УДК 004.042:004.65
DOI: 10.26102/2310-6018/2026.59.8.003

Инкрементальный метод обновления многомерного куба по неупорядоченному потоку событий журналов информационных систем

Г.А. Уфимцев[1]✉, С.В. Зыков[1,2]

[1]МИРЭА – Российский технологический университет, Москва, Российская Федерация
[2]Национальный исследовательский университет «Высшая школа экономики», Москва, Российская Федерация

Резюме. Информационные системы формируют большие объёмы событийных журналов, которые используются для анализа работы приложений и сервисов. При этом события могут поступать в аналитический контур позже момента их фактического возникновения и не в исходном порядке. Такая рассинхронизация приводит к ошибкам при построении агрегированных временных показателей, а регулярный полный пересчёт многомерного аналитического куба требует значительных вычислительных затрат. Целью работы является разработка подхода к обновлению многомерного куба по потоку журнальных событий с учётом задержек и нарушения порядка их поступления. Ведущим подходом является разделение итогового представления куба на базовый куб и компенсационный слой. Своевременно поступающие события обновляют базовый куб, а запаздывающие события в пределах заданного горизонта компенсации вносят поправки в компенсационный слой. Для исключения повторного учёта событий используется проверка уникальных идентификаторов. В работе представлена формальная модель события, описаны правила отнесения событий к ячейкам куба по времени возникновения, предложен алгоритм инкрементального обновления и проведён вычислительный эксперимент на программном прототипе. Результаты эксперимента показали, что предложенный метод снижает ошибку агрегатов по сравнению с оконным методом и требует меньшего объёма повторной обработки данных по сравнению с пакетным пересчётом. Материалы статьи представляют практическую ценность для разработки аналитических контуров мониторинга, аудита, анализа событий безопасности и пользовательской активности.

Ключевые слова: инкрементальное обновление, OLAP-куб, событийный журнал, поток событий, неупорядоченное поступление данных, запаздывающие события, компенсационный слой, потоковая обработка данных.

Для цитирования: Уфимцев Г.А., Зыков С.В. Инкрементальный метод обновления многомерного куба по неупорядоченному потоку событий журналов информационных систем. *Моделирование, оптимизация и информационные технологии.* 2026;14(8). URL: https://moitvivt.ru/ru/journal/article?id=2404 DOI: 10.26102/2310-6018/2026.59.8.003

An incremental method for updating a multidimensional cube based on a disordered stream of information system log events

G.A. Ufimtsev[1]✉, S.V. Zykov[1,2]

[1]MIREA – Russian Technological University, Moscow, the Russian Federation
[2]National Research University Higher School of Economics, Moscow, the Russian Federation

Abstract. Information systems generate large volumes of event logs, which are used to analyze the operation of applications and services. In this case, events may arrive in the analytical circuit later than the moment of their actual occurrence and not in the original order. Such time inconsistency leads to errors in the construction of aggregated time-based indicators, while regular full recalculation of a

© Уфимцев Г.А., Зыков С.В., 2026

multidimensional analytical cube requires significant computational costs. The purpose of the work is to develop an approach to updating a multidimensional cube based on the flow of log events, taking into account delays and violations of the order in which they arrive. The leading approach is to divide the final cube representation into a base cube and a compensation layer. Timely events update the base cube, while delayed events within a given compensation horizon add corrections to the compensation layer. Unique event identifiers are checked to exclude repeated event accounting. The paper presents a formal event model, describes rules for assigning events to cube cells by the occurrence time, proposes an incremental update algorithm, and provides a computational experiment on a software prototype. The experimental results show that the proposed method reduces aggregate errors compared with the window-based method and requires a smaller amount of repeated data processing compared with batch recalculation. The materials of the paper are of practical value for developing analytical pipelines for monitoring, audit, security event analysis, and user activity analysis.

Keywords: incremental updating, OLAP cube, event log, event flow, disordered data arrival, delayed events, compensation layer, stream data processing.

For citation: Ufimtsev G.A., Zykov S.V. An incremental method for updating a multidimensional cube based on a disordered stream of information system log events. *Modeling, Optimization and Information Technology.* 2026;14(8). (In Russ.). 2026;14(8). URL: https://moitvivt.ru/ru/journal/article?id=2404 DOI: 10.26102/2310-6018/2026.59.8.003

Введение

Современные информационные системы (ИС) генерируют значительные объёмы событийных данных, которые представлены в виде событийных журналов приложений. Собираемые данные демонстрируют динамику работы ИС и позволяют получить информацию о действиях пользователей, обращениях к программным компонентам, ошибках, предупреждениях, изменениях состояния объектов и т. д. При этом непосредственный анализ журналов затруднён из-за высокой детализации записей, неоднородности форматов, наличия технических полей и большого объёма поступающих данных. Также сложность анализа увеличивает распределенность современных информационных систем, так как это приводит к необходимости работы с множеством источников данных.

Одним из подходов к аналитической обработке событийных данных является их преобразование в многомерные структуры данных – OLAP-кубы. OLAP-куб позволяет агрегировать события, например, по времени, типу события, источнику, компоненту информационной системы или уровню критичности. Благодаря этому при работе с данными упрощается процесс отслеживания динамики показателей и сравнения состояний системы за различные интервалы времени.

В классических сценариях использования OLAP-кубов данные предварительно загружаются в хранилище, после чего выполняется построение или обновление агрегатов. Такой подход хорошо подходит для пакетной обработки, но становится менее эффективным при работе с потоками событий. В реальных информационных системах события могут поступать в аналитический контур не в том порядке, в котором они фактически возникли. Эта ситуация может возникнуть по разным причинам, среди которых могут быть сетевые задержки, буферизация сообщений, асинхронная интеграция компонентов и повторная доставка сообщений. Если при работе с данными использовать только время поступления события в аналитический контур, то событие, поступившее в обработку значительно позже возникновения в ИС, может быть ошибочно отнесено к другому интервалу.

В исследованиях, посвящённых OLAP (Online Analytical Processing) и многомерным моделям данных, основное внимание уделяется организации, хранению и использованию многомерных структур для аналитической обработки. Так, В.А. Фролов,

Г.И. Хайруллин и Р.З. Афанасьев анализируют форматы хранения многомерных моделей данных в контексте современных аналитических систем [1]. Е.О. Неупокоева отмечает, что предварительный расчёт агрегаций в MOLAP (Multidimensional Online Analytical Processing) обеспечивает быстрые ответы на запросы, но пересчёт куба при изменении данных может требовать значительных затрат [2]. Авторы рассматривают OLAP-куб как способ организации данных, при котором факты анализируются по нескольким независимым измерениям. Однако в работе не рассматривается задача обновления OLAP-куба по потоку событий, поступающих с нарушением порядка их фактического возникновения.

Проблема различия между временем возникновения события и временем его обработки подробно рассматривается в области потоковой обработки данных. В работе T. Akidau и соавторов рассматривается механизм водяных знаков, который используется как способ оценки временной полноты бесконечного потока данных и как средство работы с неупорядоченными событиями [3]. Обзор M. Fragkoulis и соавторов показывает, что обработка неупорядоченных данных, управление состоянием и отказоустойчивость являются ключевыми направлениями развития систем потоковой обработки [4]. Однако данные подходы ориентированы прежде всего на потоковые операторы, оконные вычисления и механизмы исполнения потоковых запросов, тогда как задача сопровождения OLAP-куба как многомерной аналитической структуры требует отдельного рассмотрения.

Вопросы инкрементального обновления агрегированных данных также изучаются в области хранилищ данных и материализованных представлений. Инкрементальное сопровождение позволяет обновлять результат вычисления на основе изменений исходных данных без полного пересчёта. В работе C. Svingos и соавторов отмечается, что инкрементальное обновление представлений уменьшает время приведения материализованного представления к актуальному состоянию за счёт применения только изменений базовых таблиц [5]. В работе A. Cuzzocrea и соавторов рассматривается сопровождение хранилища данных в near-real-time (NRT) OLAP-сценариях и подчёркивается проблема устаревания предварительно рассчитанных агрегатов при изменении данных [6]. В рассмотренных исследованиях авторы основное внимание уделяют обновлению таблиц и представлений в хранилищах данных, а не обработке журнальных событий, поступающих с нарушением порядка возникновения.

Существующие исследования формируют основу для настоящей работы, но не закрывают задачу совместного учёта трёх факторов: многомерной структуры OLAP-куба, неупорядоченного поступления событий по отношению к времени их возникновения и необходимости инкрементального обновления агрегатов без полного пересчёта. Для решения этой задачи в статье предлагается метод обновления OLAP-куба по потоку событий журналов информационных систем. Метод основан на отнесении событий к ячейкам куба по времени возникновения, использовании компенсационного слоя для запаздывающих событий и идемпотентной обработке повторно поступающих событий.

Целью работы является разработка инкрементального метода обновления OLAP-куба по потоку событий журналов информационных систем, обеспечивающего корректное отнесение событий к временным ячейкам по времени их возникновения при неупорядоченном, запаздывающем и повторном поступлении данных.

Материалы и методы

Рассмотрим поток событий журналов информационной системы как:

$$S = \{e_1, e_2, \dots, e_i, \dots\}. \quad (1)$$

Поток S рассматривается как потенциально неограниченная последовательность событий, порядок поступления которых в аналитический контур может не совпадать с порядком их фактического возникновения. Каждое событие e_i задаётся кортежем:

$$e_i = (id_i, x_i, \tau_i^e, \tau_i^a, v_i), \quad (2)$$

где id_i – уникальный идентификатор события; $x_i = (x_{i1}, x_{i2}, .., x_{ip})$ – вектор значений измерений; τ_i^e – время события, то есть момент фактического возникновения события в системе; τ_i^a – время поступления события в аналитический контур, $v_i = (v_{i1}, v_{i2}, .., v_{iq})$ – вектор мер. В качестве измерений могут использоваться, например, источник события, тип события, компонент информационной системы, уровень критичности, пользователь, сервис или иные признаки, доступные в журнале. В качестве мер могут выступать количество событий, длительность операции, объём переданных данных, значение технического показателя или другая числовая характеристика события.

Для каждого события определим задержку поступления как:

$$\delta_i = \tau_i^a - \tau_i^e, \ \delta(e) \geq 0. \quad (3)$$

Задержка поступления характеризует расхождение между фактическим временем возникновения события и моментом его доступности для аналитической обработки. В предлагаемой модели этот параметр используется для классификации событий на своевременные и запаздывающие. Исходя из времени задержки события можно будет сделать вывод, что если δ_i мала, событие пришло почти вовремя, если δ_i велика, событие считается запаздывающим.

Для того, чтобы на основе событий (2) строить OLAP-куб по времени, введем функцию (4) дискретизации времени:

$$w(\tau_i^e) = \frac{\tau_i^e}{\Delta}, \quad (4)$$

где Δ – шаг временной агрегации, например 1 минута, 1 час, 1 день.

Тогда ячейка куба, в которую попадает событие e, определяется функцией:

$$g(e_i) = (x_i, w(\tau_i^e)). \quad (5)$$

Функция (5) демонстрирует, что событие относится к ячейке по времени возникновения, а не по времени поступления. Благодаря такому распределению событий по ячейкам куба, мы получаем верную аналитику работы приложений по времени. Такой подход согласуется с event-time семантикой в потоковой обработке [3, 7].

Обозначим множество всех ячеек куба как K. Для каждой ячейки $k \in K$ храним агрегаты. Для того, чтобы модель была универсальна при работе с любым форматом логов и набором агрегатов, представим агрегаты одной векторной функцией:

$$A(k) = (A_1(k), A_2(k), .., A_r(k)). \quad (6)$$

Пусть $C^*(T)$ – куб, который был бы получен к моменту T, если бы у нас были все события, упорядоченные по τ_e, то есть без потерь и с полным знанием истории. А $C^{obs}(T)$ – куб, который реально построен к моменту T на основе только тех событий, для которых $\tau_a \leq T$. В реальности, при работе с потоковыми данными будет возникать проблема – $C^*(T) \neq C^{obs}(T)$. Разность возникает из-за событий, которые уже произошли, но ещё не поступили в обработку.

Для решения этой проблемы в рамках статьи предлагается использовать инкрементальный метод. В его основе лежит концепция, что мы не храним только один куб, вместо этого определяем:

– C^{base} – базовый куб, который быстро обновляется при поступлении событий;

– C^{Δ} – компенсационный слой, в который вносятся поправки для запаздывающих событий;

– C^{res} – итоговый куб результата.

Связь между этими кубами можно в формализованном виде представить в виде формулы (7):

$$C^{res} = C^{base} + C^{\Delta}. \quad (7)$$

Итоговый куб не обязательно хранится как отдельная физическая структура. Он может формироваться логически при выполнении аналитического запроса путём объединения базового куба и компенсационного слоя.

Под сложением в этом случае понимается покомпонентное сочетание по всем ячейкам и всем мерам. Такое разделение согласуется с общими принципами event-time обработки и инкрементального сопровождения аналитических представлений: в потоковых системах различие между временем события и временем обработки принципиально, а в OLAP и хранилищах данных инкрементальное обновление представлений рассматривается как ключевой механизм сохранения аналитической актуальности без полного пересчёта [3, 8]. В предлагаемом методе компенсационный слой хранит дельта-изменения агрегатов для тех ячеек куба, которые относятся к уже сформированным временным интервалам, но были затронуты запаздывающими событиями. При выполнении условия закрытия временного окна поправки из компенсационного слоя материализуются в базовый куб, после чего соответствующие записи компенсационного слоя могут быть удалены.

Чтобы решить, когда событие надо учитывать, как обычное (то есть помещать его в базовый куб), а когда как запаздывающее (то есть помещать его в компенсационный слой), введём порог допустимой задержки λ. Событие e считаем запаздывающим, если $\delta(e) > \lambda$, где $\lambda \geq 0$. Значение порога λ может задаваться одним из двух способов. Первый способ – экспертный, когда значение выбирается на основе требований к актуальности аналитики и особенностей информационной системы. Второй способ – статистический, когда λ определяется на основе накопленной истории задержек поступления событий.

Введем правила обновления кубов инкрементального метода (его агрегатов). Если $\delta(e) \leq \lambda$, то обновление можно представить в виде формулы:

$$A^{base}(k) \leftarrow A^{base}(k) + d(e_i), \quad (8)$$

где $A^{base}(k)$ – вектор агрегатов базового куба для ячейки k; $d(e_i)$ – функция, обозначающая влияние конкретного события на агрегаты.

Если $\delta_i > \lambda$, то обновление в этом случае представляем как:

$$A^{\Delta}(k) \leftarrow A^{\Delta}(k) + d(e_i), \quad (9)$$

где $A^{\Delta}(k)$ – вектор агрегатов компенсационного слоя для ячейки k; $d(e_i)$ – функция, обозначающая влияние конкретного события на агрегаты.

При передаче потоковых данных может возникнуть ситуация дублирования события. Причин возникновения этой ситуации может быть несколько: сбои на стороне поставщика событий, сетевые проблемы, ошибки на стороне потребителя [9]. Если одно и то же событие пришло дважды, повторная обработка одного и того же события приводит к повторному изменению агрегатов и нарушает корректность итогового куба. Поэтому надо явно ввести множество уже учтённых идентификаторов: $I \subseteq \{id\}$. Обновление выполняется только если $id \notin I$, после чего: $I \leftarrow I \cup \{id\}$. Благодаря добавлению в формализованную модель метода учета уже пришедших событий достигается идемпотентность – повторная обработка того же события не меняет результат. На практике множество I не должно храниться неограниченно долго.

Горизонт хранения идентификаторов может быть связан с максимально допустимой задержкой событий и периодом возможной повторной доставки. После истечения заданного горизонта старые идентификаторы могут быть удалены из I, если соответствующие временные интервалы считаются закрытыми.

Компенсационный слой C^{Δ} предназначен для временного хранения поправок, вызванных запаздывающими событиями. Если хранить его без ограничений, объём данных будет постоянно расти. Поэтому в методе необходимо задать правило материализации компенсационного слоя. Под материализацией будем понимать перенос накопленных поправок из компенсационного слоя в базовый куб. Поправки из компенсационного слоя переносятся в базовый куб для тех ячеек, у которых истёк горизонт ожидания поздних событий. Под горизонтом будем понимать интервал времени, в течение которого система допускает поступление запаздывающих событий, относящихся к этой ячейке.

На основе предложенной формальной модели был составлен алгоритм инкрементального обновления куба при получении нового события. В качестве входных данных для алгоритма выступают: событие, базовый куб, компенсационный слой, множество обработанных идентификаторов, порог допустимой задержки, шаг временной агрегации. Результатами работы алгоритма являются: обновлённое состояние базового куба, компенсационного слоя и множества обработанных идентификаторов. Алгоритм представлен в виде блок-схемы на Рисунке 1.

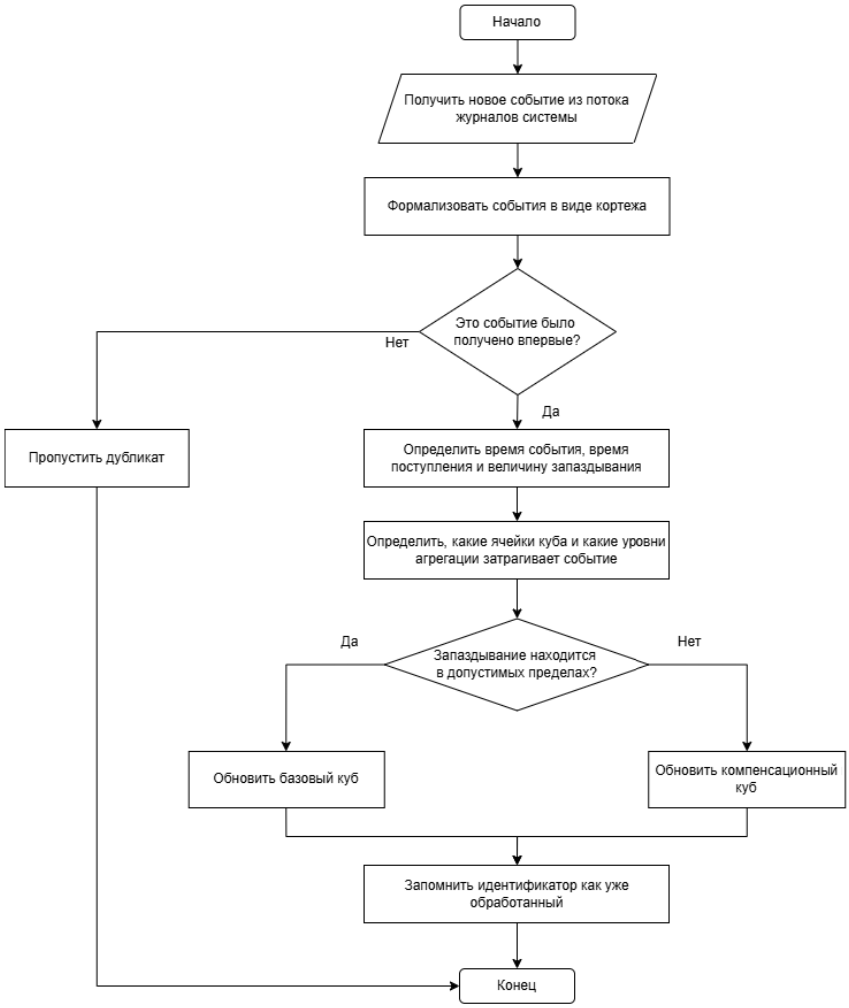

Рисунок 1 – Алгоритм инкрементального обновления куба при получении нового события
Figure 1 – The algorithm for incrementally updating the cube when a new event is received

Алгоритм имеет инкрементальный характер, поскольку при поступлении события изменяется не весь куб, а только одна или несколько ячеек, которые соответствуют значениям измерений и временному интервалу события. Это позволяет снизить объём пересчёта по сравнению с полным обновлением куба.

Результаты

Для проверки предложенного метода и сравнения его с аналогами был произведен вычислительный эксперимент на программном прототипе, реализующем логическую модель OLAP-куба. Прототип был разработан на языке Python, для генерации экспериментальных данных и обработки результатов применялись библиотеки numpy и pandas. В рамках прототипа куб представлялся как набор агрегированных ячеек, каждая из которых определялась сочетанием временного интервала, компонента информационной системы, типа события и уровня критичности. В качестве агрегатов для каждой ячейки прототипа куба использовались sum и count [10].

Целью эксперимента являлось сравнение и оценка свойств предложенного метода. В рамках эксперимента не производилось сравнение производительности конкретных промышленных OLAP-хранилищ. Использование программного прототипа позволило изолированно проверить алгоритмическую часть метода без влияния особенностей конкретной СУБД, индексов, кэширования и аппаратной платформы.

В эксперименте моделировался поток событий журналов информационной системы. Каждое событие содержало идентификатор, время возникновения, время поступления, компонент информационной системы, тип события, уровень критичности и числовую меру. Для моделирования поступления с нарушением порядка и запаздывания событий время поступления могло отличаться от времени возникновения. Также в поток добавлялись дополнительные события, имитирующие повторную доставку сообщений.

Для валидации работы методов использовалось сравнение с эталонным OLAP-кубом. Под эталонным кубом в рамках эксперимента понимается построенный заранее куб по полному набору уникальных событий. Каждое событие из экспериментального набора данных было отнесено к ячейке эталонного куба по времени возникновения.

В эксперименте сравнивалось три метода: пакетный перерасчет куба, оконный метод, предлагаемый метод.

Метод пакетного пересчета куба основан на подходе, в рамках которого при поступлении очередного микробатча куб заново строится по всем уже поступившим событиям [11]. Оконный метод основан на учете только тех событий, которые поступили в пределах допустимого времени ожидания. Если событие поступило после закрытия временного окна, оно не влияет на ранее сформированный результат [3, 12].

Основные параметры эксперимента представлены в Таблице 1.

Таблица 1 – Параметры вычислительного эксперимента
Table 1 – Parameters of the computational experiment

Параметр	Значение
Число уникальных событий	20 000
Доля повторных событий	3 %
Размер потока с учётом дублей	20 600
Шаг временной агрегации	5 минут
Число компонентов информационной системы	20
Число типов событий	10

Таблица 1 (продолжение)
Table 1 (continued)

Число уровней критичности	4
Порог своевременного поступления события	5 минут
Горизонт компенсации в основном эксперименте	45 минут
Доля запаздывающих событий	0 %, 5 %, 10 %, 20 %, 30 %
Размер микробатча для полного пересчёта	100 событий

Для оценки методов использовался ряд метрик, которые позволят сравнить корректность и производительность методов:

– *Среднее время обработки события* показывает средние вычислительные затраты на обработку одного элемента потока. В рамках эксперимента эта метрика используется только для сравнения методов внутри одного программного прототипа.

– *Пересчитано ячеек на событие* показывает средний объём обновления куба. Для пакетного пересчёта метрика отражает среднее число ячеек, пересчитываемых при очередном обновлении. Для оконного и предлагаемого методов – среднее число ячеек, изменяемых при обработке одного события.

– *Потерянные события* – число уникальных событий, которые не повлияли на итоговый куб. Для оконного метода это события, поступившие после закрытия окна. Для предлагаемого метода это события, поступившие позже горизонта компенсации.

– *Ошибка count* показывает относительное отклонение количества событий в итоговом кубе от эталонного куба.

– *Ошибка sum* показывает относительное отклонение суммы числовой меры в итоговом кубе от эталонного куба.

– *Ошибочные ячейки* показывают долю ячеек итогового куба, которые отличаются от эталонного куба хотя бы по одному из агрегатов.

Результаты основного эксперимента представлены в Таблице 2.

Таблица 2 – Сравнение методов при различной доле запаздывающих событий
Table 2 – Comparison of methods for different proportions of delayed events

Метод	**Доля запаздывающих событий, %**	**Среднее время обработки события, мкс**	**Пересчитано ячеек на событие**	**Потерянные события**	**Ошибка count, %**	**Ошибка sum, %**	**Ошибочные ячейки, %**
Пакетный пересчёт	0	433,456	93,921	0	0,000	0,000	0,000
Оконный метод	0	60,804	0,971	0	0,000	0,000	0,000
Предлагаемый метод	0	59,165	0,971	0	0,000	0,000	0,000
Пакетный пересчёт	5	420,869	94,025	0	0,000	0,000	0,000
Оконный метод	5	56,729	0,923	992	4,960	5,014	5,305
Предлагаемый метод	5	59,895	0,961	208	1,040	1,045	1,114
Пакетный пересчёт	10	416,087	93,782	0	0,000	0,000	0,000
Оконный метод	10	56,447	0,870	2068	10,340	10,562	11,047

Таблица 2 (продолжение)
Table 2 (continued)

Предлагаемый метод	10	63,226	0,950	423	2,115	2,206	2,272
Пакетный пересчёт	20	421,519	94,038	0	0,000	0,000	0,000
Оконный метод	20	51,060	0,775	4033	20,165	20,242	21,200
Предлагаемый метод	20	62,528	0,933	787	3,935	3,882	4,205
Пакетный пересчёт	30	404,534	93,913	0	0,000	0,000	0,000
Оконный метод	30	42,438	0,682	5948	29,740	29,759	31,195
Предлагаемый метод	30	57,449	0,913	1182	5,910	5,692	6,314

Из Таблицы 2 видно, что пакетный пересчёт во всех сценариях обеспечивает нулевую ошибку относительно эталонного куба по агрегатам count и sum. Это связано с тем, что при поступлении нового микробатча событий куб строится повторно. Вместе с этим полный пересчет приводит к тому, что данный метод имеет наибольший объем обновленных ячеек – в среднем 94 ячейки.

Оконный метод показывает меньшие вычислительные затраты и изменяет менее одной ячейки на событие в среднем. Однако при росте доли запаздывающих событий его ошибка быстро увеличивается. При 20 % запаздывающих событий ошибка count составила 20,165 %, ошибка sum – 20,242 %, а доля ошибочных ячеек – 21,200 %. При 30 % запаздывающих событий ошибка count достигла 29,740 %, а ошибка sum – 29,759 %. Такой большой процент ошибки связан с тем, что события, поступившие после закрытия окна, не исправляют ранее сформированный результат.

Предлагаемый метод показал меньшую ошибку по сравнению с оконным методом. При 20 % запаздывающих событий ошибка count составила 3,935 %, ошибка sum – 3,882 %, а доля ошибочных ячеек – 4,205 %. При 30 % запаздывающих событий ошибка count составила 5,910 %, ошибка sum – 5,692 %, а доля ошибочных ячеек – 6,314 %. Это объясняется тем, что часть запаздывающих событий учитывается в компенсационном слое и корректирует соответствующие ячейки куба.

По результатам эксперимента мы наблюдаем, что предлагаемый метод также совершает определенное количество ошибок, хоть и значительно меньше, чем оконный метод. Это связано с использованием ограниченного горизонта компенсации в размере 45 минут. Если событие поступало в прототип куба позже горизонта, то оно уже не учитывалось. При увеличении доли запаздывающих событий число событий, которые приходили после горизонта росло, что приводило к возрастанию ошибок агрегатов.

Для проверки влияния горизонта компенсации была проведена дополнительная серия экспериментов. Доля запаздывающих событий фиксировалась на уровне 20 %, а горизонт компенсации изменялся от 15 до 90 минут. Результаты представлены в Таблице 3.

Результаты в Таблице 3 показывают, что горизонт компенсации является значимым параметром предлагаемого метода. При увеличении горизонта с 15 до 90 минут ошибка count снизилась с 15,565 % до 1,990 %, ошибка sum – с 15,508 % до 2,002 %, а доля ошибочных ячеек – с 16,476 % до 2,130 %. Одновременно размер компенсационного слоя увеличился с 799 до 3470 ячеек. Исходя из результатов дополнительной серии экспериментов, можно сделать вывод, что увеличение горизонта

компенсации повышает точность итогового куба, но требует хранения большего объёма промежуточного состояния.

Таблица 3 – Влияние горизонта компенсации на результат предлагаемого метода
Table 3 – The effect of the compensation horizon on the result of the proposed method

Горизонт компенсации, минуты	Среднее время обработки события, мкс	Изменено ячеек на событие	Потерянные события	Размер компенсационного слоя, ячеек	Ошибка count, %	Ошибка sum, %	Ошибочные ячейки, %
15	54,486	0,820	3113	799	15,565	15,508	16,476
30	56,425	0,874	2004	1899	10,020	9,855	10,666
45	58,422	0,932	810	3070	4,050	3,955	4,330
90	59,569	0,952	398	3470	1,990	2,002	2,130

Полученные результаты показывают, что предлагаемый метод занимает промежуточное положение между пакетным пересчётом и оконным методом. В сравнении с методом полного перерасчета предлагаемый метод использует в несколько раз меньше времени на обработку события за счет отказа от повторного построения всего куба при каждом обновлении. В отличие от оконного метода, предлагаемый метод позволяет учитывать значительную часть запаздывающих событий за счёт компенсационного слоя.

Обсуждение

Результаты эксперимента демонстрируют, что предложенный метод стоит рассматривать как компромиссный вариант при работе с неупорядоченным поступлением данных. Его преимущество проявляется в тех случаях, когда полный пересчёт куба слишком затратен, а простое закрытие временных окон приводит к потере существенной части поздних событий.

Ключевым параметром метода является горизонт компенсации. Он определяет, какие события необходимо сохранить в компенсационном слое, а какие следует отбрасывать как слишком поздние. Выбор значения параметра горизонта компенсации необходимо осуществлять на основе распределения задержек события в конкретной информационной системе. Если задержки в ИС высокие и нестабильные, то значение горизонта необходимо увеличивать, тем самым повышая точность данных ценой увеличения размера компенсационного слоя.

Однако компенсационный слой нельзя рассматривать как неограниченное хранилище. Для практического применения метода необходима реализация механизма материализации. Механизм будет отвечать за перенос данных из компенсационного слоя в базовый куб и последующее удаление устаревших записей. Без материализации метод может привести к росту объёма слоя и снижению эффективности обработки.

Кроме того, ограничением предлагаемого метода является применимость только к аддитивным агрегатам, таким как count, sum и avg. Для использования предложенного метода при расчете агрегатов min, max, distict count потребуются дополнительные структуры данных, поэтому их поддержка должна рассматриваться отдельно.

Предлагаемый метод является предпочтительным при работе с журналами ИС, для которых:

– свойственно наличие запаздывающих событий;

– полный пересчет куба является избыточным и слишком затратным по вычислительным и временным ресурсам.

Заключение

В работе рассмотрена задача инкрементального обновления OLAP-куба по потоку событий журналов информационных систем при неупорядоченном поступлении данных. Различие между временем возникновения события и временем его поступления в аналитический контур может приводить к ошибкам обновления агрегатов. Итогом подобной временной рассинхронизации является искажение картины работы ИС. Полный пересчёт куба позволяет сохранить корректность агрегатов, но требует значительных вычислительных затрат при регулярном обновлении.

Для решения этой задачи предложен инкрементальный метод обновления OLAP-куба с использованием базового куба и компенсационного слоя. В зависимости от времени поступления в хранилище события попадают либо в основной куб, либо в компенсационный слой. Итоговое представление куба формируется как объединение базового куба и накопленных поправок. Для исключения повторного учёта событий в метод включён механизм проверки уникальных идентификаторов.

Проведённый вычислительный эксперимент показал, что предложенный метод снижает ошибку агрегатов по сравнению с оконным методом при наличии запаздывающих событий и требует меньшего объёма повторной обработки данных по сравнению с пакетным пересчётом. Важным параметром точности метода является горизонт компенсации. Эксперимент продемонстрировал, что увеличение горизонта повышает полноту учёта поздних событий, но приводит к росту компенсационного слоя.

Практическая значимость предложенного метода состоит в возможности поддерживать актуальные агрегаты OLAP-куба без регулярного полного пересчёта всей структуры. Это позволяет уменьшить затраты временных и вычислительных ресурсов при получении новых событий. Метод может быть использован в задачах, где требуется получать агрегированную аналитику по потокам событий с возможными задержками поступления. Примером областей использования могут быть системы мониторинга, аудит, анализ событий безопасности. По результатам текущей работы можно выделить несколько направлений дальнейших исследований. Первое направление – это разработка адаптивного выбора горизонта компенсации на основе статистики задержек событий. Следующее направление – расширение метода для использования агрегатов типа mix, max и distinct count.

СПИСОК ИСТОЧНИКОВ / REFERENCES

1. Фролов В.А., Хайруллин Г.И., Афанасьев Р.З. Анализ форматов хранения многомерных моделей данных в контексте многомерных кубов. *Computational nanotechnology*. 2025;12(4):187–194. https://doi.org/10.33693/2313-223X-2025-12-4-187-194
   Frolov V.A., Khayrullin R.Z., Afanasyev G.I. Analysis of storage formats for multidimensional data models in the context of multidimensional cubes. *Computational nanotechnology*. 2025;12(8):187–194. (In Russ.). https://doi.org/10.33693/2313-223X-2025-12-4-187-194
2. Неупокоева Е.В., Быстров В.В. Анализ OLAP-решений для исследования жизнеспособности региональных социально-экономических систем. *Труды Кольского научного центра РАН. Серия: Технические науки*. 2023;14(7):52–67. https://doi.org/10.37614/2949-1215.2023.14.7.006
   Neupokoeva E.V., Bystrov V.V. Analysis of OLAP solutions for studying resilience of regional socio-economic systems. *Transactions of the Kola Science Centre of RAS. Series: Engineering Sciences.* 2023;14(7):52–67. (In Russ.). https://doi.org/10.37614/2949-1215.2023.14.7.006

3. Akidau T., Begoli E., Chernyak S., et al. Watermarks in stream processing systems: semantics and comparative analysis of Apache Flink and Google cloud dataflow. *Proceedings of the VLDB Endowment*. 2021;14(12):3135–3147. https://doi.org/10.14778/3476311.3476389
4. Fragkoulis M., Carbone P., Kalavri V., et al. A survey on the evolution of stream processing systems. *The VLDB Journal*. 2024;33:507–541. https://doi.org/10.1007/s00778-023-00819-8
5. Svingos C., Hernich A., Gildhoff H., et al. Foreign Keys Open the Door for Faster Incremental View Maintenance. *Proceedings of the ACM on Management of Data*. 2023;1(1):1–25. https://doi.org/10.1145/3588720
6. Cuzzocrea A., Moussa R., Vercelli G. An Innovative Lambda-Architecture-Based Data Warehouse Maintenance Framework for Effective and Efficient Near-Real-Time OLAP over Big Data. In: *Big Data – BigData 2018: 7th International Congress, Held as Part of the Services Conference Federation, SCF 2018, June 25–30, 2018, Seattle, WA, USA*. Cham: Springer; 2018. P. 149–165. https://doi.org/10.1007/978-3-319-94301-5_12
7. Tahir J., Mayer R., Doblander C., et al. How Reliable are Streams? End-to-End Processing-Guarantee Validation and Performance Benchmarking of Stream Processing Systems. *Proceedings of the VLDB Endowment*. 2024;18(3):585–598. https://doi.org/10.14778/3712221.3712227
8. Ступников С.А., Скворцов Н.А., Брюхов Д.О. Перспективные методы реализации инкрементального обновления материализованных представлений в современных реляционных системах управления базами данных. *Системы и средства информатики*. 2025;35(1):95–110. https://doi.org/10.14357/08696527250105
Stupnikov S.A., Skvortsov N.A., Briukhov D.O. Advanced methods for implementation of incremental view maintenance in modern relational database management systems. *Systems and Means of Informatics*. 2025;35(1):95–110. (In Russ.). https://doi.org/10.14357/08696527250105
9. Логиновский О.В., Шинкарев А.А., Коваль М.Е. Разработка архитектуры систем информационного поиска на основе очередей сообщений в корпоративных информационных системах. *Прикладная математика и вопросы управления*. 2021;(1):119–140. (In Russ.). URL: https://doi.org/10.15593/2499-9873/2021.01.07
Loginovskiy O.V., Shinkarev A.A., Koval M.E. Development of architecture of message queue based information search systems for enterprise information systems. *Applied Mathematics and Control Sciences*. 2021;(1):119–140. (In Russ.). URL: https://doi.org/10.15593/2499-9873/2021.01.07
10. Tang B., Han S., Yiu M.L., et al. Extracting Top-K Insights from Multi-dimensional Data. In: *Proceedings of the 2017 ACM International Conference on Management of Data: SIGMOD '17, May 14–19, 2017, Chicago, IL, USA*. New York: Association for Computing Machinery; 2017. P. 1509–1524. https://doi.org/10.1145/3035918.3035922
11. Dehne F., Kong Q., Rau-Chaplin A., et al. Scalable real-time OLAP on cloud architectures. *Journal of Parallel and Distributed Computing*. 2015;79–80:31–41. https://doi.org/10.1016/j.jpdc.2014.08.006
12. Самарев Р.С. Обзор состояния области потоковой обработки данных. *Труды Института системного программирования РАН*. 2017;29(1):231–260. (In Russ.).
Samarev R.S. Survey of streaming processing field. *Proceedings of the Institute for System Programming of the RAS*. 2017;29(1):231–260. (In Russ.).

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Уфимцев Георгий Александрович, аспирант МИРЭА – Российского технологического университета, Москва, Российская Федерация.
e-mail: georgeuf@yandex.ru

Georgy A. Ufimtsev, Postgraduate of MIREA – Russian Technological University, Moscow, the Russian Federation.

Зыков Сергей Викторович, доктор технических наук, профессор МИРЭА – Российского технологического университета; профессор национального исследовательского университета «Высшая школа экономики», Москва, Российская Федерация.
e-mail: szykov@hse.ru
ORCID: 0000-0002-2115-5461

Sergey V. Zykov, Doctor of Engineering Sciences, Professor of MIREA – Russian Technological University; Professor of National Research University Higher School of Economics, Moscow, the Russian Federation.

Статья поступила в редакцию 18.05.2026; одобрена после рецензирования 03.07.2026; принята к публикации 05.08.2026.

The article was submitted 18.05.2026; approved after reviewing 03.07.2026; accepted for publication 05.08.2026.