CKANAlyze

v0.1.x

Alberto Zanella <a.zanella@trentorise.eu>
Juan Pane <pane@disi.unitn.it>

October 10, 2013

© 2013 TrentoRISE – All Rights Reserved

Table of Content

1. Introduction

CKANAlyze is a JAVA application which provides statistical information about specified CKAN Catalogs. For each CKAN catalog, required to be analyzed, **CKANAlyze** performs statistical analysis on catalog resources and provides aggregated information on the whole catalog.

CKANAlyze is composed of two main parts: **ckanalyze-engine,** which performs analysis and stores results into the database and **ckanalyze-web,** which exposes the analysis over a Web Service API.A Java client is also provided for easy access to the web services (**ckanalyze-client**) .

CKANAlyze is used by the OpenDataTrentino project (https://github.com/opendatatrentino/) . Statistics provided by **CKANAlyze are** integrated into the OpenDataRise platform (https://github.com/opendatatrentino/OpenDataRise)

CKANAlyze adopts maven as its building solution, dependency manager and deployment tool.

CKANAlyze is composed of five Maven projects:

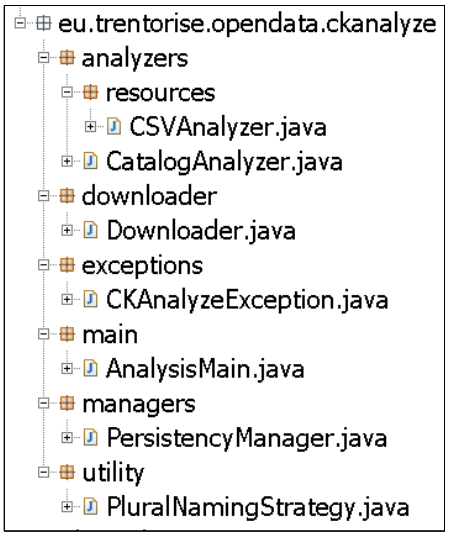
ckanalyze-engine	: the engine standalone application
ckanalyze-jpa	: contains Entity classes for persistency management
ckanalyze-web	: a java web application containing the Web API
ckanalyze-model	: contains JAXB tagged classes for XML / JSON responses
ckanalyze-client	: a java library provided as an easy client for ckanalyze-web services

In next sections we briefly describe each module of **CKANAlyze**. We will present each software in a different section. An overview of the project structure is provided first. In second paragraph we will describe behavior of software. Next the intended usage of the software is presented. Finally we will report test cases and functional dependencies where present.

2. CKANAlyze Engine

Name of the Maven Project	ckanalyze-engine
Binary Distribution	YES, Executable JAR
Published in Maven Central	NO
GitHub URL	https://github.com/opendatatrentino/CKANalyze/tree/master/ckanalyze-client
License	**AGPLv3 - http://www.gnu.org/licenses/agpl-3.0.html** since it uses TrCkanClient which is distributed under AGPL

2.1 Project structure



```
eu.trentorise.opendata.ckanalyze
  analyzers
    resources
      CSVAnalyzer.java
    CatalogAnalyzer.java
  downloader
    Downloader.java
  exceptions
    CKAnalyzeException.java
  main
    AnalysisMain.java
  managers
    PersistencyManager.java
  utility
    PluralNamingStrategy.java
```

The project is organized in six packages, as can be seen in Figure 1:

analyzers : contains classes for the analysis of catalog (CatalogAnalyzer.java) and a sub-package, **analyzers.resources** which provide analysis services for resource formats that are supported (in the current version only CSV analysis).

downloader : contains class to retrieve resource files

managers : contains PersistencyManager which offers easy access methods to persistency functions.

utility : contains classes not strictly related to the business-logic.

main : contains main class to run the application.

FIGURE 1 CKANALYZE-ENGINE : PROJECT ORGANIZATION

There are three configuration files, one of them is required to be present, the others are optional.

- **hibernate.cfg.xml :** contains persistency configuration (used DBMS, database URL, username, password, size of connection pool…), an example is provided with the name hibernate.cfg.template.xml. For more information about available options, please refers to the hibernate reference (see http://docs.jboss.org/hibernate/core/3.3/reference/en-US/html/session-configuration.html). Please note that entity mapping is already configured in the application code.
- **log4j.properties :** (Optional) contains logging configuration. This is the standard log4j configuration file. An example is available with the name log4j.properties.template
- **ckanalyze.properties** : (Optional) Contains the temp-dir (temporary download directory) configuration. You can pass the parameter also as argument when you start the JAR. An example is provided with the name ckanalyze.properties.template

2.2 Provided functions

ckanalyze-engine reads information from **configuration** table of the database, where a list of CKAN catalogs is stored. For each catalog, **ckanalyze-engine** operates as follow.

- **Lock reading operations on the updating catalog :** preventing web service users to obtain inconsistency stats but leave users to get information about other catalogs;

For each compatible resource (CSV for now), which is available in the catalog, the following steps are performed (*AnalysisMain.resourceAnalysis*):

- **Download and store useful metadata :** The program stores the CKAN Id, File name, Size of file, File format and direct download url of resource. Then the file is downloaded to the temporary directory (*Downloader*) specified in configuration file or by command line parameter.
- **Resource analysis :** For each CSV resource, following statistics are computed (*CSVAnalyzer*) and stored in DB (*AnalysisMain.resourceAnalysis*):
  - **rowCount :** Number of rows in the file (except heading row);
  - **columnCount :** Number of columns in file;

    - **colsDataType : ckanalyze-engine** performs column type identification. INT, STRING, FLOAT, DATE, GEOJSON,EMPTY are supported right now. colsDataType is a list of tuples in which, for each datatype which has been identified in the file, the number of columns of that type is associated;
    - **stringDistribution :** Complete string length distribution of all field in columns identified to have type STRING;
    - **stringAvg :** average string length of all string field in the file.

  After analysis, each resource file is automatically deleted from the temp directory.

- **Catalog metadata and analysis :** For each catalog some metadata are stored: CKAN url and total number of datasets. Other data are computed after resources analysis:
    - **totalResourcesCount :** number of analyzed resources of the catalog;
    - **totalFileSizeCount :** sum of resource file size that were analyzed.

  Aggregated statistics are also computed for each catalog (*CatalogAnalyzer*) and stored in DB:
    - **avgColumnCount :** average number of columns over all analyzed resources ;
    - **avgRowCount :** average number of rows over all analyzed resources ;
    - **avgStringLength :** average string length over all strings fields of all analyzed resources;
    - **stringDistribution :** string length distribution of all strings fields of all analyzed resources.

- **Updating Configuration information :** at the end of the process lastUpdate field is updated to the current date/time and the reading lock for the catalog is removed (isUpdating = false).

2.3 Intended usage
ckanalyze-engine is intended to be a standalone application running in background (a periodic cron process). After the initial configuration it does not requires user interaction.

2.4 Tests
A complete test set is provided in order to test correctness of resource analysis on samples CSV files (eu.trentorise.opendata.ckanalyze.analyzers.resources. TestCSVAnalyzer). Catalog analysis is tested on a limited number of resources from CKAN catalog *dati.trentino.it* (eu.trentorise.opendata.ckanalyze.analyzers. TestCatalogAnalyzer).

2.5 Functional dependencies
Major dependencies of the current version of **ckanalyze-engine** are:

- **TrCkanClient (AGPLv3)**: CKAN client to retrieve information from the catalog (for additional information see https://github.com/opendatatrentino/TrCkanClient);
- **nlprise (LGPLv2.1):** for date field identification (for additional information see https://github.com/opendatatrentino/NLPrise);
- **hibernate (LGPLv2.1):** for persistency management;
- **postgresql (BSD License): for database connection;**
- **ckanalyze-jpa (LGPLv2.1):** for JPA Entity classes.

3. CKANAlyze JPA

Name of the Maven Project	ckanalyze-jpa
Binary Distribution	YES, JAR
Published in Maven Central	YES
GitHub URL	https://github.com/opendatatrentino/CKANalyze/tree/master/ckanalyze-jpa
License	**LGPLv2.1 - http://www.gnu.org/licenses/lgpl-2.1.html**

3.1 Project structure

ckanalyze-jpa is organized in a single package (*eu.trentorise.opendata.ckanalyze.jpa*), which contains all classes annotated with JPA standard.

3.2 Provided functions

All classes are annotated with JPA annotations. A developer is able to generate the database schema from classes in an automatic way. In this first version, the PluralNamingStrategy class (which is present in ckanalyze-engine and ckanalyze-web) is required to create a working database schema. (see issue #7)

3.3 Intended usage

ckanalyze-jpa is intended to be used from two components of the CKANAlyze application: **ckanalyze-engine** and **ckanalyze-web** in order to easily manage persistence of statistics and configuration data.

3.4 Functional dependencies

Major dependency of current version of **ckanalyze-jpa** is:

- **hibernate-core (LGPLv2.1)**: for JPA tagging library

4. CKANAlyze Web (Web API)

Name of the Maven Project	ckanalyze-web
Binary Distribution	YES, Deployable WAR
Published in Maven Central	NO
GitHub URL	https://github.com/opendatatrentino/CKANalyze/tree/master/ckanalyze-web
License	**LGPLv2.1 - http://www.gnu.org/licenses/lgpl-2.1.html**

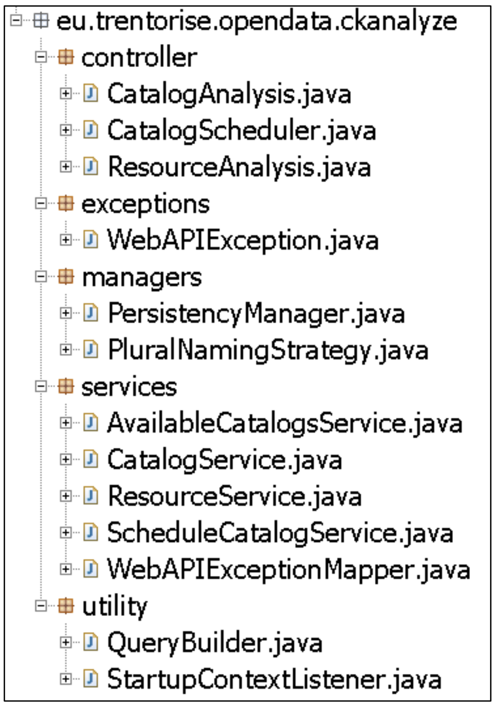
4.1 Project structure

Ckanalyze-web is a standard JAVA Web Application. It does not contains custom servlet or JSP since it uses **Jersey** to expose web services. Services are exposed via HTTP, in APPLICATION/JSON format.

Jersey automatically generates JSON output from specific model classes tagged using **JAXB** annotations and contained into the **ckanalyze-model** project.

By this way development and maintenance of client is easer and future versions of the Web API could easily expose services in other formats (like XML) without any needs to modify models.

Most of Jersey configuration is contained in the **web.xml** file. Java sources are organized in five packages, as can be seen in Figure 2:



```
eu.trentorise.opendata.ckanalyze
  controller
    CatalogAnalysis.java
    CatalogScheduler.java
    ResourceAnalysis.java
  exceptions
    WebAPIException.java
  managers
    PersistencyManager.java
    PluralNamingStrategy.java
  services
    AvailableCatalogsService.java
    CatalogService.java
    ResourceService.java
    ScheduleCatalogService.java
    WebAPIExceptionMapper.java
  utility
    QueryBuilder.java
    StartupContextListener.java
```

FIGURE 2 CKANALYZE-WEB : PROJECT ORGANIZATION

services : Contains all available web services tagged with JAX-RS annotations;

controller : Contains the business logic used by services classes and performs additional analysis which involves data manipulation. For instance it dynamically computes the datatype count of the catalog using datatype counts of resources.

utility: contains the QueryBuilder class, which is the single endpoint of HQL queries, and the StartupContextListner which initializes database at the web application startup.

managers: contains classes for persistency management

exceptions: Contains classes to map managed exception to JAXB objects. In this way exceptions are exposed like any other response.

Before the deployment of the web application, database configuration needs to be configured. This should be done editing the **hibernate.cfg.xml** file. It contains persistency configuration (used DBMS, database URL, username, password, connection pool …), an example is provided with the name hibernate.cfg.template.xml . For more information about available options, please refers to the hibernate reference (see http://docs.jboss.org/hibernate/core/3.3/reference/en-US/html/session-configuration.html). Please note that entity mapping is already configured into the application code.

4.2 Provided functions

ckanalyze-web performs three main tasks:

1. Let the user to add a catalog to be processed by the **ckanalyze-engine** (Write Configuration table on database);
2. Computes some statistics on-the-fly

3. Exposes stored and computed statistics to users, marshaling standard JAXB objects (**ckanalyze-model**) to APPLICATION/JSON.

We describe them one by one:

Let the user to manage the configuration

This task is performed by the CatalogScheduler controller class and is exposed by two different webservices:

- **AvailableCatalogService :** which takes a catalog name and returns the status of required catalog (available or not)
- **ScheduleCatalogService :** which takes a catalog name and adds it into the catalog list if it is not already present, otherwise it returns the last updating time.

Computes some statistics on-the-fly

This activity is perform directly into the QueryBuilder class. In this way aggregate operations are performed from the DBMS instead of JAVA and no more than needed data are extracted.

Exposes stored and computed statistics

Model objects are created and populated into the controller classes (CatalogAnalysis and ResourceAnalysis). Model objects are then returned to service classes (CatalogService and ResourceService). If a managed exception is thrown it will be mapped to JSONIZEDException object (a ckanalyze-model class) by the WebAPIExceptionMapper. **Jersey** automatically marshall JAXB response objects to JSON.

Error management

Basic error management is implemented. Situations like wrong parameters name, wrong number of parameters, invalid parameters are clearly reported to users. If an user adopts the **ckanalyze-models** these errors are automatically reported as a common JAVA exception (see *CkanalyzeClientRemoteException*).

4.3 Intended usage

ckanalyze-web is designed to provide statistics via REST web services. A complete web-API documentation can be found at: https://github.com/opendatatrentino/CKANalyze/wiki/Specs

ckanalyze-web is working with the **ckanalyze-client.** Integration tests were designed to ensure compatibility between these two components.

4.4 Tests

No unit tests are available for this component. Integration tests are provided (**ckanalyze-test)** which tests both **ckanalyze-web** and **ckanalyze-client.**

4.5 Functional dependencies

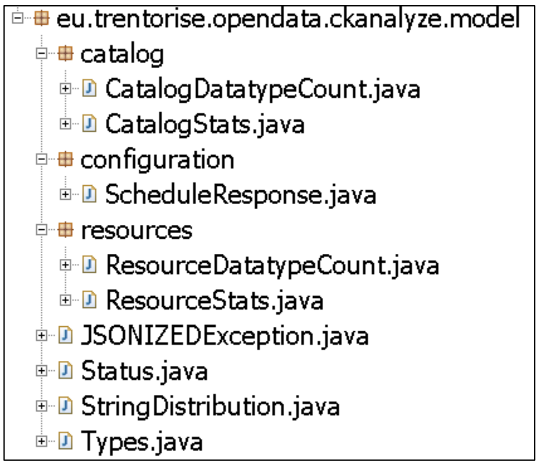
Major dependencies of the current version of **ckanalyze-web** are:

- **jersey (simil-LGPL):** A JAX-RS implementation framework;
- **jeckson (LGPLv2.1):** to expose web services in APPLICATION/JSON format;
- **hibernate-core (LGPLv2.1)**: to manage persistency;
- **ckanalyze-jpa (LGPLv2.1):** to easily access database entities via HQL;
- **ckanalyze-model (LGPLv2.1)**: which contains JAXB entity classes.

5. CKANAlyze Model

Name of the Maven Project	ckanalyze-model
Binary Distribution	YES, JAR
Published in Maven Central	YES
GitHub URL	https://github.com/opendatatrentino/CKANalyze/tree/master/ckanalyze-model
License	**LGPLv2.1 - http://www.gnu.org/licenses/lgpl-2.1.html**

5.1 Project structure



```
eu.trentorise.opendata.ckanalyze.model
  catalog
    CatalogDatatypeCount.java
    CatalogStats.java
  configuration
    ScheduleResponse.java
  resources
    ResourceDatatypeCount.java
    ResourceStats.java
  JSONIZEDException.java
  Status.java
  StringDistribution.java
  Types.java
```

FIGURE 3 CKANALYZE-JPA : PROJECT ORGANIZATION

ckanalyze-model is organized in four packages, which contains all classes annotated with JAXB.

root package : root package (*eu.trentorise.opendata.ckanalyze.model*) contains basics responses objects: JSONIZEDException (which represents a server-side managed exception), Status (which represents a simple status with a Boolean value), StringDistribution (which represents a string distribution, shared between resources and catalogs analysis). Types is an enumeration which contains all datatypes that are currently supported.

catalog and resources : catalog and resources packages contains entity specific for the catalog and resources analysis (respectively).

configuration : **contains entity to represent the ScheduleCatalogService response.**

5.2 Provided functions

All classes are annotated with JAXB annotation. Since **ckanalyze-model** is shared between **ckanalyze-web** and **ckanalyze-client** and the client returns **ckanalyze-model** class instances as results of its methods, this component is also part of the client for the end user (developer). For this reason, some classes contains extra methods which can be used from the client user for an easier access to statistical data. An example is the *public Map<Types, Long> getColsPerTypeMap()* which gives to the user an easy access to the column per type array.

5.3 Intended usage

ckanalyze-model is intended to be used by the **ckanalyze-web** web application to exposes information via Web Services and as fundamental part of the **ckanalyze-client** to give the user an easy access to web services information.

6. CKANAlyze Client

Name of the Maven Project	ckanalyze-client
Binary Distribution	YES, JAR
Published in Maven Central	YES
GitHub URL	https://github.com/opendatatrentino/CKANalyze/tree/master/ckanalyze-client
License	**LGPLv2.1 - http://www.gnu.org/licenses/lgpl-2.1.html**

6.1 Project structure

ckanalyze-client is a simple **Jersey** client designed to work with the **ckanalyze-web** web application. It is composed of a single class, named CkanalyzeClient on the root package. There is also another package, named *exceptions* which contains following classes:

- **CkanalyzeClientRemoteException :** which represents an exception thrown by the WebAPI and represented by a JSONIZEDException in model
- **CkanalyzeClientResourceNotFoundException** : a specific server-side exception which means that specified resource was not found
- **CkanalyzeClientLocalException** : an exception occurred locally on the **ckanalyze-client**

All of them inherits from RuntimeException.

6.2 Provided functions

The **ckanalyze-client** offers an easy access by following methods:

- **getCatalogStats :** which retrieves all statistics for a specified CKAN catalog.
- **getResourceStats :** which retrieves all statistics for a specified CKAN resource ID.
- **isScheduledCatalog** : which returns true if the catalog is in the Configuration list and false otherwise.
- **scheduleCatalog :** which schedule specified catalog if it is not already scheduled.

6.3 Intended usage

ckanalyze-client is intended to be used as a Java library to retrieve ckanalyze statistics in a quick and easy way.

6.4 Tests

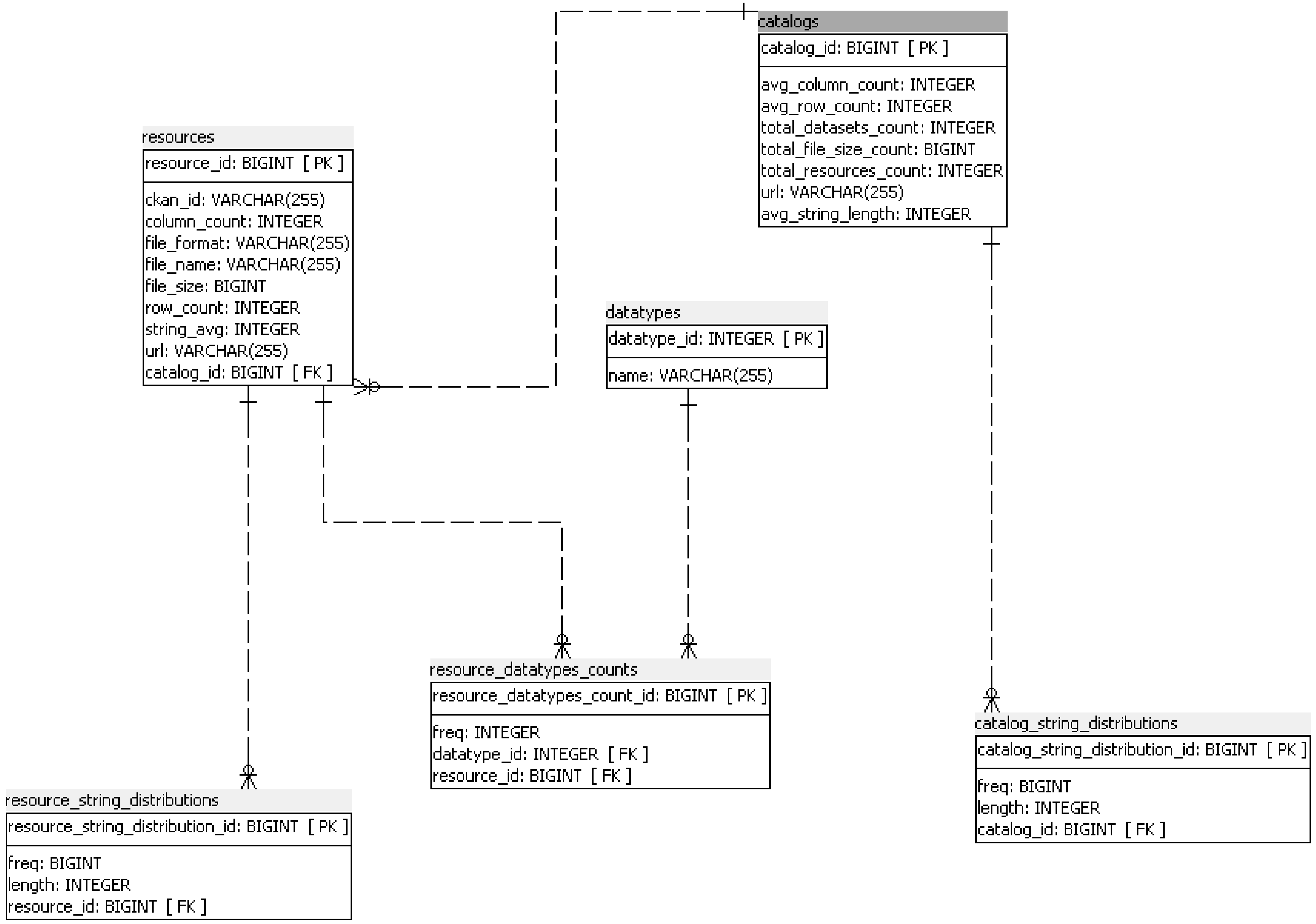
No unit tests are available for this component. Integration tests are provided (**ckanalyze-test)** which tests both **ckanalyze-web** and **ckanalyze-client**

6.5 Functional dependencies

Major dependencies of the current version of **ckanalyze-client** are:

- **jersey-client** (simil-LGPL): to easily access jersey web services
- **jeckson** (LGPLv2.1): to unmarshall JSON response to Model objects
- **ckanalyze-model (LGPLv2.1)**

7. E-R Diagram of ckanalyze database


catalogs
catalog_id: BIGINT [PK]
avg_column_count: INTEGER
avg_row_count: INTEGER
total_datasets_count: INTEGER
total_file_size_count: BIGINT
total_resources_count: INTEGER
url: VARCHAR(255)
avg_string_length: INTEGER
resources
resource_id: BIGINT [PK]
ckan_id: VARCHAR(255)
column_count: INTEGER
file_format: VARCHAR(255)
file_name: VARCHAR(255)
file_size: BIGINT
row_count: INTEGER
string_avg: INTEGER
url: VARCHAR(255)
catalog_id: BIGINT [FK]
datatypes
datatype_id: INTEGER [PK]
name: VARCHAR(255)
resource_datatypes_counts
resource_datatypes_count_id: BIGINT [PK]
freq: INTEGER
datatype_id: INTEGER [FK]
resource_id: BIGINT [FK]
catalog_string_distributions
catalog_string_distribution_id: BIGINT [PK]
freq: BIGINT
length: INTEGER
catalog_id: BIGINT [FK]
resource_string_distributions
resource_string_distribution_id: BIGINT [PK]
freq: BIGINT
length: INTEGER
resource_id: BIGINT [FK]