Metadb 1.0: an open-source data platform for analytics

Nassib Nassar
Director of Metadb Analytics Platform
Head of Research
Index Data ApS

Overview of Metadb

Metadb is open-source software that continuously reads data from transaction processing databases or other dynamic data sources, and helps with integrating the data within its own database to support analytics.

Overview of Metadb

Metadb is open-source software that continuously reads data from transaction processing databases or other dynamic data sources, and helps with integrating the data within its own database to support analytics. The internal databases of FOLIO and ReShare are examples of data sources that Metadb can read from.

Overview of Metadb

Metadb is open-source software that continuously reads data from transaction processing databases or other dynamic data sources, and helps with integrating the data within its own database to support analytics. The internal databases of FOLIO and ReShare are examples of data sources that Metadb can read from. When state changes occur in FOLIO's database, Metadb updates its database correspondingly, with a few differences:

Overview of Metadb

Metadb is open-source software that continuously reads data from transaction processing databases or other dynamic data sources, and helps with integrating the data within its own database to support analytics. The internal databases of FOLIO and ReShare are examples of data sources that Metadb can read from. When state changes occur in FOLIO's database, Metadb updates its database correspondingly, with a few differences:

- No overwrite: deleted data are preserved and timestamped.

Overview of Metadb

Metadb is open-source software that continuously reads data from transaction processing databases or other dynamic data sources, and helps with integrating the data within its own database to support analytics. The internal databases of FOLIO and ReShare are examples of data sources that Metadb can read from. When state changes occur in FOLIO's database, Metadb updates its database correspondingly, with a few differences:

- No overwrite: deleted data are preserved and timestamped.
- JSON and MARC data are transformed into tables to simplify cross-domain SQL queries.

Overview of Metadb

Metadb is open-source software that continuously reads data from transaction processing databases or other dynamic data sources, and helps with integrating the data within its own database to support analytics. The internal databases of FOLIO and ReShare are examples of data sources that Metadb can read from. When state changes occur in FOLIO's database, Metadb updates its database correspondingly, with a few differences:

- No overwrite: deleted data are preserved and timestamped.
- JSON and MARC data are transformed into tables to simplify cross-domain SQL queries.
- With ReShare, data are combined to support cross-tenant consortial queries.

Overview of Metadb

Metadb is open-source software that continuously reads data from transaction processing databases or other dynamic data sources, and helps with integrating the data within its own database to support analytics. The internal databases of FOLIO and ReShare are examples of data sources that Metadb can read from. When state changes occur in FOLIO's database, Metadb updates its database correspondingly, with a few differences:

- No overwrite: deleted data are preserved and timestamped.
- JSON and MARC data are transformed into tables to simplify cross-domain SQL queries.
- With ReShare, data are combined to support cross-tenant consortial queries.

This enhanced view of FOLIO data as well as the data in their original form are both accessible to Metadb users.

Overview of Metadb

Metadb is open-source software that continuously reads data from transaction processing databases or other dynamic data sources, and helps with integrating the data within its own database to support analytics. The internal databases of FOLIO and ReShare are examples of data sources that Metadb can read from. When state changes occur in FOLIO's database, Metadb updates its database correspondingly, with a few differences:

- No overwrite: deleted data are preserved and timestamped.
- JSON and MARC data are transformed into tables to simplify cross-domain SQL queries.
- With ReShare, data are combined to support cross-tenant consortial queries.

This enhanced view of FOLIO data as well as the data in their original form are both accessible to Metadb users. Metadb also allows users to import external data sets, and in general organizes its database into a shared workspace where users can partner on reporting and analytics.

Levels of interaction (examples)

Reporting

- Connect using the LDP Reporting App within FOLIO/ReShare
- Query data and run reports using web UI (SQL not required)

Levels of interaction (examples)

Reporting

- Connect using the LDP Reporting App within FOLIO/ReShare
- Query data and run reports using web UI (SQL not required)

Beginning SQL analytics

- Connect using cloud database client such as CloudBeaver
- Query data using web UI or basic SQL
- Run and query reports using basic SQL

Levels of interaction (examples)

Reporting

- Connect using the LDP Reporting App within FOLIO/ReShare
- Query data and run reports using web UI (SQL not required)

Beginning SQL analytics

- Connect using cloud database client such as CloudBeaver
- Query data using web UI or basic SQL
- Run and query reports using basic SQL

Intermediate SQL analytics

- Connect using desktop database client such as DBeaver
- Query data and run reports using SQL
- Create reports using SQL and share them with other users

A sample SQL query

Suppose we have a query that counts the number of loans in a library for each circulated item within a range of dates:

A sample SQL query

Suppose we have a query that counts the number of loans in a library for each circulated item within a range of dates:

```
SELECT item_id,
       count(*) AS loan_count
    FROM folio_circulation.loan__t
    WHERE '2023-01-01' <= loan_date AND
                          loan_date < '2024-01-01'
    GROUP BY item_id;
```

A sample SQL query

Suppose we have a query that counts the number of loans in a library for each circulated item within a range of dates:

```
SELECT item_id,
       count(*) AS loan_count
    FROM folio_circulation.loan__t
    WHERE '2023-01-01' <= loan_date AND
                          loan_date < '2024-01-01'
    GROUP BY item_id;
```

The range of dates is defined by a start and end date, in this case, `'2023-01-01'` and `'2024-01-01'`.

A sample SQL query

Suppose we have a query that counts the number of loans in a library for each circulated item within a range of dates:

```
SELECT item_id,
       count(*) AS loan_count
    FROM folio_circulation.loan__t
    WHERE '2023-01-01' <= loan_date AND
                          loan_date < '2024-01-01'
    GROUP BY item_id;
```

The range of dates is defined by a start and end date, in this case, `'2023-01-01'` and `'2024-01-01'`.

We can make this query more general by defining the start and end dates as *parameters* in a user-defined function.

The query as a function

```
CREATE FUNCTION lisa.count_loans(
    start_date date DEFAULT '2000-01-01',
    end_date date DEFAULT '2050-01-01')
RETURNS TABLE(
    item_id uuid,
    loan_count integer) AS
$$
SELECT item_id,
       count(*) AS loan_count
    FROM folio_circulation.loan__t
    WHERE start_date <= loan_date AND
                        loan_date < end_date
    GROUP BY item_id
$$
LANGUAGE SQL;
```

Calling the function

Since the function returns a table, a good way to call it is to SELECT from it:

```
SELECT * FROM lisa.count_loans(
                    start_date => '2022-01-01',
                    end_date => '2023-01-01');
```

Note that p => a_p defines the parameter name p for argument a_p. This should not be confused with the inequality operator in x >= y which means x is greater than or equal to y.

Calling the function

Since the function returns a table, a good way to call it is to SELECT from it:

```
SELECT * FROM lisa.count_loans(
                  start_date => '2022-01-01',
                  end_date => '2023-01-01');
```

Note that p => a_p defines the parameter name p for argument a_p. This should not be confused with the inequality operator in $x >= y$ which means x is greater than or equal to y.

Function parameters that have default values can be omitted. For example

```
SELECT * FROM lisa.count_loans(
                  start_date => '2023-01-01');
```

omits the parameter

```
end_date date DEFAULT '2050-01-01'
```

Sharing the function

Suppose that a user `lisa` has created `lisa.count_loans` and would like to share it with the users `celia` and `rosalind`, so that they also can call it.

Sharing the function

Suppose that a user `lisa` has created `lisa.count_loans` and would like to share it with the users `celia` and `rosalind`, so that they also can call it.

First we have to grant them the privilege to use the `lisa` schema (unless that has been done before):

```
GRANT USAGE ON SCHEMA lisa
    TO celia, rosalind;
```

Sharing the function

Suppose that a user lisa has created lisa.count_loans and would like to share it with the users celia and rosalind, so that they also can call it.

First we have to grant them the privilege to use the lisa schema (unless that has been done before):

```
GRANT USAGE ON SCHEMA lisa
    TO celia, rosalind;
```

Then grant the privilege to execute the function:

```
GRANT EXECUTE ON FUNCTION lisa.count_loans
    TO celia, rosalind;
```

Sharing the function

Suppose that a user lisa has created lisa.count_loans and would like to share it with the users celia and rosalind, so that they also can call it.

First we have to grant them the privilege to use the lisa schema (unless that has been done before):

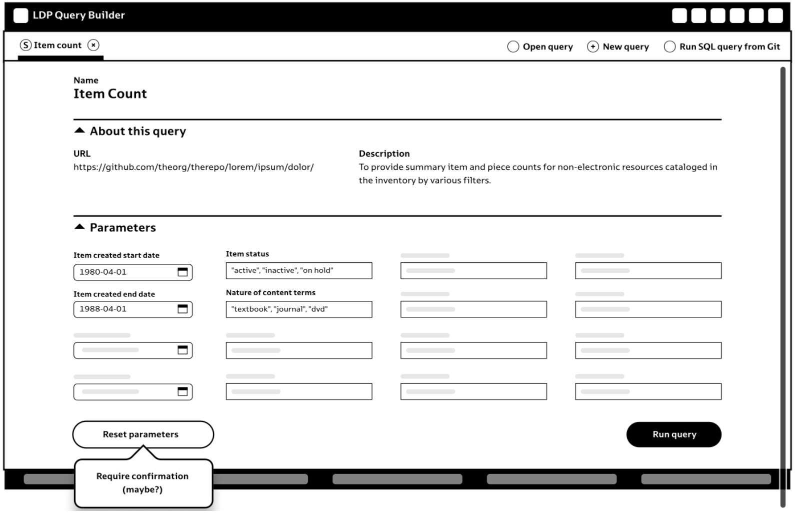
```
GRANT USAGE ON SCHEMA lisa
    TO celia, rosalind;
```

Then grant the privilege to execute the function:

```
GRANT EXECUTE ON FUNCTION lisa.count_loans
    TO celia, rosalind;
```

This method can be used with the LDP Reporting App, or a web-based database tool such as CloudBeaver, to make reports available to users that do not have a database tool installed locally.

LDP Reporting App



LDP Query Builder

Item count

Open query · New query · Run SQL query from Git

Name
Item Count

About this query

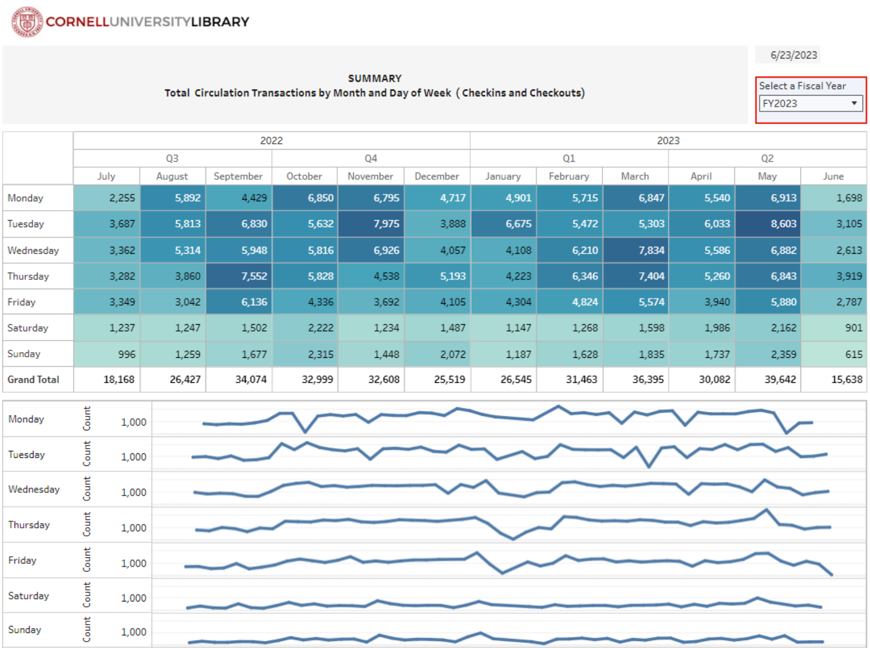
URL
https://github.com/theorg/therepo/lorem/ipsum/dolor/

Description
To provide summary item and piece counts for non-electronic resources cataloged in the inventory by various filters.

Parameters

Item created start date	1980-04-01
Item created end date	1988-04-01
Item status	"active", "inactive", "on hold"
Nature of content terms	"textbook", "journal", "dvd"

Interactive dashboards using Tableau


CORNELLUNIVERSITYLIBRARY

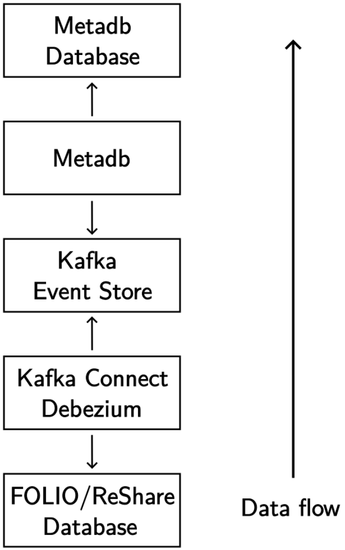
6/23/2023

Select a Fiscal Year: FY2023

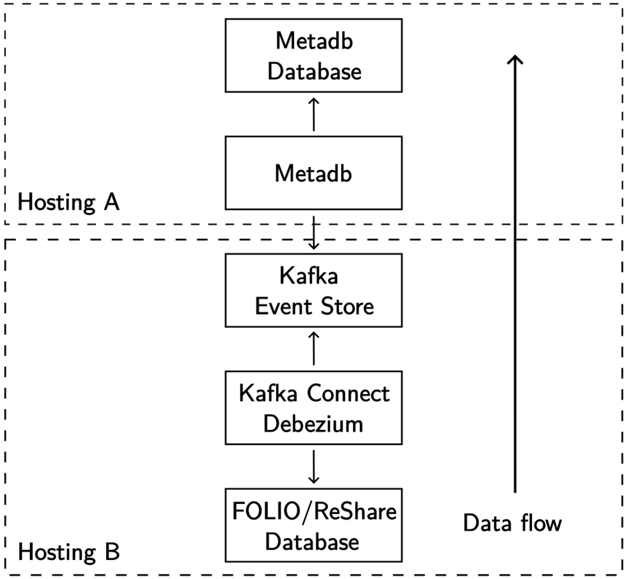
SUMMARY
Total Circulation Transactions by Month and Day of Week (Checkins and Checkouts)

	2022						2023					
	Q3			Q4			Q1			Q2		
	July	August	September	October	November	December	January	February	March	April	May	June
Monday	2,255	5,892	4,429	6,850	6,795	4,717	4,901	5,715	6,847	5,540	6,913	1,698
Tuesday	3,687	5,813	6,830	5,632	7,975	3,888	6,675	5,472	5,303	6,033	8,603	3,105
Wednesday	3,362	5,314	5,948	5,816	6,926	4,057	4,108	6,210	7,834	5,586	6,882	2,613
Thursday	3,282	3,860	7,552	5,828	4,538	5,193	4,223	6,346	7,404	5,260	6,843	3,919
Friday	3,349	3,042	6,136	4,336	3,692	4,105	4,304	4,824	5,574	3,940	5,880	2,787
Saturday	1,237	1,247	1,502	2,222	1,234	1,487	1,147	1,268	1,598	1,986	2,162	901
Sunday	996	1,259	1,677	2,315	1,448	2,072	1,187	1,628	1,835	1,737	2,359	615
Grand Total	18,168	26,427	34,074	32,999	32,608	25,519	26,545	31,463	36,395	30,082	39,642	15,638

Metadb architecture with one data source



Split hosting



Metadb roadmap

1.2 (January 2024)

- Improved performance of synchronization
- Support for multiple tenants in a shared database server

1.3 (July 2024)

- Data anonymization
- Granular user permissions
- Configuration of job scheduler

1.4 (January 2025)

- Support for multiple data sources
- Improved concurrency control & process scheduling

Migrating from LDP: data updates and data types

	LDP	**Metadb**
FOLIO/ReShare	FOLIO	FOLIO & ReShare
Source tables	Daily snapshots	Continuously
Historical data	Daily snapshots	Continuously
MARC transform	Daily	Every few hours
Derived tables	Daily	Daily

Table: Data updates

	LDP 1.x	**LDP 2.x**	**Metadb**
JSON	json	jsonb	jsonb
UUID	varchar(36)	uuid	uuid

Table: Data types

Porting a query from LDP to Metadb

Step 1: Update table names in FROM clauses to use Metadb tables.

```
SELECT id FROM user_groups;          [LDP]
SELECT id FROM folio_users.groups;   [Metadb]
```

Porting a query from LDP to Metadb

Step 1: Update table names in `FROM` clauses to use Metadb tables.

```
SELECT id FROM user_groups;          [LDP]
SELECT id FROM folio_users.groups;   [Metadb]
```

In LDP, JSON data and columns extracted from the JSON data are stored together in one table. In Metadb, the extracted columns are in a separate table ending in "__t". If a query needs data from both tables, it is simpler and more efficient to use the function `jsonb_extract_path_text()` to extract the JSON data, rather than joining the two tables together to get the extracted columns.

```
SELECT jsonb_extract_path_text(jsonb, 'desc'),
       creation_date
    FROM folio_users.groups;
```

Porting a query from LDP to Metadb

Step 2: The "data" column in LDP, which refers to JSON data, should be changed to "jsonb" (or "content" in the case of the SRS tables).

```
SELECT data FROM user_groups;             [LDP]
SELECT jsonb FROM folio_users.groups;   [Metadb]
```

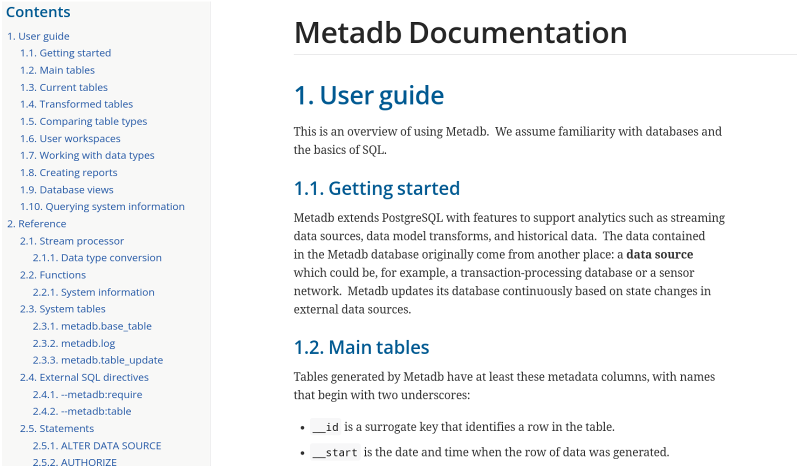
Porting a query from LDP to Metadb

Step 2: The "data" column in LDP, which refers to JSON data, should be changed to "jsonb" (or "content" in the case of the SRS tables).

```
SELECT data FROM user_groups;              [LDP]
SELECT jsonb FROM folio_users.groups;   [Metadb]
```

Step 3: Calls to the function json_extract_path_text() should be changed to jsonb_extract_path_text(), etc.

```
SELECT json_extract_path_text(data, 'group')
    FROM user_groups;                            [LDP]
SELECT jsonb_extract_path_text(jsonb, 'group')
    FROM folio_users.groups;                  [Metadb]
```



Contents

Metadb Documentation

1. User guide

This is an overview of using Metadb. We assume familiarity with databases and the basics of SQL.

1.1. Getting started

Metadb extends PostgreSQL with features to support analytics such as streaming data sources, data model transforms, and historical data. The data contained in the Metadb database originally come from another place: a **data source** which could be, for example, a transaction-processing database or a sensor network. Metadb updates its database continuously based on state changes in external data sources.

1.2. Main tables

Tables generated by Metadb have at least these metadata columns, with names that begin with two underscores:

- `__id` is a surrogate key that identifies a row in the table.
- `__start` is the date and time when the row of data was generated.

https://metadb.dev/doc