SeSQL data model guide

Contents

1 General

SeSQL data model is described in a 'sesql_config.py' file in the project.
It should define three variables :

FIELDS A list (or tuple) of fields.

MASTER_TABLE_NAME The name of the master table to use.

TYPE_MAP Association between Django types and tables.

2 Concepts

SeSQL data model is composed of *fields*. A field is something on which you can perform queries and order the data, it's similar to a Django field.

Fields are computed from *sources*. A source will fetch the values to compose a field, taking object attributes, concatenating several of them, calling methods, or following relations.

3 Fields

Each field has a type, a name, a source and may have some type dependant options.
Known types are :

IntField Normal integer field.

StrField Normal string field, up to 255 characters wide,“size“ can be specified.

ClassField Handle the class of the object.

DateField Date field, without time.

DateTimeField Date field, with time.

IntArrayField Field storing multi-valued integers.

FullTextField The beast for which SeSQL was designed, a full-text index, can be made `primary` to be used in rankings. You can specify a different text search configuration than the default on a given FullTextField with the `dictionnary` parameter.

4 Sources

Each field requires a 'source'. The source can be one of the following classes :

SimpleField Will just fetch an attribute from the Django object, can specify a condition (Q object) to filter on.

MethodCaller Will call a method from the Django object.

SubField Will walk accross one or several many-to-many mappings, fetching attributes of the related objects.

TextAggregate Concatenate the result of other sources.

WeightedAggregate Allow to ponder the result of different sources for ranking by relevance. Weights can go from A (highest) to D (lowest). Look at PosgreSQL documentation for more information on weights.

FirstOf Gets the first non-None value of a list of other sources (useful for example to track the creation date which is called `created_at` on some models and `created_on` on others).

5 Easy writing of sources

Source can also be given in a more friendly way :

- as a normal string for a `SimpleField` (ie, `workflow_state`) ;
- as a normal string terminated with `()` for a `MethodCaller` (ie, `"getFullName()"`) ;
- as a path separated by `.` for a `SubField` (ie, `".authors.firstname"`) ;

- as a list or tuple for a `TextAggregate` (ie `( "firstname", "lastname" ))`;
- a dictionary for a `WeightedAggregate` (ie `{ 'A': ("firstname", "lastname"), 'B': ("nickname",) }`).

If the source is not specified, it'll be a `SimpleField` of the same name that the index.

6 Type map

The type map is a list (or tuple) of `(class, table_name, recursive)`. All Django objects of this class will be indexed into the given table. All objects of a subclass too, unless the recursive parameter is set to False (it defaults to True).

If the same class is reachable twice (due to multiple inheritance or to specifying both a base class and a derivative in the map), the first entry that matches is taken.

You specify None as the table to explicitly ban indexing a content type even if a base class has to be indexed.

Example

```
TYPE_MAP = ((models.Photo, "sesql_photo", False),
            (models.Comment, "sesql_comment"),
            (models.BaseModel, "sesql_default"))
```

7 Dependency tracking

SeSQL provides semi-automatic dependency tracking. This works in two steps.

1. By implementing a method `get_related_objects_for_indexation` on your models, which must return a lit of (classname, id) pairs (or of Django objects). This method will be called when an object is indexed. All "related objects" will then be inserted into a special table, called `sesql_reindex_schedule`.

2. Then, asynchronously, a daemon will fetch rows from this table, and reindex objects.

8 Cleanup in text fields

Before indexing text fields, cleanup has to be performed. In SeSQL, the cleanup is a two-phase process.

The first phase is user-configurable. It should strip the text of all meta-information (HTML tags, wiki syntax, ...) and gives plain text.

The second phase is automatic, it consists in stripping all accents, converting upper-case letters to lower-case letters and replacing special characters with spaces.

To configure the first phase, you have two options :

1. Specify the `ADDITIONAL_CLEANUP_FUNCTION` in the configuration file. This should contain a function, which takes the text as parameter and returns the cleaned up text.

2. Add a `cleanup` parameter to the `FullTextField`, with a similar function.

The `ADDITIONAL_CLEANUP_FUNCTION` will be used only if the `cleanup` parameter was not specified, or set to None.

9 Additional text search configurations

If you want additional text search configurations to be created when SeSQL tables are created, you can add the SQL lines in a ADDITIONAL_TS_CONFIG variable in the config file. Refer to PosgreSQL documentation for more details.