evaluation data will be saved to *Submission* table with the following values:    SubmissionID - identity column  SubmissionTypeId = 3 (ODP Staff Members Evaluation)  EvaluationId = FK  UserId = UserId from *aspnet\_Users* table  StatusID = 1 (*Active*)  UpdatedBy = NULL  UpdatedDate = NULL  SubmissionDateTime = current time  Notes – user’s data from application Form  UnableToCode – user’s data from application Form  Additional tables exist in the database for storing answers with *SubmissionID* as a Foreign Key. |
| 2A | After all 3 ODP Staff Members saved data into *Submission* table for the same EvaluationId; Abstract’s Status Code will be changed to 2A. Application code will insert a new row into *AbstractStatusChangeHistory* table with  AbstractStatusID = 8 |
| 2B | On ODP Staff Member’s Consensus starting application code will update row for the current EvaluationId and set ConsensusStartedBy = UserId.  On ODP Staff Member’s Consensus submission application code will do the following:  Validate that UserId is the same as ConsensusStartedBy value in *Evaluation* table – only the user who started consensus is allowed to submit consensus data;  Consensus data is saved into *Submission* table for the same EvaluationId and SubmissionTypeId = 4 (ODP Staff Member Consensus);  Insert a new row into *AbstractStatusChangeHistory* table with AbstractStatusID = 9  Call store procedure: KappaBaseData\_Insert\_ByAbs\_EvlID with the following parameters: @AbstractID int, @EvaluationId int, @AbstractStatusID int. ***Note: since 1B and 2B in different eveluation processing step, so for one abstract,*** it has different EvaluationID value. Will insert or modify data on tables:   * KappaBaseData * KappaData\_Abs\_Evl\_Team * KappaDataTracking * KappaBaseLog |
| 2C | On ODP Staff Member Comparison starting application code will validate that the Abstract Status Code is 2B and that UserId is the same as the value in ConsensusStartedBy column in evaluation table for the current process.  On ODP Staff Member Comparison submission application code will do the following:  Validate that UserId is the same as ConsensusStartedBy value in *Evaluation* table – only the user who started consensus is allowed to submit consensus data;  Consensus data is saved into *Submission* table for the same EvaluationId and SubmissionTypeId = 5 (ODP Staff Member Comparison);  Update row for the same EvaluationId in *Evaluation* table, set IsComplete = 1, DateTimeEnded = current time;  Insert a new row into *AbstractStatusChangeHistory* table with AbstractStatusID = 10 |
| 2N | When ODP Staff Supervisor uploads Scanned evaluation forms, application code will do the following:   1. Insert row into *AbstractScan* table   AbstractScanID – identity column  EvaluationId – FK  FileName – Naming convention: concatenate AbstractID, current date, and add suffix “\_odp”  UpdatedBy = UserId  UpdatedDateTime = current time  **Details: see details for 1N**  The same scan could be uploaded again (overwritten on hard drive);   1. Insert a new row into *AbstractStatusChangeHistory* table with AbstractStatusID = 12 |
| 3 (application version 1.1) | Admin or ODP Supervisor can change Abstract Status from 1N and 2N to 3 through application interface. When it happens, a record is inserted into the *AbstractStatusChangeHistory* table with AbstractStatusID=13 |
| 4 (application version 1.1) | Admin or ODP Supervisor can change Abstract Status from 3 to 4 through application interface. When it happens, a record is inserted into the *AbstractStatusChangeHistory* table with AbstractStatusID=14 |
|  |  |