Noise Reduction, ANR). Системы активного шумоподавления основаны на процессе интерференции волн. Наиболее эффективно такие системы справляются с шумом от 100 Гц до 1 КГц. Несмотря на то, что сам по себе метод позволяет эффективно подавлять окружающие звуки, реальные устройства не всегда справляются с этой задачей, особенно с акустическими колебаниями с частотой более тысячи Герц. Дело в том, что на регистрацию звука и вычисления противоположной волны у микроконтроллера уходит некоторое время. Из-за этого испускаемый ими звук уже не полностью противоположен входящему звуку, а отстает от него по фазе. Этот недостаток можно уменьшить, если сигнал, который надо подавить, подавать на вход такого устройства по электрическому или электромагнитному каналу. Благодаря тому, что электрический сигнал распространяется быстрее звука, прибор начинает обрабатывать сигнал ещё до его прихода в виде акустической волны. Благодаря этому микроконтроллер успевает подобрать «противоположный» звук, совпадающий по фазе с оригинальным, с меньшим запаздываем. Шумоподавление таких систем работает для звуков с частотой до 4 кГц. Таким образом, на границах контролируемой зоны можно понизить уровень акустических волн от источников режимной информации до безопасной величины. Такие системы можно с успехом использовать в режимных помещениях, где циркулирование акустической (речевой) информации запрещено нормативными документами или инструкциями.

Ключевые слова: акустика, акустическая защищенность, активное шумоподавление, защита информации, распространение звука в среде, система активного шумоподавления, активное звукоподавление, система активного звукоподавления.

Лізунов Сергій Іванович, к.т.н., доцент, доцент кафедри «Захист інформації» Національного університету «Запорізька політехніка».
E-mail: s.i.lizunov@i.ua.
Orcid ID: 0000-0001-8977-8705.

Лизунов Сергей Иванович, к.т.н., доцент, доцент кафедры «Защита информации» Национального университета «Запорожская политехника»

Lizunov Sergey, assistant professor of the Information Security Department, National University «Zaporizhzhia Polytechnic».

Філобок Євгеній Віталійович, студент магістратури кафедри «Захист інформації» Національного університету «Запорізька політехніка».
E-mail: filobock1999@gmail.com.
Orcid ID: 0000-0002-4105-3841.

Филобок Евгений Витальевич, студент магистратуры кафедры «Защита информации» Национального университета «Запорожская политехника».

Filobok Evgenij, graduate student of the Information Security Department, National University «Zaporizhzhia Polytechnic».

DOI: 10.18372/2410-7840.23.15432
УДК 004.652.4

АУДИТ ИЗМЕНЕНИЙ ТАБЛИЦ БАЗЫ ДАННЫХ SQL SERVER

Михаил Коломыцев, Светлана Носок

Неотъемлемым компонентом информационных систем является подсистема регистрации и аудита. Все современные СУБД обладают возможностью регистрировать и обрабатывать информацию о выполняемых операциях. SQL Server начиная с версии с 2008 имеет возможность определять спецификацию аудита [1] на уровне сервера или базы данных. Однако данные системного аудита не учитывают требования бизнес-модели информационной системы. Возникает необходимость настройки процесса регистрации с учетом специфики предметной области. Кроме того, важнейшей задачей защиты БД является обеспечение целостности данных. В современных сложных ІІС многие таблицы должны быть защищены от нежелательных операций изменений (вставок, обновлений и удалений). Данные аудита могут использоваться для отмены таких нежелательных действий. В этом случае, информации в системных журналах недостаточно. В данной статье рассматривается подход к решению задачи аудита изменений в таблицах БД с целью предотвращения нежелательных изменений данных. Такой подход реализован в виде методики создания объектов базы данных, с помощью которых осуществляется регистрация действий пользователей и отмена нежелательных изменений. Для решения задачи регистрации всех действий пользователя предлагается использовать отдельную схему БД для аудита, специальную таблицу аудита и триггеры информационных таблиц БД. Для отмены нежелательных изменений предложены SQL-процедуры. Для каждого этапа методики приведена программная реализация, что позволяет использовать ее как часть автоматизированной защиты БД. Неотъемлемым компонентом информационных систем является подсистема регистрации и аудита. Все современные СУБД обладают возможностью регистрировать и обрабатывать информацию о выполняемых операциях. SQL Server начиная с версии с 2008 имеет

возможность определять спецификацию аудита [1] на уровне сервера или базы данных. Однако требования бизнеса могут потребовать более тонкой настройки процесса регистрации. Кроме того, если возникает задача отмены нежелательных действий, то информации в системных журналах недостаточно. В современных сложных ІІС многие таблицы должны быть защищены от нежелательных операций изменений данных (вставок, обновлений и удалений). В данной статье рассматривается подход к решению задачи аудита и предотвращения нежелательных изменений таблиц БД. Для решения задачи регистрации всех действий пользователя предлагается использовать отдельную схему БД для аудита, специальную таблицу аудита и триггеры информационных таблиц БД.

Ключевые слова: *база данных, защита данных, регистрация и аудит, отмена нежелательных изменений в базе данных.*

ВСТУПЛЕНИЕ

Актуальность. В условиях все большего количества целенаправленных кибератак особенную важность приобретает своевременность выявления угрозы и реакции на возможный инцидент со стороны службы информационной безопасности.

Для решения такой задачи выполняется комплекс мер под общим названием Реагирование на инциденты кибербезопасности и управление ими (Cybersecurity Incident Responseand Management) [5], [6].

Особую актуальность тематика выявления угроз и реагирования на инциденты приобрела из-за точности и скорости процессов реагирования. Чтобы минимизировать ущерб необходимо снизить время реагирования до приемлемых значений иногда до секунд. Добиться этого можно создав инфраструктуру аудита, способную выявлять и реагировать на инциденты в автоматизированном режиме.

Постановка задачи. При выполнении сценария атаки нарушитель ИБ выполняет последовательность определенных шагов.

Сведения о каждом конкретном шаге атаки могут быть зафиксированы в журналах аудита систем, на которых направлена атака, а возможность реагирования на инцидент должна позволять устранять последствия атаки.

Задачей является разработка инфраструктуры аудита, позволяющей обнаруживать и регистрировать действия злоумышленника, а также устранять последствия атаки.

В данной статье результаты представлены в виде методики, реализующей такой подход применительно к базам данных.

Показана работоспособность методики.

Цель данной статьи: разработка методики построения подсистемы аудита изменений в базе данных, позволяющей отменять нежелательные изменения.

ОСНОВНАЯ ЧАСТЬ РАБОТЫ

Формирование инфраструктуры аудита. В работе для решения задачи регистрации всех действий пользователя предлагается сформировать в базе данных соответствующую инфраструктуру, состоящую из таких объектов как:

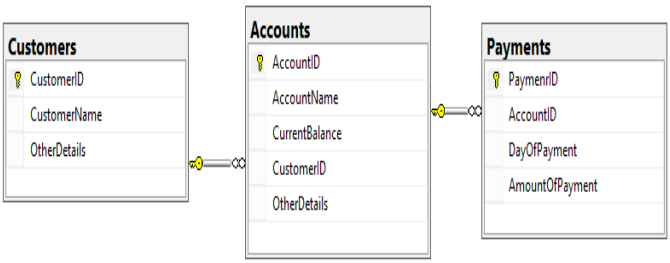
– схема БД, предназначенная для аудита,

– специальная таблица (или таблицы) с информацией аудита,

– триггеры информационных таблиц БД.

Для создания этих объектов необходимо выполнить ряд действий, описанных в статье.

Демонстрировать создаваемую инфраструктуру будем на примере базы данных с названием **Payments** со следующей структурой:



1. Создание схемы БД для таблиц аудита. Создадим определяемую пользователемсхему базы данных с именем **audit**. Важная особенность создаваемых пользователем схем - возможность группировать похожие объекты и назначать разрешения для группы [2].

```
DROPSCHEMAIFEXISTS[audit]
GO
CREATESCHEMA[audit]AUTHORIZATION[dbo]
GO
```

2. Создание пользователя БД. Все пользователи базы данных **Payments** должны быть определены как пользователи автономной БД. Функция автономной базы данных была представлена в SQL Server 2012 [3].

Такие пользователи не будут зависеть отбазы данных**master**. Это особенно важно, когда для высокой доступности используется активная репликация в географически разнесенные узлы, как например, в SQL Azure.

Для такого пользователя в окне установки соединения с сервером БД необходимо нажать кнопку **Параметры** и указать имя базы данных.

Для создания пользователя, предварительно необходимо переключить БД в автономный режим с помощью следующих команд:

```
Use MASTER;
GO
EXEC sp_configure 'containeddatabaseauthentication', 1;
GO
RECONFIGURE;
GO
ALTER DATABASE Payments
SET containment=partial
GO
```

Команда создания пользователя:

```
CREATEUSER[customer]
WITH PASSWORD=N'qwerty',
DEFAULT_SCHEMA=[CRM]
GO
```

3.Создание определяемой пользователем роль базы данных. Это важная с точки зрения безопасности возможность [4].

Такие роли служат той же цели, что и, группы в ActiveDirectory. Код ниже создает роль с именем **customer_role**.

```
DROP ROLE IFEXISTS[customer_role]
GO
CREATE ROLE [custom-
er_role]AUTHORIZATION[dbo]
GO
```

4. Установка разрешений роли. Следующим шагом является предоставление разрешений роли и добавление пользователя в роль.

```
GRANTINSERTONSCHEMA:[CRM]TO[customer_role]
GRANTUPDATEONSCHEMA:[CRM]TO[customer_role]
GRANTDELETEONSCHEMA:[CRM]TO[customer_role]
GRANTSELECTONSCHEMA:[CRM]TO[customer_role]
GO
EXECsp_addrolememberN'customer_role',N'customer'
GO
```

5. Создание таблицы аудита. В ранее созданной схеме **audit** создадим таблицу аудита - **log_table**. В нее будем записывать с помощью системных функций (SUSER_SNAME(),APP_NAME() и HOST_NAME ()) информацию о дате изменения, типе изменения, собственно изменение, название схемы и объекта. Информация в таблицу будет заноситься при срабатывании табличных триггеров ихранится в формате XML.

```
DROPTABLEIFEXISTS[audit].[log_table]
GO
CREATETABLE[audit].[log_table]
(
[id][numeric](18,0)IDENTITY(1,1)NOTNULL,
[_date][datetime]NOTNULL,
[_type][varchar](20)NOTNULL,
[_by][nvarchar](256)NOTNULL,
[app_name][nvarchar](128)NOTNULL,
[host_name][nvarchar](128)NOTNULL,
[schema_name][sysname]NOTNULL,
[object_name][sysname]NOTNULL,
[xml_recset][xml]NULL,
CONSTRAINT[pk_id]PRIMARYKEYCLUSTERED([id
]ASC)
);
GO
-- Задаем значения по умолчанию
ALTERTABLE[audit].[log_table]
ADDCONSTRAINT[DF_LTC_DATE]DEFAULT(getd
ate())FOR[_date];
ALTERTABLE[audit].[log_table]
ADDCONSTRAINT[DF_LTC_TYPE]DEFAULT('')F
OR[_type];
ALTERTABLE[audit].[log_table]
ADDCONSTRAINT[DF_LTC_BY]DEFAULT(coalesce
(suser_sname(),'?'))FOR[_by];
ALTERTABLE[audit].[log_table]
ADDCONSTRAINT[DF_LTC_APP_NAME]DEFAUL
T(coalesce(app_name(),'?'))FOR[app_name];
ALTERTABLE[audit].[log_table]
ADDCONSTRAINT[DF_LTC_HOST_NAME]DEFA
ULT(coalesce(host_name(),'?'))FOR[host_name];
GO
```

6. Создание табличных триггеров. Если количество таблиц небольшое, создание триггера таблицы вручную не является проблемой. Если в базе данных множество таблиц необходимо автоматизировать процесс создания триггеров. Авторы разработали хранимую процедуру с именем **audit_triggers**, которая автоматизирует процесс создания триггеров аудита (код процедуры доступен по ссылке[5]).

Хранимая процедура **audit_triggers** принимает два входных параметра:

– **@schema_name**. Имя схемы, по умолчанию **dbo**. Можно указать любое имя схемы. Однако учтите, что нельзя создавать триггер для

таблицы **audit_triggers**, это вызовет бесконечный цикл при срабатывании триггера;

– **@action**имеет два корректных значения. Значение **drop** удаляет триггеры аудита из всех таблиц в данной схеме. Значение **create** заменяет текущий триггер аудита новым. Значение по умолчанию **create**.

Необходимо учитывать, что создание триггеров для большого количества таблиц в интенсивно обновляемой БД может сказаться на ее производительности. Рекомендуется создавать их только для тех таблиц, целостность которых критически важна.

Приведенный ниже код создает триггеры в схеме**CRM** и выводит список созданных триггеров.

```
EXEC [audit].[audit_triggers] @schema_name = 'CRM', @action = 'create'
GO
-- Список созданных триггеров
SELECT
s.name,tr.name,tr.type_desc,tr.is_instead_of_trigger
FROM
sys.triggerstr
joinsys.tables t ontr.parent_id=t.object_id
joinsys.schemas s ONt.schema_id=s.schema_id
WHERE
  s.name ='CRM'
GO
```

С точки зрения безопасности, необходимо, чтобы созданные журналы аудита и триггеры не могли быть подделаны обычным пользователем. Пользователь, действия которого контролируются, не должен видеть таблицу аудита и хранимую процедуру. В нашем случае, созданная учетная запись **customer** не имеет никаких прав на схему **audit**.

Тестирование аудита изменений в таблицах.

Для проверки работоспособности созданной инфраструктуры подключимся к серверу БД с учетной записью **customer**, не забыв указать, что подключаемся к базе **Payments**и выполним ряд действий по изменению информации в таблице.

Создадим запись в таблице **Payments**.

```
insertintoCRM.Paymentsvalues (1,1,getdate(), 1000);
go
```

Сделаем обновление записи.

```
updateCRM.Payments
setAccountID = 2, AmountOfPayment = 5000
whereAccountID = 1
go
```

Удалим запись из таблицы.

```
deletefromCRM.PaymentswhereAccountID = 1;
go
```

В таблицу аудита будет записана информация об операциях вставки, обновления и удаления.

	id	_date	_type	_by	schema_name	object_name	xml_recset
1	5	2020-03-25 16:37:32.260	INSERT	customer	[CRM]	[Payments]	<RecordSet><Record><PaymenrID>6</PaymenrID><Acco...
2	6	2020-03-25 16:37:32.287	UPDATE	customer	[CRM]	[Payments]	<RecordSet><Record><PaymenrID>6</PaymenrID><Acco...
3	7	2020-03-25 16:37:32.290	INSERT	customer	[CRM]	[Payments]	<RecordSet><Record><PaymenrID>7</PaymenrID><Acco...
4	8	2020-03-25 16:37:32.307	DELETE	customer	[CRM]	[Payments]	<RecordSet><Record><PaymenrID>7</PaymenrID><Acco...

Если дважды щелкнуть поле xml_recset в SQL Server Management Studio, откроется окно с информацией о выполненной операции.

```
<RecordSet>
  <Record>
    <PaymenrID>7</PaymenrID>
    <AccountID>1</AccountID>
    <DayOfPayment>2020-03-25T16:37:32.290</DayOfPayment>
    <AmountOfPayment>1000.0000</AmountOfPayment>
  </Record>
</RecordSet>
```

Эта запись более подробная, чем запись системного аудита SQL Server. Но, что особенно важно, эту информацию легко использовать для отмены нежелательных действий.

Администратору базы данных иногда приходиться отменить нежелательные действия, выполненные пользователями. Эту задачу можно решить, используя записи аудита.

Для решения задачи учтем, что оператор **delete** может быть отменен с помощью оператора вставки. Оператор **update** может быть отменен другим оператором обновления, а оператор **insert** может быть отменен с помощью оператора удаления.

Пример отмены команды удаления записей.

```
DECLARE @xml1 XML
SELECT @xml1
=xml_recsetFROM[audit].[log_table]WHERE
_type='DELETE';

WITHCaptured_Record
```

```
as
(
SELECT
Tbl.Col.value('PaymentID[1]','int')asPaymentID,
Tbl.Col.value('AccountID[1]','int')asAccountID,
Tbl.Col.value('DayOfPayment[1]','datetime')asDayOfPay
ment,
Tbl.Col.value('AmountOfPayment[1]','money')asAmount
OfPayment
FROM @xml1.nodes('//Record')Tbl(Col)
)
INSERTINTO[CRM].[Payments]
SELECT * FROMCaptured_Record;
GO
```

Пример отмены команды изменения записей.

```
DECLARE @xml2 XML
SELECT @xml2
=xml_recsetFROM[audit].[log_table]WHERE
_type='UPDATE';

WITHCaptured_Record
as
(
SELECT
Tbl.Col.value('PaymentID[1]','int')asPaymentID,
Tbl.Col.value('AccountID[1]','int')asAccountID,
Tbl.Col.value('DayOfPayment[1]','datetime')asDayOfPay
ment,
Tbl.Col.value('AmountOfPayment[1]','money')asAmount
OfPayment
FROM @xml2.nodes('//Record')Tbl(Col)
)
UPDATEorig
SET orig.AccountID=log.AccountID,
orig.DayOfPayment =log.DayOfPayment,
orig.AmountOfPayment=log.AmountOfPayment
FROM[CRM].[Payments]asorigJOINCaptured_Recordasl
og
ONorig.PaymentID=log.PaymentID
GO
```

Пример отмены команды **insert**.

```
DECLARE @xml3 XML
SELECT @xml3
=xml_recsetFROM[audit].[log]WHERE_type='INSERT';
WITHCaptured_Record
as
(
SELECTTbl.Col.value('PaymentID[1]','int')asPaymentID
FROM @xml3.nodes('//Record')Tbl(Col)
)
DELETE
FROM[CRM].[Payments]
WHEREPaymentI-
Din(SELECTPaymentIDFROMCaptured_Record)
GO
```

Данные процедуры могут быть легко адаптированы к требованиям избирательной отмены операций по различным критериям.

ЗАКЛЮЧЕНИЕ

Предложенный в данной статье подход имеет определенные преимущества по сравнению со стандартным аудитом SQL Server:

– фактически измененные данные зафиксированы в виде документа XML, их проще визуализировать и обрабатывать.

– любые нежелательные действия вставки, обновления или удаления могут быть отменены с использованием данных записи XML. Пример кода для каждого оператора DML был создан и протестирован.

Для больших таблиц, таблица аудита может быть создана для каждой информационной таблицы. Распределение данных аудита по нескольким таблицам позволит масштабировать систему. Такой подход применим как к локальной базе данных, так и к облачным базам данных, поскольку использует объекты, присутствующие в таких базах данных. Предложенный подход удовлетворяет более сложным бизнес-требованиям к аудиту.

ЛИТЕРАТУРА

[1] *SQL Server* [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008/dd392015 (v=sql.100)?redirecte dfrom=MSDN.

[2] *SQL_Server* [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/previous-versions/sql/sql-erver08/dd283095(v=sql.100)?redirected from=MSDN.

[3] *Security* [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/sql/relational-databases/security/contained-database-users-making-your-database-portable?view=sql-server-ver15.

[4] *SQL Server* [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/sql/t-sql/statements/create-role-transact-sql?view=sql-server-ver15.

[5] *Audit_triggers* [Електронний ресурс] – Режим доступу до ресурсу:https://github.com/ForAudit/SQLaudit/blob/master/Audit_triggers_proc.sql.

АУДИТ ЗМІН ТАБЛИЦЬ БАЗИ ДАНИХ SQL SERVER

Невід'ємним компонентом інформаційних систем є підсистема реєстрації і аудиту. Всі сучасні СУБД во-

лодіють можливістю реєструвати і обробляти інформацію про виконувані операції. SQL Server починаючи з версії з 2008 має можливість визначати специфікацію аудиту [1] на рівні сервера або бази даних. Однак дані системного аудиту не враховують вимоги бізнес-моделі інформаційної системи. Виникає необхідність настройки процесу реєстрації з урахуванням специфіки предметної області. Крім того, найважливішим завданням захисту БД є забезпечення цілісності даних. В сучасних складних ІС більшість таблиць повинні бути захищені від небажаних операцій змін (вставок, оновлень і вилучень). Дані аудиту можуть використовуватися для скасування таких небажаних дій. В цьому випадку, інформації в системних журналах недостатньо. У даній статті розглядається підхід до вирішення завдання аудиту змін в таблицях БД з метою запобігання небажаних змін даних. Такий підхід реалізований у вигляді методики створення об'єктів бази даних, за допомогою яких здійснюється реєстрація дій користувачів і скасування небажаних змін. Для вирішення завдання реєстрації всіх дій користувача пропонується використовувати окрему схему БД для аудиту, спеціальну таблицю аудиту і тригери інформаційних таблиць БД. Для скасування небажаних змін запропоновані SQL-процедури. Для кожного етапу методики наведена програмна реалізація, що дозволяє використовувати її як частину автоматизованої захисту БД.

Ключові слова: база даних, захист даних, реєстрація і аудит, відміна небажаних змін в базі даних.

AUDIT OF CHANGES TO SQL SERVER-DATABASE TABLES

The registration and audit subsystemis an inalienable component of information systems. All modern DBMS have the ability to register and process information about performedoperations. SQL Server since version 2008 has the ability to define an audit specification [1] at the server or database level. However, the data from the system audit does not consider the requirements of theinformation system's business model. It becomes necessary to customize the registration process considering the specifics of the subject area. In addition, the most important task of database protection is to ensure data integrity. In modern complex IS a number of tables must be protected from unwanted change operations (inserts, updates and deletions). Audit data can be used to cancel such unwanted actions. In this case, there is not enough information in the system logs. This article discusses an approach to solving the audit of changes to DB tables problem in order to prevent unwanted data changes. This approach is implemented in the form of a methodology of creating database objects with the help of which user actions are registered and unwanted changes can be canceled. To solve the problem of user actions registration it is proposed to use a separate database schema for audit, a special audit table and triggers of the DB information tables.SQL procedures are proposed for unwanted changes cancellation. The software implementation that makes it possible to use it as part of automated DB protection is given for each stage of methodology.

Key words: database, data protection, registration and audit, cancelation of unwanted changes in the database.

Коломиец Михайло Володимирович, кандидат технічних наук, доцент Фізико-технічного інституту НТУУ «КПІ».
E-mail: box144.85@gmail.com.
ORCID ID 0000-0001-8460-3041.

Коломыцев Михаил Владимирович, кандидат технических наук, доцент Физико-технического института НТУУ «КПИ».

Kolomytsev Myhailo, candidate of technical sciences, associate professor of Institute of Physics and Technologies of the NTUU "KPI".

Носок Світлана Олександрівна, кандидат технічних наук, доцент Фізико-технічного інституту НТУУ «КПІ».
E-mail: nos.sv.ol@gmail.com.
ORCID ID 0000-0002-0016-9346.

Носок Светлана Александровна, кандидат технических наук, доцент Физико-технического института НТУУ «КПИ».

Nosok Svitlana, candidate of technical sciences, associate professor of Institute of Physics and Technologies of the NTUU "KPI".

DOI: 10.18372/2410-7840.23.15153
УДК 004.056.5:61:621.397

АНАЛІЗ ПРОБЛЕМИ ЗАБЕЗПЕЧЕННЯ КІБЕРБЕЗПЕКИ МЕДИЧНИХ КОМП'ЮТЕРНИХ СИСТЕМ

Олена Трофименко, Ярослав Дубовой, Наталія Логінова, Юлія Прокоп, Олександр Задерейко

За умов суворого карантину через пандемію COVID-19, завдяки можливостям сучасних інформаційно-телекомунікаційних систем, значна частина медичних послуг трансформувала у цифрове середовище в режим онлайн. Позитивний ефект цього полягає насамперед у знищенні цифрового розриву та реалізації прав громадян на рівноправне отримання медичної допомоги в електронному форматі. Проте цей процес зумовив потенційні небезпеки витоків