Match Rules in Reltio MDM

Data Sheet | LumenData

A comprehensive overview of the powerful matching capabilities within the Reltio platform and how they enable accurate data matching and deduplication.

Configuring Match Rules in MDM

Match - Reltio matching is case-insensitive. Match is a process that determines if an entity is semantically the same as another entity.

Match Rule Types in Reltio

Match Rule Types

Rule TYPEs and their ability to issue directives to the match engine

type	rule formula	Can issue a directive of 'merge'	Can issue a directive of 'queue for review'	Can publish an event of 'auto link'	Can publish an event of 'potential link'
automatic	boolean	Yes	No	No	No
suspect	boolean	No	Yes	No	No
<your rule name>	boolean	Yes	Yes	Yes	Yes
relevance_based	arithmetic	Yes	Yes	Yes	Yes

matchGroups section of the L3 file

- matchGroups section is required in its L3 configuration if we need match & merge for an entity type.
- The matchGroups section can have as many match groups as we need.
- On average an entity type will use 3 to 7 match groups.
- Each match group consists of a single match rule.
- A match rule consists of 3 parts -

  1. Comparison formula
  2. Comparator class
  3. Tokenization class

- Comparison Operators are high-level.
- Actual comparison is dictated by the comparator class we choose.
- Generally, a match rule whose comparison formula evaluates to true for a given pair of records will produce a directive of “merge” or “queue for review” depending on the match rule type.

Below are the different components present in matchGroups section

```
"matchGroups": [
    {
    {
    {
        "uri": "configuration/entityTypes/Contact/matchGroups/Rule3",
        "label": "Fuzzy(Full Name) Exact(Phone)",
        "type": "suspect",
        "scope": "INTERNAL",
        "useOvOnly": "true",
        "rule": {
            "exact": [
                "configuration/entityTypes/Contact/attributes/Phone/attributes/Number"
            ],
            "fuzzy": [
                "configuration/entityTypes/Contact/attributes/FullName"
            ],
            "matchTokenClasses": {
            "comparatorClasses": {
            "and": {
        },
        "scoreStandalone": 10,
        "scoreIncremental": 10
    }
],
```

Contact

matchgroups

matchGroup1

Type: suspect if TRUE, sends directive of 'queue for review'

MatchGroup2

Type: suspect if TRUE, sends directive of 'queue for review'

MatchGroup3

Type: suspect if TRUE, sends directive of 'queue for review'

MatchGroup4

Type: automatic if TRUE, sends directive of 'merge"

MatchGroup5

Type: automatic if TRUE, sends directive of 'merge"

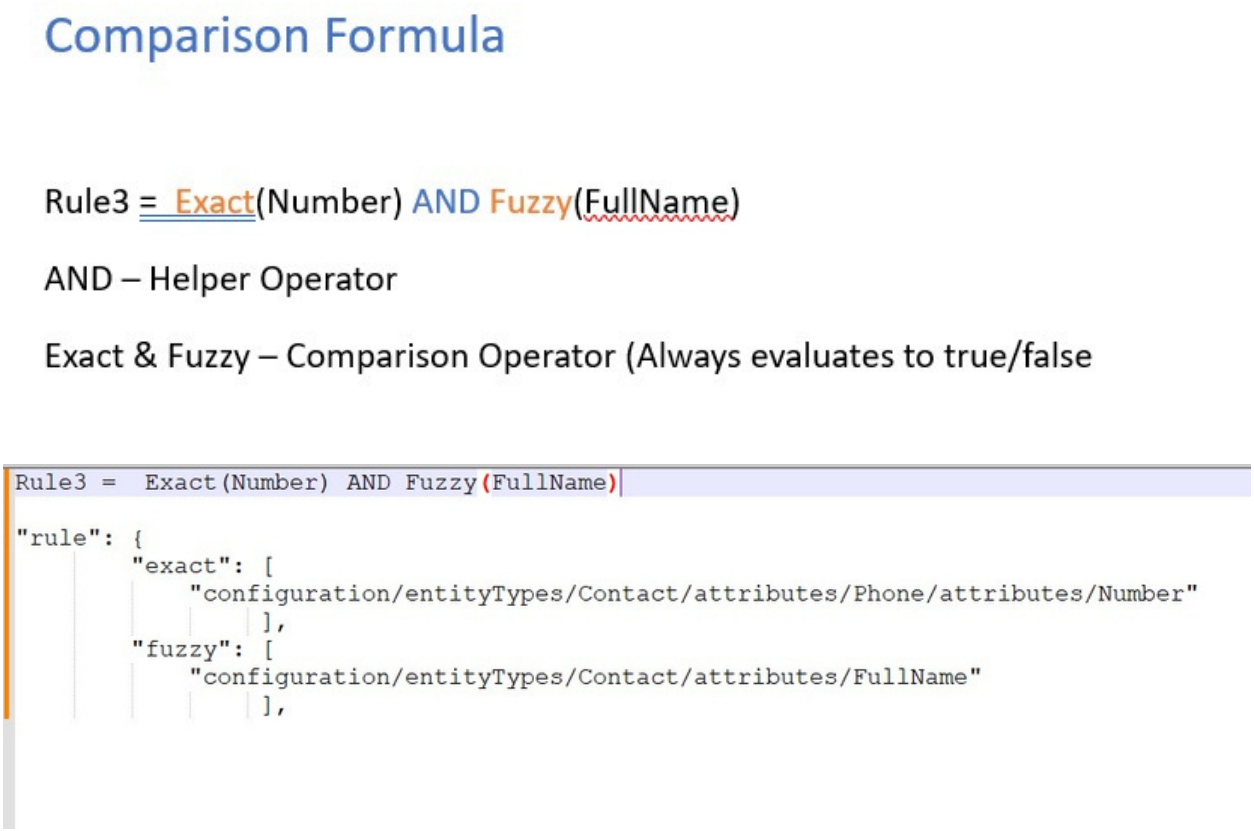
MatchGroup6

Type: relevance_based if score>.7, sends directive of 'merge"

MatchGroup7

Type: custom_rule1 if TRUE, sends event to external queue

Comparison Formula in Reltio Match Rule




```
Rule3 =  Exact(Number) AND Fuzzy(FullName)

"rule": {
        "exact": [
            "configuration/entityTypes/Contact/attributes/Phone/attributes/Number"
                ],
        "fuzzy": [
            "configuration/entityTypes/Contact/attributes/FullName"
                ],
```

There are 5 comparison Operators

- Fuzzy
- Exact
- ExactOrNull
- ExactOrAllNull
- NotExactSame

Each time we specify an operator that has one of these five, we must then additionally map 1 of 16 comparator classes to the corresponding attribute that more precisely specifies what we mean by “Exact” or “Fuzzy”.

The mapping of comparator classes in JSON form looks like this:

```
"and": {
  "fuzzy": [
    "configuration/entityTypes/Contact/attributes/FirstName",
    "configuration/entityTypes/Contact/attributes/LastName",
    "configuration/entityTypes/Contact/attributes/Addresses/attributes/AddressLine1"
  ],
  "exactOrNull": [
    "configuration/entityTypes/Contact/attributes/Phone/attributes/Number"
  ]
},

  "comparatorClasses": {
    "mapping": [
      {
        "attribute": "configuration/entityTypes/Contact/attributes/FirstName",
        "class": "com.reltio.match.comparator.BasicStringComparator"
      },
      {
        "attribute": "configuration/entityTypes/Contact/attributes/LastName",
        "class": "com.reltio.match.comparator.BasicStringComparator"
      }
    ]
```

Out of the box comparator classes:

- AddressLineComparator
- BasicStringComparator
- BasicTokenizedOrganizationNameComparator
- DamerauLevenshteinDistance
- DistinctWordsComparator
- DynamicDamerauLevenshteinDistance
- DoubleMetaphoneComparator
- MetaphoneComparator
- OrganizationNamesComparator
- PhoneNumberComparator
- RangeNumericComparator
- SoundexComparator
- StringCharactersComparator
- StringComparatorIgnoringNulls
- ProximateGeoComparator
- CrossMultiComparator
- ExactOrNullCrossMultiComparator
- (custom comparator)

There are 7 Helper Operators

- Equals
- NotEquals
- In
- NullValues
- And, Or, Not

Comparison Operators - Fuzzy

Fuzzy evaluates to true if the values being compared are deemed semantically the same, even if they are not identically the same.

Example-

"John" vs "Jon"

"ABC Ltd" vs "ABC Limited“

Comparators used

1. PhoneNumberComparator
2. AddressLineComparator
3. MetaphoneComparator
4. SoundexComparator
5. DamerauLevenshteinComparator
6. OrganizationNamesComparator
7. RangeNumericComparator

LUMENDATA

Comparison Operators - Exact

Exact evaluates to true if the values being compared are identically the same.

Example-

"John" vs “john"

"ABC Ltd" vs "ABC ltd“

Comparators used

1. BasicStringComparator

LUMENDATA

Comparison Operators - ExactOrNull

ExactOrNull evaluates to true based on the same principle as Exact, but allows either value to be absent.

Example-

"John" vs “john"

“John" vs <null>

<null> vs “john”

Comparators used

1. BasicStringComparator

Typically used for the attributes that are unreliably populated, and not extremely important if used in a suspect rule Ex –Suffix, Gender

LUMENDATA

Comparison Operators - ExactOrAllNull

ExactOrAllNull evaluates to true based on the same principle as Exact, but also requires that both values are present or both values are missing.

Example-

"John" vs “john"

<null> vs <null>

Comparators used

1. BasicStringComparator

Typically used in an automatic rule where the attribute is unreliably populated but if it exists in one record, it is required to exist in other record too for comparison.

Ex. – Suffix, Gender

LUMENDATA

Comparison Operators - NotExactSame

NotExactSame evaluates to true if the two values compared are not the same..

Example-

"John" vs “Sr"

“88899” vs “88898”

Comparators used

1. BasicStringComparator
2. PhoneNumberComparator

Typically used in a negative rule to downgrade the outcome of the ‘automatic’ rule to that of the ‘suspect’ rule. Ex. – Suffix, Gender

LUMENDATA

Helper Operators - Equals

Equals requires that an attribute in the record, but not necessarily in the rule, is equal to a specific value.

Example-

Country = “India”

Gender = “Male”

Comparators used

N/A

Typically used to restrict the rule to operate only on a chosen subset of the records

LUMENDATA

Helper Operators - NotEquals

NotEquals requires that an attribute in the record, but not necessarily in the rule, is NOT equal to a specific value.

Example-

Country NotEquals “India”

Gender NotEquals “Male”

Comparators used

N/A

Typically used to restrict the rule to operate only on a chosen subset of the records

LUMENDATA

Helper Operators - In

In requires that an attribute in the record, but not necessarily in the rule, is equal to any of a list of values.

Example-

Country IN [“India”, “UK’, “USA”]

Comparators used

N/A

Typically used to restrict the rule to operate only on a chosen subset of the records

LUMENDATA

Helper Operators - NullValues

For the purpose of matching NullValues, sets the values of the declared string to null.

Example-

1. Assume any occurrence of “9999-999-999” is <null>
2. Assume any occurrence of “Not Specified” is <null>

Comparators used

N/A

Typically used in conjunction with the ExactOrNull or ExactOrAllNull operators. Ex – Consider one record with attribute “8876-994-556” and other record with “9999-999-999”. And the rule contains ExactOrNull(Phone). This would evaluate to True.

LUMENDATA

Helper Operators – And, Or, Not

For The helper operators support complex Boolean logic with the other operators

Example-

(Fuzzy(FirstName) OR Fuzzy(LastName) AND Not(Exact(Gender))

Comparators used

N/A

LUMENDATA

Custom rule type (<your rule name>)

Classes that can be used within the matchAction object:

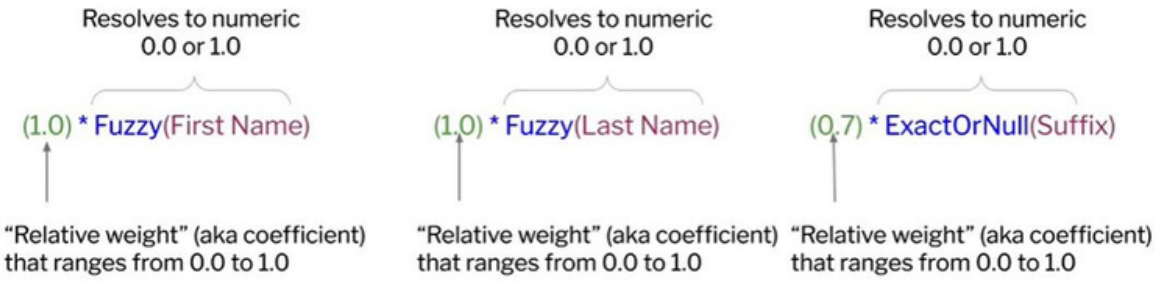
- AutoMergeHandler (issues a directive to merge the profile pair)
- SuspectMatchHandler (issues a directive to queue the profile pair for review
- StreamingPotentialLinkActionHandler (publishes the event, "POTENTIAL_LINK_FOUND")
- StreamingAutoLinkActionHandler (publishes the event, "AUTO_LINK_FOUND")

Notes:

POTENTIAL_LINK_FOUND is designed to suggest creating a relationship between the records

AUTO_LINK_FOUND is designed to strongly suggest creating a relationship between the records

relevance_based rule type



- Relevance-based rule type uses arithmetic scoring of the candidate pair instead of Boolean.
- It can reduce the complexity and overall number of rules you need because the directives of merge and queue for review which normally require a combination of 'automatic' and 'suspect' rules can often be represented by separate thresholds within a single relevance-based rule.

- It supports match actions that you can define, to achieve additional directives or combinations of events.
- We assign a relative weight (0.0 to 1.0) to each attribute to make some attributes more important than others in the calculation.

relevance_based rule type


```
"type": "relevance_based",
    "rule": {
      "and": {
        "exact": [
          "configuration/entityTypes/HCP/attributes/FirstName",
          "configuration/entityTypes/HCP/attributes/LastName"
        ],
        "exactOrAllNull": [
          "configuration/entityTypes/HCP/attributes/Suffix"
        ]
      },
      "weights": [
        {
          "attribute": "configuration/entityTypes/HCP/attributes/Suffix",
          "weight": 0.7
        }
      ],
      "actionThresholds": [
        {
          "type": "auto_merge",
          "threshold": "0.8-1.0"
        },
        {
          "type": "potential_match",
          "threshold": "0.4-0.8"
```

ABOUT LUMENDATA:

LumenData is a leading provider of Enterprise Data Management, Cloud & Analytics solutions. We help businesses navigate their data visualization and analytics anxieties and enable them to accelerate their innovation journeys. Founded in 2008, with locations in multiple countries, LumenData is privileged to serve over 100 leading companies, including KwikTrip, Versant Health, US Food & Drug Administration, US Department of Labor, Cummins Engine, BCG, and others. LumenData is SOC2 certified and has instituted extensive controls to protect client data, including adherence to GDPR and CCPA regulations.

Get in touch to discuss how we can facilitate data-driven transformation for your organization.

MEET OUR AUTHORS

Mohd Imran
Senior Consultant