ORACLE

ORACLE

Oracle Enterprise Data Quality

Matching Essentials

Product Development

What is Matching?

Matching allows you to identify records which may relate to a single entity for the purpose of running your business.

CU_NO	CU_ACCOUNT	TITLE	NAME	ADDRESS1	ADDRESS2	ADDRESS3	POSTCODE
10782	95-15134-SH	Mr	Victor CARSON	1A Spire Road, Glover Road Est East	Washington	NE37 3ES	
15906	98-21229-PB	Ms	J CARSON	Spire Road, Glover Estate East	WASHINGTON	Tyne & Wear	NE37 3ES

Should we treat these two customers as one?

COMPANY_NAME
DIRECT LINE GROUP SERVICES
DIRECT LINE GROUP SERVICES LIMITED
DIRECT LINE GROUP SERVICES LTD
DIRECT LINE FINANCIAL SERVICES

How many different companies are there here?

What Makes Matching Difficult?

Free text fields allow users to enter data in different formats and using different conventions

Source data is likely to be incomplete and/or incorrect.
Cannot assume that data will be in the "right" place

CU_NO	CU_ACCOUNT	TITLE	NAME	ADDRESS1	ADDRESS2	ADDRESS3	POSTCODE
10782	95-15134-SH	Mr	Victor CARSON	1A Spire Road, Glover Road Est East	Washington	NE37 3ES	
15906	98-21229-PB	Ms	J CARSON	Spire Road, Glover Estate East	WASHINGTON	Tyne & Wear	NE37 3ES

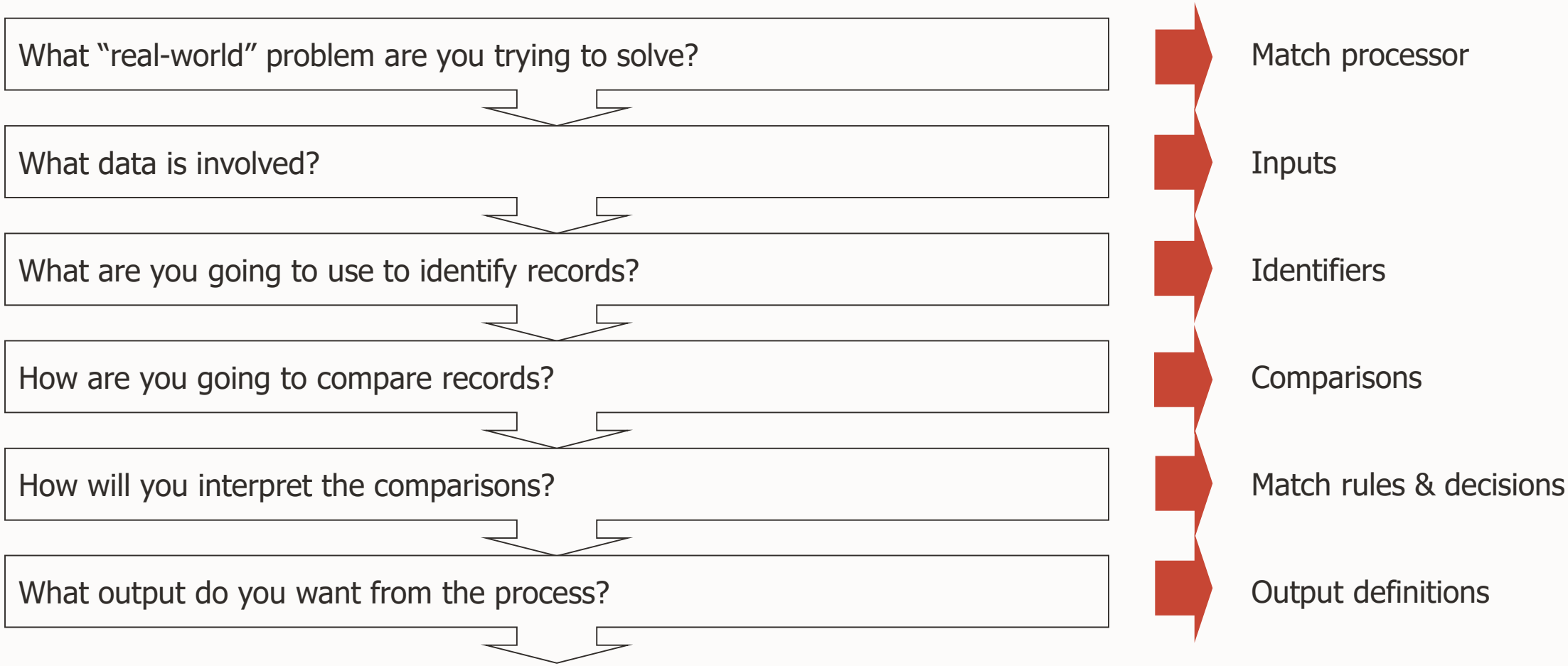
Matching configuration must capture and replicate the users' knowledge and experience together with the business rules surrounding the data.

Context is critical.
Knowledge that these 4 fields represent an address allows us to see they may be a match.

Matching Scenarios

- **De-duplication**
  - Find & remove duplicate entries in a system
- **Consolidation**
  - Combine a number of systems, eliminating duplicates and creating the “best” records
- **Enhancement**
  - Improve data by bringing in trusted reference data
- **Linking**
  - Establishing links between multiple data sets

The Matching Process



Matching in Oracle Enterprise Data Quality

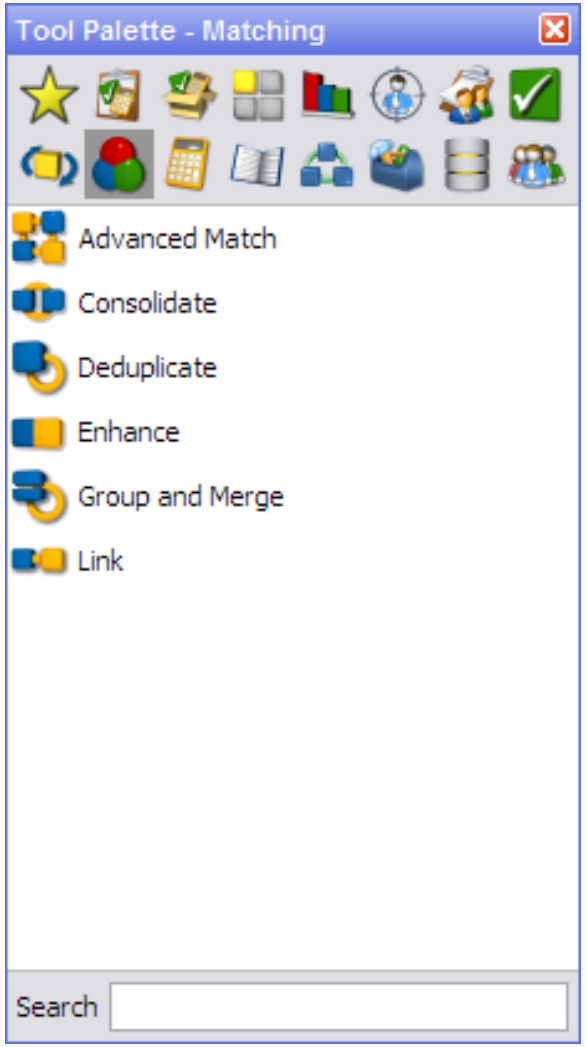
Before We Can Start Matching

- **What problem are we trying to solve?**
  - Do we have a business definition of a match?
  - What output do we want?
- **What data can we use to find matches?**
  - Profile each dataset to establish quality/reliability of identifying information
    - Names, Addresses, Part descriptions
  - What sort of variability is in the data?
    - Mis-fielding? Typos? Missing data?

The Matching Palette

Each processor solves a specific problem:

- **Advanced Match gives you control over all the configuration options**
- **Consolidate matches and merges multiple datasets**
- **Deduplicate matches a single dataset**
- **Enhance adds information from one or more reference data sets**
- **Group & Merge gives a simple merge based on exact match only**
- **Link find matches between datasets but doesn't merge the data**



The Matching Sub-processors

Icon	Sub-processor	Description
	Input	Select the attributes from the data streams included in the matching process.
	Identify	Create identifiers to use in matching, and map them to attributes.
	Cluster	Divide the data streams into clusters. See the Clustering concept guide.
	Match	Choose which comparisons to perform, and how to interpret them with match rules.
	Merge	Optionally, use rules to merge matching records, to create a 'best' set of output records

Identify and Cluster

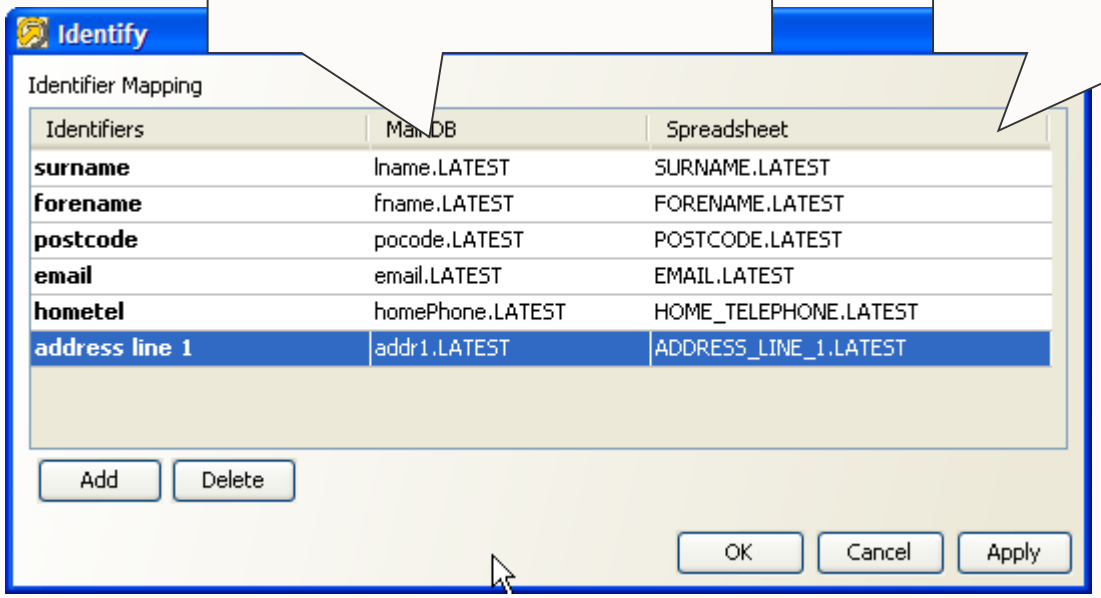
Identifiers

- **We need to create an ‘Identifier’ for each piece of information that we want to use in matching**
- **For example, within a system storing information about books, a book could be identified by:**
  - A Primary Key (System identifier)
  - Its ISBN (Real-world identifier)
  - A combination of Title, Author and Publication date. (Alternative identifier)

We map Attributes onto Identifiers

We map the attribute names from the 'MainDB' dataset onto the identifiers

And similarly for the 'Spreadsheet' dataset.



Clustering – Why We Need It

- **Let's try the 'brute force' approach to de-duplicating 10 million records:**
- **Start at record 1 and compare it with:**
  - Record 2, Record 3, … , Record 10 million
- **Now move onto record 2 and compare it with:**
  - Record 3, Record 4, … , Record 10 million

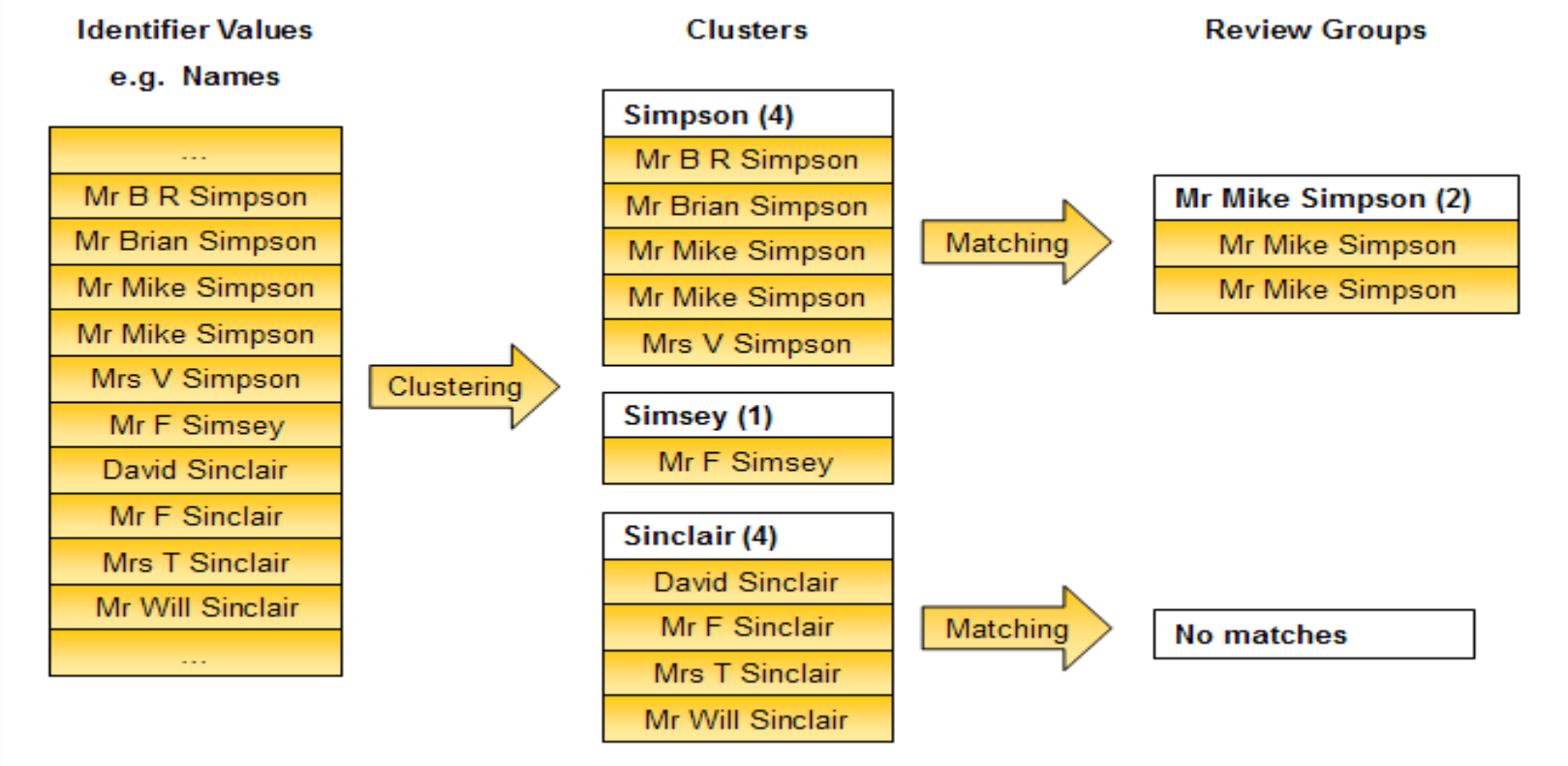
- **This could take some time…**

How Long Would It Really Take?

- **The number of comparisons is about**
  - Half of 10 million x 10 million, which is
  - 50,000,000,000,000!
- **If a server can do 100,000 per second it will take**
  - 500,000,000 seconds OR
  - 138889 hours OR
  - 15.85 years
- **Which is rather too long to wait!**
- **So we need to work a bit smarter…**

Clustering Avoids Unnecessary Work

- **Only compare records with 'some similarity'**

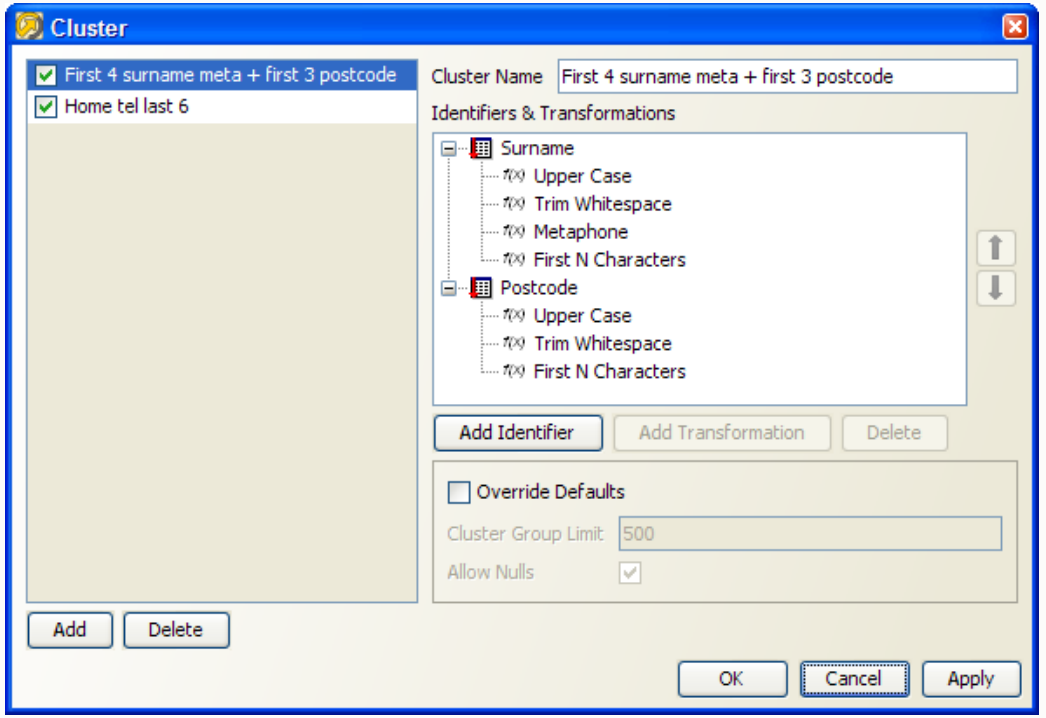


Simple Clustering Examples

- **All people with the same Surname**
  - What if some have a typo?
- **All people with the same Postal code**
  - What if the postcode has changed?
- **All people with the same phone number**
  - What format variability do we have?
    - Area codes, extension numbers etc.
- **All people with the same date of birth**
  - Do we have an accurate DOB for every record?

A more complex Cluster

- **Combines identifiers in a composite cluster**
- **Standardizes the identifiers to tolerate minor variation**



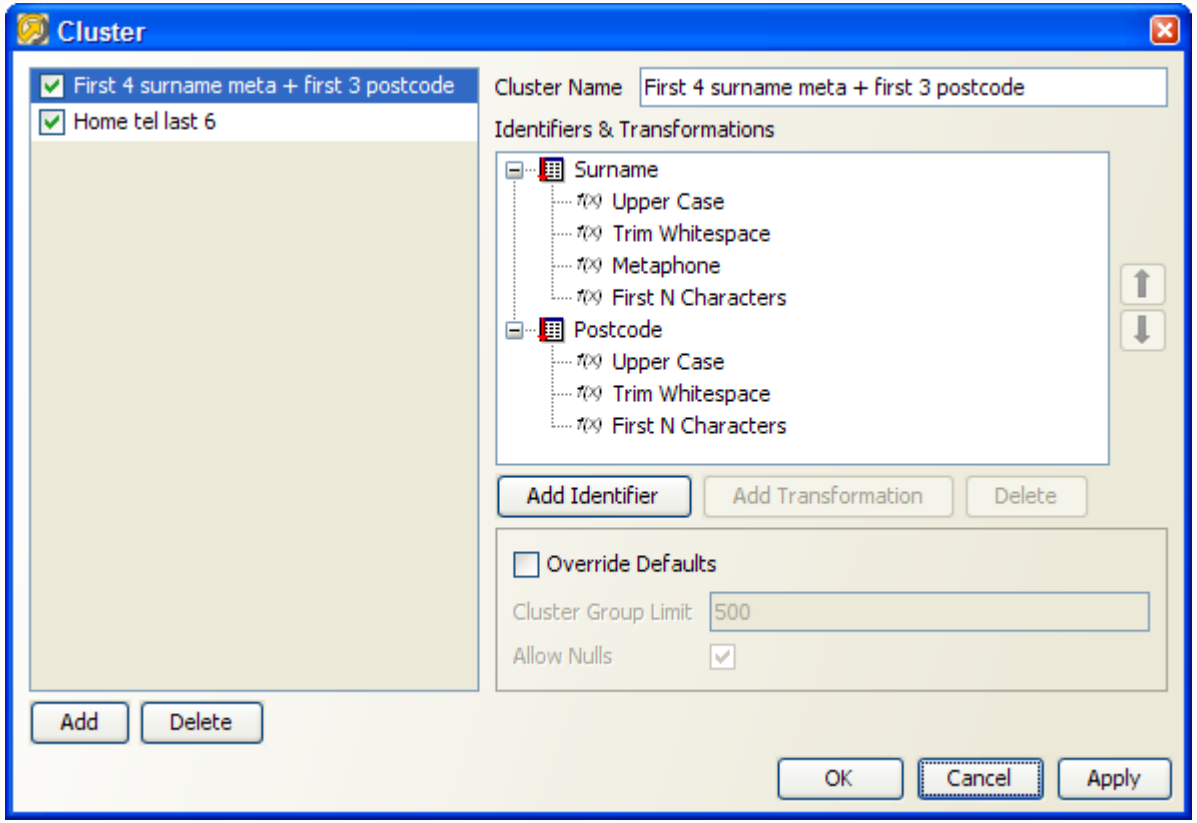
Surname	Postcode	Cluster key
Matthews	CB13 7AG	M0SCB1
Foster	CB4 1YG	FSTRCB4
JONES	SW11 3QB	JNSSW1
Jones	sw11 3qb	JNSSW1

A Good Clustering Strategy

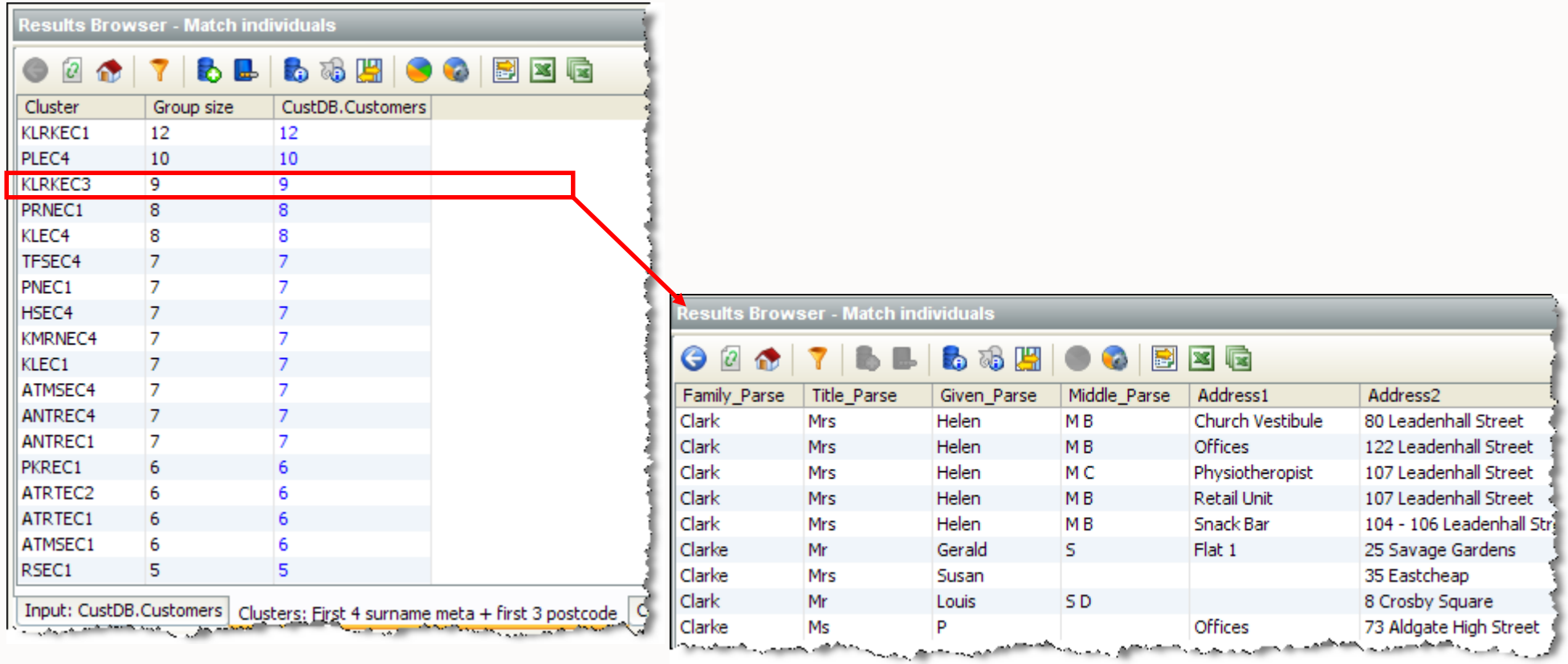
- **Tolerates errors and missing data**
- **Balances the trade-off between cluster group size (and therefore performance) and error tolerance**
- **Normally uses multiple clustering methods e.g.**
  - Surname + DOB
  - Surname Metaphone + Postcode
  - Postcode + DOB
  - If 1 of the 3 pieces of data is wrong/missing, we will still be OK
- **May use more complex keys created by transforming the data before matching**

Configuring Clusters

- **Add the Identifiers**
- **Add transformations to each identifier**
- **Transformations are applied in order**
- **Identifiers are concatenated to form a composite cluster**



Run, Review & Refine Clusters

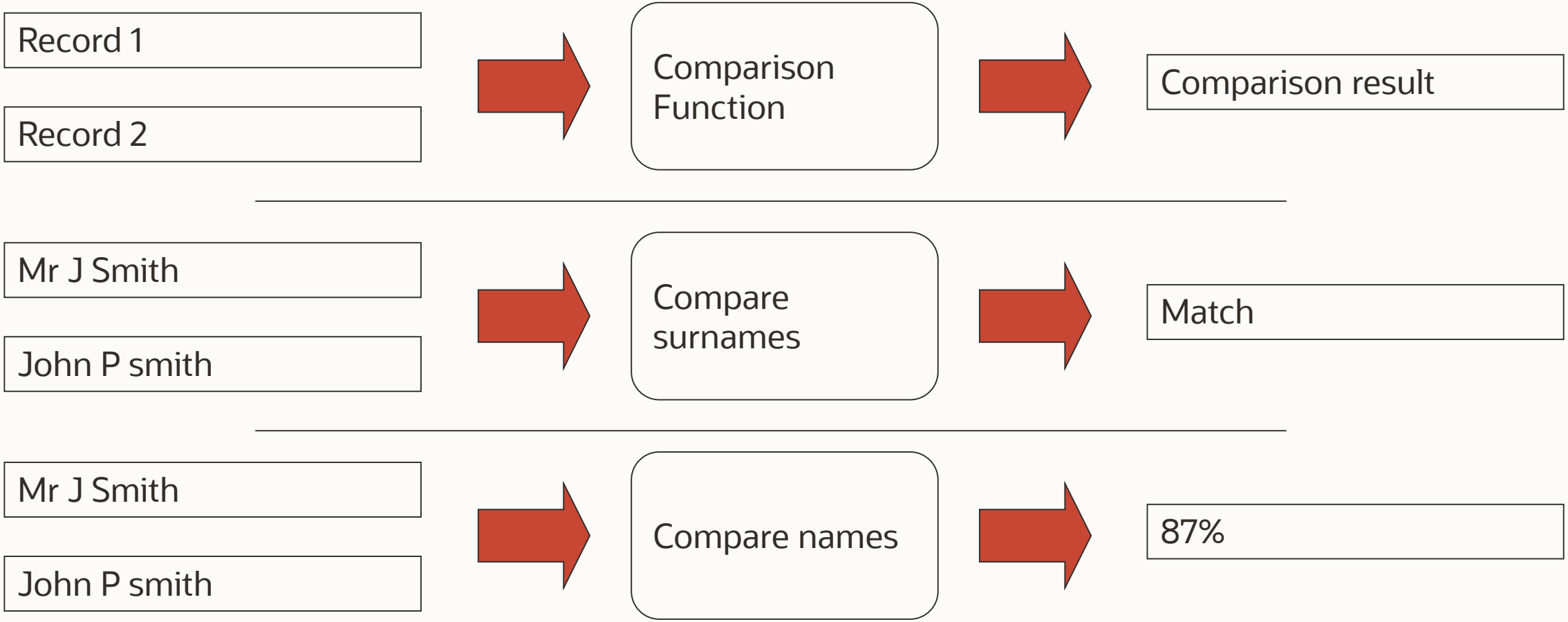


Reminder: Clustering => Performance

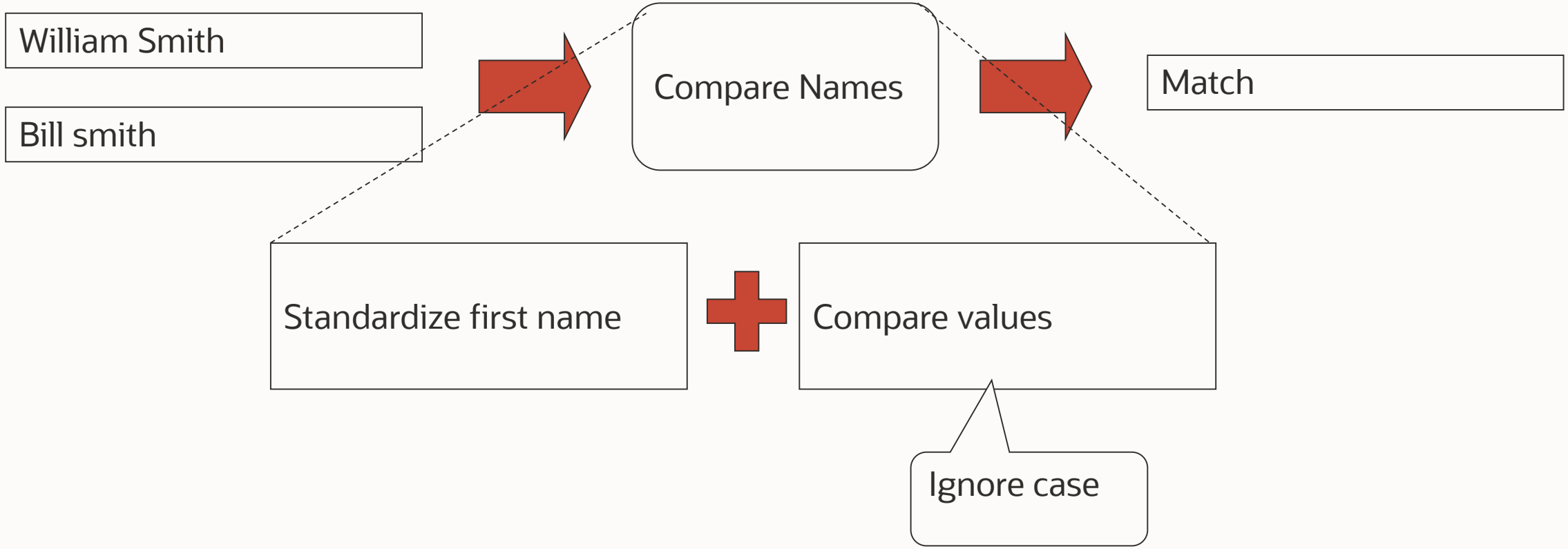
- **Clustering dictates how much work we get to do in the matching process**
- **If you have modest volumes, you can afford for the clusters to be simple**
- **When you have big volumes and tough performance constraints, you need to configure clusters carefully**
  - Use multiple clusters
  - Use composite clusters

Comparisons and Match Rules

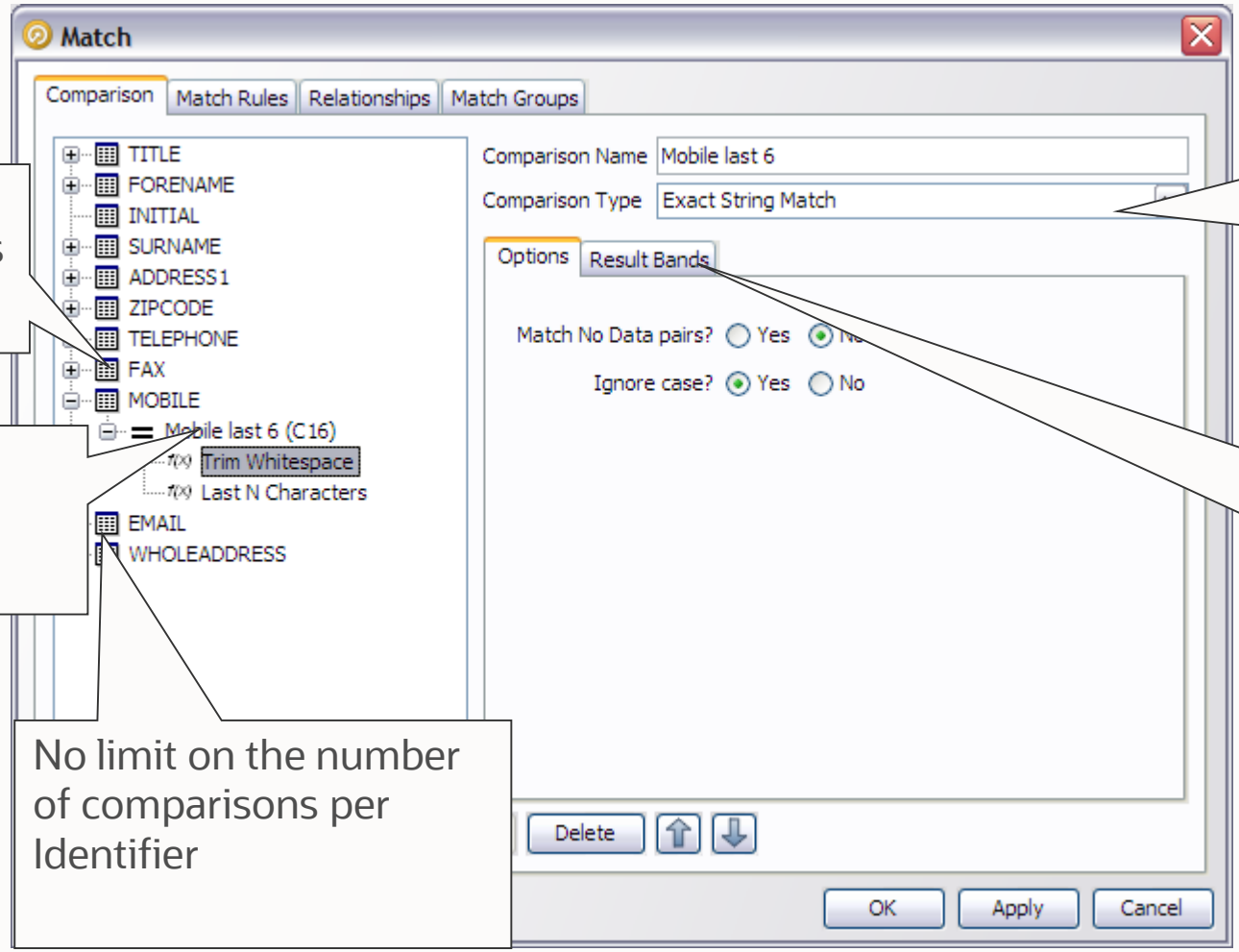
Comparisons



Comparisons Can Include Transforms



Configuring Comparisons



Comparisons apply to Identifiers

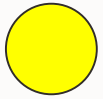
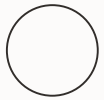
Transformations are applied before the comparison

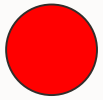
No limit on the number of comparisons per Identifier

Select the comparison type (algorithm)

If the comparison gives a numeric answer rather than Yes/No, we define bands for the result

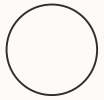
Results of Comparisons = Match Rules

Gender & Initial & Surname	Premise No. & Locality	Postcode	Decision
			Match
			Match
			No Match
			Review

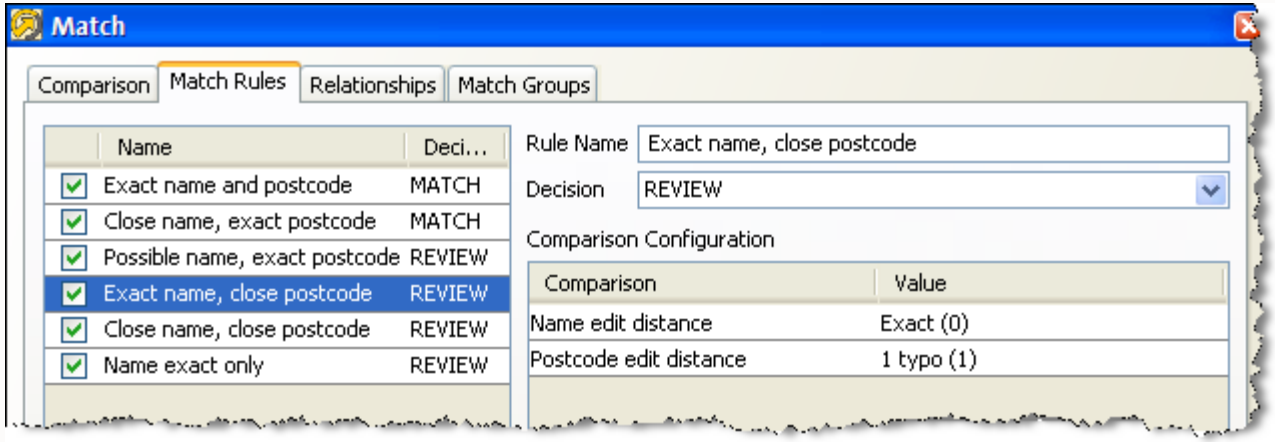
 No Match

 Close Match

 Equivalent

 No Data

Configuring Match Rules



- A Match Rule is simply the combination of comparison results
- Rules are evaluated in order and if one hits, we stop
- Rules can be ‘negative’ to eliminate pairs that are too different with a ‘No Match’ rule
- Rules can easily be turned on & off during the tuning process

Compound Comparisons and Scoring

Compound Comparisons

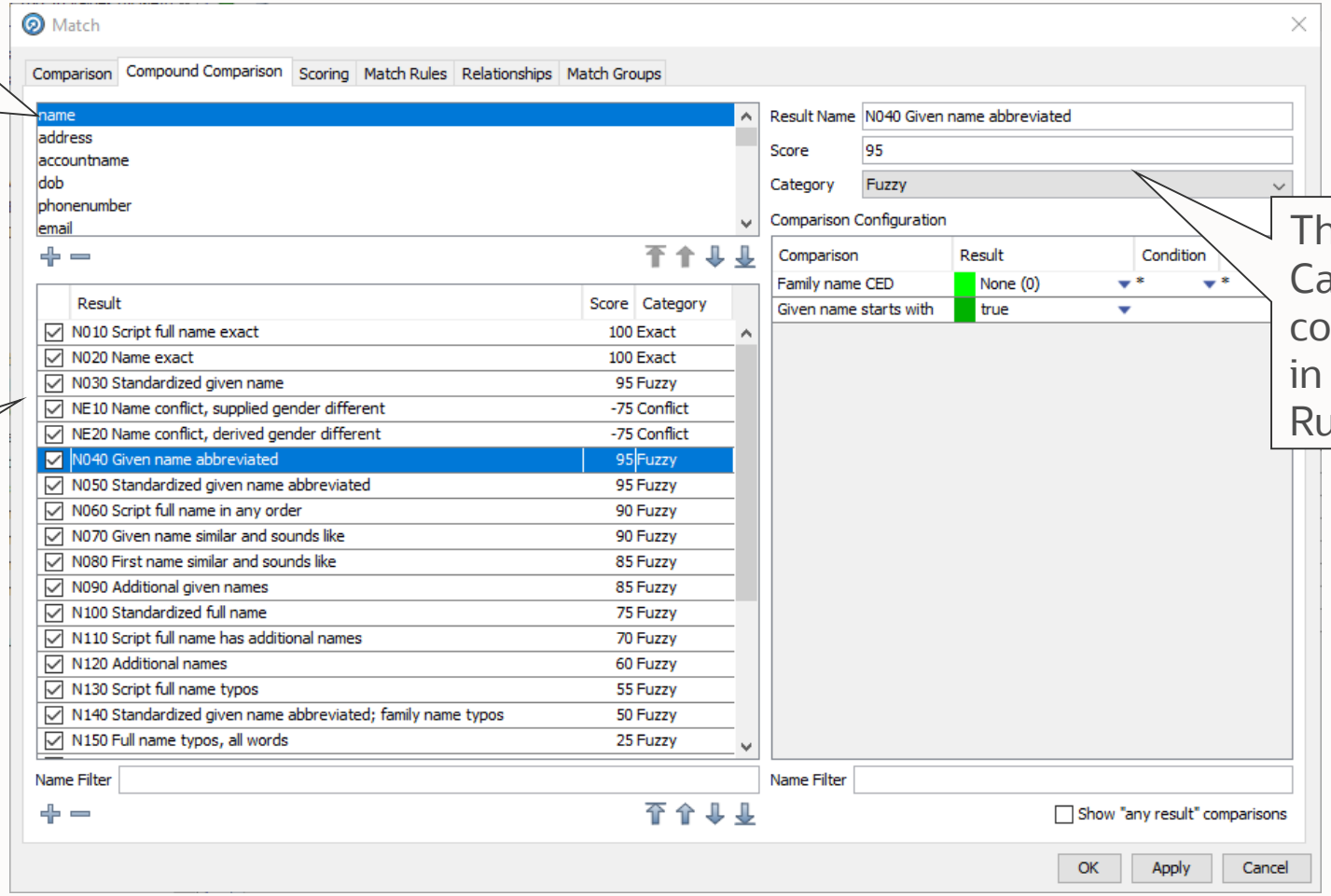
- Where a Match process involves a lot of identifiers, and especially where it may need to be used for many different input data sets, it can be useful to use Compound Comparisons
- This approach allows you to define a rich matching configuration to match data with many variations across many identifiers without using too many match rules
- A Compound Comparison is like a match process within a match process; it uses base comparisons and match rules, normally on a single identifier (or a set of related identifiers, e.g. Name and Gender) to come up with a constructed comparison algorithm for that identifier, with a Score and a result Category (Exact/Fuzzy/Conflict/No Data)
- These outputs can then be used in Scoring and in the final Match Rules for the match processor
- Compound Comparisons are used extensively in the Customer Data Services Pack's out-of-the-box match services

Compound Comparison Example

- **Screenshot of EDQ-CDS Individual Matching:**

A compound comparison exists for each major identifier

The compound comparison is made up of rules, similar to match rules, but only used to derive the output of this compound comparison



The output Score and Category of the comparison can be used in Scoring and in Match Rules

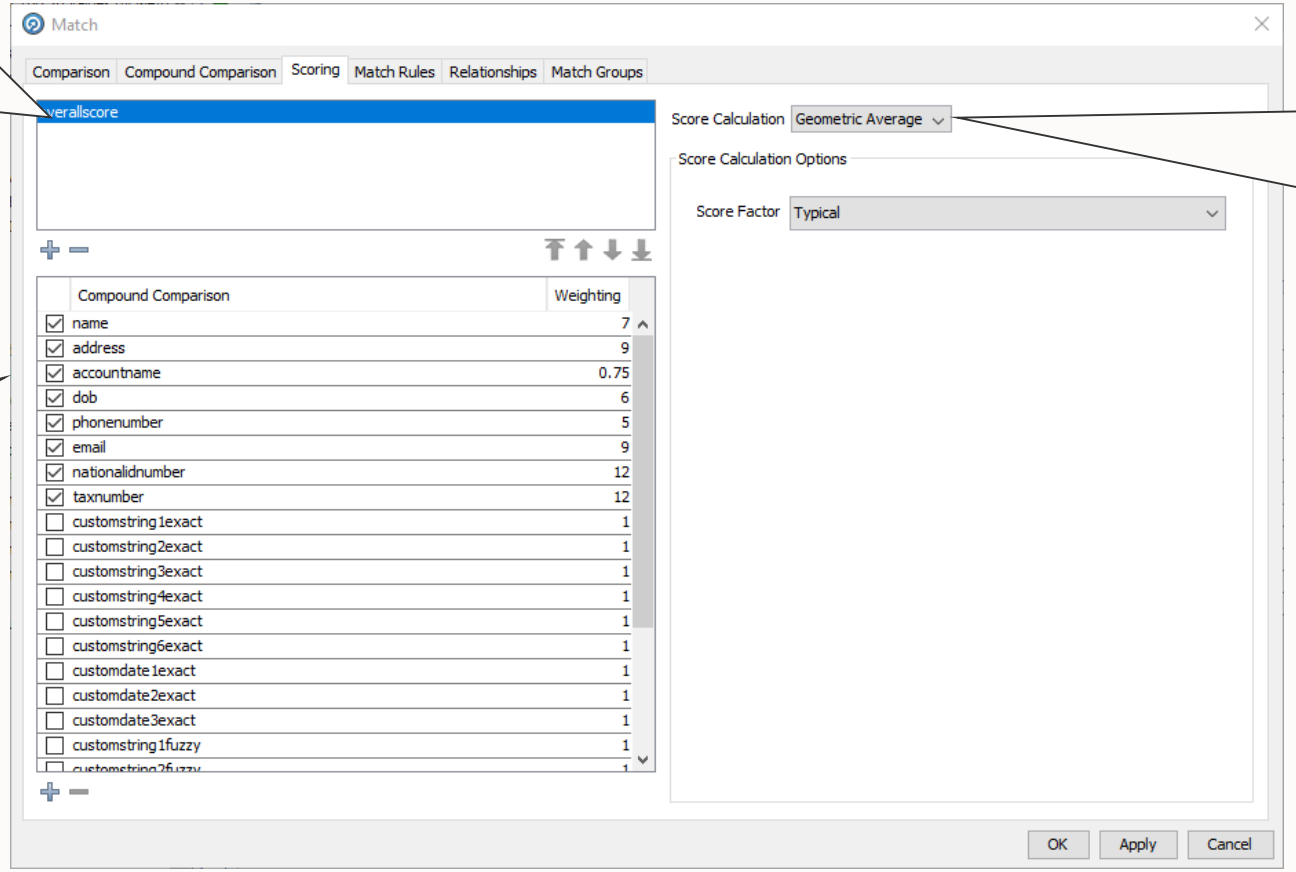
Scoring

- The Scoring functionality in EDQ Matching can use the results of Compound Comparisons and apply dynamic weightings to them to come up with an Output Score, that can then be used in match rules
- The weightings can be altered per message (for real-time matching) meaning a match process can run all the time but apply different weightings and scoring for different consumers
- Scoring is optional and intended for cases where external applications need to be able to tune matching without altering the EDQ configuration
- In other cases, a simple Priority Score can be used with each Match Rule to guide data stewards in the likelihood of a match… and tuning is easier to do by tuning the comparisons and rules to the data
- Note that most match options can also be overridden on a per job basis using overrides, for example to enable/disable rules or change cluster limits

Scoring Example

- **Screenshot of EDQ-CDS Individual Matching:**

A single Overall Score is used in this case. It is possible (but rare) to use many scores

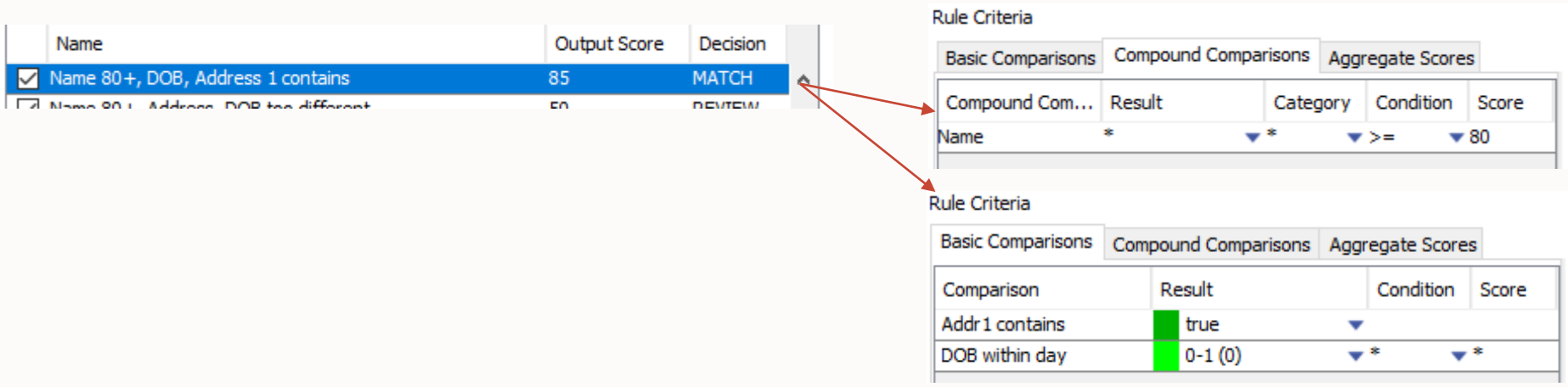


EDQ comes with two different ways of computing scores (Geometric Average, and Weighted Average). This is extensible.

Scoring works by applying weightings to the configured Compound Comparison scores. The Weightings, and which comparisons to use, can be overridden per message

Using Compound Comparisons and Scores in Match Rules

- **Compound Comparison and Scoring outputs can be used in Match Rules to decide the final outcome from matching**
- **This could be as simple as 2 match rules, for example:**
  - Overall Score > 89 = Match
  - Overall Score >69 = Review
- **Or, the use of Scores and Compound Comparison results can be combined with other rules and criteria**
- **This Match Rule uses the score of a Compound Comparison on a Name identifier, as well as Basic Comparison results on Address and Date of Birth:**



Tips for common Matching scenarios

Date Matching Tips

- **Profile the dates to establish how good they are**
  - Date profiler will show unusual distributions
- **Consider the source of the data**
  - DDMMYY vs MMDDYY formats in text fields?
  - Be careful on translation from text to date
- **Useful comparisons:**
  - Date Transposition Match (DDMM vs MMDD)
  - Date Edit Distance – number of typos
  - Date Difference

People Matching Tips

- **Name is a good starting point, but not as helpful as you would hope**
  - Lots of John Smiths
- **Date of Birth is very useful if populated, but be sure to check the quality & completeness before you rely on it**
- **Other supporting information includes things like**
  - Gender – can help distinguish otherwise close matches such as Paul/Paula
  - Address – but people move house…
  - Telephone numbers – esp. mobile number
  - E-mail – but sometimes shared
  - SSN/Passport no if you have them!

Address Matching Tips

- **Understand how good your data is**
  - Consistency of field use
  - Consistency of spelling
  - Has an address standardization package been used?
- **Some bits of address information are more identifying than others:**
  - Building number
  - Apt number
  - Zipcode
- **It's often useful to extract these with the parser to create additional attributes to match on**
- **Using a fuzzy match on a 'whole address' field can deliver great results for some applications**

Our mission is to help people
see data in new ways, discover
insights, unlock endless possibilities.