About the Blueriq stores

This document is still under construction.

Recommended reading material

As we do not aim to repeat ourselves, it is highly recommended to read (or at least carefully scan) the following pages first:

About cases, registers, processes

To know the difference between a case and a register: Identifying business functions, cases and registers

To know why we use an aggregate for a case: Designing cases using aggregates

More on the processes within a dynamic case: Designing dynamic processes

About Reporting and BAM

Why Blueriq is not a datawarehouse: Reporting and sharing data with other systems

Business Activity Monitoring: the chapter Business activity monitoring in Reporting and sharing data with other systems

About default persistency and aggregates

Persistency Management guide

About web services

Web Services guide

About traceability

Tracing aggregates: Aggregate Traceability

Tracing processes: Process Traceability

Tracing decisions: Decision Requirements Graph or DRG in runtime

Tracing API calls: REST API

How to use the traceability engine: How to use the traceability engine

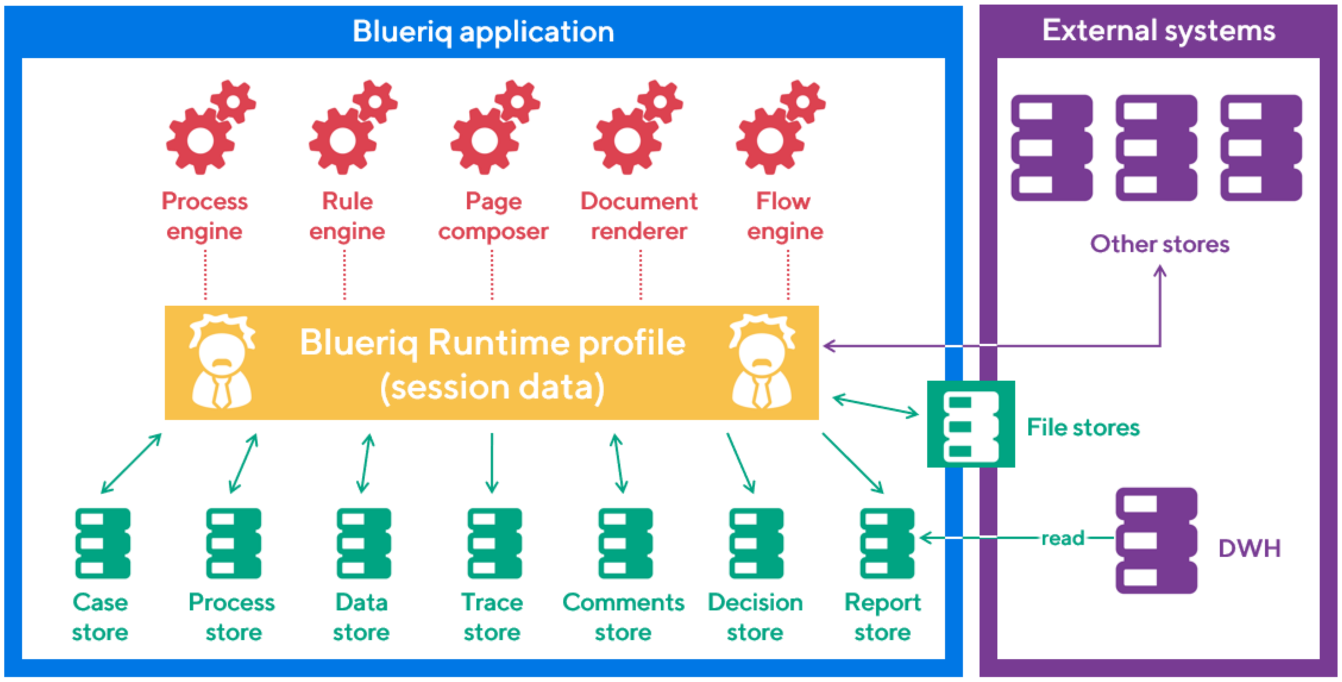
Trace database: Trace DAO plugin

Definition

Data management is a container term, used to combine many types of stores that have some sort of information or data.
To manage data properly, Blueriq has a lot of stores:

- **Case store:** Containing cases, with the goal to make them accessible for multiple users, activities etc.
- **Process store:** Containing data that is used by Blueriq to steer a process
- **Data store:** Containing data that is indigenous to the application
- **Trace stores:** The trace of everything that "happened" when running the Blueriq application
- **Comments store**: Containing comments provided by menas of a comments container are stored
- **Report store:** Containing excerpts of the profile used by e.g. a datawarehouse
- **File stores:** Containing files or documents that are uploaded into a Blueriq application
- **Other stores:** Containing data that is owned by the customer (system of records) or a third party

Overview



The profile

The most important thing to keep in mind is that a Blueriq application works with the profile. In layman's terms: Almost the only thing a Blueriq application does is add information to memory (i.e. the profile) and then let the inferencer infer new information from this: **The core of each Blueriq application is the profile.**

Data store

Containing data that is indigenous to the application

In many applications built with Blueriq, at some point we want to store data, for instance data about a customer or a product. When this data is NOT read from or stored into another system, it is regarded indigenous and can best be stored using our out of the box persistency framework using aggregates.

The main reason and benefit to do this is that aggregates are agnostic of how they are persisted and therefore the business engineer can change the domain as much as he or she likes, thus completely adhering to the agile standard of today's application development.

The business engineer can model which entities and relations are part of an aggregate and with which metadata they can be found. He or she can use services to create, read, update, delete aggregates (possibly with versioning) and search for aggregates and a smart list container that can show them. That's it.

Blueriq does everything else, such as storing only user-set data so that Blueriq's truth maintenance will derive all derivable data when restoring the profile. Also restoring the profile is done automatically when reading an aggregate. Versioning is done automatically, as well as tracing (see further on in this document).

The manner in which the aggregates are stored and read is really not important. Neither is for instance the manner in which a word-document is stored or the manner in which a SQL table is stored. However, people want to know, so here we go:

	An aggregate is stored as a closed **box**, with a tight rope around it. The analogy of the tight rope is to make sure no one ever ever thinks about opening each box when searching for the boxes containing information about customers with glasses who love golf. This is not possible! What should have been done here is making Glasses and Hobby (or whatever) part of the **metadata** of the aggregate. The metadata is the summary of what's inside the box and is stored separately and highly indexed, for fast access.
	So why not **tables** then? With optimal normal form, foreign keys, fancy backdoor-DBA-access, etc. This is a question that we hear a lot. Really, a lot. Our answer might seem a bit lame, but we're serious: Why *should* we use tables? Because everyone else does this? Because they're easy? (not) Or highly performing? (not) Or cheap? (not). The profile (as said the core of each Blueriq application) does not easily fit in tables at all, but it fits in a box perfectly! That is why we use boxes to store aggregates.

Case store

Containing cases, with the goal to make them accessible for multiple users, activities etc.

We would prefer to have a case store by default, containing all cases of certain types that were modeled in the Studio. This store could then be used to open a case when necessary, saving it after each activity etc. It could be locked in some situations, for instance when someone is writing in it. Or not locked when someone else is merely reading the case. This case store could have a lot of default behavior.
However, Blueriq does not have a case store (nor a casetype), so we use the aggregate store for this and subsequently use aggregates for case types. And this is actually a perfect match! Cases resemble the data store very much: cases - like indigenous data - belong to the Blueriq application.

What about BAM then?

Since we now know that cases are stored as aggregates, it is only logical that it is necessary to define statistics on those aggregates, since our customers need statistics about cases. Combined with statistics about processes and tasks, a full blown Business activity monitoring (BAM) dashboard can be created.

Other stores

Containing data that is owned by the customer (system of records) or a third party

So now we've tackled the data store with indigenous data (created by a Blueriq application, e.g. based on customer input) and the case store containing data about the case, preconditions, milestones etc, it's time to discuss the vast majority of data: other stores.

In short: all stores that might have information that Blueriq needs in its profile and will not be asked by means of a dialog. Think about customer relation management systems, client systems. But also all information that the Blueriq profile derives but should not be part of the Blueriq indigenous store, since the data is of importance for the company itself. Think about mortgage-data. These stores are often referred to as **system of records**. Blueriq does not provide a system of records and does not want to. We simply send data to such stores or retrieve data from them. this can be done by means of a web service.

Process store

Containing data that is used by Blueriq to steer a process

When using the Blueriq process engine, orchestration of processes is necessary at runtime. This orchestration is stored in the process store, containing a handful of tables about process instances, task instances, timers etc.

Trace stores

The trace of everything that "happened" when running the Blueriq application

Blueriq will trace a lot of information, all in different stores:

- The audit trace: Who did what when during the execution of a process, limited to task execution.
- The aggregate trace: Who created/updated/deleted (not read!) which aggregate when and what was its content at that moment.
- The decision trace: If enabled by means of a service in the model, runtime decision requirements graphs are stored in XML format.

Comments store

Containing comments provided by means of a comments container are stored

See Comment and Comment list

File store

Containing files or documents that are uploaded into a Blueriq application

See How to handle multiple files.

Report store

Containing excerpts of the profile used by e.g. a datawarehouse

Reporting concerns data in the (near) past and is in most cases done using data marts that originate in a datawarehouse. Blueriq is not a datawarehouse! We should not try to mimic one either!

In order to feed the datawarehouse with proper data, a reporting service can be used. This service is called when necessary and generates an XML file with live/actual/realtime data. The XML is stored in a database, along with some metadata. Since the amount of data that is sent to other systems might grow rapidly - and also not to unnecessarily burden the production environment - reporting data is stored separately from the production environment (the data store with indigenous data and/or the case store).

See Reporting and sharing data with other systems