

УДК 004.657

ПРИНЦИПЫ ФУНКЦИОНИРОВАНИЯ ТАБЛИЧНОГО ФОРМАТА APACHE ICEBERGB АНАЛИТИКЕ BIG DATA

Мамонова В.Н., студент гр.482571

*Белорусский государственный университет информатики и радиоэлектроники
Институт информационных технологий
г. Минск, Республика Беларусь*

Скудняков Ю.А. – канд. техн. наук, доцент кафедры ИСиТ

Аннотация. Рассмотрена актуальная тема — обработка больших данных Big Data. Приведен наиболее эффективный и развивающийся открытый формат таблиц, который обеспечивает высокую производительность и мощные функции для работы с данными. Проанализирована архитектура формата Apache Iceberg, принципы его функционирования, механизмы управления метаданными, а также оценка его преимуществ при использовании в аналитических системах обработки больших данных. Представлена математическая модель для наглядности алгоритмов взаимодействия с Apache Iceberg.

Ключевые слова. Метаданные, транзакции, Apache Iceberg, база данных, снапшот.

Введение. В настоящее время одной из важных, актуальных, перспективных и быстро развивающихся областей информационных технологий является разработка методов и средств анализа, хранения и обработки больших данных (Big Data).

Результаты анализа и обработки больших данных могут использоваться в различных сферах человеческой деятельности, в частности, в создании и практического использования искусственных интеллектуальных систем, прогнозирования погоды, в фармакологии для получения эффективных лекарственных средств, в принятии сложных решений и получении многих важных результатов в других областях.

Поскольку обработка больших массивов данных достаточно трудоемкая работа, то ее эффективное решение возможно при использовании распределенных компьютерных систем. Для построения и развития системы Big Data необходимо использовать широкий набор программных средств и другого инструментария, например, такого как: аналитические базы данных GreenPlum, Vertica, фреймворки распределённой обработки данных Hadoop, Spark, брокеры сообщений RabbitMQ, Kafka, инструменты трансформации и загрузки данных Apache Airflow, Prefect, сервисы координации Consul, ZooKeeper и многие другие системы и средства. Однако, ситуация может вызвать осложнения из-за постоянного изменения стека используемых технологий.

Разработка эффективных алгоритмов и новых подходов позволяет решать задачи интеграции Big Data с учетом закономерностей исходных данных с их последующей систематизацией в индивидуально заданную форму.

Анализ закономерностей исходных данных происходит за счет использования аналитических методов. Основной особенностью применения данных методов является использование ранее полученных экспериментальных данных. Необходимость в экспериментальных значениях обусловлена отсутствием единой математической модели, способной описать закономерности данных, полученных в результате различного рода деятельности. Например, изменения значений курса валют на фондовом рынке могут существенно отличаться от ряда показаний научных приборов.

К аналитическим методам можно отнести:

- получение экспериментальных данных;
- систематизация данных;
- проведение тестирования полученного решения на практике;
- проведение процесса на соответствие полученных результатов реальным эксплуатационным требованиям;
- составление новых данных.

Современные аналитические системы обработки больших данных Big Data всё чаще строятся на использовании архитектуры Data Lake, предполагающей хранение данных в виде файлов в распределённых хранилищах.

Такой подход обеспечивает высокую масштабируемость и низкую стоимость хранения, однако одновременно порождает ряд фундаментальных проблем, связанных с управлением схемой данных, согласованностью операций записи, производительностью аналитических запросов и поддержкой конкурентного доступа.

Традиционные файловые форматы и табличные абстракции, используемые в экосистеме Hadoop (например, Apache Hive), изначально не проектировались с учётом требований транзакционности и интенсивной аналитической нагрузки.

Таким образом, возникает противоречие между преимуществами архитектуры Data Lake, ориентированной на масштабируемость и экономичность, и требованиями к надёжности,

согласованности и гибкости управления данными, характерными для классических реляционных систем управления базами данных. Разрешение данного противоречия привело к появлению нового класса решений — табличных форматов нового поколения, обеспечивающих семантику реляционных таблиц поверх распределённых файловых хранилищ.

В период 2017–2019 г.г. крупные зарубежные компании, такие как Facebook и Uber, столкнулись с существенными ограничениями при использовании Apache Hive для анализа больших объёмов данных. Hive предоставляет возможность обращения к данным, хранящимся в распределённой файловой системе Hadoop (HDFS), с использованием SQL-подобного языка запросов.

Однако данная NoSQL-система (нереляционная) не предназначена для полноценной поддержки транзакционных нагрузок и динамических изменений схемы данных.

В частности, таблицы Hive используют фиксированную схему партиционирования, что делает изменение логики разбиения данных сложной и ресурсоёмкой операцией.

Для перераспределения данных зачастую требуется создание новой таблицы и полная перезагрузка набора данных, что приводит к значительным временным и вычислительным затратам [1].

К основным недостаткам Apache Hive относятся следующие:

- проблема консистентного чтения данных (операции записи в Hive не являются атомарными, а в случае сбоя выполнения задания записи в таблице могут оставаться частично записанные или неконсистентные данные, так называемые «мусорные» файлы, обнаружение и удаление которых представляет собой нетривиальную задачу);

- производительность сканирования метаданных (для получения списка файлов таблицы Hive вынужден рекурсивно обходить каталоги файловой системы, а при наличии миллионов файлов и большого числа партиций данная операция становится крайне медленной и негативно влияет на время выполнения аналитических запросов);

- хрупкость и сложность эволюции схемы и партиционирования (изменение типов столбцов, их порядка или схемы партиционирования в Hive является рискованной операцией и зачастую требует переписывания всей таблицы).

Кроме того, Apache Hive зависит от централизованного HiveMetastore, который может становиться узким местом при масштабировании системы.

Указанные ограничения привели к появлению нового класса решений — табличных форматов нового поколения, обеспечивающих семантику реляционных таблиц поверх файловых хранилищ.

Для наглядного сравнения традиционного подхода и форматов нового поколения в таблице 1 приведено сопоставление ключевых характеристик Apache Hive и Apache Iceberg.

Таблица 1 – Сравнение Apache Hive и Apache Iceberg

Критерий	Hive	Iceberg
ACID	Нет	Да
Time Travel	Нет	Да
Эволюция схемы	Ограничена	Полная
Конкурентная запись	Нет	Да
Централизация метаданных	Нет	Да

Основная часть. В данной статье подробно рассматривается архитектура Apache Iceberg, принципы его функционирования, механизм управления метаданными, а также его преимущества по сравнению с альтернативными решениями.

Apache Iceberg представляет собой спецификацию табличного формата для обработки крупных аналитических наборов данных. Iceberg предоставляет вычислительным системам, таким как Apache Spark, Trino, PrestoDB, Flink, Hive и Impala, унифицированный и высокопроизводительный табличный интерфейс, работающий аналогично таблицам в реляционных СУБД.

Ключевой особенностью Apache Iceberg и подобных ему форматов является обеспечение централизации метаданных: такой подход привел к существенному повышению производительности аналитических запросов и устранить архитектурные ограничения Apache Hive.

В отличие от Hive, где метаданные тесно связаны с физической структурой файловой системы, Iceberg хранит данные и метаданные в объектном хранилище и управляет ими через многоуровневую архитектуру.

Архитектура Apache Iceberg включает три иерархически организованных слоя. Верхний уровень представлен слоем каталога, отвечающим за управление версиями таблиц.

Следующий уровень — слой метаданных, включающий файлы метаданных, списки манифестов и манифест-файлы. Нижний уровень образуют непосредственно файлы данных, хранящие записи таблицы [2].

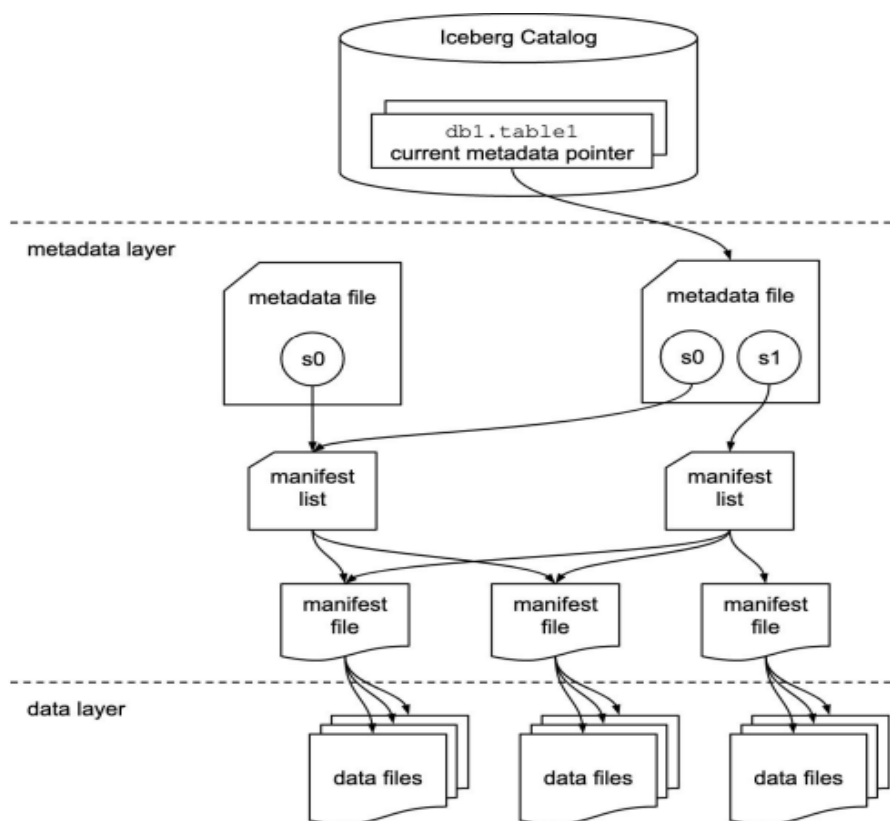


Рисунок 1 – Уровни архитектуры Iceberg

В слое метаданных хранится информация о консистентных состояниях таблицы, которые в терминологии Apache Iceberg называются снимками (snapshot). Снимок представляет собой неизменяемый набор файлов, описывающих результат конкретной транзакции. Каждый снимок содержит ссылки на соответствующие manifest-файлы, которые, в свою очередь, описывают наборы data-файлов.

По мере выполнения операций записи количество снимков увеличивается, что позволяет обращаться к предыдущим версиям таблицы. Для ограничения объёма хранимых метаданных используется политика хранения резервных копий (BackupRetention Policy), определяющая срок хранения снимков, который в типичных сценариях составляет около семи дней.

Iceberg предоставляет инженерам данных расширенные средства для эффективного управления жизненным циклом данных в аналитических хранилищах.

Эволюция схемы данных (Schema Evolution) в Apache Iceberg основана на использовании уникальных идентификаторов полей. Формат поддерживает добавление, удаление и переименование столбцов, а также изменение их порядка без переписывания существующих данных. Благодаря идентификации полей по ID корректная интерпретация данных сохраняется независимо от изменений схемы.

Эволюция партиционирования (Partition Evolution) позволяет изменять стратегию разбиения таблицы, например переходить от дневного партиционирования к часовому, без модификации ранее записанных данных. Apache Iceberg обеспечивает корректную обработку запросов к данным, используя различные схемы партиционирования, за счёт хранения информации о партициях в метаданных.

Поддержка временных запросов (Time Travel) реализуется через механизм снимков. Каждая операция фиксации изменений приводит к созданию нового снимка, что позволяет выполнять запросы к состоянию таблицы на определённый момент времени или по идентификатору снимка. Данная возможность особенно важна для задач аудита, анализа ошибок в конвейерах обработки данных и восстановления после сбоев.

Скрытое партиционирование (HiddenPartitioning) позволяет автоматически формировать партиционные значения на основе значений столбцов. Например, из значения временной метки может быть автоматически сформирована партиция по дате. Такой подход абстрагирует пользователей от физической структуры хранения данных и упрощает написание аналитических запросов [3-4].

Поддержка ACID-транзакций в Apache Iceberg реализуется за счёт механизма атомарной замены указателя на файл метаданных таблицы. Механизм фиксации изменений и изоляции снимков наглядно представлен на рисунке 2.

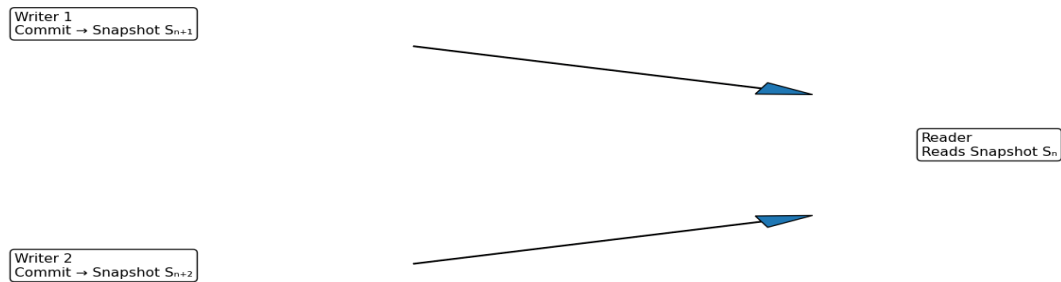


Рисунок 2 – Механизм commit-операций и snapshotisolation в Apache Iceberg

Данный подход обеспечивает выполнение свойств атомарности, согласованности, изоляции и долговечности транзакций. Все операции чтения выполняются относительно фиксированного снимка таблицы, что соответствует модели snapshotisolation и позволяет безопасно выполнять параллельные операции чтения и записи без нарушения целостности данных. Для формального описания механизма управления версиями данных в Apache Iceberg введём следующие обозначения.

Пусть T обозначает логическую таблицу Apache Iceberg, представляющую собой абстракцию над множеством версий данных. Состояние таблицы T в момент времени i задаётся снимком S_i , соответствующим результату i -й зафиксированной транзакции.

Математическая модель управления снимками:

Состояние таблицы T в дискретный момент времени i описывается снимком S_i , при этом множество всех снимков таблицы определяется как:

$$S = \{ S_0, S_1, \dots, S_n \}.$$

Каждый снимок S_i представлен в виде множества manifest-файлов:

$S_i = \{ M_1, M_2, \dots, M_k \}$, где S_i — i -й снимок таблицы, M_j — manifest-файлы, входящие в снимок.

Каждый manifest-файл M_j содержит описание набора физических файлов данных:

$M_j = \{ F_1, F_2, \dots, F_m \}$, где F_m — физические файлы данных, размещённые в распределённом хранилище.

Переход между снимками может быть описан функцией:

$S_{i+1} = \text{Commit}(S_i, \Delta_i)$, где Δ_i — множество изменений данных, выполненных в рамках i -й транзакции.

Алгоритм фиксации изменений в Apache Iceberg может быть представлен в виде блок-схемы, приведённой на рисунке 3.

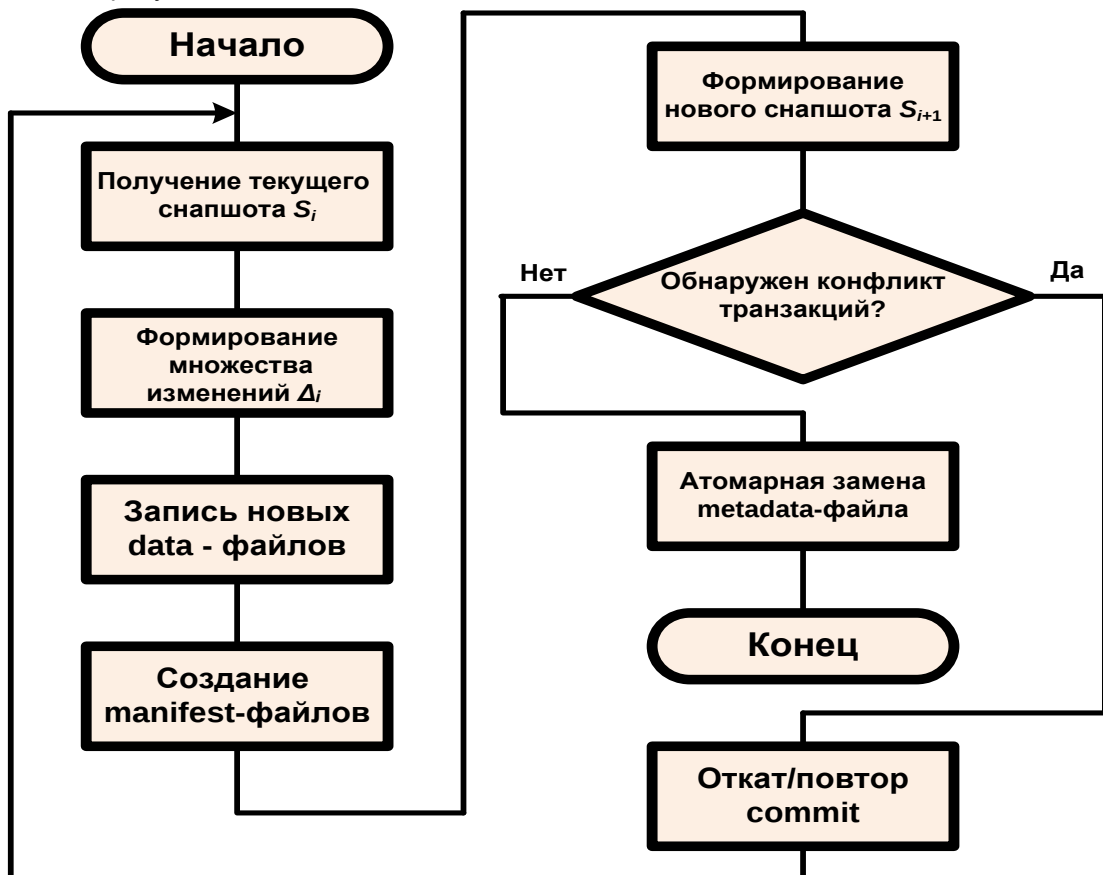


Рисунок 3 – Блок-схема алгоритма фиксации изменений (commit) в Apache Iceberg

Изменение и удаление данных в Apache Iceberg: в отличие от традиционных реляционных СУБД, Apache Iceberg не выполняет физическое удаление строк из файлов данных. Это связано с использованием неизменяемых файлов (immutable datafiles) в распределённом хранилище. Операции удаления данных (DELETE) реализуются с использованием стратегий Copy-on-Write и Merge-on-Read. В режиме Copy-on-Write создаётся новый файл данных без удалённых строк, после чего старый файл исключается из активного снимка.

В режиме Merge-on-Read удалённые строки фиксируются в отдельных delete-файлах, которые применяются во время выполнения запросов. Операции обновления данных (UPDATE) логически представляют собой комбинацию операций удаления и вставки. Физическое удаление устаревших файлов данных выполняется асинхронно в процессе очистки (garbage collection), что обеспечивает согласованность и отказоустойчивость системы [5].

Кроме того, Apache Iceberg собирает статистическую информацию, необходимую для оптимизации выполнения запросов. Точные статистики, такие как минимальные и максимальные значения столбцов и количество NULL-значений, хранятся в manifest-файлах и используются для dataskipping.

Приблизительные статистики, включая количество различных значений (NDV), вычисляемые с использованием theta-sketches, сохраняются в отдельных бинарных файлах в формате Puffin и применяются в cost-based оптимизации.

Заключение. В процессе исследования было проведено изучение существующих методов, технологий и средств организации хранения и обработки данных на основе архитектуры DataLake и табличных форматов Apache Iceberg и Apache Hive. Проанализированы их основные достоинства и недостатки, в том числе с точки зрения обеспечения целостности данных, согласованности состояний и устойчивости к конкурентному доступу.

На основе полученных результатов была составлена математическая модель и алгоритма фиксации изменений.

Предложенная модель позволяет формализовать механизм работы со снимками, выявления конфликтов транзакций и атомарной публикации изменений, что способствует повышению надёжности и предсказуемости функционирования систем хранения данных.

В качестве перспективного направления дальнейшего развития полученных результатов можно рассматривать интеграцию предложенного подхода с технологиями искусственного интеллекта, а также его адаптацию для высокопроизводительных и распределённых вычислительных сред.

В перспективе практического внедрения не исключается использование квантовых технологий, обладающих существенно большей вычислительной производительностью по сравнению с современными классическими компьютерными системами.

Список использованных источников:

1. Уайт, Т. Hadoop. Подробное руководство / Т. Н. Уайт; пер. с англ. – 4-е изд. – СПб.: Питер, 2020. – 672 с.
2. Хьюз, Дж. Apache Iceberg: архитектурный взгляд изнутри: руководство / Дж. Хьюз; Dremio[Электронный ресурс]. – Электрон. дан. – Доступ свободный. – Режим доступа: <https://www.dremio.com/resources/guides/apache-iceberg-an-architectural-look-under-the-covers/> – Загл. с экрана. – Яз. англ. – Дата доступа: 26.01.2026.
3. Хайман, Д., Apache Iceberg: The Definitive Guide. Архитектура табличных форматов для аналитических данных в масштабе петабайт / Д. Хайман, Дж. Хьюз; пер. с англ. – М.: ДМК Пресс, 2024. – 450 с.
4. Apache Iceberg. // Big Data School[Электронный ресурс]. – Электрон. дан. – Доступ свободный. – Режим доступа: <https://bigdataschool.ru/wiki/apache-iceberg/> – Загл. с экрана. – Яз. рус. – Дата обращения: 29.01.2026.
5. The Apache Software Foundation. Apache Iceberg. Official Documentation and Specifications [Электронный ресурс]. – Электрон. дан. – Доступ свободный. – Режим доступа: <https://iceberg.apache.org/> – Загл. с экрана. – Яз. англ. – Дата доступа: 01.02.2026.

UDC 004.657

PRINCIPLES OF FUNCTIONING OF THE APACHE ICEBERG TABLE FORMAT IN BIG DATA ANALYTICS

Mamonova V.N.

*Institute of Information Technologies of the Belarusian State University of Informatics and Radioelectronics,
Minsk, Republic of Belarus*

Skudnyakov Yu.A. – Candidate of Engineering Sciences, Associate Professor

Annotation. This article examines the topic of Big Data processing, a highly relevant topic. It presents the most efficient and evolving open table format, which provides high performance and powerful data manipulation features. It analyzes the Apache Iceberg format's architecture, operating principles, and metadata management mechanisms, as well as assesses its advantages for use in big data analytics systems. A mathematical model is presented to illustrate the algorithms for interacting with Apache Iceberg.

Keywords. Metadata, transactions, Apache Iceberg, database, snapshot.