Oracle® E-Business Suite Concepts

Database Features

Introduction

This chapter describes the Oracle E-Business Suite data model, including schemas, Oracle user IDs, and related database server features.

Schemas

A given Oracle database can store the objects associated with a single installation of Oracle E-Business Suite. In general, product *code* objects are stored in the *APPS schema*, whereas product *data* objects are stored in the relevant *base product schemas*. These schemas are described further below.

The EBS_SYSTEM Schema

AD-TXK Delta 13 modernized the Oracle E-Business Suite database architecture by introducing a new schema, *EBS_SYSTEM*, which is analogous to the Oracle Database's SYSTEM schema. Prior to the introduction of the EBS_SYSTEM schema, Oracle E-Business Suite installed application objects in the Oracle Database SYS and SYSTEM schemas. Migration to the EBS_SYSTEM schema obviates the need for any objects owned by Oracle E-Business Suite to reside in the SYS or SYSTEM schemas.

Key characteristics of a Oracle E-Business Suite installation that has been migrated to the EBS_SYSTEM schema include:

- All Oracle E-Business Suite database objects that once resided in the SYS or SYSTEM schemas were migrated to appropriate Oracle E-Business Suite schemas. Depending upon the Oracle E-Business Suite object type and function, the object was migrated to EBS_SYSTEM, APPS, or APPS_NE.
- All Oracle E-Business Suite administration actions (such as running `adop`, `adadmin` and other utilities) will now prompt for the EBS_SYSTEM password instead of the SYSTEM password. Highly privileged operations that were previously run by the SYS or SYSTEM accounts are now run by EBS_SYSTEM.
- Access to the Oracle Database SYS and SYSTEM accounts, and the Oracle Database server operating system, are no longer required for Oracle E-Business Suite system administrative functions.

Additional Information: To learn more about migrating to the EBS_SYSTEM schema and the procedures for managing privileges in a migrated environment, refer to My Oracle Support Knowledge Document 2755875.1, *Oracle E-Business Suite Release 12.2 System Schema Migration*.

The APPS Schema

The APPS schema has access to the complete Oracle E-Business Suite data model. Oracle E-Business Suite responsibilities connect to an APPS schema, and the environment variable FNDNAM is set to the name of the APPS schema. The APPS schema owns all the code objects for the Oracle E-Business Suite, and has access to all data objects. There is one APPS schema for every product installation group.

Utilizing a single schema that has access to all objects avoids cross-product dependencies, and creates a hub-and-spoke access model rather than the spider web model that would otherwise be needed. The APPS schema also improves the reliability of and reduces the time needed for installation, upgrading, and patching, by eliminating the need for cross-product grants and synonyms.

The following code objects are installed in the APPS schema:

- Packages
- Procedures
- Functions
- Triggers
- Views
- Materialized views
- Java classes
- Queues

Base Product Schemas

All data objects for a product are owned by a specific schema for that product, known as the base product schema.

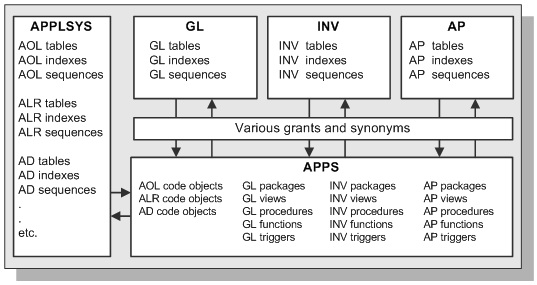
The following objects are installed in the base product schemas:

- Tables
- Sequences
- Indexes
- Constraints
- Queues

Relationship Between APPS Schema and Base Product Schemas

The base product schemas also contain grants from various tables and sequences to the APPS schema, as well as synonyms from the APPS schema to the same objects.

APPS Schema and Base Product Schemas



Custom Schema Access

In some circumstances, you may wish to create a schema that has limited or read-only access to Oracle E-Business Suite data.

Warning: Since the APPS schema has all privileges to all Oracle E-Business Suite objects, you should never give users direct access to this schema.

You will need to grant access on objects to the user schema from the base product schema.

Note: You may need to re-grant access if the underlying object is dropped and recreated.

Schemas and Data Access

Some views access packages or functions, where the value returned by the package or function may depend on the environment having been set up properly. The environment is initialized automatically when accessing Oracle E-Business Suite through the Sign-On screen, or when using concurrent processing with Oracle Reports or SQL scripts.

Consequently, if you connect directly to a schema, the rows returned by the view may be different from those returned when you are running in an Oracle E-Business Suite environment. For example, a view may reference a profile option, but when accessed from SQL*Plus the site value of the profile option – rather than the setting for a particular Oracle E-Business Suite user – will be used.

Oracle User IDs

Each Oracle E-Business Suite product has a default Oracle user ID, with the product abbreviation used as both the schema name and password (unless you set a different password durining installation). For example, the default Oracle user ID/password combination for Oracle General Ledger is *GL/GL*.

Important: For security, Oracle recommends taking advantage of the option to change the default passwords during installation. However, you should not change the default user IDs.

A product's schema determines the ownership of the product's data objects, such as sequences, tables, and indexes. If two products are installed under the same schema, that schema owns the data objects for both products.

Since a product's data objects are created in their own schema (such as the GL schema), but the user accesses all data objects through the APPS schema, appropriate grants and synonyms are required between the APPS schema and the base product schemas (see earlier figure for details of the schemas).

Space Management

This section discusses how the Oracle database is set up to meet the space management needs of Oracle E-Business Suite. It provides information on tablespaces, firstly outlining the basic tablespaces required, then discussing the traditional tablespace structure used to support Oracle E-Business Suite products, and finally describing the tablespace model that is used as standard with Oracle E-Business Suite Release 12.2.

Important: Oracle E-Business Suite Release 12.2 requires an Oracle database block size of 8K. No other block size may be used.

Introduction to Tablespaces

An Oracle 11g database always requires the following tablespaces to be available:

- **System Tablespace** - This tablespace holds data dictionary tables owned by the SYS account, and is created when the database is installed.
- **Undo Tablespace** - This tablespace holds undo (rollback) information that is used to track database changes until they are either committed or undone (rolled back).
- **Temporary Tablespace** - Temporary tablespaces are used to sort data while it is being processed. It is possible to use a single temporary tablespace, typically called TEMP, for all Oracle E-Business Suite products. Alternatively, separate temporary tablespaces can, if desired, be created for individual products. Since users access Oracle E-Business Suite objects through the APPS schema, the temporary tablespace for that schema (initially the same as that for the Oracle Application Object Library) is used by all products.

The traditional Oracle E-Business Suite tablespace model employed separate tablespaces for a product's tables and indexes. The resulting tablespaces were named by appending 'D' for data or 'X' for an index to the product's short name or Oracle schema name. For example, APD was the tablespace for Oracle Payables data, and APX was the tablespaces for Oracle Payables indexes.

Additional Information: For further information about tablespaces, see the *Oracle Database Administrator's Guide 11g*.

Employing separate table and index tablespaces for each product made it easier maintain products, and helped to improve database performance. However, with an increasing number of products, this model could easily require several hundred product tablespaces, plus a system tablespace, undo (rollback) tablespace, and temporary tablespace.

In addition, the traditional tablespace model used a database *sizing factor* to set the extent sizes for an Oracle E-Business Suite product's tables and indexes. The value of this factor was a percentage of the typical estimated growth rate for Oracle E-Business Suite database objects. The sizing factor affected only the size of subsequent extents, as determined by the NEXT database object creation parameter. Most objects were defined with small first extents and larger additional extents.

During installation, Rapid Install provides the option of distributing tablespaces across different disks, to reduce disk head contention and improve overall system performance. In addition to this, many production systems utilize sophisticated disk and volume management technologies at operating system level to further enhance performance.

Oracle Applications Tablespace Model

Oracle E-Business Suite Release 12.2 utilizes as standard a modern infrastructure for tablespace management, the *Oracle Applications Tablespace Model* (OATM). The Oracle Applications Tablespace Model is similar to the traditional model in retaining the system, undo, and temporary tablespaces. The key difference is that Oracle E-Business Suite products in an OATM environment share a much smaller number of tablespaces, rather than having their own dedicated tablespaces.

Oracle E-Business Suite schema objects are allocated to the shared tablespaces based on two main factors: the type of data they contain, and I/O characteristics such as size, life span, access methods, and locking granularity. For example, tables that contain *seed data* are allocated to a different tablespace from the tables that contain *transactional data*. In addition, while most indexes are held in the same tablespace as the base table, indexes on transaction tables are held in a single tablespace dedicated to such indexes.

The Oracle Applications Tablespace Model provides a variety of benefits, which are summarized in the list below:

- Simplifies maintenance and recovery by using far fewer tablespaces than the older model.
- Uses locally managed tablespaces, whose choice of extent management types (autoallocate or uniform) allows more precise control over unused space and hence can help reduce fragmentation.
- Takes advantage of automatic segment space management, eliminating the need for manual space management tasks.
- Increases block-packing compared to the older model, reducing the overall number of buffer gets and thereby improving runtime performance.
- Makes best use of the restricted number of raw devices available in Oracle Real Applications Cluster (Oracle RAC) and other environments, where every tablespace requires its own raw device.
- Maximizes usefulness of wide disk stripe configurations.

In Oracle database server releases prior to Oracle9i, undo space management was performed using rollback segments. For clarity, this method is now referred to as *manual undo management*. Its successor, *automatic undo management* is based on the use of a small number of *undo tablespaces*, in contrast to the larger number of variously-sized rollback segments typically used in manual undo management.

Note: For more information on the Oracle Applications Tablespace Model, refer to *Oracle E-Business Suite Setup Guide*.

Performance Features

Database performance features include optimization, resource usage, space management, and access rights.

Query Optimization

The SQL used in Release 12.2 has been extensively tuned for *cost-based optimization*. In calculating the lowest cost (most efficient) method of executing an SQL statement, the Oracle query optimizer evaluates many factors to calculate the most efficient way to execute a SQL statement. For example, the optimizer considers the available access paths, factoring in statistical information for the tables and indexes that the SQL statement will access. The optimizer also considers *hints*, which are optimization suggestions placed in a comment of the SQL statement.

As part of its operation, the optimizer creates a set of potential execution plans for the SQL statement, based on the available access paths and any hints. It then estimates the *cost* of each execution plan, based on data dictionary statistics for the data distribution and storage characteristics of the tables, indexes, and partitions. Finally, the optimizer compares the costs of the execution plans and chooses the one with the smallest cost, which means optimum execution characteristics.

For some operations, such as batch processing, Release 12.2 uses cost-based optimization to achieve the most efficient means of processing *all rows* that are accessed by the statement. For other operations, such as accessing forms or communication with the desktop client, Release 12.2 uses cost-based optimization to achieve the best response time for processing the *first rows* that are accessed by the statement.

Several other Oracle database performance features in Release 12.2 also require use of the cost-based query optimizer.

Note: For further details of optimization, see: *Oracle Database Concepts* and *Oracle Database Performance Tuning Guide*.

Database Resource Manager

The *Database Resource Manager* gives the system administrator extensive control over processing resources on the database node. The administrator can distribute server CPU based on business rules, ensuring that the highest priority activities always have sufficient CPU resources. The administrator could, for example, guarantee Order Entry users 40% of CPU resources during business hours, regardless of the load or number of users in other groups on the system.

System administrators can also use the Database Resource Manager to limit the impact of any inefficient ad hoc queries. For example, a limit of 5% of CPU resources could be placed on ad hoc queries against the database.

Additional Information: For further details, see *Oracle Database Concepts* and *Oracle Database Administrator's Guide*

Database Partitioning

Partitioning helps support very large tables and indexes by dividing them into smaller, more manageable pieces called *partitions*. Once the desired partitions have been defined, SQL statements containing the partition key can access them instead of the original tables or indexes and thereby reduce the I/O needed.

Partitioning can significantly enhance performance and manageability. For example, the speed of operations that involve copying or deleting data may be greatly improved by the use of partitioned tables. Operations that might have taken hours might now be completed in minutes. This can be useful in data warehouse applications.

Important: Custom partitioning of standard Oracle E-Business Suite tables in Release 12.2 is fully supported for objects that are not already partitioned.

Partitioning should always be planned and tested carefully before being implemented on a critical system. After implementation, it is essential to check that the desired performance benefits have been achieved.

Note: For more information about partitioning, refer to My Oracle Support Knowledge Document 554539.1,*Using Database Partitioning with Oracle E-Business Suite*.

Segment Creation on Demand

This feature allows initial segment creation for nonpartitioned tables and indexes to be delayed until data is first inserted into an object. This is beneficial because while some applications modules need to be delivered with large schemas that contain many tables and indexes, only a subset of these objects may actually be used. Use of delayed segment creation means that empty database objects do not consume any space, reducing disk usage and speeding up installation.

Nologging Operations

The *nologging* Oracle database feature is used to enhance performance in certain areas of Oracle E-Business Suite. For example, it may be used during patch installation, and when building summary data for Business Intelligence.

Use of nologging in an operation means that the database redo logs will contain incomplete information about the changes made, with any data blocks that have been updated during the nologging operation being marked as *invalid*. As a result, a database restoration to a point in time (whether from a hot backup or a cold backup) may require additional steps in order to bring the affected data blocks up-to-date, and make the restored database usable. These additional steps may involve taking new backups of the associated datafiles, or by dropping and rebuilding the affected objects. The same applies to activation of a standby database.

Note: Oracle Database 11g also allows logging to be forced to take place, ensuring all data changes are written to the database redo logs in a way that can be recreated in a restored backup, or propagated to a standby database. See *Oracle Data Guard Concepts and Administration 11g* for details of the *force logging* clause for database and tablespace commands.

Nologging Principles

At certain times, Oracle E-Business Suite uses the database nologging feature to perform resource-intensive work more efficiently. When an operation uses nologging, blocks of data are written directly to their data file, rather than going through the buffer cache in the System Global Area (SGA).

Instance recovery uses the online redo logs to reconstruct the SGA after a crash, rolling forward through any committed changes in order to ensure the data blocks are valid. Use of nologging does not affect instance recovery.

Database recovery requires rolling forward through the redo logs to recreate the requisite changes, and hence restore the database to the desired point in time. Since nologging operations write directly to the data files, bypassing the redo logs, the redo logs will not contain enough data to roll forward to perform media recovery. Instead, they will only contain enough information to mark the new blocks as invalid. Rolling forward through a nologging operation would therefore result in invalid blocks in the restored database. The same problems will potentially occur upon activating a standby database.

To make the restored backup or activated standby database usable after a nologging operation is carried out, a mechanism other than database recovery must be used to get or create current copies of the affected blocks.

There are two options, either of which may be appropriate depending on the specific circumstances:

- Create a new copy of the data files after the nologging operation is complete, either by backing up the tablespace again, or by refreshing the specific data files in the standby database.
- Drop and recreate the object with the invalidated blocks, using the program that maintains the object.

Nologging Usage

Nologging is used in the following situations in the Oracle E-Business Suite:

- Building new objects during patch application, where use of nologging makes the initial build faster, and the downtime required for patching shorter.
- Changing the physical structure of existing objects during patch application (such as partitioning a table), where use of nologging reduces the time needed for the operation itself, and consequently the overall downtime.
- Certain specialized tasks where logging is not required, such as manipulating data for data warehousing applications, or maintaining summary data for business intelligence queries.
- Certain concurrent manager jobs. In most such cases, the object affected by nologging will be dropped at the end of the job, and the invalidated blocks cleaned up. If a recovery is needed while concurrent jobs are in progress, re-running the affected jobs will clean up any invalidated blocks that may exist.

Actions Needed

To monitor nologging activity in your environment, you should periodically query your production database to identify any datafiles that have experienced nologging operations. You should also run the query before and after applying an Oracle E-Business Suite patch, to determine whether any nologging activity was carried out.

A suitable query can be run via monitoring software such as Oracle Enterprise Manager. Alternatively, you can construct a query based on the *unrecoverable_change#* and *unrecoverable_time* columns of the data dictionary view *v$datafile*. These are updated every time an unrecoverable or nologging operation marks blocks as invalid in the datafile.

The results of a query can be saved as a snapshot and compared to the last snapshot. You can then identify each occasion when nologging operations have been carried out in the database, and hence when you need to refresh backup datafiles with new copies that will be usable in the event of restoration being needed.

Scalability Features

As well as providing more computing power, multi-node systems facilitate the addition of machines to meet increases in demand. They also provide resilience in the event of failures of individual components.

Oracle Real Application Clusters

Oracle Real Application Clusters (Oracle RAC) harness the processing power of multiple interconnected computers. Oracle RAC software called *Oracle Clusterware* and a collection of computers (known as a *cluster*) harness the processing power of each component to create a robust and powerful computing environment. A large task divided into subtasks and distributed among multiple nodes is completed more quickly and efficiently than if the entire task was processed on one node. Cluster processing also facilitates deployment of additional hardware resources for larger workloads and rapidly growing user populations.

In Oracle RAC environments, all active instances can concurrently execute transactions against a shared database. Oracle RAC coordinates each instance's access to the shared data, to provide data consistency and data integrity. From a developer's point of view, Oracle RAC enables applications to be scaled to meet increasing data processing demands, without the need to change the application code.

All Oracle E-Business Suite modules can be successfully deployed against a Oracle RAC-enabled database. Using Parallel Concurrent Processing (see Chapter 1), concurrent managers on separate application tier machines can be configured to direct requests to different database servers in an Oracle RAC cluster.

Oracle E-Business Suite supports use of the Oracle Database 11gR2 *Single Client Access Name* (SCAN) feature, which allows all clients to use a single network name (typically defined in DNS) to access an Oracle RAC database. For example, in a non-SCAN two-node cluster the application tier TNS connection descriptor would look something like this:

```
VISION = (DESCRIPTION= (ADDRESS_LIST= (ADDRESS=(PROTOCOL=tcp)(HOST=CNODE1)(PORT=1521)) (ADDRESS=(PROTOCOL=tcp)(HOST=CNODE2)(PORT=1521))) (CONNECT_DATA=(SERVICE_NAME=VISION)))
```

With SCAN, the equivalent descriptor would be:

```
VISION = (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=CSCAN)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=VISION)))
```

Using SCAN means that clients do not have to be modified when the cluster configuration is changed. The feature also facilitates use of load balancing with an Oracle RAC database.

Automatic Storage Management

Automatic Storage Management (ASM) provides a file system and volume manager dedicated to the storage of Oracle database files. It extends the concepts of disk striping and mirroring, to optimize performance and remove the need for manual I/O tuning.

Edition-Based Redefinition

Traditionally, applications such as Oracle E-Business Suite have had to be unavailable while their associated database objects are patched or upgraded. The *Edition-Based Redefinition* (EBR) feature of the Oracle 11gR2 Database allows this downtime to be avoided.

Every database has at least one *edition*, and can support multiple editions. A new edition is created as a child of an existing edition. The changes are made in the child edition, while the parent edition remains available for use. Changes to data (such as when a patch is applied) are made by writing only to new columns or new tables not seen by the old edition.

EBR allows an application's database objects to be changed without interrupting the application's availability. For example, an Oracle E-Business Suite patch can be applied while users are logged in and active. This is accomplished by making the changes in the privacy of a new edition.

Key objects used by EBR include:

- *Editioning views* expose a different projection of a changed table to each edition, allowing each to see only its own columns.
- *Cross-edition triggers* propagate data changes made by the old edition into the columns of the new edition, which subsequently replaces the old edition.

For details of how Oracle E-Business Suite utilizes EBR to support online patching, see the Patching and Management Tools chapter of this book. For a detailed description of how to prepare for and use EBR in online patching, refer to *Oracle E-Business Suite Maintenance Guide*.