Как я строил проект DWH.

Появилась идея собрать данные с сайта поиска работы и узнать какие технологии сейчас наиболее востребованы на рынке. Чтобы изучать то, что точно требуется и отложить то, что никому не нужно.

Начал с того, что решил написать, что мне от нового проекта, собственно, нужно.

* Сбор данных с сайта.
* Выбор нужной информации по вакансиям.
* Сохранение в базу чтобы можно было как-то всё отсортировать и подсчитать.

По ходу пришла идея что это можно собирать постоянно. Тогда в каждый момент времени у меня будет достаточно актуальная информация по требованиям рынка вакансий. А чтобы аналитика была точнее собирать стоит не только по своему городу и своей профессии. Но и в целом по стране и по всему набору IT направлений.

В итоге получилось, что мне нужен код, который будет сам периодически запускаться, собирать большой объем данных и отправлять их в базу. Я буду брать данные из базы и сортировкой с фильтрами получать нужную аналитику.

В первом приближении план готов, погнали.

Нашел документацию по api hh. Понял, что нужен токен на который надо подавать заявку. Хотелось первые результаты получить быстрее и стал дальше читать документацию чтобы понять можно ли что-то сделать без него. К тому же в интернете видел статью, где использовали api hh без токена. Оказалось, что поиск вакансий и правда можно проводить без него. Там даже много информации по вакансии получить можно и никаких проверок при этом не проходит. Собрал парсер на requests, получил первые данные, загрузил в базу. Дело пошло, но по ограничениям api я смог собрать только 2000 вакансий с одного запроса.

Не проблема. Накидал макет словаря с параметрами запроса, включая регион, текст поискового запроса и специализацию. Составил массивы с кодами регионов, поисковыми запросами и кодами специализаций. Через циклы всё это перебрал и получил большой массив параметров для поиска. Запустил и стал смотреть.

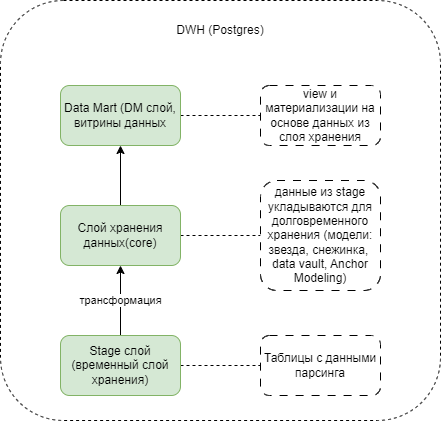
В итоге у меня получилось 4500 наборов параметров для поиска. Каждый набор обрабатывался 5-10 секунд, результаты сохранялись в память, а потом это всё должно было уйти в базу. После 3000 работа сильно замедлилась, а к 3500 у меня окончательно закончилась оперативная память и парсер встал. Не проблема. Разбил наборы параметров на пакеты по 1000 штук и начал обрабатывать их по очереди.

Всё сработало, но хранить это всё в одной таблице показалось опасным. Рано или поздно я наберу столько данных что работать она будет очень медленно, а места занимать очень много. Решил изучить как вообще хранятся большие базы данных и как там справляются с моими проблемами. Понял, что ближе всего к тому, что мне нужно это Data Warehouse (DWH). Для моего проекта основная идея состоит в разделении базы данных на слои и разбиении данных на основную таблицу и максимальное количество словарей.

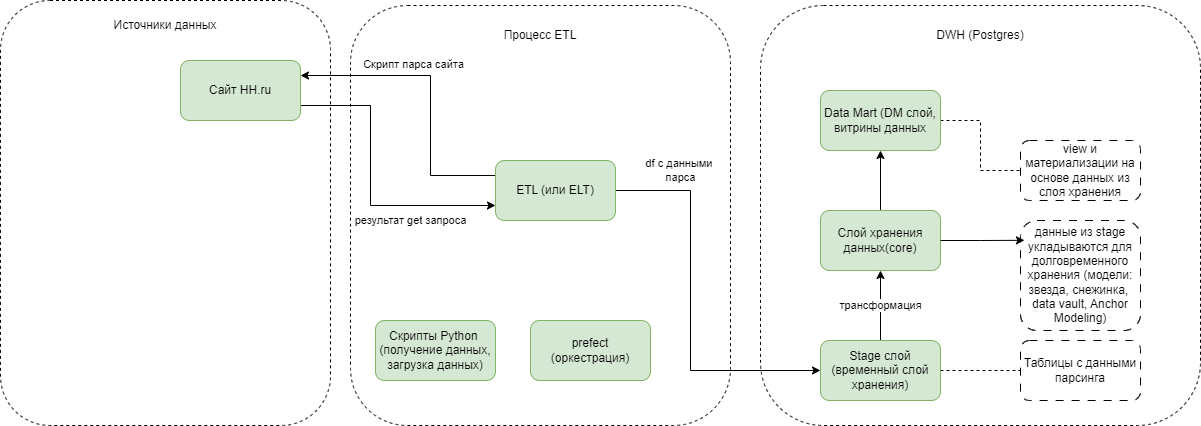
Первый слой это Stage. Туда данные попадают без обработки. В том виде в котором они получены с сайта или любого другого источника данных. Это дает больше свободы в процессе парсинга. Не надо загромождать код обработкой данных и парсинг в целом идет быстрее.

Второй слой Core предназначен для длительного хранения данных. Тут основная цель уменьшить вес базы. Происходит это за счет создания словарей. Допустим у нас есть столбец отвечающий за необходимый опыт. На HH.ru в нем всего четыре возможных значения: «Нет опыта», «от 1 года до 3 лет», «от 3 до 6 лет», «более 6 лет». Эти значения заносятся в виде текста, самого тяжелого для базы типа данных. И этих строчек должно быть по одной на каждую запись которых может быть и сто тысяч и миллионы в зависимости от базы. А мы переносим сами значения в словарь где будет всего четыре строчки которым соответствуют уникальные id. А в основной таблице заполняем столбец опыта этими самыми id. Вместо текстовых значений у нас остается всего четыре цифры для которых можно выбрать очень легкий тип данных. И так со всем что можно убрать в словари.

Третий слой Data Mart. Нужен для того чтобы было легко получать данные из базы. Там создаются представления и материализации таблиц. Один раз пишем код на sql и модем запрашивать нужные данные практически в один щелчок. Это же помогает настроить безопасность. Можно создать пользователей с доступом на чтение только тех элементов Data Mart на содержание которых у них есть доступ.



Разные слои создаются как разные слои в базе PostgreSQL. Сразу создал все таблицы и связи между ними. Трансформация между слоями Stage и Core будет происходить в самой базе с помощью процедур. Останется только запустить их из кода в конце парсинга. К этой схеме дорисовал свой парсер чтобы было наглядно видно что предстоит реализовывать.



Реализовал схему в коде. Провел первый сбор данных, начал писать процедуры для переноса данных в Core и понял, что главного не хватает. Нет скилов которые идут отдельно в конце большинства вакансий. Снова чтение документации по api. Выяснилось, что получить подробную информацию без токена всё же нельзя и придется получать токен.

Пока шло одобрение моей заявки на токен добавил оркестрацию через prefect, теперь мой код будет сам запускаться ночью и работать пока не получит все данные. Добавил в базу таблицу с календарем.

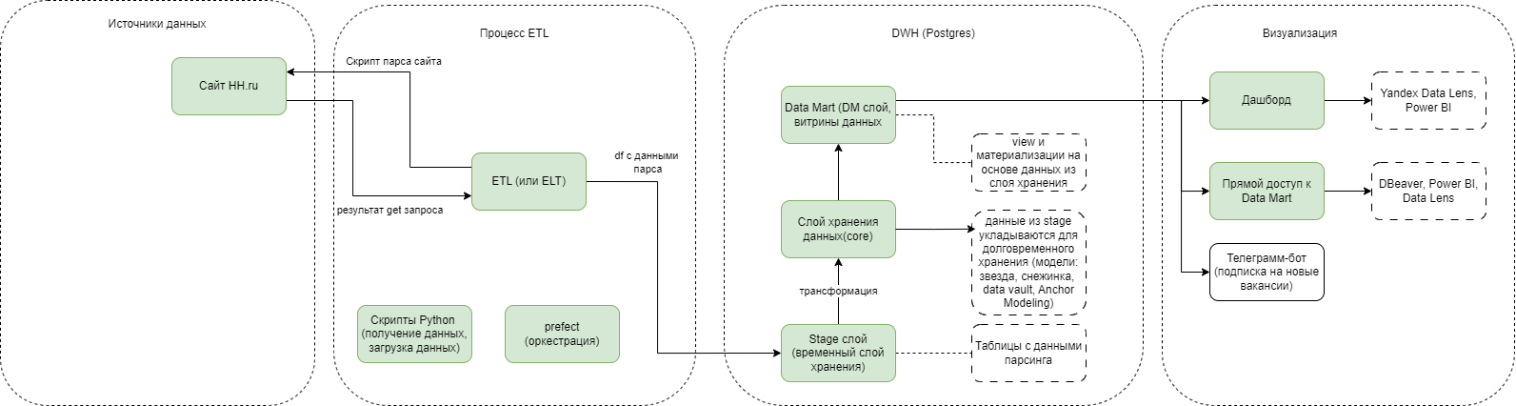
После получения токена пришлось переписать основной код парсера. Теперь я сначала получал id вакансий тем же способом что и раньше. А потом запрашивал подробную информацию по этим вакансиям. Так как для каждой вакансии указано несколько скилов пришлось сразу выделить для них отдельную таблицу в Stag. Иначе у меня бы размножились вакансии в основной таблице с и этим пришлось бы что-то делать.

Попробовал сделать что-нибудь со скоростью работы кода, но не смог. Вся проблема в том, что теперь я на каждую вакансию делаю один отдельный запрос. Их много, на них требуется время, а api вообще ограничивает запросы в секунду. В результате десятки тысяч вакансий обрабатываются почти весь день.

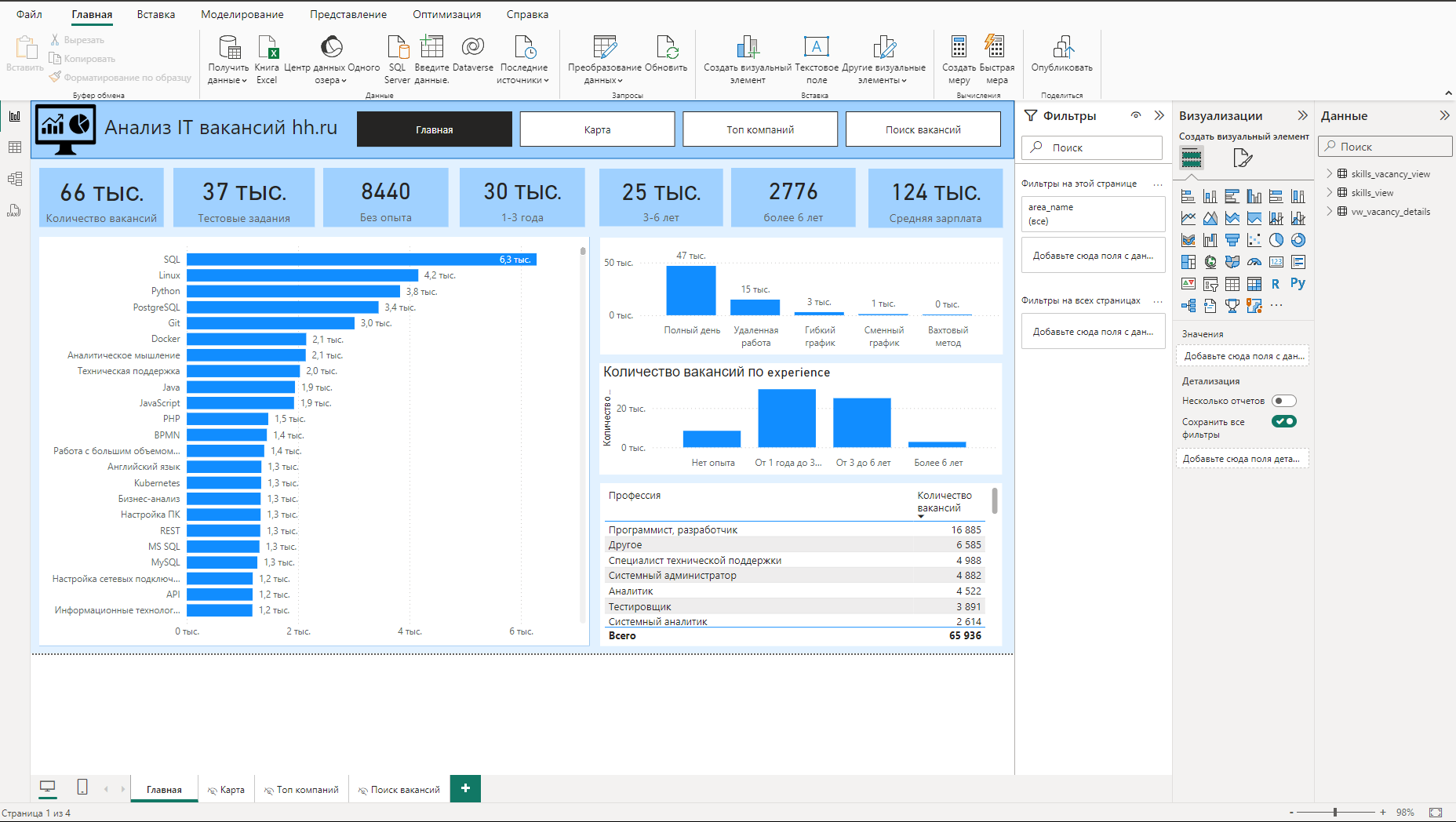
Написал процедуры для укладывания данных в Core. Там обошлось почти без проблем. Разве что даже в моей маленькой базе пришлось думать об оптимизации в некоторых местах.

И на этом этапе чисто случайно заметил, что некоторые зарплаты указаны в долларах. Знал я об этом и раньше, но совершенно не обращал внимания. В будущем зарплаты в валюте не дадут нормально отсортировать данные, а значит это надо исправлять. Нужно получить курсы валют и перевести зарплаты в рубли. Причем делать это надо динамически при переходе в Data Mart внутри представления. Тогда я получу актуальные зарплаты в рублях, а не те, которые были на момент сохранения данных. Ну, а с всё просто. Забрать актуальные курсы с сайта ЦБ, внести в отдельную таблицу. В коде представления умножать на последний курс нужной валюты.

И вот оно получилось. У меня есть база с нужными данными и с ней можно работать. Сразу дополнил схему тем как хочу с ней работать.



В блоке визуализации я работаю только с теми данными, которые попадают в Data Mart. Только нужные столбцы. Уже с собранными данными, а не в отдельных словарях. На основе этих данных составил дашборд по которому можно сразу увидеть нужную информацию.



Так же можно получить доступ напрямую к таблицу используя программы для работы с базами данных.