DOI 10.36622/VSTU.2023.4.26.007
УДК 004.056.5

РАЗРАБОТКА РЕСУРСНО-ЭФФЕКТИВНОЙ ТЕХНОЛОГИИ ДИНАМИЧЕСКОГО МАСКИРОВАНИЯ КОНФИДЕНЦИАЛЬНОЙ ИНФОРМАЦИИ В РЕЛЯЦИОННЫХ БАЗАХ ДАННЫХ

В.П. Лось, Д.Д. Маланьин

Защита конфиденциальной и персональной информации является актуальной задачей информационной безопасности. В особенности это касается реляционных баз данных, которые широко используются в различных сферах деятельности: от банковского сектора до интернет-магазинов. Нарушение конфиденциальности часто становится проблемой безопасности данных, особенно при работе с персональными данными клиентов. В статье рассмотрена методика построения ресурсно-эффективной, с точки зрения используемой постоянной памяти сервера базы данных, технологии динамического маскирования конфиденциальной информации в реляционных структурах баз данных. Методика позволяет обезопасить данные при обращении к ним аналитических сервисов и приложений, сохраняя при этом функциональные возможности.

Ключевые слова: информационная безопасность, системы управления базами данных, маскирование данных.

Введение

В современном мире, где цифровые технологии проникают во все сферы жизни, защита персональных данных стала актуальной и жизненно важной проблемой. Объемы персональных данных (ПДн), обрабатываемых в государственных, коммерческих и частных информационных системах растут с каждым годом. Симметрично возрастает и вероятность возникновения инцидентов, при которых персональные данные теряют свою конфиденциальность в результате утечек.

Как в Российской Федерации, так и за рубежом, имеются нормативные правовые документы, предусматривающие порядок обработки персональных данных.

В Российской Федерации одними из основных документов, регламентирующих жизненный цикл персональных данных, являются:

– Федеральный закон №152-ФЗ "О персональных данных" от 27 июля 2006 г.;

– Постановление Правительства РФ от 1 ноября 2012 г. N 1119 "Об утверждении требований к защите персональных данных при их обработке в информационных системах персональных данных", содержащее требования, которые необходимо выполнять при обработке персональных данных;

– Приказ Роскомнадзора от 5 сентября 2013 г. №996 и другие документы.

По данным компании InfoWatch количество утечек информации ограниченного доступа за первую половину 2022 г. во всем мире возросло почти двукратно (на 93,2%), а по России почти в 1,5 раза (на 45,9%), по сравнению с тем же периодом 2021 г. (рис. 1).

Согласно другому исследованию, посвященному актуальным киберугрозам второго квартала 2023 года, более 50% всей похищенной конфиденциальной информации при атаках на организации, являются персональными данными.

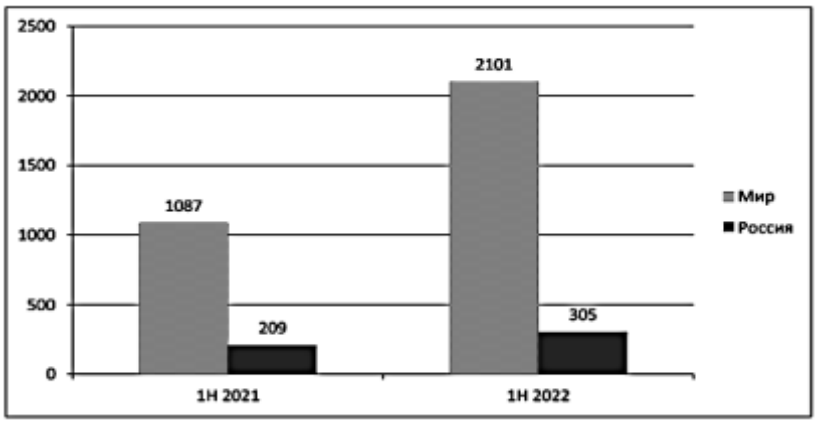

Рис. 1. Количество утечек информации ограниченного доступа за первую половину 2021/2022 гг.

В связи с вышеприведенными фактами, становится очевидно, что требуется сокращать количество лиц, задействованных непосредственно в работе с ПДн и другого рода чувствительной информацией, так как в большинстве случаев итоговая выборка их не содержит. В качестве перспективного подхода к решению вопроса обеспечения сотрудников необходимыми для их работы данными и единовременного ограничения в доступе к ПДн, обрабатываемых в качестве исходных, можно назвать подход по обезличиванию данных.

Деперсонализированные (обезличенные) ПДн — это данные, которые потеряли свою связь с субъектом информации. Принцип деперсонализации подразумевает обработку персональных данных, при которой невозможно будет определить первоначального субъекта информации.

Как отмечено в методических рекомендациях по применению приказа Роскомнадзора от 5 сентября 2013 года № 996, при выборе методов и процедур обезличивания персональных данных следует руководствоваться целями и задачами обработки персональных данных, что говорит о потребности в одних и тех же данных, но деперсонализированных различными алгоритмами, в зависимости от решаемых задач, в случае большого количества таких деперсонализированных срезов, изначально базирующихся на одной и той же персонализированной информации, представляется нецелесообразным сохранение каждого из них в виде физической таблицы реляционной СУБД (системы управления базами данных). При наличии значимого объема данных и появления большого числа деперсонализированных срезов, хранящихся в виде таблиц, зачастую появляется необходимость своевременной актуализации этих данных на основе изменений привнесенных в исходную таблицу или таблицы, поверх которых был выбран срез, что требует создание, актуализацию, поддержку и своевременное выведение из эксплуатации не только созданных объектов, но и обеспечивающих их наполнение данными процессов «извлечения, преобразования, загрузки данных» (ETL-процессов).

Предшествующие работы

В силу различий законодательной базы у зарубежных исследователей преимущественно рассматривается необратимое обезличивание [1-2].

Исследования в России по теме обезличивания персональных данных в большей степени ориентированы на разработку, реализацию и внедрение алгоритмов обезличивания, а также на построение методик оценки эффективности алгоритмов обезличивания (П.Ю. Пушкин, Е.В. Никульчев, В.П. Лось, А.Н. Соколов и Ю.Е. Мищенко, И.П. Карпова и другие авторы) [3-5].

Вопросы эффективной обработки данных в реляционных СУБД рассматривали в своих работах как российские ученые, в том числе, В. А. Белов, Д. Ю. Ильин, Е. В. Никульчев [6], так и зарубежные P. Martins, P. Tome, S. Andjelic, S. Obradovic, B. Gacesa и другие [7].

В данной статье разработана методика построения ресурсно-эффективного, с точки зрения используемой постоянной памяти сервера базы данных, технического решения, реализующего динамическое маскирование данных, предназначенное для построения аналитической отчетности, обеспечивающее совершенствование существующих средств защиты информации.

Приведенная методика предлагается в первую очередь для реляционных СУБД, поскольку они остаются наиболее распространенными решениями, используемыми для хранения связанных, исторических и коммутативных данных, на основе которых производится анализ, построение отчетности, однако, методика может быть применена и для других типов СУБД.

Основные задачи, решенные в статье:

1) определены структурные элементы архитектуры и выстроена взаимосвязь между ними;

2) разработан алгоритм маскирования, реализуемый в реляционных СУБД, для оперативного предоставления маскированных данных субъектам информационных процессов (разработчикам, аналитикам, тестировщикам информационно-аналитических систем);

3) проведена оценка эффективности предложенных решений.

Методика построения технологии динамического маскирования конфиденциальной информации в реляционных субд

Методика состоит из 5 этапов.

1 Этап. Формирование архитектуры, реализующей динамическое маскирование

Предполагается создание следующих элементов:

– СУБД, содержащей чувствительную информацию (DBc);

– СУБД, содержащей маскированную информацию, доступную для обработки (DBm);

– маскирующего преобразования, реализуемого в DBc (MT);

– механизма, позволяющего подключение маскированного объекта DBc в DBm (FDP).

Архитектура, включающая основные элементы и их взаимосвязи, представлена на рис. 2.

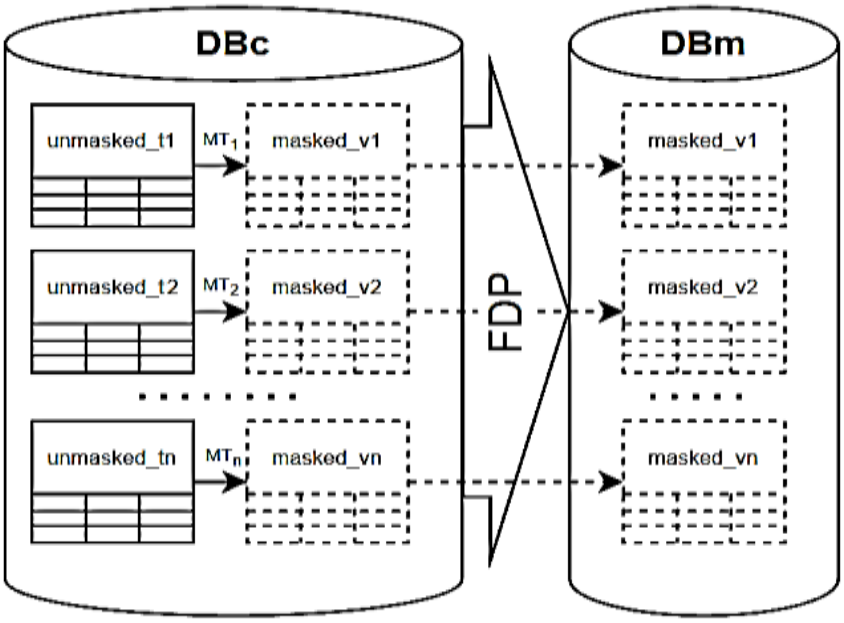

Рис. 2. Основные элементы архитектуры и их взаимосвязь

2 Этап. Разграничение доступа

В рамках методики подразумевается, что субъекты информационных процессов, не обладающие доступом к чувствительной информации, однако, обладающие необходимостью получения данных из

объектов их содержащих, будут обращаться в DBm, на которой будет представлен виртуальный маскированный объект данных, физически немаскированных, хранящихся в DBc.

Предполагается, что разграничение доступа на DBm и DBc может осуществляться как встроенными средствами СУБД, так и сторонними системами и средствами.

3 Этап. Создание маскированных данных в форме виртуальных объектов

Предлагается реализовать маскирование данных полностью на стороне элементов группы DBc, путем применения маскирующих функций, исполняемых непосредственно в момент обращения к данным. Полученные в результате маскированные данные на DBc не сохранять физически, а также представлять в виде виртуальных объектов.

Таковым виртуальным объектом в реляционных СУБД может являться представление (view).

4 Этап. Обработка первоначальных таблиц перед взаимодействием с ними пользователем

Представление – объект базы данных, являющийся результатом выполнения запроса к определенным таблицам и полям с помощью оператора выборки (например, SELECT), в момент обращения к представлению. Также представления предполагают и обработку первоначальной таблицы перед взаимодействием с ними пользователем, например, осуществление сортировки, применение процедурных функции и др. Представление не является объектом, обеспечивающим физическое сохранение информации, тем самым, при наличии необходимости маскирования различными способами одних и тех же данных (такая необходимость в маскировании различными способами может возникать в результате различий выполняемых пользователями задач), не требует дополнительного места в СУБД для хранения каждого такого маскированного объекта. Важным свойством подобной схемы является то, что изменение исходных (немаскированных) данных на DBc (добавление, удаление, обновление) может быть обработано субъектом данных маскированного представления на DBm оперативно, без необходимости выстраивания и запуска отдельных ETL-процессов. Однако, стоит отметить, что несмотря на ресурсную эффективность с точки зрения экономии постоянной памяти СУБД, при каждом обращении к объекту маскированных данных будут требоваться ресурсы на осуществление динамического маскирования.

5 Этап. Подключение внешней функции к конфиденциальной информации через отображение, включающее динамическое маскирование

Отображение маскированного представления из элемента DBc в элемент DBm (будем обозначать такое отображение как *FDP[DBc;DBm]*) предлагается реализовывать посредством механизма, позволяющего подключать внешние сущности к целевой реляционной СУБД.

АЛГОРИТМ ДИНАМИЧЕСКОГО МАСКИРОВАНИЯ

Предположим, что в реляционной СУБД *DBc* содержится объект (таблица) T_{DBc}, состоящий из записей L_i:

$$T_{DBc} = \{L_1, L_2 \dots L_i, \dots L_V\},$$
$$L_i = \{A_1, A_2 \dots A_j, \dots A_K\},$$

где $i = 1 \dots V$ – номер записи (строки) таблицы T_{DBc};

V – объем таблицы T_{DBc};

A_j – атрибут (столбец) таблицы T_{DBc}, $j = 1 \dots K$;

K – количество атрибутов (столбцов).

При этом существует наперед выделенное подмножество C атрибутов множества $A_{1 \dots K}$, которые по отдельности или в совокупности являют собой чувствительные данные.

Тогда для динамического маскирования чувствительной информации требуется:

1. Предоставить техническому пользователю *DBc* права на чтение T_{DBc}.

2. Определить маскирующее преобразование *MT(),* исходя из целей и задач маскирования.

3. Создать в *DBc* представление V^T_{DBc}, которое при обращении будет формировать

следующие маскированные записи L'_i на основе записей исходной таблицы T_{DBc}:

$$V^T_{DBc} = \{L'_1, L'_2 \ldots L'_i, \ \ldots L'_V\},$$
$$L'_i = \{A'_1, A'_2 \ldots A'_j, \ \ldots A'_K\},$$

где A'_j – маскированный атрибут (столбец) таблицы T_{DBc}: $A'_j = A_j$, *если* $A_j \notin C$; $A'_j = MT(A_j)$, *если* $A_j \in C$.

4. Поверх настроенного отображения *FDP[DBc;DBm]* создать V^T_{DBm} в DBm, предварительно предоставив права на чтение V^T_{DBc} техническому пользователю *DBc*, используемому *DBm* в качестве удаленного пользователя в *DBc*.

5. Предоставить права на V^T_{DBm} пользователям маскированных данных в *DBm*.

На рис. 3 приведена визуализация шагов алгоритма.

Преимуществом использования данного решения является ресурсная эффективность с точки зрения использования постоянной памяти, сравнительная легкость реализации в различных СУБД, масштабируемость и повышение общей защищенности чувствительных данных. Далее каждый из пунктов будет рассмотрен более детально:

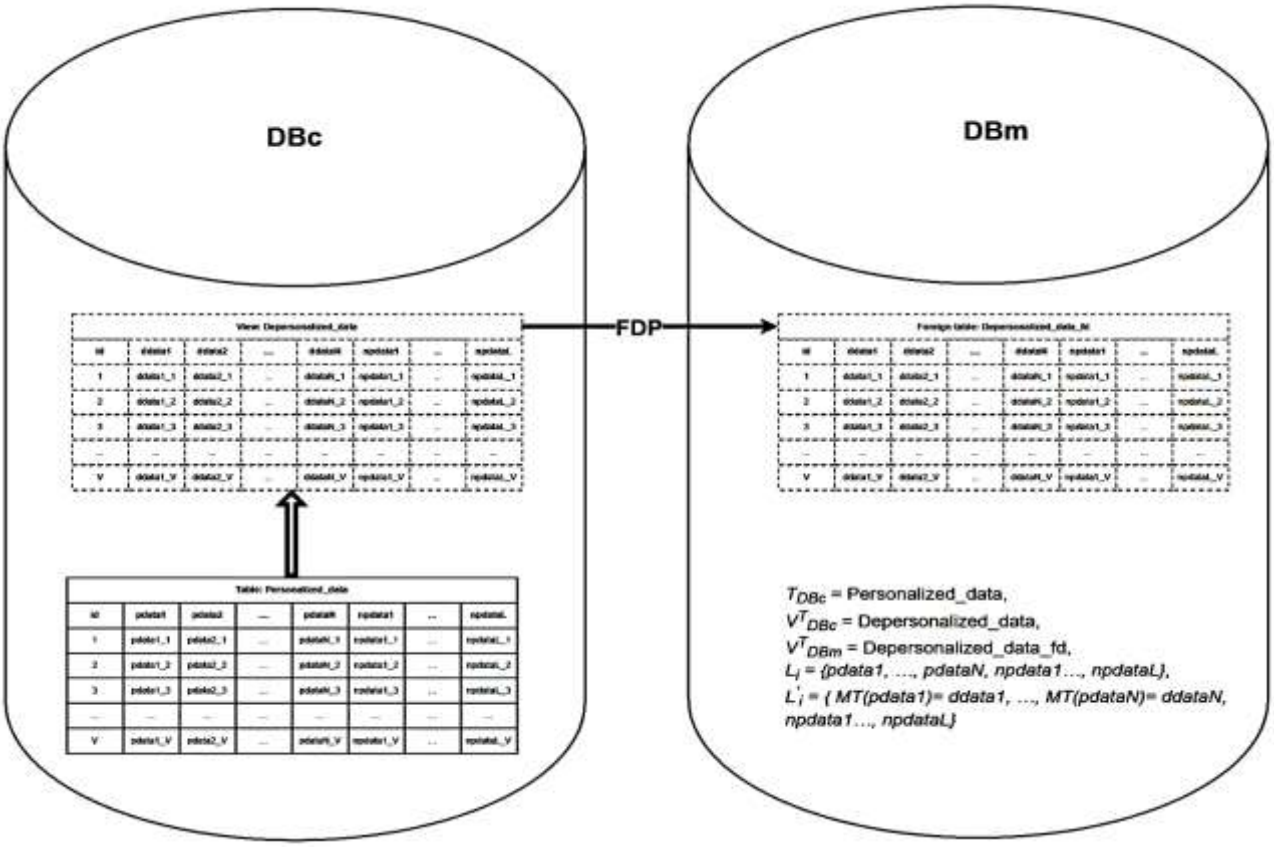

Рис. 3. Алгоритм динамического обезличивания

1. Ресурсная эффективность с точки зрения использования постоянной памяти СУБД достигается за счет использования представлений, которые являются объектами, не хранящими данные физически, а предоставляющими их вычисление при обращении.

2. Сравнительная легкость реализации приведенного решения достигается за счет того, что объекты и процессы, используемые в данном решении, могут быть построены на уровне СУБД, при условии наличия в СУБД встроенных маскированных преобразований МТ или же возможности создать таковые, например, посредством функций, а также встроенного механизма, позволяющего выполнить отображение FDP[DBc;DBm], настройка дополнительных инструментов или дополнительного программного обеспечения в этом случае не требуется, эти действия

могут быть выполнены администратором баз данных.

3. Масштабируемость решения может быть достигнута за счет применения горизонтального масштабирования системы путем добавления:

– нескольких СУБД $DBc_1 \ldots DBc_k$, содержащих чувствительные данные;

– нескольких СУБД $DBm_1 \ldots DBm_n$, содержащих маскированные представления;

– необходимого числа различных маскирующих преобразований $MT_1 \ldots MT_m$.

4. Повышение защищенности достигается за счет того, что пользователи, имеющие доступ к представлению в DBm не имеют доступ к представлению в DBc, не имеют доступа к исходной таблице DBc и, в общем случае, к СУБД DBc в целом, как отмечалось ранее подобное разграничение доступа может быть достигнуто не только за счет разграничения прав доступа на уровне СУБД, но, например, и использованием межсетевого экранирования для сетевого разграничения доступа.

Имплементация и оценка предложенной методики в субд PostgreSQL

Имплементация описанной методики может быть проведена с использованием любой реляционной СУБД, поддерживающей механизм подключения внешних сущностей из других СУБД. Например, таковым механизмом (*fdw*) во встроенной поставке обладают СУБД Postgres Pro и Postgres. Строго говоря, в предложенной схеме СУБД могут быть как однотипными, так и различными, главным требованием остается поддержка (нативная или добавленная модулем) подключения внешних сущностей из других СУБД, входящих в кластер маскирования данных.

В качестве примера рассмотрим имплементацию алгоритма маскирования с использованием двух СУБД Postgres 16.0, встроенный механизм подключения внешних сущностей *postgres_fdw*, в качестве функции маскирования персональных данных приведем следующую, написанную с использованием встроенных возможностей языка PL/pgSQL, используемого в СУБД Postgres, получающей на вход строку, а на выходе предоставляющую исходную строку, но все буквы, в словах которой, кроме первой, заменены на символ «*».

Пример: на вход маскирующей функции *pii.mask_string_regex* предлагается строка «Иванов Иван Иванович», в этом случае результатом выполнения функции будет строка «И***** И*** И*******», код функции приведен ниже.

```
    CREATE OR REPLACE FUNCTION
pii.mask_string_regex(input_str VARCHAR)
    RETURNS VARCHAR AS $$
    BEGIN
        RETURN regexp_replace(input_str,
'(?<=[\S])\S', '*', 'g');
    END;
    $$ LANGUAGE plpgsql;
```

Маскирующих функций, подобной вышеописанной, может быть создано достаточно много, на любой из специфических случаев, в котором требуется особый подход в маскировании содержащихся данных.

В качестве исходных данных выберем таблицу, содержащую поля различных типов данных (*INTEGER*, *VARCHAR*, *TIMESTAMP*) и произведем ее наполнение семплированными данными.

Для создания маскированных представлений используется объект VIEW, в котором, в простейшем случае выбирается исходная немаскированная таблица, все поля, содержащие чувствительную информацию, выбираются в представлении с использованием ранее созданных маскирующих функций, пример приведен ниже, а оставшиеся поля выбираются без изменений:

```
    CREATE OR REPLACE VIEW
pii_v.masked_pii_data AS
    select
            id as id,
            pii.mask_string_regex
(first_name) as masked_first_name,
            pii.mask_string_regex
(full_name) as masked_full_name,
            email,
            phone_number,
            created_at
    from pii.pii_data;
```

Отличие приведенного решения, например, от механизма DDM, имеющегося в СУБД SQL Server 2016 (13.x) и более поздних версиях заключается в неподверженности его к обходу маскировки с помощью статистической атаки или атаки путем подбора [10].

Сравним время, затрачиваемое на выборку всех записей таблицы/представления (с помощью запроса *select count(*) <table/view_name>*) и выбор отдельных записей, у которых в поле *email* содержится буква *z* с использованием оператора *LIKE* (*select count(*) <table/view_name> where email like '%z%'*), в каждом из приведенных объемов выборок запрос возвращает число записей равное 5% от объема выборки.

Для каждого из запросов и объемов выборки было произведено по 3 измерения, приведенные на графиках значения являются результатом усреднения этих измерений.

По приведенным графикам (рис. 4-5) можно видеть, что при большом объеме итоговой выборки скорость работы запроса значительно снижается, данное явление можно связать с низкой производительностью выбранной маскирующей функции *pii.mask_string_regex*, выберем другую маскирующую функцию, которая будет заменять все символы исходной строки на символ «*», кроме первого. Код функции приведен ниже:

```
    CREATE OR REPLACE FUNCTION
pii.mask_string_substr(input_str VARCHAR)
    RETURNS VARCHAR AS $$
    BEGIN
        RETURN substr(input_str, 1,
1)||rpad ('*', length (input_str)-1, '*');
    END;
    $$ LANGUAGE plpgsql;
```

Результаты тестирования скорости работы функций *pii.mask_string_substr* и *pii.mask_string_regex* приведены на рис. 6. Из него можно сделать вывод о том, что реализация функции маскирования оказывает сильное влияние на конечную производительность при запросе данных.

Ресурсная эффективность с точки зрения используемого дискового пространства достигается за счет того, что данные формируются динамически и не требуют места в постоянной памяти для своего хранения, что проиллюстрировано в табл. 1, данные о занимаемом объеме получены с помощью встроенной функции *pg_total_relation_size*.

Таблица 1

Рекурсивная эффективность алгоритма маскирования

Количество записей в исходной таблице	Объем исходной таблицы, занимаемый в постоянной памяти, МБ	Объем представления, занимаемый в постоянной памяти на DBc, МБ	Объем представления, занимаемый в постоянной памяти на DBm, МБ
500000	168	0	0
1000000	324	0	0
2000000	643	0	0

Таким образом, можно говорить о том, что приведенное решение является эффективным при наличии большого числа таблиц сравнительно небольшого объема, требующих маскирования приведенных в них данных, или же при частых задачах по выгрузке ограниченного объема данных, что является типовым случаем в работе аналитиков, разработчиков и тестировщиков.

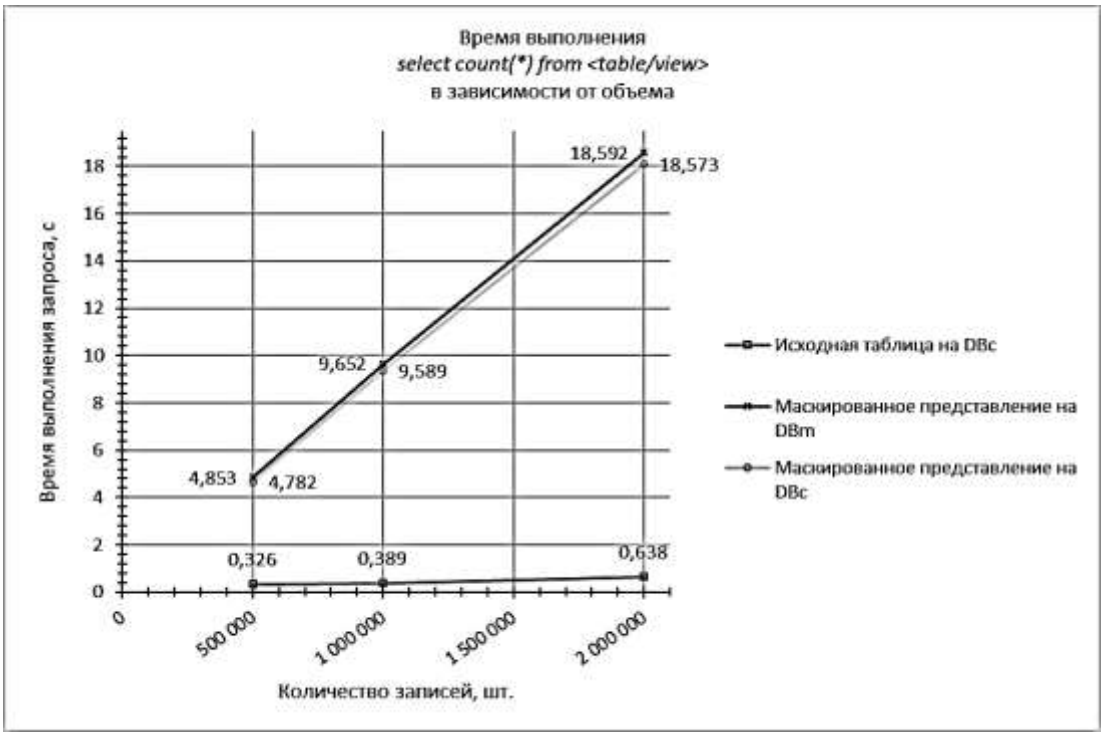

Рис. 4. Время выполнения запроса ко всем данным таблицы/представления в зависимости от объема

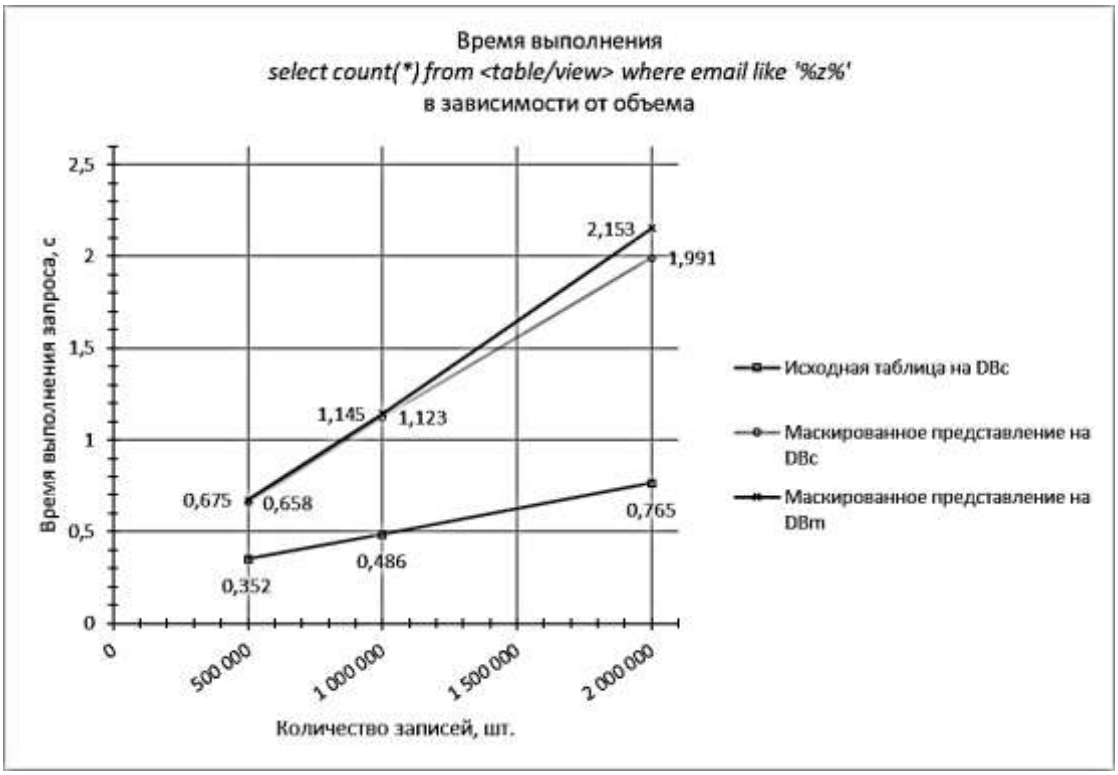

Рис. 5. Время выполнения запроса к части данных таблицы/представления в зависимости от объема

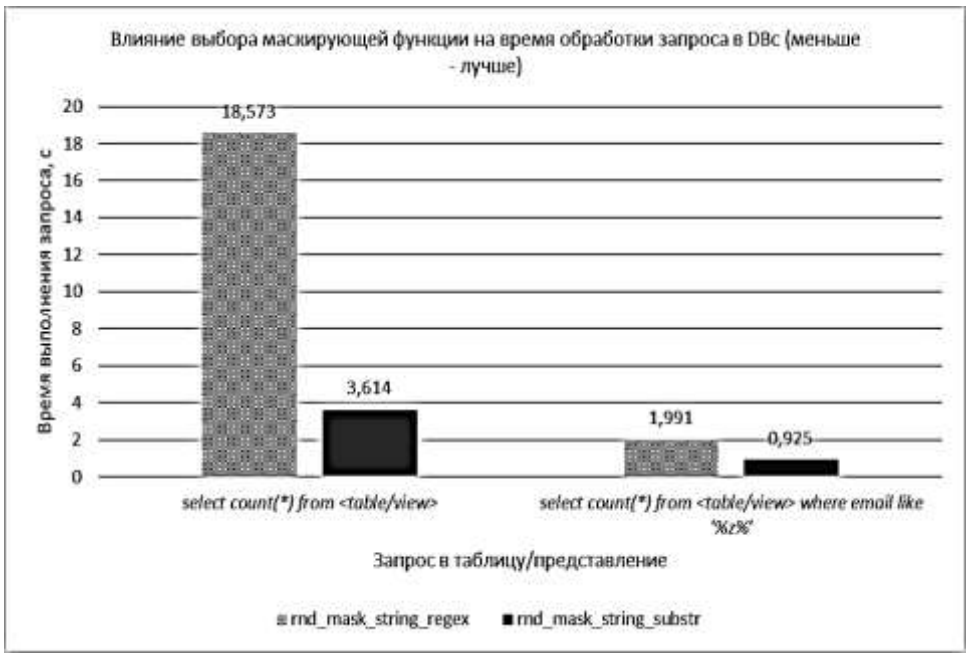

Рис. 6. Время выполнения запросов при использовании различных маскирующих функций (исходная таблица содержит 2000000 записей)

Заключение и направление дельнейших исследований

В заключении отметим, что была приведена и обоснована архитектура решения по динамическому маскированию чувствительной информации, что имеет ряд преимуществ над нестатическим.

Направлениями дальнейших исследований можно назвать следующие:

– разработка и имплементация архитектуры для нереляционных СУБД;

– разработка методов и алгоритмов по выявлению чувствительной информации для ее последующего маскирования в рамках инфраструктуры.

Список литературы

1. Gkountouna O., Terrovitis M. Anonymization in the Presence of Structural Knowledge. Technical Report, NTUA, 2013.

2. Guo N. Data anonymization based on natural equivalent class / N. Guo et al. //2019 IEEE 23rd International Conference on Computer Supported Cooperative Work in Design (CSCWD). IEEE, 2019. P. 22-27.

3. Мищенко Е.Ю. Алгоритмы реализации методов обезличивания персональных данных в распределенных информационных системах / Е.Ю. Мищенко, А.Н. Соколов // Доклады Томского государственного университета систем управления и радиоэлектроники. 2019. Т. 22. № 1. С. 66–70.

4. Пушкин П.Ю. Проблемы сбора, обработки и защиты персональных данных обучающихся – участников онлайн конкурсов и тестирования / П.Ю. Пушкин, Е.В. Никульчев, В.П. Лось // Информация и безопасность. 2020. Т. 23. № 2. С. 179-190.

5. Карпова И.П. О реализации метода обезличивания персональных данных / И.П. Карпова // Вестник компьютерных и информационных технологий. 2013. № 6. С. 56–60.

6. Белов В. А. Оценка эффективности обработки больших объемов данных в реляционных и колоночных форматах / В.А. Белов, Д.Ю. Ильин, Е.В. Никульчев // Вычислительные технологии. 2022. Т. 27. № 3. С. 46–65.

7. Andjelic S. A performance analysis of the DBMS – MySQL vs PostgreSQL. / S. Andjelic, S. Obradovic, B. Gacesa // Communications – Scientific Letters of the University of Zilina. 2008. No 10(4). P. 53–57.

8. Microsoft: Dynamic data masking. / URL: https://learn.microsoft.com/en-us/sql/relational-databases/security/dynamic-data-masking?view=sql-server-ver16 (дата обращения 25.10.2023).

МИРЭА – Российский технологический университет
MIREA – Russian Technological University

Поступила в редакцию 5.11.2023

Информация об авторах

Лось Владимир Павлович – д-р воен. наук, профессор, МИРЭА – Российский технологический университет, e-mail: alexanderostapenkoias@gmail.com

Маланьин Данила Дмитриевич – аспирант, МИРЭА – Российский технологический университет, e-mail: alexanderostapenkoias@gmail.com

DEVELOPMENT OF RESOURCE-EFFICIENT TECHNOLOGY DYNAMIC MASKING OF CONFIDENTIAL INFORMATION IN RELATIONAL DATABASES

V.P. Los, D.D. Malanin

The task of protecting conference and personal information is an urgent task of information security. This is especially true for relational databases, which are widely used in various spheres of activity: from the banking sector to online stores. Violation of confidentiality often becomes a problem of data security, especially when working with personal data of clients. In the article we consider the method of building resource-efficient (in terms of used permanent memory of the database server) technology of dynamic masking of confidential information in relational structures of databases. The technique allows to secure the data when analytical services and applications access them, while preserving functional capabilities.

Keywords: information security, database management systems, data masking.

Submitted 5.11.2023

Information about the authors

Vladimir P. Los – Dr. Sc. (Military), Professor, MIREA – Russian Technological University, e-mail: alexanderostapenkoias@gmail.com.

Danila D. Malanin – Graduate Student, MIREA – Russian Technological University, e-mail: alexanderostapenkoias@gmail.com.