

Oracle® Agile Product Lifecycle Management for Process Hierarchy Denormalization Guide

Feature Pack 4.1

E64702-01

July 2015

ORACLE®

Copyrights and Trademarks

Agile Product Lifecycle Management for Process

Copyright © 1995, 2015, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and

services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

| | |
|---|-----------|
| PREFACE | 5 |
| Audience | 5 |
| Variability of Installations | 5 |
| Documentation Accessibility | 5 |
| Access to Oracle Support | 5 |
| Software Availability | 5 |
| CHAPTER 1—HIERARCHY DENORMALIZATION INTRODUCTION | 6 |
| Purpose | 6 |
| Overview | 6 |
| CHAPTER 2—INSTALLATION | 7 |
| Preparation | 7 |
| Database Setup | 7 |
| Application Installation | 8 |
| Performance Estimation for Initial Denormalization | 8 |
| CHAPTER 3—CONFIGURATION | 10 |
| CHAPTER 4—EXTENDING HIERARCHY DENORMALIZATION | 12 |
| Detectors | 12 |
| Denormalizers | 12 |
| Relationship Context Definitions | 14 |
| Denormalization Processor | 15 |
| Supported Relationships that are Denormalized in GSM | 18 |
| Supported Relationships that are Denormalized in SCRM | 20 |
| CHAPTER 5—UNDERSTANDING THE HIERARCHY DENORMALIZATION DATA MODEL | 21 |
| Denormalization Results under Nested-Set Model | 23 |
| CHAPTER 6—EXTENSIBILITY REFERENCES | 25 |
| Implementation Example | 26 |

Preface

Audience

This guide is intended for client programmers involved with integrating Oracle Agile Product Lifecycle Management for Process. Information about using Oracle Agile PLM for Process resides in application-specific user guides. Information about administering Oracle Agile PLM for Process resides in the *Oracle Agile Product Lifecycle Management for Process Administrator User Guide*.

Variability of Installations

Descriptions and illustrations of the Agile PLM for Process user interface included in this manual may not match your installation. The user interface of Agile PLM for Process applications and the features included can vary greatly depending on such variables as:

- Which applications your organization has purchased and installed
- Configuration settings that may turn features off or on
- Customization specific to your organization
- Security settings as they apply to the system and your user account

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Software Availability

Oracle Software Delivery Cloud (OSDC) provides the latest copy of the core software. Note the core software does not include all patches and hot fixes. Access OSDC at:

<http://edelivery.oracle.com>

Chapter 1—Hierarchy Denormalization Introduction

Purpose

This guide describes how to install, configure, and use the Hierarchy Denormalization solution pack.

Solution packs are designed to be pluggable modules that can be added to the Agile PLM for Process application suite without modifying the existing release code base.

Overview

Oracle Agile PLM for Process stores objects, such as specifications, along with the relationships to each other, in a normalized database schema, making inserts, updates and deletes highly efficient while minimizing its size. The challenges with having a normalized schema are that it can make custom SQL queries complex and possibly not optimal for bulk data retrieval. For example, to construct a report that returns the entire hierarchy of a trade specification, would require a deep understanding of many relationship tables and would be extremely difficult to do in SQL alone, due to the varying number of possible layers in the hierarchy. A hierarchy of a specification is defined as that specification plus all descendant specifications as well as other related objects. For example these objects would be considered part of a trade hierarchy:

- The main trade specification
- All lower level trade specifications
- The material specification directly associated to the trade specification
- The formula to create the above material
- All inputs and outputs to the above formula
- All formulas that create the above inputs
- All inputs and outputs to the above formulas
- Etc ...

By continuing to drill down into the formula and intermediate formulas that comprise a trade specification, you will have what we are referring to as the Trade Hierarchy. This hierarchy is not limited to the relationships defined above but covers many of the relationships that are defined in PLM for Process.

Hierarchy Denormalization provides a solution to this data access problem by storing the object relationship information in a single table, allowing for simple and performant hierarchy retrieval.

Many solutions can use this table to provide functionality such as hierarchical navigation and reporting.

Chapter 2—Installation

Note: A prerequisite for installing Hierarchy Denormalization is Oracle Agile PLM for Process 6.2.

Preparation

1. Unzip the contents of Feature Pack 4.1.
2. Unzip the contents of the Utilities\DenormServices\HierarchyDenorm\HierarchyDenorm.zip file and note the location.

Database Setup

1. Create a backup of the PLM for Process database.
2. Apply the database scripts.
 - a. SQL Server
 - i. Open a command prompt and navigate to the directory where you unzipped the solution pack.
 - ii. Change directories (cd) to the Database directory.
 - iii. Apply the scripts using the following calls to the ApplyScripts.exe utility:

```
ApplyScripts -c "server=<database_server>;uid=<user>;  
password=<password>;database=<database>" -f HierarchyDenorm.xml
```
 - iv. After the ApplyScripts call, you can confirm that the database scripts have been applied successfully when the system prompts you with the following message: "Complete – with no errors".

- b. Oracle
 - i. Open a command prompt and navigate to the directory where you unzipped the solution pack.
 - ii. Change directories (cd) to the Database directory.
 - iii. Apply the scripts using the following calls to the ApplyScripts.exe utility:

```
ApplyScripts -c "User Id=<user>;Password=<password>; Data  
Source=<datasource>" -dbvendor="orcl" -f HierarchyDenorm-orcl.xml
```
 - iv. After the ApplyScripts call, you can confirm that the database scripts have been applied successfully when the system prompts you with the following message: "Complete – with no errors".

Application Installation

1. Run the HierarchyDenormSetup.exe file from the location where you unzipped the solution pack and follow the onscreen instructions to install the necessary files to your specified PLM for Process directory.

A backup folder called “SolutionPackBackup” is created during the installation to facilitate the uninstall process. Inside of that folder, a new folder called “HierarchyDenorm” is created to hold the files that were changed during the installation.

2. Restart RemotingContainer Service.

Performance Estimation for Initial Denormalization

Hierarchy Denormalization is a background process running on the Remoting Container, which processes updates as needed. After the initial startup of the Remoting Container after installation, all the hierarchies must be denormalized. The amount of time to perform this is dependent on some factors such as hardware performance, number of specifications, depth of hierarchies and Hierarchy Denormalization configuration settings.

For an approximation of how long it will take to complete the initial processing, tests were performed on three data sets. Below are the test details and results.

Testing server (Virtual Machine) information:

- APP server: Xeon 2.93G Dual, 8G RAM, 1000M Intranet, Windows 2008 R2 (64-bit) with IIS7
- DB server A: Xeon 2.93G Dual, 8G RAM, 1000M Intranet, Oracle 11g Release 2 for Windows (64-bit)
- DB server B: Xeon 2.93G Dual, 8G RAM, 1000M Intranet, Oracle 11g Release 2 for Linux

The following is a snapshot of the configuration file “HierarchyDenormConfig.xml” used.

```
<HierarchyDenormConfig>
  <Settings>
    <!-- The interval of the denorm scanning. (unit: seconds) -->
    <PollingIntervalInSeconds_Detector>90</PollingIntervalInSeconds_Detector>
    <PollingIntervalInSeconds_Processor>1</PollingIntervalInSeconds_Processor>

    <!-- The denorm process will stop when CurrentLevel reaches the value. -->
    <DenormMaxLevel>12</DenormMaxLevel>
    <!-- Overwrite current branch's MaxLevel when the processor meets a node which was created by BreakdownComponent resolver. -->
    <DenormMaxLevel_BreakdownComponent>3</DenormMaxLevel_BreakdownComponent>
    <!-- Overwrite current branch's MaxLevel when the processor meets a node which was created by AlternateOutput resolver. -->
    <DenormMaxLevel_AlternateOutput>2</DenormMaxLevel_AlternateOutput>

    <!-- Determine which type of sub-relationships should be resolved. (TargetOnly, HostOnly, Both) -->
    <DenormMode_AssociatedSpec>TargetOnly</DenormMode_AssociatedSpec>
    <!-- Determine how many valid requests will be resolved in a denormalization period. -->
    <DenormMode_RequestBatchCount>200</DenormMode_RequestBatchCount>

    <!-- Determine how many types of logs should be written. (ErrorOnly, WithWarning, WithWarningAndInfo) -->
    <LoggingLevel>WithWarningAndInfo</LoggingLevel>
    <!-- The listed spec types would always skip writing any denorm warning no matter what LoggingLevel is. (Separated by comma) -->
    <LoggingWhiteList_SkipWarning>5816,1004,2147,1009,2280,2121,1006,2076,1010,6500,6501,5002,5001,5012,5019</LoggingWhiteList_SkipWarning>
  </Settings>
</HierarchyDenormConfig>
```

Four database samples with legacy data have been selected for the estimation. Their involved object types were listed in the below table.

| Scope | Object | Spec Type | Table Name |
|-------|----------------------------|-----------|-----------------------------|
| GSM | PrintedPackaging | 2121 | FinishedPackagingSpec |
| | Menu | 6500 | FoodServiceMenuItem |
| | Formulation | 5816 | formulationSpecification |
| | Material | 1004 | MaterialSpec |
| | Trade | 2147 | gsmBaseTradeSpec |
| | Product | 6501 | FoodServiceProduct |
| | Delivered Material Packing | 1010 | PackingSpec |
| | Label | 1006 | LabelingSpec |
| | Packaging | 1009 | PackagingSpec |
| | Equipment | 2280 | gsmEquipmentSpecification |
| | Packing Configuration | 2076 | PackingConfigurationSpec |
| | Activity | 2283 | SpecActivitySpecification |
| | | | |
| SCRM | Company | 5002 | scrmCompany |
| | Facility | 5001 | scrmFacility |
| | SourcingApproval | 5012 | scrmSourcingApproval |
| | NonSpecSourcingApproval | 5019 | scrmSourcingApprovalNonSpec |

- Sample A - Oracle for Windows

Total: ~ 26,000 requests

Duration: ~ 173 minutes

Average: ~ 2.5 requests/second

- Sample B - Oracle for Linux

Total: ~ 26,000 requests

Duration: ~ 172 minutes

Average: ~ 2.5 requests/second

- Indicators on Oracle platform:

RemotingContainer CPU usage:

Average: 6%

Peak: 12%

Valley: less than 1%

RemotingContainer Memory usage:

150MB ~ 350MB

Chapter 3—Configuration

By default Hierarchy Denormalization will execute without the need to update the configuration settings. The configuration settings can be used to change how often the denormalization process runs as well as what will be denormalized. To understand when data will be denormalized, it's helpful to understand the processes involved.

Hierarchy Denormalization is designed as two endless processing services: Denormalization Detector and Denormalization Processor service. The Detector service checks whether the hierarchy was changed after the last time Denormalization Processor was executed. If changes were detected, it will create denormalization requests. The Processor service will orchestrate the execution of denormalizers which does the actual denormalization work. Both the Detector and Processor service are designed to be triggered in a configurable frequency as shown in the table below. This table also shows the configuration for controlling what is denormalized.

The following are the configurable properties located in the “HierarchyDenormConfig/Settings” section of the configuration file located at “<PLM for Process>\config\Extensions\HierarchyDenormConfig.xml”.

| Property Name | Acceptable values | Default Value | Description |
|------------------------------------|-------------------|---------------|---|
| PollingIntervalInSeconds_Detector | Number in seconds | 90 | Interval of detector service running frequency. |
| PollingIntervalInSeconds_Processor | Number in seconds | 100 | Interval of processor service running frequency. |
| DenormMaxLevel | Number | 12 | Denormalized hierarchy tree max depth limitation. |
| DenormMaxLevel_BreakdownComponent | Number | 3 | Denormalized hierarchy branch max depth limitation specific for a Breakdown Component. If current branch has a node whose relationship context is “BreakdownComponent” it would perform a level-limited denormalization. The default value indicates the parent will at most have 3-levels children. |

| Property Name | Acceptable values | Default Value | Description |
|--------------------------------|---|--|--|
| DenormMaxLevel_AlternateOutput | Number | 2 | Denormalized hierarchy branch max depth limitation specific for an Alternate Output. If current branch has a node whose relationship context is "AlternateOutput" it would perform a level-limited denormalization. The default value indicates the parent will at most have 2-levels children. |
| DenormMode_AssociatedSpec | "TargetOnly"; "HostOnly"; "Both" | "TargetOnly" | Determines which type of sub-relationships should be resolved. |
| DenormMode_RequestBatchCount | Number | 200 | Determine how many valid requests will be resolved in a denormalization period. |
| LoggingLevel | "ErrorOnly"; "WithWarning"; "WithWarning AndInfo" | "ErrorOnly" | Determine how many types of logs should be written. |
| LoggingWhiteList_SkipWarning | TypeID string separated by comma | "5816,1004,2147,1009,2280,2121,1006,2076,1010,6500,6501,5002,5001,5012,5019" | The listed spec types would always skip writing any denormalization warning no matter what LoggingLevel is. |

Chapter 4—Extending Hierarchy Denormalization

Hierarchy Denormalization is designed as a pluggable architecture that enables customers to extend the out of the box functionality by adding other relationships to the denormalized table. The two components necessary to make this possible are Detectors and Denormalizers. Detectors are components that determine what and when a relationship should be denormalized. Denormalizers are components that do the actual work of populating the denormalized table. These are both pluggable components written in C#.

A full reference of how to extend Hierarchy Denormalization can be found below in the [Chapter 6—Extensibility References](#) section.

Detectors

The purpose of detectors is to find objects that have been modified since the denormalizers last ran. To determine this it compares last updated dates of the objects to the last run dates of the denormalizers. Each object type will have a specific detector that is responsible for performing this operation.

Detector settings were organized in “HierarchyDenormConfig/Detectors” section of the “<PLM for Process>\config\Extensions\HierarchyDenormConfig.xml” file. It can accept a sequence of “Detector” nodes within the section.

Each “detector” should have an “objectURL” and “id” attribute. The value of “id” attribute should be unique across the section. “objectURL” is configured as the full class name of the detector producing factory.

The user can either add new detectors or customize the existing ones. (See more at [Chapter 6—Extensibility References](#)). Example of Detector settings:

```
<Detectors configChildKey="id">
  <Detector
objectURL="Class:Oracle.PLM4P.SolutionPack.HierarchyDenorm.HDGSMLib.Detectors.RequestArchiveDe
tector,HDGSMLib" id="RequestArchiveDetector"/>
  <Detector
objectURL="Class:Oracle.PLM4P.SolutionPack.HierarchyDenorm.HDGSMLib.Detectors.ActivityDetector
,HDGSMLib" id="ActivityDetector"/>
  ...
</Detectors>
```

Denormalizers

The purpose of denormalizers is to actually populate the denormalized table.

Denormalizer settings were organized in “HierarchyDenormConfig/Denormalizers” section of the “<PLM for Process>\config\Extensions\HierarchyDenormConfig.xml” file. It can accept a sequence of “Denormalizer” nodes within the section.

Each “Denormalizer” should have an “objectURL”, “type” and “dataTable” attribute. The “type” attribute will represent the type id defined in PLM for Process. “objectURL” is configured as the full class name of the denormalizer producing factory. And “dataTable” determinates the table in which the corresponding denormalization results should be saved. Note the available data tables must have been pre-defined in installer scripts. Currently, this solution pack supports the following spec type and repository mappings:

| ClassName | Type | DataTable/Repository |
|-----------------------------------|------|--------------------------|
| IngredientSpecification | 1004 | DENORM_HD_HIERARCHY_GSM |
| LabelingSpecification | 1006 | DENORM_HD_HIERARCHY_GSM |
| PackagingSpecification | 1009 | DENORM_HD_HIERARCHY_GSM |
| PackingSpecification | 1010 | DENORM_HD_HIERARCHY_GSM |
| PackingConfigurationSpecification | 2076 | DENORM_HD_HIERARCHY_GSM |
| FinishedPackagingSpecification | 2121 | DENORM_HD_HIERARCHY_GSM |
| GSMTradeSpecDO | 2147 | DENORM_HD_HIERARCHY_GSM |
| EquipmentSpecification | 2280 | DENORM_HD_HIERARCHY_GSM |
| FormulationSpecification | 5816 | DENORM_HD_HIERARCHY_GSM |
| FoodServiceMenuItemDO | 6500 | DENORM_HD_HIERARCHY_GSM |
| FoodServiceProductDO | 6501 | DENORM_HD_HIERARCHY_GSM |
| SCRMFacilityDO | 5001 | DENORM_HD_HIERARCHY_SCRM |
| SCRMCompanyDO | 5002 | DENORM_HD_HIERARCHY_SCRM |
| SourcingApproval | 5012 | DENORM_HD_HIERARCHY_SCRM |
| SourcingApprovalNonSpec | 5019 | DENORM_HD_HIERARCHY_SCRM |

For example, when a formulation specification, type 5816 is denormalized, the resulting denormalized data will be stored in the “DENORM_HD_HIERARCHY_GSM” table.

For each “Denormalizer”, the relationships that should be denormalized should also be defined. The relationships also have attributes of “objectURL” and “id”. An expected “RelationshipContext” should be bound with a relationship resolver by adding its unique name to the end of the “objectURL” (Refer to “Relationship Context Definitions”).

Customers can create new denormalizers or customize the existing ones. (See more at [Chapter 6](#)—Extensibility References).

Example of Denormalizer settings:

```
<Denormalizer
objectURL="Class:Oracle.PLM4P.SolutionPack.HierarchyDenorm.HDGSMLib.Denormalizers.FormulationDenormalizer,HDGSMLib" type="5816" dataTable="DENORM_HD_HIERARCHY_GSM">
  <Relationships configChildKey="id">
    <Relationship
objectURL="Class:Oracle.PLM4P.SolutionPack.HierarchyDenorm.HDGSMLib.DescendentRelationships.ComActivities,HDGSMLib$PrimaryActivity" id="ComActivities"/>
    <Relationship
objectURL="Class:Oracle.PLM4P.SolutionPack.HierarchyDenorm.HDGSMLib.DescendentRelationships.Co
```

```
mMasterSpec,HDGSMLib$ExplicitMaster" id="ComMasterSpec"/>
...
</Relationships>
</Denormalizer>
```

Relationship Context Definitions

Specifications can be related to each other in different ways. For instance a formulation can have an input BOM item or it can have an alternate BOM item. If that formulation was part of a trade specification then the relationship between the trade and the BOM items would also have certain relationship types. Hierarchy Denormalization captures these relationships types in 2 columns in the DENORM_HD_HIERARCHY_XXXX table, one for the Parent and one for the Ancestor, “fkParentRelationshipContext” and “fkAncestorRelationshipContext”. These fields are foreign keys to the “DENORM_HD_RELATIONSHIP_CTX” table and “DENORM_HD_RELATIONSHIP_CTX_ML” table, which both store the readable relationship type, the latter stores the multilingual types. These tables should not be modified.

Example:

```
SELECT PARENT.Context,ANCESTOR.Context,H.* FROM DENORM_HD_HIERARCHY_XXXX H
LEFT JOIN DENORM_HD_RELATIONSHIP_CTX PARENT ON
H.fkParentRelationshipContext=PARENT.PKID
LEFT JOIN DENORM_HD_RELATIONSHIP_CTX ANCESTOR ON
H.fkParentRelationshipContext=ANCESTOR.PKID
WHERE H.fkAncestor='<AncestorPKID>'
```

| Column | Data type | Description |
|---------------|-----------|--|
| PKID | Number | Identifier |
| Context | Varchar | Relationship context unique name |
| MaxLevelLimit | Number | Indicate the max allowable level under current relationship type |
| IsAlternate | Bool | IsAlternate flag |

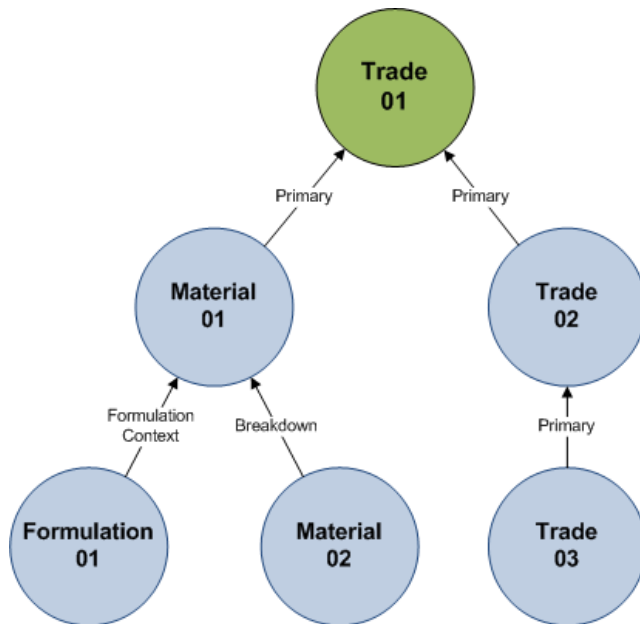
And Table “DENORM_HD_RELATIONSHIP_CTX_ML” is the corresponding translations. Link it for multi-language support.

| Column | Data type | Description |
|---------|-----------|-----------------------------------|
| PKID | Number | Relationship context identifier |
| LangID | Number | Language ID. 0: English |
| Context | Varchar | Relationship context translation. |

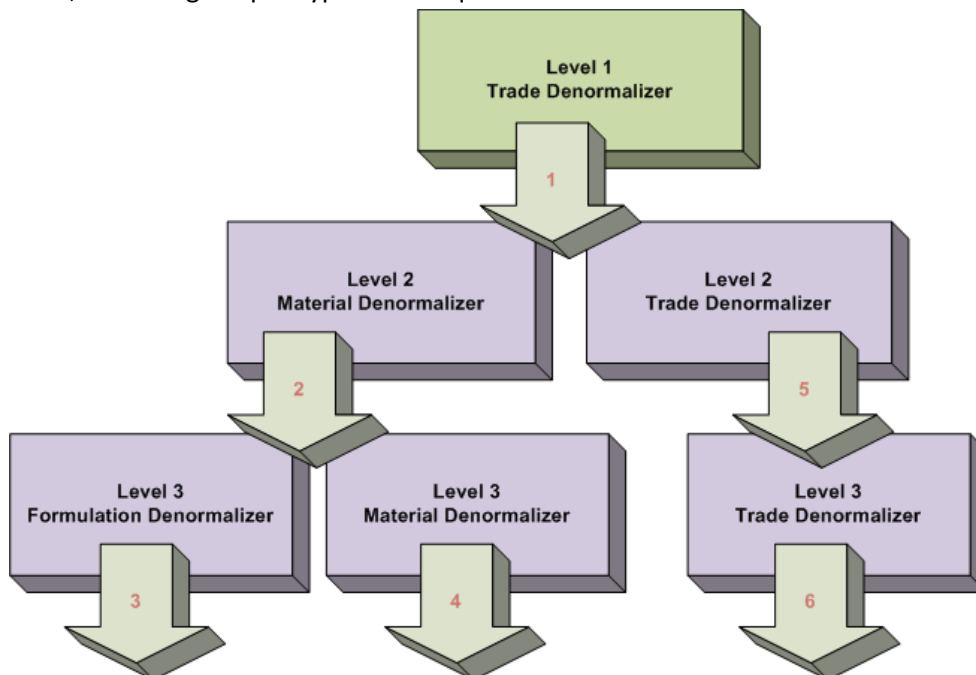
Denormalization Processor

The processor is a background service running in RemotingContainer. The responsibility of this process is to orchestrate the execution of the denormalizers. Each denormalizer is responsible for updating one level of the hierarchy, so the generation of the complete hierarchy will take many denormalizers.

For example, this is an expected tree:



For example, in order to denormalize the above hierarchy, a denormalizer would be executed for each node, according to spec type. The sequence of execution would be similar to the below diagram.



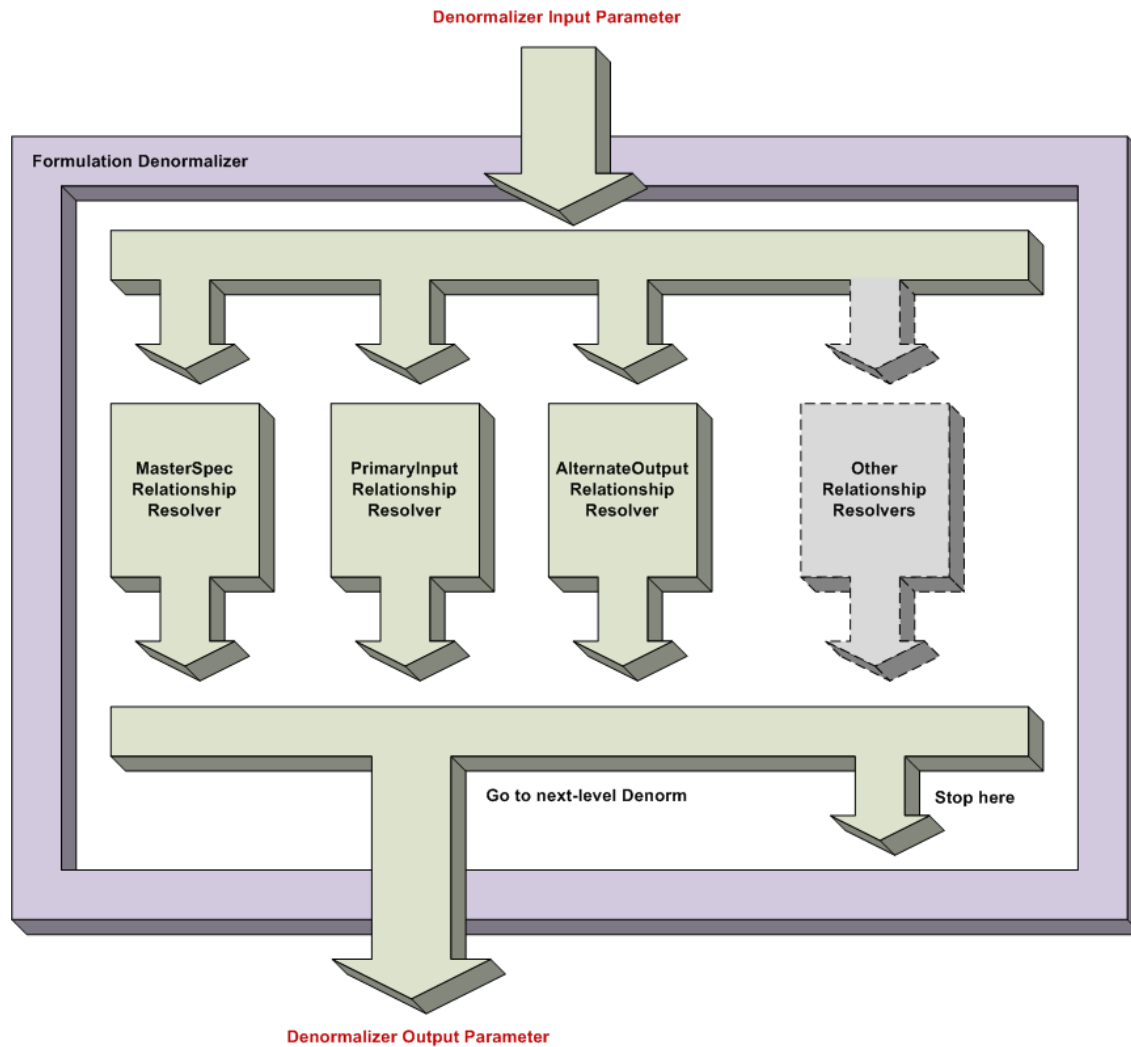
Each denormalizer contains multiple child processes called relationship-resolvers, which are responsible for updating the relationships specific to an object type. For instance if a formulation specification contains a master specification, primary Inputs and an alternate output, within the Formulation Denormalizer, three relationship-resolvers will be executed; the Master Specification relationship-resolver will update the relationships to the master specification, the Primary Input relationship-resolver will update the relationships to the primary inputs and so on.

Customers wanting to add new types of relationships to the denormalized output can add their own relationship resolvers with the runtime context parameters given by the relationship resolver interface.

The input parameter of denormalizer is taking the denormalization context transferred from the previous denormalizer. It includes the parent PKID, the relationship context, some referenced resource entries, and so on. These data ensures the denormalizer finish its work as designed.

Similarly, the output parameter is taking the specific denormalization context that should be transferred to the next denormalizer. Actually, a parent output parameter can be rapidly converted to a child input parameter directly.

This is depicted in the below diagram.



Finally, every denormalizer and its related relationship-resolvers are configurable. The corresponding configuration node for the above FormulationDenormalizer is:

```

<Denormalizer type="5816"
objectURL="Class:Oracle.PLM4P.SolutionPack.HierarchyDenorm.HDGSMLib.Denormalizers.FormulationDenormalizer,HDGSMLib" dataTable="DENORM_HD_HIERARCHY_GSM">
  <Relationships configChildKey="id">
    <Relationship id="ComMasterSpec"
objectURL="Class:Oracle.PLM4P.SolutionPack.HierarchyDenorm.HDGSMLib.DescendentRelationships.ComMasterSpec,HDGSMLib$ExplicitMaster"/>
    <Relationship id="FrmInput"
objectURL="Class:Oracle.PLM4P.SolutionPack.HierarchyDenorm.HDGSMLib.DescendentRelationships.FrmInput,HDGSMLib$Input"/>
    <Relationship id="FrmAlternateOutput"
objectURL="Class:Oracle.PLM4P.SolutionPack.HierarchyDenorm.HDGSMLib.DescendentRelationships.FrmAlternateOutput,HDGSMLib$AlternateOutput"/>
    <Relationship ... />
  </Relationships>

```

```
</Denormalizer>
```

You can find the full official mappings in the configuration file located in the following path “<PLM for Process>\config\Extensions\HierarchyDenormConfig.xml”.

Attention: Each relationship-resolver ObjectURL is taking an additional parameter separated by “\$” character. That means this relationship-resolver is binding with a “Relationship Context” so that the resolver’s outputs can also take the context as its “ParentRelationshipContext” property. Please refer to [Relationship Context Definitions](#) section.

Supported Relationships that are Denormalized in GSM

| Object (Parent) | Related Object (Child) | Relationship | Resolver Name in HDGSMLib |
|-----------------|--|---|---|
| Trade | Trade Specifications | Primary | TrdNextLowerLevelItems |
| | Primary Packaging Specifications | Primary | TrdPackagingMaterials |
| | Alternate Packaging Specifications | Alternate | TrdAlternatePackaging |
| | Material | Primary | TrdRelatedMaterial |
| | Formulation that produces associated Material | FormulationContext | MatFormulationContext |
| | Breakdown Materials of the associated Material | BreakdownComponent | ComBreakdown |
| | Nutrient Profiles | Primary | ComNutrientProfile |
| | Sourcing Approval | Primary | ComSourcingApproval |
| | Associated Specs | Associated | ComAssociatedSpec |
| | Activities | PrimaryActivity | ComActivities |
| | Master Specs | ExplicitMaster | ComMasterSpec |
| Formulation | Materials | Input, Output | FrmInput, FrmOutput |
| | Alternate Materials | Alternate, AlternateOutput | FrmAlternateInput, FrmAlternateOutput |
| | Packaging | Input | FrmInput |
| | Alternate Packaging | Alternate | FrmAlternateInput |
| | Formulation context | FormulationContext, AlternateFormulationContext | MatFormulationContext, MatAlternateFormulationContext |
| | Associated Specs | Associated | ComAssociatedSpec |
| | Activities | PrimaryActivity | ComActivities |
| | Master Specs | ExplicitMaster | ComMasterSpec |
| Menu | Product/Menu | Primary | MenuMenuItemsBuild |
| | Packaging | Primary | MenuPackagingMaterial |
| | Alternate Pkg | Alternate | MenuAlternatePackaging |
| | Global/Regional Standard | - | - |

| Object (Parent) | Related Object (Child) | Relationship | Resolver Name in HDGSMLib |
|--------------------|--------------------------|--------------------|-----------------------------|
| | Alternate Standards | AlternateStandards | ComAlternateStandards |
| | Nutrient Profile | Primary | ComNutrientProfile |
| | Associated Specs | Associated | ComAssociatedSpec |
| | Activities | PrimaryActivity | ComActivities |
| | Master Specs | ExplicitMaster | ComMasterSpec |
| Product | Breakdown Materials | BreakdownComponent | ComBreakdown |
| | Global/Regional Standard | - | - |
| | Alternate Standards | AlternateStandards | ComAlternateStandards |
| | Packing config | Primary | ComPackingConfigurationSpec |
| | Sourcing Approval | Primary | ComSourcingApproval |
| | Associated Specs | Associated | ComAssociatedSpec |
| | Activities | PrimaryActivity | ComActivities |
| | Master Specs | ExplicitMaster | ComMasterSpec |
| Material | Breakdown Materials | BreakdownComponent | ComBreakdown |
| | Substitute Material | Substitute | ComSubstituteMaterial |
| | Packing config | Primary | ComPackingConfigurationSpec |
| | Produced By Formulation | Primary | MatProducedBy |
| | LIO Profile | - | - |
| | Sourcing Approval | Primary | ComSourcingApproval |
| | Associated Specs | Associated | ComAssociatedSpec |
| | Activities | PrimaryActivity | ComActivities |
| | Master Specs | ExplicitMaster | ComMasterSpec |
| Packaging | Sub Components | SubComponent | PkgSubComponents |
| | Packing Config | Primary | ComPackingConfigurationSpec |
| | Equipment | Primary | PkgEquipmentSpec |
| | Substitute Material | Substitute | ComSubstituteMaterial |
| | Sourcing Approval | Primary | ComSourcingApproval |
| | Associated Specs | Associated | ComAssociatedSpec |
| | Activities | PrimaryActivity | ComActivities |
| | Master Specs | ExplicitMaster | ComMasterSpec |
| Equipment | Sub Components | SubComponent | EquSubComponent |
| | Sourcing Approval | Primary | ComSourcingApproval |
| | Associated Specs | Associated | ComAssociatedSpec |
| | Activities | PrimaryActivity | ComActivities |
| | Master Specs | ExplicitMaster | ComMasterSpec |
| Delivered Material | Labeling | Primary | DmatLabelingSpec |
| | Associated Specs | Associated | ComAssociatedSpec |

| Object (Parent) | Related Object (Child) | Relationship | Resolver Name in HDGSMLib |
|-----------------|------------------------|-----------------|----------------------------------|
| | Activities | PrimaryActivity | ComActivities |
| Packing Config | Delivered Material | Primary | PcfgDeliveredMaterialPackingSpec |
| | Associated Specs | Associated | ComAssociatedSpec |
| | Activities | PrimaryActivity | ComActivities |
| Labeling | Associated Specs | Associated | ComAssociatedSpec |
| | Activities | PrimaryActivity | ComActivities |

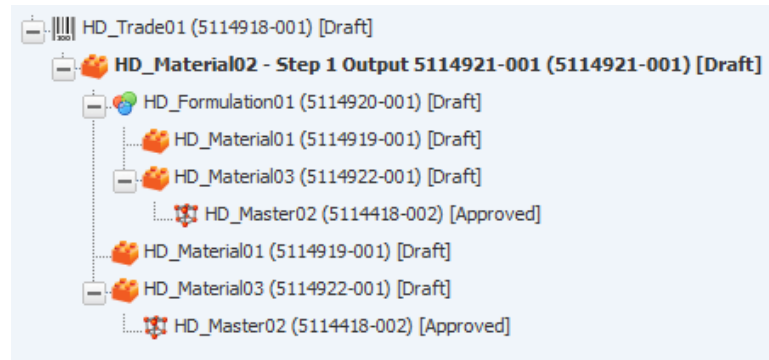
Supported Relationships that are Denormalized in SCRM

| Object (Parent) | Related Object (Child) | Relationship | Resolver Name in HDSCRMLib |
|-----------------------|------------------------|--------------|----------------------------|
| Company | Company | Primary | CompChildCompany |
| | Facility | Primary | CompFacility |
| Facility | Sourcing Approval | Primary | FacNonSAC, FacSAC |
| Sourcing Approval | Specification | Primary | SACSpec |
| | Facility | Primary | ComReceivingFacility |
| Non Sourcing Approval | Facility | Primary | ComReceivingFacility |

Chapter 5—Understanding the Hierarchy Denormalization Data Model

All hierarchies will be stored in the denormalized table, with each node represented by one row. Each row will contain information such as a reference to the parent object, level in the hierarchy and the type of relationship. These nodes are tied together by a column, `fkAncestor`, - that references the relative top node of the hierarchy. By using this column as the query criteria all the nodes for a specific hierarchy can be returned.

For Example:



Retrieve the single tree with this script:

```
select * from DENORM_HD_HIERARCHY_GSM
where fkAncestor='21475f5fc62c-1ccb-4067-80f8-4f5810806fb5';
```

And following is the sample data table from above script:

| PKID | fkAncestor | fkDescendent | fkDescendentParent | CurrentLevel | fkParentRelationshipContext | | |
|-------------------------------|--|--|--|-------------------|-----------------------------|--------|--------|
| 2 | 21475f5fc62c-1ccb-4067-80f8-4f5810806fb5 | 21475f5fc62c-1ccb-4067-80f8-4f5810806fb5 | | 0 | 1 | | |
| 3 | 21475f5fc62c-1ccb-4067-80f8-4f5810806fb5 | 10044dd5a3b8-0bae-4c93-b711-ecaf8f7b4e6a | 21475f5fc62c-1ccb-4067-80f8-4f5810806fb5 | 1 | 6 | | |
| 4 | 21475f5fc62c-1ccb-4067-80f8-4f5810806fb5 | 10044209f6dc-05d8-4c82-9e11-a54a7bb443c5 | 10044dd5a3b8-0bae-4c93-b711-ecaf8f7b4e6a | 2 | 14 | | |
| 5 | 21475f5fc62c-1ccb-4067-80f8-4f5810806fb5 | 100499340adc-87ec-4fd2-adf6-7c93718e5568 | 10044dd5a3b8-0bae-4c93-b711-ecaf8f7b4e6a | 2 | 14 | | |
| 6 | 21475f5fc62c-1ccb-4067-80f8-4f5810806fb5 | 581603d28db4-0ce2-4b13-be70-ae22f6480079 | 10044dd5a3b8-0bae-4c93-b711-ecaf8f7b4e6a | 2 | 13 | | |
| 7 | 21475f5fc62c-1ccb-4067-80f8-4f5810806fb5 | 1005a6fd6f82-31a9-4ad2-92da-332c504b5f80 | 10044209f6dc-05d8-4c82-9e11-a54a7bb443c5 | 3 | 8 | | |
| 8 | 21475f5fc62c-1ccb-4067-80f8-4f5810806fb5 | 10044209f6dc-05d8-4c82-9e11-a54a7bb443c5 | 581603d28db4-0ce2-4b13-be70-ae22f6480079 | 3 | 9 | | |
| 9 | 21475f5fc62c-1ccb-4067-80f8-4f5810806fb5 | 100499340adc-87ec-4fd2-adf6-7c93718e5568 | 581603d28db4-0ce2-4b13-be70-ae22f6480079 | 3 | 9 | | |
| 10 | 21475f5fc62c-1ccb-4067-80f8-4f5810806fb5 | 1005a6fd6f82-31a9-4ad2-92da-332c504b5f80 | 10044209f6dc-05d8-4c82-9e11-a54a7bb443c5 | 4 | 8 | | |
| fkAncestorRelationshipContext | | ObjectType | fkObjectSubType | fkObjectSubTypeEx | fkRelationshipIdentifier | BoxLft | BoxRgt |
| 1 | | 2147 | | | | 1 | 18 |

| | | | | | | |
|---|------|--|--|--|----|----|
| 6 | 1004 | 2210083d0660-b2a6-4256-ad98-3d3dcc2d5d43 | | 5826d5ba6106-6df9-4f97-8dab-8c59a0ed48c3 | 2 | 17 |
| 6 | 1004 | 2210e2704850-c6e1-4c3f-8bd4-7e11bbe862b0 | | 101323a10b8c-45d8-4eb8-8dbf-9f09d138ae24 | 3 | 6 |
| 6 | 1004 | 2210e2704850-c6e1-4c3f-8bd4-7e11bbe862b0 | | 1013fa756721-35a9-4f34-9f68-4a1319b3be7b | 7 | 8 |
| 6 | 5816 | | | 5826d5ba6106-6df9-4f97-8dab-8c59a0ed48c3 | 9 | 16 |
| 6 | 1005 | | | 104058d05bf4-afbe-40ef-bb7e-373aa23217b1 | 4 | 5 |
| 6 | 1004 | 2210e2704850-c6e1-4c3f-8bd4-7e11bbe862b0 | | 5817c375fbf5-b990-42ba-8d7b-b10bfc917303 | 10 | 13 |
| 6 | 1004 | 2210e2704850-c6e1-4c3f-8bd4-7e11bbe862b0 | | 5817d503e570-aa02-4f60-9b58-e67bf550bb1a | 14 | 15 |
| 6 | 1005 | | | 104058d05bf4-afbe-40ef-bb7e-373aa23217b1 | 11 | 12 |

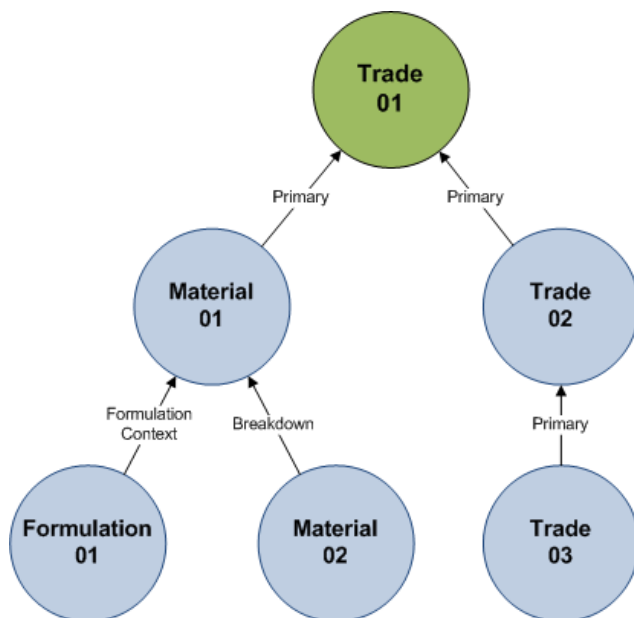
The structure definition of the table is shown in below. The current node's primary key is stored in the fkDescendent column.

| Column | Data type | Description |
|-------------------------------|-----------|---|
| PKID | Number | Primary key |
| fkAncestor | Char | PKID of the top node of the hierarchy. |
| fkDescendent | Char | PKID of the object for the current node of the hierarchy. |
| fkDescendentParent | Char | PKID of the parent object of the current node. |
| CurrentLevel | Number | Level of the hierarchy for the current node. Top node is 0, first level down is 1 and so on. |
| fkParentRelationshipContext | Number | The relationship type between current node and its parent node. (Refer to "Relationship Context Definitions") |
| fkAncestorRelationshipContext | Number | The relationship type between current node and the ancestor node. (Refer to Relationship Context Definitions). |
| ObjectType | Number | The 4 digit object type ID of current node. |
| fkObjectSubType | Char | The first item type ID for current node if it is a Trade, a Material or a Packaging. |
| fkObjectSubTypeEx | Char | The second item type ID for current node if it is a Material. |
| fkRelationshipIdentifier | Char | This is the row PKID when a node exists as one row within in a collection. For example when multiple Materials exist within a Formula as BOM items, this represents the ID that can be tied back to retrieve more |

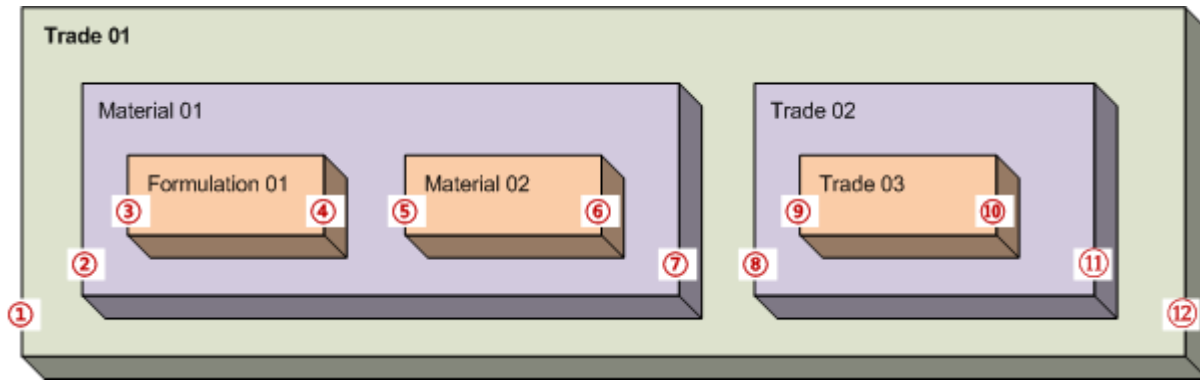
| Column | Data type | Description |
|--------------|-----------|---|
| | | row detail such as Quantity. |
| LastEdit | Datetime | Last edit date of the object represented by this node. |
| Remark | Varchar | (For internal use). |
| MaxLevel | Number | (For internal use). Indicate the max allowable level for current node's children. |
| ContextOwner | Char | (For internal use). Keep the formulation context from parent denormalizer. |
| BoxLft | Number | Nested-set mode LEFT value. Help to identify the tree branch. |
| BoxRgt | Number | Nested-set model RIGHT value. Help to identify the tree branch. |

Denormalization Results under Nested-Set Model

The nested-set model is a high efficient model for storing hierarchical data or trees. A metaphor used to describe this model is that each parent node is a box, and all its children are also boxes inside the parent box. For hierarchies greater than two levels there will be boxes inside of boxes. Each box will have 2 numbers, one for each side, box-left and box-right. These numbers facilitate the retrieval and reconstruction of the hierarchy. The solution guarantees that each box will have two serial numbers (Box-Left and Box-Right). They help to identify each node and facilitate the retrieval of all parents or all children nodes for a selected node. Example:



The visual nested-set model of the picture is like below:



And this is the corresponding data table:

| Object | Level | BoxLft | BoxRgt |
|----------------|-------|--------|--------|
| Trade 01 | 0 | 1 | 12 |
| Material 01 | 1 | 2 | 7 |
| Formulation 01 | 2 | 3 | 4 |
| Material 02 | 2 | 5 | 6 |
| Trade 02 | 1 | 8 | 11 |
| Trade 03 | 2 | 9 | 10 |

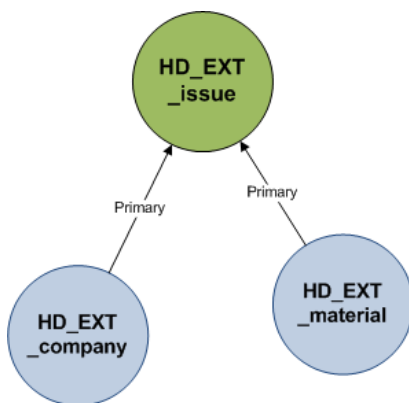
As above shown, those unique numbers (BoxLft and BoxRgt) could help to search or traverse the whole result tree. Assuming there is a node named “X”. Here are some tips:

- When “ $X.BoxRgt - X.BoxLft == 1$ ”, it must be a leaf node.
- If “ $\{NODE\}.BoxLft < X.BoxLft \ \&\& \ \{NODE\}.BoxRgt > X.BoxRgt$ ”, {NODE} must be X’s parent.
- If “ $\{NODE\}.BoxLft > X.BoxLft \ \&\& \ \{NODE\}.BoxRgt < X.BoxRgt$ ”, {NODE} must be X’s child.
- With b) and c), it’s convenient to get the direct parent, the direct children or the whole branch members in the tree as well.

Chapter 6—Extensibility References

Below is a complete implementation of how to extend Hierarchy Denormalization by identifying a new relationship to denormalized, creating a Detector to determine when a relationship has changed and create a request for processing and creating a Denormalizer which will process the request by adding the appropriate data in the denormalization table.

In this example we will use the PQM Issue relationship with its supplier and affected items (material specification).



HD_EXT_issue (10000073)
Pending

[Summary](#)
[Ext Data](#)
[Related Items](#)
[Supporting Documents](#)
[Audit Trail](#)

▼ Summary Information

Title: HD_EXT_issue

Description: HD_EXT_issue

Type: Problem Report

Status: Pending

Issue #: 10000073

Occurrence Date: Thursday, July 25, 2013

Severity:

Resolution:

Workflow: Issue - test

Product Lines:

Customer:

Originator:

Expected Resolution Date:

▼ Suppliers/Facilities

| | Company | Facility |
|---|---------------------------|----------|
| 1 | HD_EXT_company1 (5013923) | |

▼ Affected Items

| | System # | Equivalent # | Description | Rev Found | Failure Type | Qty | Rev Fixed | SKU / GTIN | Site Affected |
|---|----------|--------------|-------------------------|-----------|--------------|-----|-----------|------------|---------------|
| 1 | 5080651 | | HD_EXT_material [Draft] | 001 | | | | | |

▼ Cross References

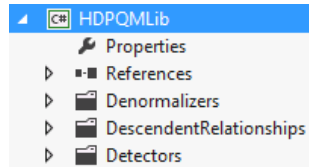
| System Name | System ID | Equivalent | Externally Managed | Status |
|-------------------|-----------|------------|--------------------|--------|
| No records found. | | | | |

Implementation Example

1. Create Visual Studio Project

Open Visual Studio 2010 (or above version) and create a class library.

For this example it is called “HDPQMLib” under .NET Framework 3.5. Normally, it’s strongly recommended to create sub-directories for better organization.



2. Create Detector

Create a Detector for PQM Issue object so that the Denormalization service would know if an Issue has been changed for any reason.

The “PQMIssueDetector” is responsibility to detect the edit event in the UI and deliver the related object PKID(s) to Denormalization request queue.

To add the item to the queue it is possible to use a SQL insert statement to write to the “DENORM_HD_REQUEST” table directly or by using pre-defined HierarchyDenormRequest data object.

Below is an example of the SQL solution.

```
namespace Oracle.PLM4P.SolutionPack.HierarchyDenorm.HDPQMLib.Detectors
{
    public class PQMIssueDetector : DetectorBase
    {
        private const string DETECTOR_SQL = @"insert into DENORM_HD_REQUEST (PKID, fkSpec)
        select NEWID(), a.SpecPKID
        from
        (
            select distinct fkaaffectedObject SpecPKID
            from {0}
            where eventType in (1,2,4) and eventSource in ('PQM.Editor'))
            and timestamp >= {1} and timestamp < {2}
        ) a
        ";

        [ContainsDynamicSQL]
        public override void Process(DetectorCheckpoint c)
        {
            var sql = string.Format(DETECTOR_SQL, EnumMetaClassInfo.LifecycleEventDO.TableName,
                FormatDateTimeValue(c.LastRunDate), FormatDateTimeValue(c.Now));
            var i = executeNonQuery(sql);
        }
    }
}
```

The most critical point is the new Detector must inherit from

“Oracle.PLM4P.SolutionPack.HierarchyDenorm.DetectorService.DetectorBase” and implement the “Process” method. The delivery work should be done in the designated method.

In this case, the Detector makes a scan on the “commonLifecycleEventLog” table in which many events are captured and then inserts the appropriate PKIDs in the “DENORM_HD_REQUEST” table. To help understand the “CommonLifecycleEventLog” better, below are some of the details of what this table captures.

| Action Category | eventType | eventSource |
|---------------------|-----------|--------------------------------------|
| Create | 1 | GSM.CreateFromTemplate |
| | 1 | GSM.Editor |
| | 1 | GSM.SmartIssue |
| | 1 | SCRM.Clone |
| Update | 2 | GSM.Editor |
| | 2 | GSM.GlobalSuccession |
| | 2 | GSM.Workflow.Resolve |
| | 2 | SCRM.Editor |
| | 2 | SCRM.Features.Facility.CompanyChange |
| | 2 | SCRM.Workflow.Resolve |
| Workflow | 3 | GSM.Workflow.Resolve |
| | 3 | GSM.Workflow.Transition |
| | 3 | SCRM.Workflow.Resolve |
| Copy | 4 | GSM.Clone |
| | 4 | GSM.CreateFromTemplate |
| | 4 | GSM.NewIssue |
| | 4 | GSM.SmartIssue |
| | 4 | SCRM.Clone |
| Get Latest Revision | 5 | Revision |
| Substitute | 6 | GSM.Substitute |

Alternatively, the following code shows how to create a request by using the data object.

```

IHierarchyDenormRequest newRequest = new HierarchyDenormRequest("<Target object PKID>");
newRequest.Save();

```

3. Register the Detector

In order for the RemotingContainer to recognize the new Detector it must be registered properly. To do this, open the “<PLM for Process>\config\Extensions\HierarchyDenormConfig.xml” file with any text editor. Add the following node to the “/HierarchyDenormConfig/Detectors” section.

```

<Detector id="PQMIssueDetector"
objectURL="Class:YourCompany.HDPQMLib.Detectors.PQMIssueDetector,HDPQMLib"/>

```

4. Create Denormalizer

Find the specification type ID for the main object, in this example it is the PQM Issue, 7002. This can be done by querying the “ORClassMetaInfo” table.

```

select * from orclassmetainfo where classname = 'PQMIssueDO'

```

Normally, there is no need to create a new specific denormalizer for each object type. Use “Oracle.PLM4P.SolutionPack.HierarchyDenorm.ProcessorService.DenormalizerBase” instead if the current target object doesn’t have to overwrite some denormalization runtime Context. In current release version, only FormulationSpecification uses this feature. This screenshot helps to explain the concept.

Oracle 5114938-001: HD_Formulation02 - Agile PLM for Process: GSM Login Name: [User]

HD_Formulation02 (5114938-001) Draft ContextOwner. It should be transferred to next level. Otherwise, some dependent children may miss.

Summary Formulation Process Ext Data Related Specs CSS Supporting Documents References Approval/Audit Trail

Inputs

| Step | Material | Qty | G/L | Yld | % Batch | USD/100g | EXT Cost |
|------|---|------------|---------|------------|---------|----------|----------|
| 1 | HD_Material02 - Step 1 Output 5114921-001 | 0.00000 lb | 1.00000 | 0.000 lb | 0.00000 | 0.000 | 0.00000 |
| | Context: HD_Formulation01 (5114920-001) | | | | | | |
| | (5114921-001) | 0.00000 lb | | 0.00000 lb | 0.00000 | | 0.00000 |

Without above ContextOwner, current spec wouldn't have this child. So ContextOwner must be taken with denormalization runtime context object.

Outputs

| Output | Input | Material | Output Type | Qty | Material G/L | Moisture G/L | Solids G/L | Yld | % Formula |
|--------|-------|---------------------------|-------------|------------|--------------|--------------|------------|------------|-----------|
| 1 | | Step 1 Output 5114939-001 | Internal | -- | -- | -- | -- | -- | -- |
| | | | | 0.00000 lb | | | | 0.00000 lb | |

Steps

| Step name | Step Qty | Step Yield |
|-----------|------------|------------|
| 1 Step 1 | 0.00000 lb | 0.00000 lb |

As a result, this configuration node should be added to “/HierarchyDenormConfig/Denormalizers” section.

```
<Denormalizer type="7002"
objectURL="Class:Oracle.PLM4P.SolutionPack.HierarchyDenorm.ProcessorService.DenormalizerBase,H
ierarchyDenormProcessorService" dataTable="DENORM_HD_HIERARCHY_PQM">
  <Relationships configChildKey="id">
  </Relationships>
</Denormalizer>
```

Please note the marked “dataTable” property here. This property determines the repository of the data generated by this Denormalizer. The direct table name should be given there; however, the user must ensure the target data table exists. If not, use following scripts to create a new repository for the new customized denormalization.

```
-- SQL Server
IF EXISTS(SELECT 1 FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = 'DENORM_HD_HIERARCHY_PQM')
  DROP TABLE DENORM_HD_HIERARCHY_PQM
GO
CREATE TABLE DENORM_HD_HIERARCHY_PQM(
  PKID int NOT NULL,
  fkAncestor char(40) NOT NULL,
  fkDescendent char(40) NOT NULL,
  fkDescendentParent char(40) NULL,
  CurrentLevel int NOT NULL,
  fkParentRelationshipContext int NOT NULL,
  fkAncestorRelationshipContext int NOT NULL,
```

```

        ObjectType int NOT NULL,
        fkObjectSubType char(40) NULL,
        fkObjectSubTypeEx char(40) NULL,
        fkRelationshipIdentifier char(40) NULL,
        LastEdit datetime NOT NULL CONSTRAINT DF_DENORM_HD_HIERARCHY_PQM_LastEdit DEFAULT
(getdate()),
        Remark nvarchar(512) NULL,
        MaxLevel int NOT NULL,
        ContextOwner char(40) NULL,
        BoxLft int NOT NULL,
        BoxRgt int NOT NULL,
        CONSTRAINT PK_DENORM_HD_HIERARCHY_PQM PRIMARY KEY CLUSTERED
        (
            PKID ASC
        )
    )
GO

-- Oracle
DECLARE cnt NUMBER;
BEGIN
    cnt:=0;
    Select count(*) into cnt from user_tables where table_name =
upper('DENORM_HD_HIERARCHY_PQM');
    If(cnt>0) then
        execute immediate ('DROP TABLE DENORM_HD_HIERARCHY_PQM');
    End if;
    execute immediate ('
CREATE TABLE DENORM_HD_HIERARCHY_PQM(
    PKID number(10,0) NOT NULL,
    fkAncestor char(40) NOT NULL,
    fkDescendent char(40) NOT NULL,
    fkDescendentParent char(40) NULL,
    CurrentLevel number(10,0) NOT NULL,
    fkParentRelationshipContext number(10,0) NOT NULL,
    fkAncestorRelationshipContext number(10,0) NOT NULL,
    ObjectType number(10,0) NOT NULL,
    fkObjectSubType char(40) NULL,
    fkObjectSubTypeEx char(40) NULL,
    fkRelationshipIdentifier char(40) NULL,

```

```

LastEdit timestamp DEFAULT CURRENT_TIMESTAMP,
Remark nvarchar2(512) NULL,
MaxLevel number(10,0) NOT NULL,
ContextOwner char(40) NULL,
BoxLft number(10,0) NOT NULL,
BoxRgt number(10,0) NOT NULL,
CONSTRAINT PK_DENORM_HD_HIERARCHY_PQM PRIMARY KEY
(
    PKID
)
);
END;

```

5. Create Relationship Resolvers

Next create two relationship resolvers for “Suppliers” and “AffectedItems”.

This will be similar to the Detector development. The user should create two classes named “IssueAffectedItem” and “IssueSupplier” under the DescendentRelationships directory of the project.

Both of them must inherit from

“Oracle.PLM4P.SolutionPack.HierarchyDenorm.HDPQMLib.DescendentRelationshipBase” and implement the “GetDescendents” method. In the implementation of the method, with the input parameter (packaging the denormalization context of current scenario), “IssueAffectedItem” resolver should retrieve the AffectedItem relationship children of current PQM Issue whose PKID was being assigned in “input.fkDescendentParent”. And “IssueSupplier” resolver should retrieve the Supplier relationship children of current PQM Issue specification whose PKID was being assigned in “input.fkDescendentParent” as well. In fact, the input parameter should have provided whatever the task needs.

This is the demo code:

```

namespace Oracle.PLM4P.SolutionPack.HierarchyDenorm.HDPQMLib.DescendentRelationships
{
    public class IssueAffectedItem : DescendentRelationshipBase
    {
        public override ICollection<IRelationshipOutput> GetDescendents(IRelationshipInput input)
        {
            var r = new ReferencedObjectPropertyCollectionRetriever(
                input.fkDescendentParent,
                "AffectedItems",
                "ItemInternalID"
            );
            return r.Retrieve();
        }
    }
}

```

```

namespace Oracle.PLM4P.SolutionPack.HierarchyDenorm.HDPQMLib.DescendentRelationships
{
    public class IssueSupplier : DescendentRelationshipBase
    {
        public override ICollection<IRelationshipOutput> GetDescendents(IRelationshipInput input)
        {
            var r = new ReferencedObjectPropertyCollectionRetriever(
                input.fkDescendentParent,
                "PQMSummary.Suppliers",
                "ItemInternalID"
            );
            return r.Retrieve();
        }
    }
}

```

The solution relies on the reflection feature provided by PLM for Process core. It will be helpful to make a basic understanding of the PLM for Process objects.

Then the object property retrievers provided by this solution pack would help to get the direct PKIDs, and then wrap them as an output for next phase. The actual code is not complex.

6. Register the Resolvers

The new resolvers should be registered in configuration.

Add the two nodes to “/HierarchyDenormConfig/Denormalizers/Denormalizer/Relationships” section. Put them under the “7002” one.

```

<Relationship id="ComAffectedItem"
objectURL="Class:Oracle.PLM4P.SolutionPack.HierarchyDenorm.HDPQMLib.DescendentRelationships.IssueAffectedItem,HDPQMLib$PrimaryXApp"/>
<Relationship id="ComCompany"
objectURL="Class:Oracle.PLM4P.SolutionPack.HierarchyDenorm.HDPQMLib.DescendentRelationships.IssueSupplier,HDPQMLib$PrimaryXApp"/>

```

Meanwhile, the expected relationship context for current resolver will be assigned here. Refer to [Relationship Context Definitions](#) for more detail about the additional parameter at the end of “objectURL” value.

7. Create DLL

Finally, compile the new project to be a DLL file and put it to “<PLM for Process>\RemotingContainer\dependentAssemblies”.

