Project: Finding Duplicate Records (LightSys)

Team:

* Luke Avent
* Nathan Moyes
* Bryan Smith

Table of Contents

[Reviewing Documentation: 3](#_Toc98417199)

[Implementing UTF-8 Compatibility <>: 5](#_Toc98417200)

[Creation and merging of new branch for UTF-8 Compatibility <DONE> 5](#_Toc98417201)

[Reevaluation of Levenshtein, similarity function, and fuzzy compare algorithms in input processing 5](#_Toc98417202)

[False Positives With Shared Last Names: <INCOMPLETE> 6](#_Toc98417203)

[Parent-Partner Key Issue: <RESEARCHED> 6](#_Toc98417204)

[Note on Keys 7](#_Toc98417205)

[Needs: 7](#_Toc98417206)

[Solutions: 7](#_Toc98417207)

[IMPLEMENTATION: 8](#_Toc98417208)

[Future project suggestions 10](#_Toc98417209)

[Clean Up Excess Code 10](#_Toc98417210)

[Nonduplicate Flagging (Code) 10](#_Toc98417211)

[Nonduplicate Flagging (UI) 10](#_Toc98417212)

[Optimize Functions and Algorithms 11](#_Toc98417213)

[Prevent False Positives From Matching Names 11](#_Toc98417214)

[Resolve Repeating Rows Issue 11](#_Toc98417215)

# Reviewing Documentation:

* Previous documentation packets can be found on the riot session.
* READMEs can be located within most GitHub directories
* Test cases are found under centrallix/centrallix/tests/ in either the duplicate-checking or in the utf-8-duplicate-checking branches of Centrallix on GitHub.

Queries and Functions Added:

Files Edited/Added:

Weaknesses/Vulnerabilities of New:

# Implementing UTF-8 Compatibility <>:

### Creation and merging of new branch for UTF-8 Compatibility <DONE>

* Began by creating a new branch under Centrallix (utf-8-duplicate-checking) for work
* Merged the duplicate-checking branch of Centrallix (duplicate-checking)
  + Merge conflicts were resolved

### Reevaluation of Levenshtein, similarity function, and fuzzy compare algorithms in input processing

* After discussing with project owner, a decision was made to design modularly by region (since some language characters cannot be processed/converted properly)
* Levenshtein analysis was reevaluated along with the method of using similarity analysis and the fuzzy-compare
  + A (logic diagram/gate here) was used to parse input without compromising integrity of certain characters
  + Initial tests started with \_\_\_ and Latin character sets: test cases are in \_\_\_\_ (or manually entered?)

# False Positives With Shared Last Names: <INCOMPLETE>

* No current solution in progress- test after implementing manual flagging code

# Parent-Partner Key Issue: <RESEARCHED>

* Each data entry corresponds with an identifying key (primary key for person entries, partner key for partner organization entries).
* There are a couple of functions that check for duplicates based off similarities in keys and biographical data
* When there is a close match, false positives occur (i.e. Greg Beeley & Greg and Dorinda Beeley would both be flagged as duplicates even though they are intentionally separate)
  + Which function is checking these keys for duplicates?
  + Which ones assign/generate these keys?
* Searched GitHub directory for files editing:
  + "parent\_key" (NEVER USED)
  + "partner\_key"
  + "p\_partner\_key"
  + Primary Keys
  + "duplicate\_checking.qy"
  + "duplicate\_records.qy"
  + "duplicate\_create\_pairs.qy"
  + Function that generates table “tmp\_pcl\_duplicates”
  + "tmp\_partner\_data"
  + "tmp\_partner\_condensed"
  + "tmp\_partner\_pair

Note on Keys:

* Parent keys are a type of foreign key (points back to a primary key) that relates partner keys (a type of primary key) with the primary key of each input
* The problem is when the program does not recognize that the parent key connects two entries and falsely flags them as duplicates (they are actually intentionally separate entries)
  + This is because parent keys (now relate\_key) were never implemented by previous teams

### Needs:

* Program must check against parent keys (**never implemented before the 3/14/22-3/18/22 sprint**) to prevent false marking of duplicates
* Must find the table that each partner key/primary key is drawn from
* Must also locate table or function that the parent key is drawn from/given by
  + Since it doesn’t exist, can it use the same tables as primary or partner keys?

### Solutions:

* Add code that creates and assigns a parent key to specific entries
  + How to make sure non-related entities aren't accidentally connected?
  + Manual flagging by user:
    - Prompt to flag duplicates as non-related during output
    - Added to table during initial entry of info?
* Rewrite code to draw from/check for parent keys when comparing/locating duplicates and not include them in output (or not pass them to the table-generating function)
  + Use in temp tables of duplicate\_checking.qy query and its join function

### IMPLEMENTATION[[1]](#footnote-1):

* Edit all files that implement/initialize tables to include a category for relate\_key[[2]](#footnote-2) and allow users to set relationships (entries sharing the same relate\_key) during initial data entry
  + api\_partner.qyt (pulls partner data from a number of tables): kardia/kardia-app/modules/base/api\_partner.qyt
    - Data tables location: /apps/kardia/data/Kardia\_DB
  + new\_partner.cmp (Adds new partner entries, generates partner key, and checks for duplicate partner entries): kardia/kardia-app/modules/base/new\_partner.cmp
  + p\_partner\_relationship.cmp (Sets a relationship value between a partner and person entry): kardia/kardia-app/modules/base/p\_partner\_relationship.cmp
  + partner\_edit.cmp (allows editing of a partner's data): kardia/kardia-app/modules/base/partner\_edit.cmp
  + person\_data.qy (pulls data and keys for a person entry): kardia/kardia-app/modules/base/person\_data.qy
  + p\_person.cmp (creates UI for pulling/editing person data): kardia/kardia-app/modules/base/p\_person.cmp
* Add a method to duplicate\_checking.qy that checks potential duplicates to see if their relate\_key values match and avoid printing these related entries as duplicates.
  + duplicate\_checking.qy (evaluates data to detect duplicates and feeds results to tmp\_pcl\_duplicates):
  + kardia/kardia-app/modules/base/duplicate\_checking.qy
* Add an option (via query, code, and/or eventually UI) to use the relate\_key value to flag “duplicates” given by duplicate\_records.qy and duplicate\_create\_pairs.qy as related nonduplicates
  + duplicate\_create\_pairs.qy (sorts duplicate results into pairs for comparison): kardia/kardia-app/modules/base/duplicate\_create\_pairs.qy
    - table tmp\_sorted\_duplicates (where duplicate\_create\_pairs.qy data is stored): LOCATION UNKNOWN- Generated/destroyed during runtime?
  + duplicate\_records.qy (returns data for each "duplicate" especially primary and partner keys): kardia/kardia-app/modules/base/duplicate\_records.qy
  + dup\_check.cmp (presents results of duplicate checking queries): kardia/kardia-app/modules/base/dup\_check.cmp

Future project suggestions:

### Clean Up Excess Code

* It has been observed through testing that there are some files/queries that are currently unused, and that appear to have been copied into other code, making them unnecessary. Removing these would reduce confusion while navigating files, during testing, and with final product.
* A list (though not comprehensive) of known excess code is below:
  + letter\_frequency (replaced by similarity query): centrallix/centrallix/test (duplicate-checking and utf-8-duplicate-checking branches)

### Nonduplicate Flagging (Code)

* As discussed earlier in this documentation, there is a need to add code that allows flagging of detected “duplicates” as being related by a relate\_key value which tells the program and users that the two entries are not duplicates, but intentionally related. This can be done by:
  1. Revising and expanding code that creates data tables for partner and person entries to include the necessary flags as a field/category
  2. Write a method in files such as duplicate\_checking.qy to compare “duplicates” and check their relate\_key -if they match, they are no longer duplicates and are not passed to other operations that collate duplicates into tables for output.
  3. Add queries or other operations that allow for code to change/add a value to the relate\_key field.
* There may need to be separate flags for “duplicates” that are related (i.e. spouses registered both individually and together) and “duplicates” that are unrelated, but are intentional nonduplicates (i.e. two persons with similar names and residency, but different demographics).

### Nonduplicate Flagging (UI)

* Building on the above, add UI and widget elements so users can directly relate person and partner entries via relate\_key or at least flag nonduplicates.

### Optimize Functions and Algorithms

* Previous teams suggested

### Prevent False Positives From Matching Names

### Resolve Repeating Rows Issue

1. All filepaths are under the duplicate-checking branches of Centrallix and Kardia, respectively. [↑](#footnote-ref-1)
2. A.K.A. parent\_key as called in previous documentation by the March 11th team (pg. 7) [↑](#footnote-ref-2)