SeSQL installation guide

Contents

1 Prerequisite

- Python >= 2.5
- Django >= 1.2 (or SQLAlchemy, see sqlalchemy.txt)
- PostgreSQL >= 8.4
- GenericCache from http://pypi.python.org/pypi/GenericCache/1.0.2

SeSQL may work with later versions of those software, but was not tested with them.

2 Installation

SeSQL is a standard Django application. Just drop it in the `apps/` directory, and add it to enabled applications into `settings.py` .

3 Middleware for Django

Since version 0.18.0, SeSQL comes with a Django middleware. It's optional but advised to use it. The middleware will collect all indexation scheduled during a request, and perform them at the end of the request. If the same object is modified more than once in a single request, it'll therefore be reindexed only once.

To enable the middleware, add `'sesql.orm.django.middleware.UpdateQueueMiddleware'` to the `MIDDLEWARE_CLASSES` in the Django settings.

This feature is not (yet) available using another ORM.

4 Generating stop word files

4.1 What is this about ?

SeSQL needs to be able to filter out stop words from the indexed text, for performances reasons. Stop words are words like "a", "the", "is" in English, or "il", "le", "les" in French, which will occur in (almost) every text and does not carry any significant information.

PostgreSQL provides stop words files for common languages, but those files can contain accentuated characters, like "été" in French. Since SeSQL strips all accentuated characters from indexed text, they will not work directly.

You need to use one of the following three methods.

4.2 Automatic conversion

A script is now included with SeSQL, in the `scripts/` directory, to automatically convert all the stop words files provided by PostgreSQL.

Please note that this script has to be executed as root, since it'll have to write in PostgreSQL data directory.

Just run, as root, `scripts/generate_stop.sh`. It'll ask you for the directories to handle (by default, it'll handle all standard PostgreSQL installations) and languages to handle (by default, it'll handle all of them).

Generated files will be named `ascii_<language>.stop`.

4.3 Manual conversion

Alternatively, you can convert one (or more) stop words file(s) manually, with a command (as root) like (for the French language, and PostgreSQL 8.4) :

```
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
```

4.4 Provide a manual file

If your language isn't included, or if you want to use a custom stop words list, you can provide any text file, with a word on each line, as long as it is in the appropriate directory and doesn't contain accentuated or other kind of special characters.

5 Configuration

Before using SeSQL you **must** configure it. The configuration file must be named `sesql_config.py` in the python path (usually the project).

For a summary of how to quickly configure SeSQL, please refer to the tutorial. This document contains a detailed list of all recognized options.

5.1 ORM choice

Since version 0.10, SeSQL can work with non-django ORM. See sqlalchemy.txt for more informations about it.

5.2 Text search configuration

PostgreSQL's full text search is based on the concept of *text search configuration* (TSC). Those configuration are detailed on the PostgreSQL manual, and allow to control things like stopwords. SeSQL requires a default TSC, and can support additional TSCs to be used on specific indexes.

SeSQL recognizes the following options related to text search configuration :

TS_CONFIG_NAME Name of the primary *text search configuration* to create in the PostgreSQL database and use in full text fields where a specific TSC is not specified.

STOPWORDS_FILE Name of the stopwords file. This file **must** be where PostgreSQL will look for (`/usr/share/postgresql/8.4/tsearch_data/`) and **must** only contain plain ascii characters. See above for how to generate them.

ADDITIONAL_TS_CONFIG

This should be a list of SQL statements, to define extra TSCs that can be used in specific fields.

5.3 Clean-up/filters

CHARSET Name of the charset to use. Note that SeSQL was only tested in utf-8. SeSQL will store all data in plain ASCII, the charset will be used for preprocessing, cleanup and conversion.

ADDITIONAL_CLEANUP_FUNCTION This function (usually a lambda, but not necessarily) will be called to process text both at indexation and search time. It can be used for example to remove html tags or convert entities back to normal letters.

SKIP_CONDITION A function (or lambda) that is called on every object, is not `None`. If it returns a true value, the object will not be indexed. Useful, for example, to filter on workflow state.

5.4 Indexes and fields

FIELDS A list or tuple of fields (see `datamodel.txt`), including at least `classname` and `id`.

CROSS_INDEXES This list contains all additional indexes to create in the database. Each index is just a list of column. Indexes that are worth creating depend of the kind of queries you do frequently.

5.5 Types and tables

MASTER_TABLE_NAME The name of the master table, from which all others will inherit. This table should not contain any data, but a query done to it will query all SeSQL tables.

TYPE_MAP This list of `(class, table)` couples describes the mapping of Django classes to SeSQL tables. Django classes not present in the list will not be indexed by SeSQL. Subclasses will, by default, be sent to the same table of the superclass.

5.6 Query parameters

DEFAULT_ORDER Default sort order for queries, when sort order is not specified. Should be a tuple of index names, with an optional - to indicate reverse order.

DEFAULT_LIMIT The default number of items returned by a short query.

SMART_QUERY_INITIAL, SMART_QUERY_THRESOLD, SMART_QUERY_RATIO Control of the smart query heuristic.

QUERY_CACHE_MAX_SIZE Maximal number of long query to store in the query cache. Older queries will be discarded first. The cache is used to ensure stability of paginated results, and avoid redoing the search on very page.

QUERY_CACHE_EXPIRY Maximal time to store long queries in the cache.

5.7 Reindexing daemon

DAEMON_DEFAULT_CHUNK Number of elements to proceed on each iteration of the reindex daemon.

DAEMON_DEFAULT_DELAY Delay, in seconds, between two chunks.

DAEMON_DEFAULT_PID Pid file to use for the reindex daemon. The user running the daemon must have write permission to it, and the directory must exists.

ASYNCHRONOUS_INDEXING If set to True, all indexing will be forwarded to the daemon for asynchronous indexing. Locking will then not be used in indexation. This is recommended behavior if you have frequent changes in your database, and can tolerate a few minutes of delay between modification of the database and the results being reflected in SeSQL.

This is only supported with Django ORM for now.

DAEMON_CASCADE_RELATED If set to True (default value), then related items will be cascaded, their own related items will be reindexed, and so forth (careful, this can create loops).

If set to False, related items will only be checked for items that are explicitely indexed.

5.8 Search history and statistics

HISTORY_DEFAULT_FILTER Queries giving less than this amount of results will be ignored in history.

HISTORY_ALPHA = 0.95 Erode factor for time-based decay of recent searches score. The closer to 0, the faster old searches will see their score go down, the closer to 1 the longer they'll remain with high scores.

HISTORY_BETA Weight of the frequency at which the search was performed in the final score. This is on an arbitrary scale, and is only meaningfull compared to the HISTORY_GAMMA parameter.

HISTORY_GAMMA Weight of the number of results given by the query in the final score.

HISTORY_BLACKLIST A list of queries that will be ignore by the history feature.

5.9 Additional features

ENABLE_SESQL_ADMIN If set to yes, you'll be able to use `sesql:<fieldname>` in your admin options classes to search on SeSQL indexes from Django's admin. Please note that this feature reallies on a monkey-patch of core Django code, and is therefore disabled by default.

6 Constraints

Current version of SeSQL has a few constraints :

- it requires to have a `ClassField` called `classname` and a `IntField` called `id`, referring to the object class and id ;
- since all data is converted to plain ASCII internally, it'll not work out-of-the-box with languages written in non-latin script. Patches are welcomed to handle that.

7 Upgrading

If you need to rebuild all SeSQL indexes (because you changed them too heavily for example) you can do :

```
./manage.py createsesqltables | ./manage.py dbshell
./manage.py sesqlreindex
```