Version 4 Release 2

IBM i2 Analyze
Correlation Guide

Note

Before you use this information and the product that it supports, read the information in "Notices" on page 23.

This edition applies to version 4, release 2, modification 1 of IBM® i2® Analyze and to all subsequent releases and modifications until otherwise indicated in new editions. Ensure that you are reading the appropriate document for the version of the product that you are using. To find a specific version of this document, access the Configuring section of the IBM Knowledge Center, and ensure that you select the correct version.

Contents

Information Store data correlation

This documentation provides an overview of the correlation functions that are available during data ingestion into the Information Store in IBM i2 Analyze. Later sections describe how to configure correlation with an example use case.

Intended audience

This documentation is intended for users who want to correlate data as it is being ingested into the Information Store.

Users must understand how to ingest data into the Information Store. For more information about ingesting data into the Information Store, see Information Store data ingestion.

Users must understand the i2 Analyze data model, and i2 Analyze record structure. For more information, see Data in i2 Analyze records.

Overview of correlation

Correlation is the process of associating data based on strong identifiers. For the process of ingesting data into the Information Store, i2 Analyze can use correlation identifiers that you provide to determine how to process and represent data in i2 Analyze records.

Correlation in i2 Analyze

During the ingestion process for the Information Store, correlation can be used to determine when data that is ingested is associated with existing records, and must be represented by a single i2 Analyze record. Conversely, if data no longer represents the same object, the data can be represented by multiple i2 Analyze records. In i2 Analyze, these operations are known as *merge* and *unmerge*.

During the correlation process, an identifier is used to determine how each row of data is associated. You present the identifier to the Information Store with the other staging data during ingestion.

Correlation uses

You might want to use correlation when you are ingesting data that originates from disparate sources, which have common properties, or have the potential to represent the same real-world objects. For example, if you have two data sources that contain information about people.

Another scenario where you might use correlation, is when the data that you are ingesting is in the form of event driven models (crime or complaint reports) where the same actors (people, locations, phones, and vehicles) might be referred to frequently.

Correlation can be used in these scenarios to combine multiple source records into single i2 Analyze records for link analysis.

You can limit the use of correlation to data from a specific source, of certain item types, or per row of data ingested into the Information Store. You do not have to provide a correlation identifier for all the data that you ingest into the Information Store.

Correlation method

In i2 Analyze, correlation identifiers and implicit discriminators are used to determine how the Information Store processes data during ingestion.

When you ingest data into the Information Store, you can provide a correlation identifier type and key value that are used to construct the correlation identifier for each row of data in the staging table. The type and key values that you provide are used to process data that is determined to represent the same real world object. Implicit discriminators are formed from parts of the i2 Analyze data model in the Information Store. Even if correlation identifiers match, if values for elements of the i2 Analyze data model are not compatible, that data cannot be represented by the same i2 Analyze record. For more information about correlation identifiers and implicit discriminators, see "Correlation identifiers" on page 2.

During the ingestion process, i2 Analyze compares the correlation identifiers of the data to be ingested and existing data in the Information Store. The value of the correlation identifiers determines the operations that occur. For more information about the correlation operations that can occur, see "Correlation operations" on page 4.

Example data sets demonstrate the correlation behavior available in this release of i2 Analyze. For more information, see "Ingesting example correlation data" on page 19.

Correlation identifiers

The role of a correlation identifier is to indicate that data is for a specific real-world object. If multiple pieces of data are about the same specific real-world object, they have the same correlation identifier. At ingestion time, the correlation identifier of incoming data informs the Information Store how to process that data. The incoming data is associated with i2 Analyze records based on the origin identifier of the data. Depending on the current state of the i2 Analyze record that is associated with the data, a match with the correlation identifier on an inbound row of data determines the outcome of the association.

You specify the values for the correlation identifier in the staging table that you are ingesting the data from. The correlation identifier is made up of two parts, the *correlation identifier type* and the *correlation identifier key*.

type

The *type* of a correlation identifier specifies the type of correlation key that you are using as part of the correlation identifier. If you are generating correlation keys by using different methods, you might want to distinguish them by specifying the name of the method as the correlation identifier type. If your correlation keys are consistent regardless of how they are created, you might want to use a constant value for the correlation identifier type.

When you specify the identifier type, consider that this value might be seen by analysts.

Note: The length of the value for the *type* must not exceed 100 bytes. This value is equivalent to 100 ASCII characters.

key

The *key* of a correlation identifier contains the information necessary to identify whether multiple pieces of data represent the same real-world object. If multiple pieces of data represent the same real-world object, they have the same correlation identifier key.

Note: The length of the value for the *key* must not exceed 1000 bytes. This value is equivalent to 1000 ASCII characters.

To prepare your data for correlation by i2 Analyze, you might choose to use a matching engine or context computing platform. Matching engines and context computing platforms can support the identification of matches that enable you to identify when data that is stored in multiple sources represents a single entity. You can provide these values to the Information Store at ingestion time. An example of such a tool is IBM InfoSphere Identity Insight. InfoSphere Identity Insight provides resolved entities with an entity identifier. If you are using InfoSphere Identity Insight, you can populate the correlation identifier type to record this, for example `identityInsight`. You might populate the correlation identifier key with the entity identifier, for example 1234. This generates a correlation identifier of `identityInsight.1234`. For more information about IBM InfoSphere Identity Insight, see Overview of IBM InfoSphere Identity Insight.

Alternatively, as part of the data processing to add data to the staging tables, you might populate the correlation identifier with values from property fields that distinguish entities. For example, to distinguish People entities you might combine the values for their date of birth and an identification number, and you might specify the type as `manual`. This generates a correlation identifier of `manual.1991-02-11123456`.

The complete correlation identifier is used for comparison. Only data with correlation identifiers of the same type is correlated.

For more information about specifying a correlation identifier during the ingestion process, see Information Store staging tables.

Implicit discriminators

In addition to the correlation identifier that is created from the type and key values that you provide, *implicit discriminators* are also used during the matching process. If the correlation identifier matches, the following implicit discriminators are also compared. The implicit discriminators must be compatible to enable correlation to occur.

Item type

The item type of the data that you are ingesting must be the same as the item type of the existing i2 Analyze record that is matched by the correlation identifier. If the item types are not the same, then no correlation operations occur.

Security dimension values

The security dimension values of the data that you are ingesting must be the same as the data that is matched by the correlation identifier. If the security dimension values are not the same, then no correlation operations occur.

Link direction and ends

For link data, the link direction and ends of the data that you are ingesting must be the same as the data that is matched by the correlation identifier. If the link direction and ends are not the same, then no correlation operations occur.

The direction and ends of a link are inspected, and direction is respected. For example, a link from A to B of direction 'WITH' matches with a link from B to A of direction 'AGAINST'. A link from A to B of direction 'WITH' does not match with a link from A to B of direction 'AGAINST'.

Correlation operations

When the Information Store receives a correlation identifier, the way the system responds depends on the values of the correlation identifiers, and the state of the records that are associated with them.

The following section explains the *merge* and *unmerge* operations that the system can respond with when it receives a correlation identifier. This response is in addition to the insert and update operations that are part of the standard ingestion process.

Merge

When one or more pieces of data are determined to represent the same real-world object, the data is merged into a single i2 Analyze record. For the Information Store to merge data, the correlation identifiers must match and the implicit discriminators must be compatible.

During ingestion, a merge operation can occur in the following scenarios:

- New data in the staging table contains the same correlation identifier as an existing record in the Information Store. The new data has an origin identifier that is not associated with the existing record.
- An update to an existing record with a single piece of provenance in the Information Store causes the correlation identifier of that record to change. The new correlation identifier matches with another record in the Information Store.
- Multiple rows of data in the staging table contain the same correlation identifier. The Information Store ingests the data as a new i2 Analyze record, or the record merges with an existing record in the Information Store.

After a merge operation, the following statements are true for the merged record:

- The record has a piece of provenance for all of the source information that contributed to the merged record.
- By default, the property values for the merged i2 Analyze record are taken from the source information associated with the provenance that has the most recent value for the `SOURCE_LAST_UPDATED` column.

  If only one piece of provenance for a record has a value for the `SOURCE_LAST_UPDATED` column, the property values from the source information that is associated with that provenance are used. Otherwise, the property values to use are determined by the ascending order of the origin identifier keys that are associated with the record. The piece of provenance that is last in the order is chosen. To ensure data consistency, update your existing records with a value for the `SOURCE_LAST_UPDATED` column before you start to use correlation, and continue to update the value.

  If the default behavior does not match the requirements of your deployment, you can change the method for defining property values for merged records. For more information, see “Define how property values of merged records are calculated” on page 10.
- If an existing record to be merged contained any notes, the notes are moved to the merged record.
- If an existing record to be merged was an entity record at the end of any links, the links are updated to reference the merged record.

  Note: Any links that were created through Analyst's Notebook Premium are also updated to reference the merged record.

During ingestion, the number of merge operations that occur is reported in the `MERGE_COUNT` column of the ingestion report.

The following diagrams demonstrate the merge operation.

In the first example of a merge operation, data in the staging table is merged into an existing i2 Analyze entity record because the correlation identifiers match.

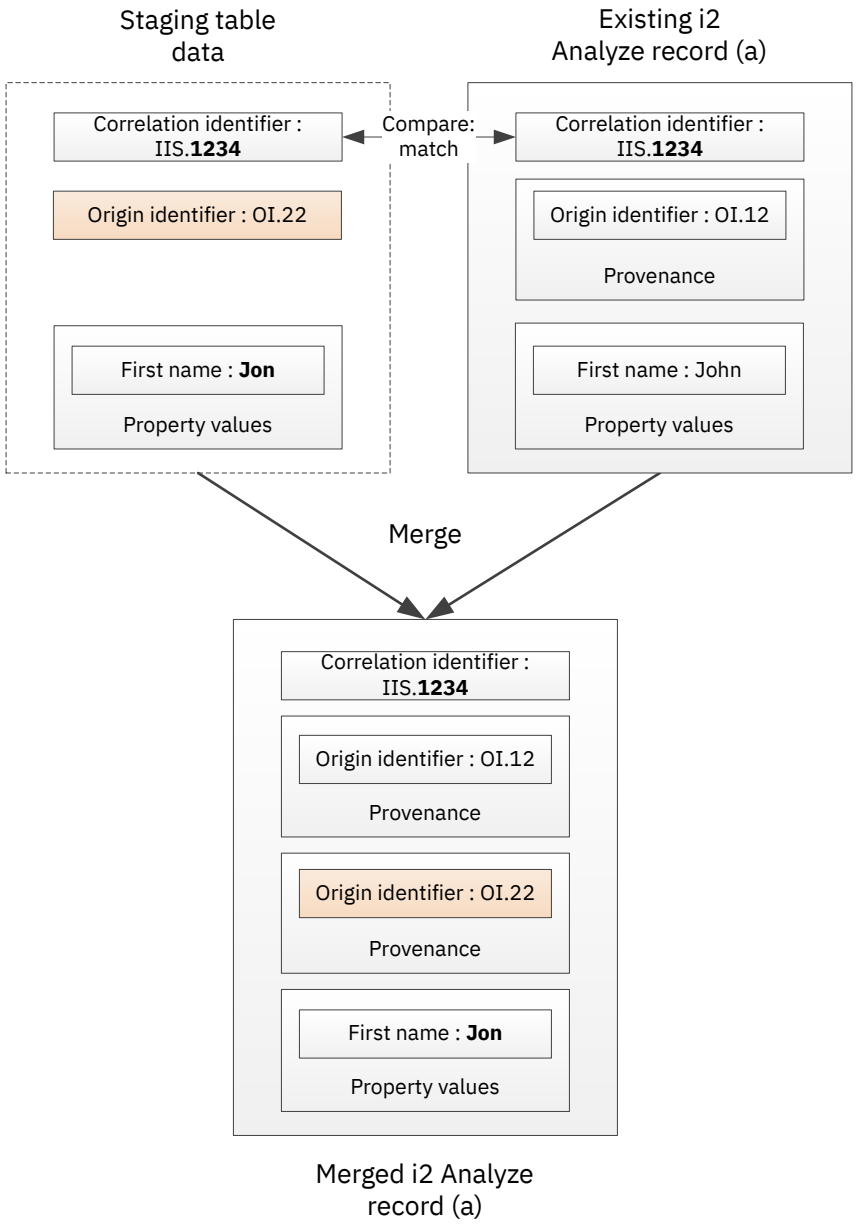

Figure 1: Incoming staging data merges with an existing i2 Analyze record.

In the diagram, the correlation identifiers of data in the staging table and the existing i2 Analyze record (a) match, which causes a merge operation. The existing i2 Analyze record (a) is not associated with the origin identifier of the incoming data. In this example, it is assumed that the staging table data is more recent than the existing data. As part of the merge, the property values from the data in

the staging table row are used. This results in a change to the value for the first name property from "John" to "Jon". The merged i2 Analyze record (a) now contains provenance for the origin identifier `OI.12` and one for the new data, `OI.22`.

In the second example of a merge operation, data in the staging table causes an update to an existing record (a) that changes the correlation identifier to match another record (b) in the Information Store, causing a merge.

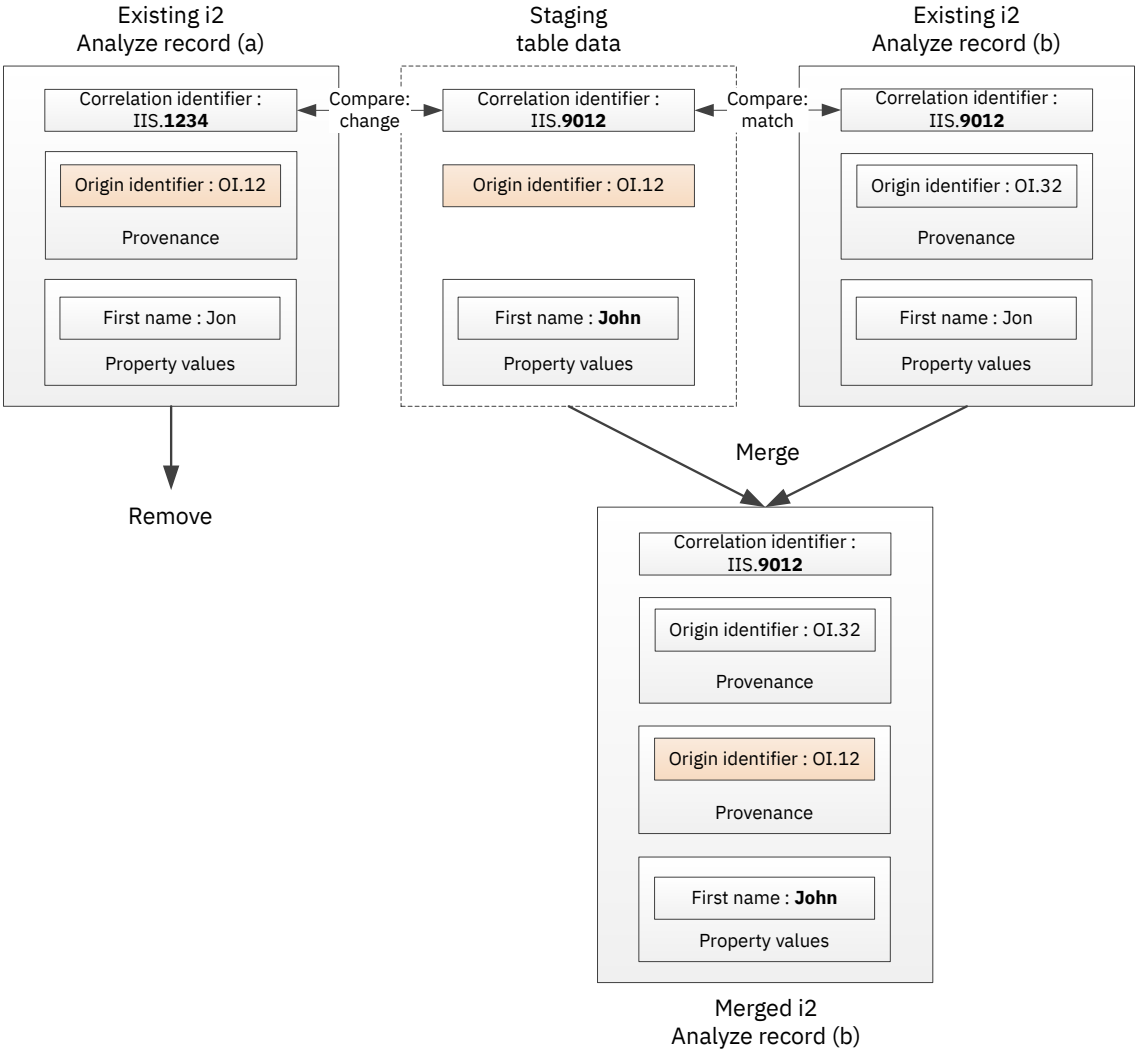

Figure 2: Incoming data updates an existing record, which causes the existing record to merge with another existing record in the Information Store. One of the existing records now has no provenance, and it is removed.

In the diagram, the data in the staging table has a different correlation identifier to the record (a) that it is currently associated with by its origin identifier, and the same correlation identifier as another existing record (b). This causes a merge operation. The first existing record (a) no longer has any provenance associated with it, and is removed. In this example, it is assumed that the staging table data is more recent than the existing data. As part of the merge, the property values from the staging

table row are used. This results in a change to the value for the first name property from "Jon" to "John". The merged i2 Analyze record (b) contains multiple pieces of provenance, one for the origin identifier `0I.32` and one for the new data, `0I.12`.

Unmerge

If the data for a merged i2 Analyze record is determined to no longer represent the same real-world object, the i2 Analyze record can be unmerged into two i2 Analyze records. For the Information Store to unmerge records, the correlation identifier or implicit discriminators of the data associated with that record must be changed.

Assuming that the implicit discriminators are compatible, the unmerge operation occurs when the correlation identifier of a row in the staging table is different from the correlation identifier on the merged record that it is currently associated with. As part of the unmerge operation, the piece of provenance for the staging table data is unmerged from its existing record.

After the unmerge operation, the following statements are true for the existing i2 Analyze record:

- The piece of provenance for the source information that caused the operation is unmerged from the record.
- The property values for the record are taken from the source information associated with the provenance that has the most recent value for `SOURCE_LAST_UPDATED`.

  If only one piece of provenance for a record has a value for the `SOURCE_LAST_UPDATED` column, the property values from the source information that is associated with that provenance are used. Otherwise, the property values to use are determined by the ascending order of the origin identifier keys that are associated with the record. The piece of provenance that is last in the order is chosen. To ensure data consistency, update your existing records with a value for the `SOURCE_LAST_UPDATED` column before you start to use correlation, and continue to update the value.

  If the default behavior does not match the requirements of your deployment, you can change the method for defining property values for merged records. For more information, see “Define how property values of merged records are calculated” on page 10.
- All notes remain on the record.
- The *last updated time* of the record is updated to the time that the unmerge operation occurred.
- If the existing record was an entity record at the end of any links, any links to the unmerged piece of provenance are updated to reference the record that now contains the provenance.

After the unmerge operation, depending on the change in correlation identifier that is presented to the Information Store, either a new i2 Analyze record is inserted or a merge operation is completed. During ingestion, this process is reported in the `UNMERGE_COUNT`, `INSERT_COUNT`, and `MERGE_COUNT` columns of the ingestion report.

The following diagrams demonstrate the unmerge operation. In each diagram, the data that is ingested from the staging table contains a different correlation identifier to the one on the record that it is associated with. One unmerge operation results in a new record, and one results in a merge operation.

In the first example of an unmerge, the correlation identifier of the provenance that is unmerged from an existing record (a) does not match with another correlation identifier in the staging table or the Information Store. A new i2 Analyze record (b) is created with the property values from the staging table.

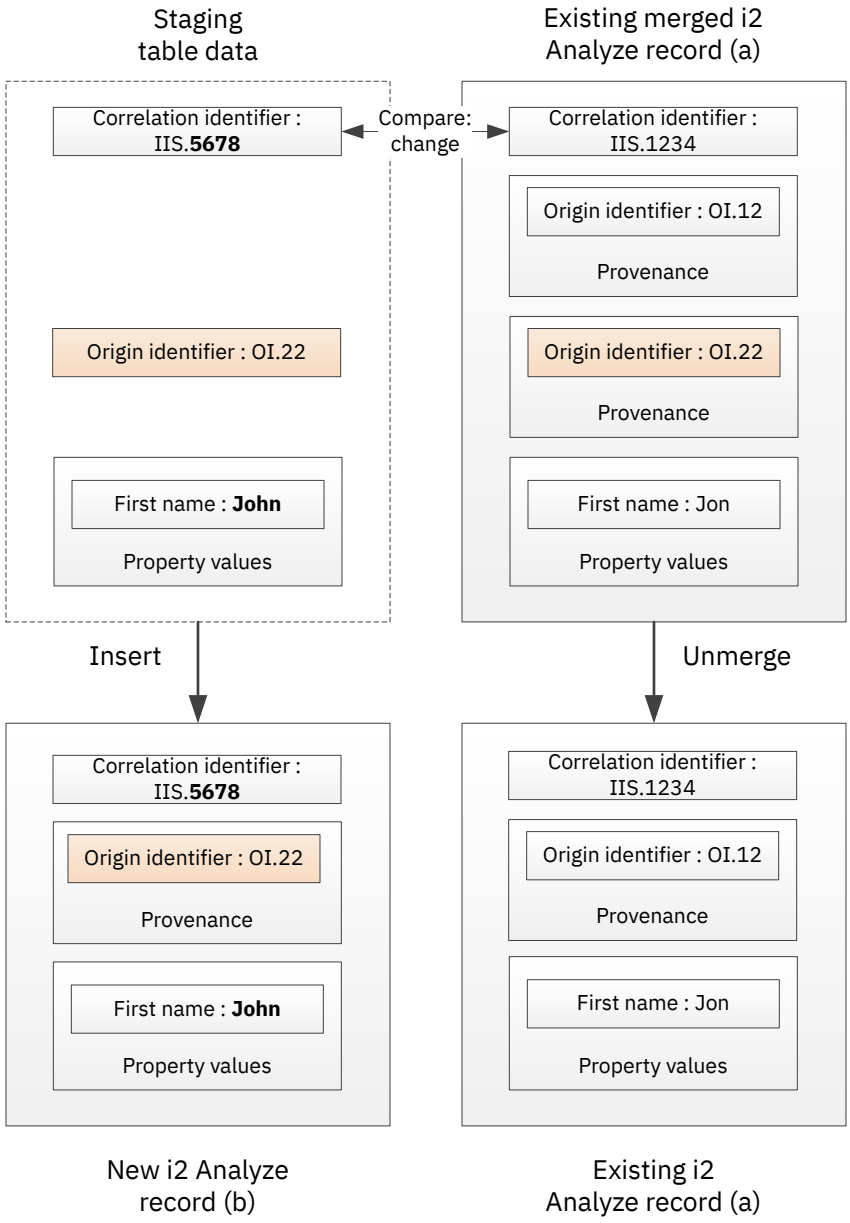

Figure 3: Incoming staging table data causes an unmerge operation. After the unmerge, a new record is inserted.

In the diagram, the data in the staging table has a different correlation identifier to the record (a) that it is currently associated with. This causes an unmerge operation. The provenance is unmerged from the existing record (a). The existing record (a) only contains the provenance for the origin identifier `OI.12` and the property values from the source information that is associated with that provenance. The correlation identifier of the staging table data does not match with any others in the Information Store, so a new record (b) is inserted.

In the second example of an unmerge, if the correlation identifier of the provenance that is unmerged from an existing record (a) now matches the correlation identifier of another record (b), a merge operation is performed.

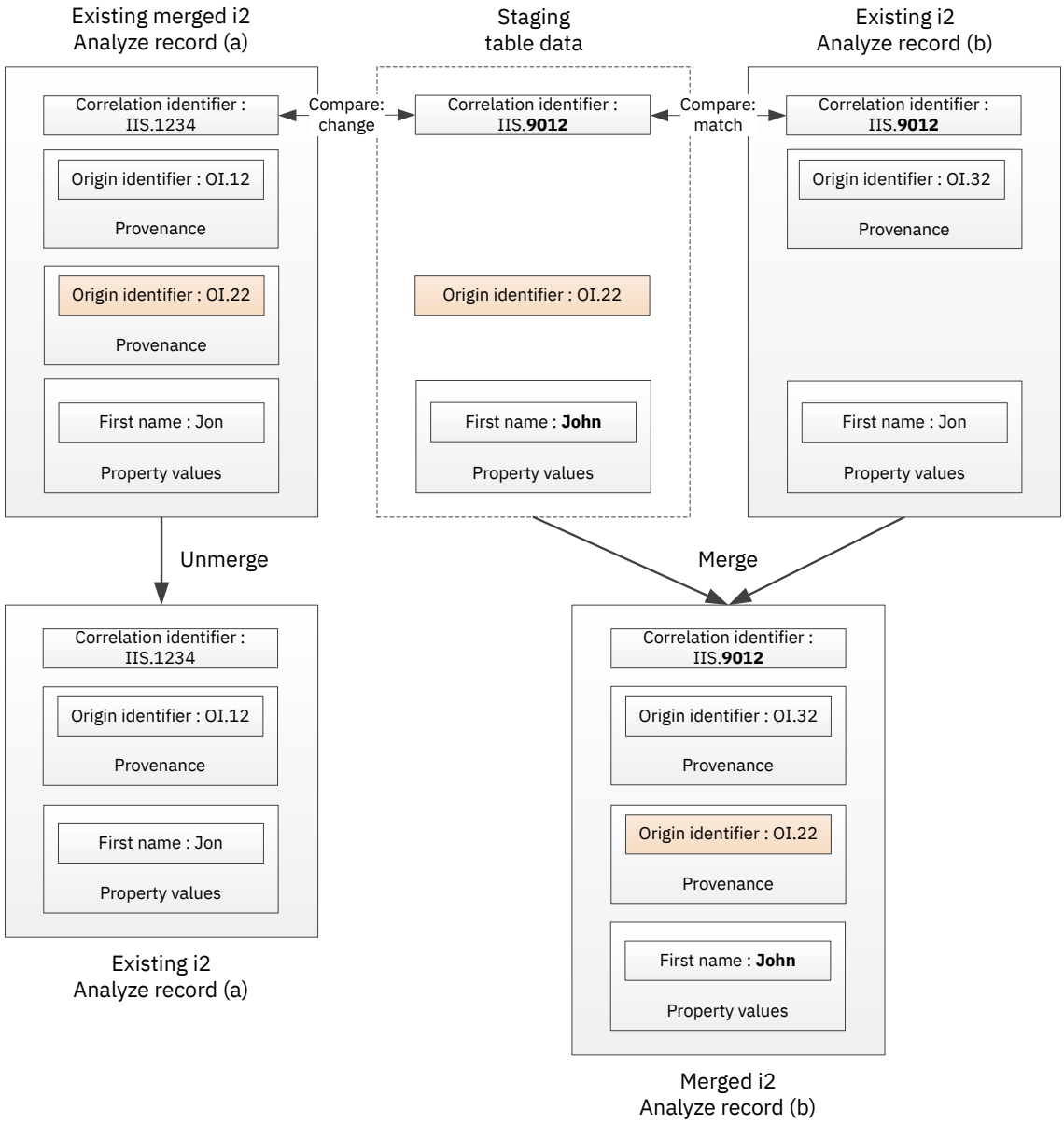

Figure 4: Incoming staging table data causes an unmerge operation. After the unmerge, a merge operation occurs.

In the diagram, the data in the staging table that is ingested has a different correlation identifier to the record (a) that it is currently associated with. This causes the unmerge operation. The origin identifier and provenance are unmerged from the existing record (a). The existing record (a) now only contains the provenance for the origin identifier `OI.12` and the property values from the source information that is associated with that provenance.

The correlation identifier of the staging table data now matches with another record (b) in the Information Store, so a merge operation occurs. In this example, it is assumed that the staging table data is more recent than the existing data. As part of the merge, the property values from the staging table row are used. This results in a change to the value for the first name property from "Jon" to "John". The merged i2 Analyze record (b) now contains two pieces of provenance, one for the origin identifier `OI.32` and one for the new data, `OI.22`.

For more information about the behavior of a merge operation, see “Merge” on page 4.

Hidden circular links

As a result of a merge operation, circular links might occur in the Information Store. The Information Store does not support circular links.

If data in the staging table causes existing records that are linked to each other in the Information Store to merge, the link becomes circular. As part of this operation, the existing link record is hidden in the Information Store. When a link record is hidden in the Information Store, analysts are unable to discover the link in search results and analysis.

A hidden link is revealed when the merged record at its ends is unmerged. After a hidden link is revealed, analysts are able to discover the link in search results and analysis.

Define how property values of merged records are calculated

In the Information Store, each property of an i2 Analyze record can have only one value. When multiple pieces of source data contribute to an i2 Analyze record, the system must calculate a single value for each property type.

Intended audience

The information about defining the property values of merged records is intended for users who are database administrators and experienced in SQL. To define the property values, you must write complex SQL view definition statements.

Important: You must write and test the SQL view statements for defining the property values of merged records in a non-production deployment of i2 Analyze. If you create an incorrect view, you might have to clear all the data from the system. Before you implement your view definitions in a production system, you must complete extensive testing in your development and test environments.

Why define the property values

When a record contains more than one piece of provenance, it is a merged record. The properties for an i2 Analyze record are calculated when a merge or unmerge operation occurs, or when provenance is removed from a record.

By default, all of the property values for a record come from the source data that contributed to the record with the most recent source-last-updated-time. If no source data has a source-last-updated-time, the property values to use are determined by the ascending order of the origin identifier keys that are associated with the record. The source data that is last in the order is chosen.

If the default source-last-updated-time behavior does not match the requirements of your deployment, you can define how the property values are calculated for merged i2 Analyze records. You might define your own rules when multiple data sources contain values for different properties of an item type or one data source is more reliable for a particular item or property type.

To demonstrate when it is useful to define how to calculate the property values for merged records, imagine that the following two pieces of source data contributed to an i2 Analyze record of type Person:

Origin Identifier	Correlation Identifier	Ingestion Source Name	Source Last Updated	First Given Name
DVLA1234	II1	DVLA	12:20:22 09/10/2018	John
PNC5678	II1	PNC	14:10:43 09/10/2018	Jon

In the default behavior, the property values are used from the source data with the most recent value for the `Source Last Updated` column. The property values from the row with the `Ingestion Source Name` of *PNC* are used for the merged i2 Analyze record, and the record gets the value of *Jon* for the `First Given Name` property.

If you know that data from the *DVLA* ingestion source is more reliable for this item type, you can define that source data with the value of *DVLA* for the `Ingestion Source Name` takes precedence. By using this definition, the i2 Analyze record gets the value of *John* for the `First Given Name` property.

After you define this rule for the Person entity type, all future updates to the records of this item type take the property values from the *DVLA* ingestion source if it is present in any of the source data that contributed to a merged i2 Analyze record.

For more information about how to enable this function, and create your own rules, see "Defining the property values of merged i2 Analyze records" on page 11.

Defining the property values of merged i2 Analyze records

By default, the mechanism for controlling the property values of merged records is not enable in a deployment of i2 Analyze. To define which property values are used for merged records, you must inform i2 Analyze that you intend to do this and then customize views in the Information Store database.

Before you begin

Important: You must write and test the SQL view statements for defining the property values of merged records in a non-production deployment of i2 Analyze. If you create an incorrect view, you might have to clear all the data from the system. Before you implement your view definitions in a production system, you must complete extensive testing in your development and test environments.

About this task

To inform i2 Analyze that you intend to define the property values of merged records, you must run the `enableMergedPropertyValues` toolkit task. You can take control of the property values for records of specific item types, or all item types in the i2 Analyze schema.

When you run the toolkit task, two views are created for each item type that you specify. The views are created in the `IS_PUBLIC` schema, and are named with the display name of each item type. The two views provide the mechanism for viewing the data that contributes to the merged i2 Analyze records, and for creating rules to calculate a single value for each property:

- The view whose name is suffixed with `_MCV` contains all the data that contributed to each merged i2 Analyze record of that item type. This is the merge contributors view. This view is designed for you

to inspect the column names and property values from the data in the Information Store. You must not modify the merge contributors view.

For more information about the merge contributors view, see "The merge contributors view" on page 13.

- The view whose name is suffixed with `_MPVDV` is where you can define the rules for calculating the property values of merged records. This is the merged property values definition view. You can modify this view to define how the merged property values are calculated from the data in the merge contributors view.

  When you generate the merged property values definition view, the default source-last-updated-time behavior is implemented in the view.

  For more information about modifying the definition, and some examples, see "The merged property values definition view" on page 15.

For example, the Person entity type from the example law enforcement schema produces the `IS_PUBLIC.E_PERSON_MCV` and `IS_PUBLIC.E_PERSON_MPVDV` views.

Procedure

1. Identify the item types that you want to define the property values for.

   For more information about why you might want to do this, see "Define how property values of merged records are calculated" on page 10.
2. Run the `enableMergedPropertyValues` toolkit task for each of the item types that you identified in step 1.
   a) Open a command prompt and navigate to the `toolkit/scripts` directory.
   b) Run the `enableMergedPropertyValues` toolkit task:
   For example, to create the views for the Person entity type from the example law enforcement schema, run the following command:

   ```
   setup -t enableMergedPropertyValues -p schemaTypeId=ET5
   ```

   You can run the toolkit task with `schemaTypeId=`*`ItemTypeId`* or without any arguments to create the views for all item types in the i2 Analyze schema. You can run the task multiple times with different item type identifiers.
3. Inspect the merge contributors view (`_MCV`) that is created in the `IS_PUBLIC` schema and identify any rules to create for calculating the property values of merged records.

   For more information about the merge contributors view, see "The merge contributors view" on page 13.
4. Modify the SQL create statement for the merged property values definition view (`_MPVDV`) that is created in the `IS_PUBLIC` schema to create a view that implements the rules you identified in step 3.

   For more information about modifying the definition, and some examples, see "The merged property values definition view" on page 15.

What to do next

After you ingest data that causes correlation operations, ensure that the property values for the merged i2 Analyze records match your expectations by searching for them. If you can ingest data successfully, and the property values are correct in your testing, the merged property values definition view is correct. If the property values are not correct in all cases, you must continue to modify the merged property values definition view and test the results until your requirements are met.

It is recommended that you keep a backup of your complete merged property values definition view.

During the testing of your deployment, you might need to clear any test data from the system. It is useful to clear test data from the system when you are developing your merged property values definition view, to ensure that your view is behaving correctly. For more information, see Clearing data from the system.

You must maintain the merged property values definition view for the lifetime of your deployment. If you change the i2 Analyze schema file, you must update the SQL statement for your view to match any changes. For example, if you add a property type to an item type, the new property must be added to the SQL statement for your merged property values definition view. You do not need to update the merge contributors view. The merge contributors view is updated when the schema is updated so that you can inspect the column name for the new property. For more information, see "How to maintain your merged property values definition view" on page 18.

After you test the view definition in your non-production database, you can implement the view on your production database.

The merge contributors view

In the merge contributors view you can see the source data that contributes to merged i2 Analyze records. By inspecting the view, you can gain an understanding about the data in your system and identify any rules to create that change the way that the property values are calculated.

To understand the merge contributors view, you must first understand the identifiers that the Information Store and i2 Analyze records use. For more information about i2 Analyze identifiers, see [data model section].

The merge contributors view contains the following details about each piece of source data:

ITEM_ID

The identifier that the Information Store adds to each i2 Analyze record that distinguishes records of a particular type uniquely within the Information Store. Multiple rows in the merge contributors view can have the same item identifier. Each row that has the same item identifier represents one piece of source data that contributed to the i2 Analyze record with that identifier.

In the merge contributors view, the item identifier is stored in the `ITEM_ID` column.

ORIGIN_ID_TYPE and ORIGIN_ID_KEYS

The origin identifier is used to reference data in its original source, and to identify that data throughout an i2 Analyze deployment. Each piece of source data has a unique origin identifier.

In the merge contributors view, the origin identifier is stored in the `ORIGIN_ID_TYPE` and `ORIGIN_ID_KEYS` columns.

SOURCE_LAST_UPDATED

When data is ingested into the Information Store, you can provide values for when the data was last updated in the data source from which it originated. The values in this column are used for the default source-last-updated-time behavior.

In the merge contributors view, the last updated time is stored in the `SOURCE_LAST_UPDATED` column.

Property values

There is a column for each property type in the i2 Analyze schema. Each column is the display name of the property type prefixed with `P_`.

For example, values for the First Given Name property type for a Person entity type from the example law enforcement schema are stored in the `P_FIRST_GIVEN_NAME` column.

CORRELATION_ID_TYPE and CORRELATION_ID_KEY

The correlation identifier is used to indicate that data is about a specific real-world object. For multiple pieces of data to contribute to the same i2 Analyze record, they must have the same correlation identifier.

In the merge contributors view, the correlation identifier is stored in the `CORRELATION_ID_TYPE` and `CORRELATION_ID_KEY` columns.

INGESTION_SOURCE_NAME

When data is ingested into the Information Store, you must provide a data source name that identifies the data source from which it originated. This is stored as the ingestion source name in the Information Store.

In the merge contributors view, the ingestion source name is stored in the `INGESTION_SOURCE_NAME` column.

In most scenarios, the `ORIGIN_ID_TYPE`, `ORIGIN_ID_KEYS`, `SOURCE_LAST_UPDATED`, and `INGESTION_SOURCE_NAME` values can be used for choosing the source data to provide the property values.

The following table shows an example merge contributors view for a Person entity type where two pieces of source data contribute to the same i2 Analyze record:

ITEM_ID	1	1
ORIGIN_ID_TYPE	DVLA	PNC
ORIGIN_ID_KEYS	1234	5678
SOURCE_LAST_UPDATED	12:20:22 09/10/2018	14:10:43 09/10/2018
P_FIRST_GIVE_NAME	John	Jon
P_MIDDLE_NAME	James	
P_...	...	...
CORRELATION_ID_TYPE	II	II
CORRELATION_ID_KEY	1	1
INGESTION_SOURCE_NAME	DVLA	PNC

In this example merge contributors view, you can see the following details:

- The data originates from two ingestion sources; DVLA and PNC
- The data from the PNC ingestion source was updated more recently than the data from the DVLA ingestion source
- The data from the PNC ingestion source is missing a value for the middle name property

You might already know some of these details about the data in your system, or you might be able to determine these differences from inspecting the contents of the view. If you decide that you want to change the default behavior for calculating property values, you must customize the merged property values definition view. For more information about how to customize the merged property values definition view, see "The merged property values definition view" on page 15.

The merged property values definition view

The merged property values definition view is used to calculate the property values of merged i2 Analyze records. You can modify the view to define which property values records receive.

The merged property values definition view is used to calculate a single value for each property type for each merged record.

After you inspect the merge contributors view and identify the rules that you want to define, you must modify the SQL statement to create a view that matches your requirements. For more information about the merge contributors view, see "The merge contributors view" on page 13.

After you create a merged property values definition view, you must maintain the view through the lifetime of your deployment. For more information, see "How to maintain your merged property values definition view" on page 18.

Requirements of the view

The merged property values definition view is over the merge contributors view. To produce a single value for each property type, the view must contain one row for each `ITEM_ID` and populate each column with a value. You can include NULL values for property types that have no value.

After you customize the merged property values definition view, it must conform to the following requirements:

- The view must have columns for the item identifier and every property type column in the merge contributors view. The column names in both views must be the same.

  The merged property values definition view must not contain the `ORIGIN_ID_TYPE`, `ORIGIN_ID_KEYS`, `SOURCE_LAST_UPDATED`, `CORRELATION_ID_TYPE`, `CORRELATION_ID_KEYS`, or `INGESTION_SOURCE_NAME` columns.
- Each item identifier must be unique.

  Important: You cannot ingest data into the Information Store if the view contains multiple rows with the same item identifier.

Default behavior

By default, the merged property values definition view uses the default source-last-updated-time behavior to define which source data the property values come from. You can see the SQL statement

for the view in IBM Data Studio. An example of the SQL statement for the generated view for the Person entity from the example law enforcement schema is:

```
SELECT item_id, p_unique_reference, p_title,
     ... (all property type columns) ...
     p2_issued_date_and_time, p3_issued_date_and_time
   FROM
   (
       SELECT MCV.*,
            ROW_NUMBER ( ) OVER (PARTITION BY item_id
                                 ORDER BY source_last_updated DESC NULLS LAST,
                                          origin_id_keys DESC,
                                          origin_id_type DESC)
                           AS partition_row_number
       FROM IS_PUBLIC.E_Person_MCV AS MCV
   )
   WHERE partition_row_number = 1
```

The SQL statement for the default view works in the following way:

- To ensure that the view contains the correct columns, the first SELECT statement returns the ITEM_ID and all property type columns. For example, P_TITLE.
- To split the rows into groups that contain only the data that contributed to a single merged record, the OVER and PARTITION BY clauses are used on the ITEM_ID column.
- To define the value for each property, the ROW_NUMBER function and ORDER BY clause are used. For the default behavior, the ORDER BY clause is used on the SOURCE_LAST_UPDATED, ORIGIN_ID_KEYS, AND ORIGIN_ID_TYPE columns. Each column is in descending order. By using NULLS LAST, you ensure that if the SOURCE_LAST_UPDATED column does not contain a value, it is listed last.

  The ROW_NUMBER function returns the number of each row in the partition. After the rows are ordered, the first row is number 1. The initial SELECT statement uses a WHERE clause to select the values for each row with a value of 1 for partition_row_number. Because only one row has a value of 1, the view contains only one row for each item identifier.

Examples

You can modify the following example views to match the data in your i2 Analyze schema or to meet your property value requirements:

Ingestion Source Name precedence

In this example, the data is ordered so that a specified ingestion source takes precedence. For example, the *DVLA* ingestion source takes precedence over the *PNC* one.

```
SELECT item_id, p_unique_reference, p_title,
     ... (all property type columns) ...
     p2_issued_date_and_time, p3_issued_date_and_time
   FROM
   (
       SELECT MCV.*,
            ROW_NUMBER ( ) OVER (PARTITION BY item_id
                                 ORDER BY CASE ingestion_source_name
                                             WHEN 'DVLA' THEN 1
                                             WHEN 'PNC' THEN 2
                                             ELSE 3
                                          END)
                           AS partition_row_number
       FROM IS_PUBLIC.E_Person_MCV AS MCV
   )
   WHERE partition_row_number = 1
```

If an ingestion source name does not match *DVLA* or *PNC*, it is ordered last. If multiple contributing pieces of data have the same ingestion source name, the behavior is non-deterministic. In a production system, you must include another clause that provides deterministic ordering in all scenarios.

To order the ingestion sources, the `ORDER BY CASE` clause is used. The rows are ordered by the value that they are assigned in the `CASE` expression. For more information about the `ORDER BY CASE` statement, see ORDER BY clause and CASE expression.

The `ROW_NUMBER` function and the `OVER` and `PARTITION BY` clauses are used in the same was as the view for the default behavior.

Source last updated and non-NULL

In this example, the data is ordered by the source last updated time. However if there is a NULL value for a particular property type, the value is taken from the data that has the next most recent

time. This process is completed until a value for the property is found, or no data is left for that record.

```
SELECT DISTINCT item_id,

       FIRST_VALUE(p_first_given_name, 'IGNORE NULLS')
                      OVER (  PARTITION BY    item_id
                              ORDER BY        source_last_updated DESC NULLS
LAST
                              RANGE BETWEEN   UNBOUNDED PRECEDING AND
UNBOUNDED FOLLOWING)
                      AS p_first_given_name,

... (FIRST_VALUE functions for all property types) ...

       FIRST_VALUE(CAST(p_additional_informatio AS VARCHAR(1000)), 'IGNORE
NULLS')
                      OVER (  PARTITION BY    item_id
                              ORDER BY        source_last_updated DESC NULLS
LAST
                              RANGE BETWEEN   UNBOUNDED PRECEDING AND
UNBOUNDED FOLLOWING)
                      AS p_additional_informatio

FROM        IS_Public.E_Person_MCV
```

This example uses the FIRST_VALUE function to identify which data has the first non-NULL value for a property type. The FIRST_VALUE function returns the first value in an ordered set of values. You must specify a scalar expression, and PARTITION_BY, ORDER_BY, and RANGE clauses. In the example, the scalar expressions are the property type columns. The PARTITION_BY and ORDER_BY clauses are similar to the other examples, where they act on the item identifier and the source last updated time. The row range clause ensures that only one row is returned. For more information about the FIRST_VALUE function, see OLAP specification - FIRST_VALUE.

To use the FIRST_VALUE function, you must define a function for each property type separately. In the above example, there are two examples of the FIRST_VALUE function for two property types:

- The first acts on the P_FIRST_GIVEN_NAME column.
- The second acts on the P_ADDITIONAL_INFORMATIO column. The same process is completed, however in the merge contributors view the P_ADDITIONAL_INFORMATIO column is of data type CLOB(32M). This data type is not supported by the FIRST_VALUE function. To use this column, you must cast it into a supported data type. For example, VARCHAR(1000). You must ensure that if a column is cast, that you do not lose any data.

How to maintain your merged property values definition view

The merged property values definition views include all of the property types for an item type as they were when the view was created. If you change the i2 Analyze schema to add a property type to an item type, the merged property values definition view is now incorrect. The merged property values definition view does not contain a column for the new property type, and therefore cannot populate an i2 Analyze record with a value for that property. When you add property types to the i2 Analyze schema for an item type, you must update any merged property values definition view to include any

new property types. i2 Analyze updates the merge contributors view when the schema is updated, you can use the updated merge contributors view to see the name of the new column.

Note: If you do not update your merged property values definition view, you cannot ingest data into the Information Store.

To stop controlling the property values of merged records for an item type, you can inform i2 Analyze that you no longer intend to use this function. Inform i2 Analyze by running the `disableMergedPropertyValues` toolkit task. You can specify `schemaTypeId=`*`ItemTypeId`* to remove the views for one item type only, or without any arguments to remove the views for every item type in the i2 Analyze schema. After you run the toolkit task, the default source-last-updated-time behavior is used.

You can modify the merged property values definition view or run the `disableMergedPropertyValues` toolkit task at any time. The property values of i2 Analyze records that are already in the Information Store are not updated. If you ingest any data that is associated with an i2 Analyze record, the property values for that record are recalculated with the new merged property values definition view.

Ingesting example correlation data

The correlation example provides sets of data that were passed through a matching engine so that each row of data in the set is associated with a correlation identifier. You can ingest the example data, and then inspect the items in the Information Store from Analyst's Notebook Premium to demonstrate the correlation behavior.

About this task

Use the `ingestExampleData` toolkit task to ingest example data sets that demonstrate the correlation behavior in i2 Analyze. For more information about the operations that occur, and how the i2 Analyze records are effected, see “Correlation operations” on page 4.

`law-enforcement-data-set-2`

This data set contains data with correlation identifiers. After you ingest this data set, the Information Store database contains i2 Analyze records with correlation identifiers. No correlation operations occur when you ingest this data set.

`law-enforcement-data-set-2-merge`

This data set contains data that causes a number of merge operations to occur with some of the i2 Analyze records that were created from the first data set.

`law-enforcement-data-set-2-unmerge`

This data set contains data that causes a number of unmerge operations to occur with some of the i2 Analyze records that were merged from the second data set.

You can find the example data sets in the `toolkit\examples\data` directory.

Procedure

1. In a command prompt, navigate to the `toolkit\scripts` directory.
2. Ingest the first example data set.
   a) Run the following command to ingest the first example data set:

```
setup -t ingestExampleData -e law-enforcement-data-set-2
```

The `examples\data\law-enforcement-data-set-2` directory contains a set of CSV files with data that was passed through a matching engine and processed to meet the requirements of the staging tables. Each row of data contains a correlation identifier type and a unique key value. The `LoadCSVDataCommands.db2` file is called to populate the staging tables with the example data.

The Information Store now contains i2 Analyze records with correlation identifiers. However, no correlation operations occurred during the ingestion.

b) In Analyst's Notebook Premium, search for "Julia Yochum" and add the returned entity to the chart.

3. Ingest the `law-enforcement-data-set-2-merge` data to demonstrate merge operations.

a) Run the following command to ingest the second example data set:

```
setup -t ingestExampleData -e law-enforcement-data-set-2-merge
```

The `examples\data\law-enforcement-data-set-2-merge` directory contains a set of CSV files with data that was passed through a matching engine and processed to meet the requirements of the staging tables. Each row of data contains a correlation identifier type and key value. Each row contains a unique value for the `SOURCE_ID`, which represents the origin identifier.

In this scenario, the matching engine identified that some of the data represents the same real-world objects as data in the first data set. As part of this process, the correlation identifiers match with some of the existing data in the database, but the origin identifiers are different. These matches cause a number of merge operations to occur during the ingestion.

For example, you can see on line 2 of the `person.csv` file the correlation identifier key is `person_0`, which matches the correlation identifier of the record for person "Julia Yochum". This match causes a merge between the existing record and the incoming row of data. As part of the merge, the property values from the incoming row of data are used for the record, this results in the full name changing to "Julie Yocham". This is an example of the scenario that is described in Figure 1 on page 5.

In the ingestion reports, you can see the number and type of correlation operations that occurred during the ingestion. For more information about understanding the ingestion reports, see Understanding ingestion reports.

b) In Analyst's Notebook Premium, select the chart item that represents Julia Yochum and click **Get changes**.

You can see that the name changes due to the merge operation described previously.

4. Ingest the `law-enforcement-data-set-2-unmerge` data to demonstrate unmerge operations.

a) Run the following command to ingest the third example data set:

```
setup -t ingestExampleData -e law-enforcement-data-set-2-unmerge
```

The `examples\data\law-enforcement-data-set-2-unmerge` directory contains a set of CSV files with data that was passed through a matching engine and processed to meet the requirements of the staging tables.

In this scenario, the matching engine identified that some of the data no longer represents the same real-world objects as it did previously. As part of this process, the correlation identifiers of previously merged data are changed. These changes cause a number of unmerge operations to occur during the ingestion.

In this example, the correlation identifier of the data that was ingested has changed, but the origin identifier remained the same. For example, on line 2 of the `person.csv` file, the correlation identifier key is now `person_1101` for origin identifier `PER:GEN\1101`. Before, this value was `person_0`. This causes an unmerge operation on the record that the data is currently associated with. This is an example of the scenario that is described in Figure 3 on page 8.

You can see the number of unmerge operations that occurred in the ingestion reports.

b) In Analyst's Notebook Premium, search for "Julie Yocham".

There are now two entities for Julie Yocham.

What to do next

After you investigate the correlation behavior, you can clear the example data from your system. For more information, see Clearing data from the system.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing IBM Corporation North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM United Kingdom Limited Hursley House Hursley Park Winchester, Hants, SO21 2JN UK

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other names may be trademarks of their respective owners. Other company, product, and service names may be trademarks or service marks of others.