Тесты на SQlite, Pandas, SQLAlchemy проводились при подключении к базе данных, хранящейся на диске для равных условий. У DuckDB получился немного другой способ хранения данных - в своем собственном файле «.duckdb»

Вес «.db»: 2,21 ГБ

Вес «.duckdb»: 676 МБ

Получается, что «.duckdb» как-будто сжатый. Там хранится что-то по типу разреженной таблицы (поэтому NULL везде почти)

PostgreSQL (тесты на 200 мб)

Очень медленно работает операция вставки, поэтому данные из csv грузятся очень медленно.

Мое мнение, мои выводы:

Очень удобная среда для работы с БД. Запросы на 200 мб выполняются достаточно быстро (~ 1,2 секунды самый долгий).

Объяснение, сравнение:

У PostgreSQL оптимизированная архитектура. Как раз это и позволяет ей работать с огромными данными быстро, обрабатывать тысячи запросов в секунду.

SQlite

Мое мнение, мои выводы:

БД на SQlite удобно хранить, перемещать вместе с файлами Python. Кроме того, мне не понадобилось отдельное приложение для хранения данных, все записалось в файл .db. По скорости получается, что это самая медленная библиотека (по тестам).

Объяснение, сравнение:

В SQlite очень медленная скорость вставки, что дает медленный импорт БД.

DuckDB работает намного быстрее (по тестам), чем SQLite. Причиной может служить отсутствие индексации в таблице при работе SQLite (при импорте поставил index=False). Кроме того, DuckDB использовал другой файл для хранения базы данных. Можно заметить, что он весит намного меньше его аналога «.db». Следовательно, обработка данных будет быстрее в DuckDB, как я понял.

SQLAlchemy

Мое мнение, мои выводы:

Понравилось, что можно выбрать два пути для работы с данными. Обычные SQL запросы и ORM (не использовал в тестах). Суть SQLAlchemy – работа с ORM, но для того, чтобы условия были одинаковыми, я делал обычными SQL запросами и с подключением к файлу «.db»

Движок SQLAlchemy engine создает общий интерфейс к базе данных для выполнения инструкций SQL.

Объяснение, сравнение:

У SQLite и SQLAlchemy немного разный подход. SQlite - это движок хранения баз данных. В основном он используется для хранения и извлечения данных из файлов (структурированных. Таблица, например). А SQLAlchemy создана, как ORM библиотека, которая нужна для преобразования python классов в таблицы. Однако при подключении к одной базе данных на диске и выполнении обычных SQL запросов выигрывает SQLAlchemy. Это происходит из-за уровня абстракции БД, доступа к базе данных и структуры процессора. Умные люди пишут, что при использовании большого числа потоков процессора (от 2 хотя бы) для обработки SQlite, он обгонит по времени SQLAlchemy.

SQLAlchemy проигрывает Pandas и DuckDB из-за более медленных алгоритмов обработки данных

Pandas

Мое мнение, мои выводы:

Для меня Pandas показался очень функциональным и удобным. Кроме того оказался еще и библиотекой, работающей достаточно быстро (по тестам)

Pandas очень гибкий - позволяет работать с множеством форматов xlsx, csv, pickle, sql и тд. У Pandas есть возможность работать с DataFrame, не записывая его в таблицу, а просто используя методы подобно ORM в SQLAlchemy.

Объяснение, сравнение:

Pandas такой быстрый, потому что использует numpy под капотом. В силу того, что NumPy базируется на Fortran - это быстрая библиотека, а в силу того, что поддерживает векторные операции с многомерными массивами — крайне удобная (Numpy реализует высокоэффективные операции с массивами).

Именно поэтому время работы для разных запросов почти одинаковое (как показано на графике) и достаточно быстрое по сравнению с SQlite и SQLAlchemy.

DuckDB

Мое мнение, мои выводы:

Эта библиотека понравилась мне больше всего. Импорт данных занимает несколько секунд, создается специальный файл «.duckdb». По результатам это самая быстрая библиотека для работы с данными

Объяснение, сравнение:

Столбцово-векторный механизм этой базы данных позволяет выполнять аналитические типы запросов быстро. DuckDB оказывается значительно производительней Pandas и других библиотек. Pandas плохо работает на больших данных, т.к. эта библиотека не создана для работы на нескольких ядрах процессора. Pandas использует только одно ядро процессора для выполнения задач по манипулированию данными.

DuckDB поддерживает запросы через DuckDB CLI с SQL-интерфейсом в командной строке – также большой плюс DuckDB