How to install and configure SeSQL ?

Contents

1 Introduction

This small tutorial should guide you to install SeSQL on an existing Django project.

2 First step : install prerequisite

If you have a running Django instance, you should have most of the prerequisite for SeSQL. You only have to do two things :

1. Ensure that you're using PostgreSQL 8.3 or above (8.4 preferred). If you are using another RDMS, SeSQL will not work. It should be possible to use PostgreSQL for SeSQL tables, and another engine for the rest of data, but it is not supported by this tutorial.

2. Install GenericCache from pypi or from Debian packages : http://pypi.python.org/pypi/GenericCache/1.0.2 .

3 Second step : install SeSQL

Installing SeSQL is simple : it's just a Django application. Add it alongside with your others Django applications, and then add it to the `INSTALLED_APPS` list in your `settings.py` file.

4 Third step : configure SeSQL

This is the only slightly tricky part.

4.1 Create a configuration file

SeSQL will import its configuration file with `import sesql_config` statements. So it should be somewhere in your `PYTHONPATH`, called `sesql_config.py`. The best place to put is usually alongside with the `settings.py` file.

To create it, the best is to copy the `config.py.sample` file (which is provided with SeSQL) as `sesql_config.py` on your chosen location.

If you are in Django project, you can define an alternative module from where to load the `sesql_config`, useful when you cannot alter your `PYTHONPATH`, with this setting in your project configuration:

```
SESQL_CONFIG_PATH = 'my_app.sesql_configuration'
```

4.2 Adjust the text search configuration

The sample configuration file is tuned for the french language. If you want to use another language, replace french inside `simple_french` and `ascii_french` for `TS_CONFIG_NAME` and `STOPWORDS_FILE` by the name of the language you want to use.

4.3 Provide a stopwords file

PostgreSQL (and therefore SeSQL) requires a stopwords file. Stop words are words like "the", "a", "of", ... in english, that should be ignore in searches because they are too frequent to have any value.

You've two ways to generate this stopwords file :

1. Your language is among those handled by PostgreSQL. Then, you have a file called `<language>.stop` in your PostgreSQL installation; for example on Debian it would be inside `/usr/share/postgresql/8.4/tsearch_data/`. In that case, you just need to create a plain ASCII version of the file, with a command line:

   ```
   LC_ALL=fr_FR.UTF-8 iconv -f utf-8 -t ascii//TRANSLIT \
   /usr/share/postgresql/8.4/tsearch_data/french.stop > \
   /usr/share/postgresql/8.4/tsearch_data/ascii_french.stop
   ```

2. Your language isn't supported by PostgreSQL. In that case, you'll need to write the stopwords file yourself, just put one word per line, in pure ascii.

4.4 Create the type map

The type map has two purposes : select which models to index and which to ignore, and to group models that are often searched at the same time for better performances.

If you're not sure about it, you can just index all models into one big table. That should work very well unless you have several millions of rows in your database. If you have such a high volume, you should consider having a finer type map.

The simplest type map possible would look like:

```
from django.db import models
TYPE_MAP = ((models.Model, "sesql_default"),)
```

4.5 Define your fields

Then you need to define your fields (that is, the indexes on which you'll query and/or sort SeSQL). This is done with the `FIELDS` variable.

The first two items of the sample configuration file, the `classname` and `id` ones, are required and should be kept as they are.

For the rest, adjust the fields on the example to match what you want to index. Refer to the datamodel documentation for details.

4.6 Fine-tune configuration

Now, you can adjust other parameters of the configuration file. Refer to the install documentation for more details on various options you can tweak.

5 Forth step : reindex your content

Once everything is ready, it is very simple, and can be done in two commands :

1. `./manage.py syncdb` to create all SeSQL tables ;

2. `./manage.py sesqlreindex` to reindex the content. That step can take a long time if you have lots of content, but if it gets interrupted, it'll restart where it was.

6 Fifth step : test queries

Use `./manage.py shortquery` and/or `./manage.py longquery` to perform queries, like :

```
./manage.py shortquery 'Q(my_index__containswords="a small sentence")'
```

7 Sixth step : integrate SeSQL with your code

Now, you can integrate SeSQL in your code, calling shortquery, longquery.

8 Seventh step : unleash sesql's full potential

For more information check out this documents:

- admin: list of commands provided by sesql application
- datamodel: fine tune sesql data model to perform more complex queries (fast)
- extra: dive into text highlighting, dependency tracking, search history and benchmark features
- install: complete install and configuration instructions
- query: explains how to build queries and which ones you are able to do based on you data model.