УДК 004.023:519.16

ОТКРЫТЫЕ ТАБЛИЧНЫЕ ФОРМАТЫ В БАНКОВСКОЙ АНАЛИТИКЕ: ПРАКТИЧЕСКОЕ ИССЛЕДОВАНИЕ ICEBERG И PAIMON

OPEN TABLE FORMATS IN BANKING ANALYTICS: A PRACTICAL STUDY OF ICEBERG AND PAIMON

Иевлев К.О., аспирант, ассистент кафедры «Интеллектуальный анализ данных», ФГБОУ ВО «Московский технический университет связи и информатики»

Сурпин В.П., канд. техн. наук, исполнительный директор, Департамент управления данными (Sberdata), Сбербанк России

Городничев М.Г., канд. техн. наук, доцент, декан факультета «Информационные технологии», заведующий кафедрой «Математическая кибернетика и информационные технологии», ФГБОУ ВО «Московский технический университет связи и информатики»

Ievlev K.O., graduate student, Lecturer assistant at the department of data mining, Moscow Technical University of Communications and Informatics, Moscow, Russia

Surpin V.P., Ph.D. tech. sciences, Sberbank Moscow, Russia

Gorodnichev M.G., Ph.D. tech. sciences, associate professor, Dean of Information Technologies Faculty, Head of Mathematical Cybernetics and Information Technologies Department, Technical University of Communications and Informatics, Moscow, Russia

Цифровизация бизнеса, особенно в таких динамичных и конкурентных сферах, как финтех, реклама и телеком, привела к необходимости обрабатывать огромные объемы гетерогенных данных, поступающих от независимых поставщиков. Практическая невозможность строгой координации обмена данными и потребность принимать данные «как есть» обусловили преобладание ELT-подхода и гибких data lake в первичных слоях, отодвинув традиционный ETL и строгие реляционные БД ближе к слою витрин данных. При этом привычная реляционная модель, SQL-семантика и потребность в ACID-гарантиях привели к появлению открытых табличных форматов и архитектуры lakehouse. В работе рассматривается применение Open Table Formats, типичных для lakehouse архитектуры, в аналитической платформе финансовой организации. Приводятся результаты двух экспериментов: первый эксперимент моделирует сопровождение витрины данных на таблице 1 ТБ (~10 млрд ключей) с последовательными SELECT/UPDATE по 10–50% записей. Сравниваются партицированный Parquet, Iceberg (copy-on-write и merge-on-read) и Paimon (merge-on-read).

Исследование предоставляет практические рекомендации по выбору табличного формата для lakehouse-архитектур с учетом профиля нагрузки и операционных требований.

Business digitalization, particularly in dynamic and competitive sectors such as fintech, advertising, and telecommunications, has led to the necessity of processing massive volumes of heterogeneous data from independent providers. The practical impossibility of strict data exchange coordination and the need to accept data "as is" have resulted in the predominance of the ELT approach and flexible data lakes in primary layers, pushing traditional ETL and strict relational databases closer to the data mart layer. At the same time, the familiar relational model, SQL semantics, and the need for ACID guarantees have led to the emergence of open table formats and lakehouse architecture. This paper examines the application of Open Table Formats, typical for lakehouse architecture, in the analytical platform of a financial organization. The results of two experiments are presented: the first experiment models data mart maintenance on a 1 TB table (~10 billion keys) with sequential SELECT/UPDATE operations on 10–50% of records. Partitioned Parquet, Iceberg (copy-on-write and merge-on-read), and Paimon (merge-on-read) are compared.

The study provides practical recommendations for selecting a table format for lakehouse architectures, taking into account workload profiles and operational requirements.

Ключевые слова: Apache Paimon, Apache Iceberg, data lakehouse, OLAP, copy-on-write, merge-on-read.

Для цитирования: Иевлев К.О., Сурпин В.П., Городничев М.Г. Открытые табличные форматы в банковской аналитике: практическое исследование Iceberg и Paimon // Информационно-экономические аспекты стандартизации и технического регулирования. 2026. № 1(88). С. 84–89.

Keywords: Apache Paimon, Apache Iceberg, data lakehouse, OLAP, copy-on-write, merge-on-read.

For citation: Ievlev K., Surpin V., Gorodnichev M. Open Table Formats In Banking Analytics: A Practical Study Of Iceberg And Paimon. Information and Economic Aspects of Standardization and Technical Regulation. 2026; 1(88): 84–89. (In Russ.).

ВВЕДЕНИЕ

Современный банк – это сотни информационных систем, обслуживающих операционную деятельность разных направлений. Исторически отчетное хранилище, ориентированное на ретроспективные данные, эволюционировало в интеграционную платформу и де-факто стало участником операционных процессов, требующих обработки данных из нескольких автоматизированных систем. Это повышает требования к оперативности и гибкости интеграции, для чего удобны открытые табличные форматы. При этом организации не стремятся радикально перебирать инфраструктуру с нуля, предпочитая поэтапную модернизацию и интеграцию с уже существующими решениями [1, 2].

В 2016–2019 годах были анонсированы Apache Hudi, Delta Lake и Apache Iceberg – открытые табличные форматы, добавляющие транзакционность и управляемость поверх файлов (обычно в форматах Parquet и ORC) в распределенных хранилищах, таких как S3 и HDFS. У всех проектов общее назначение: создание единого табличного слоя поверх «сырого» файлового формата в data lake. Все форматы поддерживают ACID на уровне таблиц, снэпшоты, эволюцию схемы таблиц, а также версионирование данных. В совокупности эти характеристики позволяют Lakehouse-подходу поддерживать широкий спектр корпоративных сценариев – от BI и ad hoc-аналитики до MLOps и потоковой нагрузки, сокращая совокупную стоимость владения и ускоряя получение инсайтов [3, 4].

1. ОБЗОР ОСОБЕННОСТЕЙ СОВРЕМЕННЫХ OTF

Несмотря на открытый исходный код и формальное пребывание под эгидой Linux Foundation, экосистема Delta во многом остается привязанной к коммерческим решениям компании Databricks: ключевые оптимизации и лучший пользовательский опыт доступны в их облачной платформе (интеграции с Unity Catalog, Photon, Delta Live Tables, продвинутые оптимизаторы и автоматизации). Это создает более заметную вендор-зависимость по сравнению с альтернативами.

Если сравнивать показатели Github репозиториев проектов Iceberg и Hudi по состоянию на ноябрь 2025 года, можно заметить, что Iceberg заметно лидирует по звездам и форкам (8233/2879 против 6021/2441), что указывает на более широкий интерес сообщества. Total commits у Iceberg выше (7336 vs 6574), несмотря на более поздний старт репозитория. У Iceberg меньше открытых issues при большем масштабе сообщества, что может говорить о лучшей пропускной способности по устранению багов.

Таблица 1

Данные Github репозиториев Apache Hudi и Apache Iceberg за ноябрь 2025 г.

Движок	Hudi	Iceberg
total commits	6805	7714
stars	6021	8223
forks	2441	2879
open	1152	594
Открытых pull requests	430	187
watchers	6021	8233
Опубликован в открытом доступе	2016	2017

Исходя из собранной информации было принято решение сосредоточиться в данном исследовании именно на Apache Iceberg.

В 2022 г. в отдельный проект был выделен Apache Paimon – унифицированный формат хранения данных для архитектуры Lakehouse, разработанный для обеспечения высокой пропускной способности при обновлении данных (streaming updates) и низкой задержки при аналитических запросах [5, 6].

Как и во многих популярных решениях для bigdata аналитики, например RocksDB, ClickHouse и HBase, в основе архитектуры данного Open Table Formats (OTF) лежит структура LSM (Log-Structured Merge-tree), реализованная поверх распределенных файловых систем (HDFS, S3) [7, 8]. Paimon также позволяет выбирать строковый или колоночный формат по уровням LSM [9], что открывает путь к HTAP (Hybrid transactional/analytical processing) сценариям.

Мы не сравниваем Paimon и Iceberg по GitHub-метрикам напрямую, т.к. Paimon выделился в отдельный проект сравнительно недавно.

2. АРХИТЕКТУРНЫЕ ХАРАКТЕРИСТИКИ ICEBERG И PAIMON.

Недостатки хранения таблиц в классическом data lake

Для исследования преимуществ lakehouse архитектуры сравним его с обработкой данных с классическим datalake на основе экосистемы Hadoop: при необходимости обновления табличных данных (чаще всего используется Hive и Hive Metastore) наиболее распространенным решением является полная перезапись таблицы или отдельной партиции [10]. У этого подхода есть ограничения:

– при большом количестве файлов требуются значительные вычислительные и I/O-ресурсы;

– во время перезаписи читатели могут получать ошибки или наблюдать неконсистентные состояния данных из-за отсутствия изоляции на уровне версий.

Открытые табличные форматы, такие как Apache Iceberg и Apache Paimon, вводят абстракцию «снимка состояния» (snapshot) данных. Снимок – это метаданные, указывающие на конкретный набор неизменяемых файлов с данными. При обновлениях заменяется лишь подмножество файлов, а метаданные атомарно переключаются на новый снимок [10]. Таким образом обеспечивается атомарная смена версии и мгновенная согласованность для читателей: потребитель читает строго определенный снимок, не наблюдая частично примененных изменений. В результате уменьшается объем перезаписываемых данных по сравнению с «полной перезаписью», что позитивно влияет на скорость выполнения запросов и потребление ресурсов при выполнении клиентских задач. При этом интеграция для существующих пайплайнов, как правило, прозрачна: операции чтения/записи получают выигрыш в производительности и функциональности без существенных изменений пользовательского кода [3, 10].

ОТЛИЧИЯ В МЕХАНИКЕ ОБНОВЛЕНИЙ ДАННЫХ В ICEBERG И PAIMON

Отличия Iceberg и Paimon в значительной степени определяются механизмами фиксации изменений и организации хранения. В Iceberg поддерживаются два режима работы с построчными изменениями: copy-on-write (COW) и merge-on-read (MOR). В MOR-режиме обновления и удаления записываются через «delete-файлы» (position deletes и/или equality deletes), а новые/замещающие строки – в новые файлы данных; читатели применяют удаляющие маски при чтении соответствующего снимка. В COW-режиме файлы, содержащие измененные строки, переупаковываются целиком. Оба подхода фиксируются как новые снимки метаданных и обеспечивают моментальную смену версии для читателей.

В Paimon применяется иная организация хранения – при наличии первичного ключа обновление представляется как upsert: в следующий снимок добавляется новая версия ключа без необходимости чтения прежнего состояния [11]. Консолидация и упорядочивание версий выполняются через фоновые компакции. В типичных сценариях такой подход снижает стоимость записи и задержку по сравнению с перепаковкой файлов или широким применением delete-файлов. Сравнение MOR реализации обновления данных в Iceberg и Paimon представлена на рис. 1.

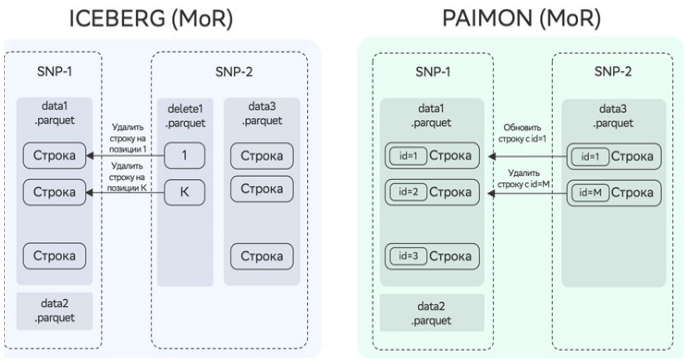

Рис. 1. Обновление данных в Iceberg и Paimon

3. МЕТОДОЛОГИЯ ЭКСПЕРИМЕНТА

Проводился эксперимент сравнения скорости выполнения различных операций на движке Iceberg и Paimon. Для тестирования сравнительной производительности работы с данными в файловых форматах Parquet, Iceberg и Paimon используется подход, состоящий в генерации 1Tb входных данных и последовательном выполнении операций select и update на них.

Данная схема представляет собой упрощенный процесс построения витрин данных и позволяет продемонстрировать различия в подходах к хранению данных в перечисленных выше форматах и на практике показать закономерности работы табличных форматов, которые необходимо учитывать при выборе того или иного формата.

Для экспериментов используются следующие наборы данных (табл. 2):

Таблица 2

Список таблиц для проведения эксперимента

Название таблицы	Описание
parquet_10b_part	«Партицированный parquet» в варианте 1Tb
parquet_10b_cow	«Партицированный parquet» в варианте copy-on-write 1Tb
parquet_10b_mor	«Партицированный parquet» в варианте merge-on-read 1Tb
parquet_10b_part	«Партицированный parquet» в варианте merge-on-read 1Tb

Все наборы данных состоят из одной таблицы следующего формата (табл. 3):

Таблица 3

Описание данных в тестовых таблицах

Название поля	Тип поля	Описание
id	bigint	int значение от 0 до 10 миллиардов
id_str	string	string формат значения «id_» + число от 1 до 10 миллиардов, выравненное (padded) слева нулями до фиксированной длины 11 символов
data	string	случайные ASCII символы (строка 100 символов)
dt	string	текущий datetime (string) в формате «%Y-%m-%d %H:%M:%S»
part	int	int значение от 0 до 10000

В табл. 4 дается описание аппаратного обеспечения кластера, использующегося для проведения исследования.

Таблица 4

Описание аппаратного обеспечения кластера

Параметр	Значение
CPU	Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz
RAM	768 GB (32 Gb x 24) DDR4 2400MHz
Жесткие диски	6Tb SATA III 7200rpm 128Mb
Количество рабочих узов, где запускаются задачи обработки данных	4

Информация о версиях компонентов, использующихся в ходе исследования, представлена в табл. 5.

Таблица 5

Версии программного обеспечения на тестовом кластере

ПО	Версия
Apache Parquet	1.13.1
Apache Iceberg	1.6.1
Apache Paimon	1.1.1
Apache Spark	3.5.1
Apache Flink	1.18

В ходе эксперимента осуществлялись следующие операции:

- Выполнить чтение 10% данных с использованием фильтрации по текстовой копии идентификатора
- select count(*) from {table} where id_str < 'id_01000000000'
- Выполнить обновление 10% данных с использованием фильтрации по текстовой копии идентификатора
- update {table} set dt = {current} where id_str < 'id_01000000000' where id % 3 = 0
- Выполнить чтение ранее обновленных 10% данных с использованием фильтрации по текстовой копии идентификатора
- select count(*) from {table} where id_str < 'id_01000000000'
- Выполнить обновление 20% данных, при этом первые 10% будут обновлены два раза update {table} set dt = {current} where id_str < 'id_02000000000' where id % 3 = 1
- Выполнить чтение 10% данных, которые были обновлены дважды select count(*) from {table} where id_str < 'id_02000000000'
- Выполнить обновление 50% данных, при этом первые 10% будут обновлены три раза, а первые 20% - два раза update {table} set dt = {current} where id_str < 'id_05000000000'
- Выполнить чтение 10% данных, которые были обновлены трижды select count(*) from {table} where id_str < 'id_02000000000' where id %3 = 2
- Операция %3 используется для того, чтобы избежать перезаписи файлов с данными целиком.

4. РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТА

Результаты эксперимента описаны в табл. 6.

Таблица 6

Результаты описанного эксперимента

Операция	Тип	Parquet part	Iceberg (COW)	Iceberg (MOR)	Paimon
1. SELECT 10%	Чтение	00:57	00:33	00:47	00:37
2. UPDATE 10%	Запись	06:00	02:18	02:58	03:22
3. SELECT 10%	Чтение	00:57	00:33	00:45	00:37
2. UPDATE 20%	Запись	12:00	03:54	05:22	06:00
5. SELECT 10%	Чтение	00:57	00:34	00:47	00:35
6. UPDATE 50%	Запись	30:00	07:39	11:28	14:28
7. SELECT 10%	Чтение	00:57	00:35	00:50	00:36

Изучив полученные результаты можно сделать вывод, что простая линейная модель (1) с высокой точностью объясняет наблюдаемое время записи и предоставляет практически применимые прогнозы и пороговые значения эффективности для сравнения Parquet, Iceberg (режимы COW/MOR) и Paimon при выполнении пакетных обновлений (UPDATE) таблицы объемом 1 ТБ в рамках указанной конфигурации кластера:

$$T\,write = a + b \cdot p, \qquad (1)$$

где $T\,write$ – полное время записи на данном шаге (в минутах); p [в процентах от таблицы объемом 1 ТБ] – доля строк, обновленных на данном шаге; a – накладные расходы уровня движка и системы (минуты, фиксированные накладные расходы), которые не масштабируются вместе с p; а также учитывает планирование и инициализацию задания (например, планирование в Spark), операции с метаданными и фиксацией транзакций (snapshot/commit), а также минимальный ввод-вывод, необходимый даже для очень небольших обновлений. В пределе при $p \to 0$, $T\,write \approx a$; b [минуты на процентный пункт] – предельная стоимость за каждый дополнительный 1% обновленных данных. Этот член отражает масштабируемую часть работы: создание/перезапись файлов, генерацию и обработку файлов удалений (для режимов MOR), перемешивание (shuffle) и сортировку, а также бакетирование в Paimon. Меньшее значение b указывает на более дешевое масштабирование обновлений.

Визуализация полученной математической модели представлена на рис. 2.

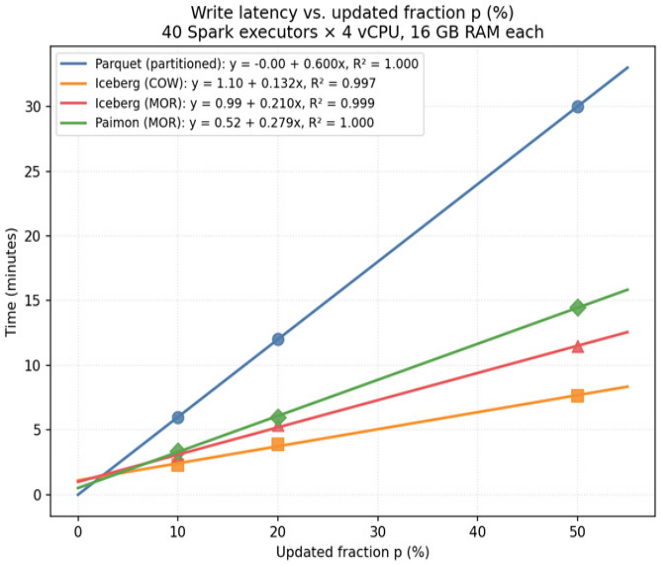

Рис. 2. Линейная аппроксимация результатов тестирования

Линейные модели обновлений демонстрируют очень высокую объясняющую способность ($R^2 \geqslant 0.995$) для всех форматов. Это указывает на то, что в исследованных масштабах время записи на каждом шаге хорошо аппроксимируется суммой фиксированных накладных расходов и слагаемого, линейно зависящего от доли обновленных данных.

Эмпирические оценки в нашей конфигурации (40 экзекьюторов Spark; по 4 vCPU и 16 ГБ RAM на каждого) иллюстрируют распределение этих ролей:

– Parquet (партиционированный): $a \approx 0, b \approx 36.0$ ($R^2 = 1.000$)
Интерпретация: стоимость перезаписи чисто пропорциональна объему.

– Iceberg (COW): $a \approx 57.8, b \approx 8.03$ ($R^2 \approx 0.995$)
Интерпретация: значительные фиксированные накладные расходы (планирование/коммит), низкая предельная стоимость за каждый процент, так как перезаписываются только файлы с затронутыми данными.

– Iceberg (MOR): $a \approx 50.5, b \approx 12.75$ ($R^2 \approx 0.998$)
Интерпретация: фиксированные накладные расходы ниже, чем у COW, но более высокая предельная стоимость (файлы удалений + последующее слияние).

– Paimon (MOR): $a \approx 35.5, b \approx 16.65$ ($R^2 \approx 0.999$)
Интерпретация: малые фиксированные накладные расходы; более высокая предельная стоимость из-за перемешивания (shuffle)/бакетирования и последующих компакций.

ВЫВОДЫ

В проведенных экспериментах оба OTF превзошли партицированную parquet-таблицу как по скорости обновления данных, так и по чтению.

Объяснение результата: Ускорение в чтении достигается за счет использования фильтрации по статистикам из файлов с метаданными в случае Iceberg и Paimon. В случае Parquet приходится прочитать метаданные (footer) всех файлов для осуществления фильтрации по условию id_str < 10%.

Iceberg в режиме MOR работает дольше как на чтение (предсказуемо), так и на запись. Последнее требует дополнительного изучения.

Предположение о причинах замедления – накладные расходы на создание и коммит большого количества файлов удалений (position deletes) в данной конфигурации превзошли затраты на последовательную перезапись данных в режиме COW. Замедление в обоих случаях пропорционально объему измененных данных.

Paimon оказался медленнее Iceberg COW как при записи (существенно), так и при чтении (пропорционально объему измененных данных). При этом работает быстрее Iceberg MOR при чтении.

В случае записи замедление объясняется дополнительным shuffle, используемым для сортировки и бакетирования данных.

Список литературы / References

1. Турсынбаева С.Ж. Особенности построения хранилищ данных в финансовых организациях // Молодой ученый. 2021. № 16(358). С. 14–19 / Tursynbayeva, S. Features of Data Warehouse Design in Financial Organizations, Molodoy Uchenyy, 2021, No. 16 (358), pp. 14–19.
2. Ермаков С.Г., Мандрика О.С., Патеюк Д.С. Архитектура данных на основе Lakehouse: современные подходы к построению модели управления образовательных организаций на основе анализа данных // ИВД. 2025. № 6 (126). С. 844–856 / Ermakov S., Mandrika O., Pateyuk D. Lakehouse-Based Data Architecture: Modern Approaches to Building a Data-Driven Management Model for Educational Organizations. IVD, 2025, No. 6 (126), pp. 844–856.
3. Armbrust M., Ghodsi A., Xin R., Zaharia M. Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics // 11th Annual Conference on Innovative Data Systems Research (CIDR '21). 2021. С. 1–8.
4. Alleni R. AI/ML optimized lakehouse architecture: A Comprehensive framework for modern data science // World Journal of Advanced Engineering Technology and Sciences. 2025. С. 2099–2104.
5. Apache Software Foundation Announces New Top-Level Project Apache Paimon // The Apache Software Foundation Blog. [Электронный ресурс]. Режим доступа: https://news.apache.org/foundation/entry/apache-software-foundation-announces-new-top-level-project-apache-paimon. (дата обращения: 01.11.2025).
6. Apache Paimon // Apache Paimon Documentation. [Электронный ресурс]. Режим доступа: https://paimon.apache.org/docs/1.1. (дата обращения: 01.11.2025).
7. Understand Files // Apache Paimon Documentation. [Электронный ресурс]. Режим доступа: https://paimon.apache.org/docs/1.1/learn-paimon/understand-files. (дата обращения: 01.11.2025).
8. Mishra S.A survey of LSM-Tree based Indexes, Data Systems and KV-stores // IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS). 2024. Vol. 1. С. 1–6.
9. Configurations // Apache Paimon Documentation. [Электронный ресурс]. Режим доступа: https://paimon.apache.org/docs/master/maintenance/configurations (дата обращения: 01.11.2025).
10. Shiran T.T., Hughes J., Merced A. Apache Iceberg: The Definitive Guide. O'Reilly, 2024.
11. Chaudhar V., Charate P.A. Optimizing Data Lakehouse Architectures for Scalable Real-Time Analytics // International Journal of Scientific Research in Science Engineering and Technology. 2025. Vol. 12. С. 809–822.