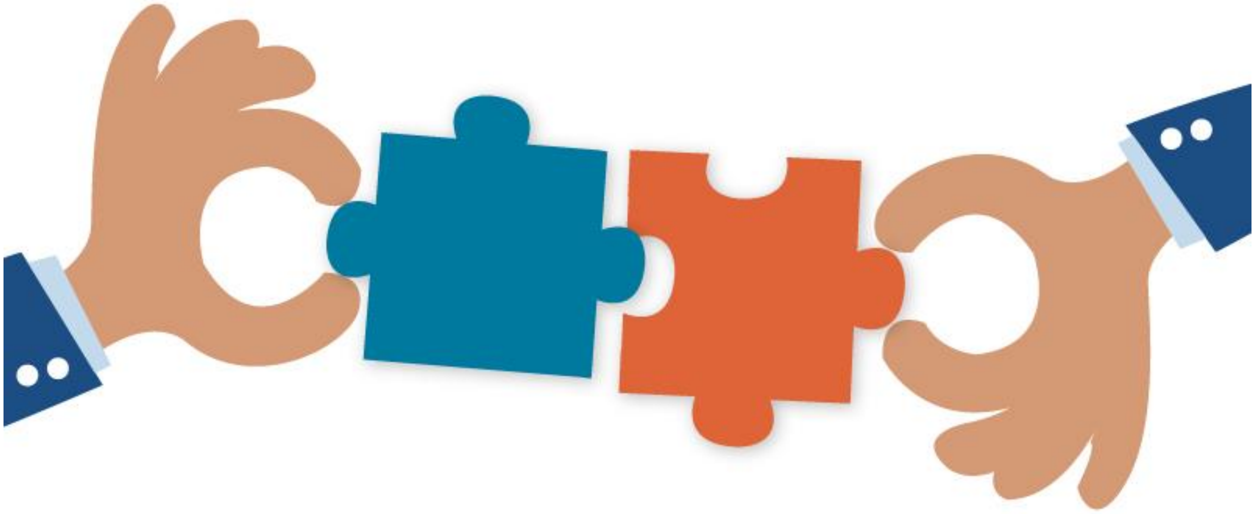
XDRO Data Linkage Application



Project PI: Dr. William Trick

Project Lead: Vicky Gao

Lead Developer: Kruti Doshi

Project Date: February 2nd, 2019

Github: https://github.com/k-doshi/CRU-PPRL-registry-hashing-application

Index

Data Linkage Application Overview

The data linkage application is designed to disambiguate patients across sites / health systems, with minimal risk from transferring PHI by hashing the patient identifiers with a secured salt and hashing algorithm. The hashed data from the health systems / clinics / participating sites are matched against the hashes from patient data Registry (maintained by Illinois Department of Public Health). The matches trigger an alert system (separate system, integrated with the application) and alert the site of a patient match.

Main Components of the application are:

1. Data Standardization, exceptions, and hashing
2. Disambiguation / matching

Each component is explained in more detail below:

1. **Data Standardization, exceptions, and hashing**
   This component includes a data pipeline to digest and validate the data, standardize the data, manage data exceptions, create composite variables from patient identifiers and hash using SHA512 algorithm and Salt

2. **Disambiguation / matching**
   This component allows the aggregator to merge files, disambiguate the hashes and assign Universal patient ID to matching patients

Data Transformations and Filters

1. Transform first and last name – Remove leading and trailing spaces, remove common suffix & prefix, convert hyphen to space, retain 1 space in between, remove all other special characters, and upper case
2. Transform SSN – Keep only last 4 numbers
3. Filter records – Remove all rows with names separated by space beginning or ending in BOY, GIRL, TWIN A, TWIN B, TWIN B (e.g. BABY BOY)
4. Split last name – Split the last name separated by space, retain the original record, populate two new rows with each part of the spilt last name (data in other cells remains same) and flag them as shy_der_flg
5. Transform first and last name – Keep alpha characters only
6. Filter records – Remove rows with the date of birth is missing, the name is missing, the length of name is 1. Remove rows with common name errors (e.g.: ‘UNKNOWN', 'MALE', 'FEMALE', 'RESEARCH ' etc.)

Composite Identifiers and Hashing

1. Concatenate cells to hash –
    a. first name + last name + dob + last 4 ssn (only for records with ssn)
    b. last name + first name + dob + last 4 ssn (only for records with ssn)
    c. first name + last name + dob
    d. last name + first name + dob
    e. first name + last name + Transposed dob + last 4 ssn (only for records with ssn)
    f. first name + last name + Transposed dob
    g. first name 3 initial characters + last name + dob + last 4 ssn (only for un-flagged records)
    h. first name 3 initial characters + last name + dob (only for un-flagged records)
    i. first name + last name + dob + 1 day + last 4 ssn (only for records with ssn)
    j. first name + last name + dob + 1 year + last 4 ssn (only for records with ssn)
2. Hash – Hash using SHA512 algorithm

Patient Matching Rules

Below are the matching rules that trigger an alert:

No.	Composite identifiers (Hashed)	Matching rule
1	First Name + Last Name + DOB + SSN	FULL MATCH
2	First Name + Last Name + DOB	FULL MATCH
3	Last Name + First Name + DOB + SSN	TRANSPOSED NAME FULL MATCH
4	Last Name + First Name + DOB	TRANSPOSED NAME FULL MATCH
5	First Name + Last Name + transposed DOB + SSN	TRANSPOSED DATE OF BIRTH FULL MATCH
6	First Name + Last Name + transposed DOB	TRANSPOSED DATE OF BIRTH FULL MATCH
4	First 3 characters First Name + Last Name + DOB + SSN	PARTIAL MATCH
5	First 3 characters Last Name + First Name + DOB + SSN	TRANSPOSED NAME PARTIAL MATCH
6	First Name + Last Name + DOB + 1 day + SSN	MODIFIED DATE OF BIRTH FULL MATCH
7	First Name + Last Name + DOB + 1 year + SSN	MODIFIED DATE OF BIRTH FULL MATCH

Source Codes: Data transformation functions

```
USE $(Database)
GO

--while running sql only script, please replace these values:
----$(Database),$(schema)

--one time set up file to create functions and stored procedures to hash demographic table

--hash and salt function type 2
IF OBJECT_ID(N'$(schema).fnHashBytes2', N'FN') IS NOT NULL
BEGIN
  DROP FUNCTION $(schema).fnHashBytes2
END
GO

CREATE FUNCTION [$(schema)].[fnHashBytes2] (@DataToHash VARCHAR(MAX), @Salted
VARCHAR(30))
RETURNS VARCHAR(128)
AS
BEGIN
   DECLARE @HashedResult VARCHAR(128)
   IF @DataToHash IS NOT NULL
      BEGIN
         SET @HashedResult = CONVERT(VARCHAR(128), HASHBYTES('SHA2_512',
@DataToHash+@Salted), 2)
      END
      RETURN @HashedResult
END
GO

--remove double space and hyphen
IF OBJECT_ID(N'$(schema).stripDoubleSpaces', N'FN') IS NOT NULL
BEGIN
 DROP FUNCTION $(schema).stripDoubleSpaces
END
GO

CREATE FUNCTION [$(schema)].[stripDoubleSpaces](@prmSource VARCHAR(100))
RETURNS VARCHAR(100)
AS
BEGIN
   DECLARE @keepValues AS VARCHAR(50)
   SET @keepValues = '%[-]%'
   WHILE PATINDEX(@keepValues, @prmSource)>0
      SET @prmSource = STUFF(@prmSource, PATINDEX(@keepValues, @prmSource), 1, ' ')
          WHILE (PATINDEX('%  %', @prmSource)>0)
      SET @prmSource = STUFF(@prmSource, PATINDEX('%[  ]%', @prmSource), 1, '')
                 --@prmSource = replace(@prmSource  ,'  ',' ')
   RETURN @prmSource
END
GO

--keep alphabets, space and hyphens only
IF OBJECT_ID(N'$(schema).fnAlHySpOnly', N'FN') IS NOT NULL
BEGIN
 DROP FUNCTION $(schema).fnAlHySpOnly
END
GO

CREATE FUNCTION [$(schema)].[fnAlHySpOnly](@string VARCHAR(100))
RETURNS VARCHAR(100)
BEGIN
   WHILE PATINDEX('%[^A-Z -]%', @string) > 0
            SET @string = STUFF(@string, PATINDEX('%[^A-Z -]%', @string), 1, '')--('%[^A-Z ''^-]%'
         WHILE charindex('  ',@string  ) > 0
                  SET @string = replace(@string, '  ', ' ')
         WHILE charindex('--',@string  ) > 0
                  SET @string = replace(@string, '--', '-')
RETURN LTRIM(RTRIM(@string))
END
GO

--keep alphabets only
IF OBJECT_ID(N'$(schema).fnAlphaOnly', N'FN') IS NOT NULL
BEGIN
 DROP FUNCTION $(schema).fnAlphaOnly
END
GO

CREATE FUNCTION [$(schema)].[fnAlphaOnly](@string VARCHAR(100))
RETURNS VARCHAR(100)
BEGIN
   WHILE PATINDEX('%[^A-Z]%', @string) > 0
            SET @string = STUFF(@string, PATINDEX('%[^A-Z]%', @string), 1, '')
RETURN @string
END
```

```
GO

--keep numbers only
IF OBJECT_ID(N'$(schema).fnNumberOnly', N'FN') IS NOT NULL
BEGIN
  DROP FUNCTION $(schema).fnNumberOnly
END
GO

CREATE FUNCTION [$(schema)].[fnNumberOnly](@string VARCHAR(30))
RETURNS VARCHAR(30)
AS
BEGIN
DECLARE @stringint INT
    SET @stringint = PATINDEX('%[^0-9]%', @string)
    BEGIN
        WHILE @stringint > 0
        BEGIN
            SET @string = STUFF(@string, @stringint, 1, '' )
            SET @stringint = PATINDEX('%[^0-9]%', @string )
        END
    END
RETURN RIGHT(ISNULL(@string,0),4)
END
GO

--remove commonly known prefixes
IF OBJECT_ID(N'$(schema).fnRemovePrefix2', N'FN') IS NOT NULL
BEGIN
  DROP FUNCTION $(schema).fnRemovePrefix2
END
GO

CREATE FUNCTION [$(schema)].[fnRemovePrefix2](@Name VARCHAR(100))
RETURNS VARCHAR(100)
BEGIN
    SET @Name=UPPER(LTRIM(RTRIM([$(schema)].[stripDoubleSpaces](@Name))))
    SET @Name = CASE WHEN LEN(@Name)>4 AND LEFT(@Name,5) IN ('MISS ','MRS.
','MISS-','MRS.-' ) THEN RIGHT(@Name, LEN(@Name) - 5)
                    WHEN LEN(@Name)>3 AND LEFT(@Name,4) IN ('MRS ',
'MR. ', 'MS. ', 'DR. ','MRS-', 'MR.-', 'MS.-', 'DR.-') THEN RIGHT(@Name, LEN(@Name) - 4)
                    WHEN LEN(@Name)>2 AND LEFT(@Name,3) IN ('MR ',
'MS ', 'DR ','MR-', 'MS-', 'DR-') THEN RIGHT(@Name, LEN(@Name) - 3)
                ELSE (@Name)
```

```
                         END
RETURN (@Name)
END
GO

--remove commonly known suffixes
IF OBJECT_ID(N'$(schema).fnRemoveSuffix2', N'FN') IS NOT NULL
BEGIN
 DROP FUNCTION $(schema).fnRemoveSuffix2
END
GO

CREATE FUNCTION [$(schema)].[fnRemoveSuffix2](@Name VARCHAR(100))
RETURNS VARCHAR(100)
BEGIN
      SET @Name = CASE WHEN LEN(@Name)>4 AND RIGHT(@Name,4) IN (' III',' 1ST',' 2ND','
3RD',' JR.',' SR.','-III','-1ST','-2ND','-3RD','-JR.','-SR.') THEN
[$(schema)].[fnAlHySpOnly](LEFT(@Name, LEN(@Name) - 4))
                               WHEN LEN(@Name)>3 AND RIGHT(@Name,3) IN (' II','
IV',' VI',' JR',' SR',' MA',' MD','-II','-IV','-VI','-JR','-SR','-MA','-MD')     THEN
[$(schema)].[fnAlHySpOnly](LEFT(@Name, LEN(@Name) - 3))
                               WHEN LEN(@Name)>2 AND RIGHT(@name,2) IN (' I',' V','-
I','-V')   THEN [$(schema)].[fnAlHySpOnly](LEFT(@Name, LEN(@Name) - 2))
                         ELSE [$(schema)].[fnAlHySpOnly](@Name)
                         END
RETURN (@Name)
END
GO

--format date
IF OBJECT_ID(N'$(schema).fnFormatDate', N'FN') IS NOT NULL
BEGIN
 DROP FUNCTION $(schema).fnFormatDate
END
GO

CREATE FUNCTION [$(schema)].[fnFormatDate] (@Datetime DATETIME, @FormatMask
VARCHAR(32))
RETURNS VARCHAR(32)
AS
BEGIN
  DECLARE @StringDate VARCHAR(32)
  SET @StringDate = @FormatMask
  IF (CHARINDEX ('YYYY',@StringDate) > 0)
```

```
    SET @StringDate = REPLACE(@StringDate, 'YYYY',
                DATENAME(YY, @Datetime))

  IF (CHARINDEX ('YY',@StringDate) > 0)
    SET @StringDate = REPLACE(@StringDate, 'YY',
                RIGHT(DATENAME(YY, @Datetime),2))

  IF (CHARINDEX ('Month',@StringDate) > 0)
    SET @StringDate = REPLACE(@StringDate, 'Month',
                DATENAME(MM, @Datetime))

  IF (CHARINDEX ('MON',@StringDate COLLATE SQL_Latin1_General_CP1_CS_AS)>0)
    SET @StringDate = REPLACE(@StringDate, 'MON',
                LEFT(UPPER(DATENAME(MM, @Datetime)),3))

  IF (CHARINDEX ('Mon',@StringDate) > 0)
    SET @StringDate = REPLACE(@StringDate, 'Mon',
                        LEFT(DATENAME(MM, @Datetime),3))

  IF (CHARINDEX ('MM',@StringDate) > 0)
    SET @StringDate = REPLACE(@StringDate, 'MM',
           RIGHT('0'+CONVERT(VARCHAR,DATEPART(MM, @Datetime)),2))

  IF (CHARINDEX ('M',@StringDate) > 0)
    SET @StringDate = REPLACE(@StringDate, 'M',
                CONVERT(VARCHAR,DATEPART(MM, @Datetime)))

  IF (CHARINDEX ('DD',@StringDate) > 0)
    SET @StringDate = REPLACE(@StringDate, 'DD',
                RIGHT('0'+DATENAME(DD, @Datetime),2))

  IF (CHARINDEX ('D',@StringDate) > 0)
    SET @StringDate = REPLACE(@StringDate, 'D',
                        DATENAME(DD, @Datetime))

RETURN @StringDate
END
GO
```

Source Codes: Combing data and hashing

```
/****Hashing Patient identifiers for purpose of matching***/
/****Replace $(Database),$(hashTable),$(temptablename),$(schema),$(sourceTable) with real
values****/
/****Replace/map the fields $(patientid),$(name1),$(name2),$(dob),$(ssn) to real
columns****/

USE $(Database)
GO

DECLARE @privateSalt VARCHAR(30); SET @privateSalt =''; --up to 30 random characters, do
not share this salt
DECLARE @siteid VARCHAR(10); SET @siteid ='Registry';
DECLARE @projectSalt VARCHAR(30); SET @projectSalt ='';--salt used by all sites and registry

IF OBJECT_ID(N'$(hashTable)', N'U') IS NOT NULL
BEGIN
  DROP TABLE $(hashTable)
END;

IF OBJECT_ID(N'tempdb..$(temptablename)', N'U') IS NOT NULL
BEGIN
  DROP TABLE $(temptablename)
END;

SELECT CONCAT(COUNT(*), ' records read')
FROM $(sourceTable)

;with cteclean AS (
SELECT siteid = @siteid
    ,$(patientid) internalid
    ,$(schema).fnRemoveSuffix2($(schema).fnRemovePrefix2($(name1))) name1_0
    ,$(schema).fnRemoveSuffix2($(schema).fnRemovePrefix2($(name2))) name2_0
         ,CASE WHEN $(dob) IN ('') THEN NULL ELSE CAST($(dob) AS DATE) END dob
         ,CASE WHEN $(ssn) IN ('0') THEN NULL ELSE $(schema).fnNumberOnly($(ssn)) END ssn
FROM $(sourceTable)
),

cteunion AS (
SELECT DISTINCT siteid,internalid,$(schema).fnAlphaOnly(name1_0)
name1,$(schema).fnAlphaOnly(name2) name2,
               dob,ssn,CASE WHEN names IN ('name2_1','name2_2') THEN 1 ELSE 0 END
shy_der_flag
```

```
FROM
(
     SELECT siteid
           ,internalid
           ,name1_0
           ,name2_0
           ,CASE WHEN PATINDEX('% %',name2_0)>0  THEN RIGHT(name2_0,
PATINDEX('% %',reverse(name2_0))-1)
                    ELSE NULL
            END name2_1
           ,CASE WHEN PATINDEX('% %',name2_0)>0  THEN LEFT(name2_0, PATINDEX('%
%',(name2_0))-1)
                    ELSE NULL
            END name2_2
           ,dob
           ,CASE WHEN ssn IN ('','0000') OR LEN(ssn)<>4 THEN NULL ELSE ssn END ssn
     FROM cteclean
     WHERE
      name1_0 NOT LIKE '% BOY %' AND
      name1_0 NOT LIKE '% GIRL %' AND
      name1_0 NOT LIKE '% BABY %' AND
      name1_0 NOT LIKE '% TWIN %' AND
  name1_0 NOT LIKE '% BOY' AND
      name1_0 NOT LIKE '% GIRL' AND
      name1_0 NOT LIKE '% BABY' AND
      name1_0 NOT LIKE '% TWIN' AND
      name1_0 NOT LIKE 'BOY %' AND
      name1_0 NOT LIKE 'GIRL %' AND
      name1_0 NOT LIKE 'BABY %' AND
      name1_0 NOT LIKE 'TWIN %' AND
      name2_0 NOT LIKE '% BOY %' AND
      name2_0 NOT LIKE '% GIRL %' AND
      name2_0 NOT LIKE '% BABY %' AND
      name2_0 NOT LIKE '% TWIN %' AND
  name2_0 NOT LIKE '% BOY' AND
      name2_0 NOT LIKE '% GIRL' AND
      name2_0 NOT LIKE '% BABY' AND
      name2_0 NOT LIKE '% TWIN' AND
      name2_0 NOT LIKE 'BOY %' AND
      name2_0 NOT LIKE 'GIRL %' AND
      name2_0 NOT LIKE 'BABY %' AND
      name2_0 NOT LIKE 'TWIN %'
) AS cp
UNPIVOT
```

```
(
 name2 FOR names IN (name2_0,name2_1,name2_2)
) AS up
)

SELECT * INTO $(temptablename)
FROM cteunion
WHERE name1 NOT IN

('UNKNOWN','MALE','FEMALE','BABY','BOY','GIRL','TWINA','TWINB','TWIN','JOHNDOE','JANEDOE',
'UNK','TRA','UNKTRA','UNKTRAUMA','UNKNOWNTRAUMA','TRAUMA','PMCERT','UNTRA','PMCERT','') AND
		name1 IS NOT NULL AND LEN(name1)>1 AND
    name2 NOT IN

('UNKNOWN','MALE','FEMALE','BABY','BOY','GIRL','TWINA','TWINB','TWIN','JOHNDOE','JANEDOE',
'UNK','TRA','UNKTRA','UNKTRAUMA','UNKNOWNTRAUMA','TRAUMA','PMCERT','UNTRA','PMCERT','') AND
		name2 IS NOT NULL AND LEN(name2)>1 AND
    dob IS NOT NULL

CREATE NONCLUSTERED INDEX ix_0 ON $(temptablename) (internalid);
CREATE NONCLUSTERED INDEX ix_1 ON $(temptablename) (name1);
--CREATE NONCLUSTERED INDEX ix_2 ON $(temptablename) (name2);
CREATE NONCLUSTERED INDEX ix_3 ON $(temptablename) (dob);
CREATE NONCLUSTERED INDEX ix_4 ON $(temptablename) (ssn);
--CREATE NONCLUSTERED INDEX cx_123 ON $(temptablename) (name1,name2,dob);
--CREATE NONCLUSTERED INDEX cx_213 ON $(temptablename) (name2,name1,dob);
--CREATE NONCLUSTERED INDEX cx_1234 ON $(temptablename) (name1,name2,dob,ssn);
--CREATE NONCLUSTERED INDEX cx_2134 ON $(temptablename) (name2,name1,dob,ssn);

SELECT CONCAT(COUNT(DISTINCT internalid), ' records met criteria')
FROM $(temptablename)

SELECT * INTO $(hashTable) FROM (
SELECT
siteid
,internalid
,PIDHASH = $(schema).fnHashBytes2(CONCAT(internalid,siteid),@privateSalt)
--,PIDHASH = $(schema).fnHashBytes2(CONCAT(internalid,siteid,datediff(dd,dob,'$(privateDate)')),@privateSalt)
```

```
,fnamelnamedobssn = CASE WHEN ssn IS NULL THEN CONVERT(VARCHAR(128), NULL)
                  ELSE $(schema).fnHashBytes2(CAST(CONCAT(name1,name2,dob,ssn) as
varchar(max)),@projectSalt)
                                                END
,lnamefnamedobssn = CASE WHEN ssn IS NULL THEN CONVERT(VARCHAR(128), NULL)
                  ELSE $(schema).fnHashBytes2(CAST(CONCAT(name2,name1,dob,ssn) as
varchar(max)),@projectSalt)
                                                END
,fnamelnamedob = $(schema).fnHashBytes2(CAST(CONCAT(name1,name2,dob) as
varchar(max)),@projectSalt)
,lnamefnamedob = $(schema).fnHashBytes2(CAST(CONCAT(name2,name1,dob) as
varchar(max)),@projectSalt)
,fnamelnameTdobssn = CASE WHEN ssn IS NULL THEN CONVERT(VARCHAR(128), NULL)
                   ELSE
$(schema).fnHashBytes2(CAST(CONCAT(name1,name2,$(schema).fnFormatDate(dob,'YYYY-DD-
MM'),ssn) as varchar(max)),@projectSalt)
                END
,fnamelnameTdob =
$(schema).fnHashBytes2(CAST(CONCAT(name1,name2,$(schema).fnFormatDate(dob,'YYYY-DD-
MM')) as varchar(max)),@projectSalt)
,fname3lnamedobssn = CASE WHEN ssn IS NULL OR shy_der_flag=1 THEN
CONVERT(VARCHAR(128), NULL)
                   ELSE
$(schema).fnHashBytes2(CAST(CONCAT(substring(name1,1,3),name2,dob,ssn) as
varchar(max)),@projectSalt)
                                                 END
,fname3lnamedob = CASE WHEN shy_der_flag=1 THEN CONVERT(VARCHAR(128), NULL)
                 ELSE $(schema).fnHashBytes2(CAST(CONCAT(substring(name1,1,3),name2,dob) as
varchar(max)),@projectSalt)
              END
,fnamelnamedobDssn = CASE WHEN ssn IS NULL THEN CONVERT(VARCHAR(128), NULL)
                   ELSE
$(schema).fnHashBytes2(CAST(CONCAT(name1,name2,dateadd(dd,1,dob),ssn) as
varchar(max)),@projectSalt)
                                                 END
,fnamelnamedobYssn = CASE WHEN ssn IS NULL THEN CONVERT(VARCHAR(128), NULL)
                   ELSE
$(schema).fnHashBytes2(CAST(CONCAT(name1,name2,dateadd(YYYY,1,dob),ssn) as
varchar(max)),@projectSalt)
                                                 END
FROM $(temptablename)
) t1
```