Hewlett Packard Enterprise

HPE Structured Data Manager

Software Version: 7.51

Concepts Guide

Document Release Date: August 2017

Software Release Date: August 2017

Legal notices

Warranty

The only warranties for Hewlett Packard Enterprise Development LP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HPE shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted rights legend

Confidential computer software. Valid license from HPE required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright notice

© Copyright 2016-2017 Hewlett Packard Enterprise Development LP

Trademark notices

Adobe™ is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

This product includes an interface of the 'zlib' general purpose compression library, which is Copyright © 1995-2002 Jean-loup Gailly and Mark Adler.

Documentation updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent software updates, go to https://softwaresupport.hpe.com/patches.

To verify that you are using the most recent edition of a document, go to https://softwaresupport.hpe.com/manuals.

This site requires that you register for an HPE Passport and sign in. To register for an HPE Passport ID, go to https://hpp12.passport.hpe.com/hppcf/login.do.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HPE sales representative for details.

Support

Visit the HPE Software Support Online web site at https://softwaresupport.hpe.com.

This web site provides contact information and details about the products, services, and support that HPE Software offers.

HPE Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Access product documentation
- Manage support contracts
- Look up HPE support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HPE Passport user and sign in. Many also require a support contract.

To register for an HPE Passport ID, go to https://hpp12.passport.hpe.com/hppcf/login.do.

To find more information about access levels, go to https://softwaresupport.hpe.com/web/softwaresupport/access-levels.

Contents

About this document

HPE Structured Data Manager provides powerful tools to design an archive solution that copies or moves data out of your production database and into less expensive storage.

This guide provides conceptual information in the following areas:

- installation
- archive data store
- design
- deployment
- selection
- data movement
- Groovy scripting

Prerequisites

Prerequisites for using this product include:

- Knowledge of the operating system
- Database knowledge
- Application knowledge

Intended audience

This guide is intended for:

- Archive developers building custom archive projects
- Archive developers customizing existing archive projects

New and revised information

This document includes the following new and revised features in HPE Structured Data Manager:

- Archive Query Server Cache
- Partition-based delete for Oracle
- Enhanced Web Console user interface

See the *HPE Structured Data Manager Installation Guide* for more details about new and revised features for this release.

Related documentation

- *HPE Structured Data Manager API Reference Guide*

  Provides information about the Groovy script API files for HPE Structured Data Manager.

- *HPE Structured Data Manager Concepts Guide*

  Explains the major concepts of database archiving in general and HPE Structured Data Manager in particular.

- *HPE Structured Data Manager Installation Guide*

  Explains how to use the Installer to install the product.

- *HPE Structured Data Manager Tutorial*

  Provides step-by-step instructions to build a sample archiving module, deploy it, run it, and troubleshoot errors.

- *HPE Structured Data Manager Developers Guide*

  Explains how to use the Designer component to design, build, test, and deploy your archiving projects.

- *HPE Structured Data Manager Runtime Guide*

  Explains how to use the Web Console component to run, monitor, and administer business flows that move data to and from the database.

- *HPE Structured Data Manager Troubleshooting Guide*

  Explains how to diagnose and resolve errors, and provides a list of common errors and solutions.

- *HPE Structured Data Manager Upgrade Guide*

  Explains how to upgrade the product and upgrade the archive schema generated by earlier versions of the product.

- *HPE Structured Data Manager Release Notes*

  Lists any items of importance that were not captured in the regular documentation.

- *HPE Structured Data Manager PeopleSoft Modules Installation and Deployment Guide*

  Explains how to install the PeopleSoft integration kit.

- *HPE Structured Data Manager Oracle E-Business Suite Modules Installation and Deployment Guide*

  Explains how to install the Oracle E-Business Suite integration kit.

Chapter 1: A structured data management overview

HPE Structured Data Manager provides:

- Powerful tools to design, deploy, and run structured data management solutions that copy or move inactive data out of your production databases and into less expensive storage and, when necessary, restore it back to your active database or upload it to another database.
- Flexible access methods that enable you to query the data.

This chapter includes:

HPE Structured Data Manager overview

Typically, you use HPE Structured Data Manager for one of these reasons:

- To preserve database data for purposes of corporate governance and electronic discovery
- To remove older and inactive data from a production database and archive it
- To make database data available for searching or query

Preservation of data

As organizations review their current portfolio of applications, they realize that a significant percentage of applications are used infrequently. In many cases, these older applications are kept online purely for data access purposes in case of an emergency, such as a lawsuit or an audit of some kind. Corporate governance rules may require you to maintain the data in some accessible form.

Placing the data from the retiring application into a structured file is an excellent way to preserve the data in a usable form while getting rid of the obsolete application and database. Once the data is archived in this way, the application can be retired and the resources re-purposed for newer applications.

While archived data may be accessed infrequently, it must remain accessible for critical situations, such as legal hold and eDiscovery. Depending on the needs of your users, you might make your archived data accessible in a variety of ways:

- **Access XML data through the archive query server.** The archive query server provides application-independent access to data archived in files. This type of access is best for data that you archived for long term retention where the original application has been retired or is otherwise unavailable.
- **Maintain transparent access to the data.** By storing archived data in a different database location, you can maintain transparent access to it. In this case, it appears to the user of the application as though the data were still in the active database even though it resides in a different location. This approach provides maximum ease of access for your users and is best employed in scenarios where access to the data must be immediate and indistinguishable from active data.

- **Reload data from the archive data store to the active database.** If it becomes necessary, you can always reload data from the archive data store to the active database. Once the data is back in the active database, your users can access it and perform further transactions upon it using the original application.

- **Upload data from the archive data store to another database.** Instead of reloading to the active database, you may need to upload data from the archive data store into a completely different database. This approach is particularly useful in dealing with heterogeneous databases. For example, you may have archived from a SQL Server database but need to restore to an Oracle database.

Ongoing archiving of inactive data

Databases that have been in service for long periods of time, particularly those running large applications such as PeopleSoft or Oracle E-Business Suite, accumulate large amounts of data. Some portion of this data is typically inactive (infrequently accessed) as shown in Figure 1   Active and inactive data in the database.

Figure 1   Active and inactive data in the database
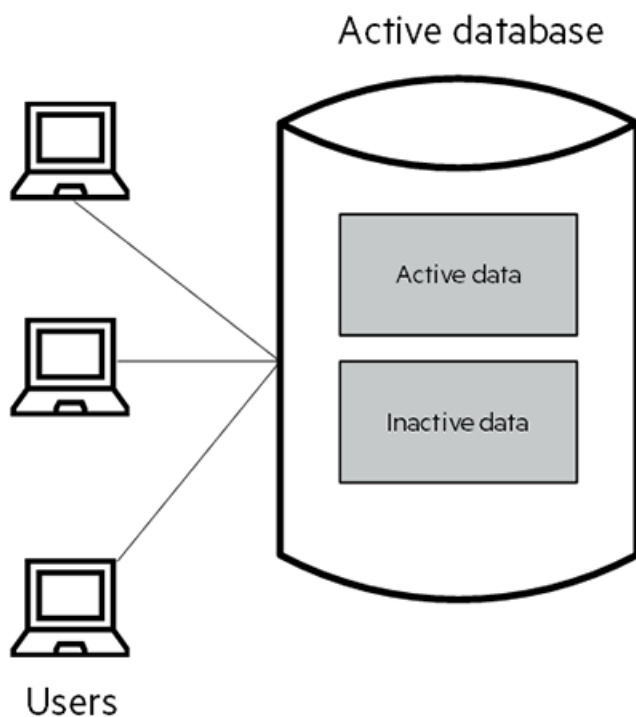
An active database, such as an online transaction processing (OLTP) database, is typically not the most cost effective solution for inactive data. Older, less frequently accessed data can be more efficiently stored in less expensive storage outside of the active database. In addition to not being cost effective, allowing very old records to accumulate in your active database slows performance and leads to the need for additional, expensive hardware.

A good solution to this problem is moving the inactive data out of your active database and into an archive data store, while maintaining application integrity. Thus, you balance the need for of long term records retention with optimal database performance.

Figure 2   Inactive data moved to archive data store
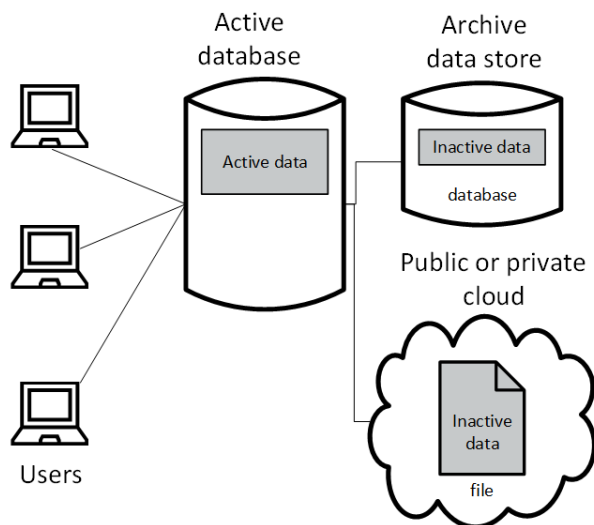
Critical to this is approach is the ability to preserve application integrity while moving data out of the active database. HPE Structured Data Manager includes an understanding of application models, which enables you to archive data while keeping your original application running.

Apart from cost reduction, two of the most common reasons for archiving the data from the active database are:

- **Data retention:** An organization needs to retain the data for long term while keeping it accessible for the purposes of corporate governance, ediscovery, and/or legal hold. Database archiving enables you to store data from your active database with its application structure intact in a structured data file, such as XML, comma separated values (CSV), or JavaScript Object Notation (JSON). Since such formats are open standards, the data stored in such files can be accessed even after the original application and/or its designers are gone. At the same time, a structured file represents a cost effective solution for the long term retention of data.

- **Performance improvement:** Archiving older, infrequently accessed data from the active database enables you to move eligible data to cost-effective storage while also reorganizing the active database and reduces footprint of your active data in production database. This reduction in footprint tends to improve the performance of the active database.

Indexing of databases for search

You can use HPE Structured Data Manager to index your database records for HPE IDOL. Once so indexed, you can search your structured data along with your unstructured data using HPE IDOL.

Analyze your requirements

Before building an archive, you must consider your requirements and plan your archiving solution accordingly. You should consider all of the following requirements:

- **Selective data versus all data archival.** You might archive structured data for a variety of reasons, such as corporate governance or system performance. Depending upon your goals for archiving, you may choose different approaches:

  - If your goal is to selectively copy or archive data on an ongoing basis from an active database, then you should consider a **model-based** approach. Model-based archiving enables you to select and move particular records using specific eligibility criteria, while the active database continues operations. For additional, model-based requirements analysis, see Model-based archiving requirements analysis.

  - If your goal is to retire an entire database or set of tables within a database, a **schema-based** approach is probably more suitable. Schema-based archiving enables you to easily select entire tables to copy or archive.

- **Multiple tiered solution.** Sometimes, the best way to balance the requirements of long term retention, active database performance, and user access to archived data is to employ a multi-tiered approach.

  In a multi-tiered configuration, you can store some archived data in another database (second tier) while the rest is stored in files (third tier). The second tier provides ready access while still improving the performance of the active database. As the access requirements of the data on the second tier lessen over time, that data can be moved to the third tier.

- **Data access - during and after archival.** One of the key considerations for all archiving decisions is user access to data. Once the data is archived, users may still need to access it, albeit less frequently than active data. You should understand their expectations and needs for accessing the archived data. For example, you need to know if users expect to access the data with their existing programs or if they are prepared to use other tools to get at the data once it is archived.

- **Reload/Upload requirements.** When you move archived data back to an active database, to the same database, from which it was archived (reload) or to a completely different database (upload).

  You must understand your reload or upload requirements when building your archive because how and where you plan to reload or upload can affect how you design your archive solution.

- **Disposition and disposal of archive files.** If you store archived data in files, you need to consider where to locate those files and how to manage them over time. You might choose to migrate the files to tape when they reach a certain age. When the archived data is no longer legally required, you will want to dispose of the files altogether.

- **Disposition and disposal of archive files.** If you store archived data in files, you need to consider where to locate those files and how to manage them over time. You might choose to migrate the files to tape when they reach a certain age. When the archived data is no longer legally required, you will want to dispose of the files altogether.

Structured records management

HPE Content Manager is a content management system, which can be an extremely powerful tool for managing your archived data files. Using HPE's structured records management solution (HPE SRMS), you can automatically push archive files from HPE Structured Data Manager into HPE Content Manager as records. Once in HPE Content Manager, you can govern these records according to your defined business and legal policies for management and retention. Contact your HPE sales representative for more information about HPE SRMS.

- **Existing data and programs.** You should be aware of which programs your audience typically uses to access the data.
- **How existing data might need to be transformed.** If you are archiving data that requires transformation, you need to factor that into your archiving solution as well.

Model-based archiving requirements analysis

For model-based archiving, you must gather and analyze information about the data model and business rules. You need a good understanding of all of the following before proceeding to build your model-based archiving solution:

- **Description of the target users.** You must consider the users who access the data that you are planning to archive, and how they typically access and use it.

  Ensure that you consider all of the potential user types and situations. For example, a user searching data on legal hold may have different access requirements from a day-to-day business user of the data.

- **The entities with which users work.** You must understand how your audience looks at the data. For example, if dealing with an order processing system, then the main entity is an order. Your archiving solution should be designed to reflect these business entities.

- **How the business entities are stored in tables.** Before you begin modelling your archive in Designer, you must understand which tables are required for the business entities involved and how those tables are related to one another. To return to the example of an order processing system, you would need to know where the various aspects of a customer order are stored in database tables and how those tables are related to one another (for example, what, if any, foreign keys are used to relate tables to one another). You may find it useful to create an entity relationship diagram (ERD), if you do not already have one.

- **The business conditions and exceptions that must be translated into rules on model tables.** To implement an effective archiving solution, you must understand the business rules and implement them as rules on your data model tables. For example, if the legal department requires that orders remain in the active database for at least two years before becoming eligible for archiving, you need to create a rule in your model that reflects that legal requirement.

- **Data dependencies.** To design an effective model, you must know which business processes rely on the data for continued operation and which other applications can use the data. You must also understand the business rules that govern the closing and reactivating of transactions within the business process.

- **The database types in your environment.** You need to consider the database types (Oracle, SQL Server, Sybase, DB2, or JDBC data sources) with which you are working and whether you need to

construct a model that can run against multiple database types. For example, if you want your project to archive from both an Oracle and a SQL Server database, you may need to have different versions of your rules for each database type.

Chapter 2: Archive to file or database

To support different archive and reload/upload scenarios, HPE Structured Data Manager provides two major types of archive data store, file and database. You can use these two types of archive data store separately or in combination to formulate your archive solution.

This chapter includes:

- Database to file
- Database to database
- Archive solution types

Database to file

Database to file moves (copies and purges) or copies data from an active database to an archive file store (XML, CSV, or JSON). The file storage location can be:

- a file system
- Amazon-S3 (Amazon Simple Storage Service)
- Archive Query Server
- EMC Centera
- Digital Safe
- HDFS (Hadoop File System)
- Another HPE Structured Data Manager instance (SDM Gateway)
- SSH location
- a content management system, such as HPE Content Manager

> **NOTE:** You can use HPE SRMS to automatically push archive files from HPE Structured Data Manager into HPE Content Manager as electronic records. Contact your HPE sales representative for more information about HPE SRMS.

When to use

Database to file archiving is most useful for application retirement. You store archived data in a structured file, such as XML CSV, or JSON. The file can be stored and managed in a variety of locations, such as Vertica or Amazon-S3. Application integrity is maintained and that data can be accessed using standard methods instead of relying on an application that may become obsolete.

For example, suppose that the data must be kept for 10 years. If you need to access the data 10 years after it was archived, the tools and applications that originally operated on it might be long since obsolete and unavailable. XML and CSV storage provide self-contained, independently accessible formats that are supported over long spans of time.

Database to database

Database to database moves (copies and purges) data from an active database to an archive database, which is typically located on a less expensive platform. This type of data movement provides transparent access in the sense that users access the data as if it were still in the active database using the same application program and protocols.

When to use

Database to database works best for situations in which you need free space in your active database for performance while still maintaining easy access to the data.

You might choose to move data from your active database to an archive database in the following scenarios:

- **Transparent access.** If users require transparent access to archived data, you must use database to database. Data access for viewing or reporting is available for all transactions through a combination of synonyms and views. Only an archive database location can be linked to the active database in such a way that the data can be transparently accessed as if it was still in the active database.
- **2nd tier in a 3-tiered archive.** In a 3-tiered archive, an archive database is the 2nd tier. Data moves first from the active database to the archive database. As the access requirements of the data in the archive database (2nd tier) lessen over time, that data can be moved to a file (3rd tier). The 2nd tier archive database provides a good solution for data that is older but still has relatively high access requirements.

Archive types

With database to database archiving, you have the option to move eligible data from the active database to a separate location within the same database (single instance archive) or to a completely separate database (distributed archive). The archive type determines the location of the archived data:

Archive type	Location of data
Single instance	- for Oracle and DB2, separate archive (history) tables within the active database - for SQL Server, a separate database located on the same server as the active database
Distributed	- for Oracle and DB2, archive tables within a separate archive database - for SQL Server, a database located on a separate history server

The single instance archive provides the performance improvements that is derived from reducing the amount of data stored in the active tables, but it does not realize the space savings that come from

moving the inactive data out of the active database altogether. Single instance also avoids the overhead of managing additional databases.

With the distributed archive configuration, eligible inactive data is moved to a separate tablespace in the active database and, subsequently, moved to a separate archive database. This configuration provides the performance improvement of single instance as well as the space savings in the active database that come from moving the inactive data completely out of the active database.

Data transparency

Data transparency refers to the ability of users to see archived data as though it were still in the active database. If you implement data transparency, both active and archived data can be accessed through the current user interface. All data relationships necessary to maintain data and application integrity are retained.

Active data includes transactions that are open, experiencing change, currently being updated, or in use. Archive data includes data that is inactive, has met data retention and deletion eligibility requirements, has been deleted from the managed tables, and is now stored permanently in a separate location.

By running the Create Archive Access job to enable data transparency, you can access the archived data through the existing interfaces and reports in your native application. Your data is accessed differently depending on whether you choose the archive only or union view option.

> **NOTE:**
> If you choose to use database to database archiving without transparency, you can still access your archived data by running SQL queries against the archive database. You may need to create synonyms or views against the data in your active database.

Archive only option

This selection allows access to either active or archived data but does not provide a combined view.

| To access	Procedure
Active data	1. Start the application user interface on the active database. 2. Enter the user information as you always have.
Archived data	1. Start the application user interface on the active or archive database. 2. Enter the user information for the Archive Access user. > **NOTE:** > A view is only available for active data. It is excluded from archive data.

Union view option

This selection allows access to a combined view of active and archived data.

To access	Procedure
Active data	1. Start the application user interface on the active database. 2. Enter the user information as you always have.
All data (active and archived)	1. Start the application user interface on the active or archive database. 2. Enter the user information for the Archive Access user.

Chapter 3: Archive solution types

You can use database to file and database to database movement independently of one another or in combination. If you use database to file and database to database independently, it is called a two-tier solution. If you use them together, it is called a three-tier solution.

Figure 3   Archive solution types
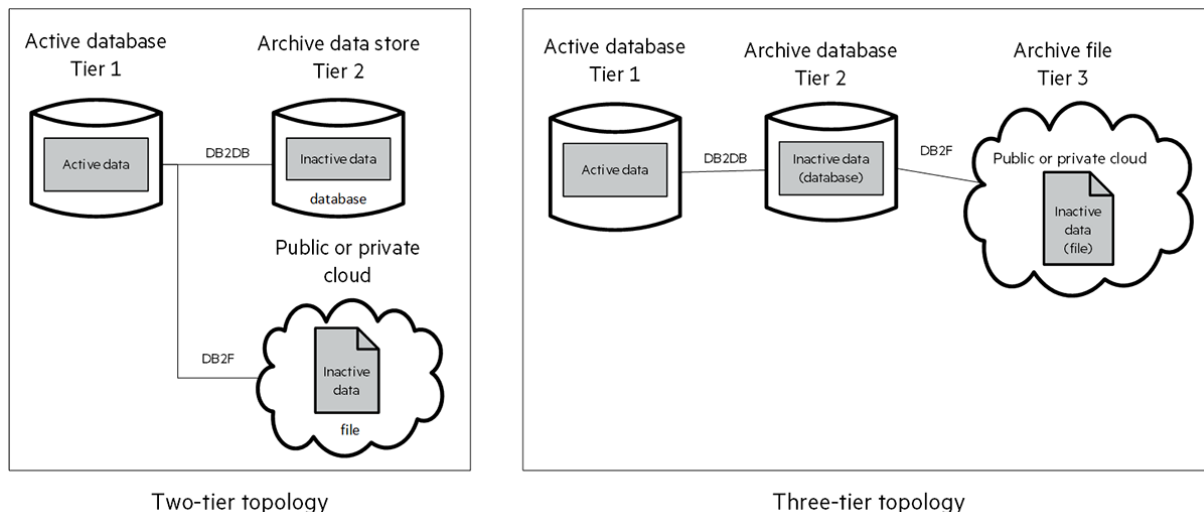
A two-tier solution provides a basic approach to database archive. It moves or copies inactive data from a production database to either an another less expensive database or a file (XML, CSV, JSON, etc.) - an archive data store. A two-tier solution might employ database to database or database to file archiving, depending on your goals for the archive.

For example, if your primary objective is transparent access, you would use database to database archiving. If your objective was long term retention, you would instead implement database to file archiving.

A three-tier solution provides the most comprehensive approach to database archiving. In a three-tier topology, your archive is layered into three tiers.

- As data in the active database becomes less frequently used and meets certain business and legal conditions (for example older than 3 years),it moves from production database (tier one) to a less expensive archive database (tier two) using database to database archiving.

- When the data on your second tier is ready to be archived for long term retention, you use database to file archive to move it from the archive database to XML, CSV, or JSON files (tier three). This file constitutes your archive's third tier and represents your long term retention solution.

Chapter 4: Data selection and movement

When moving data, an algorithm must first select the data eligible for movement. Second, the selected data is actually moved using any one of a variety of methods. Based on your requirements, you choose the optimal selection and movement methods.

This chapter includes:

- Data selection and movement options
- Standard selection
- Advanced selection
- Transactional movement

Data selection and movement options

HPE Structured Data Manager offers different options for performing the selection and movement of data from one tier to another in your archive. The available methods will vary somewhat depending upon the type of the archive cartridge.

- For selection:
  - **Standard selection**
  - **Advanced selection**
- For data movement:
  - **Transactional**

Standard selection

Standard selection is a method of data selection that treats the model as a tree with the root node being the driving table. Starting at the top of the tree, standard selection populates one selection table for each node in the model. Standard selection does not support the case where a child has multiple parents. Therefore, multiple uses of the same table can break the tree model, if each use of the table does not map to a disjoint set of rows.

- **Driving table.** Most of the selection logic is evaluated in a single query during the population of the driving selection table. This query evaluates all rules that do not have eligibility analytics enabled. The rules on child or grandchild table are evaluated by including them in a subquery to the child table.

- **Child tables.** After the driving table rows are populated, each child table is populated in top-down order.

  For example, suppose that ORDER_HEADER is the driving table and its selection table is already populated. If the table hierarchy were ORDER_HEADER, ORDER_LINE, and ORDER_LINE_DIST, the ORDER_LINE selection table (ORDER_LINE_SEL) would be populated next and then the ORDER_ LINE_DIST selection table (ORDER_LINE_DIST_SEL).

  The child table is populated by a join between the parent table, the parent selection table, and the base table, as well as any conditional relationships. For example, ORDER_LINE_SEL is populated by joining

the driving table (ORDER_HEADER), the driving table's selection table (ORDER_HEADER_SEL), and the active child table (ORDER_LINE).

> **NOTE:**
> No special handling of null foreign keys is performed. Therefore, if a child has a null value for its foreign key then it is inferred that it is not related to the parent.

- **Conditional relationships.** A conditional relationship on a virtual foreign key adds an additional condition on the relationship between two tables. For example, if the model defines a relationship of ORDERS.ORDID = ITEMS.ORDID, it is possible to add a constraint or filter on the relationship:

 ITEMS.PARENT_ITEM_ID is null

  Any rows which do not pass the original relationship and then the added conditional relationship are not children of the parent table.

- **Multiple table uses.** Creating additional instances of a table in a model indicates that the table is used to store different types of data. Each such instance represents a separate, logical table with its own selection table (one for each node in the model). The selection tables are populated in the order in which they appear in the model.

  With multiple table uses, you must ensure the rows do not overlap, that is they must be disjoint. Otherwise, you will encounter unique constraint errors when moving the data. You can avoid row overlap by creating conditional relationships.

  For example (Figure 5 Multiple table use in a model), suppose that you have an ORDER_ATTACHMENT table that contains attachments for both ORDER_HEADER and ORDER_LINE. The ATTTYPE column contains OL or OH to indicate whether the row pertains to ORDER_HEADER or ORDER_LINE.

Figure 5   Multiple table use in a model
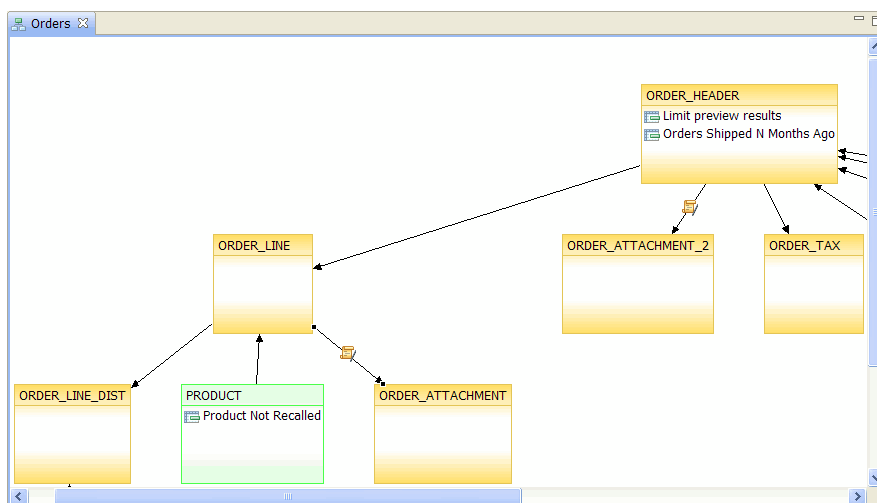
To avoid overlapping rows in this case, you could add two conditional relationships. The relationship between ORDER_LINE and ORDER_ATTACHMENT is:

${FK_ALIAS}.ATTTYPE = 'OL'

The relationship between ORDER_HEADER and ORDER_ATTACHMENT_2 is:

${FK_ALIAS}.ATTTYPE = 'OH'

When to use

In most cases, it makes sense to use standard selection rather than advanced selection. The following indicate situations where standard selection should be used:

- **When using a non-Oracle database.** Advanced selection only works with Oracle databases. Hence, if you are using a non-Oracle database, you must use standard selection.
- **When performance is a concern.** Because standard selection does not attempt to discover interrelated rows, it typically runs faster than advanced selection.

Advanced selection

Advanced selection is a method of data selection that discovers all of the interrelated rows from multiple tables and conceptually places them in the same application partition for archiving. Because advanced selection discovers related rows, it is a powerful method for selecting data for archiving.

> **NOTE:**
> Application partitioning is a concept unique to HPE Structured Data Manager. This notion of partitioning contrasts with the more common table partitioning offered by the database management software. A table partition is only defined in one table. It does not contain all related rows from multiple tables. In order to achieve application integrity during and after archiving, identifying application partitions is key and table partitioning is often insufficient in this regard.

In advanced selection, the model is considered as a standard entity-relationship model. You can select multiple driving tables for advanced selection. Furthermore, you need not ensure disjointedness for multiple table uses and there is no restriction on null foreign key values.

- **Driving table.** In advanced selection, you select as many driving tables as necessary to drive selection. You are not restricted to one driving table as in standard selection.
- **Child tables.** In advanced selection, since the model is viewed as an entity-relationship diagram with potentially many driving tables, the model is not a tree. It has no hierarchical predecessors/successors like you see in tree diagrams. Even though the model editor presents as a tree, this visualization does not apply to advanced selection. Once the driving set of rows is identified, advanced selection follows the relationships in the model to spread the eligibility to other tables. In the process of spreading the selections, the application restricting conditions are evaluated, which can keep or drop a selection.
- **Conditional relationship.** For advanced selection, a condition in a relationship is considered as part of the relationship. In entity-relationship modeling, relationships do not influence the definitions of entities.

  For example, adding a condition on the relationship between ORDER_HEADER and ORDER_LINE does not influence which rows are considered part of the ORDER_LINE (or ORDER_HEADER) entity. The simple rule of thumb for designing for advanced selection is to restrict the entity and then add the rule to the entity, rather than the relationship.
- **Multiple table use.** Multiple table use is the key to treating a tree model as an entity relationship model. If a table appears twice in the model, advanced selection can find loops in rows that start

from one table, follow relationships in the model to other tables, and eventually return to the starting table. For example, suppose the table ORDERS joins to two tables, NOTES and ITEMS. Now further suppose that we start from a row in ORDERS and that row joins to a row in the ITEMS table. The row in ITEMS in turn joins to a row in NOTES_2, which is a second use of the table NOTES. The row in NOTES_2 is also a row in NOTES, which leads back to the ORDERS table, where the traversal originated.

If, upon returning to ORDERS, the rows have already been found eligible, nothing changes. However, if new rows are discovered upon the return to ORDERS, these rows are marked eligible for archive and the looping from ORDERS to ITEMS to NOTES_2/NOTES and back to ORDERS must continue. This kind of data looping is sometimes referred to as chaining.

When to use

- **When application integrity is complex.** If you have a strong requirement to maintain application integrity and the data relationships are more complex, advanced selection is a good option. The application can actually run correctly using the archived data from advanced selection and, after data removal, the application is still able to run correctly. Since advanced selection identifies application partitions which encompass all the necessary data for the application, it ensures application integrity.

  > **NOTE:**
  > - **When using an Oracle database.** Advanced selection requires an Oracle database. You cannot use it with any other database.
  >
  > - **When performance is not a special concern.** Because advanced selection attempts to discover interrelated rows, it typically runs somewhat slower than standard selection.

Eligibility analytics

If you choose to turn them on, eligibility analytics show you which rows have been selected for movement and which have been excluded. For excluded rows, eligibility analytics will also show you which of your rules caused the exclusion.

Eligibility analytics provide a powerful decision-making tool for database archiving. You can review the analytics prior to data movement in order to determine whether you want to proceed or abort the data movement. For example, you might review the analytics to determine if too many or too few rows are selected for movement. If your review indicates that too many or too few rows are selected for movement, you can abort the operation. Otherwise, you can continue it.

> **TIP:** Enabling eligibility analytics can significantly affect the performance of data selection, depending upon the size of your data set. You should consider the possible performance trade-off before enabling eligibility analytics and only enable them when you expect to make use of them.

> **NOTE:** To perform eligibility analytics on a rule, a data movement key is required on the table where the rule is specified.

Transactional movement

Transactional movement is the default method for both database to database and database to file, but it behaves differently depending upon whether you are moving to a database or file.

- Transactional movement to file
- Transactional movement to database

Transactional movement to file

For database to file, the only available movement method is transactional movement. You can choose to only copy, or copy and delete (archive) the data when you go from database to file.

When to use

- **For snapshotting data.** You should use the copy method for database to file when you are creating a snapshot of the data. For example, you might need a snapshot for legal reasons or for creating a test environment.

Transactional movement to database

For database to database, transactional movement uses set-based data movement.

When to use

Transactional data movement is best employed under the following circumstances:

- **When you cannot afford a production outage.** Transactional data movement does not require you to take the application offline. Hence, users can continue to access data while the archiving jobs are running.

As shown in Figure 6 Transactional movement for database to database, transactional data movement between databases operates as follows:

- One set of rows is processed and moved, then after a commit, the next set of rows is processed and moved.

- The archive process occurs while the application remains online and accessible to users.

Figure 6   Transactional movement for database to database
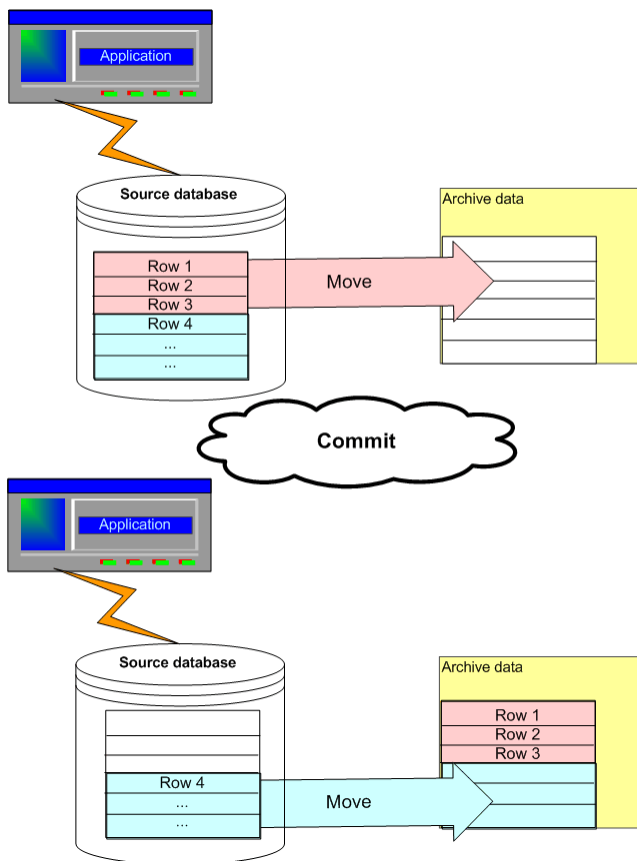
Transactional movement between databases can be parallelized in one of two ways to enhance performance:

● Table parallelism

● Fully transactional

Table parallelism typically provides faster performance but requires some downtime for the archive database. Fully transactional is typically slower but users can continue to access the archive access schema during archiving.

Chapter 5: Models, parameters, cartridges, and business flows

In Designer, you use models, parameters, cartridges, and business flows to specify:

- which data to move
- where to move the data
- how to move the data
- what, if any, additional logic to apply

This chapter includes:

- Lifecycle
- Data modeling
- Parameters
- Cartridges and business flows
- Groovy scripts

Lifecycle

Implementing an archive solution is an iterative process consisting of multiple, cyclical phases. It requires an understanding of the semantics of the parts that are moved between the active database and the archive data store:

1. Before you begin, analyze the requirements of your users and their environment. See Analyze your requirements.

2. Capture metadata about the tables (tables, synonyms, views) to be archived/reloaded.

3. Create a schema-based cartridge to archive all the tables in a database, or design a model representing the list of tables to be archived or reloaded with the relationships between these tables. For a model-based cartridge:

    a. Create rules for the model tables to refine which records are eligible for archiving or reloading. See Rules.

    b. Test the model and rules by previewing their execution and navigating the results to understand what data would be moved under this model.

4. Create an archive or reload cartridge, which encapsulates a particular usage of the model you just created. The same model can be reused in multiple cartridges. See Cartridges.

    > **TIP:** For database to database cartridges, it is a good practice to create a corresponding reload cartridge for each archiving cartridge. The reload cartridge enables you to restore selected data from the archive data store to the active database.

5. Test the cartridge by previewing its execution and navigating the results to understand what data would be archived or reloaded under this cartridge.

6. Optionally, create a business flow that specifies the sequence and dependencies between one or many cartridges and other activities required to accomplish an archive solution. See Business flows.

> **TIP:** For each of your archive business flows, it is always good practice to create a corresponding undo business flow. The undo business flow enables you to quickly reverse a previous run of an archive business flow in case you discover a problem.

7. Determine where the archived data will reside. You need to decide the archive data store's type (database or file) and ensure that it has sufficient space to hold the archived data.

8. Deploy the business flows to the server where the active database resides.

> **NOTE:** If you deploy a cartridge itself, HPE Structured Data Manager wraps it with a business flow upon deployment. Therefore, at runtime, you always run a business flow, even if you deployed a cartridge.

9. Ensure that the latest version of the business flow you need to run has been deployed.

10. Ensure that the configuration parameters are set to the appropriate values. For example, if you are archiving your data to CSV, and you want the CSV compressed and zipped, you could set the Compression Algorithm parameter for the business flow to GZIP.

11. Launch the job from the Web Console or the command line.

   - Select the desired job, set the runtime parameters to the appropriate values, and launch the job.
   - From the command line, review the syntax, ensure you have the required parameter values and environment information, and launch the job.

   > **TIP:** Ensure that you have enough disk space available before running jobs.

12. Monitor the progress of the job.

13. Repeat this procedure as many times as necessary. For example, when the active database changes or new business rules come into play, you typically need to update your project, redeploy your revised business flows, and run them.

In HPE Structured Data Manager, the Designer is the main component where an archive solution is built. You design, test, and deploy your archive solutions in Designer (step 1 through step 7 in the preceding procedure). You run the archive solution typically by launching a job from the Web Console or the command line (step 9 in the preceding procedure).

See also

HPE Structured Data Manager Developer's Guide

HPE Structured Data Manager Runtime Guide

HPE Structured Data Manager Troubleshooting Guide

Data modeling

Depending on your analysis of requirements and understanding of your database, you must first decide on the basis for choosing data for your archive:

- **Schema-based**. If you choose the schema-based approach, you simply create cartridges and choose tables or views for archiving. You can choose to archive all data, no data (table structure only), or selected data using a WHERE clause. Schema-based archiving is useful when retiring an entire database or cleaning up orphaned tables from earlier archive operations.

  The primary reason for using schema-based cartridges is to archive all of the data from a database that you plan to retire. If you are archiving all of the data, you do not need a model because you are not selectively choosing records for archiving.

- **Model-based**. In the model-based approach, you design a data model. This data model governs the selection of data for inclusion in the archive. Unlike the schema-based approach, model-based archiving chooses data from the specified tables based upon the relationships and rules of the model. Model-based archiving is best when you want to choose data at a more granular level than schema-based archiving.
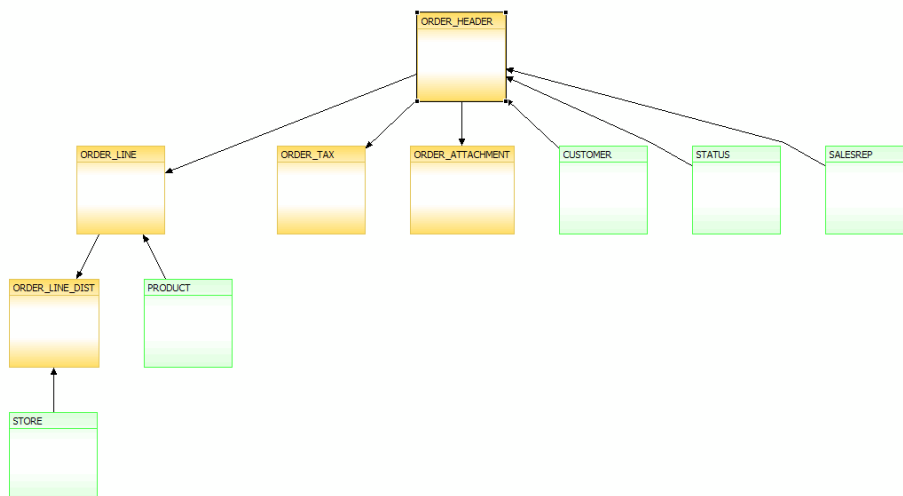
Models

In HPE Structured Data Manager, a model, like the one in Figure 7   Sample model, can be used to identify which data to include in an archive or reload business flow. The model graphically represents the tables (including views and synonyms) and how they are all linked through relationships.

Tables and table uses

In models, we use the term tables for the sake of simplicity, but it is important to understand that tables (including views and synonyms) mean table uses or instances and do not necessarily have a one-to-one correspondence to the actual tables stored in the database. That is, you might have a model where the same table is used and appears multiple times. This concept of multiple table uses is described more completely in Multiple table use.

Figure 7   Sample model

A model can include tables of the following types:

- Driving table

  The driving table is the root of the model hierarchy. Its relationship to the child tables drives the selection of transactions. Typically, a model only has one driving table, but, when using advanced selection, you can specify multiple driving tables. If you use advanced selection, you can mark any table as a driving table. When a table is selected as a driving table, its selections are then propagated to other tables, thus impacting their row selections.

  In Figure 7 Sample model, ORDER_HEADER is a driving table. The driving table can have zero to many children. The children may be transactional, lookup, or chaining tables. In Designer, child tables are related to the parent by foreign keys (database or virtual), optionally with a conditional relationship. A unique key (database or virtual) is the referencing constraint for the foreign key.

- Lookup table

  A lookup table contains helpful non-transaction information. You might need these lookup values present for the purposes of a rule or for the sake of making an archive file more complete. For example, nontransaction information could be status definitions, or the names of the sales representatives. In Figure 7 Sample model, STORE, PRODUCT, CUSTOMER, STATUS, and SALESREP are all lookup tables.

  > **TIP:** Since lookup tables contain non-transactional, reference information, you can choose to copy the data to the archive database, or keep it in the source database and reference it with a database link.
  >
  > In either case, you typically do not purge lookup tables from the source database because they may be referenced by both archived and active tables.

- Transactional table

  A transactional table contains information about the business transaction. For example, a transactional table might contain detailed tax or payment information related to each business transaction. In Figure 7 Sample model, ORDER_LINE, ORDER_LINE_DIST, ORDER_TAX, and ORDER_ATTACHMENT are all transactional tables. Transactional tables are always part of the archive. For this reason, they are also known as managed tables.

- Chaining table

  If a many to one (or many to many) relationship exists between a higher level table and a lower level table, the lower level table is labeled as a chaining table.

- Relationship

  All tables in a model are linked through relationships. These relationships are derived from captured foreign keys or defined by you at design time.

- Unique constraints

  A unique constraint is a column or a list of columns that enables you to identify rows in the database. For example, in a table of employees, employee numbers or Social Security Numbers are often unique constraints. Database unique constraints are defined in the database. If you have database unique constraints, HPE Structured Data Manager discovers them. Thus, you can make use of these database constraints in Designer. If you do not have such database constraints, you can define your own, known as virtual unique constraints, in Designer. Virtual unique constraints only apply within HPE Structured Data Manager. They are not added to your database and are only available for use inside of Designer. You must ensure yourself that virtual unique constraints are

correctly defined. Furthermore, virtual constraints should be indexed, otherwise you may encounter significant performance issues.

> **TIP:** Unique keys perform the same function as primary keys and can be defined for any table in the model. Note, though, that unique keys are less strict than primary keys.
>
> For example, in a table, you could define a primary key on column A and a unique key on column B. In that case, column A is mandatory, but column B is not unless the column is defined as not null. The unique key enforces the rule that you cannot enter duplicate values for column B, but null values are not considered duplicates.

- Foreign key constraints

  A foreign key constraint is a column or a list of columns that enables you to relate rows in the database. For example, in a table of sales orders, employee number might be a foreign key that enables you to relate to the employee table. Database foreign key constraints are defined in the database. If you have database foreign key constraints, HPE Structured Data Manager discovers them. Thus, you can make use of these database constraints in Designer. If you do not have such database constraints, you can define your own, known as virtual foreign key constraints, in Designer. Virtual foreign key constraints only apply within HPE Structured Data Manager. They are not added to your database and are only available for use inside of Designer. You must ensure yourself that virtual foreign key constraints are correctly defined. Furthermore, virtual constraints should be indexed, otherwise you may encounter significant performance issues.

Multiple table use

In some cases, you may find it necessary to reference the same table more than once in your model. Each use of the table may have different rules associated with it.

In most situations, such multiple table use poses no problems. In some cases, though, you need to proceed with caution:

- If you have multiple table uses in your model that do not correspond to disjoint (non-overlapping) sets of rows in the table, you must use advanced selection. If the uses are not disjoint, standard selection can lead to duplicate data in your archive data store. Hence, you can only use standard selection when the multiple uses are disjoint from each other.

- For advanced selection, you can use the same table in a model multiple times and each use can be of a different type (transactional, lookup, or chaining).

Rules

In a model, rules define the extent of the archiving operation or further refine the model by excluding certain records. Rules typically fall into two broad categories:

- A rule can articulate a business condition, often parameterized, to determine the rows eligible for archiving or reloading data. This type of rule defines the scope of the data to be archived. For example, if you have ten years of data, and define a policy requiring that four years be retained in your production environment, six years of data is considered for archiving, and those six years are displayed when you Preview the data.

- A rule can be a requirement or restrictions for an application or business entity that ensures application data integrity. For example, a rule might specify that an order's status be closed before it can be archived. Therefore, even if the order were otherwise eligible for archiving, it would be excluded by this rule if its status were anything other than closed.

You can create rules on any model table and incorporate parameters into them.

Parameters

You can set up parameters that provide values within your project. In HPE Structured Data Manager, you have three types of parameters:

- **Runtime parameters** have their values set at runtime by the user running the job. Runtime parameters tend to be best for operational values that tend to change with each execution of a job. For example, if your archive is based on a specified cutoff date, you most likely need to update that date every time you run the job.
- **Configuration parameters** have their values set by an administrator (someone who has repository privileges from the Web Console) through the administrator interface. Typically, this type of parameter represents values that should be changed very infrequently, perhaps only at deployment time. A retention period defined by the legal department is a good example of a value that should not change very often and that only administrators would be allowed to change.
- **Dynamic parameters** have their values computed by every run of a Groovy script that executes at runtime. For example, this type of parameter can supply the type or version of a database or application, which can be obtained programmatically at runtime.

Regardless of a parameter's type, it can be referenced from several places within your archive project:

- WHERE clauses within your rules
- Groovy scripts in runtime parameter validations
- Groovy scripts in business flows
- SQL for runtime parameter list of values
- Interrupts in your business flows
- Custom selection programs in your cartridges

Cartridges and business flows

Once you complete and test your model and rules, you create a cartridge to apply them (database to file or database to database). You can then incorporate the cartridge into a business flow that runs it, as well as other cartridges or logic you wish to implement.

- Cartridges
- Business flows

Cartridges

Cartridges capture the application and business rules to ensure referential integrity of the data. Cartridges allow inactive data to be segregated from active data based on the defined data retention rules.

After you have your model and rules defined, you must consider how to deploy that archive. A cartridge defines a specific set of characteristics to apply when deploying a model with rules.

There are several types of cartridges:

- **Archive database to database cartridges** archive data, based upon a model, from your active database to another database, typically on a less expensive platform with lower performance characteristics.
- **Reload database to database cartridges** reload data, based upon a model, from your archive database back into your active database.
- **Archive database to file cartridges** archive data, based upon a model, from a database (active or archive) to XML, CSV, or JSON files, typically for the purpose of long term retention.
- **Schema-based database to database cartridges** archive selected tables from your active database to an archive database without use of a model.
- **Schema-based database to file cartridges** archive selected tables from an active database to file without use of a model.
- **Data access cartridges** query the archived data in order to provide access for business users.

A cartridge specifies a number of characteristics. Some of the more common ones are:

- Managed tables to be archived. In the case of model-based archives, managed tables typically see the driving table and the transactional and chaining tables in the model.
- Lookup tables to be copied (but not purged) by the cartridge.
- Which rules to apply for scoping and restricting the data to be archived.
- How to handle transactional and chaining tables (copy, delete, or both).
- Whether to copy lookup tables (for database to file only).
- Columns to include or exclude (for database to file only).
- Whether to apply data masking to a column and, if so, what type of masking.
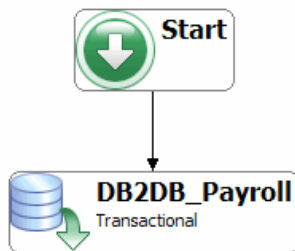
Business flows

You can deploy a cartridge by itself or as part of a business flow that performs some additional processing. A business flow is a series of activities, such as archive operations and scripts, that run in a sequence. Business flows enable you to split your cartridge into multiple activities and insert Groovy scripts to perform additional processing in between those activities. You can create a business flow to run one or more cartridges and perform any additional operations you require. You construct business flows in the Designer with drag and drop gestures.

> **TIP:** You can even create a business flow that includes no cartridges at all. Such a business flow might only include Groovy scripts.

For example, suppose that you have two models, one for payroll and another for balances. For each model, you create two cartridges, one for database to database archiving and another for database to file archiving. At the end of each month, you only want to run the database to database payroll cartridge. At the end of each quarter, you want to run all of the cartridges. Figure 8 End of month business flow example illustrates an example of an end of month business flow that runs a single cartridge (database to database) based on a single model (payroll).

Figure 8 End of month business flow example

For your end of quarter business flow, you could run multiple cartridges in one business flow, as shown in Figure 9 End of quarter business flow example.

Figure 9 End of quarter business flow example
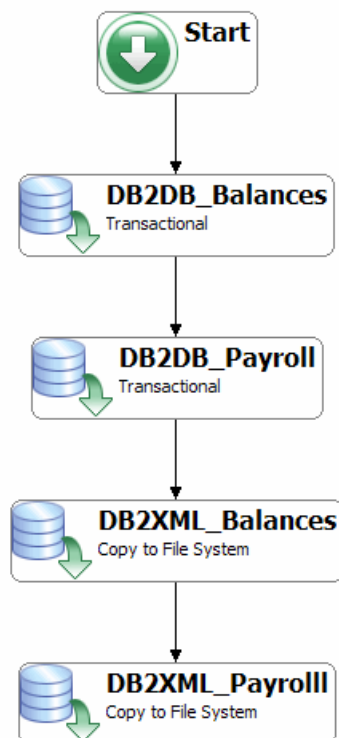
A business flow can also include activities other than cartridges that are necessary to accomplish an entire archive solution. Such activities might include:

- Operating system commands, such as copying a file, deleting a file, compressing a file, or transferring a file (using FTP).

- Generating an email message or other notification.
- Accomplishing a database operation, such as dropping indexes, making a tablespace read-only, or analyzing statistics for a table.

Groovy scripts

HPE Structured Data Manager enables you to include Groovy code to extend your archive solution for the following purposes:

- Groovy scripts can be inserted in business flows to perform additional processing.
- Interrupts and conditions in a business flow use Groovy code to define their branching criteria.
- Parameter validations are performed in Groovy code.
- Dynamic parameters are assigned their values by Groovy code.
- Groovy scripts can be set to execute before and after business flow deployment.
- Groovy scripts can be set to run before and after running a business flow.

The Groovy code in these places is processed as follows:

- The Groovy in dynamic parameters executes at runtime.
- The Groovy in validation and dynamic parameters runs just before execution (during job initialization).
- The Groovy in interrupts, conditions, and business flow scripts runs when the business flows are executing.

See also:

HPE Structured Data Manager Developer's Guide

Chapter 6: Repository, deployment environments, and Web Console

After your cartridge or business flow is complete and tested, you are ready to try running it on the actual active database, outside of Designer. In order to run against your active database, you must first deploy the business flow or cartridge in the deployment environment.

- Repository
- Deployment environments
- Web Console
- Deployment options

Repository

The HPE Structured Data Manager repository contains the metadata required to perform and monitor operations on the active and archive databases. The repository is stored in a database of your choice (any database, including the active or archive database, installed with HPE Structured Data Manager).

In addition to the repository, each active and archive database upon which HPE Structured Data Manager will perform operations may contain interface schema, which contains some additional metadata used by HPE Structured Data Manager.

Figure 10 Repository, interface, and BUILDAA illustrates the relationship between repository and interface schema. For the sake of clarity, the diagram shows each of these schema on separate databases, but, depending on your configuration, they could reside on the same database. For example, the repository could reside on the active database and the archive database and the active database could be on the same database.

> **TIP:** Non-intrusive environments do not include interface schema. For more information, see Non-intrusive environments, on page 38.

Figure 10   Repository, interface, and BUILDAA
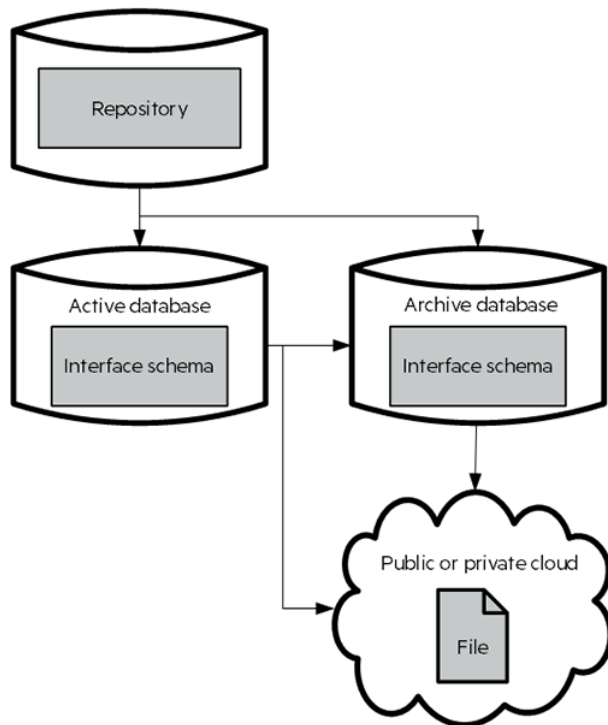
Deployment environments

An environment is a named deployment environment associated with a source (active) database. If you plan to perform database to database operations for the source database, the environment will also include a target database/location. From an environment, users with the necessary privileges can run business flows against the source database. You can deploy as many environments as you wish, thus supporting multiple source and target databases from a single installation of HPE Structured Data Manager.

Figure 11 Single repository with multiple environments illustrates how a single installation of HPE Structured Data Manager can service several database environments, each with its own defined characteristics. Depending upon their assigned privileges, various users may access one or more of these environments to perform actions such as deploying business flows, running jobs, or administering the system.

Figure 11   Single repository with multiple environments
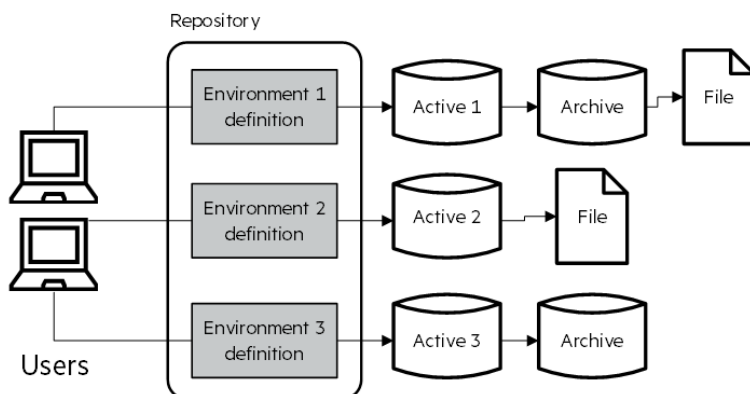
Non-intrusive environments

When you create a standard deployment environment, HPE Structured Data Manager creates interface schema in the source and target databases. These interface schema store metadata that enables HPE Structured Data Manager to function more efficiently. In some cases, though, you may be unable or unwilling to install such schema into your databases. For example, suppose that your source database is an older, read-only system that does not support basic SQL statements, such as DELETE. Even if the source database can be updated, you might simply prefer not to create additional schema in your production databases.

For database to file, you can configure a non-intrusive environment that does not require any interface schema in the source database. The data is archived without interface schema by using a generic JDBC driver.

Web Console

The Web Console provides a browser-based interface to the HPE Structured Data Manager repository and your deployment environments. You use the Web Console to:

- Perform the initial setup of the repository.
- Define deployment environments.
- Create and provision users to perform actions in the Web Console.
- Deploy, run, administer, and monitor your business flows.
- Access data.

Figure 12 Web Console and repository illustrates the relationship of the Web Console to the repository and active databases, and the archive data store (database and file). For the sake of clarity, the diagram shows each of the repository, active, and archive databases as separate, but, depending on your configuration, they could reside on the same database. For example, the repository could reside on the active database and the archive database and the active database could be on the same database.

> **NOTE:** An interface schema on the active database, as shown in Figure 12   Web Console and repository is created only when a *standard* environment is used. With *non-intrusive* environments, no interface schema is created on the active database.

Figure 12   Web Console and repository
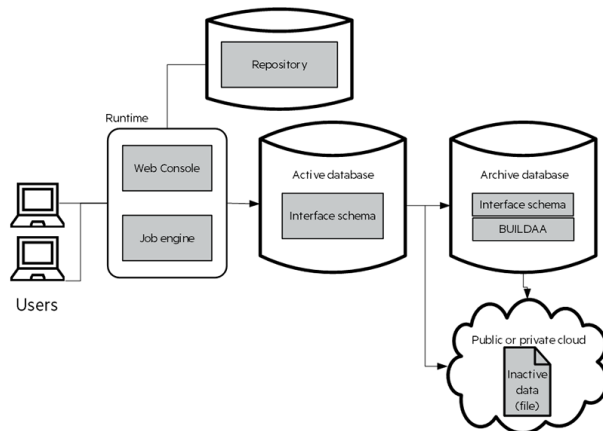
Deployment options

When deploying your cartridge or business flow from Designer, you have three options:

- **Local deployment.** If Designer and runtime (Figure 12   Web Console and repository) are installed on the same machine, you can do local deployment, and generate and deploy your cartridge or business flow on your local client. In this case, HPE Structured Data Manager generates the deployment file and then immediately deploys it in the chosen local environment.

- **Remote deployment.** If Designer and runtime are remote from each other but still accessible across a network, you can deploy from Designer to the remote runtime client. In this case, HPE Structured Data Manager generates the deployment files locally and moves the deployment files to the remote system, where they are deployed in the chosen environment.

- **Generate.** If Designer and runtime are remote from each other and you do not have network access to the remote machine, you must generate the deployment files in Designer without actually deploying them. After the files have been generated locally, you can hand them off to someone with access to the remote system. Once the deployment files are moved to a location accessible to the runtime system, they can be deployed to an environment using the Web Console.

Deployment and runtime history

When you create or modify a cartridge or business flow, you can associate a version with it. Only one version of a cartridge or business flow may be deployed at a time. If you deploy a new version of a cartridge or business flow that was previously deployed, the previous version is uninstalled before the new version is deployed.

HPE Structured Data Manager maintains the runtime history of the cartridge or business flow across multiple deployments, unless you specifically indicate that you want the history dropped at deployment time. In general, it is best practice to maintain the runtime history across deployments. Otherwise,

when the history is dropped, you can no longer view status information for previous executions of the cartridge or business flow.

Chapter 7: Integration kits and customization

Rather than developing your own projects from scratch, you might choose to obtain and customize pre-built projects (integration kits) for particular applications, such as Oracle E-Business Suite and PeopleSoft.

Using Designer, you can safely customize a project you receive from another vendor, such as HPE. When the vendor releases revisions to the project, you can merge your customizations into their revised project. Thus, you need not constantly repeat your changes each time you receive a revision from the vendor.

Integration kits

HPE provides integration kits for two popular enterprise applications, Oracle E-Business Suite and PeopleSoft. These kits provide pre-built HPE Structured Data Manager projects that implement data movement solutions for specific E-Business Suite and PeopleSoft applications. You can modify these projects in Designer to meet the special requirements of your environment, and then deploy and run the business flows.

See also

- *HPE Structured Data Manager PeopleSoft Modules Installation and Deployment Guide*
- HPE Structured Data Manager *Oracle E-Business Suite Modules Installation and Deployment Guide*

Customization

When you receive a project that was pre-built for a packaged application such as PeopleSoft or Oracle E-Business Suite, you typically need to modify the project for your environment. For example, you might need to add some tables to the model that are required by your environment but that are not part of a default implementation of the packaged application..

As a consumer of a customizable archive project, you must be aware of the following:

- The vendor typically provides a project in a locked state, which means that the various files that comprise the project are all locked. Locked files enable you to make supported changes to the project while still preserving your ability to upgrade to newer versions of the integration kit.
- When you make certain changes to the model or cartridge in a locked project, you receive confirmation dialogs with warning messages to ensure that you really want to make the change. Once you make the change, visual cues will indicate where you have made customizations.
- When the vendor releases new versions of the project, merge your customizations into the new version.

See also

HPE Structured Data Manager Developers Guide

Chapter 8: Next steps

Depending upon your role, you may want to take different approaches to learning about HPE Structured Data Manager. The road maps in this section are designed to give you a track to take through the documentation based upon what you want to do with HPE Structured Data Manager.

This chapter includes:

- Documentation road map for custom development
- Documentation road map for deploying and running business flows
- Documentation road map for application integration kits

Documentation road map for custom development

If your role is to build custom, archive solutions from scratch with HPE Structured Data Manager:

1. Read through all of the chapters in this guide to gain an understanding of the main concepts and considerations for archive development.

2. Install the HPE Structured Data Manager according to the instructions in *HPE Structured Data Manager Installation Guide*.

3. Go through *HPE Structured Data Manager Tutorial*. The tutorial enables you to get hands on with the product quickly and exposes you to many of the most commonly used features.

4. Review the advanced tutorials in the *HPE Structured Data Manager Developers Guide* and *HPE Structured Data Manager Runtime Guide*.

5. Plan your archive solution. Use the considerations in Analyze your requirements as a starting point and gather information about your own database environment before you begin.

6. Develop your archive solution.

   - See *HPE Structured Data Manager Developers Guide* and *HPE Structured Data Manager Tutorial* to help you perform the necessary tasks for developing your project.
   - See the *HPE Structured Data Manager Troubleshooting Guide* to help you diagnose problems and resolve common issues.

7. Optionally, design data access for your business users. See *HPE Structured Data Manager Developers Guide*.

8. When your business flows or cartridges are ready, generate and deploy it according to the information in *HPE Structured Data Manager Developers Guide*.

9. Run your deployed business flow.

   - See the *HPE Structured Data Manager Runtime Guide* to help you perform the necessary tasks for running your job from the Web Console or command line.
   - See the *HPE Structured Data Manager Troubleshooting Guide* to help you diagnose problems and resolve common issues.

10. As necessary, repeat step 6 through step 9 to modify your project, re-deploy the business flow, and run the job.

Documentation road map for deploying and running business flows

If your role is to deploy and run business flows, follow this road map:

1. In this guide, read A structured data management overview, Archive to file or database, and Repository, deployment environments, and Web Console.

2. Install the HPE Structured Data Manager according to the instructions in *HPE Structured Data Manager Installation Guide*.

3. Obtain the project that contains the business flows you plan to run.

4. Generate and deploy the business flow according to the information in *HPE Structured Data Manager Developers Guide*.

5. Run a job for your deployed business flow.

   - See the *HPE Structured Data Manager Runtime Guide* to help you perform the necessary tasks for running your business flow from the Web Console or command line.

   - See the *HPE Structured Data Manager Troubleshooting Guide* to help you diagnose problems and resolve common issues.

As necessary, repeat step 3 through step 5.

Documentation road map for application integration kits

If your role is to apply integration kits for packaged applications, such as PeopleSoft or Oracle E-Business Suite, and customize them for your environment, follow this road map:

1. Read through all of the chapters in this guide to gain an understanding of the main concepts and considerations for archive development.

2. Install the HPE Structured Data Manager according to the instructions in *HPE Structured Data Manager Installation Guide*.

3. Go through *HPE Structured Data Manager Tutorial*. The tutorial enables you to get hands on with the product quickly and exposes you to many of the most commonly used features. You will use these features to customize third party projects for your environment.

4. Obtain the integration kit and install it:

   - HPE Structured Data Manager *PeopleSoft Modules Installation and Deployment Guide*

   - HPE Structured Data Manager *Oracle E-Business Suite Modules Installation and Deployment Guide*

5. Customize the project.

6. When your business flow is ready, generate and deploy it according to the information in *HPE Structured Data Manager Developer's Guide*.

7. Run your deployed business flow. See the *HPE Structured Data Manager Runtime Guide* to help you perform the necessary tasks for running your business flow from the Web Console or command line.

8. As necessary, repeat step 5 through step 7 to implement additional customizations, re-deploy the business flow, and run the jobs.

9. When your receive a new version of the project from the vendor, merge the customizations in your project according to the information in *HPE Structured Data Manager Developers Guide*.

Glossary

active database

The database from which you plan to move or copy data. Typically, this database is your online transaction processing (OLTP) or production database. In a two- or three-tiered configuration, the active database resides on tier one and is the source for data movement operations.

active environment

The Web Console views and acts upon only one environment at a time, the active environment. To switch the active environment, you use the Change Active option in the Web Console.

activity

In Designer, a component of a business flow, which is added by using the toolbar. For example, you can add archive and reload activities to your business flow. Note, activities in a business flow are different from what you see at runtime and therefore do not necessarily map directly to what you see in the Web Console.

advanced selection

A method of data selection that discovers all of the interrelated rows from multiple tables and conceptually places them in the same application partition for archiving.

annotation

In Designer, a comment associated with the project, or one of its objects or components. These comments are collected and published in a PDF file when you right click a project or business flow and select Generate Documentation.

application partitioning

The concept of partitioning related rows together during data selection, regardless of whether they are in one or more tables. Application partitioning is unique to HPE Structured Data Manager and contrasts with the more common table partitioning offered by the database management software, which only groups related rows from one table.

archive data store

The location where the data is to be archived. Can be a separate database, separate space on the same database, or an XML file. In a two-tiered configuration, the archive data store resides on tier two and can be a database or XML. In a three-tiered configuration, the archive data store is a database on tier two and XML on tier three, and is the target for data movement operations.

archive query server

The component that provides SQL access to XML database archives.

Autonomy Consolidated Archive (ACA)

A managed, scalable repository that consolidates electronic communications, attachments, and files, and provides complete control over corporate information assets, facilitating compliance with internal corporate governance policies and procedures as well as externally mandated laws and regulations.

business flow

A series of activities, such as archive operations and scripts, that run in sequence. You build business flows in Designer.

business flow status

The Web Console shows the last run of each business flow. The states are Complete/Error/Running.

cartridge

An instance of model- or schema-based eligibility criteria used to move or copy data from one location to another. Cartridges capture the application and business rules to ensure referential integrity of the data. For any one model in your project, you may have many cartridges that use it.

chaining table

The lower level table in a many-to-one or a many-to-many relationship between higher level and lower level tables in the model hierarchy.

classification

The HPE Content Manager (formerly HP Records Manager) classification to be applied to the data moved by HPE Structured Data Manager. This classification specifies where to place the data when it is ingested by HPE Content Manager. For more information, see the HPE Content Manager documentation.

collection

The configuration of a directory location and file pattern to match a set of archived XML files, thus allowing SQL access to the archived data.

comma separated values (CSV)

A database-to-file output format that stores the data as values separated by commas and a metadata file. Each line in the CSV file corresponds to a row in a table. Within a line, fields are separated by commas, each field belonging to one table column. CSV files provide a simple format that many applications can import.

command

Command files or JavaScript files launched by the Web Console on your behalf with status displays.

condition

In Designer, the way you branch your business flow to run or skip an activity based on some criteria.

configuration parameter

A type of parameter that has its values set by an administrator (someone who has repository privileges from the Web Console) through the administrator interface. Typically, this type of parameter represents values that should be changed very infrequently, perhaps only at deployment time.

console user

The Web Console identifies individual users, who are distinct from database users. The properties for a user are User Name, Full Name, Password, Enabled, Description, Email, Phone, and Privileges.

console user name

The login name associated with a Web Console user.

constraint

A column or a list of columns that enables you to identify rows in the database and relate them to one another.

custom properties

User-created name/value pairs in cartridges and business flows. These values are exposed at runtime as parameters.

customization

A change that an administrator or DBA makes to a project provided by a third party, typically for a packaged application like Oracle PeopleSoft or Oracle E-Business Suite. As long as the customization is allowable by the project, the user can merge the customization into newer revisions of the third party project.

customization mode

A Designer mode that provides visual cues to indicate customizations in the model. In a project with locked files, customization mode is on by default, but you can toggle it on and off from the toolbar in the model editor.

data access cartridge

A cartridge that provides lightweight query access to retired or archived data. Data access cartridges are designed by the archive developer but can be run by business users with no technical expertise.

data masking

The process of replacing private or confidential data during movement with a specified mask. You can choose from pre-defined masks that are part of HPE or create your own mask. A mask may or may not be reversible upon reload from the archive data store.

data movement

The method used by HPE Structured Data Manager to actually move data (transactional, bulk or partitioned for database to database, and copy or archive for database to file).

data transparency

The ability to access archived data through your standard application interfaces for data access. Data transparency enables users to access archived data as though it were still in the active database.

database constraint

A constraint that exists in the database and can be discovered and referenced from Designer.

database to database

A movement in which data goes from an active database to an archive database, or separate tablespaces inside the active database. Typically, the archive database is located on cheaper storage devices.

database to file

A movement in which data goes from an active database to a file (XML or CSV format), which is offline but still accessible through SQL using the archive query server and a client tool of your choice.

deployment assistant

The user interface component within Designer used to deploy or generate business flows.

description

A technical description created by the developer for her own reference. These descriptions do not appear in the generated PDF file for the cartridge or business flow.

Designer

The user interface component used to develop, test, and deploy your archiving solution. Designer is a powerful graphical development environment for archive solutions.

distributed instance

A configuration option for database-to-database archiving where the data you archive is stored on a separate database from the source or active database.

DRE

See Dynamic Reasoning Engine (DRE).

driving table

A driving object is a root of a model hierarchy. Its relationship to the child tables drives the

selection of transactions.

dynamic list of values

A list of values for a parameter that obtains its members from a SELECT statement that returns identifiers and labels.

dynamic parameter

A type of parameter that has its value set by a Groovy script that runs at deployment time to obtain a value. For example, this type of parameter can supply the type or version of a database or application, which can be obtained programmatically at deployment time.

Dynamic Reasoning Engine (DRE)

A platform technology that uses high performance pattern-matching algorithms to search for content stored in Autonomy repositories. Performs core information operations for contextual analysis and concept extraction, enabling solutions for the categorization, summarization, personalization, hyperlinking, and retrieval of all forms of information.

environment

The source and (optional) target credentials against which you plan to run commands. You can define multiple environments within your installation to identify various source and target databases.

error

One of the ways in which you can interrupt a business flow. Error indicates that the business flow failed for some reason.

exclusive rules

One of the ways in which HPE Structured Data Manager determines whether to include or exclude rows from the archive operation. Exclusive rules require all rows in the constraint table to match for inclusion. Exclusive rules exclude the instance if the condition on any child is false, like STATUS='CLOSED'.

exit

One of the ways in which you can interrupt a business flow. You can exit successfully or with a warning.

export

The way that you save an HPE Structured Data Manager project to an exchange format (.hdp) from the File menu. See also *import*.

export data

The way that a user can send data to CSV format from Preview using the toolbar item.

generate documentation

The process of collecting and grouping all annotations into a PDF file that also describes the business flow or cartridge structure.

history schema

For database-to-database archiving, the schema in the target database where the archived data is stored.

IDOL

See Intelligent Data Operating Layer (IDOL).

import

The way that you transfer projects from exchange format (.hdp) into the Project Navigator. You can also use import to migrate cartridges created in 5.1 to 6.x. See also *export*.

inclusive rules

One of the ways in which HPE Structured Data Manager determines whether to include or exclude rows from the archive operation. Inclusive rules require only one row in the constraint table to match the rule and be included. Inclusive rules include the instance if the condition on any child is true, like PRODUCT_RECALLED='Y'.

indexing cartridge

A cartridge that indexes your data for better searching. For example, you might associate an indexing cartridge with a database-to-file archiving cartridge to improve performance when querying the archive data files.

Intelligent Data Operating Layer (IDOL)

An information processing layer that collects indexed data from connectors and stores it in a structure optimized for fast processing and retrieval, integrating unstructured, semi-structured, and structured information from multiple repositories.

interrupt

The way to stop or pause a business flow (pause, error, exit with warning, exit successfully).

local deployment

The generation and deployment of your cartridge or business flow to an environment on your local, Designer client. Deployment files are generated locally and then deployed to the designated, local environment.

lookup table

A table that contains helpful non-transactional information. For example, non-transactional information could be status definitions, or the name of the sales representative.

managed table

A table in the model that is copied and then purged from the active database by a cartridge. Transactional, chaining, and driving tables in a model are all typically managed tables.

model

A model identifies the tables and table relationships representing a business entity or related business entities. A project can have multiple models. Each model contains a driving table and all of its child and descendent tables.

model compatibility

Each model in your project can have one or more dynamic parameters associated with it to verify the compatibility with the target environment. If the compatibility parameter returns false, then the cartridge referencing the model will not deploy or run and throw an error. For example, the script could return false for Oracle 10.2 and true for Oracle 11.1 to indicate that a cartridge referencing the model can only deploy and run against Oracle 11.1.

model-based cartridge

A cartridge that moves data based upon a defined data model with relationships. This type of cartridge is typically used for ongoing archive operations.

non-intrusive environment

In a non-intrusive environment, data is archived without an interface schema and a generic

JDBC driver is used. A non-intrusive environment enables you to copy or archive data from read-only sources, which is especially helpful in cases where the data is associated with older technologies that might not support basic SQL statements or when the database administrator or company policy prohibits write access to the production environment.

OLTP database

The online transaction processing database that typically is your active or source database.

pause

One of the ways in which you can interrupt a business flow. Pausing suspends the business flow while awaiting operator intervention.

Records Manager

Enterprise document and records management software designed to simplify the capture, management, security, and access to information. Records Manager enables organizations to more easily comply with regulations and corporate policies, and it helps secure information from inappropriate access and misuse.

reload

The act of taking data from an archive data store and loading it into the active database.

remote deployment

The generation and deployment of your cartridge or business flow to an environment on a system that is remote from your Designer client. Deployment files are generated locally and then deployed to the designated, remote environment.

repository

The location that holds business flow metadata, product configuration data, and data collected during runtime. The repository can be located on your active database or another logical database.

rule

Qualifications added to the model in order to include or exclude data based on certain criteria. For example, you might add a rule to exclude from archiving any orders that are not yet closed.

runtime parameter

A type of parameter that has its values set by the operator executing the job in Console or on the command line. Typically, this type of parameter represents operational values that tend to change frequently and therefore need to be set each time the job is run.

schema-based cartridge

A cartridge that moves data based upon the database schema rather than a defined data model with relationships. This type of cartridge is typically used for database retirement or the cleanup of orphan tables.

selection

The form of data selection to use (standard or advanced) for choosing data. When deploying a cartridge or adding it to a business flow, you must specify the selection method.

single instance

A configuration option for database-to-database archiving where the data you archive is stored on the same database (Oracle) or the same server (SQL Server) as the source or active database.

source

The location (database) from which you are copying or moving data.

SQL access server

See *archive query server*.

standard selection

A method of data selection that restricts itself to the rows identified by the model. Unlike advanced selection, it does not attempt to traverse related rows across multiple tables.

structured records management

A type of solution that extracts structured data from a source application and moves it into XML format. The XML is then ingested into the corporate records management system for long term management and eventual disposal according to corporate policy.

table use

A database table, view, or synonym that is referenced in Designer, for example, in the model. The same table can be used multiple times in a model. For example, a table could be

appear as a transactional table and a lookup table in the same model.

target

The location (database or XML) to which you are copying or moving data.

tier

A level in your database archiving configuration. You can have two- or three-tiered configurations. In a two-tiered configuration, tier one contains your active database and tier two your archive data store, which can be a database or XML. In a three-tiered configuration, tier one contains your active database, tier two an archive database, and tier three XML.

transactional data movement

Transactional movement uses set-based data movement and is the default method of movement.

transactional table

A table that contains information about the business transaction. For example, a transactional table might contain detailed tax or payment information related to each business transaction.

unique identifiers (UIDs)

A 16 hexadecimal identifier calculated based on the content of a Designer file. This value is used to determine if the user has customized key pieces of a project.

unmanaged table

A table in a model that is copied but not purged from the active database by a cartridge. Lookup tables in a model are typically unmanaged tables.

Vertica

Column-oriented SQL database management software for storing and analyzing structured data. Used to manage large, fast-growing volumes of data and provide fast query performance for data warehouses and other query-intensive applications.

virtual constraint

A constraint that you define in Designer that only exists within HPE Structured Data Manager.

Web Console

A browser-based interface where you can create and manage your deployment environments, and deploy, run, administer, and monitor your business flows.

Index

Send documentation feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on Concepts Guide (HPE Structured Data Manager 7.51)

Add your feedback to the email and click **Send**.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to autonomyTPfeedback@hpe.com.

We appreciate your feedback!