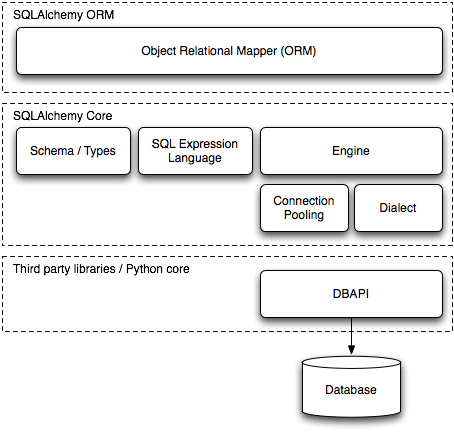
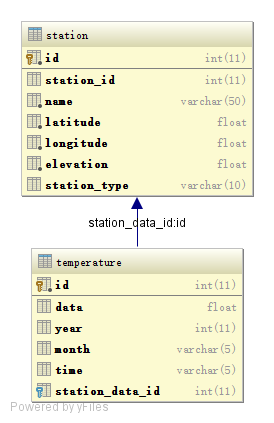
# Collect and sort out data

**Overview**

我们首先观察了British Antarctic Survey 网站上的数据形式，为了之后方便研究，我们提取出了需要数据形式并设计了数据储存模型（in model\_mixins.py，smet\_reader\_data/models.py）。最后编写爬虫将所需要的数据抓取并储存在本地数据库中。

**Data storage model**

**为了兼顾数据提取的灵活性与数据处理程序的高效性，我们使用了Object Relational Mapping技术****将程序中的数据模型类与数据库中的数据连接，最终，我们舍去了不需要的数据类型，设计出了以下数据储存模型**



Spider Architecture

我们基于Scrapy网络爬虫框架编写了一个高性能多线程异步I/O的爬虫来抓取数据。

The following diagram shows an overview of the Scrapy architecture with its components and an outline of the data flow that takes place inside the system (shown by the green arrows).

[](http://doc.scrapy.org/en/master/_images/scrapy_architecture.png)

**我们的爬虫程序使用了以下几个组件**

**Scrapy Engine**

The engine is responsible for controlling the data flow between all components of the system, and triggering events when certain actions occur. See the Data Flow section below for more details.

**Scheduler**

The Scheduler receives requests from the engine and enqueues them for feeding them later (also to the engine) when the engine requests them.

**Downloader**

The Downloader is responsible for fetching web pages and feeding them to the engine which, in turn, feeds them to the spiders.

**Spiders**

Spiders are custom classes written by Scrapy users to parse responses and extract items (aka scraped items) from them or additional URLs (requests) to follow. Each spider is able to handle a specific domain (or group of domains).

**Item Pipeline**

The Item Pipeline is responsible for processing the items once they have been extracted (or scraped) by the spiders. Typical tasks include cleansing, validation and persistence (like storing the item in a database).

**Downloader middlewares**

Downloader middlewares are specific hooks that sit between the Engine and the Downloader and process requests when they pass from the Engine to the Downloader, and responses that pass from Downloader to the Engine. They provide a convenient mechanism for extending Scrapy functionality by plugging custom code.

**Spider middlewares**

Spider middlewares are specific hooks that sit between the Engine and the Spiders and are able to process spider input (responses) and output (items and requests). They provide a convenient mechanism for extending Scrapy functionality by plugging custom code.

**Data flow**

The data flow in Scrapy is controlled by the execution engine, and goes like this:

1. The Engine opens a domain, locates the Spider that handles that domain, and asks the spider for the first URLs to crawl.
2. The Engine gets the first URLs to crawl from the Spider and schedules them in the Scheduler, as Requests.
3. The Engine asks the Scheduler for the next URLs to crawl.
4. The Scheduler returns the next URLs to crawl to the Engine and the Engine sends them to the Downloader, passing through the Downloader Middleware (request direction).
5. Once the page finishes downloading the Downloader generates a Response (with that page) and sends it to the Engine, passing through the Downloader Middleware (response direction).
6. The Engine receives the Response from the Downloader and sends it to the Spider for processing, passing through the Spider Middleware (input direction).
7. The Spider processes the Response and returns scraped items and new Requests (to follow) to the Engine.
8. The Engine sends scraped items (returned by the Spider) to the Item Pipeline and Requests (returned by spider) to the Scheduler
9. The process repeats (from step 2) until there are no more requests from the Scheduler, and the Engine closes the domain.

data statistics

经过抓取之后，共获得 113 条站点数据，230785 条温度数据（1971-2015），我们已经将数据库导出为Excel格式，位于附录中的

http://www.aosabook.org/en/sqlalchemy.html

http://doc.scrapy.org/en/master/topics/architecture.html