InfluxDB

История

Компания Errplane, поддерживаемая Y Combinator, начала разработку InfluxDB как проекта с открытым исходным кодом (https://github.com/influxdata/influxdb) в конце 2013 года для мониторинга производительности и оповещения.

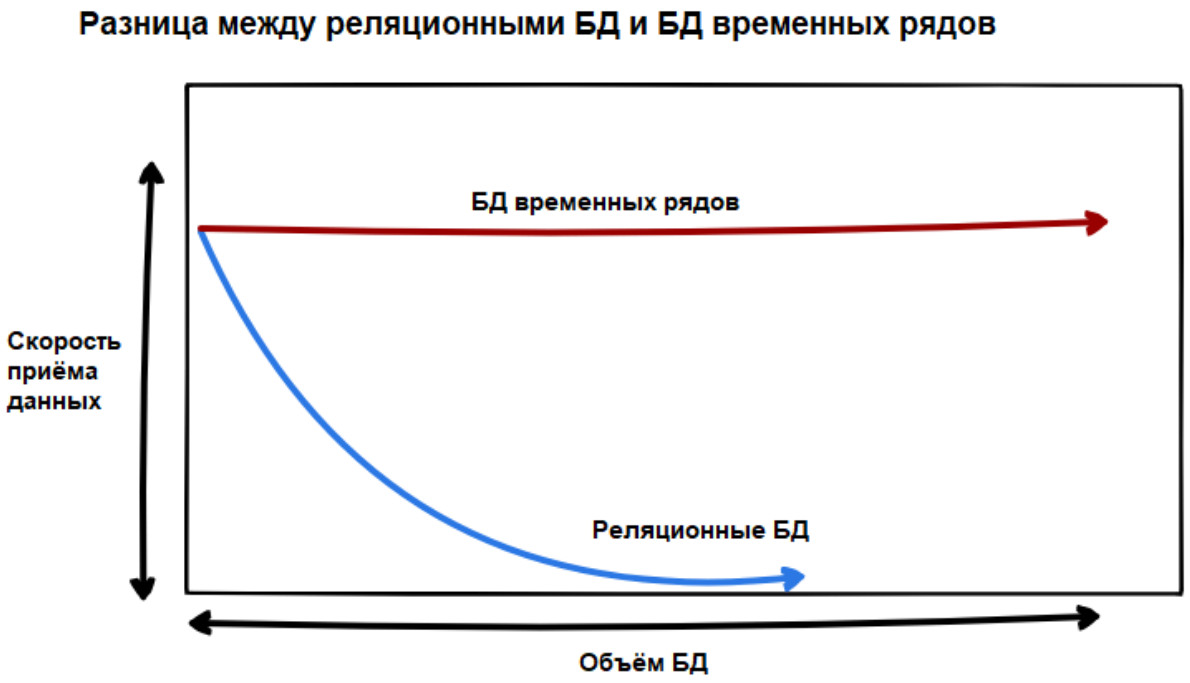
Основные сведения

Написанная на языке Go СУБД предназначена для работы с временными рядами или любыми другими данными, которые содержат в себе упорядоченные метки.

Синтаксис InfluxDB является SQL-подобный и предоставляет абстракции для работы с временными рядами. В основе лежат структуры данных: измерения, ряды и точки.

Точка представляет из себя набор пар ключ-значения, также называемые полем, и временная метка. Набор точек и ключа (teg sat) формируют ряды (series). Ряды, объединенные общим названием, называются измерением.

Целевым применением данной СУБД является запись большого объёма данных, с которым не справляются обычные реляционки:



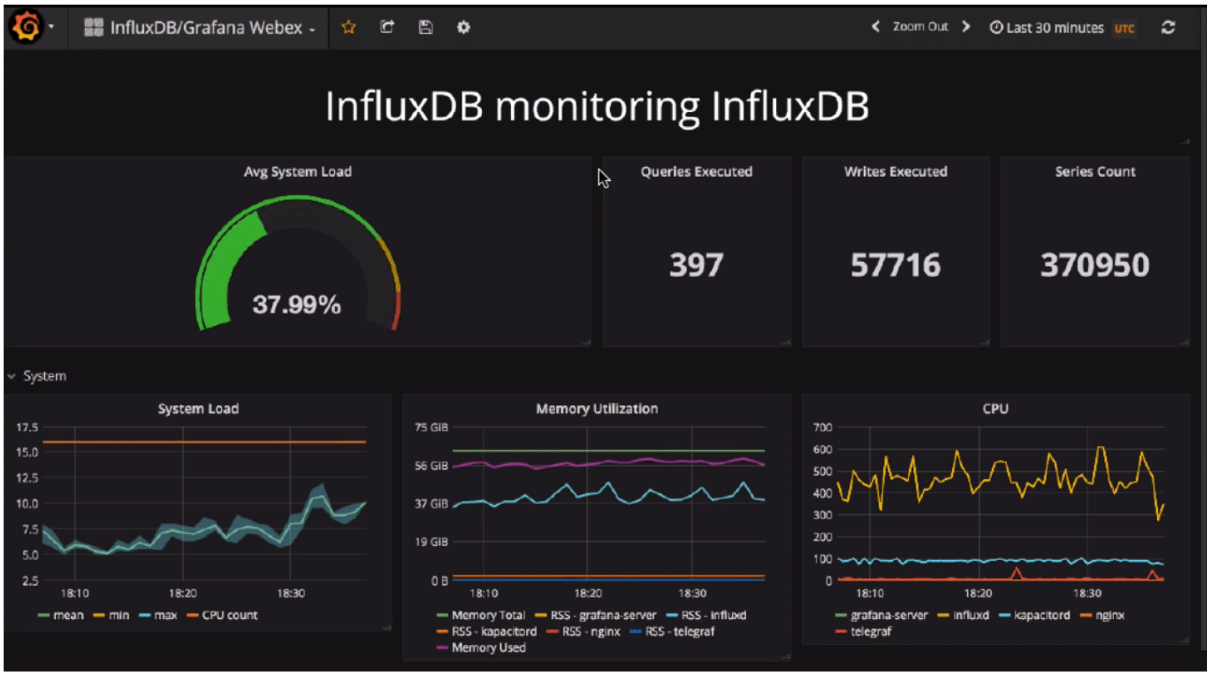
сурс: https://tproger.ru/translations/influxdb-guide/

Применение

Данная СУБД была создана для работы под большой нагрузкой: мониторинг датчиков, приложений, утилизации ресурсов. Исходя из прочитанных ресурсов, СУБД

построена на принципе, что любая информация будет записана, но нет гарантии, что запись будет только одна. Отсутствие проверки на дубликаты сильно увеличивает пропускную способность на запись для СУБД, однако делает её применение сомнительным в сфере финансов.

Боевым примером использование InfluxDB является связка InfluxDB + Grafana.



На работе моя команда использует InfluxDB+Grafana для мониторинга утилизации видеокарт на серверах. (скрин прикладывать не буду)

Кто может участвовать в разработке?

На данный момент InfluxDB является open-source под лицензией MIT. Основной разработкой занимается команда InfluxData. Репозиторий с их наработками находится в открытом доступе на GitHub. Каждый может предложить свои изменения. Для этого достаточно форкнуть main branch (актуальная версия СУБД) и после внесения изменений создать PR, согласно этому гайду

Поддерживаются ли транзакции в вашей СУБД?

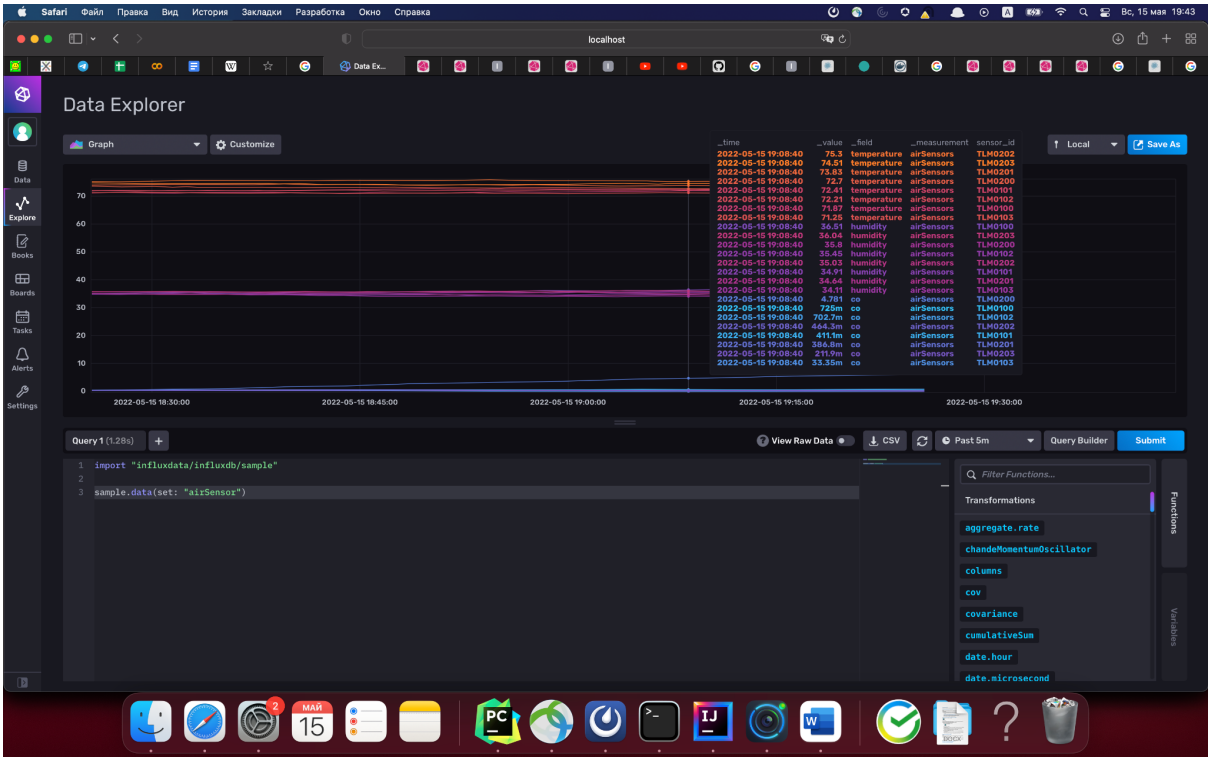
Основными транзакции InfluxDB является запись, чтения и агрегация данных. Встроенные механизмы позволяют удобно группировать данные по временному признаку. Например, надо отобразить данные за прошедший год. Данные сохранялись с timestamp каждые 5 минут и сами по себе они на экран не поместятся. Но встроенный функционал позволит удобно сгруппировать данные по месяцам или дням.

Напротив, транзакции, которые требуют изменения записей в InfluxDB почти не предусмотрены. Система не предусматривает операцию обновления записи (Update): для схожих целей используется вставка данных с тем же ключом. Операция удаления

записей (Delete) выполняется очень долго, от нескольких секунд на удаление одной записи.

Инструменты для взаимодействия с СУБД

Из коробки поставляется GUI со встроенным редактором для запросов.



Где найти документацию и пройти обучение?

Официальный репозиторий с туториалами: https://github.com/influxdata/influxdb

Сайт с документацией: https://docs.influxdata.com/influxdb/v2.2/

Гугл: https://letmegooglethat.com/?q=influxDB+tutorial

Как быть в курсе происходящего?

Разработчики ведут твиттер аккаунт, где сообщают новости: https://twitter.com/InfluxDB

Создание демобазы

Разработчики предоставили демобазы с данными: https://docs.influxdata.com/influxdb/cloud/reference/sample-data/#sample-datasets

В них представлено описание данных (данные с датчиков температуры, с сейсмических данных и так далее), их примерных размер. Данные добавляются в бд при помощи команды import.

Создайте свои собственные данные для демонстрации работы СУБД

InfluxDB потребляет данные из разных источников. Одним из них являются csv файлы. Создадим искусственный csv файл. Каждая строчка такого файла представляет из себя отдельное измерение и должна содержать в себе: измерение, field set (ключ), tag set (ключ более высокого уровня, является опциональный параметром) и метка времени.

Шаблон:

<measurement>[,<tag_key>=<tag_value>[,<tag_key>=<tag_value>]]
<field_key>=<field_value>[,<field_key>=<field_value>] [<timestamp>]

Пример:

myMeasurement,tag1=value1,tag2=value2 fieldKey="fieldValue" 1556813561098000000

Какие методы защиты поддерживаются вашей СУБД?

InfluxDB поддерживает создание секретов. Секреты - это пары ключ-значение, содержащие конфиденциальную информацию, доступ к которой вы хотите контролировать, например ключи API, пароли или сертификаты. Существует два варианта хранения секретов с помощью InfluxDB:

По умолчанию секреты кодируются в кодировке Base64 и хранятся во встроенном хранилище значений ключей InfluxDB, BoltDB.Вы также можете настроить сервер хранилища для хранения секретов.

InfluxDB поддерживает шифрование TSL. Если у сервера и у пользователя будут подписанные сертификаты, то они смогут общаться по защищённому каналу.

Какие методы восстановления поддерживаются в вашей СУБД?

Используйте команду influx backup для резервного копирования данных и метаданных, хранящихся в InfluxDB. InfluxDB копирует все данные и метаданные в набор файлов, хранящихся в указанном каталоге вашей локальной файловой системы.

influx backup <backup-path> -t <root-token>

где backup-path - путь к директории для сохранения бэкапа. root-token токен доступа.

Для восстановления данных из бэкапа запустите команду

influx restore <backup-path>

На время восстановления данные будут перемещены во временное хранилище и, если бэкап не удастся, будут возвращены на прежнее время.

Возможно ли применить термины Data Mining, Data Warehousing и OLAP в вашей СУБД?

InfluxDB очень специализированный инструмент. При работе с данными с временной меткой он позволяет быстро обрабатывать и трансформировать подобные данные. На форумах InfluxDB много аналитиков, которые используют InfluxDB. Однако, при использовании другого формата данных, применение InfluxDB теряет свой смысл.

Пример запроса

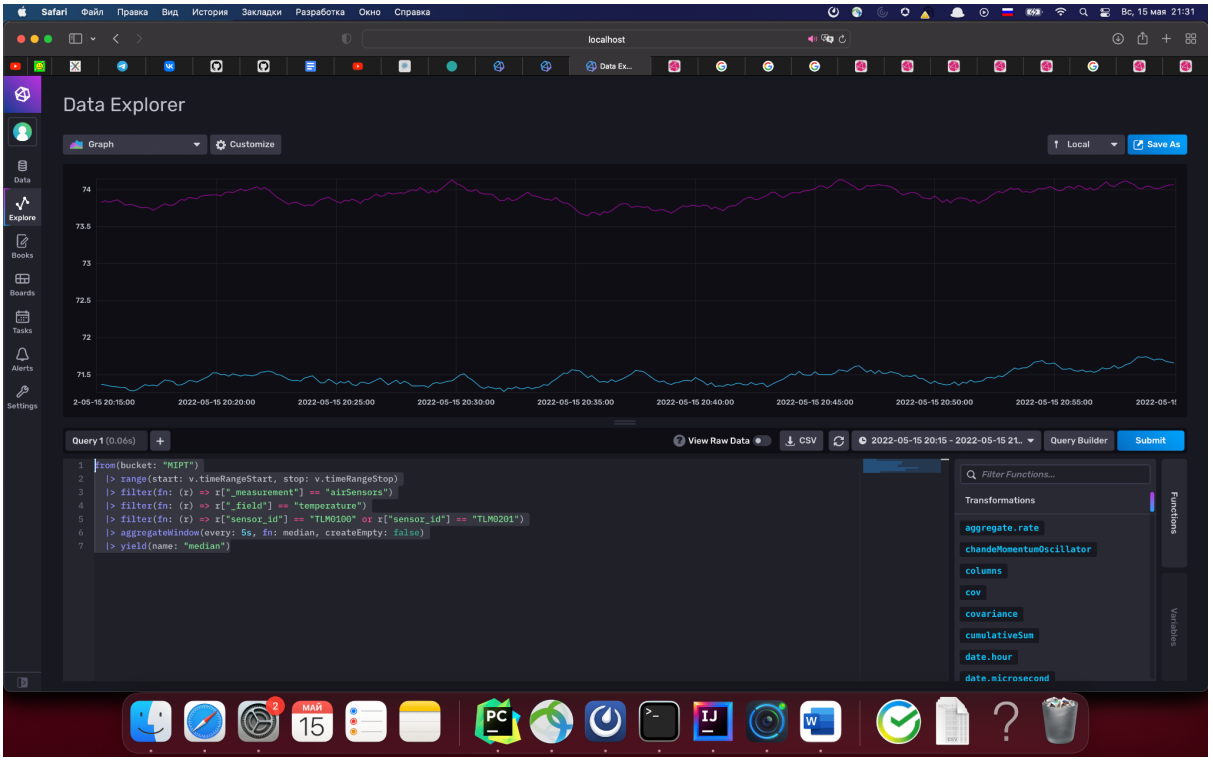
Воспользовавшись предоставленным демобазой, содержащей информацию с датчиков, я написал запросы. Данный запрос показывает средние значения влажности с выбранных датчиков за указанный период времени

```
from(bucket: "MIPT")
|> range(start: v.timeRangeStart, stop: v.timeRangeStop)
|> filter(fn: (r) => r["_measurement"] == "airSensors")
|> filter(fn: (r) => r["_field"] == "humidity")
|> filter(fn: (r) => r["sensor_id"] == "TLM0100" or r["sensor_id"] == "TLM0101" or r["sensor_id"] == "TLM0102" orr["sensor_id"] == "TLM0103" or r["sensor_id"] == "TLM0200" or r["sensor_id"] == "TLM0201" or r["sensor_id"] =="TLM0202" or r["sensor_id"] == "TLM0203")
|> aggregateWindow(every: 10s, fn: mean, createEmpty: false)
|> yield(name: "mean")
```

Этот скрипт запрашивает медианное значение температуры с двух датчиков с шагом в 5 секунд

```
from(bucket: "MIPT")
|> range(start: v.timeRangeStart, stop: v.timeRangeStop)
|> filter(fn: (r) => r["_measurement"] == "airSensors")
|> filter(fn: (r) => r["_field"] == "temperature")
|> filter(fn: (r) => r["sensor_id"] == "TLM0100" or r["sensor_id"] == "TLM0201")
|> aggregateWindow(every: 5s, fn: median, createEmpty: false)
|> yield(name: "median")
```

Результат:


Safari Файл Правка Вид История Закладки Разработка Окно Справка
Вс, 15 мая 21:31
localhost
Data Explorer
Graph
Customize
Local
Save As
Data
Explore
Books
Boards
Tasks
Alerts
Settings
74
73.5
73
72.5
72
71.5
2022-05-15 20:15:00
2022-05-15 20:20:00
2022-05-15 20:25:00
2022-05-15 20:30:00
2022-05-15 20:35:00
2022-05-15 20:40:00
2022-05-15 20:45:00
2022-05-15 20:50:00
2022-05-15 20:55:00
Query 1 (0.06s)
View Raw Data
CSV
2022-05-15 20:15 - 2022-05-15 21..
Query Builder
Submit
from(bucket: "MIPT")
|> range(start: v.timeRangeStart, stop: v.timeRangeStop)
|> filter(fn: (r) => r["_measurement"] == "airSensors")
|> filter(fn: (r) => r["_field"] == "temperature")
|> filter(fn: (r) => r["sensor_id"] == "TLM0100" or r["sensor_id"] == "TLM0201")
|> aggregateWindow(every: 5s, fn: median, createEmpty: false)
|> yield(name: "median")
Filter Functions...
Transformations
aggregate.rate
chandeMomentumOscillator
columns
cov
covariance
cumulativeSum
date.hour
date.microsecond
Functions
Variables