|  |
| --- |
| EPAM Systems, RD Dep. |
| MTN.BI.08 Transportation |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| REVISION HISTORY | | | | | |
| Ver. | Description of Change | Author | Date | Approved | |
| Name | Effective Date |
| 1.0 | Initial status | [Elias Nema](mailto:Elias_Nema@epam.com) | 20-JAN-2014 |  |  |
|  |  |  |  |  |  |

Contents

[1. Overview of Transportation in Data Warehouses 3](#_Toc384019707)

[1.1. Introduction to Transportation Mechanisms in Data Warehouses 3](#_Toc384019708)

[1.1.1. Transportation Using Flat Files 3](#_Toc384019709)

[1.1.2. Transportation Through Distributed Operations 3](#_Toc384019710)

[1.1.3. Transportation Using Transportable Tablespaces 3](#_Toc384019711)

[1.2. Transportable Tablespaces Example 4](#_Toc384019712)

[1.2.1. Step 1. Place the Data to be Transported into its own Tablespace 4](#_Toc384019713)

[1.2.2. Step 2. Export the Metadata 4](#_Toc384019714)

[1.2.3. Step 3. Copy the Datafiles and Export File to the Target System 5](#_Toc384019715)

[1.2.4. Step 4. Import the Metadata 5](#_Toc384019716)

[1.3. Other Uses of Transportable Tablespaces 5](#_Toc384019717)

[2. Database Links 5](#_Toc384019718)

[2.1. What Are Database Links? 5](#_Toc384019719)

[3. Synonyms 6](#_Toc384019720)

[3.1. Create Synonym 7](#_Toc384019721)

[3.1.1. CREATE SYNONYM EXAMPLE 7](#_Toc384019722)

[3.2. Drop Synonym 7](#_Toc384019723)

[3.2.1. DROP SYNONYM EXAMPLE 7](#_Toc384019724)

[4. Source Books and Articles 8](#_Toc384019725)

# Overview of Transportation in Data Warehouses

Transportation is the operation of moving data from one system to another system. In a data warehouse environment, the most common requirements for transportation are in moving data from:

* A source system to a staging database or a data warehouse database
* A staging database to a data warehouse
* A data warehouse to a data mart

Transportation is often one of the simpler portions of the ETL process, and can be integrated with other portions of the process. For example, distributed query technology provides a mechanism for both extracting and transporting data.

## Introduction to Transportation Mechanisms in Data Warehouses

You have three basic choices for transporting data in warehouses:

1. Transportation Using Flat Files
2. Transportation Through Distributed Operations
3. Transportation Using Transportable Tablespaces

### Transportation Using Flat Files

The most common method for transporting data is by the transfer of flat files, using mechanisms such as FTP or other remote file system access protocols. Data is unloaded or exported from the source system into flat files using techniques discussed in Chapter 13, "Extraction in Data Warehouses", and is then transported to the target platform using FTP or similar mechanisms.

Because source systems and data warehouses often use different operating systems and database systems, using flat files is often the simplest way to exchange data between heterogeneous systems with minimal transformations. However, even when transporting data between homogeneous systems, flat files are often the most efficient and most easy-to-manage mechanism for data transfer.

### Transportation through Distributed Operations

Distributed queries, either with or without gateways, can be an effective mechanism for extracting data. These mechanisms also transport the data directly to the target systems, thus providing both extraction and transformation in a single step. Depending on the tolerable impact on time and system resources, these mechanisms can be well suited for both extraction and transformation.

As opposed to flat file transportation, the success or failure of the transportation is recognized immediately with the result of the distributed query or transaction. See Chapter 13, "Extraction in Data Warehouses" for further information.

### Transportation Using Transportable Tablespaces

Oracle transportable tablespaces are the fastest way for moving large volumes of data between two Oracle databases. Previous to the introduction of transportable tablespaces, the most scalable data transportation mechanisms relied on moving flat files containing raw data. These mechanisms required that data be unloaded or exported into files from the source database, then, after transportation, these files were loaded or imported into the target database. Transportable tablespaces entirely bypass the unload and reload steps.

Using transportable tablespaces, Oracle data files (containing table data, indexes, and almost every other Oracle database object) can be directly transported from one database to another. Furthermore, like import and export, transportable tablespaces provide a mechanism for transporting metadata in addition to transporting data.

Transportable tablespaces have some limitations: source and target systems must be running Oracle8i (or higher), must use compatible character sets, and, before Oracle Database 10g, must run on the same operating system. For details on how to transport tablespace between operating systems, see Oracle Database Administrator's Guide.

The most common applications of transportable tablespaces in data warehouses are in moving data from a staging database to a data warehouse, or in moving data from a data warehouse to a data mart.

## Transportable Tablespaces Example

Suppose that you have a data warehouse containing sales data, and several data marts that are refreshed monthly. Also suppose that you are going to move one month of sales data from the data warehouse to the data mart.

### Step 1. Place the Data to be transported into its own Tablespace

The current month's data must be placed into a separate tablespace in order to be transported. In this example, you have a tablespacets\_temp\_sales, which holds a copy of the current month's data. Using the CREATE TABLE ... AS SELECT statement, the current month's data can be efficiently copied to this tablespace:

CREATE TABLE temp\_jan\_sales NOLOGGING TABLESPACE ts\_temp\_sales

AS SELECT \* FROM sales

WHERE time\_id BETWEEN '31-DEC-1999' AND '01-FEB-2000';

Following this operation, the tablespacets\_temp\_sales is set to read-only:

ALTER TABLESPACE ts\_temp\_sales READ ONLY;

A tablespace cannot be transported unless there are no active transactions modifying the tablespace. Setting the tablespace to read-only enforces this.

The tablespacets\_temp\_sales may be a tablespace that has been especially created to temporarily store data for use by the transportable tablespace features. Following "Copy the Datafiles and Export File to the Target System", this tablespace can be set to read/write, and, if desired, the table temp\_jan\_salescan be dropped, or the tablespace can be re-used for other transportations or for other purposes.

In a given transportable tablespace operation, all of the objects in a given tablespace are transported. Although only one table is being transported in this example, the tablespacets\_temp\_sales could contain multiple tables. For example, perhaps the data mart is refreshed not only with the new months’ worth of sales transactions, but also with a new copy of the customer table. Both of these tables could be transported in the same tablespace. Moreover, this tablespace could also contain other database objects such as indexes, which would also be transported.

Additionally, in a given transportable-tablespace operation, multiple tablespaces can be transported at the same time. This makes it easier to move very large volumes of data between databases. Note, however, that the transportable tablespace feature can only transport a set of tablespaces which contain a complete set of database objects without dependencies on other tablespaces. For example, an index cannot be transported without its table, nor can a partition be transported without the rest of the table. You can use the DBMS\_TTS package to check that a tablespace is transportable.

In this step, we have copied the January sales data into a separate tablespace; however, in some cases, it may be possible to leverage the transportable tablespace feature without even moving data to a separate tablespace. If the sales table has been partitioned by month in the data warehouse and if each partition is in its own tablespace, then it may be possible to directly transport the tablespace containing the January data. Suppose the January partition, sales\_jan2000, is located in the tablespace ts\_sales\_jan2000. Then the tablespace ts\_sales\_jan2000 could potentially be transported, rather than creating a temporary copy of the January sales data in the ts\_temp\_sales.

However, the same conditions must be satisfied in order to transport the tablespace ts\_sales\_jan2000 as are required for the specially created tablespace. First, this tablespace must be set to READ ONLY. Second, because a single partition of a partitioned table cannot be transported without the remainder of the partitioned table also being transported, it is necessary to exchange the January partition into a separate table (using the ALTER TABLE statement) to transport the January data. The EXCHANGE operation is very quick, but the January data will no longer be a part of the underlying sales table, and thus may be unavailable to users until this data is exchanged back into the sales table after the export of the metadata. The January data can be exchanged back into the sales table after you complete step 3.

### Step 2. Export the Metadata

The Export utility is used to export the metadata describing the objects contained in the transported tablespace. For our example scenario, the Export command could be:

EXP TRANSPORT\_TABLESPACE=y TABLESPACES=ts\_temp\_sales FILE=jan\_sales.dmp

This operation generates an export file, jan\_sales.dmp. The export file is small, because it contains only metadata. In this case, the export file contains information describing the table temp\_jan\_sales, such as the column names, column data type, and all other information that the target Oracle database needs in order to access the objects in ts\_temp\_sales.

### Step 3. Copy the Datafiles and Export File to the Target System

Copy the data files that make up ts\_temp\_sales, as well as the export file jan\_sales.dmp to the data mart platform, using any transportation mechanism for flat files. Once the datafiles have been copied, the tablespacets\_temp\_sales can be set to READ WRITE mode if desired.

### Step 4. Import the Metadata

Once the files have been copied to the data mart, the metadata should be imported into the data mart:

IMP TRANSPORT\_TABLESPACE=y DATAFILES='/db/tempjan.f'

TABLESPACES=ts\_temp\_sales FILE=jan\_sales.dmp

At this point, the tablespacets\_temp\_sales and the table temp\_sales\_jan are accessible in the data mart. You can incorporate this new data into the data mart's tables.

You can insert the data from the temp\_sales\_jan table into the data mart's sales table in one of two ways:

INSERT /\*+ APPEND \*/ INTO sales SELECT \* FROM temp\_sales\_jan;

Following this operation, you can delete the temp\_sales\_jan table (and even the entire ts\_temp\_salestablespace).

Alternatively, if the data mart's sales table is partitioned by month, then the new transported tablespace and the temp\_sales\_jan table can become a permanent part of the data mart. The temp\_sales\_jan table can become a partition of the data mart's sales table:

ALTER TABLE sales ADD PARTITION sales\_00jan VALUES

LESS THAN (TO\_DATE('01-feb-2000','dd-mon-yyyy'));

ALTER TABLE sales EXCHANGE PARTITION sales\_00jan

WITH TABLE temp\_sales\_jan INCLUDING INDEXES WITH VALIDATION;

## Other Uses of Transportable Tablespaces

The previous example illustrates a typical scenario for transporting data in a data warehouse. However, transportable tablespace scan be used for many other purposes. In a data warehousing environment, transportable tablespaces should be viewed as a utility (much like Import/Export or SQL\*Loader), whose purpose is to move large volumes of data between Oracle databases. When used in conjunction with parallel data movement operations such as the CREATE TABLE ... AS SELECT and INSERT ... AS SELECT statements, transportable tablespaces provide an important mechanism for quickly transporting data for many purposes.

# Database Links

The central concept in distributed database systems is a database link. A database link is a connection between two physical database servers that allows a client to access them as one logical database.

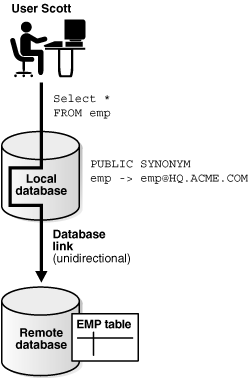
## What Are Database Links?

A database link is a pointer that defines a one-way communication path from an Oracle Database server to another database server. The link pointer is actually defined as an entry in a data dictionary table. To access the link, you must be connected to the local database that contains the data dictionary entry.

A database link connection is one-way in the sense that a client connected to local database A can use a link stored in database A to access information in remote database B, but users connected to database B cannot use the same link to access data in database A. If local users on database B want to access data on database A, then they must define a link that is stored in the data dictionary of database B.

A database link connection allows local users to access data on a remote database. For this connection to occur, each database in the distributed system must have a unique global database name in the network domain. The global database name uniquely identifies a database server in a distributed system.

Figure 1 shows an example of user scott accessing the emp table on the remote database with the global name hq.acme.com:



**Figure 1 Database Link**

Database links are either private or public. If they are private, then only the user who created the link has access; if they are public, then all database users have access.

One principal difference among database links is the way that connections to a remote database occur. Users access a remote database through the following types of links:

|  |  |
| --- | --- |
| Type of Link | Description |
| Connected user link | Users connect as themselves, which means that they must have an account on the remote database with the same username and password as their account on the local database. |
| Fixed user link | Users connect using the username and password referenced in the link. For example, if Jane uses a fixed user link that connects to the hq database with the username and password scott/tiger, then she connects as scott, Jane has all the privileges in hq granted to scott directly, and all the default roles that Scott has been granted in the hq database. |
| Current user link | A user connects as a global user. A local user can connect as a global user in the context of a stored procedure, without storing the global user's password in a link definition. For example, Jane can access a procedure that Scott wrote, accessing Scott's account and Scott's schema on the hq database. Current user links are an aspect of Oracle Advanced Security. |

# Synonyms

A synonym is an alternative name for objects such as tables, views, sequences, stored procedures, and other database objects.

You generally use synonyms when you are granting access to an object from another schema and you don't want the users to have to worry about knowing which schema owns the object.

## Create Synonym

The syntax to create a synonym in Oracle is:

CREATE [OR REPLACE] [PUBLIC] SYNONYM [schema .] synonym\_name

FOR [schema .] object\_name [@ dblink];

* OR REPLACE allows you to recreate the synonym (if it already exists) without having to issue a DROP synonym command.
* PUBLIC means that the synonym is a public synonym and is accessible to all users. Remember though that the user must first have the appropriate privileges to the object to use the synonym.
* Schema is the appropriate schema. If this phrase is omitted, Oracle assumes that you are referring to your own schema.
* object\_name is the name of the object for which you are creating the synonym. It can be one of the following:
  + table
  + view
  + sequence
  + stored procedure
  + function
  + package
  + materialized view
  + java class schema object
  + user-defined object
  + synonym

### CREATE SYNONYM EXAMPLE

Let's look at an example of how to create a synonym in Oracle.

For example:

CREATE PUBLIC SYNONYM suppliers

FOR app.suppliers;

This first CREATE SYNONYM example demonstrates how to create a synonym called suppliers. Now, users of other schemas can reference the table called suppliers without having to prefix the table name with the schema named app. For example:

SELECT \*

FROM suppliers;

If this synonym already existed and you wanted to redefine it, you could always use the OR REPLACE phrase as follows:

CREATE OR REPLACE PUBLIC SYNONYM suppliers

FOR app.suppliers;

## Drop Synonym

Once a synonym has been created in Oracle, you might at some point need to drop the synonym.

DROP SYNONYM SYNTAX

The syntax to drop a synonym in Oracle is:

DROP [PUBLIC] SYNONYM [schema .] synonym\_name [force];

* PUBLIC allows you to drop a public synonym. If you have specified PUBLIC, then you don't specify a schema.
* Force will force Oracle to drop the synonym even if it has dependencies. It is probably not a good idea to use force as it can cause invalidation of Oracle objects.

### DROP SYNONYM EXAMPLE

Let's look at an example of how to drop a synonym in Oracle.

For example:

DROP PUBLIC SYNONYM suppliers;

This DROP statement would drop the synonym called suppliers that we defined earlier.

# Source Books and Articles

1. Lane, P. Oracle Database Data Warehousing Guide, 11g Release 2 (11.2). Redwood City: Oracle, 2013.
2. Powell G. Oracle Data Warehouse Tuning for 10g. Oxford: Elsevier Digital Press, 2005.