SQL backend for BIKA LIMS

Introduction

Currently BIKA LIMS has ZODB, which is an object oriented database. It stores python objects persistently. It has the persistent class, which all the object classes should be derived from, for persistence to happen. The ZODB storage is a directed graph of python objects which point to each other. At the root of the graph there is the python dictionary. When a search query comes searching for a particular object, search start from the root and follow the pointers of the object until it meets the desired object.

But most of the BIKA LIMS users are more familiar with relational databases like postgresQL. It is important to provide the functionality for them to use the service from BIKA LIMS in the way they are already familiar. Also it is important to have the relational mirror of current ZODB instance.

Currently there is RelStorage[6] which allows the persistence backing store to be a relational database. Yet this doesn't support to solve above problem, as it backends the entire datastore into sqldb encoded as pickles. There is ORM facility provided by SQLAlchemy and it is used by many projects. But this alone is not the solution for the problem above. Contentmirror[2] is more alike in our scenario. It mirrors the content of a PLONE site into a structured external datastore. It uses SQLAlchemy to provide the ORM.

In this project proposed by this proposal, the distinction between objects in the database, and schema fields on these objects will be maintained. The object will be remained in the ZODB. The schema field values, which are needed and ONLY them (needed fields), will be saved in the relational database, the Postgres. In this implementation collective.transmogrifier [6] pipeline can be used by "independent

processes", to transform ZODB objects to sql fields as needed. More certainly a pipeline that builds on the the jsonmigrator blueprint [7][8] can be used since it is safely used for large migrations. And also it deals with JSON and so that we will be able to use existing BIKA.LIMS JSON API for imports and exports.For efficient searchability we can consider of using an extension or same concept of collective.solr. We will **NOT** match Archetypes of Bika, to SQL, as the work is considered to be a waste of time.

The "independent process", which can be introduced also as a kind of 'watchdog', which will be polling for incoming data, will resolve the data location. Obviously, this will be a python program which will run inside a zeo-client dedicated for this purpose. This is somewhat similar to what is used in contentmirror. [2] We will be able to use a separate queue runner in order to scale, if needed.

Project Goals

After identifying and resolving the issues based on various approaches mentioned below, the respective deliverables will be delivered.

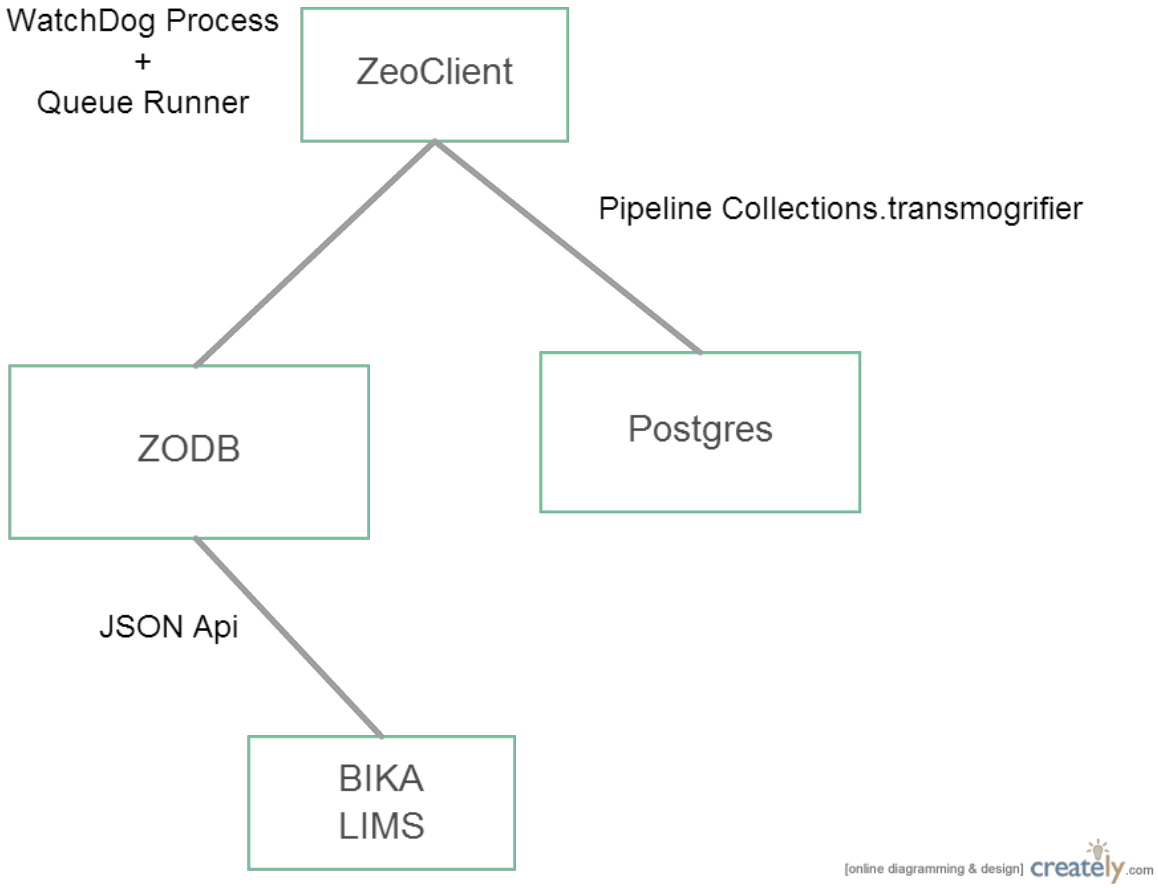
The deliverables of this project are as follows.

- Python Program which runs inside a Zeo Client which will work as a watch dog
- Python Program which runs inside the same Zeo Client which will run as a queue runner in order to scale
- Pipeline based on the collections.transmogrifier which will migrate ZODB objects to the specified schema
- PostgreSQL database instance which stores the ZODB objects as sql fields
- Program which will improve searchability and souper. (collective.solr or related implementation might be used) [11]

- he program to improve searchability and souper
- Test cases for each entity mentioned above
- Setup scripts for the system
- Documentation for the system

Implementation

The basic high level architecture is shown below.



ZeoClient

This is a separate reserved zeoclient which will work as a separate process, for the watchdog and converting processes. We can add it to the super wizard like

bin/zeoclient run -Osite_id relsqlmapper.py

This client can be connected to the collections.transmogrifier pipeline provided. We can implement that pipeline as we want and need. We can use that to map ZODB objects into any schema that is needed to be used with different applications. (Ex: Crystal Reports, Plone etc)

The program above does two main things.

- WatchDog process = Will be waiting for any update on the ZODB, will query the ZODB as client and will schedule into the queue.
- Queue runner = A queue implementation for mapping to increase the efficiency and scalability. This will force write out in the SQL form needed in to the server, depending on the settings of pipeline.
- Program which will improve the searchability of the sql db relatively. This will work with the pipeline and the relative database.

Pipeline based on collections.transmogrifier

Approach 1

This pipeline is implemented using the collections.transmogrifier to be able to map ZODB objects to various schemas as needed. There will be settings to set at the connection point of this pipeline to the required schema. To be specific we may use, pipeline that builds on the the jsonmigrator blueprint. We can get the settings for the specific instance when the set up is being done using program parameters.

Approach 2

To make the architecture and system simple we can provide the specific pipeline which will map ZODB objects to the PostgresSQL schema. This approach is more flexible than the first approach. If this approach is used, we should provide a functionality to allow specification of desired schemas for export as pipelines that run incrementally. I think this is more straightforward and extendible.

Postgres

Approach 1

If the approach one is used, as a deliverable and a profound example, this project will provide a schema for postgres. Simply a PostgreSQL database where the ZODB objects fields are stored. And that's it. If someone want he can use a pipeline to map this to whatever the schema he want from here. We can provide a generic pipeline. (This seems complex and unnecessary work for me comparing with the second approach. Will be discussed with the mentor and take a decision)

Approach 2

If the approach 2 is taken in above, then we can allow the specification of desired schemas for exports as pipelines that runs incrementally, from the Postgresql database. This pipeline is used between PostgresSQL and required schema. But this require two schemas then. Yet, this is simple and no repetition of same work. (Hope to discuss more about this issue with the and select the best or find a solution for this)

Time Line

From	To	Tasks
27/3/2015	27/4/2015	● Get more familiar with BIKA LIMS, Plone ● collective.transmogrifier ● collective.blueprint.jsonmigrator ● collective.solr ● watchdog process ● queue scheduling process ● figure out the correct approaches ● finalize the workflow and working plan after discussing with the developers ● find out how to improve the searchability and souper
28/4/2015	25/5/2015	● Start developing the basic workflow in the project ● Start developing the pipeline to match ZODB objects to given (specified) schema ● Plan the watchdog process and queue scheduling process ● Plan the schema of the PostgreSql ● Design the PostgreSql as desired ● Take feedbacks from the mentors and discuss with the mentor about the progress
25/5/2015	9/6/2015	● Finish the generic pipeline ● Instantiate the pipeline to provide the pipeline with the specifications for the PostgreSQL schema ● Test the pipeline ● Take feedbacks from the mentors and discuss with the mentor about the progress

		● Fix the bugs reported by mentors
9/6/2015	23/6/2015	● Start the implementation of zeo client and watchdog process ● Finalize configuring the zeo client ● Develop the watchdog process ● Start testing watchdog process ● Take feedbacks from the mentors and discuss with the mentor about the progress
24/6/2015	8/7/2015	● Start the implementation of queue runner ● Finalize the testing of watchdog process ● Start the program to improve the searchability and souper ● Take feedbacks from the mentors and discuss with the mentor about the progress ● Fix the bugs reported by mentors
9/7/2015	23/7/2015	● Start developing the setup scripts ● Finalize the implementation of queue runner ● Finalize the program to improve searchability and souper ● Start testing queue runner ● Start testing the program to improve searchability and souper ● Take feedbacks from the mentors and discuss with the mentor about the progress
24/7/2015	7/8/2015	● Finalize testing queue runner ● Finalize testing of the program to improve searchability and souper ● Integrates all the parts together ● Take feedbacks from the mentors and discuss with the mentor about the progress

		● Fix the bugs reported by mentors
8/8/2015	15/8/2015	● Start Documentation ● Test the full system together ● Take feedbacks from the mentors and discuss with the mentor about the progress ● Fix the bugs reported by mentors
16/6/2015	24/8/2015	● Finalize the documentation ● Finalize test cases ● Submit the final evaluation materials

About Me

I am Hareendra Chamara Philips from Sri Lanka. I am 25 years old. At the moment I am in the final semester of four year degree program at University Of Moratuwa Sri Lanka. I have specialized in Computer Science Engineering and currently holding a GPA of 3.64.

I have experience in Java, C#, Python, C, Android for more than four years. My experience include six months full time internship and 6 months part time associate software engineer at a Canadian based Software Solution Company Creo 360. I have vast experience with SQL, API, WebService, Mobile Application Development and Web Application Development, collected since my school time. I have worked with open source projects like Titanium

I have specialized my last semester in Database and Management Systems and have experience in different DBMS like mysql, postgresql and mssql.

I am a member of the team who has found and implemented the new research idea of 'Autotuning Hotspot JVM', using the machine learning techniques and open source library Opentuner. My team has won the Gold medal at CGO student research competition organized by ACM in February 2015. (http://cgo.org/cgo2015/student-research-competition/) We are also selected to present the research paper at IWAPT conference which will be held at India on this May.

I am fully available for the project as I am going to finish my degree in this May first week. I could work 40 hours per week. Despite of the busy schedule due to final examinations I have being able to manage my time to bw at the IRC channel getting know about the project and got used to PLONE and Bika Lims.

References

[1]http://en.wikipedia.org/wiki/Zope_Object_Database#ZEO

[2]https://github.com/lck/collective.contentmirror

[3]https://github.com/hexsprite/plock

[4]https://pypi.python.org/pypi/collective.lead

[5]https://pypi.python.org/pypi/collective.transmogrifier

[6]https://pypi.python.org/pypi/RelStorage

[7]https://pypi.python.org/pypi/collective.blueprint.jsonmigrator/0.1.1

[8]https://github.com/collective/collective.blueprint.jsonmigrator/blob/master/export_scripts/plone2.0_export.py

[9]https://pypi.python.org/pypi/collective.solr/4.1.0

[10] “ZODB documentation and articles” , https://media.readthedocs.org/pdf/zodborg/latest/zodborg.pdf

[11]https://pypi.python.org/pypi/souper