DBRepo: A Semantic Digital Repository for Relational Databases

Martin Weise
TU Wien

Moritz Staudinger
TU Wien

Cornelia Michlits
TU Wien

Eva Gergely
University of Vienna

Kirill Stytsenko
University of Vienna

Raman Ganguly
University of Vienna

Andreas Rauber
TU Wien

Abstract

Data curation is a complex, multi-faceted task. While dedicated data stewards are starting to take care of these activities in close collaboration with researchers for many types of (usually file-based) data in many institutions, this is rarely yet the case for data held in relational databases. Beyond large-scale infrastructures hosting e.g. climate or genome data, researchers usually have to create, build and maintain their database, care about security patches, and feed data into it in order to use it in their research. Data curation, if at all, usually happens after a project is finished, when data may be exported for digital preservation into file repository systems.

We present DBRepo, a semantic digital repository for relational databases in a private cloud setting designed to (1) host research data stored in relational databases right from the beginning of a research project, (2) provide separation of concerns, allowing the researchers to focus on the domain aspects of the data and their work while bringing in experts to handle classic data management tasks, (3) improve findability, accessibility and reusability by offering semantic mapping of metadata attributes, and (4) focus on reproducibility in dynamically evolving data by supporting versioning and precise identification/cite-ability for arbitrary subsets of data.

Submitted 13 June 2022 ~ *Accepted* 13 July 2022

Correspondence should be addressed to Martin Weise, TU Wien. Email: martin.weise@tuwien.ac.at

This paper was presented at International Digital Curation Conference IDCC22, online, 13-16 June, 2022

Introduction

In many institutional settings, databases are set up locally at the level of labs or individual research teams and maintained by some IT-savvy researcher who gets the privilege of having to take care of data management tasks on top of (or rather, before being able to) performing the actual research. This leads to several challenges: (1) research may be enabled late when the relational databases are set up, (2) concerns are centred around the researcher who takes care of data management tasks, (3) data stewards do not have the semantic context to ensure findability, accessibility, interoperability and reusability (Wilkinson et al., 2016) (FAIR) of the data in the database, (4) reproducibility at arbitrary points in time may be sacrificed in favour of providing data dumps and citing them.

Since researchers understand their data best, the semantic knowledge should be collected by them during the project lifetime through assistance of the repository system. We offer an open-source prototype implementation for any organization to complement their existing digital repository system for providing semantic context to data stored in databases to ensure their fitness for the organization. In this paper, we assume the fitness for the organization is present, when challenges (1) to (4) are solved.

Related Work

The way a digital repository is described in literature is manifold and ranges from understanding and preserving cultural heritage as digital heritage (Barry et al., 2017) and digital scholarship (Zhou, 2020) to archiving research and publication data in a central place as research libraries (Hienert et al., 2019).

The University of Edinburgh developed DataShare (Rice and Haywood, 2011), a digital repository for people affiliated with the university (or verified guests) to deposit data collected from various research domains, although data from biological sciences is predominant[1]. It allows researchers to deposit their research output and make it findable, accessible and reusable. Despite their claim to deposit "interactive resources", we were not able to interact with these resources in the repository apart from file downloads. Our database repository allows for all databases (currently we only support open data) to interact with the data by i.e. executing view-only queries.

The European Organization for Nuclear Research developed Zenodo[2], a digital repository that encourages researchers to provide their work along with all supplementary files. Although useful for early data-driven research results, this repository only captures a static data set (i.e., a snapshot[3]), which can easily be cited, but does not support the key features for which databases were designed in the first place, i.e. to flexibly and efficiently store dynamic data, allow its re-combination, definition of different views, selection of subsets to answer specific needs, etc. We provide these features in DBRepo through providing view-access to the underlying relational database to anyone with Internet access after a short, automated registration process.

The United States Geological Survey developed ScienceBase (Hutchison et al., 2021), a digital repository to make federally funded data publicly available therein. Researchers within this organization are allowed to deposit their data and make it available through machine-actionable interfaces, such as an *application programming interface* (API). In comparison, their repository also collects datasets with metadata such as contact information of the responsible person and provides semantic context to a physical location by allowing annotation of spatial

[1] We found 1.237 datasets assigned to biological sciences out of 3.305 in total; accessed 2022-03-10
[2] "Zenodo". URL: https://zenodo.org, accessed 2022-03-13
[3] We found 75.820 open datasets matching the formats csv, xml, zip or txt; accessed 2022-03-13

information. We extend this specific use-case in DBRepo and allow a more generalized semantic assignment to table columns by providing a reference to a semantic concept from where, e.g.. relationships can be extracted. For example, assigning <Kelvin> to a column containing temperature values (and linked to the concept of <Temperature_Values>), a relation with <Degrees_Celsius> can be extracted and the value can be adjusted by subtracting 273.15 from it.

Database Repository

Relational databases constitute an important resource in many research and industry projects.

Motivation

Their machine-actionable interfaces allow for efficient reuse of data in services. However, a standard database system lacks many of the features offered by a repository and which are essential for data being used in scientific studies that need to ensure (i) proper mapping of metadata to increase findability; (ii) support for versioning of evolving data as well as data citation to ensure reproducibility, (iii) providing licensing information to enable reuse, and many other features covered by repository systems.

Therefore, we need to devise a repository system for databases that offers these features while maintaining the key services of a database and providing a separation of concerns, i.e. allowing each stakeholder to concentrate on their core expertise, such as, the focus on infrastructure operations including resilience, back-up and security on the IT level; metadata mapping, curation activities and findability, accessibility, interoperability and reusability on the data stewardship side; licensing and GDPR-compliance aspects on the legal and ethical review board side; and, ultimately, on the content and research for the researchers. In DBRepo the databases are deployed through Docker[4] containers that are managed by the repository itself in a *hypertext transfer protocol* (HTTP) API (c.f. Figure 1) that follows the representational state transfer (Barry and Sifton, 2008) principles.

This will help to ensure that the data is FAIR after a project ends and the data is dumped into a digital repository right from the onset of a data collection or research initiative, rather than only after a project ends and the data is dumped into a digital repository. Our database repository DBRepo addresses these problems and provides a prototypic solution for digital repositories to make relational databases available in their collection. The system is available as open-source software at GitLab[5].

Increasing Findability via Semantic Mappings

In relational databases, understanding the data stored in tables in some cases require more information than just the raw tuples holding the information. DBRepo allows for a researcher to add a semantic concept and a unit of measurement via a thorough ontology (Rijgersberg et al., 2013) at a table column-level, i.e. concept "temperature" and unit of measurement "Kelvin".

[4] "Docker". URL: https://docker.com, accessed 2022-03-10

[5] "Database Repository". URL:https://gitlab.phaidra.org/fair-data-austria-db-repository/fda-services, accessed 2022-06-11

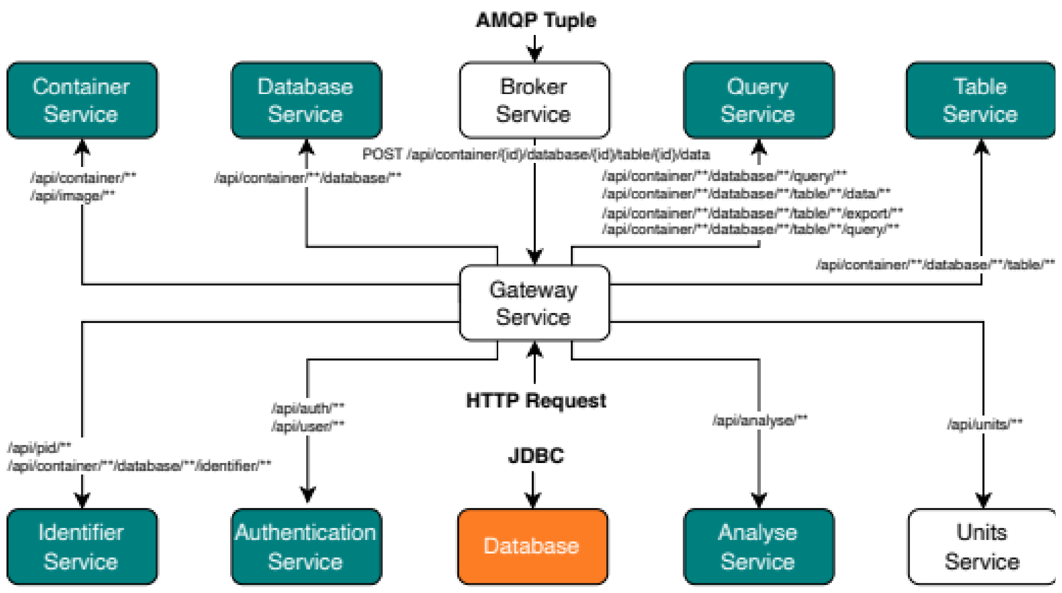
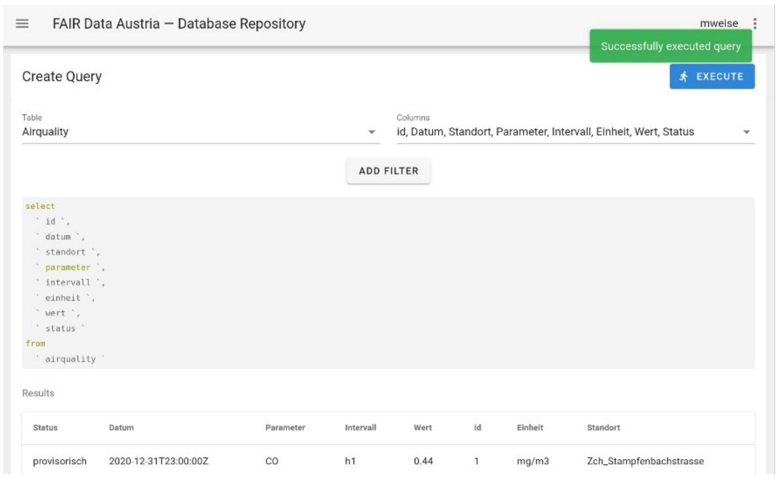

AMQP Tuple
Container Service
Database Service
Broker Service
Query Service
Table Service
/api/container/**
/api/image/**
/api/container/**/database/**
POST /api/container/{id}/database/{id}/table/{id}/data
/api/container/**/database/**/query/**
/api/container/**/database/**/table/**/data/**
/api/container/**/database/**/table/**/export/**
/api/container/**/database/**/table/**/query/**
/api/container/**/database/**/table/**
Gateway Service
HTTP Request
/api/auth/**
/api/user/**
/api/pid/**
/api/container/**/database/**/identifier/**
JDBC
/api/analyse/**
/api/units/**
Identifier Service
Authentication Service
Database
Analyse Service
Units Service

Figure 1. The gateway provides 14 endpoints from eight services as a HTTP API that is used to operate DBRepo. Green components are connected to the metadata database. Orange components are databases.

By providing this semantic information, a unit independent search across databases, i.e. to retrieve databases that contain temperature information where certain statistical properties (minimum, maximum, mean, etc.) match is possible. The FAIR Portal using Nuxt [6] (GUI), offering the query builder (through Knex.js [7]) for inexperienced users allows the assignment of semantic information to a table column (c.f. Figure 2) when using file-based upload. The same can be achieved using the HTTP API.


FAIR Data Austria — Database Repository
mweise
Successfully executed query
Create Query
EXECUTE
Table
Airquality
Columns
id, Datum, Standort, Parameter, Intervall, Einheit, Wert, Status
ADD FILTER
Results

```
select
  ` id `,
  ` datum `,
  ` standort `,
  ` parameter `,
  ` intervall `,
  ` einheit `,
  ` wert `,
  ` status `
from
  ` airquality `
```

Status	Datum	Parameter	Intervall	Wert	id	Einheit	Standort
provisorisch	2020-12-31T23:00:00Z	CO	h1	0.44	1	mg/m3	Zch_Stampfenbachstrasse

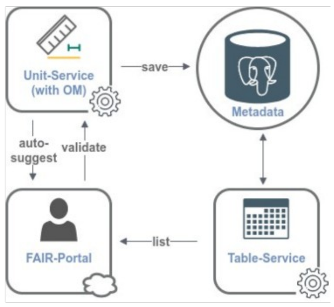


Figure 2. The FAIR Portal offers the query builder (left) and a semantic recommendation service (right) that suggests data types for researchers with little SQL and/or data engineering knowledge.

Data Versioning for Reproducible Queries

Data citation on evolving data (Rauber et al., 2015) is needed when citing data used in experiments and studies. Databases in DBRepo implement these recommendations through

[6] „Nuxt“. URL: https://nuxtjs.org/, accessed 2022-03-13
[7] “Knex.js”. URL: https://knexjs.org/, accessed 2022-03-11

adding an invisible time-period to each tuple when a modifying operation occurs, to mark the validity of each tuple in the database as a time window (valid from until valid to).

Table 1. The database repository implements system versioned tables offered by MariaDB.

ID	Sensor	Temp	Valid From	Valid To
1	A	23.1	t1	
2	B	25.8	t2	

(a) Original Table

ID	Sensor	Temp	Valid From	Valid To
1	A	23.1	t1	t3
2	B	25.8	t2	
1	A	22.1	t3	

(b) Corrected Table

We show an example of data versioning c.f. Error: Reference source not found, correcting the temperature of tuple with ID=1 at timestamp t3. This adds another tuple that is valid from t3 with the correct value and marks the original tuple as valid from t1 to t3. We illustrate these timestamps (Valid From, Valid To) in the FAIR Portal to indicate periods of time where the data was modified to give an indication for relevant timestamps for researchers that want to browse the dataset.

Repository Workflow

Researchers have different levels of expertise when it comes to database management. To support each level, the database repository assists researchers in deposit of data, interacting with the data and the persistent identification of data. Researchers can obtain an account that allows them to create databases in individual Docker containers, deposit data and create and persist queries.

Deposit of Data

Time-series data like regular sensor measurement polls, stock exchange closing values, etc. can be ingested through the HTTP API, JDBC API or AMQP API. The collected data can be inspected as current dataset or at a specific point in time using either the HTTP API or FAIR Portal. For applications that cannot use HTTP connections, i.e. continuous data feeds, the repository offers a RabbitMQ endpoint that accepts *advanced messaging queue protocol* (AMPQ) tuples. For each table that is created, the schema is stored in the metadata database and a table consumer is started that consumes the AMQP tuples in the queue. The API also allows the import of static *comma-* or *tab-separated values* datasets.

The overall approach harnesses the semantic information available for the table in the metadata database to map the AMQP tuple to a SQL query in the Query Service that inserts the data via the *Java Database Connectivity* (JDBC) API directly into the database. Further, it abstracts the internal representation of the table in the database from the data source. Researchers who want to deposit data through a GUI can add, update and delete tuples through the FAIR Portal which communicates directly with the API.

Interaction with the Data

Static datasets can be viewed in the FAIR Portal and view-only query statements can be issued to the API to interact with the dataset. DBRepo offers support to create, update and delete single tuples in the database e.g., to correct wrong sensor calibration. Any query issued is stored and can be persistently identified by adding metadata following the DataCite schema (2021). The query result is returned which can also be viewed or exported in the FAIR Portal. Whenever a sensor provides wrong measurements, the value can be corrected via the HTTP API or the help of the FAIR Portal. The API identifies a single tuple by their primary key (the repository supports primary keys that contain multiple columns).

Persistent Identification of Arbitrary Subsets

Data can be persistently identified by attaching metadata to a query statement. Internally, this operation mirrors the query information from the Query Store to the Metadata Database and makes it findable by assigning a PID (c.f. Figure 3) via the DataCite Metadata Schema (2021). The repository supports the mandatory fields such as identifier, creator, title, publisher, publication year and resource type:

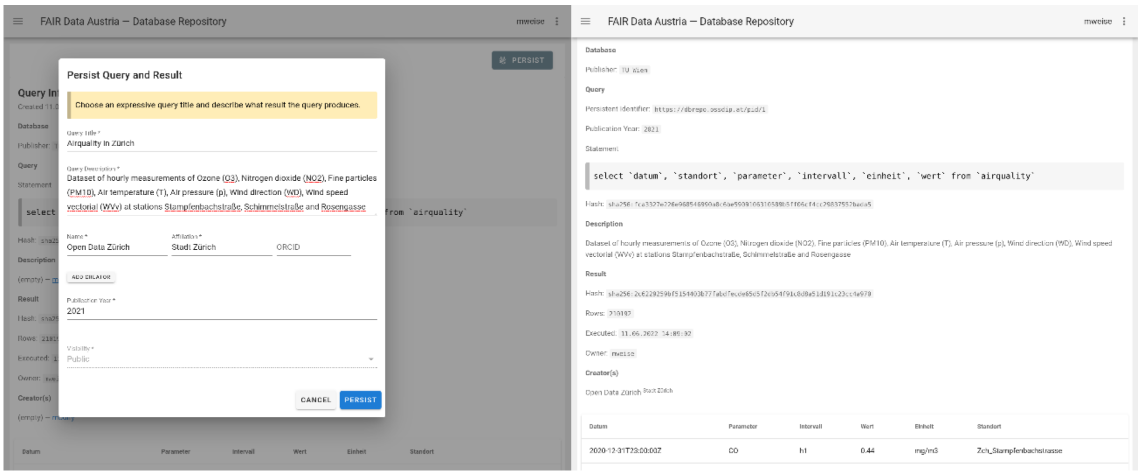
- Identifier: currently we do not mint DOIs, but a URI that points to the record (during the current pilot deployment stage the permanent availability to the databases is not foreseen. Yet, DOIs will be minted via the TU Wien DOI provider as soon as the repository is put in production mode.)

- Creator: Taking a list of creators that are involved in the subset generation, the repository supports the *creatorName* attribute and the non-mandatory *nameIdentifier* (and *nameIdentifierScheme*) to allow identification of researchers via the ORCID[8] system. The non-mandatory *affiliation* attribute is also supported.

- Title, Publisher, Publication year: Assignable via the HTTP API.

- Resource type: The repository persistently identifies a query that generates an arbitrary subset. The value is hardcoded to *Dataset*.

The FAIR Portal offers a Query Builder for simple queries that assist researchers without SQL knowledge. The researcher obtains a result that can be exported as .csv through the HTTP API or persisted. This means the researcher assigns mandatory metadata required by the DataCite Scheme to make the metadata available even when the subset is not available anymore, along with execution timestamp, query hash, result hash and result number, etc. (In this phase of development, the repository does not mint DOIs for the queries, but a persistent URI that redirects to the query page displaying the metadata instead and can be included in publications.



Persist Query and Result

Choose an expressive query title and describe what result the query produces.

Query Title *: Airquality in Zürich

Query Description *: Dataset of hourly measurements of Ozone (O3), Nitrogen dioxide (NO2), Fine particles (PM10), Air temperature (T), Air pressure (p), Wind direction (WD), Wind speed vectorial (WVv) at stations Stampfenbachstraße, Schimmelstraße and Rosengasse

Name *: Open Data Zürich | Affiliation *: Stadt Zürich | ORCID

Publication Year *: 2021

Visibility *: Public

Database

Publisher: TU Wien

Query

Persistent Identifier: https://dbrepo.ossdip.at/pid/1

Publication Year: 2021

Statement

```
select `datum`, `standort`, `parameter`, `intervall`, `einheit`, `wert` from `airquality`
```

Hash: sha256:fca3327e226e968546990a8c6be5909106310588b5ff66cf4cc29837552bada5

Description

Dataset of hourly measurements of Ozone (O3), Nitrogen dioxide (NO2), Fine particles (PM10), Air temperature (T), Air pressure (p), Wind direction (WD), Wind speed vectorial (WVv) at stations Stampfenbachstraße, Schimmelstraße and Rosengasse

Result

Hash: sha256:2c6229259bf5154403b77fabdfecde65d5f2eb54f91c8d8a51d191c23cc4a978

Rows: 210192

Executed: 11.06.2022 14:09:02

Owner: mweise

Creator(s)

Open Data Zürich Stadt Zürich

Datum	Parameter	Intervall	Wert	Einheit	Standort
2020-12-31T23:00:00Z	CO	h1	0.44	mg/m3	Zch_Stampfenbachstrasse

[8] "ORCID". URL: https://orcid.org, accessed 2022-06-11

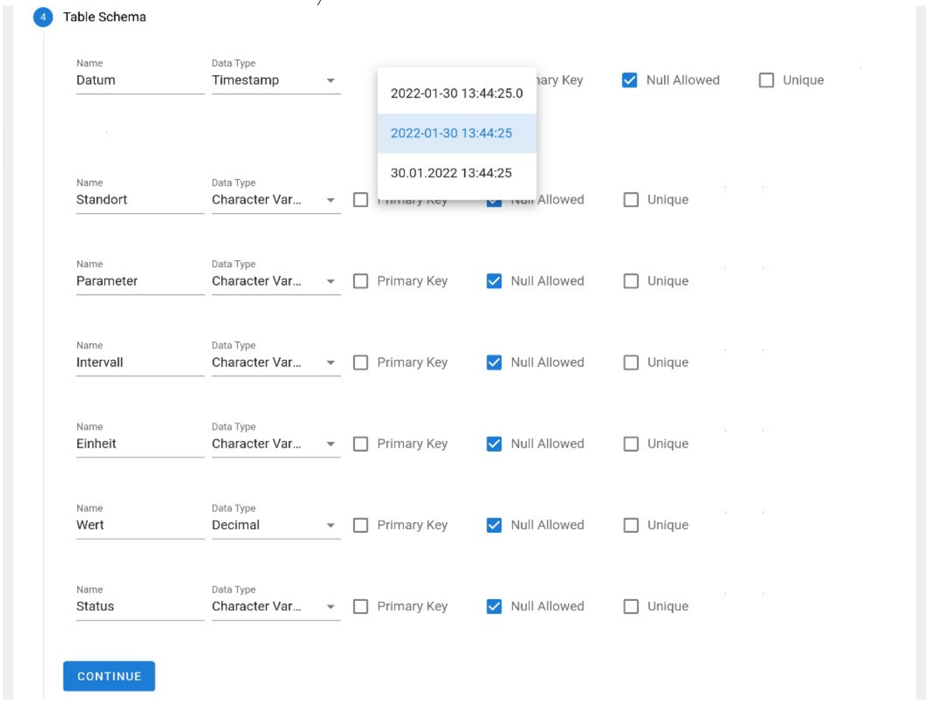
Figure 3. Persistent identification of an arbitrary subset by assigning metadata to the generating query statement (top) and landing page for a persisted query.

Import of Static Datasets

A researcher can import static datasets as .csv file into databases managed by DBRepo in two ways: (1) import a .csv file and create a table from it (2) import a .csv file into an existing table. The latter also allows replacement by the Query Service of tuples in a transaction when the primary key is already present in the table as follows:

- internally, the Query Service creates a temporary table for this import process named like the original table name but with suffix "_temporary".
- the service loads the .csv contents to this temporary table and the metadata (e.g. quotes, delimiter).
- insert all values from the temporary table to the original table and update all columns on duplicate key.

For both cases (1) and (2), the researcher needs to provide the table name, table description, separating character, optional value quote character, optional encodings for Boolean values (true/false) or the null encoding. The FAIR Portal assists the researcher to provide a data type for each column that is suggested by the Analyse Service on basis of the .csv file. Some timestamp formats still have to be manually selected if not detected automatically (c.f. Figure 4Error: Reference source not found).



4 Table Schema

Name	Data Type			
Datum	Timestamp	ary Key	✓ Null Allowed	☐ Unique
Standort	Character Var...	☐ Primary Key	✓ Null Allowed	☐ Unique
Parameter	Character Var...	☐ Primary Key	✓ Null Allowed	☐ Unique
Intervall	Character Var...	☐ Primary Key	✓ Null Allowed	☐ Unique
Einheit	Character Var...	☐ Primary Key	✓ Null Allowed	☐ Unique
Wert	Decimal	☐ Primary Key	✓ Null Allowed	☐ Unique
Status	Character Var...	☐ Primary Key	✓ Null Allowed	☐ Unique

2022-01-30 13:44:25.0
2022-01-30 13:44:25
30.01.2022 13:44:25

Figure 4. FAIR Portal assisting the researcher in defining the table schema with suggesting the column type via the Analyse Service.

To do this, we use the Python[9] library messytables[10] in the Analyse Service to determine numerical (integer, decimal), categorical (enumeration), temporal (date, timestamp) and textual

[9] "Python". URL: https://python.org, accessed 2022-06-12
[10] "Messytables". URL: https://github.com/okfn/messytables, accessed 2022-06-03

(string, text) data types. The suggestion is limited as the service is not yet capable of determining the date format, it must be selected from a drop-down list. In cases where no primary key is specified in the table schema upon creation, the Table Service automatically creates a numerical sequence starting at one and defines it as the primary key. Importing a .csv and creating a table from it takes 4 steps:

1. After login, create a new container and database through the HTTP API or FAIR Portal. If the dataset should be imported into an existing database this step is skipped.
2. Make a HTTP call to the API or use the FAIR Portal with the .csv file location for the Analyse Service to suggest data types for the columns, this step can be skipped.
3. Make a HTTP call to the API with the table specification (including the column specification) for the Table Service to create the table in the Docker container.
4. Make a HTTP call to the API with the .csv file location for the Query Service to load it into the table as described in the bullet list above.

Evaluation

In this section we evaluate to what extent the Broker Service and Query Service are able to put tuples through instantly, without having to store them in the queue, i.e. we want to find the maximum throughput where not a single tuple needs to be held back from immediate consumption in the Query Service (=stored in the database). Next to the HTTP API and JDBC API, DBRepo also offers a message queue endpoint at the Broker Service that processes AMQP[11] tuples that are data objects represented as text in *JavaScript Object Notation* (JSON).

Data Set

The Open Data Catalogue publishes non-sensitive data collected from the public administration of the city of Zürich, Switzerland and makes them available online via an open source license for anyone to use. The hourly updated air quality measurements dataset (Open Data Zürich, 2021) contains mean measurements for Stampfenbachstrasse, Schimmelstrasse, Rosengartenstrasse and Heubeeribüel stations for Ozone (O_3), nitrogen oxides (NO_x), nitrogen monoxide (NO), nitrogen dioxide (NO_2), particulate matter (PM_{10} and $PM_{2.5}$), carbon monoxide (CO) and sulfur dioxide (SO_2). The dataset contains one table with 210.192 tuples with seven columns (timestamp, station, parameter, interval, unit, value, state) and has about a 17 MegaBytes file size. We selected this dataset because it is public and contains sensor readings to also showcase the AMQP API and refer to it as *Airquality Dataset* throughout the remainder of this paper.

Experiment Setup

We deployed DBRepo on a Lenovo T14s (16 CPU threads, 32GB RAM, 1TB SSD storage) using Rocky Linux 8.6 with the latest software updates installed. Each experiment measurement instance is run in an isolated deployment such that only one user is present in the system along with one database consisting of one table with the imported Airquality Dataset.

In order to evaluate the maximum throughput of tuples per second depending on the tuple size, we wrote a Python script that is capable to produce synthetic tuple data of defined size and submits them to the Broker Service at a defined interval. The smallest possible synthetic tuple size is 0.2 kilobytes. We approximated the throughput also for tuples with 0.5, 1, 2, 3, 4 and 5 kilobytes size. For each configuration (size, throughput) the script checks if the queue remains

[11] Advanced Message Queuing Protocol

empty during publishing of AMQP tuples of fixed size and fixed throughput for a duration of 60 seconds. We iteratively increased the throughput to approximate the maximum throughput of tuples without letting the Broker Service use the queue for storage.

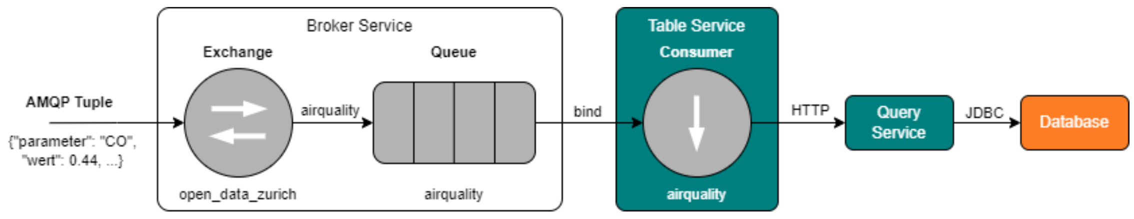


Figure 5. Ingest time-series data into a researcher database

Result

A total of seven approximations were performed to evaluate the maximum throughput of the AMQP API where tuples are still processed immediately within DBRepo. We observe that the tuple size has a strong influence on the maximum throughput, as it decreases rapidly when the tuple size increases (c.f. Figure 6).

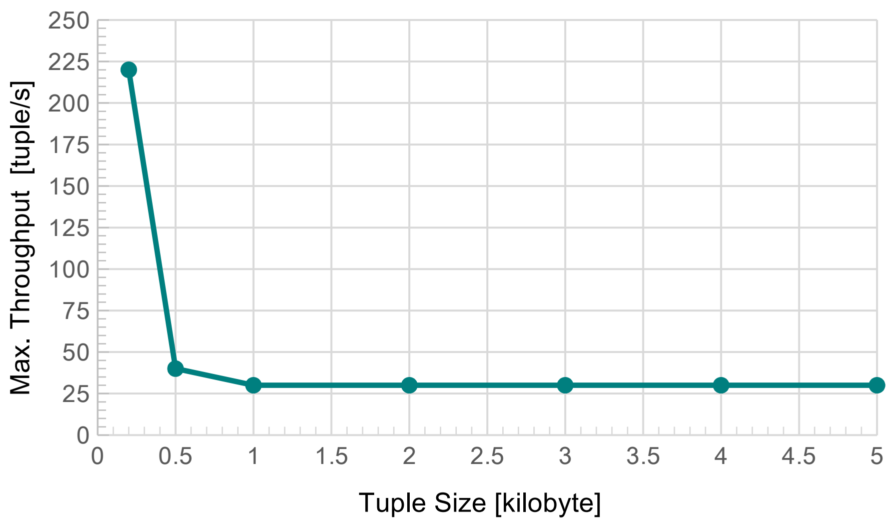


Figure 6. Maximum throughput of tuples depending on the tuple size without using RabbitMQ's queue for storage.

Conclusion & Future Work

We presented DBRepo, a semantic digital repository for relational databases in a private cloud setting as an implementation that ensures that (1) research data is stored in relational databases right from the onset of a research project, (2) separation of concerns is maintained, (3) findability, accessibility and reusability are increased by offering semantic mapping of metadata attributes, and (4) reproducibility on evolving data and data versioning is offered.

The database repository already has first cooperating institutions and external partners in Austria. We are currently working with an increasing number of initial pilot users to test the deployment of DBRepo for a range of different domains.

In the future, we want to ensure that instances of our database repository, locally deployed at an organization, can be found in the registry of repositories by adding metadata to the repository i.e., via the re3data schema (Strecker et al., 2021).

Acknowledgements

Part of this work is supported by the Austrian Ministry of Education, Science and Research under the program for digital and social transformation in higher education.

References

Barry, M., & Sifton, D. (2017). Towards a cross-Canadian digital library platform. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries*. doi:10.1109/JCDL.2017.7991616

Battle, R., & Benson, E. (2008). Bridging the semantic web and web 2.0 with representational state transfer (REST). *Journal of Web Semantics* 6(1). doi:10.1016/j.websem.2007.11.002

DataCite Metadata Working Group (2021). *DataCite metadata schema documentation for the publication and citation of research data and other research outputs. Version 4.4*. DataCite doi:10.14454/fxws-0523

Hienert, D., Kern, D., Boland, K., Zapilko, B. & Mutschke, P. (2019). A digital library for research data and related information in the social sciences. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries* (pp. 148-157). doi:10.1109/JCDL.2019.00030

Hutchison, V. B., Norkin, T., Langseth, M. L., Ignizio., D. A., Zolly, L. S., McClees-Funinan, R., & Liford, A. (2021). Leveraging existing technology: Developing a trusted digital repository for the U.S. geological survey. *International Journal of Digital Curation* 16(1). doi:10.2218/ijdc.v16i1.741

Open Data Zürich (2021). Stündlich aktualisierte Luftqualitätsmessungen. [Data set]. Zürich, Switzerland: Open Data Zürich. Retrieved from https://data.stadt-zuerich.ch/dataset/ugz_luftschadstoffmessung_stundenwerte/resource/4466ec4a-b215-4134-8973-2f360e53c33d

Rauber, A., Asmi, A., Van Uytvanck, D., & Proell, S. (2015). *Data citation of evolving data: Recommendations of the working group on data citation (WGDC)*. Report. doi:10.15497/RDA00016

Rice, R., & Haywood, J. (2011). Research data management initiatives at University of Edinburgh. *International Journal of Digital Curation* 6(2). doi:10.2218/ijdc.v6i2.199

Rijgersberg, H., van Assem, M., & Top, J. (2013). Ontology of units of measure and related concepts. *Semantic Web* 4(1). doi:10.3233/SW-2012-0069

Strecker, D., Bertelmann, R., Cousijn, H., Elger, K., Ferguson, L. M., Fichtmüller, D., … Witt, M. (2021). *Metadata schema for the description of research data repositories: Version 3.1*. re3data. doi:10.48440/re3.010

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., … Mons, B. (2016). The FAIR guiding principles for scientific data management and stewardship. *Scientific Data* 3(160018). doi:10.1038/sdata.2016.18

Zhou, P. (2020). Towards a sustainable infrastructure for the preservation of cultural heritage and digital scholarship. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries*. doi:10.1145/3383583.3398497