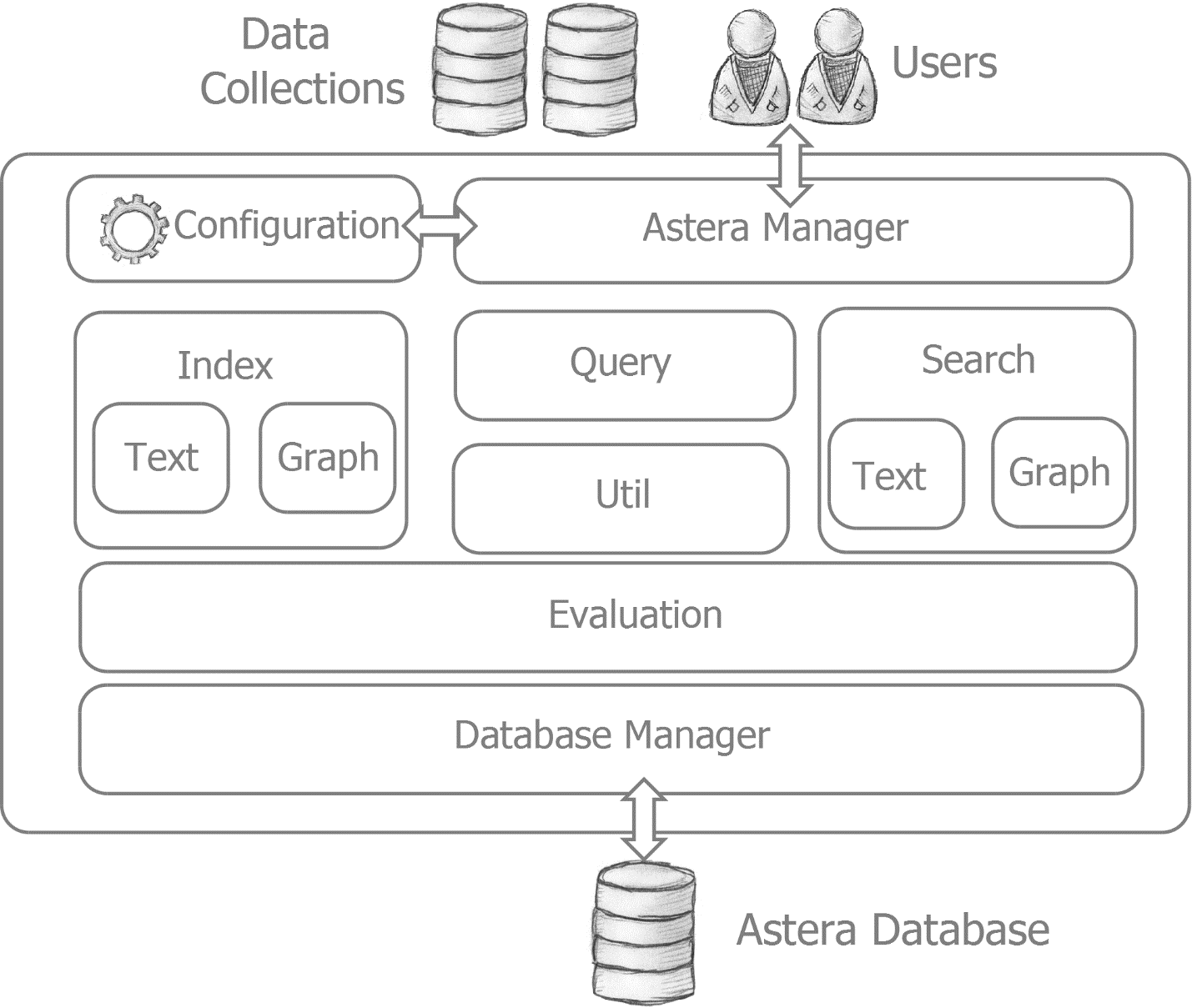
**ASTERA System Architecture**

Version 0.1

One high-level paragraph about what Astera is.



# Configuration Mechanism

Astera works with any IR collection and provides tools that help assemble, process, index and search these collections as part of an IR evaluation process. Astera however is not a library such as Lucene [[1]](#footnote-1) so researchers require an instrument to control the system without the need for programming.

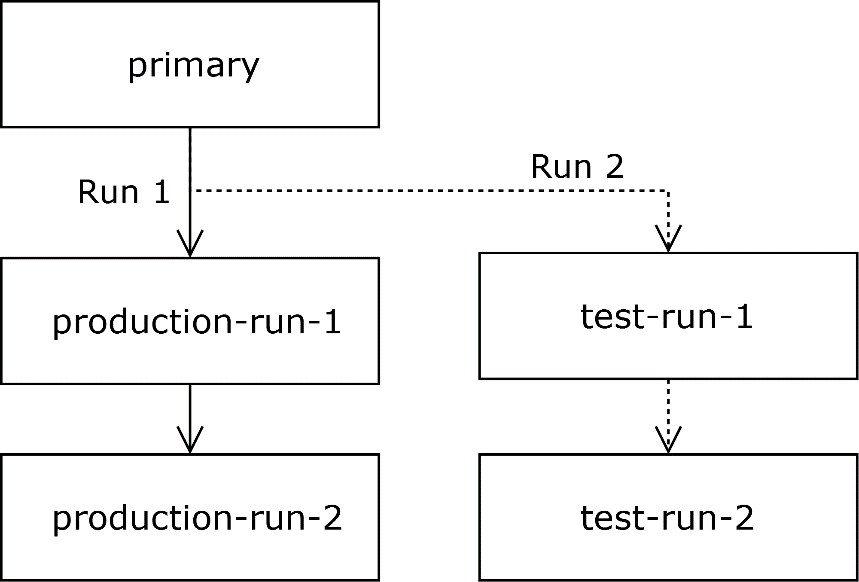
## Overview

Astera is generic in its basic architecture, however equipt with a flexible configuration mechanism to support collection-specific evaluation tasks.

Astera’s configuration is divided into two levels of configuration:

1. **Primary Configuration:** Astera requires basic settings that are needed to generally operate the system independet from what is done specifically with respect to collection(s). Such basic settings are kept in the primary configuration file called „primiary.properties“. An example for such a primiary property are the database settings to store Astera-wide evaluation results.
2. **Secondary Configuration:** Astera also manages additional configuration that is optional and specific to particular collections and particular evaluations on these collections. These secondary configurations are linked with the primary configuration file. Multiple secondary configurations can be listed and thus chained into sequences. This introduces a range of flexbilities to the system:
   1. *Specificity:* Configurations can be made specific to collections, evaluations, researchers and tasks and can define different qualites (e.g. testing versus production-level).
   2. *Partitioning:* Configuration settings can be clearly separated from each other and exchanged easily without touching the logic of the source code.
   3. *Batch Mode and Workflows:* Astera performs in batch mode on its configurations and executes a configuration as a so-called „Run“. Multiple secondary configurations can be listed which means that Astera chains them into workflows and executes them one after another.

The figure below shows an example configuration intrastructure within Astera that consists of 5 configuration files. The primary configuration provides the basic settings (e.g. information of how to connect to the Astera evaluation database). It either triggers two test runs (e.g. trying out a experimental weighting scheme on two local collections) or two production runs (e.g. two search evaluations on a collection that are compared for a publication).



These features enable that every collection can be configured with Astera and that available tools can be adapted to process these collections. Furthermore, it allows creating sequences of runs each operating with multiple collections and alternative settings to test different variables in the hypotheses space. The list of runs can be arbitrarily long thus enabling to design and run complex research agendas in batch mode. Each configuration contains all decisions that have been made in addition to the Astera code. This means, each configuration file is a descriptive research document that can be useful for later and it worth keeping.

In the following two sections, we highlight the internal structure of the two levels of configuraiton --- the primiary and the secondary configuration file.

## Structure of the Primary

The primary configuration (filename: „primary.properties“) contains two types of information:

* A set of variables that define the absoluate core of the Astera system configuration the is required to run ist basic functionality.
* A list of secondary configurations as an IR Evaluation Process for a unit of research (e.g. a paper, deliverable or even a test procedure). Each secondary configuration file is also called a „Run“.

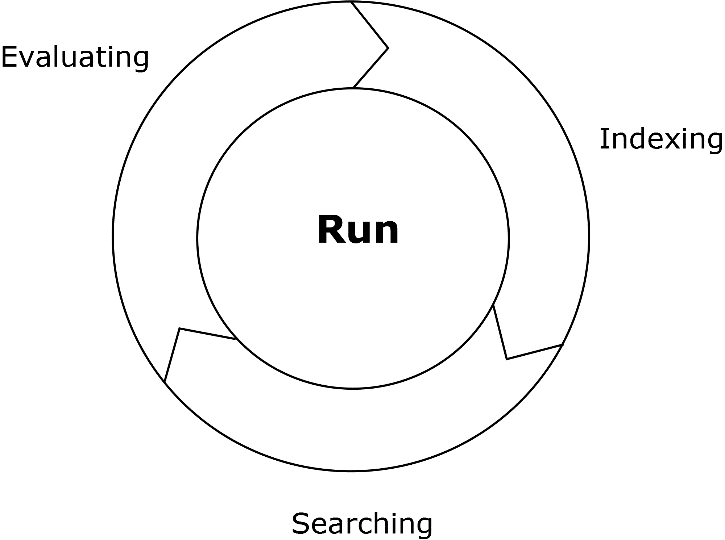
The following list shows the variables that are part of the primary with a description and its default:

|  |  |  |
| --- | --- | --- |
| Variable | Description | Default |
| run.properties | Ordered list of filenames with run configurations | secondary.properties |
| driver | JDBC Database driver for MySQL with reference to Astera database name | jdbc:mysql://localhost:3306/astera |
| user | Astera database user | root |
| Pass | Astera database password. Must be set. | N/A |
| clean.eval.db | When set true, removes everything for evaluation DB | true |

The variable „runs.properties“ can list a number of additional secondary configuration files. This list represents the IR Evaluation batch process. Each configuration file from this list is loaded in isolation from each other, in that sequence, and executed as a run. Each run loads its variables on top of the variables defined in the primary. When switching to the next run, the variables from the previous run are removed which means that each run is executed in its own environment.

## Structure of a Secondary (Run)

Each secondary configuration is a potential run. It is potential in the sense that it only becomes a run when listed in the primary configuration. A run is a single iteration through the three stages of indexing, searching and evaluating (see figure):



* **Indexing:** This configures the indexing fields and where the content for this fields resides (e.g. extracting the authors of documents with a XPath statement and putting it in a field called „author“). This configuration in this section highly depends on the collection.
* **Searching:** This part determines which queries are used for the search and how they are generated, how they are processed, how many results are extracted per query, and the scoring algorithms used.
* **Evaluating:** This section defines a set of measures and their parameterization. These depend highly on the kind of research that is created.

The cycle of all three stages represents a complete IR evaluation. However, it is also possible to disable some of its parts. The matrix below shows all sensible combinations that are supported within Astera:

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Indexing** | **Searching** | **Evaluating** |
| 1. Pure indexing: For testing or preparing an index | X |  |  |
| 1. Pure searching: For testing a search algorithm on an existing index |  | X |  |
| 1. Evaluated Search: Allows evaluating a search algorithm on an existing index |  | X | X |
| 1. Index and search: For testing a search on a newly created index | X | X |  |
| 1. Full Run: A full IR evaluation from scratch | X | X | X |

Most arrangements are used when developing or testing a search strategy with a new collection. However, the arrangements without indexing (2 and 3) are generally useful when a complex index was built previously to save system resources and personal time. The following table shows the variables that are required for a secondary configuration.

|  |  |  |
| --- | --- | --- |
| **Variable** | **Description** | **Default** |
| class | Java class for this run | N/A |
| do.index | true, if indexes is built, false otherwise | true |
| do.search | true, if search is performed, false otherwise | true |
| do.evaluate | true, if results are evaluated, false otherwise | true |
| TODO: more? |  |  |

The *class* property points to the full path of a Java class that is used for this run. It allows connecting the execution with Java code that understands additional properties on top of the basic ones listed above. The three properties *do.index*, *do.search* and *do.evaluate* can be used to control the evaluation cycle.

1. <http://lucene.apache.org/core/> [↑](#footnote-ref-1)