Титульный лист

Реферат

Содержание

Термины и определения

Перечень сокращений и обозначений

Введение

В компании «ЭР-Телеком» многие статистические данные отслеживаются с помощью OLAP-кубов, состоящих из метрик и измерений. На текущий момент в компании не существует ни одного измерения, построенного на структуре parent-child.

Целью данной работы является доказательство следующей гипотезы:

Можно создать иерархическую структуру parent-child в Oracle Business Intelligence (Oracle BI), которая будет адекватно отображать предметную область "Подразделение" с сохранением ранее созданных связей и данных, а также сможет стать основой при разработке иерархических структур parent-child для других предметных областей компании АО "Эр-Телеком Холдинг".

Далее в тексте «заказчик» равнозначно «компания АО "Эр-Телеком Холдинг"».

Существует достаточно много статей по создании различного типа иерархий в базах данных, но про возможность построения parent-child иерархии упоминается в статье «Oracle BIEE 11g – Parent Child Hierarchies – Multiple Modeling Methods» [<https://www.rittmanmead.com/blog/2010/11/oracle-bi-ee-11g-parent-child-hierarchies-multiple-modeling-methods/> - статья по созданию иерархии], и в технической документации Oracle было найдено описание построения ключевых объектов и связей для создания parent-child иерархии. [https://docs.oracle.com/middleware/12211/biee/BIEMG/GUID-0424E7A7-C7DB-447A-B0C4-0BD6790888EA.htm#hpp\_l\_value\_dimension – документация по OBIEE]. Подробнее об информации в этих источниках говорится в основной части работы.

В отличие от упомянутых источников, в данной работе требуется доказать возможность построения иерархической структуры для предметной области, которая уже существует в Oracle BI, и добавленная структура не должна нарушить имеющиеся связи и должна сохранить имеющиеся данные.

Доказательство гипотезы позволит разрабатывать иерархические структуры parent-child для OLAP-кубов различных предметных областей компании АО "Эр-Телеком Холдинг". Данные разработки позволят снизить трудозатраты аналитиков для построения отчетов и сбора статистики. На текущий момент из имеющихся данных иерархия собирается в Excel, то есть затрачивается время на построение нового отчета из множества других, а затем его анализ. Возможность построение иерархического отчета в Oracle BI позволит избежать ручного построения данных отчетов.

Объект исследования: Измерение «Подразделение», являющееся частью предметной области OLAP-кубов HR.

Предмет исследования: иерархическая структура parent-child.

Для выполнения поставленной целы были выделенные следующие задачи:

1. Проанализировать ПО и инструменты, предоставленные заказчиком для разработки;
2. Проанализировать существующие разработки в этой сфере;
3. Подтвердить или опровергнуть поставленную гипотезу на основе собственных разработок;
4. Описать полученные результаты.

Основная часть

1. Изучение ПО и инструментов
   1. **СУБД**

Данные в компании хранятся в реляционных базах данных, которые обеспечиваются СУБД Oracle.

Реляционная база данных – это связанная информация, представленная в виде двумерных таблиц. Двумерные таблицы состоят из строк и столбцов. У каждой строки есть уникальный идентификатор – это может быть одно поле или несколько – который называется первичным ключом. По нему легко выполнить поиск в таблице одной конкретной строки. Вся информация в базе хранится в виде строк – множества полей, привязанных к конкретному идентификатору, причем число этих полей совпадает с числом столбцов.[1]

Для доступа к реляционным базам данных используется декларативный язык программирования SQL.

SQL (Structured Query Language) — это структурированный язык запросов. На этом языке можно формулировать выражения (запросы), которые извлекают требуемые данные, модифицируют их, создают таблицы и изменяют их структуры, определяют права доступа к данным и многое другое. [2]

СУБД – это программное обеспечение, которое используется для создания и работы с базами данных. Главная функция СУБД – это управление данными (которые могут быть как во внешней, так и в оперативной памяти). СУБД обязательно поддерживает языки баз данных, а также отвечает за копирование и восстановление данных после каких-либо сбоев. [3]

В компании АО «ЭР-Телеком Холдинг» используется СУБД Oracle. В основе большей части Oracle лежит PL/SQL — язык программирования, который предоставляет процедурные расширения используемой в Oracle версии SQL, а также служит языком программирования инструментария Oracle. [4]

СУБД Oracle расположена на двух основных серверах: SA и DWH.

Сервер SA (System Administration) взаимодействует с другими источниками данных, обрабатывает данные, хранит данные короткие промежутки времени. Отсутствует историчность. Лишние данные и таблицы регулярно вычищаются. На этом сервере можно производить основную обработку данных и приведение к необходимому виду.

Сервер DWH позволяет хранить большой объем данных, на нем содержаться исторические данные по многим направлениям деятельности, включая блок продаж, HR-блок, блок выручки/затрат и т.д. Информация на DWH пополняется ежедневно, напрямую записывается в таблицы, без дополнительной обработки. Обработка данных на DWH производится в исключительных случаях.

В одном из источников предложено следующее определение для DWH:

DWH — это система данных, отдельная от оперативной системы обработки данных. В корпоративных хранилищах в удобном для анализа виде хранятся архивные данные из разных, иногда очень разнородных источников. Эти данные предварительно обрабатываются и загружаются в хранилище в ходе процессов извлечения, преобразования и загрузки, называемых ETL. Решения ETL и DWH — это одна система для работы с корпоративной информацией и ее хранения. [5.д https://mcs.mail.ru/blog/chto-takoe-dwh-i-pochemu-bez-nih-dannye-kompanii-bespolezny]

* 1. **OLAP кубы, метрики, измерения; ETL**

[6.д https://cyberleninka.ru/article/n/olap-tehnologii]

Куб OLAP представляет собой структуру данных, которая обеспечивает возможность быстрого анализа данных за рамками ограничений реляционных баз данных. Кубы способны отображать и суммировать большие объемы данных, также предоставляя пользователям доступ к любым точкам данных с возможностью поиска. Таким образом, данные могут быть сведены, фрагментированы и обработаны по мере необходимости для решения самых широкого спектра вопросов, относящихся к интересующей вас области пользователя.

Измерение примерно эквивалентно классу пакета управления. Каждый класс пакета управления имеет набор свойств, а каждое измерение — набор атрибутов, при этом каждый атрибут сопоставляется с одним свойством класса. Измерения позволяют выполнять фильтрацию, группирование и маркировку данных. К примеру, можно отфильтровать компьютеры по установленной операционной системе или сгруппировать людей по категориям, используя пол или возраст. Затем данные могут быть представлены в формате, где данные классифицируются по категориям и категориям, что позволяет более - глубоко анализировать анализ. Измерения также могут иметь естественные иерархии, позволяющие пользователям "детализировать" до более детального уровня детализации. К примеру, измерение даты обладает иерархией, позволяющей выполнять детализацию до уровня лет, затем — до уровней кварталов, месяцев, недель и отдельных дней.

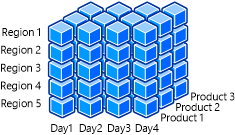


Рис.1 – OLAP куб

В этом рисунке 1 показан куб OLAP, содержащий измерения даты, региона и продукта.

Меры, метрики — это числовые значения, позволяющие пользователям создавать плоскостные и объемные срезы, выполнять агрегирование и анализ. Они являются одной из основных причин построения кубов OLAP на основании инфраструктуры хранилищ данных. При помощи служб SSAS можно создавать кубы OLAP, использующие бизнес-правила и вычисления для форматирования и отображения мер в настраиваемом формате. Большой объем времени разработки куба OLAP тратится на определение того, какие меры будут отображены, и каким образом они будут вычисляться.

Когда пользователь детализирует данные куба OLAP, он анализирует данные на другом уровне уплотнения. Уровень детальности данных повышается с каждой операцией детализации, что позволяет пользователю изучать данные на разных уровнях иерархии. По мере детализации пользователь переходит от общей информации к данным, имеющим более узкий фокус. [5]

На текущий момент в компании АО «ЭР-Телеком Холдинг» реализована уровневая иерархия. Она принципиально отличается от того, что хотели бы видеть конечные пользователи, а следовательно, от того, что требуется построить в ходе исследования. В качестве примера ниже на рис.2 представлена текущая уровневая иерархия по измерению «Календарь» OLAP-куба.

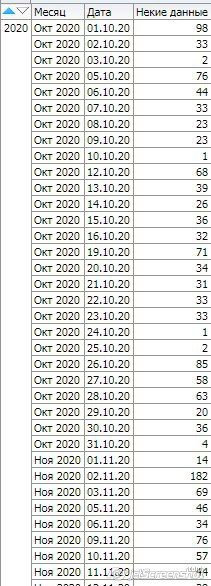


Рис.2 – Данные OLAP-куба в разрезе уровневой иерархии «Календарь»

На рис.2 представлены данные, которые можно посмотреть в разрезе года, месяца, дня. Их можно посмотреть одновременно во всех разрезах, но каждый из них будет выглядеть, как отдельный столбец. Конечные пользователи же хотят видеть иерархию на подобии файловой структуры (рис.3), которую можно свернуть/развернуть и в которой можно посмотреть метрики на каждом уровне иерархии.



Рис.3 – Пример иерархической структуры

На рис.3 представлена иерархическая структура, которая является прототипом того, что конечные пользователи хотят видеть в анализах вместо того, что представлено на рис.2. Видно, что элементы структуры могут содержать элементы нижнего уровня, разворачивать их, и, наоборот, прятать. При этом расчетные метрики должны выводить для каждого уровня корректные значения, включающие как значения самого уровня, так и все внутренних.

* 1. **BI**

Business Intelligence (BI, Бизнес-аналитика) — это набор IT-технологий для сбора, хранения и анализа данных, позволяющих предоставлять пользователям достоверную аналитику в удобном формате, на основе которой можно принимать эффективные решения для управления бизнес-процессами компании. [6]

Все уровни пользователей, от сотрудников до учредителей, получают гибкий доступ к необходимой им управленческой отчетности, не прибегая к помощи IT-специалистов.

Oracle BI – это основной инструмент для аналитической работы с данными, который используют конечные пользователи, предоставленный компанией АО «ЭР-Телеком Холдинг». Именно с помощью этого инструмента строятся анализы, которые должны будут содержать иерархическую структуру. Заказчиком предоставлено обеспечение взаимодействия BI и серверов баз данных. Для сохранения корректного взаимодействия при описании новой структуры данных необходимо придерживаться регламентов компании.

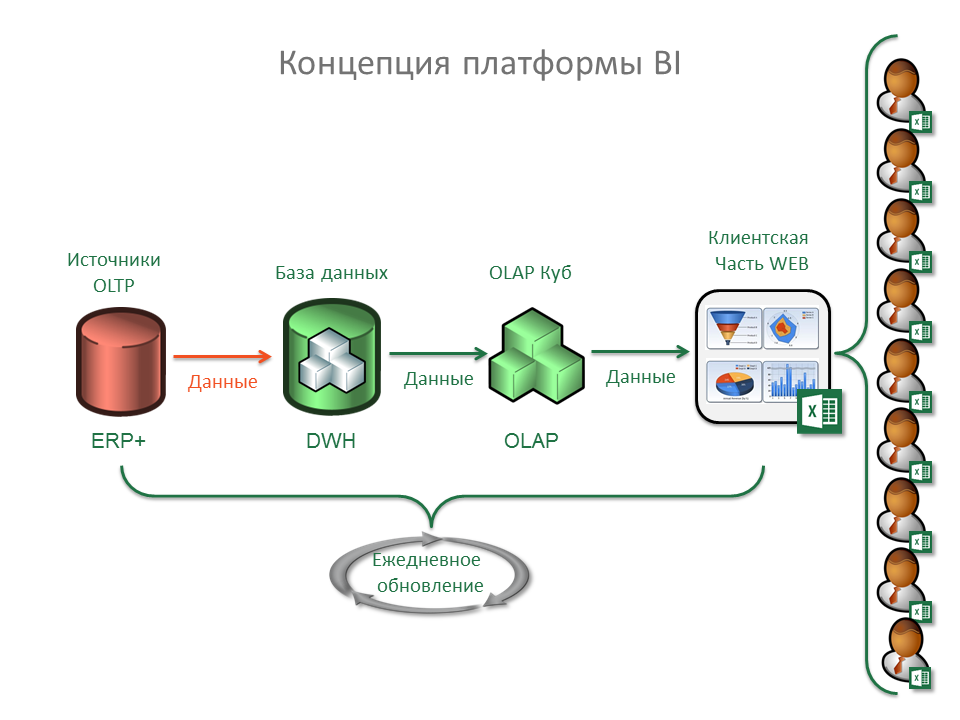
[](http://biweb.ru/wp-content/uploads/2016/09/BI.png)

Рисунок 2 – Концепция платформы BI

Концепция платформы BI, представленная на рисунке 2, состоит из следующих компонентов:

* ETL-инструменты: программы, позволяющие выполнять загрузку данных в DWH из различных учетных систем. Заказчиком в качестве ETL-инструмента предоставлена программа Oracle Date Integrator 11g.
* DWH-хранилище: полноценная база данных SQL для подготовки и хранения данных для аналитики.
* OLAP-кубы: технология, позволяющая делать в реальном времени (1-5 секунд) любые отчеты и проводить полноценный анализ данных.
* Клиентские приложения: как правило, для детального анализа данных и построения динамических отчетов пользователи используют сводные таблицы, подключенные к OLAP-кубам. В компании АО «ЭР-Телеком Холдинг» в качестве клиентского приложения выступает Oracle BI.

Основные преимущества BI:

* Скорость построения отчетов.
* Динамический анализ данных в любой детализации
* Быстрый анализ любого объема данных (технология OLAP).
* Автоматизация подготовки данных для отчетов и построения корпоративной отчетности.
* Консолидация данных (данные для отчетов могут быть в разных учетных системах).
* Анализ показателей План/Факта, Анализ выполнения различных KPI.
* Удобная визуализация данных на Web (при этом обновленные данные в отчет поступают автоматически).
* Единый и удобный доступ к аналитической отчетности для всех сотрудников через корпоративный BI портал.
* Уменьшение нагрузки на Учетные системы.

В целом — повышение общей управляемости и эффективности бизнеса. [Navid Mahlouji. Information Technologies for Business Intelligence// CiteseerX. – 2014. – С. 102 / [Электронный ресурс]. – Режим доступа: URL: https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.978.7425]

* 1. **Инструменты**

Для разработки измерения заказчиком были предоставлены следующие программы: Oracle Data Integrator 11g (ODI), Tool for Oracle Application Developers (TOAD), Oracle Business Intelligence Enterprise Edition (OBIEE). Заказчик строго ограничил использование каких-либо других инструментов, помимо указанных.

Далее приведено краткое описание каждой из используемых программ и их назначение.

Oracle Data Integrator (ODI) — это интеграционная платформа корпоративного уровня, которая обеспечивает извлечение, преобразование и загрузку данных из разнообразных источников: баз данных, файлов и других источников (например, LDAP каталогов или WEB-сервисов). Позволяет создавать автоматические пакеты сбора и обработки данных. [7]

Tool for Oracle Application Developers (TOAD) - инструмент для разработчиков приложений Oracle. Позволяет просматривать базы данных, выполнять PL/SQL код, использовать средства отладки. [8]

Oracle Business Intelligence Enterprise Edition (OBIEE) – инструмент для описания структуры BI. Позволяет описать структуру таблиц, их связи, пути поиска данных при построении анализов. Описанная в АТ структура является основной связи Oracle BI и DWH.

Структура данных в АТ состоит из трех частей:

Физический уровень – здесь хранится полное описание таблиц сервера: название, поля, типы полей, первичные ключи, внешние ключи и так далее.

Бизнес-уровень – здесь хранится описание структур запросов, а именно: к каким таблицам необходимо делать запрос в каком случае, если есть несколько таблиц агрегатов; на какую таблицу связей опираться при создании иерархического запроса; как считать метрики в разрезе измерений; и так далее.

Презентационный уровень – здесь хранится структура в таком виде, в каком видят её пользователи. Сюда выносятся необходимые метрики и измерения, группируются по предметным областям. [http://www.olap.ru/home.asp?artId=2344]

1. Анализ существующих разработок

На текущий момент в компании АО «ЭР-Телеком Холдинг» нет ни одного примера успешной реализации иерархической структуры в BI. Данная разработка является принципиально новой. На текущий момент в компании АО «ЭР-Телеком Холдинг» реализована уровневая иерархия, представленная на рис.2.

Для решения поставленной задачи было исследовано множество литературы. В статье «Oracle BIEE 11g – Parent Child Hierarchies – Multiple Modeling Methods» [<https://www.rittmanmead.com/blog/2010/11/oracle-bi-ee-11g-parent-child-hierarchies-multiple-modeling-methods/> - статья по созданию иерархии] продемонстрировано возможное решение поставленной задачи через иерархическую структуру в OBIEE. Описанная в статье структура не удовлетворяет условиям, поставленным регламентами компании, но доказывает, что существует возможность решения поставленной задачи.

Однако в статье не указано на чем строится данная иерархическая структура. В технической документации Oracle было найдено описание построения ключевых объектов и связей для создания parent-child иерархии. [https://docs.oracle.com/middleware/12211/biee/BIEMG/GUID-0424E7A7-C7DB-447A-B0C4-0BD6790888EA.htm#hpp\_l\_value\_dimension – документация по OBIEE].

Таким образом, уже доказано, что parent-child иерархию возможно построить. Это подразумевает создание многоуровневой иерархии, где каждый элемент, кроме корневых, имеет вышестоящий элемент. В отличие от уровневой иерархий, все элементы измерения иерархии parent-child находятся в одном логическом столбце.

В иерархии parent-child родительский элемент элемента находится в другой строке того же логического столбца, на который указывает родительский ключ. Это отличается от иерархии на основе уровней, где родительский элемент элемента находится в другом логическом столбце в той же строке. Другими словами, навигация в иерархии parent-child следует за значениями данных, а навигация в иерархии на основе уровней следует за структурой метаданных.

В рамках этой работы необходимо доказать, что иерархию parent-child можно построить для предметной области «Подразделение», которая уже существует в Oracle BI и имеет ряд таблиц в базе данных, и добавленная структура не должна нарушить имеющиеся связи и должна сохранить имеющиеся данные.

1. Разработка иерархической структуры для доказательства поставленной гипотезы
   1. **Входные данные**

Рассмотрим имеющиеся данные. Основой разработки является существующее измерение «Типовое подразделение». На текущий момент данные для сбора этого измерения приходят в .csv-файле из сторонней системы (1С). Поля:

* Date – дата выгрузки данных;
* TypicaSubdivision – название типового подразделения;
* TypicaSubdivision\_guid - уникальный идентификатор типового подразделения;
* Organization – название организации, к которой относится типовое подразделение;
* Subdivision – название подразделения, которое относится к этому типовому подразделению;
* Subdivision\_guid - уникальный идентификатор подразделения;
* Basic – является ли типовое подразделение базовым;
* Positions – должность, которая относится к этому типовому подразделению;
* Positions\_guid - уникальный идентификатор должности.

Ниже представлен пример того, как выглядят данные в исходном файле.

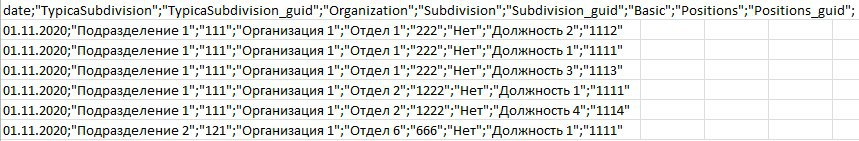


Рисунок 2.1 – Исходные данные

Как видно из рисунка 2.1, о каждом типовом подразделении приходит информация, куда оно входит: организация, - и что в него входит: отдел и должность. Эти данные необходимы для привязки измерения к кубам, у которых нет информации о подразделении, но есть информация об отделах, организациях и должностях. Сочетание этих трех показателей имеет однозначную привязку к подразделению.

Для реализации задачи необходимо добавить в выгрузку следующие поля:

* TypicaSubdivisionParent - название родительского типового подразделения;
* TypicaSubdivisionParent\_guid – уникальный идентификатор.

Однако в рамках реализации задачи был рассмотрен следующий вопрос: если в изначальной выгрузке каждое типовое подразделение дублируется множество раз, то, возможно, будет удобнее создать новую выгрузку, содержащую только информацию об иерархии, чем добавлять поля в старую выгрузку? Приблизительный подсчет показал следующее: строк в изначальной выгрузке порядка 300 тысяч, в то время как Типовых подразделений около 1 тысячи. Соответственно создание новой выгрузки, состоящей только из иерархии (даже с учетом дополнительных данных, например, даты), будет занимать примерно в 120 раз меньше ресурсов, чем дополнение существующей выгрузки. Также был рассмотрен вариант дополнять только одну строку в изначальной выгрузке по каждому подразделению, но в этом случае тратятся ресурсы на стороне BI для обработки и поиска каждой такой строки.

В конечном итоге сделан следующий вывод: необходимо сформировать новую выгрузку со следующими полями:

Date – дата выгрузки данных;

TypicaSubdivision – название типового подразделения;

TypicaSubdivision\_guid - уникальный идентификатор типового подразделения;

TypicaSubdivisionParent - название родительского типового подразделения;

TypicaSubdivisionParent\_guid – уникальный идентификатор;

* 1. **Структура данных**

Рассмотрим, какие необходимо предпринять шаги для создания подходящих таблиц в BI.

Для общего понимания ниже приведено описание структуры BI, предоставленной заказчиком. BI условно делится на три части:

1. SA – сервер, на котором расположена база данных. Взаимодействует с другими источниками данных, обрабатывает данные, хранит данные кубов только в момент сбора или до следующего сбора. Данные измерений хранятся на сервере полностью для корректного сбора кубов.
2. DWH – сервер, на котором расположена база данных. Предназначен для хранения данных, сюда обращаются пользователи через веб-оболочку. Умеет быстро анализировать данные, так как на каждой таблице есть индексы и регулярно происходит сбор статистики.
3. WEB – клиентская часть, через которую аналитики могут строить анализы. Она обращается к DWH за данными. Имеет информацию обо всей структуре данных, куда и зачем обращаться.

Сбор измерения состоит из получения данных, добавления данных в таблицы внутри базы данных, перенос на DWH.

Сбор куба состоит из получения данных, привязки данных к измерениям, добавления данных в таблицы внутри базы данных, перенос на DWH.

В ходе исследования иерархической структуры было выяснено, что для создания новой структуры необходимо:

1. Изменить таблицы, куда поступают и где обрабатываются данные таким образом, чтобы получать новые данные об иерархии;
2. Изменить таблицы хранения данных таким образом, чтобы в измерении было дополнительное поле, обозначающее родителя;
3. Изменить структуру в BI таким образом, чтобы WEB-часть имела возможность создавать иерархические запросы. [9]

Для того, чтобы WEB смог простраивать иерархический запрос, необходима еще одна таблица: таблица связей между элементами, в которой для каждого элемента указывается родитель-дистанция-признак\_листа для каждого вышестоящего элемента в дереве. Поля MEMBER\_KEY, ANCESTOR\_KEY, DISTANCE, IS\_LEAF. На основе этой таблицы WEB сможет от каждого элемента прийти к любому вышестоящему родителю.

Сбор таблицы связей является самым сложным шагом, так как требует больших ресурсов. Для сбора необходимо будет написать и оптимизировать отдельную процедуру, способную для каждого элемента выявить вышестоящих родителей.

Сбор измерения должен происходить в автоматическом режиме. Это реализуется с помощью инструмента ODI. Процедура заполнения таблицы связей должны быть включена в ежедневный сбор.

Описание структуры данных для WEB-части реализуется с помощью инструмента AdminTools. Необходимо указать, что измерение является иерархическим, по каким полям происходит привязка к иерархии, и вынести измерение на презентационный уровень.

**Вывод:** Ключевой момент, который был выяснен в течение изучения предметной области: иерархию parent-child BI поддерживает и её можно настроить на бизнес уровне AdminTools, но для этого нужна особая таблица связей, и перестройка основной таблицы.

В рамках этого необходимо создать новую выгрузку данных, где будет приходить связь parent-child. Необходимо переработать существующие таблицы сбора измерения и создать новую таблицу связей. Необходимо вынести иерархия на презентационный уровень BI.

1. **Разработка иерархического измерения**
   1. **Автоматизация сбора**

Из внешней системы был сгенерирован файл с новой выгрузкой по требуемым условиям. Пример данных из новой выгрузки представлен на рисунке ниже.

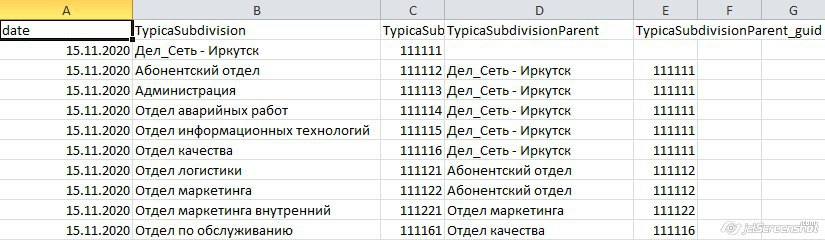


Рисунок 3.1 – Исходные данные

Как видно из рисунка 3.1 сформированная выгрузка соответствует описанию, приведенному в разделе 2.1.

Для принятия данных после перенесения файла на сервер необходима внешняя таблица. Код создания внешней таблицы представлен ниже.

Листинг 3.1 – Код создания внешней таблицы

CREATE TABLE SA\_D\_TYPIC\_SUBDIV\_PARENT\_EXT

(

EXPORT\_DATE VARCHAR2(500 BYTE),

TYPICAL\_SUBDIVISION VARCHAR2(500 BYTE),

TYPICAL\_SUBDIV\_GUID VARCHAR2(500 BYTE),

TYPICAL\_SUBDIV\_PARENT VARCHAR2(500 BYTE),

TYPICAL\_SUBDIV\_PARENT\_GUID VARCHAR2(500 BYTE)

)

ORGANIZATION EXTERNAL

( TYPE ORACLE\_LOADER

DEFAULT DIRECTORY HR\_LOG\_DIR

ACCESS PARAMETERS

( RECORDS DELIMITED BY NEWLINE

FIELDS TERMINATED BY ';'

OPTIONALLY ENCLOSED BY '"'

MISSING FIELD VALUES ARE NULL

)

LOCATION (HR\_DIR:'SVD\_ZUP\_TypicalUnit\_Parent.csv')

)

REJECT LIMIT 0

NOPARALLEL

NOMONITORING;

Рассмотрим пакет сбора, с помощью которого собирается измерение в текущий момент. Сбор реализован с помощью ODI.

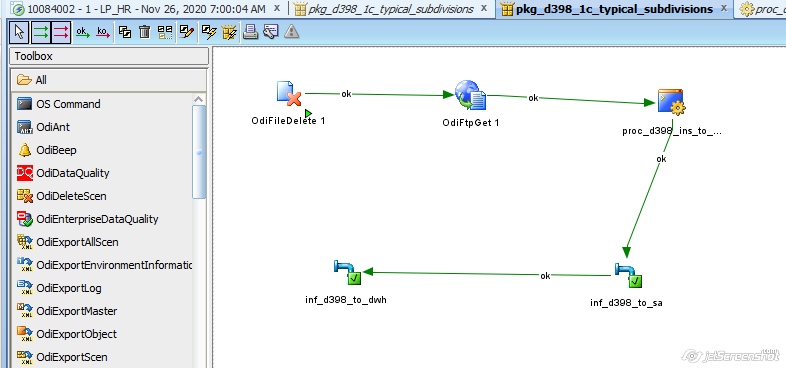


Рисунок 3.2 – Пакет сбора измерения

На рисунке 3.2 представлен пакет сбора измерения до внесения в него изменений. Сбор состоит из пяти шагов:

* 1. Удаление файла с данными на сервере SA за прошлый день.
  2. Перенос нового файла с FTP на сервер SA
  3. Заполнение таблицы трансляций (требуется для дальнейшей привязки кубов к этому измерению с помощью должностей и организаций). Процедура сбора из внешней таблицы.
  4. Сбор измерения на SA из таблицы трансляций. Пополняется измерение новыми данными.
  5. Перенос новых данных в таблицу куба на SA

Что необходимо добавить:

* 1. Перенос нового файла и удаление старого. У нас добавляется еще один файл, где будет содержаться информация о родителе каждого подразделения
  2. Введение и перенос данных в промежуточную таблицу (регламент запрещает сбор измерения напрямую из внешней таблицы).
  3. Изменить сбор измерения (не из таблицы трансляций, а из промежуточной таблицы)
  4. Добавление в собранное измерение данных о родителях подразделений
  5. Процедуру сбора таблицы связей с предварительным очищением этой таблицы

Далее описана работа по автоматизации сбора измерения шаг за шагом.

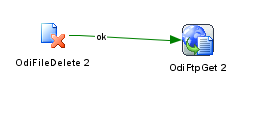


Рисунок 3.3 – Получение внешних данных

На рисунке 3.3 представлены шаги по получению внешних данных. Перенос и удаление файла реализуется с помощью двух специальных инструментов внутри ODI. OdiFileDelete – Удаляет файл с сервера. Сервер прописан в настройках самого ODI, в инструменте достаточно указать путь и название файла.

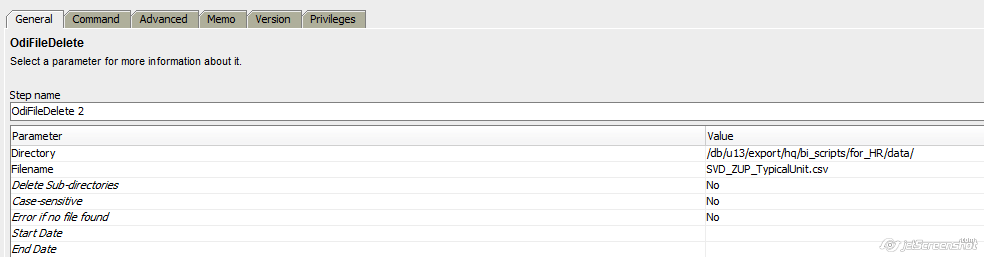


Рисунок 3.4 – Настройки удаления файла с сервера

На рисунке 3.4 представлены настройки, необходимые для заполнения для удаления файла с сервера. В настройках указаны путь и название файла.

OdiFtpGet – инструмент для получения данных с ftp-сервера. В этом инструменте необходимо указать хост сервера, данные для аутентификации, путь до директории, где лежит файл, и так же путь до директории на сервере, куда необходимо переместить файл и его название.

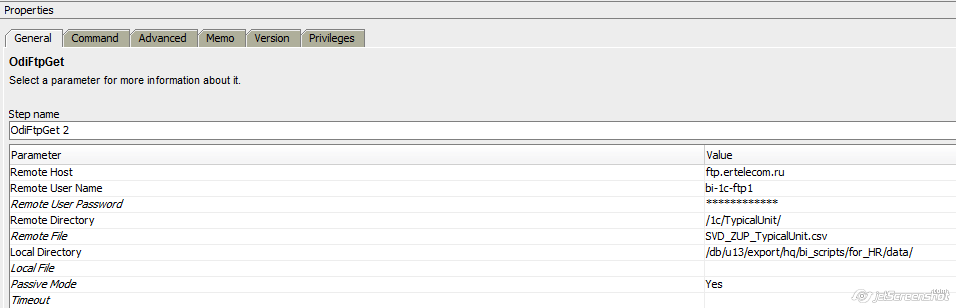


Рисунок 3.5 – Настройки получения файла с сервера

Как видно из рисунка 3.5, в настройках заполняются необходимые поля, чтобы передать данные с ftp-сервера на сервер базы данных.

Следующим шагом необходимо создать промежуточную таблицу, куда будут передаваться данные из внешней таблицы. Код создания представлен ниже.

Листинг 3.2 – Код создания промежуточной таблицы

CREATE TABLE SA\_D\_TYPIC\_SUBDIV\_PARENT

(

EXPORT\_DATE DAte,

TYPICAL\_SUBDIVISION VARCHAR2(500 BYTE),

TYPICAL\_SUBDIV\_GUID VARCHAR2(500 BYTE),

Typical\_Subdiv\_Parent VARCHAR2(500 BYTE),

Typical\_Subdiv\_Parent\_guid VARCHAR2(500 BYTE)

)

NOPARALLEL

TABLESPACE DATA\_SLOW

NOMONITORING;

CREATE UNIQUE INDEX PK\_SA\_D\_TYPIC\_SUBDIV\_PARENT ON SA\_D\_TYPIC\_SUBDIV\_PARENT

(TYPICAL\_SUBDIV\_GUID)

NOLOGGING

TABLESPACE INDEXES\_MEDIUM

NOPARALLEL;

ALTER TABLE SA\_D\_TYPIC\_SUBDIV\_PARENT ADD (

CONSTRAINT PK\_SA\_D\_TYPIC\_SUBDIV\_PARENT

PRIMARY KEY

(TYPICAL\_SUBDIV\_GUID)

USING INDEX PK\_SA\_D\_TYPIC\_SUBDIV\_PARENT

ENABLE VALIDATE);

В существующую таблицу измерения добавляем новый столбец. Код представлен ниже.

Листинг 3.3 – Код изменения таблицы измерения

alter table D\_1C\_TYPICAL\_SUBDIVISIONS add TYPICAL\_SUBDIVISION\_PARENT integer;

Перенос данных из внешней таблицы в промежуточную осуществляется с помощью интерфейса. Ниже рассмотрено подробное создание интерфейса.

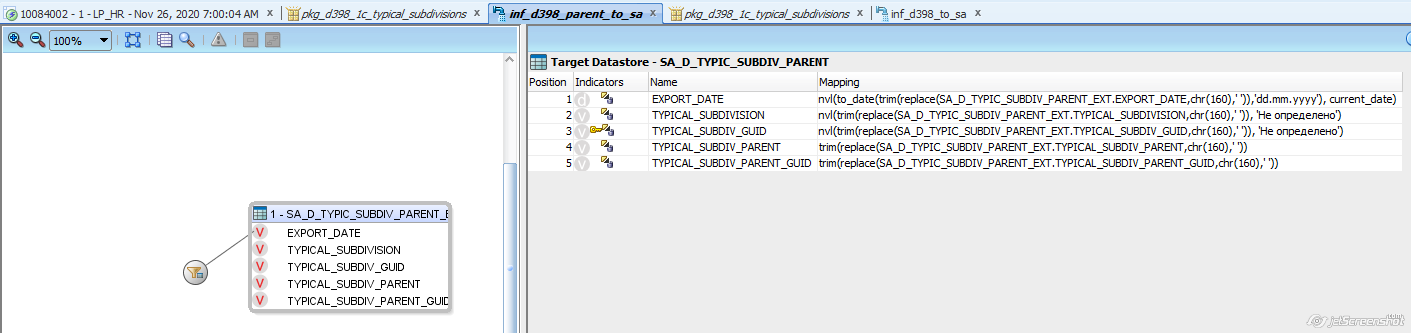


Рисунок 3.6 – Определение таблиц «источник» и «приемник»

На рисунке 3.6 представлен первый шаг создания интерфейса: определение источника и приемника данных, указание строк, которые переносятся из одной таблицы в другую. Слева указан источник данных, справа – приемник. Фильтр отсекает строку с названиями столбцов: SA\_D\_TYPIC\_SUBDIV\_PARENT\_EXT.EXPORT\_DATE != 'date'.

На данном шаге необходимо не просто перенести новые данные, но и обрезать в строках лишние символы. Данные приходят в кодировке Windows-1251 из 1С. Бывает, что там приходят символы, которые дальше при обработке не распознаются, поэтому при переносе в промежуточную таблицу необходимо очистить от таких символов. Самый частый: Код неразрывного пробела — 160. [10]

Функция Oracle / PLSQL TRIM, которая здесь используется, удаляет все указанные символы с начала или окончания строки. Если символ не указан, то удалит пробелы с начала и конца строки.

При выборе таблиц-источников можно указать не одну, а несколько таблиц и связи между ними.

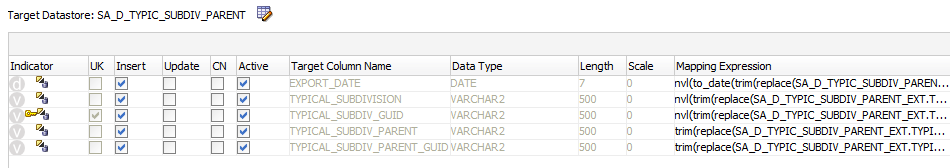


Рисунок 3.7 – Установка обновляемых строк

На рисунке 3.7 представлен второй шаг создания интерфейса. Здесь указывается, какие поля можно обновлять, какие можно добавлять, какие не участвуют в текущем процессе сбора.

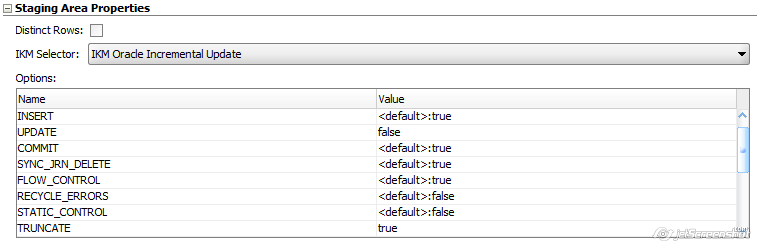


Рисунок 3.8 – Дополнительные параметры

На рисунке 3.8 показана установка дополнительных параметров. В данном примере, необходимо разрешить очистку таблицы перед заполнением и запретить обновление данных. Как итог, данные будут вставляться в пустую таблицу.

На данном этапе интерфейс считается созданным. Далее можно запустить его выполнения для проверки сбора данных.

При выполнении интерфейс генерирует большое количество запросов к базе, в основном DDL и DML запросы.

На рисунке ниже отображены все подоперации, которые генерируются при выполнении интерфейса.

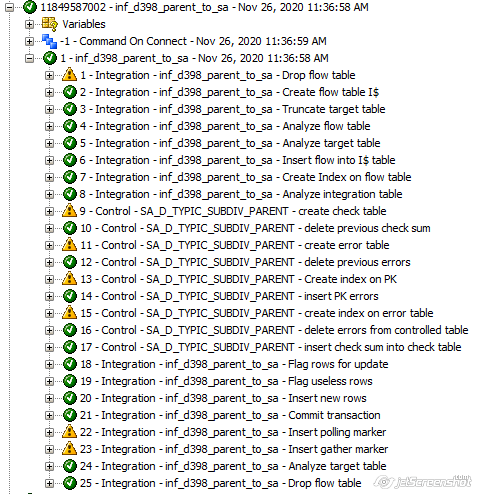


Рисунок 3.9 – Результат выполнения интерфейса

На рисунке 3.9 представлен список подопераций, которые генерируются при выполнении интерфейса. Далее представлено кратное описание того, что происходит на каждом шаге:

1. Удаление потоковой таблицы, которая была использована в прошлый раз. Потоковая таблица – это таблица, куда записываются данные в момент работы интерфейса, оттуда исключаются ошибочные данные, а все оставшееся переносится в таблицу приемник или обновляется в таблице приемнике, если вставка запрещена;
2. Создание потоковой таблицы. Создается по количеству активных полей, которые будут добавляться/изменяться в таблице приемнике, а также специальное поле. Имя генерируется как I$\_название\_таблицы\_приемника;
3. Очищение таблицы приемника, так как в интерфейсе напротив этого пункта стоит галочка. В противном случае данный шаг пропускается;
4. Сбор статистики о потоковой таблице. Это необходимо, чтобы с данными было проще работать, так как после очищения таблицы предыдущая статистика становится неактуальной;
5. Сбор статистики о таблице-приемнике. К этому моменту статистика о таблице уже была с прошлого раза, но сбор статистики на этом этапе позволяет определить изменения в структуре таблицы;
6. Вставка данных в потоковую таблицу. Здесь происходит основная выборка данных из таблицы источника с учетом всех наложенных фильтров и ограничений. Код этого участка сохранен ниже, после описания;
7. Создание индексов по ключевым полям на потоковой таблице;
8. Повторный сбор статистики о потоковой таблице, так как туда уже добавлены данные;
9. Создание чек-таблицы;
10. Очищение чек-таблицы. Удаления таблицы нет, потому что она могла уже существовать и не создаться на предыдущем шаге;
11. Создание таблицы ошибок;
12. Очищение таблицы ошибок. Аналогично чек-таблице, смотреть п.10;
13. Создание индекса на первичном ключе потоковой таблицы. В данном случае этот шаг аналогичен шагу 7. Используется для специфичных случаев, когда на шаге 7 нет возможности сразу ввести индексы;
14. Важный шаг, здесь проверяются данные в потоковой таблице, у которых совпадает первичный ключ. Информация о таких данных заносится в таблицу ошибок;
15. Создание индексов на таблице ошибок, если они отсутствуют;
16. Удаление ошибочных строк из потоковой таблицы;
17. В чек-таблицу заносится информация о количестве данных в потоковой таблице;
18. Проставляются флаги для строк, которые можно обновлять. Этот шаг является пустым в текущем случае, так как обновление запрещено;
19. Проставляются флаги для строк, которые не вставляются и не обновляются. В данном случае пустой шаг, так как выбранная стратегия не подразумевает наличие таких строк;
20. Вставка данных из потоковой таблицы в таблицу-приемник;
21. Commit проведенной транзакции;
22. Пустой шаг (не подходит текущей стратегии);
23. Пустой шаг (не подходит текущей стратегии);
24. Сбор статистики по таблице-приемнику;
25. Удаление потоковой таблицы.

Ниже представлен код, который генерируется на шаге 6 выполнения интерфейса. Данный фрагмент кода сохранен, так как создание интерфейса в первую очередь заменяет ручное написание данной dml-операции. Все остальные шаги созданы для исключения ошибок и корректного сбора данных.

Листинг 3.4 – Код сбора, сгенерированный на шаге 6

/\* DETECTION\_STRATEGY = NOT\_EXISTS \*/

insert /\*+ append \*/ into DEV\_EXCELLENT.I$\_SA\_D\_TYPIC\_SUBDIV\_PARENT

(

EXPORT\_DATE,

TYPICAL\_SUBDIVISION,

TYPICAL\_SUBDIV\_GUID,

TYPICAL\_SUBDIV\_PARENT,

TYPICAL\_SUBDIV\_PARENT\_GUID,

IND\_UPDATE

)

select

EXPORT\_DATE,

TYPICAL\_SUBDIVISION,

TYPICAL\_SUBDIV\_GUID,

TYPICAL\_SUBDIV\_PARENT,

TYPICAL\_SUBDIV\_PARENT\_GUID,

IND\_UPDATE

from (

select

nvl(

to\_date(trim(replace(SA\_D\_TYPIC\_SUBDIV\_PARENT\_EXT.EXPORT\_DATE,chr(160),' ')),'dd.mm.yyyy')

, current\_date) EXPORT\_DATE,

nvl(

trim(replace(SA\_D\_TYPIC\_SUBDIV\_PARENT\_EXT.TYPICAL\_SUBDIVISION,chr(160),' '))

, 'Не определено') TYPICAL\_SUBDIVISION,

nvl(

trim(replace(SA\_D\_TYPIC\_SUBDIV\_PARENT\_EXT.TYPICAL\_SUBDIV\_GUID,chr(160),' '))

, 'Не определено') TYPICAL\_SUBDIV\_GUID,

trim(replace(SA\_D\_TYPIC\_SUBDIV\_PARENT\_EXT.TYPICAL\_SUBDIV\_PARENT,chr(160),' ')) TYPICAL\_SUBDIV\_PARENT,

trim(replace(SA\_D\_TYPIC\_SUBDIV\_PARENT\_EXT.TYPICAL\_SUBDIV\_PARENT\_GUID,chr(160),' ')) TYPICAL\_SUBDIV\_PARENT\_GUID,

'I' IND\_UPDATE

from DEV\_EXCELLENT.SA\_D\_TYPIC\_SUBDIV\_PARENT\_EXT SA\_D\_TYPIC\_SUBDIV\_PARENT\_EXT

where (1=1)

And (SA\_D\_TYPIC\_SUBDIV\_PARENT\_EXT.EXPORT\_DATE != 'date')

) S

where NOT EXISTS

( select 1 from DEV\_EXCELLENT.SA\_D\_TYPIC\_SUBDIV\_PARENT T

where T.TYPICAL\_SUBDIV\_GUID = S.TYPICAL\_SUBDIV\_GUID

)

В результате выполнения данного интерфейса в промежуточной таблице появляются новые данные.

Далее разрабатывается интерфейс сбора измерения из промежуточной таблицы. Он мало чем отличается от прошлого шага, разве что нет необходимости обрезать строки от непечатаемых символов и пробелов, а также нельзя предварительно очищать таблицу. Этот интерфейс уже был создан и в текущей разработке только изменяется, ввиду добавления нового поля. Ниже представлен весь интерфейс сбора.

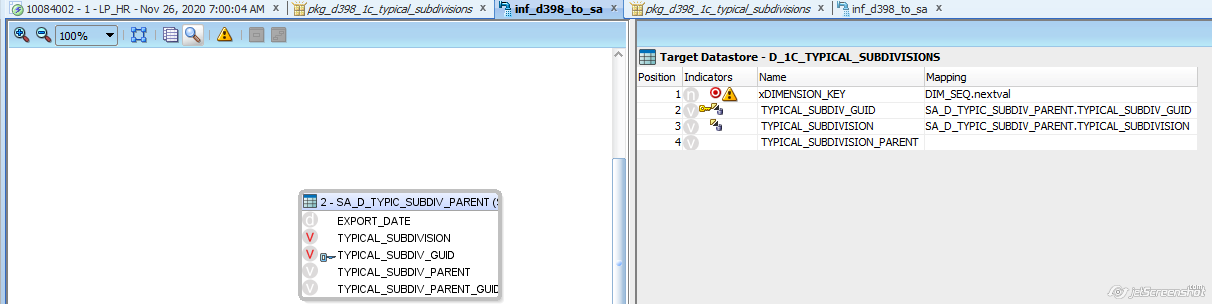


Рисунок 3.10 – Определение таблиц «источник» и «приемник»

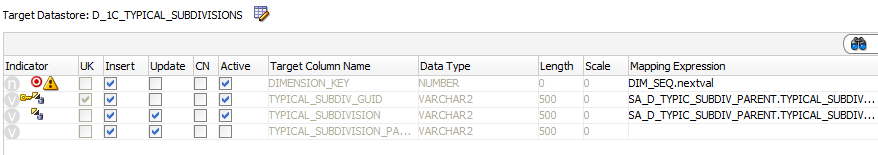


Рисунок 3.11 – Установка обновляемых строк

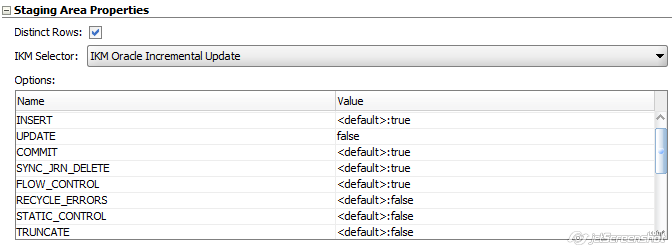


Рисунок 3.12 – Дополнительные параметры

На рисунках 3.10-3.12 представлена разработка интерфейса сбора данных из промежуточной таблицы непосредственно в таблицу измерений.

На этом шаге нет возможности заполнить сразу ключ родительского подразделения, так как происходит вставка новых данных, и родительского подразделения может еще просто не быть в таблице, следовательно, у него еще нет ключа.

Поэтому для заполнения поля, где находится ключ родительского подразделения, создан отдельный интерфейс, который проходит по уже собранным данным в измерении. Ниже представлен сбор этого интерфейса.

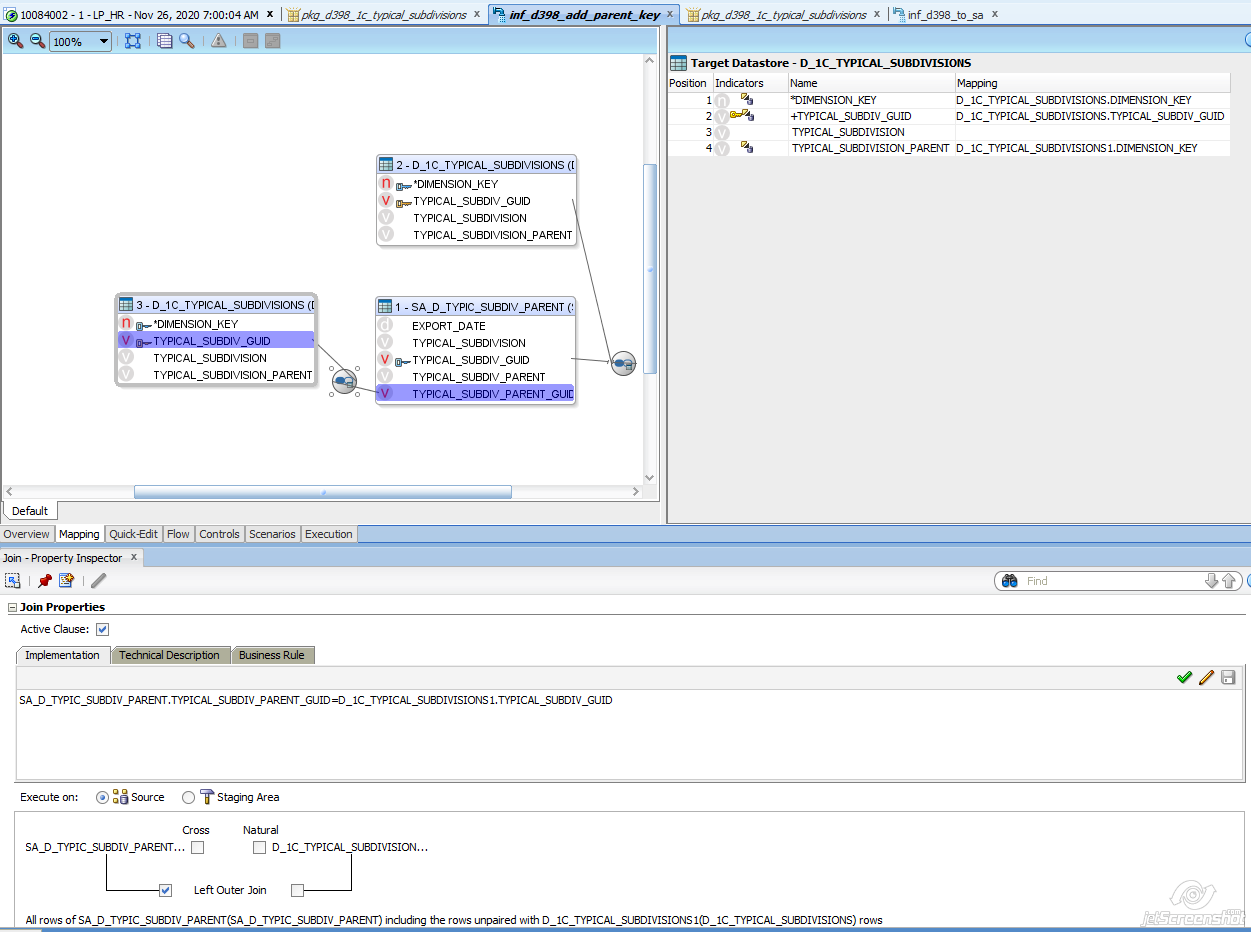
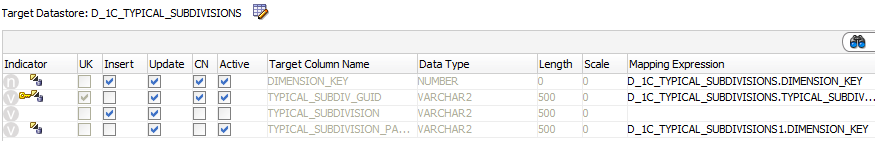
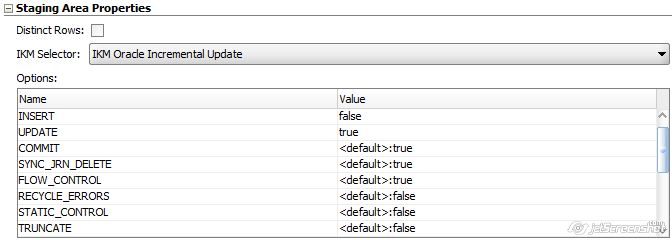


Рисунок 3.13 – Определение таблиц «источник» и «приемник»

  
Рисунок 3.14 – Установка обновляемых строк   
Рисунок 3.15 – Дополнительные параметры

На рисунках 3.13-3.15 представлен интерфейс установки данных о родительском подразделении в измерении. После выполнения этого интерфейса на SA будет находится полностью собранное измерение. Следующим шагом данные переносятся на сервер DWH, но в рамках исходного пакета сбора интерфейс переноса уже есть, он дополняется переносом еще одного поля. Поэтому демонстрацию интерфейса можно опустить.

Последний шаг – сбор таблицы связей на сервере DWH. Описание того, что она в себя включает, уже было выше, поэтому перейдем к описанию процедуры. Реализовать такое с помощью интерфейса нет возможности, так как нам необходимо рекурсивно спускаться по уровням и для каждого подразделения прописывать связь со всеми вышестоящими.

Для начала достаточно узнать количество уровней, которое будет в иерархии, и записать их в переменную. Так мы будем знать, сколько раз мы спускаемся на следующий уровень рекурсии.

Листинг 3.5 – Поиск количества уровней иерархии

declare

v\_max\_depth integer;

begin

select max(level) into v\_max\_depth

from D\_1C\_TYPICAL\_SUBDIVISIONS

connect by prior DIMENSION\_KEY=TYPICAL\_SUBDIVISION\_PARENT

start with TYPICAL\_SUBDIVISION\_PARENT is null;

end;

Дальше можно будет кусочно собрать запрос с помощью цикла.

Листинг 3.6 – Пример цикличной конструкции запроса

for i in 1..v\_max\_depth - 1 loop

...

end loop;

Но кусочный проход по таблице не слишком удобен, так как после каждого нужно использовать commit, а это может привести к пропуску ошибок где-то в середине заполнения таблицы. Поэтому было принято решение использовать динамический PL/SQL и рекурсивно собрать один большой запрос, который дальше можно выполнить за один раз.

Ниже представлен весь запрос полностью.

Листинг 3.7 – Полный код сбора таблицы связей

declare

v\_max\_depth integer;

v\_stmt varchar2(32000);

i integer;

begin

select max(level) into v\_max\_depth

from D\_1C\_TYPICAL\_SUBDIVISIONS

connect by prior DIMENSION\_KEY=TYPICAL\_SUBDIVISION\_PARENT

start with TYPICAL\_SUBDIVISION\_PARENT is null;

*-- Этот кусок посчитал, сколько всего уровней в иерархии.*

v\_stmt := 'insert into D398\_HIERARCHY\_D\_SUBD\_LINK (MEMBER\_KEY, ANCESTOR\_KEY, DISTANCE, IS\_LEAF)' || chr(10)

|| 'select DIMENSION\_KEY as member\_key, null, null, 0 from D\_1C\_TYPICAL\_SUBDIVISIONS where TYPICAL\_SUBDIVISION\_PARENT is null' || chr(10)

|| 'union all' || chr(10)

|| 'select' || chr(10)

|| ' member\_key,' || chr(10)

|| ' replace(replace(ancestor\_key, ''\p'', ''|''), ''\'', ''\'') as ancestor\_key,' || chr(10)

|| ' case when depth is null then 0' || chr(10)

|| ' else max(depth) over (partition by member\_key) - depth + 1' || chr(10)

|| ' end as distance,' || chr(10)

|| ' is\_leaf' || chr(10)

|| 'from' || chr(10)

|| '(' || chr(10)

|| ' select' || chr(10)

|| ' member\_key,' || chr(10)

|| ' depth,' || chr(10)

|| ' case' || chr(10)

|| ' when depth is null then '''' || member\_key' || chr(10)

|| ' when instr(hier\_path, ''|'', 1, depth + 1) = 0 then null' || chr(10)

|| ' else substr(hier\_path, instr(hier\_path, ''|'', 1, depth) + 1, instr(hier\_path, ''|'', 1, depth + 1) - instr(hier\_path, ''|'', 1, depth) - 1)' || chr(10)

|| ' end ancestor\_key,' || chr(10)

|| ' is\_leaf' || chr(10)

|| ' from' || chr(10)

|| ' (' || chr(10)

|| ' select DIMENSION\_KEY as member\_key, TYPICAL\_SUBDIVISION\_PARENT as ancestor\_key, sys\_connect\_by\_path(replace(replace(DIMENSION\_KEY, ''\'', ''\''), ''|'', ''\p''), ''|'') as hier\_path,' || chr(10)

|| ' case when DIMENSION\_KEY in (select TYPICAL\_SUBDIVISION\_PARENT from D\_1C\_TYPICAL\_SUBDIVISIONS ) then 0 else 1 end as IS\_LEAF' || chr(10)

|| ' from D\_1C\_TYPICAL\_SUBDIVISIONS ' || chr(10)

|| ' connect by prior DIMENSION\_KEY = TYPICAL\_SUBDIVISION\_PARENT ' || chr(10)

|| ' start with TYPICAL\_SUBDIVISION\_PARENT is null' || chr(10)

|| ' ),' || chr(10)

|| ' (' || chr(10)

|| ' select null as depth from dual' || chr(10);

for i in 1..v\_max\_depth - 1 loop

v\_stmt := v\_stmt || ' union all select ' || i || ' from dual' || chr(10);

end loop;

v\_stmt := v\_stmt || ' )' || chr(10)

|| ')' || chr(10)

|| 'where ancestor\_key is not null' || chr(10);

execute immediate v\_stmt;

end;

Задается изначальное значение переменной v\_stmt, в которой будет храниться весь запрос.

Листинг 3.8 – Поиск путей от каждой вершины до корневой

select DIMENSION\_KEY as member\_key, TYPICAL\_SUBDIVISION\_PARENT as ancestor\_key,

sys\_connect\_by\_path(replace(replace(DIMENSION\_KEY, '\', '\'), '|', '\p'), '|') as hier\_path,

case when DIMENSION\_KEY in (select TYPICAL\_SUBDIVISION\_PARENT from dev\_bitest.D\_1C\_TYPICAL\_SUBDIVISIONS ) then 0 else 1 end as IS\_LEAF

from dev\_bitest.D\_1C\_TYPICAL\_SUBDIVISIONS

connect by prior DIMENSION\_KEY = TYPICAL\_SUBDIVISION\_PARENT

start with TYPICAL\_SUBDIVISION\_PARENT is null

Представленный выше фрагмент достает путь от каждой вершины до корневой вершины. Является ли вершина листом определяется так: если нет ни одной вершины, для которой она была бы родителем, то это листовой элемент. Результат запроса выглядит следующим образом.

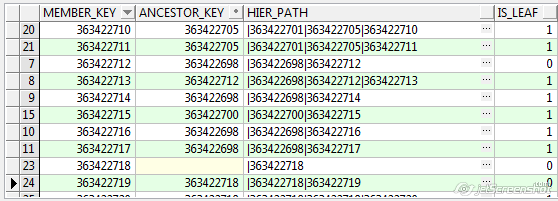


Рисунок 3.16 – Результат запроса

На рисунке 3.16 представлен результат описанного выше запроса. Для каждой вершины определен путь от корневой вершины до неё.

Ниже представлен фрагмент кода, который показывает, каким образом есть возможность опуститься до выбранного элемента, сохраняя все пройденные узлы.

Листинг 3.9 – Рекурсивный спуск по уровням глубины

sys\_connect\_by\_path – это рекурсивный запрос, которому задаются условия:

connect by prior DIMENSION\_KEY = TYPICAL\_SUBDIVISION\_PARENT

start with TYPICAL\_SUBDIVISION\_PARENT is null

Отдельно с помощью цикла получаем все возможные уровни глубины. Ниже представлен фрагмент, расписанный так, как он выглядит в момент запроса. Выше в полном запросе указано, как выглядит данный фрагмент в коде.

Листинг 3.10 – Фрагмент, который заполняется в цикле

select null as depth from dual

union all select 1 from dual

union all select 2 from dual

union all select 3 from dual

union all select 4 from dual…

Результат запроса представлен на рисунке ниже.

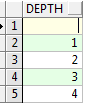


Рисунок 3.17 – Результат запроса

На рисунке 3.17 представлен результат запроса уровней глубины. При полном выполнении кода уровней глубины будет равно количеству уровней иерархии. Далее результаты двух запросов собираются в один.

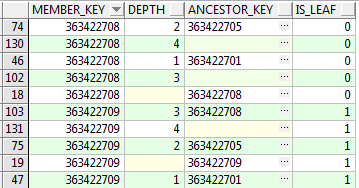


Рисунок 3.18 – Результат запроса

На рисунке 3.18 представлено совмещение двух результатов запроса в один. Для каждого элемента в запросе получается число строк по числу строк в таблице глубины, где на каждом уровне глубины написано, с каким родительским элементом этом уровне связан элемент. Элемент связан с самим собой там, где глубина null. Там, где глубина равна 1, указан вышестоящий элемент для этого элемента. Чем больше значение глубины, тем дальше находится вышестоящий элемент.

В качестве примера рассмотрим элемент 363422709, для которого цепочка связей в иерархии равна |363422701|363422705|363422708|363422709, где 363422701– корневой элемент.

Как видно из рисунка 3.18, связь элементов 363422701 и 363422709 стоит с глубиной 1, так как в данном случае под глубиной подразумевается количество уровней от корневого элемента. Но в таблице связей глубиной должно быть другое значение: насколько элементов далеко данный элемент от выбранного родителя (в данном примере на 3). Необходимые данные можно получить с помощью следующего запроса:

Листинг 3.11 – Получение глубины для каждой вершины от каждой предыдущей

select

member\_key,

replace(replace(ancestor\_key, '\p', '|'), '\', '\') as ancestor\_key,

case when depth is null then 0

else max(depth) over (partition by member\_key) - depth + 1

end as distance,

is\_leaf

from …

Где источником данных являются результаты предыдущих запросов, а именно, результат запроса с рисунка 3.18.

В конечном итоге получаются следующие данные, которые можно вставлять в таблицу связей.

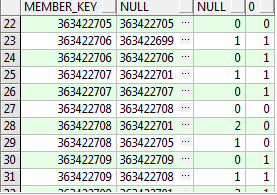


Рисунок 3.19 – Результат запроса

Описанная процедура оформляется в ODI и добавляется в общему пакеты сбора. Конечный пакет сбора измерения представлен на рисунке 3.20.

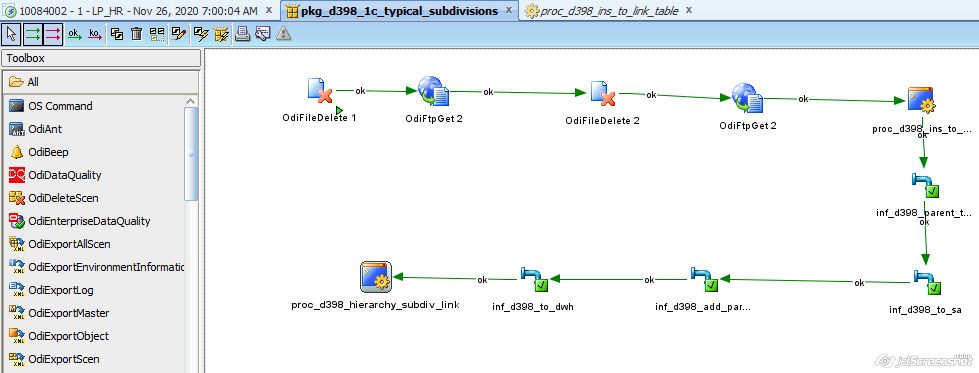


Рисунок 3.20 – Новый пакет сбора измерения

На рисунке 3.20 представлен полный пакет сбора измерения, согласно описанию в разделе 2. Все описанные выше интерфейсы и процедуры включены в сбор.