Инженерный вестник Дона, №9 (2023)
ivdon.ru/ru/magazine/archive/n9y2023/8676

Дедупликация больших объемов данных при помощи баз данных

Н.П. Плотникова[1], *В.А. Кевбрин*[1], *Д.А. Болохов*[2]

[1]*Мордовский государственный университет имени Н.П. Огарёва*
[2]*Московский технический университет связи и информатики*

Аннотация: На сегодняшний день огромное количество разнородной информации проходит через электронные вычислительные системы. Возникает критическая необходимость в анализе нескончаемого потока данных ограниченными средствами, а это, в свою очередь, требует структурирования информации. Одним из этапов решения задачи упорядочивания данных является дедупликация. В данной статье рассматривается метод удаления дубликатов с использованием баз данных, анализируются результаты тестирования работы с различными типами систем управления баз данных с разными наборами параметров.
Ключевые слова: дедупликация, база данных, поле, строка, текстовые данные, искусственная нейронная сеть, множества, запрос, программное обеспечение, неструктурированные данные.

Введение

В современном мире нескончаемые потоки данных проходят сквозь интернет и программные обеспечения различных информационных систем. Практически каждый цифровой процесс оставляет свой уникальный след. Вся эта информация находится в хаотичном распределенном состоянии. В различных сферах возникают задачи, связанные с обработкой этих неструктурированных данных: интернет-реклама, государственные услуги, банки. Достаточно вспомнить нейронные сети, которые получили широкое распространение и высокую скорость развития, благодаря наличию больших массивов доступной для исследований информации.

Обработка неструктурированных данных крайне тяжела. Почти всегда приходится прибегать к предобработке исходной информации. Одним из наиболее часто встречающихся этапов структуризации является дедупликация. Незачем хранить огромное количество дублирующейся информации. Этот избыток данных лишь требует дополнительных затрат как вычислительных, так и технических ресурсов. Наличие в данных дублей, при

работе с нейронными сетями [1, 2], может привести к ошибкам в обучении моделей.

Алгоритмы дедупликации

Существуют различные методы дедупликации данных: основанные на множествах, на блоках хранения и т. д. Многие из этих методов не предназначены для работы с большими объемами информации, либо работают с ними, но долго и неэффективно.

Рассмотрим следующий пример, пусть имеются текстовые данные определенного формата, например json [3, 4], которые являются выгрузкой из большой базы данных. Вся информация записана в одном текстовом файле, рис.1.


```
{"id":17,"z_id":11203,"b_id":11204,"p_01":"04605949001871","p_21":"0000000012345","p_91":"","p_92":""}
{"id":18,"z_id":11203,"b_id":11204,"p_01":"04605949001871","p_21":"0000000012346","p_91":"","p_92":""}
{"id":19,"z_id":11203,"b_id":11204,"p_01":"04605949001871","p_21":"0000000012347","p_91":"","p_92":""}
{"id":20,"z_id":11203,"b_id":11204,"p_01":"04605949001871","p_21":"0000000012348","p_91":"","p_92":""}
{"id":21,"z_id":11203,"b_id":11204,"p_01":"04605949001871","p_21":"0000000012349","p_91":"","p_92":""}
{"id":22,"z_id":11203,"b_id":11204,"p_01":"04605949001871","p_21":"0000000012340","p_91":"","p_92":""}
{"id":23,"z_id":11203,"b_id":11204,"p_01":"04605949001871","p_21":"0000000012341","p_91":"","p_92":""}
{"id":24,"z_id":11203,"b_id":11204,"p_01":"04605949001871","p_21":"0000000012342","p_91":"","p_92":""}
{"id":24,"z_id":11203,"b_id":11204,"p_01":"04605949001871","p_21":"0000000012342","p_91":"","p_92":""}
{"id":25,"z_id":11203,"b_id":11204,"p_01":"04605949001871","p_21":"0000000012343","p_91":"","p_92":""}
```


Рис. 1. – Пример исходных данных

На рис.1 видно, что ряд строк дублируется. В общем случае копии могут находится в любой части документа. Первое решение, которое может исключить дубли – использование множества (здесь и далее будет использована терминология и функционал языка программирования python). Но что, если объем данных превышает объем оперативной памяти?

В таком случае стоит рассмотреть другой вариант – считывать информацию из файла построчно и записывать в новый файл, предварительно сверяя, что строка ранее еще не встречалась. Недостатком данного метода является значительное количество времени, требуемое для обработки всего одного блока данных.

Алгоритм дедупликации с использованием баз данных

Стоит попробовать взять за основу последний описанный метод и попытаться устранить недостаток. В решении данной задачи могут помочь базы данных. Они оптимизированы для поиска информации, а именно это и тормозило процесс в предыдущем варианте.

Метод состоит из следующих этапов:

1) Чтение данных блоками.

2) Поиск в базе информации по обрабатываемым строкам.

3) Если информация в пункте 2 найдена, то переходим к обработке следующей части данных.

4) Если информация в пункте 2 не найдена, то сохраняем новые данные.

Данный алгоритм позволит избежать узкого места при поиске уже обработанных данных, так как современные системы управления баз данных (далее СУБД) способны быстро находить нужную информацию в своем хранилище. Использование ключевого слова UNIQUE (Postgres) гарантирует отсутствие дубликатов в данных. Рассмотренный метод удовлетворяет условию обработки информации блоками и решает проблему скорости поиска, но и у него есть свои недостатки:

1) Информация не обязательно будет представлять из себя текст, она может содержать изображения или бинарные данные.

2) Объем одного блока информации может оказаться слишком большим, что сильно усложнит работу базы данных и приведет к увеличению скорости поиска, даже с учетом оптимизаций запросов и структуры хранения.

Для решения перечисленных выше проблем используются хэш-значения [5]. В базе хранятся не сами данные, а, например, хэш посчитанный

по алгоритму md5 [6]. Хэш возможно посчитать почти от любого типа данных, и он не требует больших ресурсов для хранения.

Тестирование алгоритма

Описанный алгоритм реализован на языке программирования Python 3.9 в среде разработки PyCharm. В качестве СУБД протестированы Clickhouse [7], SSDB, Postgres [8]. Для удобства и анализа использованы: docker [9], mlflow [10]. Тестовый стенд имеет следующие характеристики: процессор – AMD Ryzen 3900, 64 гигабайт (далее Гб) оперативной памяти, SSD накопитель объемом 500 Гб. Данные для тестирования – 133327409 строк, из которых 33327409 дубликаты.

Результаты тестирования приведены на рис. 2-4, в таблице 1.

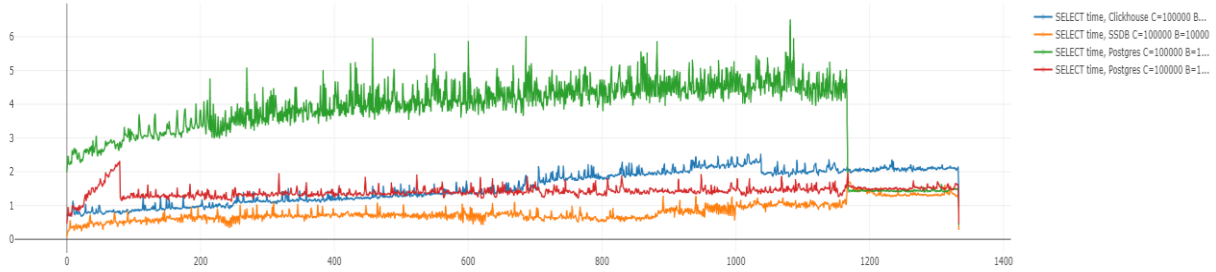

Рис. 2. – Замеры времени выполнения запроса на поиск, в зависимости от номера прочитанного блока

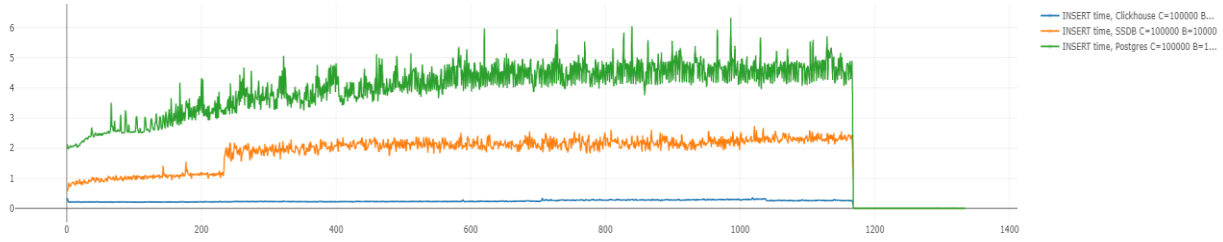

Рис. 3. – Замеры времени выполнения запроса на вставку, в зависимости от номера прочитанного блока

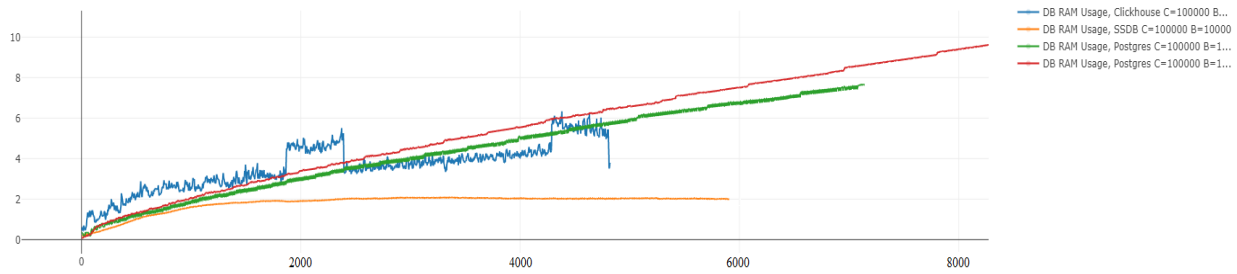

Рис. 4. – Замеры использования оперативной памяти СУБД в Гб, в зависимости от времени

Таблица № 1

Результаты тестирования алгоритма с разными типами баз данных

Параметр / Тип базы данных	Clickhouse	SSDB	Postgres №1	Postgres №2
БД CPU (среднее), %	689.7	38.61	44.34	93.49
БД RAM (среднее), Гб	3.793	1.981	7.65	10.51
Время выполнения, с	4399.9	5815.6	6762	8550.1
Занимаемый объем физической памяти, Гб	3.1	5.1	10.8	14.6

Необходимо сделать следующие замечания:

1) Метод подготовки данных во всех случаях, кроме Postgres №1 одинаков. В этом конкретном случае использован специфичный запрос и требуется структура данных json, листинг кода приведен на рис.5.

2) Один блок данных для чтения (chunksize) равен 100 тысячам строк.

3) Один блок данных для записи (batchsize) равен 10 тысячам строк.

4) Результатом работы программы является множество текстовых файлов объемом равным batchsize. Запись происходит в мультипроцессинге.

```
def save_batch_postgres(self, data_batch: list):
    data_batch = [{"hash_line": hash(i), "line": i} for i in data_batch]
    start_time = time.time()
    self.client.execute(f"""WITH select_tmp as (
                                                SELECT DISTINCT value
                                                FROM jsonb_array_elements('{json.dumps(data_batch).decode("utf-8")}'::jsonb)
                                                WHERE (
                                                    SELECT *
                                                    FROM Tests
                                                    WHERE hash_line=value->>'hash_line'
                                                ) isnull
                                            ),
                                insert_tmp as (INSERT INTO Tests SELECT value->>'hash_line' FROM select_tmp)
                            SELECT value->>'line' FROM select_tmp""")
    query_result = self.client.fetchall()
    self.logger.info(f"SELECT time: {time.time() - start_time}")
    results = [row[0] for row in query_result]
    self.connect_postgres.commit()
    return results
```

Рис. 5. – Листинг кода сохранения данных Postgres №1

Из приведенных результатов экспериментов можно сделать следующие выводы:

1) Postgres вне зависимости от типа запроса не подходит для решения поставленной задачи. Данная СУБД тратит больше всего времени как на операции вставки, так и на поиск информации. Возможно, это связано с особенностями построения индексов.

2) SSDB – самый стабильный вариант для решения поставленной задачи. Затраты оперативной памяти достигли определенной границы и больше не росли. Время на чтение и запись блоков данных также изменяется не сильно.

3) Clickhouse – оптимальный вариант для решения задачи. У данной СУБД время поиска несколько больше, чем у SSDB, но время выполнения операции вставки значительно ниже. Это также подтверждается наименьшими общим временем выполнения и объемом занимаемой физической памяти.

На основе выбранного варианта – Clickhouse, дополнительно проведены эксперименты с разными размерами chunksize и batchsize, результаты которых представлены в таблице 2.

Таблица 2

Результаты тестирования с Clickhouse при различных параметрах chunksize, batchsize

Метрика/Параметры	C=100000, B=10000	C=1000000, B=100000	C=1000000, B=1000000
БД RAM (среднее), Гб	3.793	4.312	5.314
Время выполнения, с	4399.9	3847.8	3046.6

По результатам экспериментов, приведенным в таблице 2, можно сделать следующие выводы:

1) Чем больше оперативной памяти на стенде системы, тем более крупные параметры chunksize и batchsize возможно выставить.

2) Чем выше значение рассматриваемых параметров, тем лучше результаты.

3) В случае, если chunksize=batchsize, результат, вопреки ожиданиям, не ухудшается, а наоборот улучшается. Мультипроцессинг негативно влияет на скорость работы. Вполне возможно, что затраты на работу с мультипроцессингом, в данном конкретном случае, превышают выигрываемое за его счет время.

Заключение

В ходе проведения экспериментов по дедупликации данных на испытательном стенде с разными типами СУБД, наиболее подходящим вариантом является Clickhouse с настройкой параметров chunksize и batchsize в наибольшее возможное значение.

Предложенный в настоящей статье метод дедупликации данных применен в разработанном программном обеспечении для обработки и структурирования входных потоков больших массивов разнородных данных.

Литература

1. Богомолов И.А., Дмитриев А.С., Киселев Ю.В. Использование машинного обучения для продвижения сайтов // Инженерный вестник Дона. 2023. №5. URL: ivdon.ru/ru/magazine/archive/n5y2023/8379

2. Бадашев В.В., Скоба А.Н. Информационная система прогнозирования собираемости платежей в отделениях почтовой связи «Почта России» с использованием машинного обучения // Инженерный вестник Дона. 2023. №6. URL: ivdon.ru/ru/magazine/archive/n6y2023/8493

3. Crockford D. JSON: The Fat-Free Alternative to XML // Boston 2006. URL: www.json.org/fatfree.html#:~:text=JSON%3A%20The%20Fat%2DFree%20Alternative%20to%20XML

4. Наумов Р.К., Железков Н.Э. Сравнительный анализ форматов хранения текстовых данных для дальнейшей обработки методами машинного обучения // Научный результат. Информационные технологии. 2021. №1. С. 40-47.

5. Мебония М.А. Федорова О.В. Сравнительное исследование хэш-алгоритмов в криптографии // Вестник науки. 2022. №12. С. 435-438.

6. Rivest R. The MD5 Message-Digest Algorithm // RFC 1321. MIT Laboratory for Computer Science and RSA Data Security: 1991. URL: ietf.org/rfc/rfc1321.txt

7. Mostafa J., Chilingaryan S., Wehbi S., Kopmann A. SciTS: A Benchmark for Time-Series Databases in Scientific Experiments and Industrial Internet of Things // Proceedings of the 34th International Conference on Scientific and Statistical Database Management. 2022. P. 12.

8. Hellerstein J.H. Looking back at Postgres // Making Databases Work: the Pragmatic Wisdom of Michael Stonebraker. 2018. pp. 205–224.

9. Karim Z. Take the confusion out of Docker, VMs, and microservices // IBM Developer 2019. URL: developer.ibm.com/articles/breaking-down-docker-and-microservices (дата обращения: 15.08.2023).

10. Schlegel M., Sattler K. Management of Machine Learning Lifecycle Artifacts: A Survey // ACM SIGMOD Record. 2022. №51. pp. 18-35.

References

1. Bogomolov I.A., Dmitriev A.S., Kiselev Yu.V. Inzhenernyj vestnik Dona. 2023. №5. URL: ivdon.ru/ru/magazine/archive/n5y2023/8379

2. Badashev V.V., Skoba A.N. Inzhenernyj vestnik Dona. 2023. №6. URL: ivdon.ru/ru/magazine/archive/n6y2023/8493

3. Crockford D. JSON: The Fat-Free Alternative to XML. Boston: 2006. URL: json.org/fatfree.html#:~:text=JSON%3A%20The%20Fat%2DFree%20Alternative%20to%20XML

4. Naumov R.K., Zhelezkov N.E. Nauchnyj rezul'tat. Informatsionnye tekhnologii. 2021. №1. pp. 40-47.

5. Мебония М.А. Федорова О.В. Vestnik nauki. 2022. №12. pp. 435-438.

6. Rivest R. The MD5 Message-Digest Algorithm. RFC 1321. MIT Laboratory for Computer Science and RSA Data Security: 1991. URL: www.ietf.org/rfc/rfc1321.txt

7. Mostafa J., Chilingaryan S., Wehbi S., Kopmann A. SciTS: A Benchmark for Time-Series Databases in Scientific Experiments and Industrial Internet of Things. Proceedings of the 34th International Conference on Scientific and Statistical Database Management. 2022. P. 12.

8. Hellerstein J.H. Looking back at Postgres. Making Databases Work: the Pragmatic Wisdom of Michael Stonebraker. 2018. pp. 205–224.

9. Karim Z. Take the confsion out of Docker, VMs, and microservices. IBM Developer 2019. URL: developer.ibm.com/articles/breaking-down-docker-and-microservices (дата обращения: 15.08.2023).

10. Schlegel M., Sattler K. Management of Machine Learning Lifecycle Artifacts: A Survey. ACM SIGMOD Record. 2022. №51. pp. 18-35.