(19) Евразийское патентное ведомство

(11) 034680 (13) B1

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ЕВРАЗИЙСКОМУ ПАТЕНТУ

(45) Дата публикации и выдачи патента
2020.03.05

(21) Номер заявки
201700604

(22) Дата подачи заявки
2017.12.27

(51) Int. Cl. ***G06F 8/30*** (2018.01)
G06F 17/30 (2006.01)

(54) СПОСОБ И СИСТЕМА АВТОМАТИЧЕСКОЙ ГЕНЕРАЦИИ ПРОГРАММНОГО КОДА ДЛЯ КОРПОРАТИВНОГО ХРАНИЛИЩА ДАННЫХ

(43) **2019.06.28**

(71)(73) Заявитель и патентовладелец:
ПУБЛИЧНОЕ АКЦИОНЕРНОЕ ОБЩЕСТВО "СБЕРБАНК РОССИИ" (ПАО СБЕРБАНК) (RU)

(72) Изобретатель:
Горский Владимир Борисович, Веселов Максим Витальевич, Бутин Игорь Владимирович, Чиликин Павел Сергеевич, Целовальников Станислав Петрович, Рабинович Борис Ильич (RU)

(74) Представитель:
Герасин Б.В., Астафьева С.А. (RU)

(56) US-A1-20150100542
US-B2-8504513
US-B2-7840607
US-B2-9519695
US-A1-20170220654
US-B1-6212524

(57) Техническое решение, в общем, относится к области вычислительной техники, а в частности, к системам и способам автоматической генерации программного кода для корпоративного хранилища данных. Способ автоматической генерации программного кода для корпоративного хранилища данных, в котором получают метаданные, описывающие настройку механизмов трансформации данных для их загрузки на уровень детального слоя и расчета витрин хранилища данных; формируют по меньшей мере один шаблон обновления данных детального слоя и витрины данных хранилища данных; осуществляют генерацию программного кода для загрузки данных в детальный слой хранилища и расчета витрин данных на основе полученных метаданных и сформированного шаблона обновления данных; устанавливают сгенерированный на предыдущем шаге программный код на среду хранилища данных для выполнения загрузок; осуществляют переиспользование метаданных обновления детального слоя и витрин данных. Технический результат - повышение стабильности работы алгоритмов детального слоя и витрин данных, а также уменьшение количества инцидентов в хранилище данных.

034680 B1

034680 B1

Область техники

Техническое решение, в общем, относится к области вычислительной техники, а в частности, к системам и способам автоматической генерации программного кода для корпоративного хранилища данных.

Уровень техники

Аналитические хранилища данных в настоящее время становятся необходимым атрибутом для корпораций, работающих с большими объемами данных, и служат для выявления и создания новых финансовых инструментов, сопровождения имеющихся, контроля и выявлений проблем учета, создания новых и коррекции работающих финансовых стратегий.

Источником данных для таких систем практически всегда являются разнородные системы, обслуживающие как фронт-офисные системы, представляющие лицо корпорации перед клиентами, так и бэк-офисные приложения, осуществляющие финансовый учет. Кроме того, очень часто хранилища также хранят и мидл-офисные данные для анализа своей работы. Таким образом, источником данных для корпоративного хранилища являются сотни систем, постоянно меняющиеся в соответствии с требованиями развития бизнеса.

В таких условиях создание четкой однотипной системы для сбора данных в единой модели является необходимым требованием для получений актуальных и достоверных данных, поступающих из исходных систем.

Из уровня техники известен патент № US 6604110B1 "Automated software code generation from a metadata-based repository", патентообладатель: International Business Machines Corp., дата публикации: 31.10.2000. В данном техническом решении раскрывается способ предоставления исполняемого программного кода для использования в приложении управления модели данных предприятия (EDM), который передает данные из одного или нескольких источников данных в базу данных приложений EDM, для использования в них в ответ на пользовательские команды.

Изобретение работает только в интерактивном режиме в ответ на команды пользователя, что требует определенных трудозатрат и влияет на скорость работы. Более того, данное изобретение предоставляет только исполняемый код, что значительно сужает возможности технического решения и не предоставляет сценарии или метаданные, на основании которых в режиме исполнения эти метаданные будут определять сценарий выполнения.

Сущность изобретения

Данное техническое решение направлено на устранение недостатков, присущих существующим решениям, известным из уровня техники.

Технической проблемой (или технической задачей) в данном техническом решении является осуществление автоматической генерации программного кода для корпоративного хранилища данных.

Техническим результатом, проявляющимся при решении вышеуказанной технической проблемы, является повышение стабильности работы алгоритмов детального слоя и витрин данных, а также уменьшение количества инцидентов в хранилище данных.

Дополнительным техническим результатом является повышение эффективности работы хранилища данных и сокращение времени на выполнение типовых операций для аналитиков за счет использования метаданных и шаблонов обновления данных.

Дополнительно повышается скорость мигрирования данных на другую платформу за счет переиспользования метаданных и шаблонов данных.

Указанный технический результат достигается благодаря осуществлению способа автоматической генерации программного кода для корпоративного хранилища данных, в котором получают метаданные, описывающие настройку механизмов трансформации данных для их загрузки на уровень детального слоя и расчета витрин хранилища данных; формируют по меньшей мере один шаблон обновления данных детального слоя и витрины данных хранилища данных; осуществляют генерацию программного кода для загрузки данных в детальный слой хранилища и расчета витрин данных на основе полученных метаданных и сформированного шаблона обновления данных; устанавливают сгенерированный на предыдущем шаге программный код на среду хранилища данных для выполнения загрузок; осуществляют переиспользование метаданных обновления детального слоя и витрин данных.

В некоторых вариантах осуществления метаданные хранятся в таблицах реляционной базы данных.

В некоторых вариантах осуществления метаданными являются описание структур таблиц, взаимосвязей между ними, правил секционирования, описание витрин данных.

В некоторых вариантах осуществления метаданные являются константными и расчетными.

В некоторых вариантах осуществления метаданные размещаются в форме поколоночной структуры объектов и их первичных ключей в базе данных.

В некоторых вариантах осуществления шаблон обновления данных описан в S2T-файле.

В некоторых вариантах осуществления дополнительно при осуществлении генерации программного кода получают лог работы, который включает все входные макропеременные, а также трассировочный файл.

В некоторых вариантах осуществления осуществляют пере использование метаданных обновления детального слоя и витрин данных за счет внесения дельты изменений в S2T файл.

В некоторых вариантах осуществления переиспользование метаданных обновления детального слоя и витрин данных осуществляют путем переключения базовых шаблонов обновления.

Краткое описание чертежей

Признаки и преимущества изобретения станут очевидными из приводимого ниже подробного описания изобретения и прилагаемых чертежей, на которых

на фиг. 1 показан пример осуществления способа автоматической генерации программного кода для корпоративного хранилища данных;

на фиг. 2 - архитектура области хранения данных базы данных корпоративного хранилища;

на фиг. 3 схематично показана логическая структура и ETL-процессы построения единого семантического слоя;

на фиг. 4 показана схема вычисления суррогатных ключей;

на фиг. 5 - полный путь автоматической генерации программного кода.

Подробное описание изобретения

Техническое решение может быть реализовано на компьютере, в виде автоматизированной системы (АС) или машиночитаемого носителя, содержащего инструкции для выполнения вышеупомянутого способа.

Техническое решение может быть реализовано в виде распределенной компьютерной системы.

В данном решении под системой подразумевается компьютерная система, ЭВМ (электронно-вычислительная машина), ЧПУ (числовое программное управление), ПЛК (программируемый логический контроллер), компьютеризированные системы управления и любые другие устройства, способные выполнять заданную, чётко определённую последовательность вычислительных операций (действий, инструкций).

Под устройством обработки команд подразумевается электронный блок либо интегральная схема (микропроцессор), исполняющая машинные инструкции (программы).

Устройство обработки команд считывает и выполняет машинные инструкции (программы) с одного или более устройства хранения данных. В роли устройства хранения данных могут выступать, но, не ограничиваясь, жесткие диски (HDD), флеш-память, ПЗУ (постоянное запоминающее устройство), твердотельные накопители (SSD), оптические приводы.

Программа - последовательность инструкций, предназначенных для исполнения устройством управления вычислительной машины или устройством обработки команд.

Ниже будут описаны термины и понятия, необходимые для осуществления настоящего технического решения.

ETL (от англ. Extract, Transform, Load - "извлечение, преобразование, загрузка") - один из основных процессов в управлении хранилищами данных, который включает в себя извлечение данных из внешних источников; их трансформация и очистка, чтобы они соответствовали потребностям бизнес-модели; и загрузка их в хранилище данных.

S2T - файл формата Excel, содержащий информацию обо всех преобразованиях данных предметной области, которая подлежит ETL разработке.

Суррогатные ключи - набор метаданных, определяющих правила создания первичных ключей хранилища данных, а также правила преобразования от естественных ключей исходных данных к первичным ключам хранилища.

Естественный ключ - набор атрибутов описываемой записью сущности, уникально её идентифицирующий (например, номер паспорта для человека).

Детальный слой - набор таблиц определенной физической области хранилища данных, который является финальным слоем для ETL процессов и которые содержат все бизнес данные хранилища предметной области.

Метаданные - служебная информация решения, описывающая настройки механизмов и результаты работы подсистем интеграции данных, хранилища данных и аналитического приложения (например, информация о загруженных файлах).

Фреймворк ETL - программное обеспечение в серверном исполнении, управляющее и загружающее данные из внешних систем в хранилище данных.

Семантический слой - набор физических и виртуальных витрин данных, предоставляющих данные модели FSLDM в преобразованном виде в соответствии с назначением конкретной витрины.

Витрина данных (англ. Data Mart; другие варианты перевода: хранилище данных специализированное, киоск данных, рынок данных) - срез хранилища данных, представляющий собой массив тематической, узконаправленной информации, ориентированный, например, на пользователей одной рабочей группы или департамента.

OLAP - технология обработки данных, заключающаяся в подготовке суммарной (агрегированной) информации на основе больших массивов данных, структурированных по многомерному принципу. Реализации технологии OLAP являются компонентами программных решений класса Business Intelligence.

Патч (англ. patch - заплатка) - информация, предназначенная для автоматизированного внесения определённых изменений в компьютерные файлы и базы данных.

Способ автоматической генерации программного кода для корпоративного хранилища данных, показанный на фиг. 1, осуществляют следующим образом.

Шаг 101: получают метаданные, описывающие настройку механизмов трансформации данных для их загрузки на уровень детального слоя и расчета витрин хранилища данных.

В корпоративном хранилище данных 200, показанном на фиг. 2, зафиксирована четкая система этапов загрузки и преобразования данных из внешних автоматизированных систем, сопровождающаяся единым протоколированием выполнения всех этапов, динамическим и статическим распараллеливанием и возможностью восстановления в случае сбоя с любого места загрузки.

Метаданные являются важной частью архитектуры хранилища данных 200. Метаданные - это данные, описывающие правила, по которым функционирует хранилище. Например, с точки зрения базы данных хранилища метаданными является описание структур таблиц, взаимосвязей между ними, правил секционирования, описание витрин данных и т.п. С точки зрения ETL метаданными являются описания правил извлечения и преобразования данных, периодичность выполнения ETL-процессов и т.п.

В некоторых вариантах осуществления метаданные корпоративного хранилища 200 включают

информацию о данных, находящихся в хранилище, их бизнес-описание и структуру хранения;

описание структур источников данных как внешних, так и внутренних, их доступности;

информацию о структуре процессов ETL, периодичности их выполнения, применяемых правил очистки и преобразования данных;

описание представления данных, помогающее пользователю работать с B1-приложением;

информацию о настройках безопасности, правил аутентификации и назначенных прав доступа;

статистику утилизации ресурсов, обращений к данным и др., которая помогает администратору оптимизировать работу базы данных хранилища.

В некоторых вариантах осуществления управление метаданными осуществляется отдельными инструментами для каждого из компонентов хранилища. Например, для базы данных Oracle, метаданные которой хранятся в системных таблицах и настроечных файлах, данным инструментом является Oracle Enterprise Manager.

В некоторых вариантах осуществления метаданные могут храниться в специальных таблицах PostgreSQL.

В качестве метаданных могут использоваться атрибуты, которые являются параметрами, позволяющими управлять ETL-процессом. В некоторых вариантах осуществления метаданные, и соответственно атрибуты могут быть константными и расчетными. Значения константных атрибутов не зависят от результатов предыдущих загрузок данных в хранилище. Расчетные атрибуты являются переменными значениями, зависящими от результатов предыдущих ETL-загрузок. Значения атрибутов могут заноситься в структуры ETL через веб-интерфейс. Такими атрибутами могут быть режим загрузки (например, архивный, инкрементальный), версия спецификации загрузки, порядковый номер инкрементальной загрузки, нижняя граница выгрузки для каждой загружаемой сущности и т.п.

В качестве метаданных могут использоваться статистики, которые являются сохраняемыми во время ETL-процесса значениями, которые можно использовать для вычисления расчетных атрибутов в последующих процессах. Значения статистик могут сохраняться в таблицах средствами хранимых процедур реляционной базы данных. В качестве реляционной СУБД может использоваться такая как MySQL, PostgreSQL или Oracle, не ограничиваясь. В некоторых вариантах осуществления изобретения метаданные размещаются в форме поколоночной структуры объектов и их первичных ключей в базе данных. Эти данные могут быть представлены либо в виде табличного представления и/или в виде электронных таблиц Excel, и/или в любой реляционной БД.

Метаданные получают из первичных внешних систем 230 данных и загружают в область 210 временного хранения данных (фиг. 2).

Область 210 временного хранения данных является промежуточным слоем между внешними источниками данных и областью постоянного хранения, в которой находится детальный слой и витрины данных. В данной области сохраняются извлеченные из внешних источников данных (СУБД, csv, dbf, xml файлов, web-сервисов и т.д.) данные, производится их очистка, трансформация, обогащение, подготовка к загрузке в область 220 постоянного хранения. Зачастую очередной цикл обработки и загрузки данных в хранилище не может быть начат пока не будут извлечены все необходимые данные из различных внешних источников данных, а в силу ряда причин (географической распределенности, разных циклов функционирования систем и т.п.) данные в источниках могут быть доступны в разные моменты по времени. Область 210 временного хранения служит для сбора всех необходимых данных перед началом трансформации.

Шаг 102: формируют по меньшей мере один шаблон обновления данных детального слоя и витрины данных хранилища данных.

Детальный слой является основной корпоративного хранилища данных. В этой области хранятся преобразованные и очищенные детальные данные, полученные из внешних систем 230 и источников данных, и основные классификаторы. Данная область содержит следующие типы сущностей: справочники и классификаторы; сущности, содержащие фактические значения; сущности, описывающие связи.

Справочники и классификаторы могут определять следующую информацию

участников основных бизнес-процессов - клиентов, поставщиков, филиалы, услуги, продукты и т.п.;

базовые справочники - дата и время, валюта, страны и т.п.;

прочие справочники - отражающие потребности бизнеса в необходимой аналитике данных, определяющие в разрезе каких справочников необходимо анализировать фактические данные.

Сущности, содержащие фактические значения, - транзакционные данные из внешних систем 230 и источников данных. Например, информация о совершенных платежах клиентов, выставленных счетах, проводках и т.п.

Сущности, содержащие связи, определяют взаимосвязи между всеми остальными сущностями. Например, связь может быть следующей: клиент-услуга-банк.

Витрины данных являются объектами хранения аналитической информации, нацеленными на поддержку конкретных бизнес-функций, конкретных подразделений компании. На уровне базы данных витрины обычно реализуются по схеме "звезда" или "снежинка" и содержат данные из области детального слоя. Также витрины данных могут быть реализованы в виде многомерного OLAP-куба. Витрины данных являются основой, обеспечивающей возможность проведения многомерного анализа (OLAP) данных. В корпоративном хранилище данных 200 детальный слой и витрины данных располагаются в области 220 постоянного хранения данных (фиг. 2).

Осуществление преобразований метаданных и обрамление их в хранимые процедуры, реализуется с помощью шаблонов обновления данных, работающих по типу Java run-time подстановок, аналогично Apache Jakarta Project, где такая техника использовалась как основа шаблона Model-View Controller для web-приложений. Правила преобразования могут храниться в поколоночном представлении. Эти данные могут быть представлены в виде табличного представления и/или в виде электронных таблиц Excel, и/или в любой реляционной БД.

В некоторых вариантах осуществления шаблоном обновления данных является шаблон SQL-процедуры.

В техническом решении используется стратегия обновления данных детального слоя и витрин данных, которая представляет собой способ записи исходного набора данных в формате таблицы детального слоя в таблицу детального слоя, в результате чего решается задача выстраивания истории. В отличие от способов форматирования исходных данных в формат детального слоя (слой B $\rightarrow$ слой N), шаблон обновления данных (слой N $\rightarrow$ слой T) полностью стандартен и может быть описан в S2T-файле посредством указания названия стратегии обновления и ее параметров в виде шаблона обновления данных.

В данном техническом решении могут использоваться, например, следующие шаблоны обновления данных "снимок данных - снимок данных", "снимок данных - история данных", "история данных - история данных". Данные стратегии реализуются в зависимости от вида исходных данных и требований к конечным данным.

Стратегия обновления данных "снимок данных - снимок данных" работает следующим образом. Неисторичные данные из источника данных перекладываются в неисторичные таблицы в детальный слой хранилища данных. Данный шаблон обновления данных может применяться для контейнеров суррогатных ключей (договора, клиенты и пр.), а также для транзакционных данных (финансовых операций, проводок и пр.). Если поля источника, использующиеся для расчета первичного ключа детального слоя, могут перезаписываться на источнике, то поддерживают эмуляцию режима репликации. Репликация - это процесс, под которым понимается копирование данных из одного источника на другой (или на множество других) и наоборот. Причем данный шаблон обновления данных может работать с накоплением данных и без накопления данных.

Стратегия обновления данных "снимок данных - история данных" работает следующим образом. Данные, для которых на источнике не ведется история изменения, сохраняются в детальный слой постоянного хранилища данных с созданием истории. При этом из детального слоя удаляется история за более поздние периоды, чем тот, что пришел в текущем пакете обновления. Если поля источника, использующиеся для расчета первичного ключа детального слоя, могут перезаписываться на источнике, то поддерживают эмуляцию режима репликации.

Стратегия обновления данных "история данных - история данных" работает следующим образом. Выполняется полная репликация истории данных, ведущейся на источнике с привязкой к дате/времени изменения значения атрибута (или с привязкой к периодам дат - датам/времени начала и окончания действия), в детальный слой хранилища данных. При этом для исключения проблемы с обновлением поля "дата актуальности" на источнике поддерживают эмуляцию режима репликации. При загрузке исходных наборов данных с историей на дату, разрывы в истории детального слоя не допускаются. Пересечений в истории детального слоя не допускается; усечению подлежит более ранний период из двух пересекающихся. История за определенный период может быть также загружена шаблоном "снимок - снимок", если нет необходимости в устранении пересечения периодов истории. Для этого необходимо обеспечивать эмуляцию режима репликации, причем в состав ключа должны входить как поля, на основе которых рассчитывается первичный ключ в детальном слое, так и поля, содержащие начало и окончание периода

актуальности.

Формирование суррогатного ключа может осуществляться путем ведения и мапирования исходных наборов натуральныхх ключей на последовательность целочисленных значений или, как показано на фиг. 4, может определяться на основании значения уникального ключа сущности по следующей формуле.

Для сущностей с уникальным ключом, состоящим из одного атрибута

SURROGATE KEY = SHA1(NNN || SOURCE KEY1)

Для сущностей с составным уникальным ключом

SURROGATE KEY = SHA1(NNN || SHA1(SOURCE KEY1)[[||

SHA1(SOURCE_KEY2)] || … || SHA1(SOURCE_KEYN)])

где NNN - 3-символьный код исходной системы;

SHA1 - функция получения значения хеша по алгоритму SHA1;

SOURCE KEY1 .. SOURCE KEYN - колонки, входящие в уникальный ключ таблицы из исходной системы, если уникальный ключ представлен более чем одним полем, то используется конкатенированное значение всех полей входящих в ключ с разделителем "|";

|| - операция конкатенации.

Суррогатные ключи, полученные по данной схеме, могут иметь вид 160-битного 16-ричного кода. В дальнейшем такие ключи удобно использовать для связывания не унифицированных сущностей.

В некоторых вариантах осуществления используют представление Apache Hive для расчета значений суррогатных ключей. Hive представляет из себя движок, который превращает SQL-запросы в цепочки map-reduce задач. Движок включает в себя такие компоненты, как Parser (разбирает входящие SQL-запросы), Optimizer (оптимизирует запрос для достижения большей эффективности), Planner (планирует задачи на выполнение), Executor (запускает задачи на фреймворке MapReduce).

Для работы Hive также необходимо хранилище метаданных. SQL предполагает работу с такими объектами, как база данных, таблица, колонки, строчки, ячейки и т.д. Поскольку сами данные, которые использует Hive, хранятся просто в виде файлов на hdfs, необходимо где-то хранить соответствие между объектами Hive и реальными файлами.

Шаг 103: осуществляют генерацию программного кода для загрузки данных в детальный слой хранилища и расчета витрин данных на основе полученных метаданных и сформированного шаблона обновления данных.

На фиг. 5 показан процесс автоматический генерации программного кода, который состоит из следующих этапов.

1) Получение SQL-кода на основе шаблона обновления данных, представляющих собой по меньшей мере одну хранимую процедуру, из S2T-файла;

S2T файл может выглядеть следующим образом:

№	Таблица-приемник	Поле-приемник	Правило трансформации	Трансформация табличного уровня	Стратегия обновления данных	Роль в истории
1	t_snapshot	cust_id	SK (customer_ssid)	SELECT …FROM s_block	снимок – снимок	Ключ
2		id	<sequence>			–
4		attr_a_val	customer_value1			–
5		attr_b_val	customer_value2			–

2) инкапсуляция этого программного кода в исполнимый модуль (процедуру), со стандартными входными и выходными параметрами, системой логирования и обработки ошибок и способами восстановления процесса загрузки в случае технологического падения;

3) вставка вызова этой процедуры в сценарий метаданных, который читается фреймворком в runtime режиме и последовательно и/или параллельно исполняет инструкции из шаблона.

Внутри шаблонов процедур (фиг. 5) реализуются различные "case" или "if else" ветви реализации, зависящие от флагов запуска. В свою очередь, флаги запуска задаются общими флагами проекта, флагами этапа преобразования, конкретными флагами преобразования и т.п.

Результатом обработки каждого шаблона является непосредственно сгенерированный рабочий код программы. Дополнительно в результате работы шаблона получают лог работы, который включает все входные макропеременные, а также трассировочный файл (trace-файл), в котором могут содержаться все значения переменных, используемых структур данных и массивов (коллекций).

Каждое преобразование может содержать или имя, или условное обозначение используемых шаблонов.

Входными данными для шаблона процедуры могут служить скалярные переменные, векторные

элементы (массивы, списки, хэш-таблицы), а также многомерные элементы (вектора векторов). В качестве элементов могут выступать как простейшие элементы, так комплексные объекты (объекты на основе разработанных Java-классов).

Шаг 104: автоматически устанавливают сгенерированный на предыдущем шаге программный код на среду хранилища данных для выполнения загрузок.

На данном шаге создается патч, который содержит сгенерированный программный код, в формате и структуре соответствующей стандартам релизно-конфигурационного управления, который в режиме run-time устанавливается на среду разработку. В патч также может быть включен сценарий установки патча с инструкциями для консоли администратора. Патч, готовый для установки на среду разработки корпоративного хранилища данных, является совместимым с системой автоматического распространения патчей по стендам.

Объекты, базы данных, которые входят в патч, в том же порядке устанавливаются на среду разработки. Если возникают ошибки, они присутствуют в журнале событий и видны пользователю. Набор журналов событий содержит подробный разбор возможных ошибок и предупреждений. Однотипные объекты сгруппированы в директории по типам данных и находятся внутри суффиксов имен баз данных, их последующего размещения.

Ниже в таблице показана типовая структура патча. Особо следует обратить внимание на Setup-Manual_000000.txt, в котором автоматически создается последовательность шагов для установки патча (дистрибутива) на все среды

gcode/td/		cross.txt	включаются в дистрибутив
		SetupManual_000000.txt	
		td_xxxxxx.txt	
db_stg_ddl	procedure	p.db_stg_ddl.sp108abcdef_bb_00010_0001_cus.sql	процедуры обработки данных сразу после В таблиц. Обновляют данные в В (DELETED_FLAG)
		p.db_stg_ddl.sp108abcdef_bb_00740_0002_sta.sql	
		p.db_stg_ddl.sp108abcdef_bn_00010_cust.sql	
		p.db_stg_ddl.sp108abcdef_bn_00090_cust_ind.sql	
		p.db_stg_ddl.sp108abcdef_ti_00010_cust.sql	
		p.db_stg_ddl.sp108abcdef_ti_00090_cust_ind.sql	
	table	t.db_stg_ddl.b108abcdef0001_cust.sql	
		t.db_stg_ddl.b108abcdef0002_status.sql	
		t.db_stg.s1080020_agrtremp.sql	
		t.db_stg_ddl.n108abcdef_appl_stts.sql	
		t.db_stg_ddl.n108abcdef_agr_stts_type.sql	
		t.db_stg_ddl.ti108abcdef_01370_#am_appl_su.sql	
		t.db_stg_ddl.td108abcdef_01370_#am_appl_su.sql	
		t.db_stg_ddl.ti108abcdef_01040_agr_stts_ty.sql	
		t.db_stg_ddl.td108abcdef_01040_agr_stts_ty.sql	
	view	v.db_stg_ddl.C108ABCDEF0001_cust.sql	
		v.db_stg_ddl.C108ABCDEF0002_status.sql	
db_stg_ddl	procedure		

	table	t.db_stg_ddl.s1080001_cust.sql	таблица накапливаемого стейджинга
	view	v.db_stg_ddl.sv1080001_cust.sql	
db_stg_tmd	procedure		
	table		
	view		
scripts		s.db_tmd.etl_bk9s2t.sql	для этапа T
		s.db_tmd.etl_bk9sql.sql	для этапа T
		s.db_tmd.etl_bk9tab.sql	для этапа T
		s.db_tmd.etl_bkcols.sql	для этапа B
		s.db_tmd.etl_bknscn.sql	для этапа H
		s.db_tmd.etl_bktype.sql	для этапа B
		s.db_tmd.etl_task_param.sql	

В режиме "Compile"="true" все объекты патча создаются на среде разработки. Т.е. в режиме выполнения кодогенерации все объекты, которые войдут в патч, создаются на среде разработки.

Шаг 105: осуществляют переиспользование метаданных обновления детального слоя и витрин данных.

При внесении изменений в постановку задачи осуществляют пере использование метаданных обновления детального слоя и витрин данных за счет внесения дельты изменений в вышеуказанную S2T форму.

При смене типа реляционной базы данных осуществляют переиспользование метаданных обновления детального слоя и витрин данных для формирования сценария наполнения путем переключения базовых шаблонов обновления.

Затем осуществляют переиспользование сгенерированного программного кода и метаданных. После реализации первой версии патча (дистрибутива) по какой-то задаче практически всегда следуют изменения к этой задаче, вызванные как новыми требованиями бизнеса, так и изменениями, вызванными исправлениями первой версии. Результатом этих изменений могут быть создание новых объектов, удаление старых объектов, изменение процедур, изменение сценариев.

Для реализации этих изменений могут использовать два подхода.

Первым подходом является внесение изменений в имеющийся файл (структуру) S2T и пересоздание всего патча, т.е. реализация нового патча (кумулятивного патча). После этого изменения могут быть выделены системой управления версиями (git), используемой для ведения кода.

Вторым вариантом является выполнение автоматических пометок в закладках S2T измененных преобразований и тогда только они войдут в патч. В интерфейс патч должет быть помечен как инкрементальный. Возникают изменения, вызванные потребностями работы процесса. Данные изменения вносятся в имеющиеся метаданные, описанные в S2T и при генерации осуществляется переиспользование ранее введенных метаданных. Аспекты настоящего изобретения могут быть также реализованы с помощью устройства обработки данных, являющимся вычислительной машиной или системой (или таких средств, как центральный/графический процессор или микропроцессор), которая считывает и исполняет программу, записанную на запоминающее устройство, чтобы выполнять функции вышеописанного варианта(ов) осуществления, и способа, показанного на фиг. 1, этапы которого выполняются вычислительной машиной или устройством путем, например, считывания и исполнения программы, записанной на запоминающем устройстве, чтобы исполнять функции вышеописанного варианта(ов) осуществления. С этой целью программа записывается на вычислительную машину, например, через сеть или со среды для записи различных типов, служащей в качестве запоминающего устройства (например, машиночитаемой среды). Устройство обработки данных может иметь дополнительные особенности или функциональные возможности. Например, устройство обработки данных может также включать в себя дополнительные устройства хранения данных (съемные и несъемные), такие как, например, магнитные диски, оптические диски или лента. Устройства хранения данных могут включать в себя энергозависимые и энергонезависимые, съемные и несъемные носители, реализованные любым способом или при помощи любой технологии для хранения информации, такой как машиночитаемые инструкции, структуры данных, программные модули или другие данные. Устройство хранения данных, съемное хранилище и несъемное хранилище являются примерами компьютерных носителей данных. Компьютерные носители данных включают в себя, но не в ограничительном смысле, оперативное запоминающее устройство (ОЗУ), постоянное запоминающее устройство (ПЗУ), электрически стираемое программируемое ПЗУ (EEPROM), флэш-память или память, выполненную по другой технологии, ПЗУ на компакт-диске (CD-ROM), универсальные цифровые диски (DVD) или другие оптические запоминающие устройства, магнитные кассеты, магнитные ленты, хранилища на магнитных дисках или другие магнитные запоминающие устройства, или

любую другую среду, которая может быть использована для хранения желаемой информации и к которой может получить доступ устройство обработки данных. Устройство обработки данных может также включать в себя устройство(а) ввода, такие как клавиатура, мышь, перо, устройство с речевым вводом, устройство сенсорного ввода и так далее. Устройство (а) вывода, такие как дисплей, динамики, принтер и тому подобное, также могут быть включены в состав системы.

Устройство обработки данных содержит коммуникационные соединения, которые позволяют устройству связываться с другими вычислительными устройствами, например по сети. Сети включают в себя локальные сети и глобальные сети наряду с другими большими масштабируемыми сетями, включая, но не в ограничительном смысле, корпоративные сети и экстрасети. Коммуникационное соединение является примером коммуникационной среды. Как правило, коммуникационная среда может быть реализована при помощи машиночитаемых инструкций, структур данных, программных модулей или других данных в модулированном информационном сигнале, таком как несущая волна, или в другом транспортном механизме, и включает в себя любую среду доставки информации. Термин "модулированный информационный сигнал" означает сигнал, одна или более из его характеристик изменены или установлены таким образом, чтобы закодировать информацию в этом сигнале. Для примера, но без ограничения, коммуникационные среды включают в себя проводные среды, такие как проводная сеть или прямое проводное соединение, и беспроводные среды, такие как акустические, радиочастотные, инфракрасные и другие беспроводные среды. Термин "машиночитаемый носитель", как употребляется в этом документе, включает в себя как носители данных, так и коммуникационные среды. Последовательности процессов, описанных в этом документе, могут выполняться с использованием аппаратных средств, программных средств или их комбинации. Когда процессы выполняются с помощью программных средств, программа, в которой записана последовательность процессов, может быть установлена и может выполняться в памяти компьютера, встроенного в специализированное аппаратное средство, или программа может быть установлена и может выполняться на компьютер общего назначения, который может выполнять различные процессы.

Например, программа может быть заранее записана на носитель записи, такой как жесткий диск, или ПЗУ (постоянное запоминающее устройство). В качестве альтернативы, программа может быть временно или постоянно сохранена (записана) на съемном носителе записи, таком как гибкий диск, CD-ROM (компакт-диск, предназначенный только для воспроизведения), MO (магнитооптический) диск, DVD (цифровой универсальный диск), магнитный диск или полупроводниковая память. Съемный носитель записи может распространяться в виде так называемого, продаваемого через розничную сеть программного средства.

Программа может быть установлена со съемного носителя записи, описанного выше, на компьютер, или может быть передана по кабелю с сайта загрузки в компьютер или может быть передана в компьютер по сетевым каналам передачи данных, таким как ЛВС (локальная вычислительная сеть) или Интернет. Компьютер может принимать переданную таким образом программу и может устанавливать ее на носитель записи, такой как встроенный жесткий диск. Процессы, описанные в этом документе, могут выполняться последовательно по времени в соответствии с описанием или могут выполняться параллельно или отдельно в зависимости от характеристик обработки устройства, выполняющего процессы, или в соответствии с необходимостью. Система, описанная в этом документе, представляет собой логический набор множества устройств и не ограничивается структурой, в которой эти устройства установлены в одном корпусе.

ФОРМУЛА ИЗОБРЕТЕНИЯ

1. Способ автоматической генерации программного кода для корпоративного хранилища данных, реализуемый с помощью процессора, включающий следующие шаги:

получают метаданные, описывающие настройку механизмов трансформации данных для их загрузки на уровень детального слоя хранилища данных и расчета витрин хранилища данных;

формируют по меньшей мере один шаблон обновления данных детального слоя хранилища данных и витрины данных хранилища данных;

осуществляют генерацию программного кода для загрузки данных в детальный слой хранилища данных и расчета витрин данных на основе полученных метаданных и сформированного шаблона обновления данных;

устанавливают сгенерированный на предыдущем шаге программный код на среду хранилища данных для выполнения загрузок;

осуществляют переиспользование метаданных для обновления детального слоя хранилища данных и витрин данных посредством внесения дельты изменений в вышеуказанный шаблон обновления данных в случае внесения изменений в постановку задачи или посредством переключения шаблонов обновления данных в случае смены типа реляционной базы данных для использования полученного шаблона обновления данных при последующей генерации программного кода.

2. Способ по п.1, характеризующийся тем, что метаданные хранятся в таблицах реляционной базы

данных.

3. Способ по п.1, характеризующийся тем, что метаданными являются описание структур таблиц, взаимосвязей между ними, правил секционирования, описание витрин данных.

4. Способ по п.1, характеризующийся тем, что метаданные являются константными и расчетными.

5. Способ по п.1, характеризующийся тем, что метаданные размещаются в форме поколоночной структуры объектов и их первичных ключей в базе данных.

6. Способ по п.1, характеризующийся тем, что шаблон обновления данных описан в S2T-файле.

7. Способ по п.1, характеризующийся тем, что дополнительно при осуществлении генерации программного кода получают лог работы, который включает все входные макропеременные, а также трассировочный файл.

8. Способ по п.1, характеризующийся тем, что осуществляют переиспользование метаданных обновления детального слоя и витрин данных за счет внесения дельты изменений в S2T файл.

9. Способ по п.1, характеризующийся тем, что переиспользование метаданных обновления детального слоя и витрин данных осуществляют путем переключения базовых шаблонов обновления.

10. Система автоматической генерации программного кода для корпоративного хранилища данных, содержащая

по меньшей мере одно устройство обработки данных;

по меньшей мере одно устройство хранения данных;

по меньшей мере одну программу, где одна или более программ хранится на одном или более устройстве хранения данных и исполняются на одном и более устройстве обработки данных, причем одна или более программ включает следующие шаги:

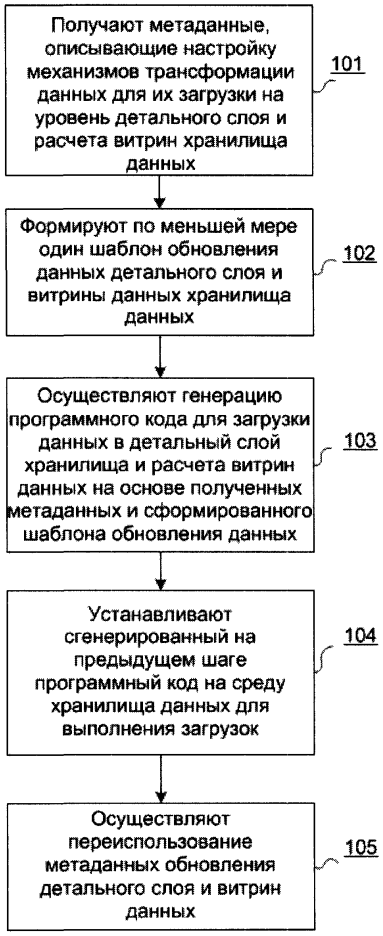
i) получают метаданные, описывающие настройку механизмов трансформации данных для их загрузки на уровень детального слоя хранилища данных и расчета витрин хранилища данных;

ii) формируют по меньшей мере один шаблон обновления данных детального слоя хранилища данных и витрины данных хранилища данных;

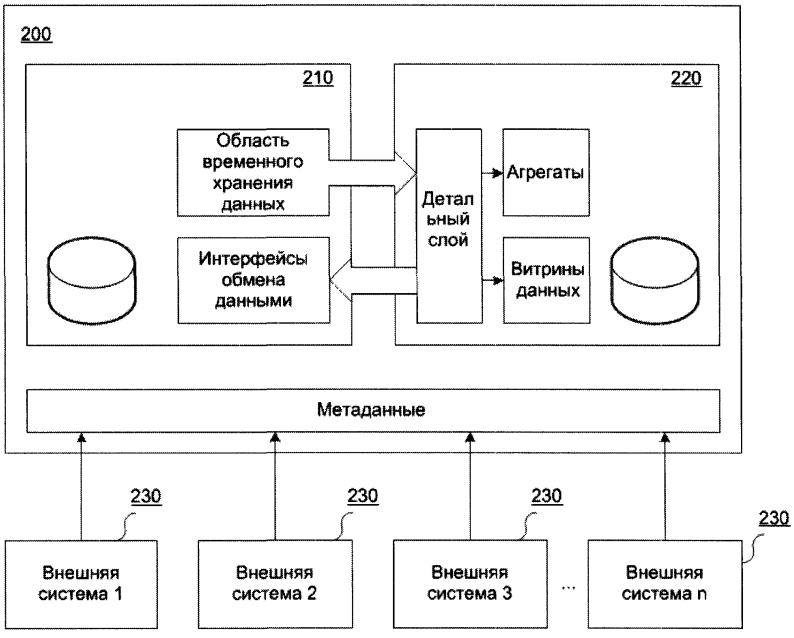
iii) осуществляют генерацию программного кода для загрузки данных в детальный слой хранилища данных и расчета витрин данных на основе полученных метаданных и сформированного шаблона обновления данных;

iv) устанавливают сгенерированный на предыдущем шаге программный код на среду хранилища данных для выполнения загрузок;

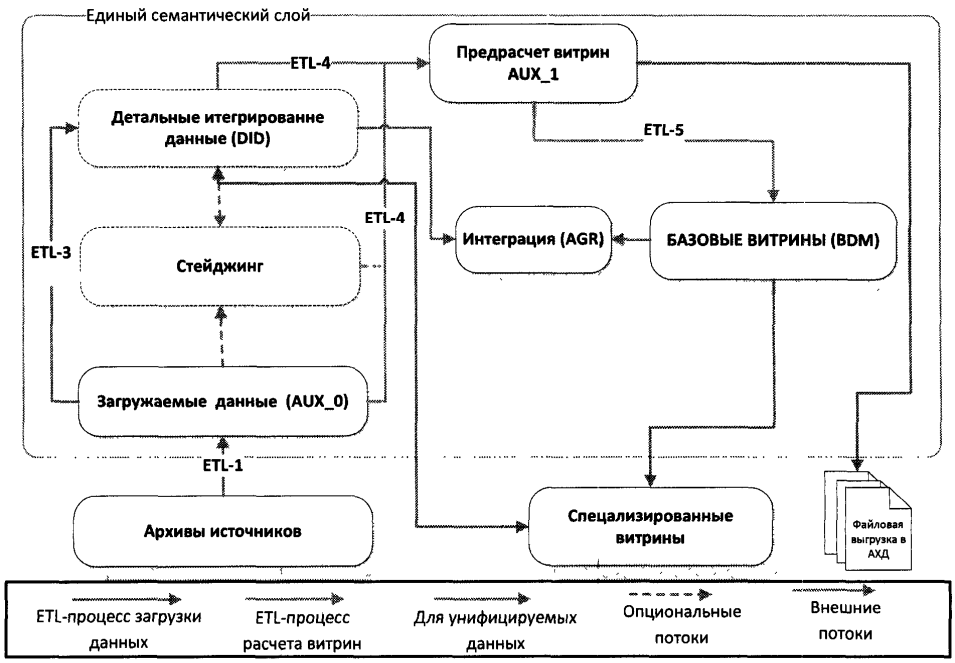
v) осуществляют переиспользование метаданных для обновления детального слоя хранилища данных и витрин данных посредством внесения дельты изменений в вышеуказанный шаблон обновления данных в случае внесения изменений в постановку задачи или посредством переключения шаблонов обновления данных в случае смены типа реляционной базы данных для использования полученного шаблона обновления данных при последующей генерации программного кода.


Получают метаданные, описывающие настройку механизмов трансформации данных для их загрузки на уровень детального слоя и расчета витрин хранилища данных
101
Формируют по меньшей мере один шаблон обновления данных детального слоя и витрины данных хранилища данных
102
Осуществляют генерацию программного кода для загрузки данных в детальный слой хранилища и расчета витрин данных на основе полученных метаданных и сформированного шаблона обновления данных
103
Устанавливают сгенерированный на предыдущем шаге программный код на среду хранилища данных для выполнения загрузок
104
Осуществляют переиспользование метаданных обновления детального слоя и витрин данных
105

Фиг. 1

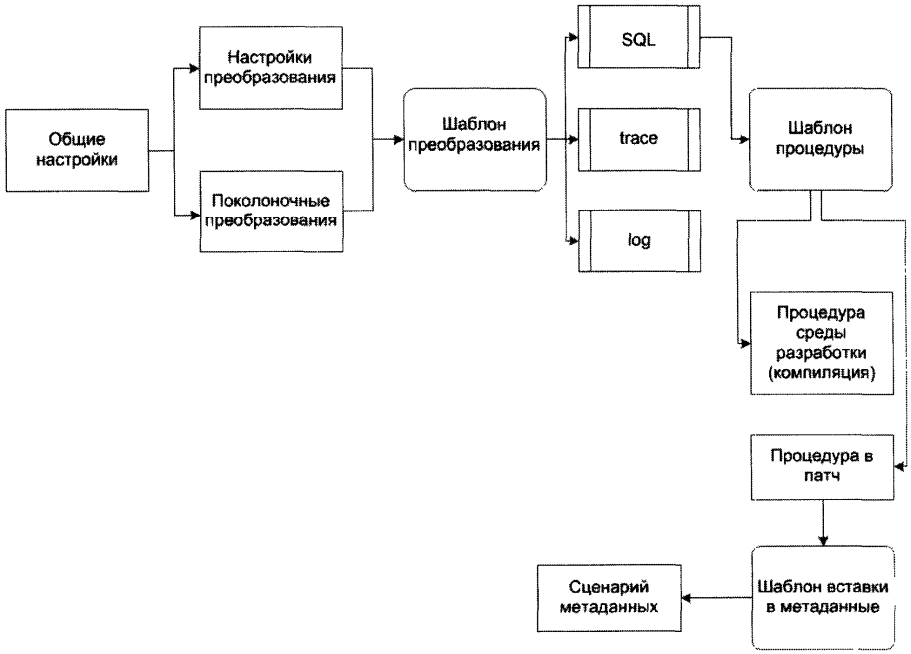

200
210
220
Область временного хранения данных
Интерфейсы обмена данными
Детальный слой
Агрегаты
Витрины данных
Метаданные
230
230
230
230
Внешняя система 1
Внешняя система 2
Внешняя система 3
...
Внешняя система n

Фиг. 2



Фиг. 3

Фиг. 4



Фиг. 5