

PAPER • OPEN ACCESS

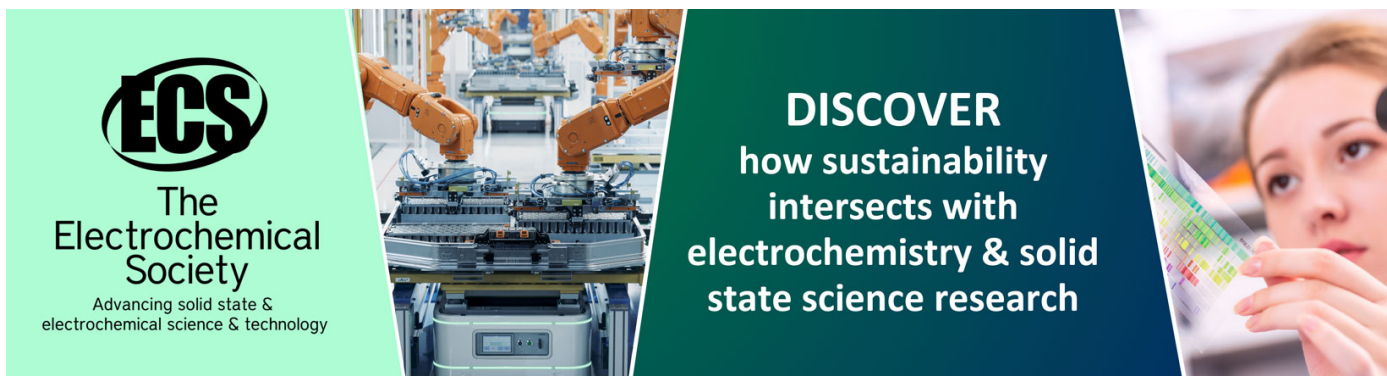
TGenBase - universal database for FAIR experiments

To cite this article: E. Lavrik 2020 *J. Phys.: Conf. Ser.* **1667** 012025

View the [article online](#) for updates and enhancements.

You may also like

- [Evolution of ROOT package management](#)
O Shadura, B Bockelman and V Vassilev
- [ROOT I/O in JavaScript](#)
Bertrand Bellenot and Sergey Linev
- [RGLite, an interface between ROOT and gLite—proof on the grid](#)
P Malzacher, A Manafov and K Schwarz



ECS
The
Electrochemical
Society
Advancing solid state &
electrochemical science & technology

DISCOVER
how sustainability
intersects with
electrochemistry & solid
state science research

TGenBase - universal database for FAIR experiments

E. Lavrik¹, for CBM Collaboration

¹ Facility for Antiproton and Ion Research in Europe GmbH, Planckstraße 1, 64291 Darmstadt, Germany

corresponding author: e.lavrik@gsi.de

Abstract. TGenBase is a ROOT-based virtual database which allows to communicate and store data in different underlying database management systems such as PostgreSQL, MySQL, SQLite, based on the configuration. It is primarily used for physics analysis parameter storage. However, it is universally applicable for any data storage task. There are several key features of the TGenBase for the user applications. It is a versioned insert only database, meaning that there is no need to update single entries and the whole history of the entries is available. The historical versions of the data can be queried with for a certain date. Being written as extension of the ROOT framework, it supports saving the ROOT objects such as graphs or histograms as well.

We provide the data description interface - a web-based application which allows the end-user to define what and in which form they want the data to be stored and define the relations between different entities. Based on this definition the database schemas, server and client side code is generated from templates and easily deployed. Another feature of this approach is that we are able to generate the full-fledged content management systems with user roles for read and write access. Data query, visualization and modification are available in C++, Python, Web and LabVIEW thin clients.

1. Introduction

The future Facility for Antiproton and Ion Research (FAIR) [1] will expand the well-established infrastructure of the GSI heavy-ion research center. It will provide unique research opportunities in the fields of nuclear, hadron, atomic and plasma physics. The Compressed Baryonic Matter (CBM) [2] experiment is one of four major scientific pillars of the FAIR. In order to build and operate its detector systems, millions of its detector components and their properties have to be described and stored centrally. Detector and experimental control data, calibration and alignment parameters, physics analysis parameters, etc., require fast and reliable storage and access interfaces.

2. TGenBase basics

TGenBase virtual database was designed to satisfy the experiment requirements mentioned above and was developed in the framework of mass characterization and quality control of the CBM silicon tracker's components. Results of such characterization had to be categorized and stored for later use during the detector assembly [3]. The term "virtual" denotes the fact that instead of managing the data it provides the unified abstracted interfaces to other underlying database management system (DBMS) which stores the data. It provides configurable role-based user permission management even for the DBMS which do not have such concept, e.g. SQLite [4]. In addition, it provides load balancing, frequent request caching and other tools to speed-up the data retrieval.



Content from this work may be used under the terms of the [Creative Commons Attribution 3.0 licence](https://creativecommons.org/licenses/by/3.0/). Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

TGenBase is based on the client-server architecture. There are several lightweight ("thin") client libraries provided for C++/ROOT [5], Python [6], JavaScript [7] and LabVIEW [8] programming languages. The clients communicate with the server by the means of the JSON message exchange over HTTPS protocol. The message exchange format is standardized and governed by RESTful API. The server implements the actual communication to the databases. There are three special classes of logic available:

- CRUD – standard create, read, update, delete logic which is used in DBMS
- Insert-only versioned logic, which prohibits the updating and deleting the records from the DB. Instead of an update, the new version of the record is appended to the database with a version identifier equal to the creation time of the record. This is very useful as it allows to query the historical versions of the records which were valid in some time interval in the past. Moreover, the full history of the record is available.
- Audited logic, which keeps track of the users creating or modifying the records in the database.

The CRUD and insert-only logics are mutually exclusive, while audited logic can be used with both.

3. Data description interface

One of the core concepts of TGenBase is the availability for non-experts. It provides a web application interface [9], where a user can describe the data they want to store in an intuitive manner.

On the front page (c.f. Figure 1) the user defines the project by giving it a name and adding the classes to it. On the class page (c.f. Figure 2) accessible by clicking "Edit" button, the class properties and relations between different classes are configured. In this notation the project maps to a database instance, class to table and class property to the table column.

Figure 1. Project data description interface.

The class properties are configured with their type, semantic sub-type, default value and ability to search the DB table by this field. Apart from built-in types such as strings, integers and doubles, the complex types such as arrays and matrices are supported. Furthermore, a property can be represented by any ROOT-compatible class. Complex and ROOT types are serialized and stored as strings in the underlying database.

Relations between classes allow to link their instances with each other. Classical "one-to-one", "one-to-many" and "many-to-many" database relations are supported. The "one-to-one" relation is useful to aggregate certain static properties of an object and offload it to a separate object or to indicate that a certain object in a collection belongs to the aggregating object. The "one-to-many" relation allows to establish a link to a collection of other objects, for example single detector can have many sub-components. The "many-to-many" relation is used to connect several collections, for example a detector can have many categories and a single category can have many different detectors.

TGenBase v1.2.0
Class Generator - Class View

☐ Expert mode

Class Name
Component

Imported ROOT classes
I TBits + -

Relations
I Has One Module Module + -

Class Properties

I	Property	Type	Searchable	Copy
I	Id	Long Integer	<input checked="" type="checkbox"/>	+ -
I	UID	String	<input checked="" type="checkbox"/>	+ -
I	CalibrationMatrix	2D Double Vector	<input type="checkbox"/>	+ -
I	ChannelMap	TBits	<input type="checkbox"/>	+ -
I	TestDate	TimeStamp	<input checked="" type="checkbox"/>	+ -
I	QaPassed	Boolean	<input checked="" type="checkbox"/>	+ -
I	Comment	String	<input checked="" type="checkbox"/>	+ -
I	CreatedAt	TimeStamp	<input checked="" type="checkbox"/>	+ -

Figure 2. Class properties definition interface.

The complete description of the classes and relations between them defines the data storage schema and ultimately it defines the contract between the client and server. Schema has a version which is incremented every time the changes to the project are introduced. The schema version is communicated between the server and client. The version mismatch results in denial of service. The schema evolution is an important aspect of any DB and in TGenBase it is implemented in the form of migration scripts which can be rolled on or rolled back.

The core concept of the data description interface that it allows to deterministically generate, based on the templates, both the server and client source code. The templating is done in the Vue.js JavaScript framework and is executed serverless on the client's machine. The generated code is put into an archive containing the minimal client which provides the basic set of utilities and functionality.

The projects can be saved to and loaded from the local disk. Additionally, the sample projects can be opened from the central repository.

4. Database Server

The TGenBase server is written in the PHP programming language and is based on the Laravel [10] framework broadly used in software industry. Laravel implements Active Record architectural pattern for object-relational mapping (ORM) and provides the abstraction to the underlying DBMS. Furthermore, Laravel provides the means to build safe and secure data queries. TGenBase takes advantage of this and exposes a subset of query building functionality to the clients which allows the end-user to take advantage of complex queries with many search criteria.

Two distributions of the server are available: development and production. The development distribution is meant to be used locally by the end-user to test the data description, develop the data import and export programs. It is preconfigured to use the file-based SQLite database, use no caching and serve data from debug server with verbose output.

The production distribution is meant to be run on the central experiment IT platform. It employs the Docker [11] containers to host essential services, orchestrated by the Laradock [12] configuration. It allows for scalability by the means of increasing the number of each service's containers.

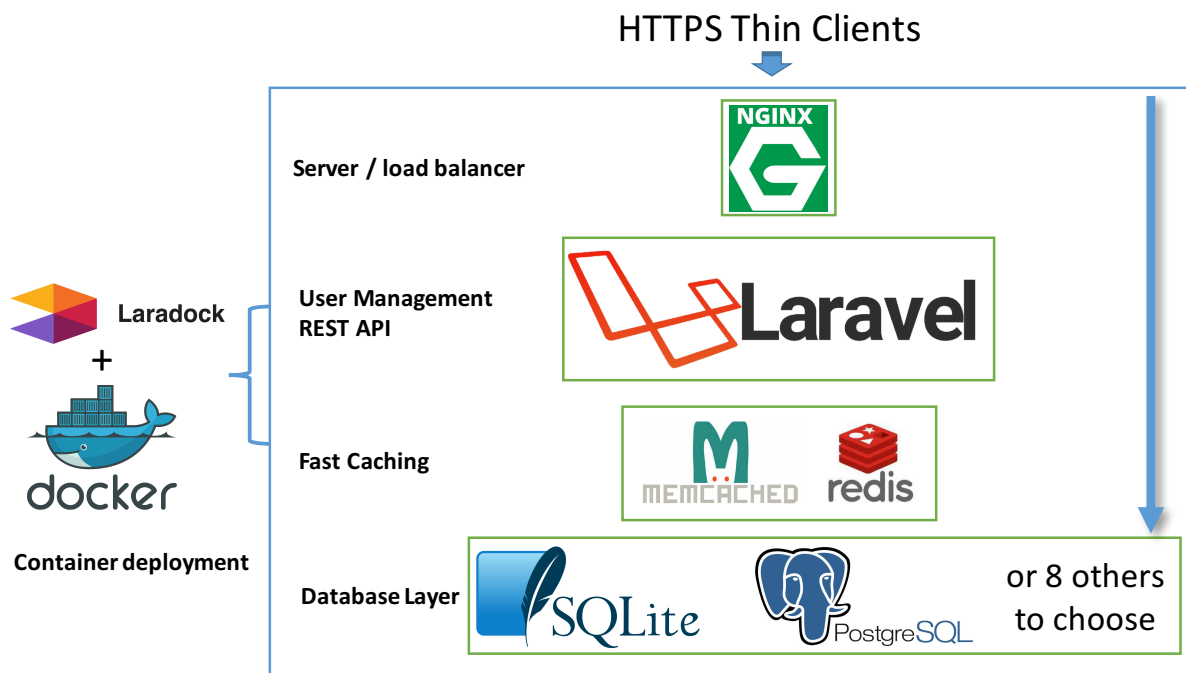


Figure 3. TGenBase server software stack with relevant technologies.

Figure 3 shows the TGenBase production server software stack and the request propagation coming from a client from top to bottom. As a web server and load balancer we use the Nginx [13]. It starts the Laravel instance, which checks the user's permissions to execute any remote method. Upon successful authentication and authorization, the fast in-memory cache (Memcached [14] or Redis [15]) is checked for prepared response. If there was no cached data, the actual database query will be evaluated at the database layer. For the production server the PostgreSQL [16] is used.

Since the setting up the software stack by no means is trivial with the amount of technologies involved, the TGenBase software distribution provides the automated scripts to install its dependencies and bootstrap the development environment. The software was developed for UNIX-like environment and tested on macOS, deb-based (Debian and Ubuntu) and rpm-based (OpenSUSE) linux flavors.

5. Thin Clients

The thin clients are generated from the project description using the templates, similar to server, and contain apart from the data classes the functionality to communicate with the server over HTTPS protocol to query and store the data. The C++/ROOT and Python clients are suited perfectly for scripting the data import, export and transfer. The C++ client can be used standalone or integrated into CBM-specific distribution of ROOT. The LabVIEW client is meant to be integrated into the software running in the laboratories for data export, analysis and visualization. The JavaScript web client provides the full-fledged content management system (CMS), generated dynamically to present the user described classes, where the data can be queried, visualized and put in by the DB operators. There is an administrator's workplace, where the users can be registered, assigned roles and permissions or be blocked. The administrator as well have an access to the server control panel, where the server configuration can be changed.

6. Summary

TGenBase is a virtual database solution which provides the means to create data storage and access applications, from design to deployment. The core of the solution is the generation of ready-to-use server and client applications from templates. The server implements custom logic sets which are

useful for general purpose data storage and storage tasks where the full history the records is maintained. The example of latter was developed for the CBM experiment's detector components. The project is open-source and can be accessed at <https://tgenbase.com> [7].

References

- [1] FAIR Baseline Technical Report, Executive Summary, GSI, Darmstadt (2006)
- [2] T. Ablyazimov et al., EUPJ A, 53(3):60, 2017
- [3] E. Lavrik, I. Panasenko and H.R. Schmidt, NIM: A, 922:3 36 – 344, 2019
- [4] SQLite database, <https://www.sqlite.org>
- [5] R. Brun and F. Rademakers, NIM: A, 389(1):81 – 86, 1997
- [6] G. van Rossum, Python tutorial, Technical Report CS-R9526, 1995.
- [7] JavaScript - Mozilla Developer Network <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
- [8] Systemdesignsoftware LabVIEW - National Instruments, <http://www.ni.com/labview>
- [9] TGenBase, <https://tgenbase.com>, <https://git.cbm.gsi.de/TGenBase>
- [10] Laravel - The PHP Framework for Web Artisans, <https://laravel.com>
- [11] Docker – container service, <https://www.docker.com>
- [12] Laradock - PHP development environment that runs on Docker, <https://laradock.io>
- [13] Nginx web server, <http://nginx.org>
- [14] Memcached - high-performance, distributed memory object caching system, <https://memcached.org>
- [15] Redis - in-memory data structure store, <https://redis.io>
- [16] PostgreSQL database, <https://www.postgresql.org>