Impact Factor:					
	ISRA (India) = 6.317	SIS (USA) = 0.912	ICV (Poland) = 6.630
	ISI (Dubai, UAE) = 1.582	РИНЦ (Russia) = 3.939	PIF (India) = 1.940
	GIF (Australia) = 0.564	ESJI (KZ) = 8.771	IBI (India) = 4.260
	JIF = 1.500	SJIF (Morocco) = 7.184	OAJI (USA) = 0.350

SOI: 1.1/TAS DOI: 10.15863/TAS

International Scientific Journal

Theoretical & Applied Science

p-ISSN: 2308-4944 (print) **e-ISSN:** 2409-0085 (online)

Year: 2022 **Issue:** 05 **Volume:** 109

Published: 15.05.2022 **http://T-Science.org**

Issue Article

Anastasia Eduardovna Ermakova
Peter the Great St. Petersburg Polytechnic University
Master's Student
Institute of Computer Science and Technology

Oleg Yurievich Sabinin
Peter the Great St. Petersburg Polytechnic University
Candidate of Engineering Sciences, Docent
Institute of Computer Science and Technology

Nikita Sergeevich Borovoy
OCS Distribution
Leader of Data Management Department

Olga Evgenievna Filatova
OCS Distribution
SQL-Developer

DEVELOPMENT OF A SYSTEM FOR AUTOMATIC DISTRIBUTION OF EMAIL NOTIFICATIONS ABOUT STATE OF THE DATA WAREHOUSE

Abstract: *The purpose of the article is to describe the developed system for automatic distribution of email notifications about state of the data warehouse.*

Key words: *Data warehouse, Business Intelligence, Data Quality, Email notifications.*

Language: *Russian*

Citation: Ermakova, A. E., Sabinin, O. Y., Borovoy, N. S., & Filatova, O. E. (2022). Development of a system for automatic distribution of email notifications about state of the data warehouse. *ISJ Theoretical & Applied Science, 05 (109)*, 301-320.

Soi: http://s-o-i.org/1.1/TAS-05-109-28 ***Doi***: https://dx.doi.org/10.15863/TAS.2022.05.109.28

Scopus ASCC: *1700.*

РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЧЕСКОЙ РАССЫЛКИ УВЕДОМЛЕНИЙ О СОСТОЯНИИ ХРАНИЛИЩА ДАННЫХ ПО ЭЛЕКТРОННОЙ ПОЧТЕ

Аннотация: *Целью статьи является описание разработанной системы автоматической рассылки уведомлений о состоянии хранилища данных по электронной почте.*

Ключевые слова: *Хранилище данных, BI аналитика, контроль качества данных, рассылки по электронной почте.*

Введение

Предметная область

Хранилище данных – предметно-ориентированная информационная база данных, специально разработанная и предназначенная для подготовки отчётов и бизнес-анализа с целью поддержки принятия решений в организации. Задача хранилища данных (Data Warehouse – DWH) – это создание единого интегрированного источника подготовленных данных для анализа информации из множества источников данных компании [1].

Для обозначения аналитических технологий и средств в целом принято использовать термин

Business Intelligence (BI). Понятие BI объединяет различные средства и технологии анализа и обработки данных масштаба предприятия. По оценкам международной исследовательской компании International Data Corporation, занимающейся изучением мирового рынка информационных технологий и телекоммуникаций, рынок BI состоит из 5 сегментов [2]:

1. средства построения хранилищ и витрин данных;
2. OLAP-продукты, позволяющие проводить многомерный анализ данных;
3. инструменты добычи данных;
4. управленческие информационные системы и приложения;
5. инструменты конечного пользователя для выполнения запросов и построения отчетов.

Одной из востребованных функций пятого сегмента является возможность осуществлять рассылку данных из базы данных по электронной почте указанным получателям по заданному расписанию. В IT-сфере наибольшую популярность имеют триггерные и контентные рассылки. Это объясняется тем, что в состав рассылок включаются актуальные статьи, кейсы, новости в какой-либо сфере (контентные) и уведомления о каких-либо событиях, действиях (триггерные), которые могут быть важны для оповещения сотрудников [3].

Создание инструмента для рассылки уведомлений по электронной почте является достаточно актуальной темой исследования. Для подтверждения этого можно рассмотреть примеры аналогичных исследований, которые были опубликованы за последние 2-3 года.

Так, например, в 2020 году группа разработчиков, представляющих общество с ограниченной ответственностью "Сапл-биз", предложила свой программный модуль email-рассылок, написанный на языке Python. Модуль является инструментом для создания, редактирования и массовой рассылки электронных писем пользователям электронной торговой платформы. Программа сокращает время, затрачиваемое на уведомление пользователей о новостях и специальных предложениях, позволяет автоматизировать процесс общения с клиентами и повысить уровень сервиса платформы [4].

Ещё одна работа по данной тематике была опубликована в 2021 году. В ней автор Трошин А. А. предлагает создание процедуры для рассылки электронных писем (e-mail) с помощью средств Microsoft SQL Server Management Studio. По мнению автора, разработанная процедура должна помочь в организации работы с документооборотом [5].

Таким образом, можно заметить высокую востребованность почтовых рассылок при организации общения с клиентами, сотрудниками предприятия и для рекламы. Но электронная почта может быть не менее полезна для проверки состояния самой базы данных. Так, например, по логированию событий, происходящих при загрузке новых данных, можно отследить сбои и уведомить разработчика о них.

Постановка проблемы

Хранилища данных являются главным ИТ-ресурсом, существующим в крупных компаниях. Его проектирование, разработка и поддержка требуют крупных вложений денежных средств, а получаемый результат позволяет иметь в быстром доступе огромное количество информации о бизнесе [6]. Правильно построенное хранилище и грамотная интеграция с остальными системами позволяют оптимизировать многие бизнес-процессы компании на любом уровне [7, с.192]. Соответственно, любые ошибки в данных могут привести к ошибочным выводам, спровоцировать принятие неправильных корпоративных решений и негативно сказаться на финансовом благополучии компании.

Чтобы принимать верные решения на основе полученной аналитики, входные данные для неё (то есть данные в хранилище) должны быть качественными. Обеспечение качества данных происходит путём его оценки и улучшения [8]. Как правило, эффективнее всего оценить качество тех или иных данных может человек, который с этими данными работает и отвечает за них.

Так, технические сбои, отражённые в системных таблицах, сможет обработать администратор базы данных или разработчик. Проблемы в данных по продажам скорее всего сможет решить специалист по продажам. И именно этот специалист лучше всех знает, какие в принципе проблемы в данных в его сфере ответственности могут возникнуть.

Важно своевременно и точно доставить проблемные данные по нужному адресу. Притом адресат должен иметь возможность самостоятельно задавать правила, по которым качественные данные отличаются от «некачественных», а при необходимости эти правила изменять и дополнять.

Предложения по решению

Для решения данной проблемы хорошо подходит механизм почтовых рассылок, относящийся к одному из сегментов BI.

Существует множество компаний, которые предоставляют свои решения по контролю качества данных, в числе возможностей которых присутствуют автоматические рассылки. Лидирует в этой области Informatica, чьи

программные продукты обладают широким кругом возможностей и удобным интерфейсом [9]. Однако стоимость этих продуктов достаточно высока, а из большого числа функций востребованы обычно далеко не все.

Некоторые BI продукты также дают возможность организации рассылки информации из базы данных. Примером может служить Oracle BI Publisher, являющийся одним из компонентов Oracle Business Intelligence. Oracle BI Publisher позволяет автоматизировать отправку аналитических отчётов по электронной почте. Процесс создания отчётов прост и быстр, что очень важно в случае крупных организаций, заинтересованных в своей конкурентоспособности [11]. Однако данный программный продукт не имеет бесплатной пробной версии, а покупка лицензии может быть затруднительна.

Существуют также и бесплатные решения с открытым исходным кодом. Однако их интеграция в существующую систему, а также вопросы безопасности и поддержки часто делают их не слишком привлекательными в глазах пользователей.

В качестве альтернативного варианта можно попробовать создать свою систему, направленную на решение задачи информирования пользователя о состоянии хранилища данных, которая присутствует в любой организации, имеющей хранилище данных и строящей аналитику по нему.

В данной работе предлагается решение проблемы своевременного информирования пользователя о состоянии хранилища данных путём создания системы автоматической рассылки уведомлений по электронной почте.

Рассылка информации пользователям должна происходить регулярно по заданному расписанию. Перед рассылкой должна происходить проверка хранилища на предмет появления новых инцидентов (ошибок в данных или проблем с объектами БД), изменения или закрытия старых инцидентов, если они были исправлены.

Проверка реализуется путём выполнения SQL запроса, который вносится разработчиком в одну из настроечных таблиц системы и выполняется регулярно. Каждая строка, полученная в результате выполнения этого запроса, попадает в реестр инцидентов, где ей присваивается определенный статус.

История изменения статусов каждого инцидента должна сохраняться, используя при этом минимальный объём памяти для логирования. Это позволит в дальнейшем построить аналитический отчёт об эффективности использования данной системы.

Пользователь должен иметь возможность запросить информацию об инцидентах, находящихся в любых вариациях статусов. Также ему должна быть предоставлена возможность настроить рассылку под свои нужды, изменяя её название, сопроводительный текст в письме, видимость колонок, состав получателей рассылки, время отправки и значения параметров, если они были заданы в запросе.

Предполагается, что для удобства пользователей предложенная система будет дополнена неким графическим интерфейсом. Однако в рамках данной работы создание интерфейса не рассматривается. Взаимодействие пользователя и системы происходит через хранимые процедуры в БД.

Цель и задачи исследования

Целью данной работы является описание разработанной системы автоматической рассылки уведомлений о состоянии хранилища данных по электронной почте.

В рамках работы будет:

- предложена архитектура системы и логика её работы;
- описаны таблицы для настройки обновления реестра инцидентов;
- описаны возможные статусы инцидентов и переходы между ними;
- описана логика работы процедуры обновления реестра инцидентов;
- описаны таблицы для настройки рассылки данных из реестра инцидентов;
- описана логика работы процедуры рассылки;
- приведен пример заполнения настроечных таблиц и функционирования системы;
- приведена статистика использования системы в компании.

Архитектура системы

Архитектура разработанной системы не привязана ни к какой конкретной СУБД. Логика её работы также подразумевает работу в любой реляционной СУБД, которая используется в качестве хранилища в компании. Программный код написан на языке T-SQL – процедурном расширении языка SQL, созданном компанией Microsoft (для Microsoft SQL Server). Большинство конструкций, используемых в нём, в том или ином виде присутствуют в любой реляционной СУБД. Поэтому для разработчика баз данных будет возможным переписать этот программный код под нужды конкретной организации.

На рисунке 1 представлена архитектура системы, отображающая основные логические особенности её работы. Визуально рисунок делится на три блока – верхний, нижний и левый – что подразумевает запуск трёх различных

хранимых процедур. Блоки начала и завершения работы каждой из них выделены белым цветом.

Работа системы начинается при условии заполнения ряда настроечных таблиц. После этого может быть запущена первая процедура UpdateIncidentRegistry, которая производит поиск инцидентов в хранилище данных (блок БД Data Warehouse) и обновляет реестр инцидентов (блок БД DQManagement в центре) в соответствии с найденной информацией. Реестр инцидентов – это несколько таблиц БД, которые в совокупности реализуют идею хранения информации о нежелательных событиях (инцидентах), придерживаясь принципа отсутствия дублей и логирования состояния каждого инцидента в течение всего периода его существования. Работа данной процедуры отражена в верхнем графическом блоке (выше пунктирной линии) и будет описана детально позднее.

Как только в реестр инцидентов вносятся новые сведения о состоянии хранилища данных, свежая информация об инцидентах может быть отправлена пользователям. Готовность рассылки к отправке проверяется каждые 15 минут с помощью процедуры StartSendingByEvent (левый графический блок на схеме).

Для каждой готовой рассылки процедура StartSendingByEvent запускает отправку, которая производится с помощью процедуры MailingIncidentRegistry (нижний графический блок). Кроме того, отправка может быть произведена путём отдельного вызова процедуры MailingIncidentRegistry с указанием номера рассылки в качестве входного параметра.

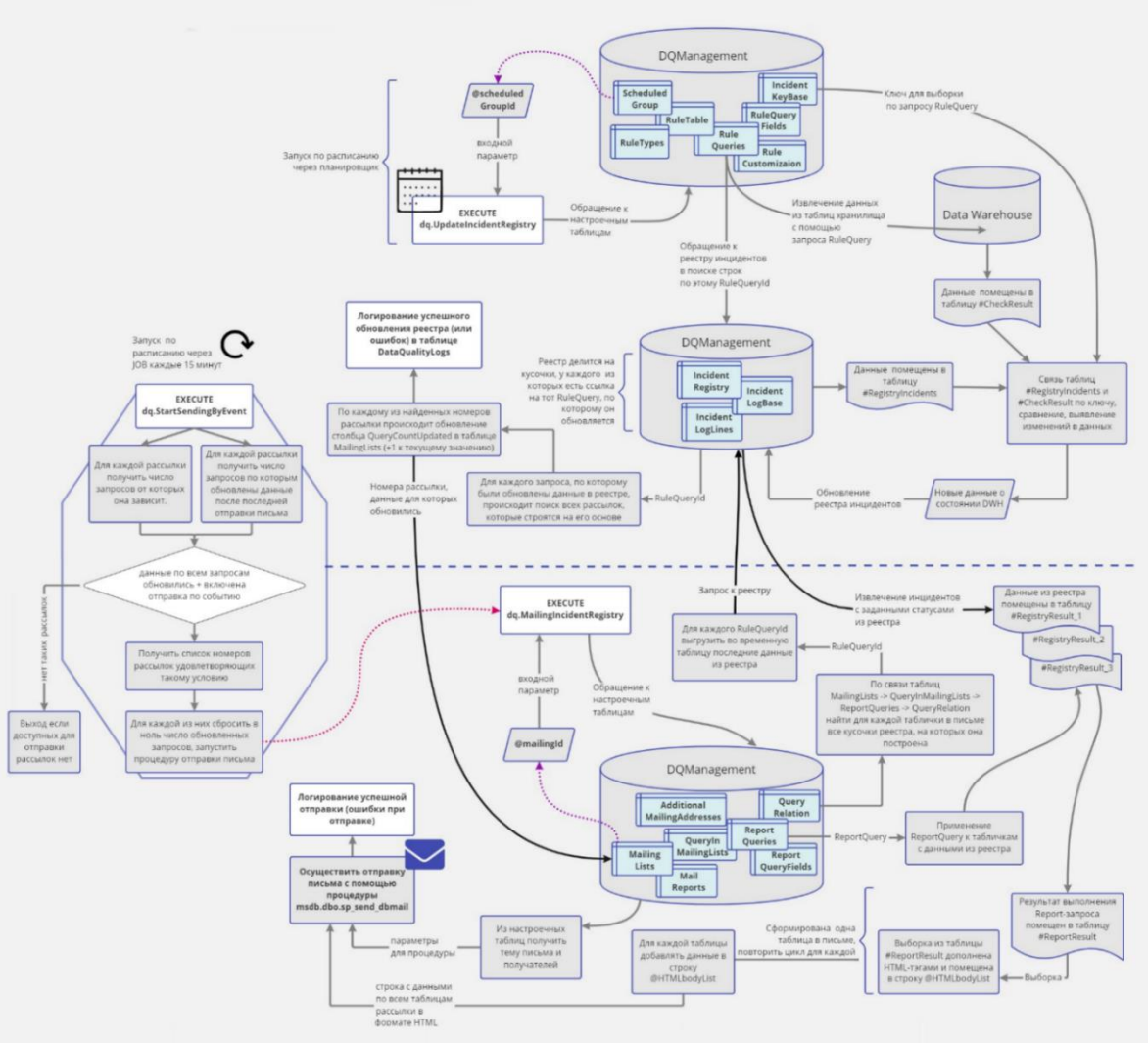


Рисунок 1 – Архитектура системы

На рисунке 2 представлена схема базы данных DQManagement, на основе которой работает система. Все настроечные таблицы, которые участвуют в заполнении реестра инцидентов, выделены синими рамками. Таблицы реестра инцидентов выделены фиолетовыми рамками. Все таблицы, которые требуется заполнить для настройки рассылки, выделены на

схеме оранжевыми рамками. Таблица DataQualityLogs не выделена цветом, так как с ней работает и процедура обновления реестра, и процедура рассылки. В этой таблице фиксируется информация о результатах выполнения процедур обновления реестра и рассылки.

Таблицы для настройки обновления реестра инцидентов

В процедуре обновления реестра UpdateIncidentRegistry используется ряд настроечных таблиц.

1. Таблица RuleTypes содержит типы проверок: техническая проверка или бизнес-проверка. Перед запуском процедуры требуется заполнить эту таблицу.

2. В таблице ScheduledGroup описаны группы для запуска проверок по расписанию. Номер группы является входным параметром в процедуре, осуществляющей проверку и обновление реестра. Группа запуска привязывается к определённому расписанию, по которому будет происходить запуск проверки. Перед запуском процедуры в таблице ScheduledGroup должна быть как минимум одна группа запуска, например, каждый день в 9 утра

3. Таблица RuleTable содержит информацию о созданном правиле, в том числе ссылки на рассмотренные ранее таблицы, то есть каждое правило имеет свой тип и свою группу запуска. Перед запуском процедуры в таблице RuleTable должно быть описано как минимум одно правило.

4. По каждому правилу в реестр инцидентов должен быть внесен набор данных. Условие отбора этих данных прописано в SQL запросе, который находится в таблице RuleQueries. Каждый запрос имеет ряд полей и ключ, который может быть составным. Обновление инцидентов происходит по ключу, поэтому очень важно, чтобы в каждом запросе соблюдались свойства первичного ключа – уникальность и отсутствие NULL-значений. Иногда бывает такое, что в выборке по тем или иным причинам ключ выделить невозможно. В этом случае можно предпринять ряд действий:

- Внести изменения в таблицу, к которой обращается запрос (это возможно только при согласовании и создании задачи).
- Внести изменения в запрос, например, заменить NULL значения или исключить часть строк из выборки.
- Разбить запрос на два-три отдельных запроса, каждый из которых возвращает выборку, в которой выполняются свойства первичного ключа.

В связи с этим, к каждому правилу может относиться один или несколько Rule-запросов. Перед запуском процедуры в таблице RuleQueries должен быть как минимум один запрос, связанный с правилом в таблице RuleTable.

5. Поля, которые содержатся в разделе SELECT каждого Rule-запроса, должны быть описаны в таблице RuleQueryFields. Важно чтобы число полей, ссылающихся на конкретный запрос, в таблице и в этом запросе совпадало. В запросе может участвовать такое поле, значения которого изменяются каждый день, например, курс валюты. Так как при изменении информации в инциденте, он меняет статус, логично указать, что изменения в таких полях не должны провоцировать смену статуса инцидента (иначе каждый день он будет отмечен как измененный, что будет вводить пользователей в заблуждение). Для того чтобы указать поля, изменения в которых не влияют на смену статуса, используется столбец NotChangeStatusIncident.

6. В таблице IncidentKeyBase указаны поля, которые являются ключевыми для того или иного Rule-запроса. Названия должны совпадать с названиями из таблицы RuleQueryFields. Использование ключей позволяет не перезаписывать результаты выполнения запросов каждый день и не дублировать их.

7. В каждом запросе может содержаться один или несколько параметров, которые определены в таблице RuleCustomization. Под параметром понимается некоторое значение, которое участвует в разделе WHERE и может быть изменено по требованию бизнес-пользователя. Например, «вывести строки, где количество больше 0» и «вывести строки, где количество больше 1». В этом случае 0 и 1 являются значениями параметра, название которого должно быть определено пользователем и задано в запросе вместо конкретных значений. В процессе работы процедура проверки обнаружит параметр в запросе и заменит его на соответствующее значение из таблицы RuleCustomization. Таблица RuleCustomization не является обязательной для заполнения. Значения там должны быть только для тех запросов, в коде которых используются переменные параметры.

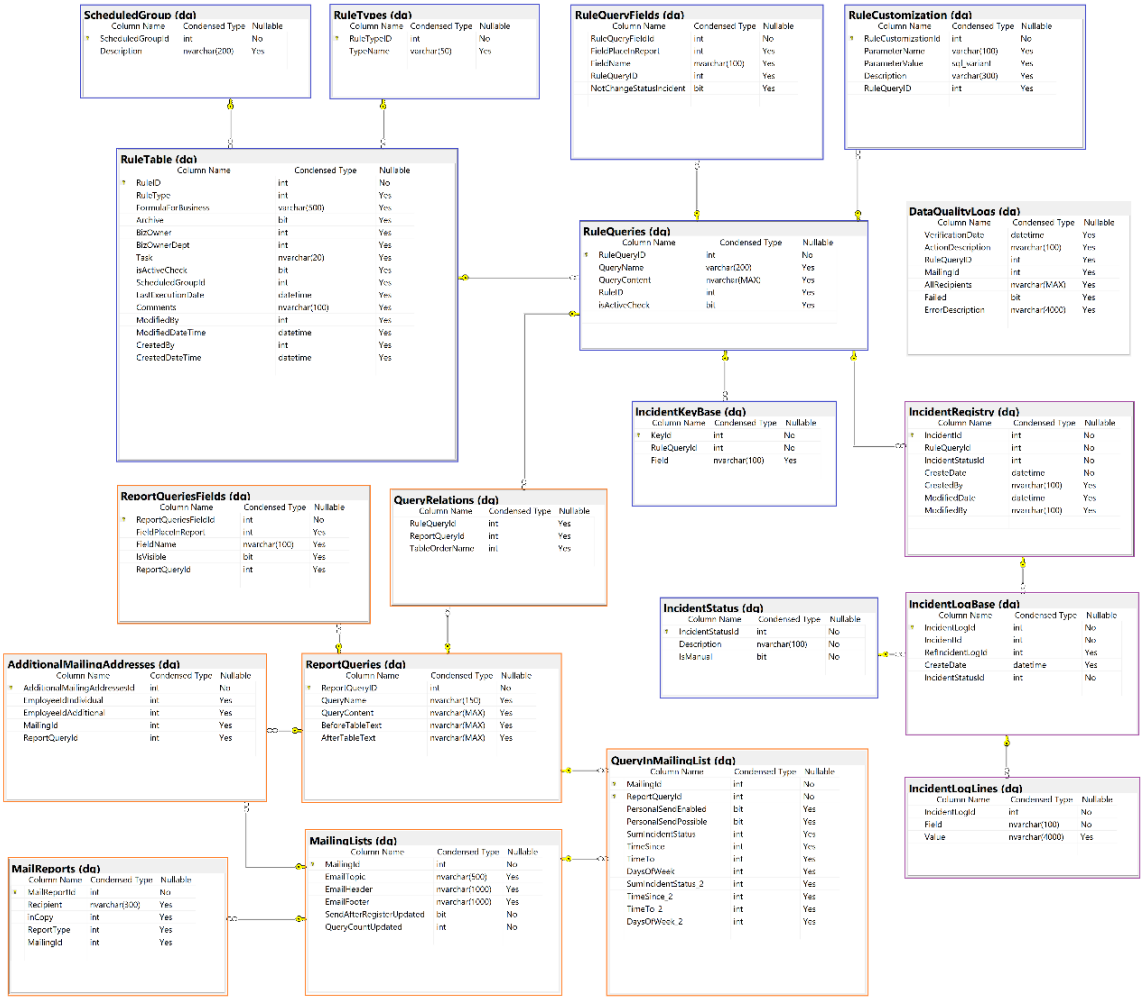


Рисунок 2 – Схема БД

Таблицы реестра инцидентов

Таблицы IncidentRegistry, IncidentLogBase и IncidentLogLines в совокупности являются таблицами для хранения информации о нежелательных событиях (инцидентах). С помощью ключа, определённого для каждого запроса, реализуется отсутствие дублей и логирование состояния каждого инцидента в течение всего периода его существования.

Состояние выражается рядом статусов, определённых в таблице IncidentStatus, которые обязательно должны быть внесены в таблицу перед запуском процедуры так, как показано на рисунке 3.

Идентификаторы статусов являются степенями двойки, что позволяет получить желаемый набор статусов при рассылке, указывая только одно число – сумму их идентификаторов. Подробнее этот технический момент будет раскрыт при описании работы процедуры рассылки.

	IncidentStatusId	Description	IsManual
1	1	Исключение	1
2	2	Новый	0
3	4	Открыт	0
4	8	Изменен	0
5	16	Закрыт	0
6	32	Переоткрыт	0

Рисунок 3 – Список статусов инцидентов

1. В таблице IncidentRegistry каждая строка представляет собой инцидент в текущем его состоянии. В каждой строке можно увидеть уникальный номер IncidentId, ссылку на запрос, по которому этот инцидент был отобран (RuleQueryId), текущий статус инцидента в виде ссылки на таблицу IncidentStatus, а также дату первого выявления инцидента и дату последнего изменения его состояния.

2. В таблице IncidentLogBase на каждый инцидент может приходиться несколько записей, каждая из которых является свидетельством об изменении состояния инцидента. То есть в данной таблице указан не только текущий статус инцидента, но и все статусы, в которых данный инцидент когда-либо побывал. Каждая строка содержит ссылку на номер инцидента в таблице IncidentRegistry, собственный первичный ключ IncidentLogId, дату создания записи, статус, в который инцидент перешёл в указанную дату, и ссылку на следующую запись состояния. Для последней записи вместо ссылки указан NULL.

3. Как уже было сказано ранее, в большинстве случаев изменение статуса связано с изменениями в данных. Для каждого изменения состояния в таблице IncidentLogLines хранятся данные, которые были изменены. Проблема хранения заключается в том, что состав и число полей, возвращаемых каждым запросом, различны. Поэтому каждая результирующая строка должна быть транспонирована и распасться на такое число строк, каково было число столбцов. Каждая полученная строка будет иметь ссылку на соответствующую строку в таблице IncidentLogBase.

Взаимодействие с таблицами реестра осуществляется только процедурой UpdateIncidentRegistry. Запрещено изменять, удалять или добавлять значения в них вручную.

Описание статусов инцидентов и переходов между ними

Остановимся подробнее на значениях каждого статуса и на условиях перехода из одного статуса в другой.

Статус «Новый» может быть присвоен инциденту всего один раз в течение всего периода логирования. Этот статус говорит о том, что строка с указанным ключом никогда раньше не появлялась в данной выборке. Даже если строка исчезнет из неё, и инцидент перейдёт в статус «Закрыт», а потом спустя некоторое время снова вернётся, то он всё равно не будет «Новым». Вместо этого инцидент получит статус «Переоткрыт». «Переоткрыт» – это единственный статус, в который инцидент может перейти из статуса «Закрыт».

В свою очередь в статус «Закрыт» может перейти любой инцидент, кроме уже находящегося в этом статусе. Закрытым является инцидент, который не был обнаружен в выборке данных, возвращенной запросом.

Инцидент в статусе «Исключение» тоже может быть «Закрыт», но при этом важно помнить, что если он вновь появится в выборке, то пользователю снова придёт оповещение о нём и потребуется вновь выставлять статус «Исключение». Статус «Исключение» является единственным статусом, который выставляется пользователем и обозначает, что указанная строка ошибкой не считается, хоть она и попадает под условия запроса.

Инцидент, находящийся в статусе «Исключение» может быть «Переоткрыт», если в строке инцидента изменилось одно из важных (влияющих на статус) полей. Например, в поле, по которому происходит отбор инцидентов в конкретном запросе. Если же изменилось одно из второстепенных, не влияющих на смену статуса полей, то происходит логирование изменений, однако инцидент остаётся в статусе «Исключение». Разделение полей на влияющие на изменения статуса и не влияющие на него происходит с помощью указания True/False в столбце NotChangeStatusIncident в таблице RuleQueryFields.

Для инцидентов, находящихся в любом другом статусе кроме «Закрыт» и «Исключение», смена значения в поле, влияющем на статус, обозначает переход в состояние «Изменен».

В статус «Открыт» инцидент может попасть в двух случаях – либо в нём по сравнению с последним запуском не изменилось ничего, либо изменилось значение в поле/полях, которые на смену статуса не влияют. Для первой ситуации логирование произойдёт только один раз, при этом значения полей сохраняться не будут. Для второй ситуации логирование будет происходить каждый раз, а в таблице IncidentLogLines будет сохраняться каждое измененное поле.

Все возможные переходы отражены на графе состояний инцидента, представленном на рисунке 4.

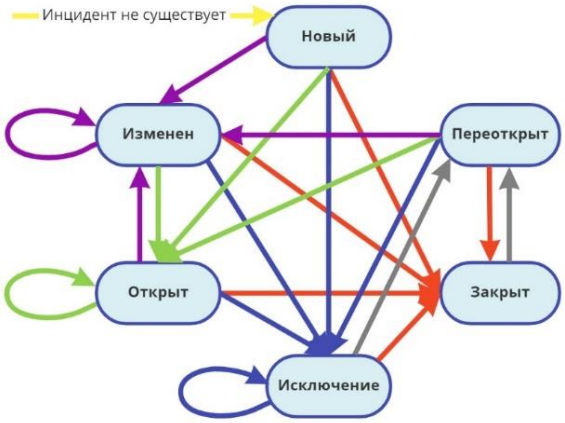


Рисунок 4 – Граф состояний инцидента

Дополним графическое отображение возможных переходов табличной интерпретацией (см. таблицу 1).

Таблица 1. Условия перехода инцидента между статусами

	В какой статус перейдёт инцидент, если…				
Текущий статус	строка исчезла из выборки	пользователь установил статус «Исключение»	строка пришла без изменений относительно предыдущего лога	строка пришла с изменением в полях, **не влияющих** на статус	строка пришла с изменением в полях, **влияющих** на статус
Новый	Закрыт	Исключение	Открыт	Открыт	Изменен
Исключение	Закрыт	----------	----------	Исключение	Переоткрыт
Закрыт	---------	----------	Переоткрыт	Переоткрыт	Переоткрыт
Переоткрыт	Закрыт	Исключение	Открыт	Открыт	Изменен
Открыт	Закрыт	Исключение	----------	Открыт	Изменен
Изменен	Закрыт	Исключение	Открыт	Открыт	Изменен

Прочерк обозначает отсутствие логирования на уровне всех таблиц реестра. Запись о возможности перехода говорит главным образом о том, что при переходе информация фиксируется в таблицах IncidentRegistry и IncidentLogBase. А вот сами значения в таблицу IncidentLogLines сохраняются не при всех переходах, а только там, где это необходимо и возможно.

Так, например, при переходе в статус «Закрыт» нет возможности узнать те значения, с которыми строка в выборку не попала (возможно, что даже в самой проверяемой таблице этой строки уже нет). При установке статуса «Исключение» проблема аналогична – на момент установки статуса пользователем нет данных о том, не изменилось ли что-то в этой строке. А вот при переходе в статус «Открыт» (в случае отсутствия любых изменений) перезаписывать все значения ещё раз просто нет смысла, хотя сделать это можно.

Полная информация о процессе логирования инцидента в зависимости от статуса, в который он переходит отражена в таблице 2.

Таблица 2. Логирование изменений статуса инцидента

Произошёл переход инцидента в статус	**Что произойдёт в таблице**		
	IncidentRegistry	**IncidentLogBase**	**IncidentLogLines**
Новый	Будет вставлена одна новая строка, в которой дата создания будет	Будет вставлена одна новая строка со статусом «Новый» и ссылкой на ID строки в таблице IncidentRegistry.	Будет вставлено столько строк со ссылкой на ID строки в таблице

	равна дате последнего изменения, установлен статус «Новый».		IncidentLogBase, сколько в строке инцидента было столбцов.
Исключение	В строке будет обновлен статус и дата последнего изменения.	Будет вставлена одна новая строка со статусом «Исключение» и ссылкой на ID строки в таблице IncidentRegistry. В предыдущей строке таблицы, имеющей ссылку на данный инцидент, колонка RefIncidentLogId будет обновлена – вместо NULL в ней будет установлен идентификатор только что вставленной строки.	При переходе из статусов «Новый», «Изменен», «Открыт», «Переоткрыт» (то есть при установке статуса «Исключение» пользователем) не изменяется.
			При логировании изменений не влияющих на статус полей (то есть при переходе из статуса «Исключение») будет вставлено столько строк, сколько полей было изменено.
Закрыт	В строке будет обновлен статус и дата последнего изменения.	Будет вставлена одна новая строка со статусом «Закрыт» и ссылкой на ID строки в таблице IncidentRegistry. В предыдущей строке таблицы, имеющей ссылку на данный инцидент, колонка RefIncidentLogId будет обновлена – вместо NULL в ней будет установлен идентификатор только что вставленной строки.	Не изменяется.
Переоткрыт	В строке будет обновлен статус и дата последнего изменения.	Будет вставлена одна новая строка со статусом «Переоткрыт» и ссылкой на ID строки в таблице IncidentRegistry. В предыдущей строке таблицы, имеющей ссылку на данный инцидент, колонка RefIncidentLogId будет обновлена – вместо NULL в ней будет установлен идентификатор только что вставленной строки.	Будет вставлено столько строк со ссылкой на ID строки в таблице IncidentLogBase, сколько в строке инцидента было столбцов.
Открыт	В строке будет обновлен статус и дата последнего изменения.	Будет вставлена одна новая строка со статусом «Открыт» и ссылкой на ID строки в таблице IncidentRegistry. В предыдущей строке таблицы, имеющей ссылку на данный инцидент, колонка RefIncidentLogId будет обновлена – вместо NULL в ней будет установлен идентификатор только что вставленной строки.	При переходе из статуса «Новый», «Изменен», «Переоткрыт» (то есть если никакое поле своё значение не изменило), ничего вставлено не будет.
			При переходе из статуса «Открыт» (то есть при изменении полей, не влияющих на статус) будет вставлено столько строк, сколько полей было изменено.
Изменен	В строке будет обновлен статус и дата последнего изменения.	Будет вставлена одна новая строка со статусом «Изменен» и ссылкой на ID строки в таблице IncidentRegistry.	Будет вставлено столько строк со ссылкой на ID строки в таблице

		В предыдущей строке таблицы, имеющей ссылку на данный инцидент, колонка RefIncidentLogId будет обновлена – вместо NULL в ней будет установлен идентификатор только что вставленной строки.	IncidentLogBase, сколько полей было изменено.

Логика работы процедуры обновления реестра инцидентов

После подготовки таблиц процедура обновления реестра UpdateIncidentRegistry может быть запущена.

На вход она принимает единственный параметр – группу запуска. В группе запуска, как правило, находится несколько проверок, поэтому открывается цикл (курсор) по проверкам одной группы. Действия, выполняемые для одной проверки, должны быть или приняты, или отклонены все вместе, то есть должны быть выполнены в рамках одной транзакции.

Далее открывается курсор по запросам одной проверки. Для каждого запроса нужно сначала проверить наличие параметров в таблице RuleCustomization и, если они есть, заменить их на соответствующие значения.

Каждый запрос имеет свой набор выводимых полей, поэтому следующим шагом программа обращается к таблице RuleQueryFields и формирует строку с названиями полей через разделитель. Далее эта строка будет использована при создании временной таблицы #CheckResult, в которую будет загружена полученная при запуске запроса выборка.

Далее содержимое таблицы #CheckResult должно пройти ряд проверок. Поскольку реестр инцидентов требует соблюдения определённых правил (уникальность вставляемых строк по ключу, отсутствие пропущенных значений в ключевых полях), выборка, которую планируется вставлять в реестр, должна быть проверена. Если в результате проверки были обнаружены ошибки, то генерируется исключение, выполнение программы прекращается, пользователю выводится сообщение о том, каким условиям не удовлетворяет набор данных.

После успешного прохождения проверок данные в таблице #CheckResult готовы к внесению в реестр инцидентов. Для того чтобы понять, какие строки должны быть вставлены, а какие уже есть в реестре и требуют обновления, нужно сформировать ещё одну временную таблицу #RegistryIncidents. Она будет содержать последнюю информацию из реестра инцидентов по данному запросу.

Для того чтобы сформировать таблицу #RegistryIncidents и заполнить её данными, используется та же строка с названиями полей, что использовалась для формирования таблицы #CheckResult. Данные из таблиц реестра отбираются по максимальному времени и транспонируются с помощью оператора PIVOT.

После того как обе временные таблицы заполнены данными, можно начинать их сравнение. Для этого из таблицы IncidentKeyBase процедура отбирает ключевое поле (или поля) для данного запроса. По полученному ключу происходит соединение таблиц. Используется два различных сценария их соединения: INNER JOIN и FULL JOIN.

FULL JOIN используется для выявления новых строк, которых ещё нет в реестре, и для обнаружения в реестре тех строк, которые пропали из выборки. В первом случае в таблицы реестра вносится информация о новых инцидентах. Во втором случае, подходящие под условие строки получают статус «Закрыт».

INNER JOIN используется для обработки ряда более сложных ситуаций, при которых строка с определённым значением ключа была найдена и в результатах проверки (в таблице #CheckResult), и в реестре (в таблице #RegistryIncidents).

Во-первых, рассмотрим наиболее простой случай из этой категории – предположим, что совпали абсолютно все значения в строках в обеих таблицах. В этом случае, если инцидент ранее находился в статусе «Новый», «Изменен», «Переоткрыт», то он должен сменить статус на «Открыт». В таблицу IncidentLogBase вставится одна новая строка, в таблице IncidentRegistry текущий статус изменится на «Открыт». В таблицу IncidentLogLines новых значений записано не будет.

Во-вторых, может быть, что строка в реестре хоть и была обнаружена, но её последний статус «Закрыт» (то есть один или несколько раз в выборке по запросу строка с указанным ключом отсутствовала, а теперь появилась вновь). В этом случае инцидент получает статус «Переоткрыт», а все его поля логируются полностью. То же самое касается и инцидентов со статусом «Исключение», у которых изменилось хоть одно влияющее на статус поле по сравнению с последним логом.

В-третьих, в результате проверки в строке могло быть обнаружено, что изменилось одно или несколько влияющих на статус значений. Если ранее инцидент находился в статусе «Открыт», «Новый», «Изменен» или «Переоткрыт», то он будет переведён в статус «Изменен». В таблицу IncidentLogBase вставится одна новая строка, в таблице IncidentRegistry изменится текущий

статус, а в таблицу IncidentLogLines будет записано столько строк, сколько было изменено полей по сравнению с последней информацией, полученной из реестра.

В-четвёртых, измениться могло значение только в поле (или полях), не оказывающем влияния на статус (например, курс валюты, который меняется каждый день). В этом случае инциденты, имевшие статус «Новый», «Изменен», «Переоткрыт» или «Открыт» перейдут в статус «Открыт», а в таблице IncidentLogLines будут сохранены изменённые значения. Если же инцидент имел статус «Исключение», то изменённые значения также будут сохранены, а новая строка, вставленная в лог, будет иметь статус «Исключение».

Кроме того, возможны ситуации, когда в одном инциденте одновременно изменились и влияющие на статус поля, и не влияющие на него. Приоритет имеет изменение, влияющее на статус – в логике процедуры его обработка расположена выше. Если после обработки этого изменения будут обнаружены так же новые значения в полях, не влияющих на статус, то они будут сохранены со ссылкой на тот же IncidentLogId, что и поля, влияющие на статус.

Каждая из приведённых ситуаций в тексте процедуры обрабатывается с помощью динамически собранного курсора по соединённым между собой по ключу таблицам #CheckResult и #RegistryIncident.

По мере прохождения по курсорам генерируются текстовые сообщения о том, сколько строк выборки по какому условию было обработано. Эта справочная информация выводится на экран после завершения работы программы. После успешного завершения обработки выборки данных по одному запросу в таблицу DataQualityLogs вносится информация о том, что данные в реестре по указанному запросу были успешно обновлены и указывается время обновления.

После этого курсор по запросам одного правила закрывается. В рамках одного правила вызывается обработчик ошибок. В случае, если в одном из запросов на каком-либо из этапов произошла ошибка, то откатываются изменения, произведённые по всем запросам этого правила, ошибка сохраняется в таблицу DataQualityLogs, разработчику отправляется письмо с сообщением об ошибке. Процедура при этом переходит на обработку следующего правила в вызванной группе запуска.

Таблицы для настройки рассылки данных из реестра инцидентов

В процедуре рассылки данных из реестра MailingIncidentRegistry используется ряд настроечных таблиц (на рисунке 2 выделены оранжевым цветом).

1. В таблице MailingLists задаются параметры письма, а именно заголовок письма (поле EmailTopic), текст до и после основной информации (поля EmailHeader и EmailFooter соответственно). Поле SendAfterRegisterUpdated является булевым и определяет, будет ли письмо отправляться сразу же после загрузки новой информации в реестр или же отправка будет определена отдельным расписанием. Поле QueryCountUpdated является техническим и определяет число Rule-запросов, на которых строится конкретное письмо. Изменение значения в этом поле происходит только через процедуру. При создании новой строки в данном поле всегда выставляется значение 0.

2. В таблице ReportQueries в полях QueryName и QueryContent хранится название и текст запроса. Этот запрос обращается к конкретному участку реестра инцидентов, транспонирует данные и выводит их в виде таблицы, которая впоследствии будет в письме. Поля BeforeTableText и AfterTableText определяют подписи перед и после таблицы. Поля PersonalSendEnabled и PersonalSendPossible определяют включена ли и возможна ли построчная отправка для этого запроса.

3. Таблица ReportQueriesFields содержит описание полей для каждого запроса из ReportQueries. В столбце FieldName следует указывать название поля так, как оно должно называться в письме, а в столбце FieldPlaceInReport номер этого поля по порядку слева направо в таблице. Столбец ReportQueryId содержит ссылку на запрос в таблице ReportQueries. Для каждого запроса может быть определено одно или несколько полей, каждое поле однозначно относится к конкретному запросу. Важно, чтобы число полей в разделе SELECT запроса соответствовало числу полей, указанному в таблице ReportQueriesFields для этого запроса.

Столбец IsVisible содержит булево значением и определяет является ли описываемое поле видимым в письме или нет. Это полезно, например, для построчной рассылки или же в том случае, если поле нужно убрать из письма временно.

Для построчной рассылки в запросе и связанных с ним полях обязательно должен присутствовать столбец с названием LetterIndividualRecipient, который содержит Email ответственного/ответственных.

4. Таблица QueryInMailingList определяет связь между рассылкой (ссылка на MailingLists) и запросом (ссылка на ReportQueries). Один и тот же Report-запрос может участвовать в разных рассылках. В одном письме может быть несколько

таблиц, каждая из которых сформирована своим Report-запросом. Столбец SumIncidentStatus позволяет указать сумму статусов инцидентов. Инциденты по конкретному запросу, имеющие статус из этой суммы, будут добавлены в указанное письмо. Так, например, запрос №1 может быть включён в рассылки №7 и №8, но в рассылке №7 по нему будут отправлены только новые инциденты, а в рассылке №8 и новые и открытые ранее. Статусы и их численные обозначения описаны в таблице IncidentStatus.

5. Таблица QueryRelations определяет связь Report-запроса с Rule-запросом, то есть указывает из какой части реестра инцидентов определённый Report-запрос будет брать данные. На основе данных, внесённых в реестр одним Rule-запросом, может строиться несколько разных Report-запросов. Один Report-запрос должен иметь возможность обратиться к нескольким разным блокам данных в реестре.

Например, Report-запрос №3 подсчитывает число инцидентов по цене товара, которые в реестр внёс Rule-запрос №5, и число инцидентов по виду товара, которые в реестр внёс Rule-запрос №6. В этом случае обращаясь в Report-запросе к реестру, мы не можем использовать только одну временную таблицу, которая содержит все данные, поскольку выборка, возвращаемая разными Rule-запросами неодинакова по числу и составу столбцов. Использование двух и более временных таблиц в одном запросе требует задания очерёдности их использования. Например, пусть первая временная таблица содержит данные из реестра по Rule-запросу №5, а вторая по Rule-запрос №6. Столбец TableOrderName в таблице QueryRelations позволяет задать эту очерёдность.

6. Таблица MailReports содержит список всех получателей для каждой рассылки, определённой в таблице MailingLists. Каждый получатель вносится отдельной строкой. Столбец Recipient содержит Email получателя. Столбец inCopy определяет будет ли этот адресат добавлен в список получателей копии письма. Столбец ReportType определяет будет ли письмо отправлено адресату, если все Report-запросы, связанные с этим письмом, вернули пустую выборку.

Если для запроса включена построчная рассылка, это значит, что все строки, возвращённые этим запросом, будут отправлены не только по тем адресам, которые указаны в таблице MailReports, но и людям ответственным за каждый конкретный инцидент.

7. Каждый ответственный может захотеть не только самостоятельно получать информацию о «своих» инцидентах, но и привлечь к этому помощников. С этой целью была создана таблица AdditionalMailingAddresses, которая содержит идентификаторы ответственного и помощника (ссылки на справочник сотрудников), а также ссылку на Report-запрос, в котором фигурирует ответственный.

Логика работы процедуры рассылки

На вход процедура MailingIncidentRegistry принимает номер отправляемой рассылки @mailingId и возвращает описание ошибки, если она произошла в ходе работы процедуры.

В переменную @strStyle записывается строка, содержащая HTML оформление таблицы в письме. Из таблицы MailReports в переменную @recipient записываются получатели данной рассылки, в переменную @copy – получатели копии письма, в переменную @letterTopic – тема письма, в переменную @letterHeader – текст приветствие (текст перед всеми таблицами, независимый от присутствия какой-то конкретной из них), в переменную @letterFooter – текст заключение (после всех таблиц).

В одном письме может содержаться несколько табличек, каждая из которых определяется Report-запросом, поэтому открывается цикл (курсор) по всем Report-запросам, связанным с этим письмом. Связь прописана в таблице QueryInMailingList.

Для каждой таблички нужно определить набор статусов инцидентов, которые попадут в письмо и сгенерировать условие с перечислением через логическое ИЛИ всех необходимых статусов. Для этого была написана отдельная функция GetConditionForSelectionByStatus, которая принимает на вход номер рассылки и Report-запроса и возвращает условие вида «IncidentStatusId = 2 OR IncidentStatusId = 4 OR IncidentStatusId = 8…».

Далее предположим, что Report-запрос содержит в себе обращение к нескольким разным участкам реестра инцидентов, которые были заполнены по разным Rule-запросам. В таком случае для формирования одной таблицы в письме необходимо создать столько временных таблиц, на сколько Rule-запросов ссылается один Report-запрос, чтобы затем можно было собрать эти данные воедино с помощью Report-запроса. Связь между Rule-запросом, по которому заполняется реестр инцидентов, и Report-запросом указана в таблице QueryRelations. Для того чтобы получить все необходимые Report-запросу данные, откроем цикл (курсор) по всем связанным с ним Rule-запросам.

По каждому Rule-запросу получим все инциденты из реестра инцидентов, которые ссылаются на рассматриваемый Rule-запрос и имеют статус из полученного в функции GetConditionForSelectionByStatus набора. Эту выборку запишем во временную таблицу

#RegistryResult_N (где N порядковый номер Rule-запроса в цикле).

По завершении цикла все данные для применения Report-запроса готовы. Запрос извлекается из таблицы ReportQueries в строковую переменную. Из таблицы ReportQueryFields извлекаются поля, возвращаемые этим запросом, формируется строка с полями и типами данных. При помощи неё создаётся временная таблица #ReportResult, в которую записывается результат применения Report-запроса к данным в таблицах #RegistryResult_N.

Далее из строки с полями выбираются только те, для которых стоит флаг видимости IsVisible = 1. С использованием этих полей формируется запрос к таблице #ReportResult, который будет возвращать итоговый набор данных одной из таблиц письма.

Для создания оформления в браузере результат запроса должен быть дополнен HTML тэгами. Это реализовано путём вызова процедуры spQueryToHtmlTable, принимающей на вход SQL-запрос и возвращающей строку, содержащую результаты запроса и тэги для визуализации данных в виде таблицы.

Далее полученная строка дополняется текстом, который должен быть расположен до и после конкретной таблицы в письме. Текст извлекается из таблицы ReportQueries (поля BeforeTableText и AfterTableText). В отличие от текста приветствия и заключения всего письма, эти поля зависят от существования конкретной таблицы. Если таблица не была сформирована, например, по причине отсутствия записей, то и подписи к таблице отображены не будут.

Также каждая таблица в письме дополняется информацией о том, когда последний раз были успешно обновлены данные в тех участках реестра, на основе которых построена данная таблица. Эти сведения можно получить из таблицы DataQualityLogs.

Далее каждая таблица в письме проверяется на необходимость формирования построчной рассылки. В таблице QueryInMailingList можно найти информацию, нужна ли построчная отправка выборки в данной рассылке (поля PersonalSendEnabled и PersonalSendPossible). Если значения в обоих столбцах равны логической единице, начинается поиск индивидуальных получателей. Для того чтобы указать их адреса электронной почты в выборке, используется столбец LetterIndividualRecipient.

Например, из реестра для отправки письма была получена информация о PBI-отчётах, авторы которых не заполнили описание (см. таблицу 3). Нужно отправить письмо каждому человеку, кто создавал или последним изменял отчёт. В каждом письме должны быть только те строки, которые актуальны для получателя, то есть не должно быть чужих отчётов.

Таблица 3. Пример для пояснения механизма построчной отправки данных

Наименование отчета	Автор отчёта	Автор последнего изменения	LetterIndividualRecipient
Задачи IT	Иванов А.	Петров Б.	a-ivanov@ocs.ru; b-petrov@ocs.ru
Резервы	Петров Б.	Петров Б.	b-petrov@ocs.ru; b-petrov@ocs.ru
Права доступа	Сидоров С.	Котова М.	s-sidorov@ocs.ru; m-kotova@ocs.ru
Продление лицензий	Иванов А.	NULL	a-ivanov@ocs.ru

Предполагается, что столбец LetterIndividualRecipient всегда будет присутствовать среди полей запроса, выборка данных по которому может отправляться построчно. Для того чтобы не отображать его в письме, рекомендуется всегда устанавливать для него признак IsVisible = 0. В данном столбце может быть указан адрес электронной почты или же несколько адресов через разделитель. Имеется проверка на принадлежность почты получателя домену компании.

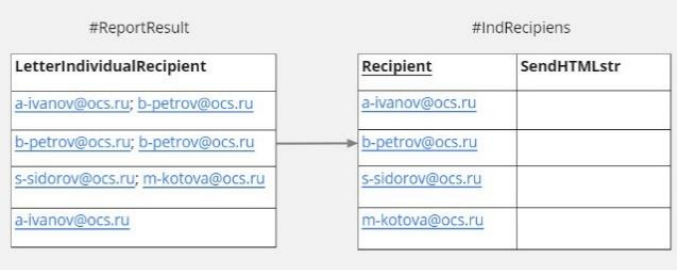


Рисунок 5 – Заполнение таблицы индивидуальных получателей

Каждая строка этого столбца разбивается по разделителю, если это необходимо. Из полученных данных отбираются уникальные значения и записываются в качестве ключа в таблицу #IndRecipiens, которая имеет два столбца – Recipient (почта получателя) и SendHTMLstr (строка с данными для конкретного получателя, дополненная HTML тэгами).

Результат внесения получателей в таблицу #IndRecipiens показан на рисунке 5.

Далее для каждого получателя в таблице #IndRecipiens требуется найти все строки выборки, которые за которые он отвечает. Из таблицы #ReportResult отбираются только те строки, где в столбце LetterIndividualRecipient присутствует рассматриваемый адрес электронной почты. Полученная выборка дополняется HTML-тегами и записывается в столбец SendHTMLstr.

При переходе к следующей таблице в данной рассылке строка с общим результатом @HTMLbodyList (для получателей @recipient) и содержимое столбца SendHTMLstr (для построчной отправки) не затирается, а дополняется. Отправка письма происходит после завершения цикла по таблицам одной рассылки.

Перед отправкой происходит подсчёт строк во всех таблицах письма. Если ни один из Report-запросов по итогу не вернул ни одной строки, то пользователь должен иметь возможность отказаться от получения такого письма. С другой стороны, для некоторых пользователей важно знать, что рассылка работает, даже если ни одного инцидента не обнаружено.

Для того чтобы сделать эту возможность настраиваемой, в таблице MailReports присутствует столбец ReportType. При установке ReportType = 1 пустое письмо указанному пользователю приходить не будет. При установке ReportType = 0 отправка будет происходить всегда, вне зависимости от числа содержащихся в таблицах строк.

Отправка писем происходит с помощью стандартной процедуры msdb.dbo.sp_send_dbmail, на вход которой передаётся список получателей @recipient, список получателей копии письма @copy, тема письма @letterTopic, тело письма – сформированная ранее HTML-строка с данными и тэгами, дополненная также приветствием и заключением (@letterHeader + @HTMLbodyList + @letterFooter), формат – HTML, название почтового агента [10].

После стандартной отправки проверяется необходимость построчной отправки с помощью запроса к таблице #IndRecipiens. Если таблица содержит строки, значит в письме был хотя бы один запрос, имеющий возможность построчной отправки. В таком случае необходимо отправить персональное письмо по каждому адресу из столбца Recipient с набором данных из столбца SendHTMLstr.

Для приведённого выше примера таких писем будет 4 – на каждого уникального получателя:

• Для Иванова А. (рисунок 6)

Наименование отчёта	Автор отчёта	Автор последнего изменения
Задачи IT	Иванов А.	Петров Б.
Продление лицензий	Иванов А.	

Рисунок 6 – Содержание письма для Иванова

• Для Петрова Б. (рисунок 7)

Наименование отчёта	Автор отчёта	Автор последнего изменения
Задачи IT	Иванов А.	Петров Б.
Резервы	Петров Б.	Петров Б.

Рисунок 7 – Содержание письма для Петрова

• Для Сидорова С. и для Котовой М. (рисунок 8)

Наименование отчёта	Автор отчёта	Автор последнего изменения
Права доступа	Сидоров С.	Котова М.

Рисунок 8 – Содержание письма для Сидорова и Котовой

После успешного завершения отправки, в таблицу DataQualityLogs записывается информация о том, что рассылка с номером @mailingId была успешно отправлена с указанием

всех получателей данной рассылки (как заранее заданных, так и индивидуальных). Ошибка, если она происходит, также фиксируется в таблице DataQualityLogs.

Идея построчной отправки данных исходит из идеи построчного доступа к ним. Механизмы построчного доступа существуют во многих СУБД. В основном их применение необходимо в целях обеспечения безопасности [12].

Пример функционирования системы

Предположим, что необходимо отслеживать и отправлять руководителям департаментов информацию о товарных линейках, за которые не назначен ответственный сотрудник или назначен сотрудник, который в компании уже не работает.

Выборка данных при этом будет выглядеть примерно так, как показано в таблице 4.

Видно, что в такой выборке не получится выделить первичный ключ, поэтому при заполнении таблиц, относящихся к обновлению реестра, запрос придётся разбить на два. Один будет возвращать выборку, где ответственного нет – здесь ключом будет товарная линейка (строки синего цвета). Второй – где ответственный уже не работает в компании, здесь ключ будет составной: товарная линейка + ответственный (строки серого цвета).

Далее показан пример заполнения настроечных таблиц для обновления реестра инцидентов и организации рассылки пользователям. На рисунке 10 показан пример работы системы (полученное письмо) по указанным в настроечных таблицах данным.

Таблица 4. Пример выборки данных, которую требуется отправить

Ошибка	Товарная линейка	Ответственный	Департамент	Руководитель департамента
Нет ответственного	AAA	NULL	Dept_1	Маков И. И.
Ответственный уволен	BBB	Иванов	Dept_2	Коньков Н. А.
Ответственный уволен	CCC	Петров	Dept_3	Семенов А. Ю.
Ответственный уволен	CCC	Сидоров	Dept_3	Семенов А. Ю.
Нет ответственного	DDD	NULL	Dept_2	Коньков Н. А.

Подготовка настроечных таблиц для обновления реестра

Перед запуском процедуры обновления реестра требуется внести данные в ряд настроечных таблиц. Пример заполнения отражён в таблицах 5–10.

В целях удобства не все поля таблицы правил RuleTable (таблица 7) приведены в примере заполнения. Часть полей была опущена, так как в них могут находиться NULL значения (как, например, номер задачи) или эти поля всегда заполняются тривиально (как, например, дата создания). Ввиду сохранения конфиденциальности структуры данных компании запросы в таблице RuleQueries (таблица 8) приведены не полностью.

В таблице RuleQueryFields (таблица 9) среди прочих полей присутствует поле «Руководитель департамента», изменения в котором должны сохраняться, но не должны влиять на смену статуса инцидента. Изменения в других полях будут влиять на статус, это задаётся с помощью признака NotChangeStatusIncident.

Таблица 5. ScheduledGroup – таблица групп запуска

ScheduledGroupID	Description
1	В 9 утра каждый день

Таблица 6. RuleTypes – таблица типов правил

RuleTypeID	TypeName
1	Техническая проверка
2	Проверка для бизнеса

Таблица 7. RuleTable – таблица правил

RuleID	RuleTypeID	FormulaForBusiness	IsActiveCheck	ShedulledGroupID
10	2	Контроль ответственных по товарным линейкам	1	1

Таблица 8. RuleQueries – таблица запросов для обновления реестра

RuleQueryID	QueryName	QueryContent	RuleID	IsActiveCheck
14	Проверка ответственных за товарные линейки, ошибка – нет ответственного	SELECT 'Нет ответственного' as [Ошибка], ProductLine as [Товарная линейка], DepartmentName as [Департамент], [Руководитель департамента] FROM ref.ProductLines pl LEFT JOIN … WHERE …	10	1
15	Проверка ответственных за товарные линейки, ошибка – ответственный уволен	SELECT 'Ответственный уволен' as [Ошибка], ProductLine as [Товарная линейка], FullNameRus as [Ответственный], DepartmentName as [Департамент], [Руководитель департамента] FROM ref.ProductLines LEFT JOIN … WHERE …	10	1

Таблица 9. RuleQueryFields – таблица полей Rule-запросов

RuleQueryFieldID	FieldPlace	FieldName	RuleQueryID	NotChange StatusIncident
101	1	Ошибка	14	0
102	2	Тов. линейка	14	0
103	3	Департамент	14	0
104	4	Руководитель департамента	14	1
105	1	Ошибка	15	0
106	2	Тов. линейка	15	0
107	3	Ответственный	15	0
108	4	Департамент	15	0
109	5	Руководитель департамента	15	1

Таблица 10. IncidentKeyBase – таблица ключей

KeyId	RuleQueryId	Field
21	14	Тов. Линейка
22	15	Тов. Линейка
23	15	Ответственный

После заполнения таблиц можно включить вызов процедуры UpdateIncidentRegistry на регулярной основе с помощью планировщика или запущенного на повтор Job. Каждый раз при её вызове полученные по запросам данные будут обрабатываться. На их основе будет происходить обновление реестра. После обновления реестра можно начинать рассылку информации из него.

Подготовка настроечных таблиц для организации рассылки данных из реестра.

Чтобы обеспечить пользователю доступ к данным реестра, необходимо заполнить вторую часть схемы, связанную с организацией рассылки. Пример заполнения отражён в таблицах 11–16.

Таблица MailingLists (таблица 11) содержит сведения о рассылках. Логическая единица в поле SendAfterRegisterUpdated обозначает, что как только данные по обоим запросам, с которыми будет связана эта рассылка, обновятся – письмо будет отправлено. Если поставить FALSE в данном поле, то данные обновляться будут, а вот для отправки придётся создавать отдельную строку в Scheduler или Job.

Поле QueryCountUpdated всегда при создании новой строки должно быть инициализировано нулём. В процессе работы

процедуры обновления реестра число в этом столбце будет увеличиваться. Как только оно станет равно общему числу запросов, от которых зависит рассылка, рассылка будет считаться готовой к отправке. После отправки письма число в этом столбце снова будет сброшено до нуля.

В таблице QueryInMailingList (таблица 14) отражена связь между рассылкой MailingId (ссылка на запись в таблице MailingLists) и запросом ReportQueryId (ссылка на запрос в таблице ReportQueries.) В целях экономии места опущена часть столбцов таблицы. Они используются для более детального отбора отправляемых данных по статусу. В данном случае будут всегда отправляться строки со статусами «Новый», «Открыт», «Изменен», «Переоткрыт» (2 + 4 + 8 + 32 = 46).

Таблица QueryRelations определяет связь Report-запроса с Rule-запросом. RuleQueryId – ссылка на запрос для внесения данных в реестр в таблице RuleQueries, ReportQueryId – ссылка на запрос для отображения данных в письме в таблице ReportQuries. TableOrderName задаёт порядок обращения к данным. То есть сначала (TableOrderName = 1) в запросе обращение идёт к данным, внесённым в реестр по запросу №14, а затем (TableOrderName = 2) – к данным, внесенным в реестр по запросу №15. Это должно быть согласовано с текстом запроса в таблице ReportQueries. TableOrderName соответствует цифре в названии временной таблицы #RegistryResult_N (#RegistryResult_1, #RegistryResult_2…).

Таблица 11. MailingLists – таблица рассылок

MailingId	EmailTopic	EmailHeader	EmailFooter	SendAfter RegisterUpdated	QueryCount Updated
5	Контроль ответственных по товарным линейкам	Добрый день!	Данное письмо сгенерировано автоматически	TRUE	0

Таблица 12. ReportQueries – таблица запросов, обращающихся к реестру инцидентов для отображения выборки в письме

Report QueryID	QueryName	QueryContent	BeforeTableText	AfterTableText
10	Нет ответственного + ответственный уволен	SELECT [Ошибка], [Тов. Линейка], NULL as [Ответственный], [Департамент], [Руководитель департамента] FROM #RegistryResult_1 UNION SELECT [Ошибка], [Тов. Линейка], [Ответственный], [Департамент], [Руководитель департамента] FROM #RegistryResult_2	Для данных товарных линеек неправильно указан ответственный	NULL

Таблица 13. ReportQueriesFields – поля, используемые в Report-запросе

ReportQueriesFieldId	FieldPlace	FieldName	IsVisible	ReportQueryId
51	1	Ошибка	1	10
52	2	Тов. Линейка	1	10
53	3	Ответственный	1	10
54	4	Департамент	1	10
55	5	Руководитель департамента	1	10

Таблица 14. QueryInMailingList – таблица определяет связь между рассылкой и Report-запросом

MailingId	ReportQueryId	PersonalSendEnabled	PersonalSendPossible	SumIncidentStatus
5	10	0	0	46

Таблица 15. QueryRelations – таблица определяет связь Report-запроса с Rule-запросом

RuleQueryId	ReportQueryId	TableOrderName
14	10	1
15	10	2

Таблица 16. MailReports – таблица определяет связь рассылки с её получателями

MailReportId	Recipient	inCopy	ReportType	MailingId
31	a-ivanov@ocs.ru	0	1	5
32	d-petrova@ocs.ru	1	1	5

После заполнения всех указанных таблиц будет возможна рассылка с помощью вызова процедуры MailingIncidentRegistry с указанием в качестве параметра номера рассылки из таблицы MailingLists.

Поскольку в столбце SendAfterRegisterUpdated указана логическая единица, то отправка письма произойдёт автоматически в течение 15 минут после обновления данных в реестре инцидентов.

Пример работы системы можно увидеть на рисунке 10, где зафиксировано письмо, пришедшее в рамках настроенной выше рассылки.

Выводы

Обеспечение качества данных является важным аспектом при построении хранилища данных. Без качественных данных не получится строить аналитику, пригодную для принятия верных корпоративных решений.

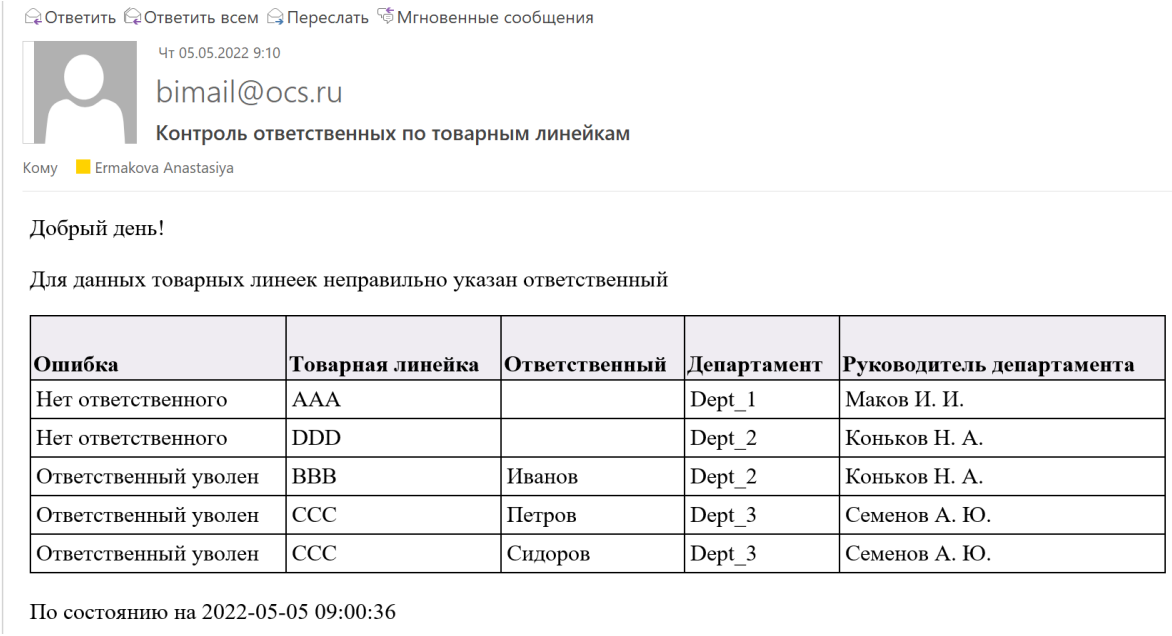
Существует множество готовых программных продуктов для контроля качества данных, но их стоимость велика, а функционал, как правило избыточен.

Для информирования пользователей и разработчиков хранилища данных об изменениях его состояния (возникновении нежелательных событий, инцидентов) был выбран механизм рассылок информации на электронную почту.

В рамках проделанной работы была предложена архитектура системы, которая подразумевает хранение информации о нежелательных событиях и её рассылку указанным адресатам, и разработаны соответствующие хранимые процедуры.

Система показала свою востребованность в компании. На существующие рассылки подписано более сорока адресатов. Поступают запросы на создание новых рассылок.

Эффективность использования системы можно проиллюстрировать диаграммой количества инцидентов для одной из рассылок (рисунок 11). Видно, что получатели рассылки постепенно исправляют найденные в данных ошибки, что в целом положительно влияет на состояние хранилища и возможность строить верную аналитику по нему.



Ответить Ответить всем Переслать Мгновенные сообщения

Чт 05.05.2022 9:10

bimail@ocs.ru

Контроль ответственных по товарным линейкам

Кому Ermakova Anastasiya

Добрый день!

Для данных товарных линеек неправильно указан ответственный

Ошибка	Товарная линейка	Ответственный	Департамент	Руководитель департамента
Нет ответственного	AAA		Dept_1	Маков И. И.
Нет ответственного	DDD		Dept_2	Коньков Н. А.
Ответственный уволен	BBB	Иванов	Dept_2	Коньков Н. А.
Ответственный уволен	CCC	Петров	Dept_3	Семенов А. Ю.
Ответственный уволен	CCC	Сидоров	Dept_3	Семенов А. Ю.

По состоянию на 2022-05-05 09:00:36

Рисунок 10 – Пример письма

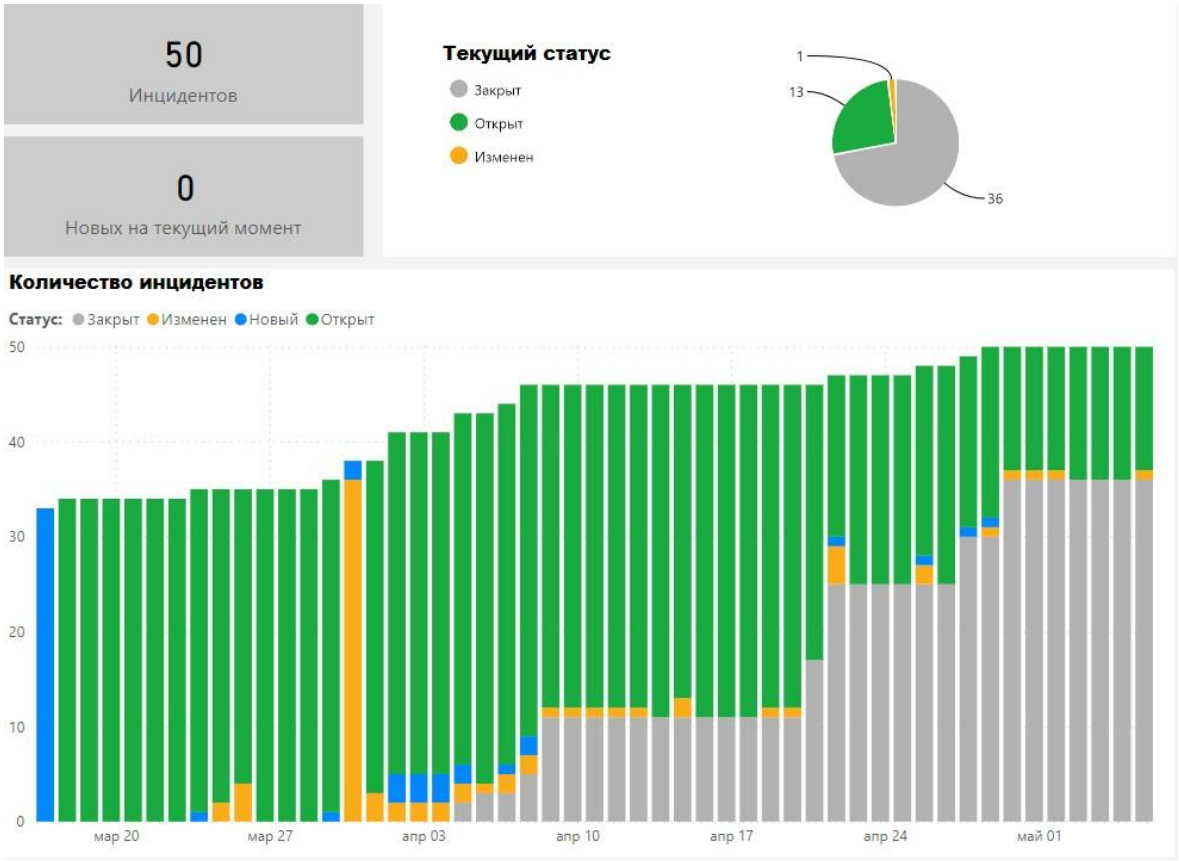


Рисунок 11 – Статистика использования системы

References:

1. Nekrasov, A. A., Gavrilov, S. O., & Belen'kaya, M. N. (2021). Sredstva sozdaniya hranilishch dannyh. *Telekommunikacii i informacionnye tekhnologii*, T. 8, № 1, pp.75-80.
2. (n.d.). Informacionno – analiticheskaya sistema kak instrument provedeniya ekonomicheskogo analiza, Retrieved 01.05.2022 from http://www.e-biblio.ru/book/bib/01_informatika/inform_analit_systemy/posob/332.2.3.html
3. Sagatelyan, S. T., & Lyubimova, V. I. (2020). *E-mail-rassylki: opredelenie, specifika, vidy*. Informacionnye tekhnologii, sistemnyj analiz i upravlenie (ITSAU-2020): Sbornik trudov HVIII Vserossijskoj nauchnoj konferencii molodyh uchenyh, aspirantov i studentov. (pp.260-263). Taganrog: YUzhnyj federal'nyj universitet.
4. (2020). *Svidetel'stvo o gosudarstvennoj registracii programmy dlya EVM № 2020614744 Rossijskaya Federaciya.* Programmnyj modul' email-rassylok: № 2020613690: zayavl. 10.04.2020: opubl. 24.04.2020 / E. A. Ryzhkova, A. S. Zelenskij, A. V. Loskutov, zayavitel' Obshchestvo s ogranichennoj otvetstvennost'yu "Sapl-biz".
5. Troshin, A. A. (2021). Sozdanie procedury dlya rassylki elektronnyh pisem (e-mail) s pomoshch'yu Microsoft SQL server management Studio. *Innovacii. Nauka. Obrazovanie*, № 38, pp. 753-756.
6. Watson, H.J., Goodhue, D.L., & Wixom, B.H. (2002). The benefits of data warehousing: why some organizations realize exceptional payoffs. *Information & Management*, Vol. 39, No. 6, pp. 491–502.
7. (2019). *Proektirovanie sistemy kontrolya kachestva dannyh vnutri hranilishcha pri ogranicheniyah storonnih ETL-instrumentov*. G.A. Sogoyan, M.V. Nesterova, V.A. Izd. dom Vysshej shkoly ekonomiki, (p.422).

8. Woodall, P., Oberhofer, M., & Borek, A. (2014). A classification of data quality assessment and improvement methods. *International Journal of Information Quality*, 3(4), pp. 298-321.
9. (2021). *Gartner Magic Quadrant for Data Quality Solutions*. Retrieved 01.05.2022 from https://www.informatica.com/data-quality-magic-quadrant.html
10. (n.d.). *Stored procedure sp_send_dbmail (Transact-SQL)*, Retrieved 01.05.2022 from: https://docs.microsoft.com/en-gb/sql/relational-databases/system-stored-procedures/sp-send-dbmail-transact-sql?view=sql-server-ver15
11. Sabinin, O. YU., & Shejkina, E. S. (2017). Avtomatizaciya postroeniya otchetov v sfere sertifikacii sistem menedzhmenta s pomoshch'yu Oracle business intelligence. *Theoretical & Applied Science*, № 3(47), pp. 121-127.
12. Ermakova, A. E., & Sabinin, O. Yu. (2020). Organization of row level security in Postgresql on the example of bank system protection. *Theoretical & Applied Science*, No 5(85), pp. 462-469.