Identifying duplicates in your data using InfoSphere QualityStage

Accelerate finding duplicates in name and address

Namit Kabra (namkabra@in.ibm.com) 30 June 2015
Senior Staff Software Engineer IBM

This article illustrates the steps required and best practices to find duplicates in data using IBM InfoSphere® Information Server. Additionally, it will provide accelerators to standardize and find duplicates in names and addresses for India, Canada, Germany, and Japan.

Overview

Information governance provides concrete business advantages. A good information governance strategy ensures communication with the correct business partners and customers, and ensures compliance with external regulations. The existence of duplicates in the data impedes communication and can lead to compliance issues. There are countless stories about the problems of duplicate records, such as increased mailing costs, and compliance issues caused by user fraud, such as the same user modifying name and address details to register multiple times for a national pension.

InfoSphere QualityStage®, a component of InfoSphere Information Server, provides data cleansing functions with an easy-to-use, design-as-you-think flow diagram. InfoSphere QualityStage and InfoSphere DataStage® are tightly coupled, sharing a common design and runtime environment. This lets you easily embed data cleansing tasks into any information integration process. QualityStage helps you understand existing data sources, correct and standardize information, and find duplicates in the data before you deliver the cleansed data to the data warehouse. The data warehouse subsequently feeds the enterprise with trusted information. Although InfoSphere QualityStage can be used to cleanse data of multiple domains, this article will focus on data cleansing for Indian names and addresses. You will learn how to identify duplicates within one input source. InfoSphere QualityStage has out-of-the-box rules for various other countries, and similar mechanisms can be employed to address their needs.

Use cases for finding duplicates in data

In each of the following examples, the solution is a data quality tool that can identify duplicates:
• There are several telecommunications companies in India. According to the Telecom Regulatory Authority Of India, a user cannot have more than nine active connections with a telecommunications company. How can a service provider identify if a user has more than nine connections?
• The government of India permits only one LPG cooking gas connection per household. How can we identify whether a household has more than one connection?
• Government agencies give pensions to senior citizens, widows, and the disabled. How can the government prevent people from fraudulently claiming more than one pension?
• The Insurance Regulatory Authority collates information about various insurance claims from insurance companies. Can it prevent fraud by identifying duplicate claims for the same health issue?

Without a data quality tool in place, organizations might be exposed to fraud and might face substantial fines for not complying with government rules. But there are multiple challenges in finding duplicates in the above use cases:

• It is impossible to manually check millions of records for duplicates.
• Finding duplicates using simple SQL statements might not be sufficient. Customers can write names or addresses in different formats. Data entry errors or fraud can create duplicates that appear to be different records.
• Terminology differs between regions. A street can be called *galli* in North India or *salai* in South India.
• In some fields — such as birthdate (dateofbirth), ID, and phone number — data entry operators might have entered a default value to get around the system. This might lead to records having incorrect data. For example, several tax IDs might be entered as 9999999999. Not detecting this earlier in the data cleansing process may result in grouping non-duplicate entities as duplicates.

**Steps for identifying duplicates in data**

Finding duplicates involves the following steps:

1. **Perform data analysis** to find the current state of the data. This helps understand the nature and scope of data anomalies to assist in matching. For example, if the email address is NULL or invalid in 97 percent of the records, and the state of residence is the same in all the records, we may choose not to match the records based on email address or state name and choose alternate fields for matching.

2. **Standardize** the individual fields based on prebuilt tables and data rules, and make them uniform according to business standards. For example, the name field entry of two records is "Mr. Gupta Akshay" and "Akshay Gupta c/o Gopal Gupta." Both entries are representations of the name of same person. Without standardization, it is challenging to identify them as duplicates.

3. **Generate frequency** of the standardized data to find the frequency of the occurrence of a data value within a particular distribution. This helps in comparison of records, which is superior to a purely deterministic comparison. For example, you may be more confident of a
match when two of the records have the same valid tax ID entry than when the records have the same value in the gender or country field.

4. **Create a match specification**, a blueprint for finding duplicates in data. The other steps outlined here are fairly straightforward and can be designed in a matter of minutes. Match specifications contain the main logic for finding duplicates, and often it takes some experience to create a match specification tailored to your need.

5. **Create an InfoSphere QualityStage match job** that uses as input the standardized data, match frequency, and match specification created earlier to identify duplicate records within data.

**Note:** Finding duplicates in data is an iterative process. It is a best practice to run the process on a representative sample of the entire data before running it on the entire data. No matching system guarantees 100-percent accuracy all the time. You must understand and define your business tolerance for missed matches, as well as false matches. This involves iteratively testing the modifications being made to the match specification. Finally, to ensure the accuracy and completeness of the processed data, data stewards must inspect critical data, assess match criteria, and provide feedback on the results.

**Data analysis**

You can perform data analysis in InfoSphere Information Analyzer or with the Investigate stage of InfoSphere QualityStage. Data analysis plays an essential part in data quality by helping determine the current state of data. It can determine whether the project is feasible and helps determine how much work will be necessary in the ETL process to clean your data. It can answer questions such as:

- Does the data column have out-of-range, default, or unique values, such as a telephone number of 9999999999?
- Does the data column have free-form text that requires parsing to extract key components? For example, a free-form address column may contain door number, street name, city name, etc., which can be extracted into different columns for optimal matching.
- Does the data column have values that do not match the type of information required in the field? For example, are there values such as DBA, C/O, driver's license numbers, or other out-of-place information in a name or address field?
- Does the data column have values that overlap adjacent fields and require a realignment of field content? For example, does name information extend into street address fields?
- Does the data column have invalid formats that can cause conversion problems, such as alphanumeric formats in numeric-only fields?
- Does the data column have blank or missing data or data that requires handling of special characters or punctuation?

**Data standardization**

Standardizing data involves moving free-form data (columns that contain more than one data entry) into appropriate new columns and modifying data to conform to standard conventions. The process identifies and corrects invalid values, standardizes spelling formats and abbreviations,
and validates the content and format of the data for the key entities such as customer, partner, or product.

Let's try to understand the process of Standardization through the means of an example:

**Table 1. Unstandardized name and address**

<table>
<thead>
<tr>
<th>Col</th>
<th>Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add1</td>
<td>Gupta Akshay</td>
<td>D- 123,GREATER KAILASH II, New Delhi 110048</td>
</tr>
<tr>
<td>Add2</td>
<td>Mr. Akshay Gupta</td>
<td>D/123 GK 2, ND 110048</td>
</tr>
</tbody>
</table>

These two addresses represent the same person at the same location. But the differences in formatting and description prevent such information from being connected. The output from the Standardize stage, using a set of rules for names and addresses, would produce:

**Table 2. Standardized name and address**

<table>
<thead>
<tr>
<th>Col</th>
<th>FirstName</th>
<th>LastName</th>
<th>DoorMatch</th>
<th>Area</th>
<th>City</th>
<th>PINCode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add1</td>
<td>Akshay</td>
<td>Gupta</td>
<td>D 123</td>
<td>Greater Kailash - 2</td>
<td>New Delhi</td>
<td>110048</td>
</tr>
<tr>
<td>Add2</td>
<td>Akshay</td>
<td>Gupta</td>
<td>D 123</td>
<td>Greater Kailash - 2</td>
<td>New Delhi</td>
<td>110048</td>
</tr>
</tbody>
</table>

The Standardization rule set identifies the following information and makes changes where necessary:

- In Add1, Gupta is the last name and Akshay is the first name.
- D 123 is the door number, and it is changed to a consistent format.
- GK 2 and Greater Kailash II are moved to a consistent format.
- In Add2, the city is correctly identified as New Delhi from the PIN code.

After standardization, when a matching algorithm is run to compare the two addresses, the probability of finding a duplicate is enhanced.

**Note:** Standardization does not manipulate the original data columns. Rather new columns are added during the standardization phase to store the normalized representation of the data. Standardization rule sets are used to create this consistent representation of the data.

**Standardization rule sets**

The InfoSphere QualityStage standardization process uses standardization rule sets to condition and standardize data based on specific domain requirements. InfoSphere QualityStage provides a range of built-in rules that can be used as-is or as the starting point to customize and fine-tune the standardization to specific requirements.

Standardization rule sets produce a fairly consistent output, but there can be exceptions, such as when you encounter new data, unexpected conditions, default or test data, and unusual formats. In such cases, the rule set can be overridden by using the InfoSphere QualityStage Standardization...
Rules Designer. The InfoSphere Standardization Rules Designer provides capabilities to enhance standardization rule sets.

Indian name and address standardization

The address entry process in India is not uniform across the country. The address can be written in different formats, and in the same address there can be multiple streets or landmarks information. Also there could be linguistic connotations for landmarks or linguistic equivalents for street. To solve this, InfoSphere QualityStage uses various pattern-recognition and pattern-resolution techniques. Address is classified into several elements that can occur multiple times in an address input. Built-in standardization rule sets are available for the Indian domain.

INAREA

Any Indian address is first passed through the INAREA rule set. The INAREA rule set identifies the city name, state name, and PIN code from the input address. A missing city name can also be generated from the PIN code. Anything apart from these gets moved into the UnhandledData column to get processed by a downstream region-specific rule set. The figure below demonstrates application of the INAREA rule set on an input address.

Figure 1. INAREA rule set processing an Indian address

![INAREA rule set processing an Indian address](image)

We see the following happening:

- City name and PIN code are identified from the free-flow text.
- State name is generated based on the city name or PIN code.
- The city name has been normalized from "Calcutta" to "Kolkata."
- A validation flag is generated, indicating that the PIN is valid.
- The part of the address that is unhandled is moved to the UnhandledData column. This will be processed by a downstream state-specific rule set.

Based on the state identified above, a rule set for WestBengal (INWBAD) would be invoked for region-specific handling of the unhandled data. The selection of the particular state rule set happens seamlessly when you use the Indian Address Shared Container for address standardization. The following section shows the use of the state-specific rule set.
INWBAD

INWBAD is a state-specific rule set chosen based on the state selected in the previous step. This rule set handles state-specific address formats. The figure below demonstrates its application on the part of input address that was unhandled in the INAREA processing.

Figure 2. State rule set processing region-specific data

This rule set shows the power of data standardization. Data from free-flowing text is captured in the appropriate rows and columns:

- Door number, floor value, apartment name, locality, etc., have been correctly identified.
- Galli has been identified as a representation of street.
- Phonetics have been generated for the important columns. For example, the concatenated name of the building is RAMKRISHNA in the figure above demonstrating region specific standardization. RANCRAAN and R526 are phonetics generated using the popular NYSIIS and SOUNDEX algorithms, respectively. In case there is a data entry error in this name, the columns, such as apartment name or street name, can be matched using the phonetics.
- The abbreviated data is also standardized in some cases (FLR becomes FLOOR, APT becomes APARTMENT).
- If some portion of the data cannot be standardized, it becomes a part of UnhandledData. This unhandled data can be grouped together to identify a pattern that is not being currently handled by the rule set. A couple of new rules can then be added to handle these to get an improved result. Typically, adding just one or two additional rule sets takes care of most unhandled data.

INNAME

The INNAME rule set is used to standardize Indian names. The figure below demonstrates the application of INNAME rule set to input data.
Figure 3. Standardizing Indian names

The data standardization for name identified or generated the following information:

- This is the name of an individual, not the name of an organization.
- The gender is male.
- The first, middle, and last name, the name prefix, and an additional name that is present.
- The phonetic code for the first and last names.

Customized rule sets

Using InfoSphere QualityStage, you can create customized rule sets. The following are examples of customized rule sets. (Note: These additional rule sets are bundled with this tutorial).

**INTAXID**

The INTAXID rule set standardizes the Indian Permanent Account Number (PAN). The figure below demonstrates the application of the INTAXID rule set to input data.

Figure 4. Standardizing PAN
The data standardization for PAN identified or generated the following information:

- The PAN is in proper format.
- The PAN is of a person, not of an organization.
- The first letter of the PAN holder's last name is also identified.

**INPHONE**

The INPHONE rule set standardizes Indian phone numbers. The figure below demonstrates the application of the INPHONE rule set to input data.

**Figure 5. Standardizing Indian phone number**

![INPHONE rule set example](image)

The data standardization for PAN identified or generated the following information:

- The city name and service area name are generated based on Subscriber Trunk Dialing code.
- The phone number and an additional phone number are identified.
- The phone number was identified as valid.

**INMOBILE**

**Figure 6. Standardizing Indian mobile phone number**

![INMOBILE rule set example](image)

Though the above input is a 10-digit number, it is marked as invalid by the rule set, which invalidates some of the default values usually entered by the user. After standardization, this value will not be used for blocking or matching. Therefore, the probability of generating a false positive based on an invalid number is reduced.
Data standardization job
The following graphic shows a job that standardizes Indian names and addresses from a file with 52 rows.

Figure 7. InfoSphere QualityStage job standardizing Indian name and address

In the figure above, Name, PAN_Number, and Phone are standardized in the Standardize_Name_PAN_Phone stage. The address is standardized in the StandardizeAddress stage, which is an out-of-the-box shared container for Indian addresses, then the results from both are merged. A match job requires two inputs: standardized data and frequency; both are generated here.

Note: The IndiaAddressSharedContainer used in the StandardizeAddress stage is a part of the India rule set.

Generating frequency
InfoSphere QualityStage works on the principle of probabilistic matching of two records. Weights are assigned if two fields match. Following is an example of matching two records and the weights assigned.

Table 3. Standardized representation of name and address

<table>
<thead>
<tr>
<th>Col</th>
<th>FirstName</th>
<th>LastName</th>
<th>DoorMatch</th>
<th>Area</th>
<th>City</th>
<th>PINCode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add1</td>
<td>Akshay</td>
<td>Kazangian</td>
<td>D 123</td>
<td>Greater Kailash - 2</td>
<td>New Delhi</td>
<td>110048</td>
</tr>
<tr>
<td>Add2</td>
<td>Akshay</td>
<td>Kazangian</td>
<td>D 123</td>
<td>Greater Kailash - 2</td>
<td>New Delhi</td>
<td>110048</td>
</tr>
<tr>
<td>Weight</td>
<td>+4</td>
<td>+20</td>
<td>+7</td>
<td>+4</td>
<td>+3</td>
<td>+5</td>
</tr>
</tbody>
</table>

InfoSphere QualityStage uses the frequency to find the rarity of a particular data value and assigns a higher weight if rare values match and a lower weight for matches of values that occur more
frequently. It gives lower weight to matches of common names and higher weight to matches of unusual names. Likewise, a match on gender gets a lower weight than a match on Social Security number. In the example above, assume that there are 1 million records and that there are only two with a last name of KAZANGIAN. InfoSphere QualityStage takes this into account, and because of the frequency, automatically adjusts the probability of a match higher, giving the comparison more weight.

**Frequency generation job**

The following graphic shows a portion of the earlier data standardization job generating frequency. The standardized data is passed through the Match frequency stage to generate the frequencies for the standardized data. If the match specification is selected in the frequency generation stage, frequency will be generated only for those columns that will be used for matching records.

**Figure 8. Frequency generation job**

To compare records, you must configure the specific match conditions by using the InfoSphere QualityStage Match Designer. Using the Match Designer tool, you can select the columns to be compared for the matching process. A Match Designer has many components you can fine-tune to get the optimal result. The most important are Blocking Column, Matching Columns, Match Passes, and Cutoffs.

**Blocking columns**

In *blocking*, the input is partitioned into mutually exclusive groups (called blocks). This is similar to grouping the same color socks together, reducing the number of pairs to be compared to identify a duplicate.
Blocking is generally implemented by automatically partitioning files based on the values of one or more fields. The size of the candidate pool created by blocking requires balance. If you make blocking criteria too conservative (small blocks), you will miss some matches. If you make blocking criteria too relaxed (large blocks with thousands of records in a block), execution time increases. It is a best practice that any block should not contain more than 1,000 records.

Use the following guidelines for selecting fields for blocking:

- Select fields that are reliable, well populated, and have a good frequency distribution of values. Data analysis results provide you with this information.
- Use multiple fields in a block.
- Go from most specific in the first pass to more lenient in future passes. Your first pass should identify the most obvious matches.

**Match commands**

In the socks analogy, once you block all the similar socks together, you need to compare each one with the other. This comparison of socks happens just within the block, based on the size, color, brand, and other attributes of the remaining socks. Similarly, when comparing two addresses, each selected column from matching columns is compared to the remaining address within a block. For example, the door number, street name, and city name of one record is compared with the door number, street name, and city name of another record and appropriate weights (called agreement or disagreement weights) are generated based on each of the comparison result.

For comparison, these match commands can use one of more than 20 available comparison types, which can be selected from the Match Designer tool. Comparison types can be either *exact-match comparison*, which does not tolerate any differences; or *error-tolerant comparison*, where you can specify the tolerance threshold that indicates how much difference is tolerated. After comparing each selected column of the various records, the Match stage adds the weights assigned to each column comparison and generates a composite weight for the record. The higher the composite weight the greater the agreement between the two records.

**Vector matching**

On rare occasions, you may need to compare values across columns to find a match. For example, a record has two phone number columns: Phone0 and Phone1. In this case, you get a better result if you can compare across columns. For example, you can compare the value of Phone0 in one record with the value of Phone0, as well as Phone1 in the second record. InfoSphere QualityStage provides this ability through a mechanism called vector matching. You can use the Make Vector Stage to create a vector called `PhoneNumber` that contains Phone0 and Phone1. You can then use this vector in the match command.

In the following standardization job, a Make Vector Stage is used to create the vector `PhoneNumber` from Phone0 and Phone1. Phone0 and Phone1 are the standardized phone numbers generated from the input.
Figure 9. Standardization job generating vector for phone number

![Diagram of standardization job](image)

This job uses the following Standardization Rule Sets:
- CANAME
- CAADDR
- CAAREA

In the Match command canvas below, the PhoneNumber vector is matched using vector comparison to achieve the cross-column comparison.

Figure 10. Match command to compare vectors

![Match command canvas](image)

**Match passes**

Let us return to the socks analogy. While matching, what if you found one pink sock that might have been white originally, but is now discolored? Due to color error, you do not put the pink sock with the white socks in your first pass of sorting socks. Using match terms, the pink sock does
not make it into the block of white socks. So there is a need of subsequent regrouping. After you group by color, you group the remaining, unpaired socks by size, then by brand. In one of these subsequent regroupings, you might find the most likely match for the pink sock. Similarly, multiple passes help overcome the problem of records not making the correct block group as they did not meet the required exact match criteria. Therefore, it is a best practice to use multiple passes in the process of finding duplicates.

**Cutoff values**

Match and clerical cutoffs are thresholds that determine how to categorize scored record pairs. Cutoff values are based on the composite weight assigned to each record pair. This weight is assigned automatically by the match engine. If desired, you can override this value. All record pairs with composite weight equal to or above the match cutoff value are considered duplicates. Record pairs below the clerical cutoff are considered non-matches. Records with composite weight between the two values are considered clerical records.

The cutoff values have direct influence on whether a record pair is considered a match. Therefore, the actual business purpose should determine how aggressive or conservative those values might be defined. For example, if you perform one-source match to create a mailing list for shopping catalogs, it might be acceptable to set a lower match cutoff value than you would set with patient records. It is common practice to set match cutoff values to 0 in the initial run.

**Match specification for Indian name and address**

In our test sample from Indian addresses, the addresses were mostly urban. The addresses have door number and PIN code well populated. Based on this information, the following blocking columns were created in the Match Designer. Note that the results on the right side are actual results generated when the sample data was provided as an input. The records are grouped together as duplicates based on the SetID key that InfoSphere QualityStage match job automatically generates to group the duplicates together.
Pass 1

Figure 11. Pass 1 blocking on first name, middle name, last name, door number, and PIN code

It is important to note that to be a part of the same block, first name, middle name, last name, door number, and PIN code values should be identical and populated. Comparisons happen within each of these blocks based on the match commands. This pass can compare records that have first, middle, and last names (for example, Subhash Chandra Bose) and whose addresses have door number and PIN code. If we just blocked on first and last names, then the block of common Indian names, such as "Amit Gupta," could potentially lead to block overflow and increase in the execution time. As a best practice, initial passes are stricter than the later ones.

Pass 2

Figure 12. Pass 2 blocking on first name, last name, door number, and PIN code

Pass 1 required records to have a middle name, so another pass is needed to match records missing the middle name. It blocks on first name, last name, door number, and PIN code. So why not just have Pass 2 and drop Pass 1? Using two passes keeps the block size manageable. Also,
the names such as "Mohammed Yonis Khan" and "Mohammad Asif Khan" have the same first and last name. In the absence of Pass 1, they might prematurely be considered duplicates.

**Pass 3**

**Figure 13. Pass 3 blocking on name phonetic, door number, and PIN code**

Phonetics of key fields were generated during the process of standardization. Pass 3 uses these phonetic columns of first and last names for blocking. This handles cases where there could be intentional or typographical errors while entering the first and last names in a record.

**Pass 4**

**Figure 14. Pass 4 blocking on first name, door number, and PIN code**

Pass 2 blocks on records that have a first and a last name. But some names in South India are written with initials and first name, for example, P.J.V.S. Prasad. So to accommodate those records, Pass 4 is created.
Pass 5

Figure 15. Pass 5 blocking on primary name, door number, and PIN code

The earlier passes do not compare cases where someone has put initials before the last name — M.K. Agarwal, for example. This pass compares those cases, but this introduces another challenge. In a household, the wife and the husband typically have the same last name, address, and possibly phone number. So this can create false positives if the first name is not included in the block columns. This pass can potentially match names of husband and wife, such as Anirban Mandal and Nioti Mandal, who reside at the same address. To overcome this, you can override and increase the disagreement weight of first name in the overall matching score. This can also be done in the Match Designer tool. In the above example, the two entities have been matched, but as a clerical pair, which means that the tool needs human intervention to decide whether it is a match or not.

Pass 6

Figure 16. Pass 6 blocking on unique ID (PAN in this case)

As the image shows, the Pass 6 blocks on the unique ID. What happens if many of these entries have a default value of 99999999 or NULL, for instance? Will this not lead to incorrect grouping?
No, because the PAN used here is the standardized valid value of the PAN. Invalid PANs were eliminated during the prior step of standardization. But why were these two records shown in the image not identified as duplicates in the previous steps? Because the blocking columns for the previous passes assumed that the records have door number and PIN code values. Since door number or PIN code was missing in one of the records, they were not included in any of the blocks.

From the image above of the Match Designer tool, it is evident that the Match Designer tool is not just used to create the match specification, but also to test each of the passes. By running these passes in an InfoSphere QualityStage Match job, you can detect duplicates and relationships, even in the absence of unique identifiers or other data values.

### Creating an InfoSphere QualityStage Match job

An InfoSphere QualityStage match job takes the input of the standardized data and frequency distributions and does the following actions:

- Identifies duplicate records within one or more data sources
- Identifies records that requires clerical review
- Generates the statistics for the match
- Enables the creation of match groups across data sources regardless of whether they have a predetermined key

### Two types of match stage

InfoSphere QualityStage provides two types of matching: One finds duplicates within a single source and another compares two sources of data for matches.

#### Figure 17. Selecting match type from Match Designer tool

Either of these two match types in their different flavors can be selected when you create the match specification that was described earlier.

### One-source match stage

The one-source match stage works on a single input data set. It groups together records with similar attributes in the input. The match specification created in the previous step is an example of one-source match. Another example is the need to eliminate duplicates when you consolidate mailing lists purchased from multiple sources. There are multiple types of one-source matching. The match specification created in the previous step uses the default selection: one-source dependent. In a one-source dependent match, duplicates are removed from match consideration.
in subsequent passes. For example, if Record1, Record2, and Record3 are considered as duplicates in Pass1, the duplicate records are not considered in subsequent passes, and the Master Records and the non-matching records make it into next pass.

**Two-source match stage**

The two-source match stage matches data between two sets of data (reference data and source). An example of a reference match is finding whether an entity that exists in the existing database of an organization is a duplicate of a new entity being added. While a lookup, join, or merge operation might also find some duplicates, those methods lack the probabilistic matching capability of the two-source match stage.

When you create a match job in the Match stage, you can override the choice of match type you specified when you created the match specification.

**Match job**

The following figure shows a one-source match job. It takes two inputs: standardized data and frequency distribution.

**Figure 18. InfoSphere QualityStage one-source match job**

A one-source match job generates matched data that consists of the following groups of records:

- **Match** — The master records considered for subsequent passes.
- **Clerical** — The duplicates that require human review.
- **Duplicate** — The duplicates that are above the match cutoff.
• Nonmatched — The records that do not fall in the category of master, duplicate, or clerical records. They are also called residuals and are considered for each subsequent pass.

The job also generates match statistics, summary statistics about the matching results, and statistics about the matching process for each match pass.

The figure below shows the stage properties for a one-source match. It requires the name of the match specification. The match type can be selected or overridden on this screen.

**Figure 19. Stage properties for one-source match stage**

![Figure 19. Stage properties for one-source match stage](image)

Following is the result data after executing the one-source match job.

**Figure 20. Result from executing one-source match job**

![Figure 20. Result from executing one-source match job](image)

The first four columns of the output provide critical information about the match process:

• qsMatchSetID — The value in this column will be identical for each row in a set of duplicates. This field is used to group duplicates together.
• qsMatchType — Identifies the type of record (match (MP), duplicate (DA), clerical (CP), or non-match (RA)).
• qsMatchPassNumber — Identifies the pass number in which the duplicate was found. Using the transformer stage in the match job, a value of 0 is set for records that do not have duplicates.
• qsMatchWeight — The composite weight of the matched record.

Next steps after finding duplicates

InfoSphere QualityStage provides data cleansing capabilities to help ensure quality and consistency by standardizing, validating, matching, and optionally merging information. Fields that do not conform to load standards are identified and possibly filtered so only the best representation of the match data is loaded into the master data record. Missing values in a record can be generated during data standardization. Missing values can also be populated with values from corresponding records identified as a group in the matching stage. You can create comprehensive and trusted information for multiple uses, including data warehousing or loading analytical views.

Conclusion

In this tutorial, you have learned about the need for finding duplicates in data. You also learned the various steps involved in finding the duplicates in the data and how it can be achieved by using InfoSphere QualityStage. In the download section, you will find the accelerators for India, Canada, Germany and Japan, which you can use to jump-start your data quality project.

Acknowledgements

The accelerator for Germany is created by Manfred Vodegel from IBM Germany. The accelerator for Japan is created by Yusuke Nishitani from IBM Japan. Walter Crockett, Jr. from IBM US has spent hours editing this, and I am grateful for the valuable input. Special thanks to Heather Stimpson (development manager, InfoSphere Data Quality) and Robert Dickson (WW technical sales) for their constant guidance and input to shape up this tutorial.
## Downloads

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Out-of-the-box accelerators for data cleansing</td>
<td>QSDemos.zip</td>
<td>2046KB</td>
</tr>
</tbody>
</table>
Resources

- Participate in the discussion forum for this content.
- Learn more about Standardization using InfoSphere QualityStage.
- Get more information about InfoSphere Information Server from the Knowledge Center.
- Learn more about Information Management at the developerWorks Information Management zone. Find technical documentation, how-to articles, education, downloads, product information, and more.
- Stay current with developer technical events and webcasts focused on a variety of IBM products and IT industry topics.
- Follow developerWorks on Twitter.
- Watch developerWorks demos ranging from product installation and setup demos for beginners, to advanced functionality for experienced developers.
- Get involved in the developerWorks Community. Connect with other developerWorks users while you explore developer-driven blogs, forums, groups, and wikis.
About the author

Namit Kabra

Namit Kabra is a software developer on the InfoSphere Information Server InfoSphere QualityStage team at IBM India Software Lab. He has 11 years of experience at IBM, working for different development teams in the Information Server product area. Apart from contributing to the product development, he helps technical sales in creation of proofs of concept for customers. He has worked on data cleansing requirements for more than 10 Indian clients. Namit also shares his knowledge to technical community through active blogging and has written more than 100 technical blogs (http://www.namitkabra.wordpress.com). He holds two patents in data cleansing.

© Copyright IBM Corporation 2015
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)