Account Matching - How to Know Who’s Who
(or Will the Real John Smith Please Stand Up?)

IAM Online
Wednesday, August 8, 2018

Benn Oshrin, Spherical Cow Group
Summer Scanlan, University of California, Berkeley
Keith Wessel, University of Illinois Urbana Champaign
What is Identity Matching?

- Not the same as identity linking, but one can lead to the other
- Answers the question: are A and B the same person?
- Useful for on-boarding or later if identity information is changed
- Harder today than it used to be with information being more sensitive
Why Bother With Identity Matching?

- Prevent someone from having multiple identities
- For users with multiple roles, it can be important for everything to be under one identity
- Fewer passwords and MFA tokens to juggle
- Better experience than waiting for the user to contact the help desk to ask for a merge
- But be careful: a false match can create a tangled mess!
Matching Overview & History
Benn Oshrin, Spherical Cow Group
First, Some Background

- This problem has been around for a while
  - Way, way back (in the mid-90s) campuses started to set up email for everyone
    - (Similar problem for ID badges, etc)
  - Problem: Who is everyone?
    - HR knows employees
    - Registrar knows students
  - Multiple Systems of Record (SORs) Make Higher Ed Special*
    - *For identity management purposes, anyway
Today, It's Basically The Same Problem

● Typical Higher Ed Scenarios
  ○ Students apply via commercial application
  ○ Students enroll via Registrar
  ○ Students become alumni and are tracked via Alumni Relations
  ○ Employees are hired through HR
    ■ But only after some early onboarding process
  ○ Guests and affiliates come from everywhere
  ○ And what about the Hospital?

● Need a single identity to properly manage access to resources
Multiple Approaches, Common Themes

- **Variation 1: Match at Registry**
  - Absent other considerations, probably the recommended approach
- **Variation 2: Match at SOR ("Standalone")**
- **Variation 3: Match before SOR**
  - Enrollee obtains unique ID before approaching SOR
- Regardless of variation, similar considerations
  - Quality of inbound attributes
  - Handling ambiguous ("fuzzy", "potential") matches
Solutions (?)

- Lots of custom (legacy) code
- Some commercial products
  - Expensive
  - Some better than others
- Not much in the way of Open Source
  - Maybe you can hack something together from MDM solutions
  - … Until now (well, soon) …
Two Parts To A Solution

1. ID Match API
   ○ Defines how match requests and responses are exchanged
     i. Match Request
     ii. Match Response (exact, potential)
     iii. Pending Matches (for review and resolution)
     iv. Update Match Attributes
   ○ Design preference: JSON + REST
   ○ Goal: Assign a "Reference Identifier"
     i. Unique identifier for a person, as defined by the Match Engine
   ○ (More later…)
Two Parts To A Solution

2. Match Engine
   ○ Implements rules for performing searches
     i. Define attribute characteristics
     ii. "Canonical" vs "Potential" rules
        1. Canonical: Can uniquely identify a person, processing stops if exact match found
        2. Potential: Can suggest ambiguous or fuzzy matches, processing does not stop even if a single match is found
   ○ Maintains match state
     i. Implies a need for attribute updates (such as name changes) to be reported to the Match Engine
Matching: A Community Timeline

- 2011: Initial ID Match Strawman drafted
- 2012: "CIFER" Strawman API drafted
- 2013: UCB develops Java-based in memory solution
- 2014: UCB develops Postgres-based solution (PoC)
- 2015: UCB internal implementation
- Nov 2017: TIER project funding allocated
- Summer 2018: Initial TIER component releases
  - Code being developed under the COmanage Project
  - API being formalized by TIER API/Reg Working Group
Matching @ UC Berkeley
Summer Scanlan, University of California, Berkeley
Identity Matching at UC Berkeley

System Elements

- Berkeley Person Registry (BPR)
- Systems of Record (SOR)
- Primary Key
- UID
- Raw data (sorObject)
- Match Engine
- Canonical Match Rules
- Potential Match Rules
- Partial Match Table
Record Provisioning and Auto Matching in BPR

- Data arrives from SORs each morning or via message queue
- Incoming new and update records are checked for an existing primary key
  - If the primary key matches an existing record, the existing record is updated
  - If the primary key does not already exist, a new sorObject is created, which is then sent through the match engine
Example of Auto Match

Incoming record:
Robert Jones
SSN: 12345
DOB: 01/01/2000
EID: 011223345
Role: Employee

Existing record:
Robert Jones
SSN: 12345
DOB: 01/01/2000
SID: 3031231231
Role: Student

Matched Record:
Robert Jones
SSN: 12345
DOB: 01/01/2000
EID: 011223345
SID: 3031231231
Roles: Employee, Student
Match Engine

- Matchable elements are extracted from the sorObject
- Rest call is made to the match service
- Match engine goes through matching rules, starting with canonical rules
- Records that meet canonical criteria are matched, and UID is reprovisioned with updated identity data
Partial Match Table

- Records go through the potential match rules after canonical rules.
- Records that are a potential match are sent to the partial match table for human review.
- CalNet staff reviews raw data for additional matching data elements.
  - Although the match engine is really quite helpful, human review is sometimes still required -- matching is hard!
- If no canonical or potential match is made, a new UID is provisioned.
### Example of Partial Match Table

#### Record to Match

<table>
<thead>
<tr>
<th>System Of Record</th>
<th>ADVCON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sor Primary Key</td>
<td></td>
</tr>
<tr>
<td>Given Name</td>
<td>Summer</td>
</tr>
<tr>
<td>Sur Name</td>
<td>Scanlan</td>
</tr>
<tr>
<td>Full Name</td>
<td>Summer Scanlan</td>
</tr>
<tr>
<td>Date Of Birth</td>
<td>Match</td>
</tr>
</tbody>
</table>

#### Probable candidate (UID: 72515)

Match rules used: "Potential #1", "Potential #2"

### Basic Information

<table>
<thead>
<tr>
<th>Name:</th>
<th>Summer S Scanlan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Affiliations:</td>
<td>FORMER-STUDENT, AFFILIATE-TYPE-ADVCON-ALUMNUS, AFFILIATE-TYPE-ADVCON-CAA-MEMBER</td>
</tr>
<tr>
<td>DoB:</td>
<td></td>
</tr>
<tr>
<td>Advcon Id:</td>
<td></td>
</tr>
</tbody>
</table>

Match
Match Rules Improvement

- After implementation, we analyzed the partial match table and duplicate records to find additional possible match rules.
- Updating rules requires a developer to update the configuration inside the match engine.
- We regularly review our systems in the hopes of continuously improving them.
Gotchas!

Examples:

- Potential Match Rule - arguably a “bad” rule
- Canonical Match Rule - a “good” rule that occasionally results in identity collision
Example of Potential Match Rule

<table>
<thead>
<tr>
<th>System Of Record</th>
<th>ADVCON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sor Primary Key</td>
<td></td>
</tr>
<tr>
<td>Given Name</td>
<td>Summer</td>
</tr>
<tr>
<td>Sur Name</td>
<td>Smits</td>
</tr>
<tr>
<td>Full Name</td>
<td>Smits, Summer</td>
</tr>
<tr>
<td>Date Of Birth</td>
<td>Match</td>
</tr>
</tbody>
</table>

Probable candidate (UID: 248574)
Match rules used: "Potential #2"

**Basic Information**

<table>
<thead>
<tr>
<th>Name:</th>
<th>Sam Smith</th>
</tr>
</thead>
<tbody>
<tr>
<td>Affiliations:</td>
<td>FORMER-STUDENT</td>
</tr>
<tr>
<td>DoB:</td>
<td>Match</td>
</tr>
</tbody>
</table>

Probable candidate (UID: 234141)
Match rules used: "Potential #8"

**Basic Information**

<table>
<thead>
<tr>
<th>Name:</th>
<th>Summer Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Affiliations:</td>
<td>FORMER-STUDENT, FORMER-EMPLOYEE</td>
</tr>
<tr>
<td>DoB:</td>
<td>Match</td>
</tr>
</tbody>
</table>
Example - Potential Match Rule

total potential givenName: SUBSTRING, surName: DISTANCE, dateOfBirth: EXACT

- Matches are difficult for records that have multiple first or last names
- This rule was meant to capture such cases
- The actual result of implementation is hundreds of non matching records hitting the partial match table each month, and having to be manually reviewed and provisioned
- It often feels like a bad rule, but it also finds matches that would otherwise be provisioned as duplicates!
Gotcha #2: Canonical Rule

canonical givenName: SUBSTRING, surName: EXACT, socialSecurityNumber: EXACT, socialSecurityNumberType: FIXED_VALUE

- Records matching on first, last, and last five of SSN are considered a canonical match
- This rule resulted in 3 identity collisions last year (out of 50,000 records provisioned)
- Luckily, these are found right easily repaired
Summary

● Matching is not easy
● Having a match engine is definitely helpful
● Analyzing your potential matches and cases of identity collision makes for a better match engine
TIER Match API & Component
Benn Oshrin, Spherical Cow Group
The ID Match API

- RESTful design
- Goal: Obtain a Reference Identifier
- Can operate synchronously or asynchronously
  - i.e.: interactive fuzzy resolution, or queue for an admin
- Can be placed behind a Registry or as a standalone service
- Can be used to transition legacy systems
ID Match API Status

- Strawman stable for quite some time
- Effort starting to "formalize" specification
  - eduPerson style, not as an RFC/etc
    - Goal: Ready for TechEx
  - Long term home TBD
  - Attribute names may change slightly from the examples
ID Match Component Initial Scoping

- **UI Driven Configuration**
  - Includes ambiguous match resolution
- **ID Match API Support**
- **Postgres Only**
  - Possible MySQL/MariaDB support later
- **Multi-tenant**
- **Distance and Substring Matching**
  - Dictionary and others later
TIER ID Match Component Status

- Being developed as part of COmanage Project
  - Does not require COmanage Registry
- Initial coding complete
  - Early access releases RSN
  - v1.0.0 by TechEx
- Documentation underway
- Up next:
  - UI
  - Packaging
Match Engine Configuration

- Platform Configuration
  - Matchgrids
  - Permissions

- Matchgrid Management
  - matchgrid_01 (Build)
    - Attribute Groups
    - Attributes
    - Rules
    - Systems of Record
    - Reconcile Unresolved Requests

  - testgrid (Build)
    - Attribute Groups
    - Attributes
    - Rules
    - Systems of Record
    - Reconcile Unresolved Requests
Matchgrid Configuration

<table>
<thead>
<tr>
<th>Table Name *</th>
</tr>
</thead>
<tbody>
<tr>
<td>testgrid</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid for basic configuration testing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Status *</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reference ID Assignment Method *</th>
</tr>
</thead>
<tbody>
<tr>
<td>UUID (Type 4)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reference ID Initial Value (sequence only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
</tr>
</tbody>
</table>
### Systems of Record

#### Add New System of Record

<table>
<thead>
<tr>
<th>Label</th>
<th>Resolution Mode</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>sis</td>
<td>External</td>
<td>Edit Delete</td>
</tr>
<tr>
<td>hrms</td>
<td>Interactive</td>
<td>Edit Delete</td>
</tr>
<tr>
<td>guest</td>
<td>External</td>
<td>Edit Delete</td>
</tr>
</tbody>
</table>
### Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Attribute Group</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>dob</td>
<td></td>
<td>Edit Delete</td>
</tr>
<tr>
<td>ssn</td>
<td></td>
<td>Edit Delete</td>
</tr>
<tr>
<td>firstname</td>
<td>1</td>
<td>Edit Delete</td>
</tr>
<tr>
<td>lastname</td>
<td>1</td>
<td>Edit Delete</td>
</tr>
<tr>
<td>netid</td>
<td></td>
<td>Edit Delete</td>
</tr>
</tbody>
</table>
Date of Birth
Configuration

Name *

dob

Description

API Name *

dateOfBirth

- Alphanumeric
- Case Sensitive
- Invalidates
- Null Equivalents
- Required

Search Distance
2

- Search Exact
New Match Request (SIS)

**URL:** https://valkyrie.local/match/api/3/v1/people/sis/368324971

**Method:** PUT

```json
{
    "sorAttributes": {
        "names": [
            {
                "type": "official",
                "given": "Jay",
                "family": "Clark"
            },
            {
                "dateOfBirth": "1999-08-23",
                "identifiers": [
                    {
                        "type": "national",
                        "identifier": "995005320"
                    }
                ]
            }
        ]
    }
}
```
Match Result (SIS)

{i 1 : "referenceId": "3965572c-e900-4afd-ad07-a13d0cd2e0ee"}
## Demo Match Rules

<table>
<thead>
<tr>
<th></th>
<th>Rule C1</th>
<th>Rule C2</th>
<th>Rule P1</th>
<th>Rule P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoB</td>
<td>Exact</td>
<td>Exact</td>
<td>Distance (2)</td>
<td>Exact</td>
</tr>
<tr>
<td>SSN</td>
<td>Exact</td>
<td>Skip</td>
<td>Distance (2)</td>
<td>Skip</td>
</tr>
<tr>
<td>First</td>
<td>Substring (1,3)</td>
<td>Skip</td>
<td>Substring (1,3)</td>
<td>Substring (1,3)</td>
</tr>
<tr>
<td>Last</td>
<td>Exact</td>
<td>Exact</td>
<td>Distance (2)</td>
<td>Distance (2)</td>
</tr>
<tr>
<td>NetID</td>
<td>Skip</td>
<td>Exact</td>
<td>Skip</td>
<td>Exact</td>
</tr>
</tbody>
</table>
New Match Request (HRMS)

URL: https://valkyrie.local/match/api/3/v1/people/hrms/H921691951

Method: PUT

```
1  { "sorAttributes":
2    { "names": [ 
3      { "type": "official", 
4        "given": "Jay", 
5        "family": "Clark"
6      },
7    ],
8    "dateOfBirth": "1999-08-23",
9    "identifiers": [ 
10      { "type": "national", 
11        "identifier": "995009302"
12      }
13    ]
14  } }
```
Fuzzy Match (300) Result (HRMS)

```json
{
  "candidates": [
    {
      "referenceId": "3965572c-e900-4afd-ad07-a13d0cd2e0ee",
      "attributes": [
        {
          "matchRequest": 21,
          "sor": "sis",
          "identifiers": [
            {
              "type": "sor",
              "identifier": "368324971"
            },
            {
              "type": "national",
              "identifier": "995005320"
            }
          ],
          "dateOfBirth": "1999-08-23",
          "names": [
            {
              "family": "Clark",
              "given": "Jay",
              "type": "official"
            }
          ]
        }
      ]
    }
  ]
}
```
Fuzzy Match Result

- Because the SoR is "interactive", there is no pending request for the Match Administrator to review
  - Option 1: Notice correct identifier and resubmit
  - Option 2: Submit "Forced Reconciliation Request" with appropriate Reference Identifier
Performance Considerations

● Search times vary according to
  ○ Configured attributes
  ○ Quantity and complexity of confidence rules
  ○ Number of records in the database

● Exact $\sim= (\text{Exact} + \text{Fuzzy}) < \text{Fuzzy}$
  ○ Exact matches can partition the search space for fuzzy matches

● Need to optimize for initial load of already-matched data
More Info

- ID Match Strawman API: https://spaces.at.internet2.edu/x/2QL5AQ
  ○ Will move to https://github.internet2.edu soon

- ID Match Component: https://spaces.at.internet2.edu/x/qwW6Bw
  ○ (In progress, content coming soon)
Please evaluate today’s session

https://www.surveymonkey.com/r/IAMOnline-Aug2018
2018 Internet2 Technology Exchange (TechEx)
October 15-19, 2018 - Orlando, Florida
https://meetings.internet2.edu/2018-technology-exchange/

- Two full tracks for Trust and Identity topics
- Advance CAMP (ACAMP)
- Pre-meeting tutorials
Next IAM Online
Wednesday, September 12, 2018
2 pm ET | 1 pm CT | Noon MT | 11 am PT

Grouper 2.4 and the Grouper Deployment Guide

- Features of the latest Grouper release (2.4)
- The Grouper Deployment Guide, NIST 800-162, access models, and how that comes together in a TIER based architecture.