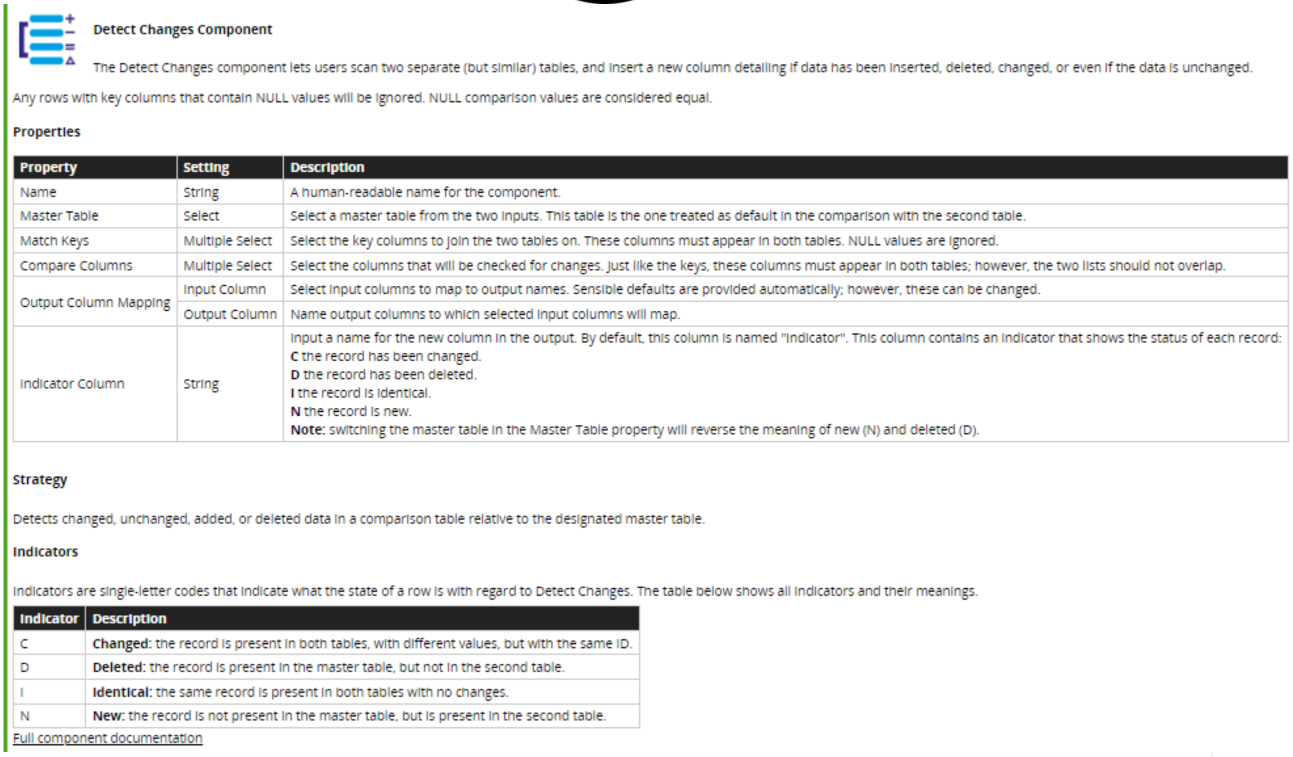
Making the most out of Matillion's "Detect Changes" Component

Businesses across industries rely heavily on timely and accurate insights to make informed decisions. However, with the exponential growth of data, managing and processing it efficiently poses significant challenges.

Matillion’s "Detect Changes" component helps teams track and manage data modifications effectively. This article explains the component capabilities and how you can use them to improve your data transformation process.

Understanding the "Detect Changes" Component

At its core, Matillion’s "Detect Changes" component identifies and classifies data alterations within datasets. Whether you're dealing with incremental updates or monitoring real-time changes, this component empowers users to pinpoint modifications efficiently. It operates by comparing incoming data against existing records, enabling you to isolate additions, updates, and deletions seamlessly.



Detect Changes Component

The Detect Changes component lets users scan two separate (but similar) tables, and insert a new column detailing if data has been inserted, deleted, changed, or even if the data is unchanged.

Any rows with key columns that contain NULL values will be ignored. NULL comparison values are considered equal.

Properties

Property	Setting	Description
Name	String	A human-readable name for the component.
Master Table	Select	Select a master table from the two inputs. This table is the one treated as default in the comparison with the second table.
Match Keys	Multiple Select	Select the key columns to join the two tables on. These columns must appear in both tables. NULL values are ignored.
Compare Columns	Multiple Select	Select the columns that will be checked for changes. Just like the keys, these columns must appear in both tables; however, the two lists should not overlap.
Output Column Mapping	Input Column	Select input columns to map to output names. Sensible defaults are provided automatically; however, these can be changed.
	Output Column	Name output columns to which selected input columns will map.
Indicator Column	String	Input a name for the new column in the output. By default, this column is named "Indicator". This column contains an indicator that shows the status of each record: **C** the record has been changed. **D** the record has been deleted. **I** the record is identical. **N** the record is new. **Note:** switching the master table in the Master Table property will reverse the meaning of new (N) and deleted (D).

Strategy

Detects changed, unchanged, added, or deleted data in a comparison table relative to the designated master table.

Indicators

Indicators are single-letter codes that indicate what the state of a row is with regard to Detect Changes. The table below shows all indicators and their meanings.

Indicator	Description
C	**Changed:** the record is present in both tables, with different values, but with the same ID.
D	**Deleted:** the record is present in the master table, but not in the second table.
I	**Identical:** the same record is present in both tables with no changes.
N	**New:** the record is not present in the master table, but is present in the second table.

Full component documentation

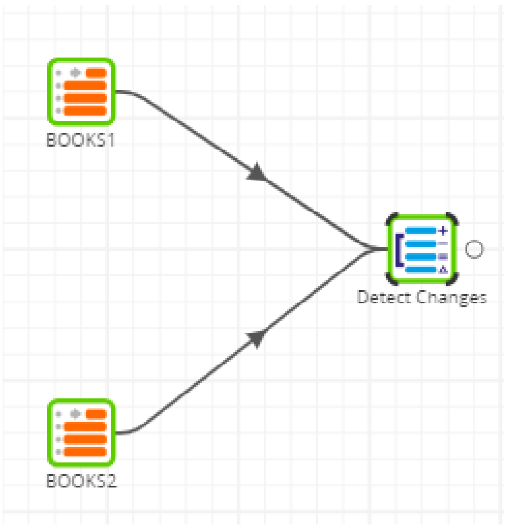
Key Features and Benefits

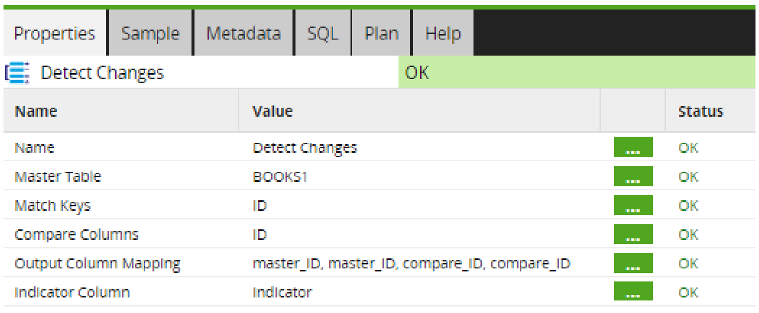
1. **Incremental Data Processing**: One of the primary use cases for the "Detect Changes" component is incremental data processing. Instead of processing entire datasets repeatedly, which can be time-consuming and resource-intensive, you can focus on processing only the new or modified records. This approach not only optimizes performance but also reduces processing overhead.
2. **Change Data Capture (CDC)**: For organizations dealing with high-velocity data streams, such as those from transactional databases or event logs, CDC becomes indispensable. Matillion's "Detect Changes" component facilitates CDC by efficiently capturing and propagating data modifications in real-time or near real-time. By staying synchronized with evolving data sources, you can ensure that downstream analytics and reporting reflect the latest changes accurately.
3. **Data Warehousing and ETL Pipelines**: Integrating the "Detect Changes" component into your data warehousing and ETL (Extract, Transform, Load) pipelines enhances data governance and integrity. By identifying and logging changes systematically, you establish an audit trail that aids in tracking data lineage and ensuring compliance with regulatory requirements. Moreover, by automating the detection of

changes, you reduce the risk of human error and improve overall data quality.

4. **Flexible Configuration Options**: Matillion offers a range of configuration options within the "Detect Changes" component, allowing users to tailor the detection logic to suit specific use cases. Whether you need to define custom key fields for comparison or specify tolerance levels for detecting incremental changes, the platform provides the flexibility to accommodate diverse requirements.

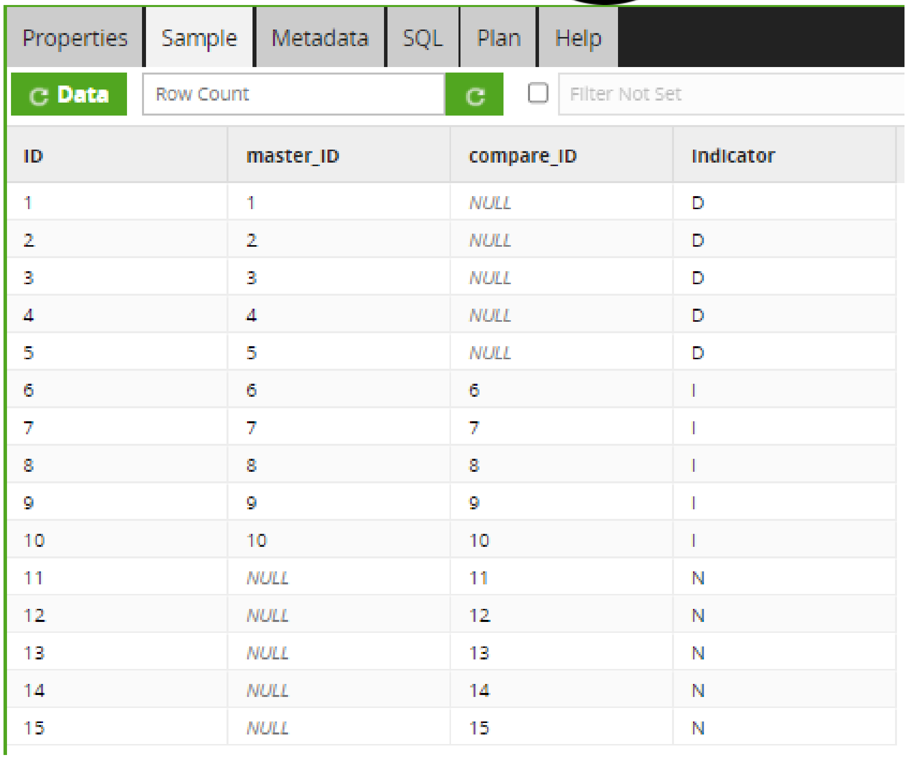
Example Use Case :





Detect Changes OK

Name	Value		Status
Name	Detect Changes	...	OK
Master Table	BOOKS1	...	OK
Match Keys	ID	...	OK
Compare Columns	ID	...	OK
Output Column Mapping	master_ID, master_ID, compare_ID, compare_ID	...	OK
Indicator Column	Indicator	...	OK

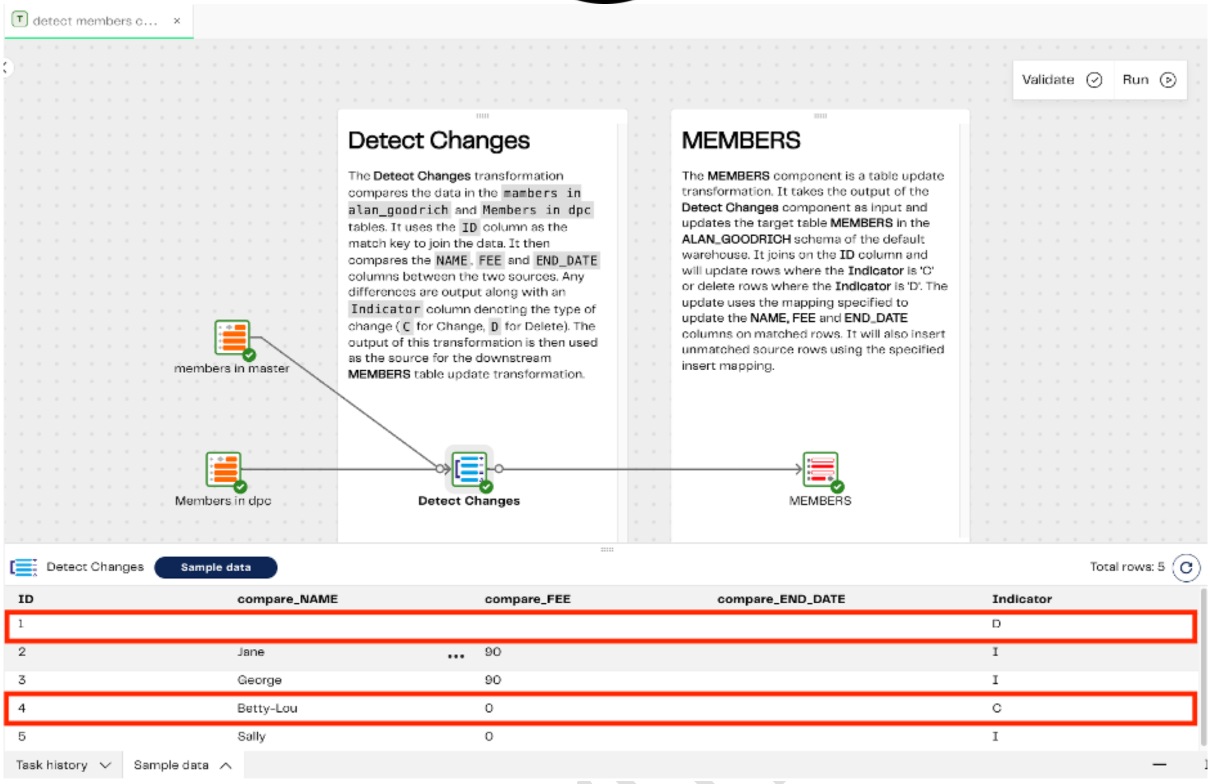


ID	master_ID	compare_ID	Indicator
1	1	NULL	D
2	2	NULL	D
3	3	NULL	D
4	4	NULL	D
5	5	NULL	D
6	6	6	I
7	7	7	I
8	8	8	I
9	9	9	I
10	10	10	I
11	NULL	11	N
12	NULL	12	N
13	NULL	13	N
14	NULL	14	N
15	NULL	15	N

In another example, we have a members table in one schema in Snowflake (Members in dpc) where data is Inserted, Updated and Deleted. We have the same table (members in master) in another schema that needs to apply any of the changes made in the table, Members in dpc. We use these two tables as input to the Detect Changes Component which will add an indicator column populating C for changed records, D for deleted records, I for identical records and N for new records.

The result for the Detect Changes Component will be used as input to the Table Update Component to apply the changes to the table members in master.

Notice the record with ID 4 pictured below, this has a record that has changed and needs to be applied to the table members in master.



Detect Changes

The **Detect Changes** transformation compares the data in the `mambers in alan_goodrich` and `Members in dpc` tables. It uses the `ID` column as the match key to join the data. It then compares the `NAME`, `FEE` and `END_DATE` columns between the two sources. Any differences are output along with an `Indicator` column denoting the type of change (`C` for Change, `D` for Delete). The output of this transformation is then used as the source for the downstream **MEMBERS** table update transformation.

MEMBERS

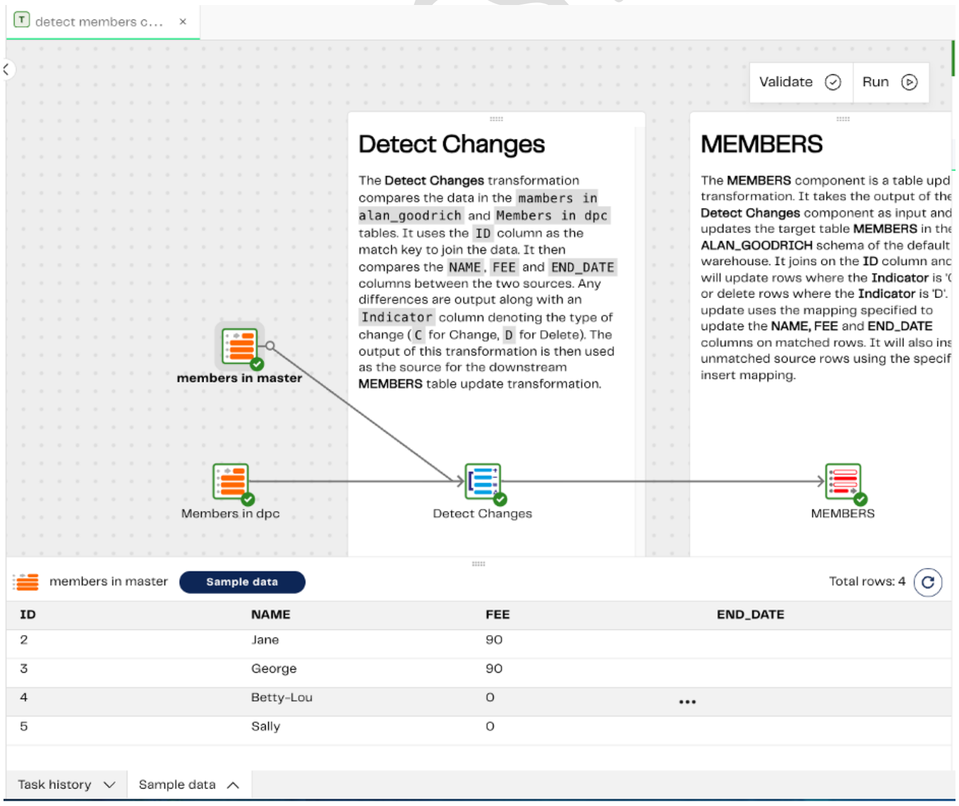
The **MEMBERS** component is a table update transformation. It takes the output of the **Detect Changes** component as input and updates the target table **MEMBERS** in the **ALAN_GOODRICH** schema of the default warehouse. It joins on the **ID** column and will update rows where the **Indicator** is 'C' or delete rows where the **Indicator** is 'D'. The update uses the mapping specified to update the **NAME, FEE** and **END_DATE** columns on matched rows. It will also insert unmatched source rows using the specified insert mapping.

Detect Changes **Sample data** Total rows: 5

ID	compare_NAME	compare_FEE	compare_END_DATE	Indicator
1				D
2	Jane	90		I
3	George	90		I
4	Betty-Lou	0		C
5	Sally	0		I

Task history Sample data

When we sample the members in the master table again we can see record 1 has been deleted and record 4 has been changed



Detect Changes

The **Detect Changes** transformation compares the data in the `mambers in alan_goodrich` and `Members in dpc` tables. It uses the `ID` column as the match key to join the data. It then compares the `NAME`, `FEE` and `END_DATE` columns between the two sources. Any differences are output along with an `Indicator` column denoting the type of change (`C` for Change, `D` for Delete). The output of this transformation is then used as the source for the downstream **MEMBERS** table update transformation.

MEMBERS

The **MEMBERS** component is a table upd transformation. It takes the output of the **Detect Changes** component as input and updates the target table **MEMBERS** in the **ALAN_GOODRICH** schema of the default warehouse. It joins on the **ID** column and will update rows where the **Indicator** is ' or delete rows where the **Indicator** is 'D'. update uses the mapping specified to update the **NAME, FEE** and **END_DATE** columns on matched rows. It will also ins unmatched source rows using the specif insert mapping.

members in master **Sample data** Total rows: 4

ID	NAME	FEE	END_DATE
2	Jane	90	
3	George	90	
4	Betty-Lou	0	
5	Sally	0	

Task history Sample data

Practical Applications of Change Detection

1. **E-commerce**: In the realm of e-commerce, where product catalogs and customer data undergo frequent updates, the "Detect Changes" component facilitates inventory management, order processing, and customer relationship management. By tracking changes in product availability, pricing, and customer preferences, businesses can optimize marketing strategies and enhance customer experiences.
2. **Financial Services**: In the financial services sector, where data accuracy and regulatory compliance are paramount, the "Detect Changes" component plays a crucial role in transaction processing, risk management, and regulatory reporting. By capturing and analyzing changes in financial transactions and market conditions in real time, organizations can mitigate risks and capitalize on emerging opportunities.
3. **Healthcare**: In healthcare settings, where patient data constantly evolves, the "Detect Changes" component enables healthcare providers to maintain up-to-date electronic health records (EHRs), track treatment outcomes, and identify trends in patient demographics and healthcare utilization. By leveraging real-time data insights, healthcare organizations can improve clinical decision-making and patient outcomes.

Conclusion

Matillion's Detect Changes component offers a powerful solution for managing data modifications effectively across various industries and use cases. Organizations can streamline data transformation workflows, enhance data quality and integrity, and unlock actionable insights from their data. Whether you're dealing with incremental updates, real-time data streams, or complex data pipelines, integrating the "Detect Changes" component into your data infrastructure empowers you to stay ahead in today's data-driven landscape.