CHAPTER 42

The Hitchhiker's Guide to the Oracle Data Dictionary

racle's data dictionary stores all the information that is used to manage the objects in the database. Although the data dictionary is usually the domain of database administrators (DBAs), it also is a source of valuable information for developers and end users.

In this chapter, you will see the data dictionary—Oracle's internal directory—from the perspective of an end user; the data dictionary tables and views are not listed alphabetically, but rather are grouped by function (tables, security, and so on). This organization is designed to let you quickly find your way to the information you need. The most-used data dictionary views are all shown here, along with examples of their usage.

Depending on which Oracle configuration options you are using, some of the groups will not apply to your database. The most commonly used views are listed first. The groupings used in this chapter are, in order:

- New Views Introduced in Oracle Database 10*g*
- New Columns Introduced in Oracle Database 10*g*
- The Road Maps: DICTIONARY (DICT) and DICT_COLUMNS
- Things You Select From: Tables (and Columns), Views, Synonyms, and Sequences
- Constraints and Comments
- Indexes and Clusters
- Abstract Datatypes, ORDBMS-Related Structures, and LOBs
- Database Links and Materialized Views
- Triggers, Procedures, Functions, and Packages
- Dimensions
- Space Allocation and Usage, Including Partitions and Subpartitions
- Users and Privileges
- Roles
- Auditing
- Miscellaneous
- Monitoring: The V$ Dynamic Performance Tables

NOTE
As new features are added, it is expected that the data dictionary will be modified with each future release of Oracle.

A Note about Nomenclature

With some exceptions, the names of the objects in the Oracle data dictionary begin with one of three prefixes: "USER," "ALL," or "DBA." Records in the "USER" views usually show information

about objects owned by the account performing the query. Records in the "ALL" views include the "USER" records plus information about objects on which privileges have been granted to PUBLIC or to the user. "DBA" views encompass all of the database objects, regardless of owner. For most database objects, "USER," "ALL," and "DBA" views are available.

In keeping with the user focus of this guide, the emphasis will be on the "USER" views or those that are accessible to all users. The "ALL" and "DBA" views will be described when they are applicable.

New Views Introduced in Oracle Database 10*g*

For users familiar with the data dictionary in Oracle9*i*, Table 42-1 lists the views that are introduced in Oracle Database 10*g*.

ALL_ Views	DBA_ Views	USER_ Views
	DBA_ADVISOR_ACTIONS	USER_ADVISOR_ACTIONS
	DBA_ADVISOR_COMMANDS	
	DBA_ADVISOR_DEFINITIONS	
	DBA_ADVISOR_FINDINGS	USER_ADVISOR_FINDINGS
	DBA_ADVISOR_JOURNAL	USER_ADVISOR_JOURNAL
	DBA_ADVISOR_LOG	USER_ADVISOR_LOG
	DBA_ADVISOR_OBJECT_TYPES	
	DBA_ADVISOR_OBJECTS	USER_ADVISOR_OBJECTS
	DBA_ADVISOR_PARAMETERS	USER_ADVISOR_PARAMETERS
	DBA_ADVISOR_RATIONALE	USER_ADVISOR_RATIONALE
	DBA_ADVISOR_RECOMMENDATIONS	USER_ADVISOR_RECOMMENDATIONS
	DBA_ADVISOR_SQLA_REC_SUM	USER_ADVISOR_SQLA_REC_SUM
	DBA_ADVISOR_SQLA_WK_MAP	USER_ADVISOR_SQLA_WK_MAP
	DBA_ADVISOR_SQLA_WK_STMTS	USER_ADVISOR_SQLA_WK_STMTS
	DBA_ADVISOR_SQLW_JOURNAL	USER_ADVISOR_SQLW_JOURNAL
	DBA_ADVISOR_SQLW_PARAMETERS	USER_ADVISOR_SQLW_PARAMETERS
	DBA_ADVISOR_SQLW_STMTS	USER_ADVISOR_SQLW_STMTS
	DBA_ADVISOR_SQLW_SUM	USER_ADVISOR_SQLW_SUM
	DBA_ADVISOR_SQLW_TABLES	USER_ADVISOR_SQLW_TABLES
	DBA_ADVISOR_SQLW_TEMPLATES	USER_ADVISOR_SQLW_TEMPLATES
	DBA_ADVISOR_TASKS	USER_ADVISOR_TASKS
	DBA_ADVISOR_TEMPLATES	USER_ADVISOR_TEMPLATES
	DBA_ADVISOR_USAGE	

TABLE 42-1. *New Data Dictionary Views in Oracle Database 10g*

ALL_ Views	DBA_ Views	USER_ Views
	DBA_ALERT_HISTORY	
ALL_APPLY_ENQUEUE	DBA_APPLY_ENQUEUE	
ALL_APPLY_EXECUTE	DBA_APPLY_EXECUTE	
	DBA_APPLY_INSTANTIATED_GLOBAL	
	DBA_APPLY_INSTANTIATED_SCHEMAS	
ALL_APPLY_TABLE_COLUMNS	DBA_APPLY_TABLE_COLUMNS	
ALL_AUDIT_POLICY_COLUMNS	DBA_AUDIT_POLICY_COLUMNS	USER_AUDIT_POLICY_COLUMNS
ALL_AW_PS		
ALL_AWS		
ALL_CAPTURE_EXTRA_ATTRIBUTES	DBA_CAPTURE_EXTRA_ATTRIBUTES	
	DBA_COMMON_AUDIT_TRAIL	
	DBA_DATAPUMP_JOBS	USER_DATAPUMP_JOBS
	DBA_DATAPUMP_SESSIONS	
	DBA_ENABLED_AGGREGATIONS	
	DBA_ENABLED_TRACES	
	DBA_FEATURE_USAGE_STATISTICS	
	DBA_HIGH_WATER_MARK_STATISTICS	
	DBA_HIST_ACTIVE_SESS_HISTORY	
	DBA_HIST_BASELINE	
	DBA_HIST_BG_EVENT_SUMMARY	
	DBA_HIST_BUFFER_POOL_STAT	
	DBA_HIST_CR_BLOCK_SERVER	
	DBA_HIST_CURRENT_BLOCK_SERVER	
	DBA_HIST_DATABASE_INSTANCE	
	DBA_HIST_DATAFILE	
	DBA_HIST_DB_CACHE_ADVICE	
	DBA_HIST_DLM_MISC	
	DBA_HIST_ENQUEUE_STAT	
	DBA_HIST_EVENT_NAME	
	DBA_HIST_FILEMETRIC_HISTORY	
	DBA_HIST_FILESTATXS	
	DBA_HIST_INSTANCE_RECOVERY	
	DBA_HIST_JAVA_POOL_ADVICE	
	DBA_HIST_LATCH	
	DBA_HIST_LATCH_CHILDREN	

TABLE 42-1. *New Data Dictionary Views in Oracle Database 10*g (continued)

ALL_ Views	DBA_ Views	USER_ Views
	DBA_HIST_LATCH_MISSES_SUMMARY	
	DBA_HIST_LATCH_NAME	
	DBA_HIST_LATCH_PARENT	
	DBA_HIST_LIBRARYCACHE	
	DBA_HIST_LOG	
	DBA_HIST_METRIC_NAME	
	DBA_HIST_MTTR_TARGET_ADVICE	
	DBA_HIST_OPTIMIZER_ENV	
	DBA_HIST_OSSTAT	
	DBA_HIST_OSSTAT_NAME	
	DBA_HIST_PARAMETER	
	DBA_HIST_PARAMETER_NAME	
	DBA_HIST_PGA_TARGET_ADVICE	
	DBA_HIST_PGASTAT	
	DBA_HIST_RESOURCE_LIMIT	
	DBA_HIST_ROLLSTAT	
	DBA_HIST_ROWCACHE_SUMMARY	
	DBA_HIST_SEG_STAT	
	DBA_HIST_SEG_STAT_OBJ	
	DBA_HIST_SERVICE_NAME	
	DBA_HIST_SERVICE_STAT	
	DBA_HIST_SERVICE_WAIT_CLASS	
	DBA_HIST_SESSMETRIC_HISTORY	
	DBA_HIST_SGA	
	DBA_HIST_SGASTAT	
	DBA_HIST_SHARED_POOL_ADVICE	
	DBA_HIST_SNAP_ERROR	
	DBA_HIST_SNAPSHOT	
	DBA_HIST_SQL_PLAN	
	DBA_HIST_SQL_SUMMARY	
	DBA_HIST_SQL_WORKAREA_HSTGRM	
	DBA_HIST_SQLBIND	
	DBA_HIST_SQLSTAT	
	DBA_HIST_SQLTEXT	
	DBA_HIST_STAT_NAME	

TABLE 42-1. *New Data Dictionary Views in Oracle Database 10g* (continued)

ALL_ Views	DBA_ Views	USER_ Views
	DBA_HIST_SYS_TIME_MODEL	
	DBA_HIST_SYSMETRIC_HISTORY	
	DBA_HIST_SYSMETRIC_SUMMARY	
	DBA_HIST_SYSSTAT	
	DBA_HIST_SYSTEM_EVENT	
	DBA_HIST_TABLESPACE_STAT	
	DBA_HIST_TBSPC_SPACE_USAGE	
	DBA_HIST_TEMPFILE	
	DBA_HIST_TEMPSTATXS	
	DBA_HIST_THREAD	
	DBA_HIST_UNDOSTAT	
	DBA_HIST_WAITCLASSMET_HISTORY	
	DBA_HIST_WAITSTAT	
	DBA_HIST_WR_CONTROL	
ALL_IND_STATISTICS	DBA_IND_STATISTICS	USER_IND_STATISTICS
ALL_INDEXTYPE_ARRAYTYPES	DBA_INDEXTYPE_ARRAYTYPES	USER_INDEXTYPE_ARRAYTYPES
	DBA_LOGMNR_LOG	
	DBA_LOGMNR_PURGED_LOG	
	DBA_LOGMNR_SESSION	
ALL_MVIEW_COMMENTS	DBA_MVIEW_COMMENTS	USER_MVIEW_COMMENTS
ALL_NESTED_TABLE_COLS	DBA_NESTED_TABLE_COLS	USER_NESTED_TABLE_COLS
	DBA_OUTSTANDING_ALERTS	
ALL_PLSQL_OBJECT_SETTINGS	DBA_PLSQL_OBJECT_SETTINGS	USER_PLSQL_OBJECT_SETTINGS
	DBA_RECYCLEBIN	USER_RECYCLEBIN
	DBA_REDEFINITION_ERRORS	
	DBA_REDEFINITION_OBJECTS	
	DBA_REGISTERED_ARCHIVED_LOG	
ALL_REWRITE_EQUIVALENCES	DBA_REWRITE_EQUIVALENCES	USER_REWRITE_EQUIVALENCES
	DBA_RSRC_GROUP_MAPPINGS	
	DBA_RSRC_MAPPING_PRIORITY	
ALL_SCHEDULER_JOB_ARGS	DBA_SCHEDULER_JOB_ARGS	USER_SCHEDULER_JOB_ARGS
ALL_SCHEDULER_JOB_CLASSES	DBA_SCHEDULER_JOB_CLASSES	
ALL_SCHEDULER_JOB_LOG	DBA_SCHEDULER_JOB_LOG	USER_SCHEDULER_JOB_LOG
ALL_SCHEDULER_JOB_RUN_DETAILS	DBA_SCHEDULER_JOB_RUN_DETAILS	USER_SCHEDULER_JOB_RUN_DETAILS

TABLE 42-1. *New Data Dictionary Views in Oracle Database 10g* (continued)

ALL_ Views	DBA_ Views	USER_ Views
ALL_SCHEDULER_JOBS	DBA_SCHEDULER_JOBS	USER_SCHEDULER_JOBS
ALL_SCHEDULER_PROGRAM_ARGS	DBA_SCHEDULER_PROGRAM_ARGS	USER_SCHEDULER_PROGRAM_ARGS
ALL_SCHEDULER_PROGRAMS	DBA_SCHEDULER_PROGRAMS	USER_SCHEDULER_PROGRAMS
ALL_SCHEDULER_RUNNING_JOBS	DBA_SCHEDULER_RUNNING_JOBS	USER_SCHEDULER_RUNNING_JOBS
ALL_SCHEDULER_SCHEDULES	DBA_SCHEDULER_SCHEDULES	USER_SCHEDULER_SCHEDULES
ALL_SCHEDULER_WINDOW_DETAILS	DBA_SCHEDULER_WINDOW_DETAILS	
ALL_SCHEDULER_WINDOW_GROUPS	DBA_SCHEDULER_WINDOW_GROUPS	
ALL_SCHEDULER_WINDOW_LOG	DBA_SCHEDULER_WINDOW_LOG	
ALL_SCHEDULER_WINDOWS	DBA_SCHEDULER_WINDOWS	
ALL_SCHEDULER_WINGROUP_MEMBERS	DBA_SCHEDULER_WINGROUP_MEMBERS	
ALL_SEC_RELEVANT_COLS	DBA_SEC_RELEVANT_COLS	USER_SEC_RELEVANT_COLS
	DBA_SERVER_REGISTRY	
ALL_SERVICES	DBA_SERVICES	
	DBA_SQL_PROFILES	
	DBA_SQLSET	USER_SQLSET
	DBA_SQLSET_BINDS	USER_SQLSET_BINDS
	DBA_SQLSET_REFERENCES	USER_SQLSET_REFERENCES
	DBA_SQLSET_STATEMENTS	USER_SQLSET_STATEMENTS
	DBA_SQLTUNE_BINDS	USER_SQLTUNE_BINDS
	DBA_SQLTUNE_PLANS	USER_SQLTUNE_PLANS
	DBA_SQLTUNE_RATIONALE_PLAN	USER_SQLTUNE_RATIONALE_PLAN
	DBA_SQLTUNE_STATISTICS	USER_SQLTUNE_STATISTICS
	DBA_STREAMS_ADMINISTRATOR	
ALL_STREAMS_MESSAGE_CONSUMERS	DBA_STREAMS_MESSAGE_CONSUMERS	
ALL_STREAMS_MESSAGE_RULES	DBA_STREAMS_MESSAGE_RULES	
ALL_STREAMS_NEWLY_SUPPORTED	DBA_STREAMS_NEWLY_SUPPORTED	
ALL_STREAMS_RULES	DBA_STREAMS_RULES	
ALL_STREAMS_TRANSFORM_FUNCTION	DBA_STREAMS_TRANSFORM_FUNCTION	
ALL_STREAMS_UNSUPPORTED	DBA_STREAMS_UNSUPPORTED	
ALL_TAB_STATISTICS	DBA_TAB_STATISTICS	USER_TAB_STATISTICS
	DBA_TABLESPACE_GROUPS	
	DBA_THRESHOLDS	
	DBA_TUNE_MVIEW	USER_TUNE_MVIEW
ALL_WARNING_SETTINGS	DBA_WARNING_SETTINGS	USER_WARNING_SETTINGS

TABLE 42-1. *New Data Dictionary Views in Oracle Database 10g* (continued)

Some of the views introduced with Oracle Database 10*g* reflect dramatic functionality additions. For example, USER_RECYCLEBIN (accessible via the RECYCLEBIN public synonym) supports the re-creation of dropped objects. See Chapter 28 for examples of the **flashback table** command. Other features supported by the new views include the enhanced advisor capabilities.

New Columns Introduced in Oracle Database 10*g*

For users familiar with the data dictionary in Oracle9*i*, Table 42-2 lists the columns that are introduced to previously existing data dictionary views in Oracle Database 10*g*.

Static Data Dictionary View	New Columns
ALL_ALL_TABLES	DROPPED
ALL_APPLY	PRECOMMIT_HANDLER MAX_APPLIED_MESSAGE_NUMBER NEGATIVE_RULE_SET_NAME NEGATIVE_RULE_SET_OWNER STATUS_CHANGE_TIME ERROR_NUMBER ERROR_MESSAGE
ALL_APPLY_DML_HANDLERS	APPLY_NAME
ALL_APPLY_PROGRESS	SOURCE_DATABASE
ALL_AUDIT_POLICIES	SEL INS UPD DEL AUDIT_TRAIL POLICY_COLUMN_OPTIONS
ALL_CAPTURE	CAPTURE_USER CAPTURED_SCN APPLIED_SCN USE_DATABASE_LINK FIRST_SCN SOURCE_DATABASE SOURCE_DBID SOURCE_RESETLOGS_SCN SOURCE_RESETLOGS_TIME LOGMINER_ID NEGATIVE_RULE_SET_NAME NEGATIVE_RULE_SET_OWNER MAX_CHECKPOINT_SCN REQUIRED_CHECKPOINT_SCN LOGFILE_ASSIGNMENT STATUS_CHANGE_TIME ERROR_NUMBER ERROR_MESSAGE VERSION CAPTURE_TYPE
ALL_DEF_AUDIT_OPTS	FBK
ALL_DIM_ATTRIBUTES	ATTRIBUTE_NAME

TABLE 42-2. *New Data Dictionary Columns in Oracle Database 10g*

Static Data Dictionary View	New Columns
ALL_ERRORS	ATTRIBUTE MESSAGE_NUMBER
ALL_EXTERNAL_TABLES	PROPERTY
ALL_INDEXES	IOT_REDUNDANT_PKEY_ELIM DROPPED
ALL_INDEXTYPES	ARRAY_DML
ALL_LOBS	TABLESPACE_NAME FORMAT PARTITIONED
ALL_LOG_GROUP_COLUMNS	LOGGING_PROPERTY
ALL_LOG_GROUPS	LOG_GROUP_TYPE GENERATED
ALL_MVIEWS	UNKNOWN_TRUSTED_FD STALE_SINCE
ALL_OBJECT_TABLES	DROPPED
ALL_OUTLINES	COMPATIBLE ENABLED FORMAT
ALL_PART_COL_STATISTICS	HISTOGRAM
ALL_POLICIES	IDX POLICY_TYPE LONG_PREDICATE
ALL_PROPAGATION	NEGATIVE_RULE_SET_OWNER NEGATIVE_RULE_SET_NAME
ALL_SUBPART_COL_STATISTICS	HISTOGRAM
ALL_SUBSCRIBED_COLUMNS	SUBSCRIPTION_NAME
ALL_SUBSCRIBED_TABLES	SUBSCRIPTION_NAME
ALL_SUBSCRIPTIONS	SUBSCRIPTION_NAME
ALL_TAB_COL_STATISTICS	HISTOGRAM
ALL_TAB_COLS	HISTOGRAM QUALIFIED_COL_NAME
ALL_TAB_COLUMNS	HISTOGRAM
ALL_TAB_MODIFICATIONS	DROP_SEGMENTS
ALL_TABLES	DROPPED
DBA_ALL_TABLES	DROPPED
DBA_APPLY	PRECOMMIT_HANDLER MAX_APPLIED_MESSAGE_NUMBER NEGATIVE_RULE_SET_NAME NEGATIVE_RULE_SET_OWNER STATUS_CHANGE_TIME ERROR_NUMBER ERROR_MESSAGE

TABLE 42-2. *New Data Dictionary Columns in Oracle Database 10g* (continued)

Static Data Dictionary View	New Columns
DBA_APPLY_DML_HANDLERS	APPLY_NAME
DBA_APPLY_PROGRESS	SOURCE_DATABASE
DBA_AUDIT_EXISTS	EXTENDED_TIMESTAMP PROXY_SESSIONID GLOBAL_UID INSTANCE_NUMBER OS_PROCESS TRANSACTIONID SCN SQL_BIND SQL_TEXT
DBA_AUDIT_OBJECT	EXTENDED_TIMESTAMP PROXY_SESSIONID GLOBAL_UID INSTANCE_NUMBER OS_PROCESS TRANSACTIONID SCN SQL_BIND SQL_TEXT
DBA_AUDIT_POLICIES	SEL INS UPD DEL AUDIT_TRAIL POLICY_COLUMN_OPTIONS
DBA_AUDIT_SESSION	EXTENDED_TIMESTAMP PROXY_SESSIONID GLOBAL_UID INSTANCE_NUMBER OS_PROCESS
DBA_AUDIT_STATEMENT	EXTENDED_TIMESTAMP PROXY_SESSIONID GLOBAL_UID INSTANCE_NUMBER OS_PROCESS TRANSACTIONID SCN SQL_BIND SQL_TEXT
DBA_AUDIT_TRAIL	EXTENDED_TIMESTAMP PROXY_SESSIONID GLOBAL_UID INSTANCE_NUMBER OS_PROCESS TRANSACTIONID SCN SQL_BIND SQL_TEXT

TABLE 42-2. *New Data Dictionary Columns in Oracle Database 10g* (continued)

Static Data Dictionary View	New Columns
DBA_AW_PS	MAXPAGES
DBA_CAPTURE	CAPTURE_USER USE_DATABASE_LINK FIRST_SCN SOURCE_DATABASE SOURCE_DBID SOURCE_RESETLOGS_SCN SOURCE_RESETLOGS_TIME LOGMINER_ID NEGATIVE_RULE_SET_NAME NEGATIVE_RULE_SET_OWNER MAX_CHECKPOINT_SCN REQUIRED_CHECKPOINT_SCN LOGFILE_ASSIGNMENT STATUS_CHANGE_TIME ERROR_NUMBER ERROR_MESSAGE VERSION CAPTURE_TYPE
DBA_DIM_ATTRIBUTES	ATTRIBUTE_NAME
DBA_ERRORS	ATTRIBUTE MESSAGE_NUMBER
DBA_EXTERNAL_TABLES	PROPERTY
DBA_FGA_AUDIT_TRAIL	STATEMENT_TYPE EXTENDED_TIMESTAMP PROXY_SESSIONID GLOBAL_UID INSTANCE_NUMBER OS_PROCESS TRANSACTIONID STATEMENTID ENTRYID
DBA_INDEXES	IOT_REDUNDANT_PKEY_ELIM DROPPED
DBA_INDEXTYPES	ARRAY_DML
DBA_LOBS	TABLESPACE_NAME FORMAT PARTITIONED
DBA_LOG_GROUP_COLUMNS	LOGGING_PROPERTY
DBA_LOG_GROUPS	LOG_GROUP_TYPE GENERATED
DBA_LOGSTDBY_LOG	APPLIED
DBA_LOGSTDBY_PROGRESS	APPLIED_THREAD# APPLIED_SEQUENCE# READ_THREAD# READ_SEQUENCE# NEWEST_THREAD# NEWEST_SEQUENCE#

TABLE 42-2. *New Data Dictionary Columns in Oracle Database 10g* (continued)

Static Data Dictionary View	New Columns
DBA_LOGSTDBY_SKIP	USE_LIKE ESC
DBA_LOGSTDBY_UNSUPPORTED	ATTRIBUTES TPROP SEGSPARE1 TFLAGS
DBA_MVIEWS	UNKNOWN_TRUSTED_FD STALE_SINCE
DBA_OBJ_AUDIT_OPTS	FBK
DBA_OBJECT_TABLES	DROPPED
DBA_OUTLINES	COMPATIBLE ENABLED FORMAT
DBA_PART_COL_STATISTICS	HISTOGRAM
DBA_POLICIES	IDX POLICY_TYPE LONG_PREDICATE
DBA_PROPAGATION	NEGATIVE_RULE_SET_OWNER NEGATIVE_RULE_SET_NAME
DBA_PROXIES	AUTHENTICATION
DBA_REGISTRY	NAMESPACE
DBA_REGISTRY_HIERARCHY	NAMESPACE
DBA_RSRC_PLAN_DIRECTIVES	MAX_IDLE_TIME MAX_IDLE_BLOCKER_TIME SWITCH_TIME_IN_CALL
DBA_SUBPART_COL_STATISTICS	HISTOGRAM
DBA_SUBSCRIBED_COLUMNS	SUBSCRIPTION_NAME
DBA_SUBSCRIBED_TABLES	SUBSCRIPTION_NAME
DBA_SUBSCRIPTIONS	SUBSCRIPTION_NAME
DBA_TAB_COL_STATISTICS	HISTOGRAM
DBA_TAB_COLS	HISTOGRAM QUALIFIED_COL_NAME
DBA_TAB_COLUMNS	HISTOGRAM
DBA_TAB_MODIFICATIONS	DROP_SEGMENTS
DBA_TABLES	DROPPED
DBA_TABLESPACES	RETENTION BIGFILE
USER_ALL_TABLES	DROPPED

TABLE 42-2. *New Data Dictionary Columns in Oracle Database 10*g (continued)

Static Data Dictionary View	New Columns
USER_AUDIT_OBJECT	EXTENDED_TIMESTAMP PROXY_SESSIONID GLOBAL_UID INSTANCE_NUMBER OS_PROCESS TRANSACTIONID SCN SQL_BIND SQL_TEXT
USER_AUDIT_POLICIES	SEL INS UPD DEL AUDIT_TRAIL POLICY_COLUMN_OPTIONS
USER_AUDIT_SESSION	EXTENDED_TIMESTAMP PROXY_SESSIONID GLOBAL_UID INSTANCE_NUMBER OS_PROCESS
USER_AUDIT_STATEMENT	EXTENDED_TIMESTAMP PROXY_SESSIONID GLOBAL_UID INSTANCE_NUMBER OS_PROCESS TRANSACTIONID SCN SQL_BIND SQL_TEXT
USER_AUDIT_TRAIL	EXTENDED_TIMESTAMP PROXY_SESSIONID GLOBAL_UID INSTANCE_NUMBER OS_PROCESS TRANSACTIONID SCN SQL_BIND SQL_TEXT
USER_AW_PS	MAXPAGES
USER_DIM_ATTRIBUTES	ATTRIBUTE_NAME
USER_ERRORS	ATTRIBUTE MESSAGE_NUMBER
USER_EXTERNAL_TABLES	PROPERTY
USER_INDEXES	IOT_REDUNDANT_PKEY_ELIM DROPPED

TABLE 42-2. *New Data Dictionary Columns in Oracle Database 10g* (continued)

Static Data Dictionary View	New Columns
USER_INDEXTYPES	ARRAY_DML
USER_LOBS	TABLESPACE_NAME FORMAT PARTITIONED
USER_LOG_GROUP_COLUMNS	LOGGING_PROPERTY
USER_LOG_GROUPS	LOG_GROUP_TYPE GENERATED
USER_MVIEWS	UNKNOWN_TRUSTED_FD STALE_SINCE
USER_OBJ_AUDIT_OPTS	FBK
USER_OBJECT_TABLES	DROPPED
USER_OUTLINES	COMPATIBLE ENABLED FORMAT
USER_PART_COL_STATISTICS	HISTOGRAM
USER_POLICIES	IDX POLICY_TYPE LONG_PREDICATE
USER_PROXIES	AUTHENTICATION
USER_PUBLISHED_COLUMNS	CHANGE_SET_NAME PUB_ID
USER_REGISTRY	NAMESPACE
USER_SUBPART_COL_STATISTICS	HISTOGRAM
USER_SUBSCRIBED_COLUMNS	SUBSCRIPTION_NAME
USER_SUBSCRIBED_TABLES	SUBSCRIPTION_NAME
USER_SUBSCRIPTIONS	SUBSCRIPTION_NAME
USER_TAB_COL_STATISTICS	HISTOGRAM
USER_TAB_COLS	HISTOGRAM QUALIFIED_COL_NAME
USER_TAB_COLUMNS	HISTOGRAM
USER_TAB_MODIFICATIONS	DROP_SEGMENTS
USER_TABLES	DROPPED
USER_TABLESPACES	RETENTION BIGFILE

TABLE 42-2. *New Data Dictionary Columns in Oracle Database 10*g (continued)

The Road Maps: DICTIONARY (DICT) and DICT_COLUMNS

Descriptions for the objects that make up the Oracle data dictionary are accessible via a view named DICTIONARY. This view, also accessible via the public synonym DICT, queries the database to determine which data dictionary views you can see. It also searches for public synonyms for those views.

The following example queries DICT for the names of all data dictionary views whose names include the string 'VIEWS'. Selecting from DICT via a non-DBA account, as shown in the following listing, returns the object name and comments for each data dictionary object that matches your search criteria. There are only two columns in this view: Table_Name and the table's associated Comments.

```
column Comments format a35 word_wrapped
column Table_Name format a25

select Table_Name, Comments
  from DICT
 where Table_Name like '%VIEWS%'
 order by Table_Name;

TABLE_NAME                COMMENTS
------------------------- -----------------------------------
ALL_BASE_TABLE_MVIEWS     All materialized views with log(s)
                          in the database that the user can
                          see
ALL_MVIEWS                All materialized views in the
                          database
ALL_REGISTERED_MVIEWS     Remote materialized views of local
                          tables that the user can see
ALL_VIEWS                 Description of views accessible to
                          the user
ALL_XML_VIEWS             Description of all XMLType
                          views that the user has privileges
                          on
USER_BASE_TABLE_MVIEWS    All materialized views with log(s)
                          owned by the user in the database
USER_MVIEWS               All materialized views in the
                          database
USER_REGISTERED_MVIEWS    Remote materialized views of local
                          tables currently using logs owned
                          by the user
USER_VIEWS                Description of the user's own views
USER_XML_VIEWS            Description of the user's own
                          XMLType views
```

You can query the columns of the dictionary views via the DICT_COLUMNS view. Like the DICTIONARY view, DICT_COLUMNS displays the comments that have been entered into the database for the data dictionary views. DICT_COLUMNS has three columns: Table_Name, Column_Name, and Comments. Querying DICT_COLUMNS lets you determine which data dictionary views would be most useful for your needs.

For example, if you want to view space allocation and usage information for your database objects but are not sure which data dictionary views store that information, you can query DICT_COLUMNS, as shown in the following example, which looks for all the dictionary tables that have a column named Blocks:

```
select Table_Name
  from DICT_COLUMNS
 where Column_Name = 'BLOCKS'
   and Table_Name like 'USER%'
 order by Table_Name;

TABLE_NAME
--------------------
USER_ALL_TABLES
USER_EXTENTS
USER_FREE_SPACE
USER_OBJECT_TABLES
USER_SEGMENTS
USER_TABLES
USER_TAB_PARTITIONS
USER_TAB_STATISTICS
USER_TAB_SUBPARTITIONS
USER_TS_QUOTAS
```

To list all of the available column names that you could have used in the last example, query DICT_COLUMNS:

```
select distinct Column_Name
  from DICT_COLUMNS
 order by Column_Name;
```

Whenever you're unsure about where to find the data you want, just check DICTIONARY and DICT_COLUMNS. If it appears that a large number of views could be useful, query DICTIONARY (as in the first example) to see the comments on each view.

Things You Select From: Tables (and Columns), Views, Synonyms, and Sequences

A user's catalog lists all of those objects that the user can **select** records from; that is, any object that can be listed in the **from** clause of a query. Although sequences are not referenced directly in **from** clauses, Oracle includes them in the catalog. In this section, you will see how to retrieve information about tables, columns, views, synonyms, sequences, and the user catalog.

Catalog: USER_CATALOG (CAT)

Querying USER_CATALOG will display all tables, views, synonyms, and sequences the user owns. The Table_Name column shows the name of the object (even if it is not a table), and the Table_Type column shows the type of object:

```
select Table_Name, Table_Type
  from USER_CATALOG
 where Table_Name like 'T%';
```

USER_CATALOG may also be referred to by the public synonym CAT.

Two additional catalogs are available. ALL_CATALOG lists everything in your USER_CATALOG view plus any objects that you or PUBLIC have been granted access to. DBA_CATALOG is a DBA-level version of the catalog showing all tables, views, sequences, and synonyms in the database. In addition to the Table_Name and Table_Type columns shown in the USER_CATALOG query, ALL_CATALOG and DBA_CATALOG include an Owner column.

Objects: USER_OBJECTS (OBJ)

USER_CATALOG only displays information for tables, views, sequences, and synonyms. To retrieve information on all types of objects, query USER_OBJECTS. You can use this view to find out about many types of objects, including clusters, database links, directories, functions, indexes, libraries, packages, package bodies, Java classes, abstract datatypes, resource plans, sequences, synonyms, tables, triggers, materialized views, LOBs, and views. USER_OBJECTS' columns are listed in Table 42-3.

Column Name	Description
Object_Name	The name of the object
SubObject_Name	The name of the subobject, such as a partition name
Object_ID	A unique, Oracle-assigned ID for the object
Data_Object_ID	The object number of the segment that contains the object's data
Object_Type	The object type ('TABLE', 'INDEX', 'TABLE PARTITION', and so on)
Created	The timestamp for the object's creation (a DATE column)
Last_DDL_Time	The timestamp for the last DDL command used on the object, including **alter**, **grant**, and **revoke**
Timestamp	The timestamp for the object's creation (same as Created, but stored as a character column)
Status	The status of the object ('VALID' or 'INVALID')
Temporary	A flag to indicate whether the object is a temporary table
Generated	A flag to indicate whether the object's name was system generated
Secondary	A flag to indicate whether the object is a secondary index created by a domain index

TABLE 42-3. *Columns in USER_OBJECTS*

USER_OBJECTS (also known by its public synonym OBJ) contains several vital pieces of information that are not found in other data dictionary views. It records the creation date of objects (the Created column) and the last time an object was altered (the Last_DDL_Time column). These columns are very useful when trying to reconcile different sets of objects in the same application.

NOTE
If you re-create objects in any way, such as via the Import utility, their Created values will change to the last time they were created.

Two additional object listings are available. ALL_OBJECTS lists everything in your USER_OBJECTS view plus any objects for which access has been granted to you or PUBLIC. DBA_OBJECTS is a DBA-level version of the object listing, showing all of the objects in the database. In addition to the columns shown in the USER_OBJECTS view, ALL_OBJECTS and DBA_OBJECTS include an Owner column.

Tables: USER_TABLES (TABS)

Although all of a user's objects are shown in USER_OBJECTS, few attributes of those objects are shown there. To get more information about an object, you need to look at the view that is specific to that type of object. For tables, that view is USER_TABLES. It can also be referenced via the public synonym TABS.

NOTE
Earlier versions of Oracle included a view called TAB. That view, which is similar in function to TABS, is still supported because of Oracle products that reference it. However, TAB does not contain the columns that TABS contains. Use TABS in your data dictionary queries.

The columns in USER_TABLES can be divided into four major categories (Identification, Space-Related, Statistics-Related, and Other), as shown in Table 42-4.

The table's name is shown in the Table_Name column. The Backed_Up column shows whether or not the table has been backed up since its last modification. The Partitioned column will have a value of 'YES' if the table has been partitioned (data dictionary views related to partitions are described in the "Space Allocation and Usage, Including Partitions and Subpartitions" section of this chapter).

The Space-Related columns, such as Pct_Free, Initial_Extent, Max_Extents, and Pct_Increase, are described in the "Storage" entry of the Alphabetical Reference. The Statistics-Related columns, such as Num_Rows, Blocks, Empty_Blocks, Avg_Row_Len, and Last_Analyzed, are populated when the table is analyzed.

NOTE
As of Oracle Database 10g, you can use the USER_TAB_STATISTICS view to access the statistics for your tables.

Identification	Space-Related	Statistics-Related	Other
Table_Name	Tablespace_Name	Num_Rows	Degree
Backed_Up	Cluster_Name	Blocks	Instances
Partitioned	Pct_Free	Empty_Blocks	Cache
IOT_Name	Pct_Used	Avg_Space	Table_Lock
Logging	Ini_Trans	Chain_Cnt	Buffer_Pool
IOT_Type	Max_Trans	Avg_Row_Len	Row_Movement
Temporary	Initial_Extent	Sample_Size	Duration
Nested	Next_Extent	Last_Analyzed	Skip_Corrupt
Secondary	Min_Extents	Avg_Space_ Freelist_Blocks	Monitoring
	Max_Extents	Num_Freelist_Blocks	Cluster_Owner
	Pct_Increase	Global_Stats	Dependencies
	Freelists	User_Stats	Dropped
	Freelist_Groups		Compression

TABLE 42-4. *Column in USER_TABLES.*

Querying the Table_Name column from USER_TABLES will display the names of all of the tables in the current account. The following query lists all of the tables whose names begin with the letter *L*:

```
select Table_Name from USER_TABLES
 where Table_Name like 'L%';
```

The ALL_TABLES view shows all of the tables owned by the user as well as any to which the user has been granted access (either directly or via grants to PUBLIC). Most third-party reporting tools that list available tables for queries obtain that list by querying ALL_TABLES. Since ALL_TABLES can contain entries for multiple users, it contains an Owner column in addition to the columns in USER_TABLES. DBA_TABLES, which lists all tables in the database, has the same column definitions as ALL_TABLES.

The Degree and Instances columns in USER_TABLES relate to how the table is queried during parallel queries. For information on Degree, Instances, and the other table parameters, see **CREATE TABLE** in the Alphabetical Reference.

NOTE
For external tables, see USER_EXTERNAL_TABLES and USER_EXTERNAL_LOCATIONS.

As of Oracle Database 10g, USER_TABLES and other data dictionary views contain a new column named Dropped. That column will have a value of 'YES' or 'NO'; see the section on the recycle bin for information on descriptions of drop objects.

Columns: USER_TAB_COLUMNS (COLS)

Although users do not query from columns, the data dictionary view that shows columns is closely tied to the data dictionary view of tables. This view, called USER_TAB_COLUMNS, lists information specific to columns. USER_TAB_COLUMNS can also be queried via the public synonym COLS.

The columns you can query from USER_TAB_COLUMNS can be separated into three major categories:

- Identification, such as Table_Name, Column_Name, and Column_ID
- Definition-Related, such as Data_Type, Data_Length, Data_Precision, Data_Scale, Nullable, and Default_Length
- Statistics-Related, such as Num_Distinct Low_Value, High_Value, Density, Num_Nulls, and others

The Table_Name and Column_Name columns contain the names of your tables and columns. The usage of the Definition-Related columns is described in the "Datatypes" entry in the Alphabetical Reference. The Statistics-Related columns are populated when the table is analyzed. Statistics for columns are also provided in the USER_TAB_COL_STATISTICS view (described shortly).

To see the column definitions for a table, query USER_TAB_COLUMNS, specifying the Table_Name in the **where** clause:

```
select Column_Name, Data_Type
  from USER_TAB_COLUMNS
 where Table_Name = 'NEWSPAPER';

COLUMN_NAME    DATA_TYPE
-------------- ---------
FEATURE        VARCHAR2
SECTION        CHAR
PAGE           NUMBER
```

The information in this example is also obtainable via the SQL*Plus **describe** command; however, **describe** does not give you the option of seeing the column defaults and statistics. The ALL_TAB_COLUMNS view shows the columns for all of the tables and views owned by the user as well as any to which the user has been granted access (either via direct grants or via grants to PUBLIC). Since ALL_TAB_COLUMNS can contain entries for multiple users, it contains an Owner column in addition to the columns in USER_TAB_COLUMNS. DBA_TAB_COLUMNS, which lists the column definitions for all tables and views in the database, has the same column definitions as ALL_TAB_COLUMNS.

Column Statistics

Most of the column statistics are available in both USER_TAB_COLUMNS (their former home) and USER_TAB_COL_STATISTICS. The columns available in USER_TAB_COL_STATISTICS are

the Statistics-Related columns of USER_TAB_COLUMNS plus the Table_Name and Column_Name columns.

USER_TAB_COL_STATISTICS contains statistics columns that are also provided via USER_TAB_COLUMNS for backward compatibility. You should access them via USER_TAB_COL_STATISTICS.

Column Value Histograms

You can use histograms to improve the analysis used by the cost-based optimizer. The USER_TAB_HISTOGRAMS view contains information about each column's histogram: the Table_Name, Column_Name, Endpoint_Number, Endpoint_Value, and Endpoint_Actual_Value. The values in USER_TAB_HISTOGRAMS are used by the optimizer to determine the distribution of the column values within the table. "ALL" and "DBA" versions of USER_TAB_HISTOGRAMS are available.

Updatable Columns

You can update records in views that contain joins in their view queries, provided the join meets certain criteria. The USER_UPDATABLE_COLUMNS view lists all columns you can update. You can query the Owner, Table_Name, and Column_Name for the column; the Updatable column will have a value of 'YES' if the column can be updated, and a value of 'NO' if the column cannot be updated. You can also query to determine whether you can insert or delete records from the view via the Insertable and Deletable columns.

Views: USER_VIEWS

The base query of a view is accessible via the USER_VIEWS data dictionary view, which contains ten columns; the three main columns are as follows:

View_Name	The name of the view
Text_Length	The length of the view's base query, in characters
Text	The query that the view uses

The other columns, described later in this section, are related to object views.

NOTE
This section only applies to traditional views. For materialized views, see "Database Links and Materialized Views" later in the chapter.

The Text column is a LONG datatype. This may cause a problem when querying USER_VIEWS via SQL*Plus, because SQL*Plus truncates LONGs. The point at which truncation occurs, however, can be changed via the **set long** command. USER_VIEWS provides a mechanism for determining the proper setting for the LONG truncation point, as you'll see in the following example.

The Text_Length column shows the length of the view's query. Therefore, the SQL*Plus LONG truncation point must be set to a value equal to or greater than the view's Text_Length value. For example, the following listing shows a view whose View_Name is AGING and whose Text_Length is 355:

```
select View_Name, Text_Length
  from USER_VIEWS
 where View_Name = 'AGING';
```

```
View_Name        Text_Length
---------------- -------------
AGING            355
```

Since the length of the view's text is 355 characters, use the **set long** command to increase the LONG truncation point to at least 355 (the default is 80) to see the full text of the view's query:

```
set long 355
```

You can then query USER_VIEWS for the view's Text, using the query shown in the following listing:

```
select Text
  from USER_VIEWS
 where View_Name = 'AGING';
```

If you had not used the **set long** command, then the output would have been truncated at 80 characters, with no message telling you why the truncation occurred. Before querying other views, you will need to recheck their Text_Length values.

NOTE
You can query the column definitions for views from USER_TAB_COLUMNS, the same view you query for tables.

If you use column aliases in your views, and your column aliases are part of the view's query, then your data dictionary queries for information about views will be simplified. Since the entire text of the view's query is displayed in USER_VIEWS, the column aliases will be displayed as well.

You can create views using this format:

```
create view NEWSPAPER_VIEW (SomeFeature, SomeSection)
    as select Feature, Section
  from NEWSPAPER;
```

Listing the column names in the header of the **create view** command removes the column aliases from the query and thus prevents you from seeing them via USER_VIEWS. The only way to see the view's column names would be to query USER_TAB_COLUMNS. If the column names are in the query, then you only need to query one data dictionary view (USER_VIEWS) for both the query and the column names.

For example, given the NEWSPAPER_VIEW view created in the last example, if you were to query USER_VIEWS, you would see

```
select Text
  from USER_VIEWS
 where View_Name ='NEWSPAPER_VIEW';

TEXT
---------------------------------------
select Feature, Section from NEWSPAPER
```

This query does *not* show you the new column names you assigned, since you did not make those column names part of the view's query. To make those column names show up in USER_VIEWS, add them as column aliases in the view's query:

```
create view NEWSPAPER_VIEW
    as select Feature SomeFeature, Section SomeSection
  from NEWSPAPER;
```

Now when you query USER_VIEWS, the column aliases will be displayed as part of the view's query text:

```
select Text
  from USER_VIEWS
 where View_Name ='NEWSPAPER_VIEW';

TEXT
--------------------------------------------
select Feature SomeFeature, Section SomeSection
  from NEWSPAPER
```

To support object views, USER_VIEWS contains the following columns:

Type_Text	Type clause of the typed view
Type_Text_Length	Length of the type clause of the typed view
OID_Text	WITH OID clause of the typed view
OID_Text_Length	Length of the WITH OID clause of the typed view
View_Type_Owner	Owner of the view's type for typed views
View_Type	Type of the view

See Chapters 33 and 36 for information on object views and types.

The ALL_VIEWS view lists all of the views owned by the user as well as any views to which the user has been granted access, either via direct grants or via grants to PUBLIC. Since ALL_VIEWS can contain entries for multiple users, it contains an Owner column in addition to the columns listed earlier in this section. DBA_VIEWS, which lists all views in the database, has the same column definitions as ALL_VIEWS.

Synonyms: USER_SYNONYMS (SYN)

USER_SYNONYMS lists all of the synonyms that a user owns. The columns are as follows:

Synonym_Name	The name of the synonym
Table_Owner	The owner of the table that the synonym refers to
Table_Name	The name of the table that the synonym refers to
DB_Link	The name of the database link used in the synonym

USER_SYNONYMS is useful when debugging programs or resolving problems with users' access to objects within applications. USER_SYNONYMS is also known by the public synonym SYN.

The DB_Link column will be **NULL** if the synonym does not use a database link. Therefore, if you want to see a list of the database links currently in use by the synonyms owned by your account, execute this query:

```
select distinct DB_Link
  from USER_SYNONYMS
 where DB_Link is not null;
```

The ALL_SYNONYMS view lists all of the synonyms owned by the user, public synonyms, and all synonyms to which the user has been granted access. Since ALL_SYNONYMS can contain entries for multiple users, it contains an Owner column in addition to the columns listed earlier in this section. DBA_SYNONYMS, which lists all synonyms in the database, has the same column definitions as ALL_SYNONYMS.

Sequences: USER_SEQUENCES (SEQ)

To display the attributes of sequences, you can query the USER_SEQUENCES data dictionary view. This view can also be queried using the public synonym SEQ. The columns of USER_SEQUENCES are as follows:

Column Name	Description
Sequence_Name	Name of the sequence
Min_Value	Minimum value of the sequence
Max_Value	Maximum value of the sequence
Increment_By	Increment between sequence values
Cycle_Flag	A flag to indicate whether the value should cycle back to the Min_Value once the Max_Value is reached
Order_Flag	A flag to indicate whether sequence numbers are generated in order
Cache_Size	The number of sequence entries cached in memory
Last_Number	The last sequence number written to disk or cached

The Last_Number column is not updated during normal database operation; it is used during database restart/recovery operations.

ALL_SEQUENCES lists all of the sequences owned by the user or to which the user or PUBLIC has been granted access. Since ALL_SEQUENCES can contain entries for multiple users, it contains a Sequence_Owner column in addition to the columns listed. DBA_SEQUENCES, which lists all sequences in the database, has the same column definitions as ALL_SEQUENCES.

Recycle Bin—USER_RECYCLEBIN and DBA_RECYCLEBIN

As of Oracle Database 10*g* you can use the **flashback table** command to recover tables and dependent objects that have been dropped. To see the objects presently in your recycle bin, query USER_RECYCLEBIN. The public synonym RECYCLEBIN can be used in place of USER_RECYCLEBIN in your queries. You will see the original object name, whether it can be recovered (the Can_Undrop column), and whether it was dropped because it was related to a base object that was dropped (the Base_Object column value).

Objects stay in the recycle bin until they are purged. Objects can be removed from the recycle bin automatically if there is no longer any free space in the tablespace they were dropped from. DBAs can see all objects in all users' recycle bins via the DBA_RECYCLEBIN view. There is no "ALL" version of this view. See the entry for the **purge** command for details on removing objects from the recycle bin.

Constraints and Comments

Constraints and comments help you to understand how tables and columns relate to each other. Comments are strictly informational; they do not enforce any conditions on the data stored in the objects they describe. Constraints, on the other hand, define the conditions under which that data is valid. Typical constraints include NOT NULL, UNIQUE, PRIMARY KEY, and FOREIGN KEY. In the following sections, you will see how to retrieve data about constraints and comments from the data dictionary.

Constraints: USER_CONSTRAINTS

Constraint information is accessible via the USER_CONSTRAINTS view. This information is very useful when trying to alter data constraints or resolve problems with an application's data. The columns of this view are listed in Table 42-5.

Column Name	Description
Owner	The owner of the constraint.
Constraint_Name	The name of the constraint.
Constraint_Type	The type of constraint: 'C' CHECK constraint; includes NOT NULLs 'P' PRIMARY KEY constraint 'R' FOREIGN KEY (reference) constraint 'U' UNIQUE constraint 'V' WITH CHECK OPTION constraint (for views) 'O' WITH READ ONLY constraint (for views)
Table_Name	The name of the table associated with the constraint.

TABLE 42-5. *Columns in USER_CONSTRAINTS*

Column Name	Description
Search_Condition	The search condition used (for CHECK constraints).
R_Owner	The owner of the table referenced by a FOREIGN KEY constraint.
R_Constraint_Name	The name of the constraint referenced by a FOREIGN KEY constraint.
Delete_Rule	The action to take on the FOREIGN KEY tables when a PRIMARY KEY record is deleted (CASCADE or NO ACTION).
Status	The status of the constraint (ENABLED or DISABLED).
Deferrable	A flag to indicate whether the constraint can be deferred.
Deferred	A flag to indicate whether the constraint was initially deferred.
Validated	A flag (VALIDATED or NOT VALIDATED) to indicate whether all data obeys the constraint.
Generated	A flag to indicate whether the constraint name has been generated by the database.
Bad	A flag to indicate whether a date was used in the constraint creation without specifying a century value for CHECK constraints, resulting in ambiguous two-digit years; applies only to constraints in databases upgraded from prior versions of Oracle. Such constraints may result in ORA-02436 errors if not dropped and re-created properly.
Rely	A flag to indicate if the constraint is enforced or unenforced.
Last_Change	The date the constraint was last enabled or disabled.
Index_Owner	Owner of the related index.
Index_Name	Name of the related index.
Invalid	Valid/invalid flag.
View_Related	Yes/No flag to record if the constraint is view related.

TABLE 42-5. *Columns in USER_CONSTRAINTS* (continued)

Although this is a "USER" view, it contains an Owner column. In this view, Owner refers to the owner of the constraint, not the owner of the table (which is the user).

Valid values for the Constraint_Type column are shown in Table 42-5. Understanding the constraint types is crucial when you are trying to get useful information about your constraints.

FOREIGN KEY constraints will always have values for the R_Owner and R_Constraint_Name columns. These two columns will tell you which constraint the FOREIGN KEY references. A FOREIGN KEY references another *constraint,* not another column. NOT NULL constraints on columns are stored as CHECK constraints, so they have a Constraint_Type of 'C'.

Querying USER_CONSTRAINTS will give you the names of all the constraints on a table. This is important when trying to interpret error messages that only provide the name of the constraint that was violated.

Once you know the name and type of a constraint, you can check the columns associated with it via the USER_CONS_COLUMNS view, described in the next section.

If you do not assign constraint names when creating them, Oracle will generate unique constraint names. See "Integrity Constraint" in the Alphabetical Reference for further details. If a constraint name has been system-generated, that fact will be indicated via the Generated column.

If you defer a constraint (as indicated by the Deferred column), the constraint will not be enforced for the duration of a transaction. For example, if you were performing bulk **update**s on related tables and could not guarantee the order of transactions, you may decide to defer constraint checking on the tables until all of the **update**s have completed.

ALL_CONSTRAINTS lists the constraints on all of the tables that the user or PUBLIC can access. DBA_CONSTRAINTS lists all of the constraints in the database. Each of these views has the same set of columns as USER_CONSTRAINTS (since they all include the Owner column).

Constraint Columns: USER_CONS_COLUMNS

You can view the columns associated with constraints via the USER_CONS_COLUMNS data dictionary view. If you have already queried USER_CONSTRAINTS to obtain the types and names of the constraints involved, you can use USER_CONS_COLUMNS to determine which columns are involved in the constraint. The columns in this view are the following:

Owner	The owner of the constraint
Constraint_Name	The name of the constraint
Table_Name	The name of the table associated with the constraint
Column_Name	The name of the column associated with the constraint
Position	The order of the column within the constraint definition

There are only two columns in USER_CONS_COLUMNS that are not in USER_CONSTRAINTS: Column_Name and Position. A sample query of this table is shown in the following listing:

```
select Column_Name, Position
  from USER_CONS_COLUMNS
 where Constraint_Name = 'SYS_C0008791';

COLUMN_NAME  POSITION
------------ --------
FIRSTNAME           1
LASTNAME            2
```

As shown in the preceding listing, the combination of FirstName and LastName forms the constraint (in this case, a primary key).

The Position column is significant. When you create a UNIQUE or PRIMARY KEY constraint, Oracle automatically creates a unique index on the set of columns you specify. The index is created based on the column order you specify. The column order, in turn, affects the performance of the index. An index comprising multiple columns may be used most efficiently if the leading column

of that index (Position=1) is used in the query's **where** clause. See Chapter 43 for further details on the optimizer.

The ALL_CONS_COLUMNS and DBA_CONS_COLUMNS views have the same column definitions as USER_CONS_COLUMNS. ALL_CONS_COLUMNS can be used to display column information about constraints on all tables that the user can access regardless of owner. DBA_CONS_COLUMNS lists the column-level constraint information for the entire database.

Constraint Exceptions: EXCEPTIONS

When enabling constraints on tables that already contain data, you may encounter constraint violations within the data. For example, you may attempt to create a PRIMARY KEY constraint on a column that contains the same value for multiple records, but such an attempt would fail due to uniqueness constraint violations.

You can capture information about the rows that cause constraint creations to fail. First, create a table called EXCEPTIONS in your schema; the SQL script that should be used to create this table is named utlexcpt.sql, and is usually located in the /rdbms/admin directory under the Oracle home directory.

The EXCEPTIONS table contains four columns: Row_ID (the ROWID of each row that violated the constraint), Owner (the owner of the violated constraint), Table_Name (the table on which the violated constraint was created), and Constraint (the constraint violated by the row). After creating the EXCEPTIONS table, attempt to enable a PRIMARY KEY constraint:

```
alter table NEWSPAPER enable PRIMARY KEY
exceptions into EXCEPTIONS;

ORA-02437: cannot enable (NEWSPAPER.SYS_C00516) - primary key violated
```

The constraint creation fails, and a reference to all rows that violate the constraint is placed in the EXCEPTIONS table. For example, if the PRIMARY KEY constraint in the last example generated exceptions, then you could query the EXCEPTIONS table as shown in the following listing:

```
select Owner, Table_Name, Constraint from EXCEPTIONS;

OWNER      TABLE_NAME CONSTRAINT
---------- ---------- ----------
PRACTICE   NEWSPAPER  SYS_C00516
PRACTICE   NEWSPAPER  SYS_C00516
```

Two rows violated the constraint named SYS_C00516 (which, in this example, is the constraint name given to the PRIMARY KEY constraint for the NEWSPAPER table). You can determine which rows of the NEWSPAPER table correspond to these exceptions by joining the Row_ID column of the EXCEPTIONS table to the ROWID pseudo-column of the table on which the constraint was being placed (in this example, the NEWSPAPER table):

```
select *
  from NEWSPAPER
 where RowID in
       (select Row_ID from EXCEPTIONS);
```

The rows that caused the constraint violations will then be displayed.

NOTE
Row_ID is an actual column in the EXCEPTIONS table, with a datatype of ROWID. It is joined to the ROWID pseudo-column in the table from which the exceptions were generated.

NOTE
Flashback and recovery operations may change the ROWIDs for rows previously inserted into tables.

Table Comments: USER_TAB_COMMENTS

You can add a comment to a table, view, or column after it has been created. Comments on the data dictionary views are the basis for the records displayed via the DICTIONARY and DICT_COLUMNS views. To display comments about your own tables, use the USER_TAB_COMMENTS view.

USER_TAB_COMMENTS contains three columns:

Table_Name	The name of the table or view
Table_Type	The object type (table, object table, or view)
Comments	Comments that have been entered for the object

To add a comment to a table, use the **comment** command, as shown in the following listing:

```
comment on table BIRTHDAY is 'Birthday list for Blacksburg
      employees';
```

Query USER_TAB_COMMENTS by specifying the Table_Name you want to see the comments for, as shown in the following listing:

```
select Comments
  from USER_TAB_COMMENTS
 where Table_Name = 'BIRTHDAY';

COMMENTS
-------------------------------------
Birthday list for Blacksburg
      employees
```

To remove a comment, set the comment to two single quotes with no space between them:

```
comment on table BIRTHDAY is '';
```

You can view the comments on all of the tables you can access via the ALL_TAB_COMMENTS view. ALL_TAB_COMMENTS has an additional column, Owner, which specifies the owner of the table. DBA_TAB_COMMENTS lists all of the tables in the database, and it has an Owner column as well.

Column Comments: USER_COL_COMMENTS

USER_COL_COMMENTS displays the comments that have been entered for columns within your tables. These comments are added to the database via the **comment** command. USER_COL_COMMENTS contains three columns:

Table_Name	The name of the table or view
Column_Name	The name of the column
Comments	Comments that have been entered for the column

To add a comment to a column, use the **comment** command, as shown in the following listing:

```
comment on column BIRTHDAY.AGE is 'Age in years';
```

Query USER_COL_COMMENTS by specifying the Table_Name and Column_Name you want to see the comments for.

```
select Comments
  from USER_COL_COMMENTS
 where Table_Name = 'BIRTHDAY'
   and Column_Name = 'AGE';
```

To remove a comment, set the comment to two single quotes with no space between them:

```
comment on column BIRTHDAY.AGE is '';
```

You can view the column comments on all of the tables you can access via the ALL_COL_COMMENTS view. ALL_COL_COMMENTS has an additional column, Owner, which specifies the owner of the table. DBA_COL_COMMENTS lists all of the columns in all of the tables in the database, and it has an Owner column as well.

Indexes and Clusters

Indexes and clusters do not change the data that is stored in tables; however, they do change the way that data is accessed and stored. In the following sections, you will see data dictionary views that describe indexes and clusters. You will see descriptions of the data dictionary views related to partitioned indexes in "Space Allocation and Usage, Including Partitions and Subpartitions" in this chapter.

Indexes: USER_INDEXES (IND)

In Oracle, indexes are very closely related to constraints. The PRIMARY KEY and UNIQUE constraints always have associated unique indexes. As with constraints, two data dictionary views are used to query information about indexes: USER_INDEXES and USER_IND_COLUMNS. USER_INDEXES is also known by its public synonym, IND.

The columns in USER_INDEXES can be grouped into four categories, as shown in Table 42-6.

The name of the index is shown in the Index_Name column. The owner and name for the table being indexed are in the Table_Owner and Table_Name columns. The Uniqueness column will

Identification	Space-Related	Statistics-Related	Other
Index_Name	Tablespace_Name	Blevel	Degree
Table_Owner	Ini_Trans	Leaf_Blocks	Instances
Table_Name	Max_Trans	Distinct_Keys	Include_Column
Table_Type	Initial_Extent	Avg_Leaf_Blocks_Per_Key	Buffer_Pool
Uniqueness	Next_Extent	Avg_Data_Blocks_Per_Key	Duration
Status	Min_Extents	Clustering_Factor	Pct_Direct_Access
Partitioned	Max_Extents	Num_Rows	Parameters
Index_Type	Pct_Increase	Sample_Size	Domidx_Status
Temporary	Pct_Free	Last_Analyzed	Domidx_Opstatus
Generated	Freelists	User_Stats	Funcidx_Status
Logging	Freelist_Groups	Global_Stats	Join_Index
Compression	Pct_Threshold		IOT_Redundant_Pkey_Elim
Prefix_Length			Dropped
Secondary			
Ityp_Owner			
Ityp_Name			

TABLE 42-6. *Columns in USER_INDEXES*

be set to UNIQUE for unique indexes and set to NONUNIQUE for nonunique indexes. Table_Type records whether the index is on a 'TABLE' or a 'CLUSTER'. The Dropped column, available as of Oracle Database 10*g*, identifies indexes that are in the recycle bin.

The Space-Related columns are described in the "Storage" entry in the Alphabetical Reference.

The Statistics-Related columns are populated when the table is analyzed (see the **analyze** command in the Alphabetical Reference). The Degree and Instances columns refer to the degree of parallelism used when the index is created.

NOTE
As of Oracle Database 10g, you can use the USER_IND_STATISTICS view to access the statistics for your indexes.To see all of the indexes on a table, query USER_INDEXES using the Table_Owner and Table_Name columns in the ***where*** *clause, as shown in the following listing:*

```
select Index_Name, Uniqueness
  from USER_INDEXES
 where Table_Owner = 'PRACTICE'
   and Table_Name = 'BIRTHDAY';

INDEX_NAME       UNIQUENES
---------------- ---------
PK_BIRTHDAY      UNIQUE
```

The Clustering_Factor column is not directly related to clusters, but rather represents the degree to which the rows in the table are ordered. The more ordered those rows are, the more efficient range queries will be (*range queries* are those in which a range of values is given for a column, such as **where LastName like 'A%'**).

To discover which columns are part of the index, and their order within the index, you need to query the USER_IND_COLUMNS view, which is described in the next section.

ALL_INDEXES shows all of the indexes owned by the user as well as any created on tables to which the user or PUBLIC has been granted access. Since ALL_INDEXES can contain entries for multiple users, it contains an Owner column in addition to the columns shown in Table 42-6. DBA_INDEXES, which lists all indexes in the database, has the same column definitions as ALL_INDEXES.

NOTE
For function-based indexes, see USER_IND_EXPRESSIONS.

Indexed Columns: USER_IND_COLUMNS

You can determine which columns are in an index by querying USER_IND_COLUMNS. The columns available via this view are as follows:

Index_Name	The name of the index
Table_Name	The name of the indexed table
Column_Name	The name of a column within the index
Column_Position	The column's position within the index
Column_Length	The indexed length of the column
Char_Length	Maximum code point length of the column (Unicode)
Descend	A Y/N flag to indicate whether or not the column is sorted in descending order

Five columns in this view are not in USER_INDEXES: Column_Name, Column_Position, Column_Length, Char_Length, and Descend. Column_Length, like the Statistics-Related columns in USER_INDEXES, is populated when the index's base table is analyzed. A sample query of this table, using the Index_Name from the USER_INDEXES example, is shown in the following listing (in this example, the Column_Position column is given the alias Pos):

```
select Column_Name, Column_Position Pos
  from USER_IND_COLUMNS
 where Index_Name = 'PK_BIRTHDAY';

COLUMN_NAME  POS
------------ ---
FIRSTNAME      1
LASTNAME       2
```

As you can see from this query, the PK_BIRTHDAY index consists of two columns: FirstName and LastName, in that order. See Chapter 43 for details on how the optimizer uses multicolumn indexes.

ALL_IND_COLUMNS can be used to display column information about indexes on all tables that the user can access regardless of owner. DBA_IND_COLUMNS lists the column-level index information for the entire database.

Bitmap Join Index Columns: USER_JOIN_IND_COLUMNS

If you create a bitmap join index (introduced in Oracle9*i*), you can query USER_JOIN_IND_COLUMNS for the join details. USER_JOIN_IND_COLUMNS records the names of the tables involved in the join, the inner and outer join columns, and the dimension and fact tables involved.

Clusters: USER_CLUSTERS (CLU)

The storage and statistics parameters associated with clusters are accessible via USER_CLUSTERS (also known by the public synonym CLU). The columns in this data dictionary view are shown in Table 42-7, separated by type.

The Cluster_Name column contains the name of the cluster. Cluster_Type specifies whether the cluster uses a standard B*-tree index or a hashing function for the cluster.

The usage of the Space-Related columns is described in the "Storage" entry in the Alphabetical Reference. The Statistics-Related columns are populated when the table is analyzed.

ALL_CLUSTERS and DBA_CLUSTERS have an additional column, Owner, since they list clusters in multiple schemas.

Identification	Space-Related	Statistics-Related	Other
Cluster_Name	Tablespace_Name	Avg_Blocks_Per_Key	Degree
Cluster_Type	Pct_Free	Hashkeys	Instances
Function	Pct_Used		Cache
	Key_Size		Buffer_Pool
	Ini_Trans		Single_Table
	Max_Trans		Dependencies
	Initial_Extent		
	Next_Extent		
	Min_Extents		
	Max_Extents		
	Pct_Increase		
	Freelists		
	Freelist_Groups		

TABLE 42-7. *Columns in USER_CLUSTERS*

Cluster Columns: USER_CLU_COLUMNS

To see the mapping of table columns to cluster columns, query USER_CLU_COLUMNS, whose columns are the following:

Cluster_Name	The name of the cluster
Clu_Column_Name	The name of the key column in the cluster
Table_Name	The name of a table within the cluster
Tab_Column_Name	The name of the key column in the table

Since a single cluster can store data from multiple tables, USER_CLU_COLUMNS is useful for determining which columns of which tables map to the cluster's columns.

There is no "ALL" version of this view. There are only USER_CLU_COLUMNS for the user's cluster columns, and DBA_CLU_COLUMNS, which shows the column mappings for all clusters in the database.

Abstract Datatypes, ORDBMS-Related Structures, and LOBs

In the following sections, you will see the data dictionary views associated with object-relational structures such as abstract datatypes, methods, and large objects (LOBs). "ALL" and "DBA" versions of all of these data dictionary views are available. Because they contain records for multiple owners, the "ALL" and "DBA" versions of these views contain Owner columns as well as the columns listed in this section.

Abstract Datatypes: USER_TYPES

Abstract datatypes created within your schema are listed in USER_TYPES, which includes columns for the Type_Name, number of attributes (Attributes), and number of methods (Methods) defined for the datatype.

For example, the ANIMAL_TY datatype has three attributes and one method (methods are defined via the **create type body** command), as shown in the following listing:

```
select Type_Name,
       Attributes,
       Methods
  from USER_TYPES
 where Type_Name = 'ANIMAL_TY';

TYPE_NAME                      ATTRIBUTES    METHODS
------------------------------ ---------- ----------
ANIMAL_TY                               3          1
```

Datatype Attributes: USER_TYPE_ATTRS

To see the attributes of a datatype, you need to query the USER_TYPE_ATTRS view. The columns in USER_TYPE_ATTRS are shown in Table 42-8.

Column Name	Description
Type_Name	Name of the type
Attr_Name	Name of the attribute
Attr_Type_Mod	The type modifier of the attribute
Attr_Type_Owner	Owner of the attribute's type, if the attribute is based on another datatype
Attr_Type_Name	Name of the type of the attribute
Length	Length of the attribute
Precision	Precision of the attribute
Scale	Scale of the attribute
Character_Set_Name	Character set of the attribute
Attr_No	Ordinal position of the attribute within the type definition
Inherited	Yes/No flag to indicate if the attribute is inherited from a supertype

TABLE 42-8. *Columns in USER_TYPE_ATTRS*

You can query USER_TYPE_ATTRS to see the relationships between nested abstract datatypes. For example, the PERSON_TY datatype uses the ADDRESS_TY datatype, as shown in the following listing:

```
select Attr_Name,
       Length,
       Attr_Type_Name
  from USER_TYPE_ATTRS
 where Type_Name = 'PERSON_TY';

ATTR_NAME                       LENGTH ATTR_TYPE_NAME
------------------------------ ---------- ------------------
NAME                                   25 VARCHAR2
ADDRESS                                   ADDRESS_TY
```

Datatype Methods: USER_TYPE_METHODS and USER_METHOD_PARAMS

If a type has methods defined for it, then you query USER_TYPE_METHODS to determine the methods' names. USER_TYPE_METHODS contains columns showing the type name (Type_Name), method name (Method_Name), method number (Method_No, used for overloaded methods), and type of method (Method_Type). USER_TYPE_METHODS also includes columns that show the number of parameters (Parameters) and results (Results) returned by the method.

For example, the ANIMAL_TY datatype has one method, a member function named AGE. The AGE method has one input parameter (Birthdate) and one output result (the age, in days).

Querying USER_TYPE_METHODS shows that *two* parameters are defined for AGE, not one as expected:

```
select Parameters,
       Results
  from USER_TYPE_METHODS
 where Type_Name = 'ANIMAL_TY'
   and Method_Name = 'AGE';

PARAMETERS    RESULTS
---------- ----------
         2          1
```

Why does AGE have two parameters if it only has one input parameter defined? To find out, you can query USER_METHOD_PARAMS, which describes the parameters for your methods:

```
select Param_Name, Param_No, Param_Type_Name
  from USER_METHOD_PARAMS
 order by Param_No;

PARAM_NAME                      PARAM_NO PARAM_TYPE_NAME
------------------------------ ---------- -----------------
SELF                                    1 ANIMAL_TY
BIRTHDATE                               2 DATE
```

USER_METHOD_PARAMS shows that for each method, Oracle creates an implicit SELF parameter. The results of your methods are shown in USER_METHOD_RESULTS:

```
select Method_Name,
       Result_Type_Name
  from USER_METHOD_RESULTS
 where Type_Name = 'ANIMAL_TY';

METHOD_NAME                    RESULT_TYPE_NAME
------------------------------ ------------------------------
AGE                            NUMBER
```

Other Datatypes: USER_REFS, USER_COLL_TYPES, and USER_NESTED_TABLES

If you use REFs (see Chapter 36), you can query the USER_REFS data dictionary view to display the REFs you have defined. USER_REFS will show the name of the table containing the REF column (Table_Name) and the column name of the object column (Column_Name). The attributes of the REF—such as whether or not it is stored with the ROWID—are also accessible via USER_REFS.

Collector types (nested tables and varying arrays) are described via the USER_COLL_TYPES data dictionary view. The columns of USER_COLL_TYPES include Type_Name, Upper_Bound (for varying arrays), and the Length and Precision of the elements. You can use USER_COLL_TYPES in conjunction with the abstract datatype data dictionary views shown earlier in this section to determine the type structure of a collector. You can also query USER_NESTED_TABLES and USER_VARRAYS to see details for your collections.

LOBs: USER_LOBS

As described in Chapter 35, you can store large objects inside the database. The USER_LOBS view provides information on the LOBs defined in your tables; a sample listing is shown here:

```
column column_name format A30

select Table_Name,
       Column_Name
  from USER_LOBS;

TABLE_NAME                     COLUMN_NAME
------------------------------ ------------------------------
PROPOSAL                       PROPOSAL_TEXT
PROPOSAL                       BUDGET
```

USER_LOBS also shows the names of the segments used to hold the LOB data when it grows large; however, it does not show the datatypes of the LOB columns. To see the LOB datatypes, you can either **describe** the table that includes the LOBs or query USER_TAB_COLUMNS (as shown earlier in this chapter).

To use BFILE datatypes for LOBs, you have to create directories (see the entry for the **create directory** command in the Alphabetical Reference). The ALL_DIRECTORIES data dictionary shows the Owner, Directory_Name, and Directory_Path columns for each directory to which you have been granted access. A "DBA" version of this view is also available. No "USER" version of this view is available.

Database Links and Materialized Views

Database links and materialized views are used to manage access to remote data. Depending on the type of materialized view you use, you may be able to use materialized view logs. In the following sections, you will see descriptions of the data dictionary views you can use to display information about database links and materialized views. For further information on database links, see Chapter 23. For further information on materialized views, see Chapter 24.

Database Links: USER_DB_LINKS

To see the database links created under your account, query USER_DB_LINKS. The columns of this view, including the link name (DB_Link), username to connect to (Username), the account password (Password), and the connection string (Host), show the information about the remote connection that the link will be used to establish. The Username and Password values will be used to log in to the remote database defined by the Host value.

The Host column stores the Oracle Net service names. This column stores the exact character string you specify during the **create database link** command, and does not alter its case. Therefore, you should be careful with the case you use when creating database links. Otherwise, your queries against USER_DB_LINKS will have to take into account inconsistencies in the case of the Host column. For example, looking for a database link that uses the 'HQ' service descriptor would require you to enter

```
Select * from USER_DB_LINKS
 where UPPER(Host) = 'HQ';
```

since it is possible that there are entries with a Host value of 'hq' instead of 'HQ'.

NOTE
If you are using default logins to the remote database, then Password will be ***NULL****.*

The ALL_DB_LINKS view lists all of the database links that either are owned by the user or are PUBLIC database links. DBA_DB_LINKS lists all database links in the database. ALL_DB_LINKS and DBA_DB_LINKS share most of the same column definitions as USER_DB_LINKS; they have an Owner column rather than the Password column.

See Chapter 23 for further information on the uses of database links.

Materialized Views

You can query USER_MVIEWS to display information about the materialized views owned by your account. This view, whose columns are listed in Table 42-9, shows the structural information about the materialized view as well as its refresh schedule.

NOTE
See Chapter 24 for details on materialized views.

The name of the materialized view is found in the Mview_Name column of USER_MVIEWS. The local base table for the view is the Container_Name column. New USER_MVIEWS columns are available as of Oracle Database 10*g*, including Stale_Since (when the materialized view became stale).To determine which database links are being used by your materialized views, query the Master_Link column, as shown in the following example:

```
select Master_Link
  from USER_MVIEWS;
```

The names of the database links returned by this query can be used as input for queries against USER_DB_LINKS. This query will display all of the database link information available for database links used in your materialized views:

```
select *
  from USER_DB_LINKS
 where DB_Link in
     (select Master_Link
        from USER_MVIEWS);
```

Additional queries that are useful in the management of materialized views are provided in Chapter 24.

The ALL_MVIEWS and DBA_MVIEWS views have the same column definitions as USER_MVIEWS. You can use ALL_MVIEWS to display information about all materialized views that the user can access regardless of owner. DBA_MVIEWS lists materialized view information for all users in the database.

Two related views—USER_REFRESH and USER_REFRESH_CHILDREN—display information about refresh groups. USER_MVIEW_REFRESH_TIMES shows the last time each materialized view

was refreshed. As of Oracle Database 10*g*, you can access comments on the materialized views via USER_MVIEW_COMMENTS, and query rewrite details via USER_REWRITE_EQUIVALENCES.

Column Name	Description
Owner	The account that owns the materialized view
Mview_Name	The name of the materialized view
Container_Name	The base table (in the local database) for the materialized view's data
Query	The query that defines the materialized view
Query_Len	The length of the materialized view's base query
Updatable	A flag to indicate whether the snapshot can be updated
Update_Log	The name of the table that logs changes made to an updatable snapshot
Master_Rollback_Seg	The rollback segment to use during snapshot population and refresh operations
Master_Link	The database link used to access the master database
Rewrite_Enabled	A Yes/No flag to indicate if query rewrite is enabled for the view
Rewrite_Capability	Rules and restrictions for query rewrites
Refresh_Mode	DEMAND, COMMIT, or NEVER, depending on the refresh mode for the view
Refresh_Method	Values used to drive a fast refresh of the view (Complete, Fast, Never, or Force)
Build_Mode	IMMEDIATE, DEFERRED, or PREBUILT instantiation of the materialized view data during its creation
Fast_Refreshable	Methods available for fast refresh of the materialized view
Last_Refresh_Type	The most recent refresh type used
Last_Refresh_Date	A timestamp to record the last time the snapshot's data was refreshed
Staleness	The status of the materialized view's data relative to its master tables
After_Fast_Refresh	Staleness status following a fast refresh
Compile_State	Validity of the materialized view
Use_No_Index	Yes/No flag to indicate if the materialized view was created with the USING NO INDEX clause
Unknown_Trusted_FD	Indicates if the materialized view used trusted constraints for the refresh.
Stale_Since	Time the materialized view became stale

TABLE 42-9. *Columns in USER_MVIEWS*

Additional Materialized View Capabilities

You can query USER_MVIEW_ANALYSIS to see the materialized views that support query rewrite. If a materialized view contains references to a remote table, it will not be listed in this view. You can query the owner of the materialized view, its name (Mview_Name), and the owner of the base table (Mview_Table_Owner). Many of the columns in this view are flags, such as Summary ('Y' if the view contains an aggregation), Known_Stale ('Y' if the view's data is inconsistent with the base table), and Contains_Views ('Y' if the materialized view references a view).

If the materialized view contains aggregations, you can query USER_MVIEW_AGGREGATES for details on the aggregations. Columns in this view are as follows:

Owner	Owner of the materialized view
Mview_Name	Name of the materialized view
Position_in_Select	Position within the query
Container_Column	Name of the column
Agg_Function	Aggregate function
DistinctFlag	'Y' if the aggregation uses the DISTINCT function
Measure	The SQL text of the measure, excluding the aggregate function

You can query details of the relations within materialized views from the USER_MVIEW_DETAIL_RELATIONS and USER_MVIEW_KEYS data dictionary views. If the materialized view is based on joins, see USER_MVIEW_JOINS for the join details. In general, USER_MVIEW_ANALYSIS will be the most commonly used data dictionary view related to materialized views.

Materialized View Logs: USER_MVIEW_LOGS

Materialized view logs can be used by many materialized views to determine which records in the master table need to be refreshed in the materialized view of that table. You can query USER_MVIEW_LOGS for information about a user's logs, including the name of the master table (Master), the table holding the log records (Log_Table), and whether the materialized view is based on the primary key or the ROWID (the Primary_Key and ROWIDs columns). USER_MVIEW_LOGS is usually queried for maintenance purposes, such as to determine the name of the trigger used to create the materialized view log records.

DBA_MVIEW_LOGS has the same column definitions as USER_MVIEW_LOGS, and shows all materialized view logs in the database. You can query USER_BASE_TABLE_MVIEWS for a listing of the materialized view master tables using materialized view logs.

Triggers, Procedures, Functions, and Packages

You can use procedures, packages, and triggers—blocks of PL/SQL code stored in the database—to enforce business rules or to perform complicated processing. Triggers are described in Chapter 30. Procedures, functions, and packages are described in Chapter 31. In the following sections, you will see how to query the data dictionary for information about triggers, procedures, packages, and functions.

Triggers: USER_TRIGGERS

USER_TRIGGERS contains information about the triggers owned by your account. This view, whose columns are listed in Table 42-10, shows the trigger type and body.

The ALL_TRIGGERS view lists the triggers for all tables to which you have access. DBA_TRIGGERS lists all of the triggers in the database. Both of these views contain an additional column, Owner, which records the owner of the trigger.

A second trigger-related data dictionary view, USER_TRIGGER_COLS, shows how columns are used by a trigger. It lists the name of each column affected by a trigger, as well as how the trigger is used. Like USER_TRIGGERS, "ALL" and "DBA" versions of this data dictionary view are available.

Procedures, Functions, and Packages: USER_SOURCE

The source code for existing procedures, functions, packages, and package bodies can be queried from the USER_SOURCE data dictionary view. The Type column in USER_SOURCE identifies the procedural object as a 'PROCEDURE', 'FUNCTION', 'PACKAGE', 'PACKAGE BODY', 'TRIGGER', 'TYPE', 'TYPE BODY', or 'JAVA SOURCE'. Each line of code is stored in a separate record in USER_SOURCE.

Column Name	Description
Trigger_Name	Name of the trigger
Trigger_Type	The type of trigger (BEFORE STATEMENT, BEFORE EACH ROW, and so on)
Triggering_Event	The command that executes the trigger (INSERT, UPDATE, or DELETE)
Table_Owner	The owner of the table that the trigger is defined for
Base_Object_Type	The type of object on which the trigger is based (TABLE, VIEW, SCHEMA, or DATABASE)
Table_Name	The name of the table or view that the trigger is defined for
Column_Name	For nested table triggers, the name of the nested table column
Referencing_Names	Names used for referencing OLD and NEW values in the trigger
When_Clause	The **when** clause used for the trigger
Status	Whether the trigger is ENABLED or DISABLED
Description	The description for the trigger
Action_Type	Action type for the trigger body (CALL or PL/SQL)
Trigger_Body	The trigger text

TABLE 42-10. *Columns in USER_TRIGGERS*

You can select information from USER_SOURCE via a query similar to the one shown in the following listing. In this example, the Text column is selected and ordered by Line number. The Name of the object and the object Type are used to specify which object's source code to display.

```
select Text
  from USER_SOURCE
 where Name = '&procedure_name'
   and Type = 'PROCEDURE'
 order by Line;
```

NOTE
The sequence of the lines is maintained by the Line column; therefore, the Line column should be used in the ***order by*** *clause.*

The ALL_SOURCE and DBA_SOURCE views have all of the columns found in USER_SOURCE plus an additional Owner column (the owner of the object). ALL_SOURCE can be used to display the source code for all procedural objects that the user can access regardless of owner. DBA_SOURCE lists the source code for all users in the database.

NOTE
To see the persistent parameter settings for PL/SQL units, query USER_STORED_SETTINGS.

Code Errors: USER_ERRORS

The **show errors** command in SQL*Plus checks the USER_ERRORS data dictionary view for the errors associated with the most recent compilation attempt for a procedural object. **show errors** will display the line and column number for each error, as well as the text of the error message.

To view errors associated with previously created procedural objects, you may query USER_ERRORS directly. You may need to do this when viewing errors associated with package bodies, since a package compilation that results in an error may not display the package body's error when you execute the **show error** command. You may also need to query USER_ERRORS when you encounter compilation errors with multiple procedural objects.

The following are the columns available in USER_ERRORS:

Name	The name of the procedural object
Type	The object type ('PROCEDURE', 'FUNCTION', 'PACKAGE', 'PACKAGE BODY', 'TRIGGER', 'TYPE', 'TYPE BODY', 'VIEW', 'JAVA CLASS', or 'JAVA SOURCE')
Sequence	The line sequence number, for use in the query's **order by** clause
Line	The line number within the source code at which the error occurs
Position	The position within the Line at which the error occurs
Text	The text of the error message
Attribute	Flag to indicate whether the selected row is an error ('ERROR') or a warning ('WARNING')
Message_Number	Numerical error number, without prefix

Queries against this view should always include the Sequence column in the **order by** clause. "ALL" and "DBA" versions of this view are also available; they feature an additional column, Owner, which records the owner of the object.

Code Size: USER_OBJECT_SIZE

You can query the amount of space used in the SYSTEM tablespace for a procedural object from the USER_OBJECT_SIZE data dictionary view. As shown in the following listing, the four separate size areas can be added together to determine the total space used in the SYSTEM data dictionary tables to store the object. The four Size columns, along with the Name and Type columns, constitute all of the columns in this view.

```
select Source_Size+Code_Size+Parsed_Size+Error_Size Total
  from USER_OBJECT_SIZE
 where Name = '&procedure_name'
   and Type = 'PROCEDURE';
```

There is also a "DBA" version of this view available: DBA_OBJECT_SIZE lists the sizes for all objects in the database.

Dimensions

You can create and maintain dimensions and hierarchies.

You can query the Dimension_Name column of the USER_DIMENSIONS view to see the names of your dimensions. USER_DIMENSIONS also contains columns for the dimension owner (Owner), status (the Invalid column, set to 'Y' or 'N'), and revision level (the Revision column). The attributes of a dimension are accessed via additional data dictionary views.

To see the hierarchies within a dimension, query USER_DIM_HIERARCHIES. This view has only three columns: Owner, Dimension_Name, and Hierarchy_Name. Querying USER_DIM_HIERARCHIES for the GEOGRAPHY dimension returns the name of its hierarchies:

```
select Hierarchy_Name
  from USER_DIM_HIERARCHIES;

HIERARCHY_NAME
------------------------------
COUNTRY_ROLLUP
```

You can see the hierarchy details for COUNTRY_ROLLUP by querying USER_DIM_CHILD_OF, as shown in the following listing. The first command creates a dimension named GEOGRAPHY that records the hierarchies of countries and continents.

```
create dimension GEOGRAPHY
   level COUNTRY_ID      is COUNTRY.Country
   level CONTINENT_ID    is CONTINENT.Continent
   hierarchy COUNTRY_ROLLUP (
      COUNTRY_ID           child of
      CONTINENT_ID
   join key COUNTRY.Continent references CONTINENT_id);
```

```
column join_key_id format a4

select Child_Level_Name,
       Parent_Level_Name,
       Position, Join_Key_Id
  from USER_DIM_CHILD_OF
where Hierarchy_Name = 'COUNTRY_ROLLUP';

CHILD_LEVEL_NAME       PARENT_LEVEL_NAME               POSITION JOIN
---------------------- ------------------------------ --------- ----
COUNTRY_ID             CONTINENT_ID                           1 1
```

To see the join key for a hierarchy, query USER_DIM_JOIN_KEY:

```
select Level_name, Child_Join_Column
  from USER_DIM_JOIN_KEY
 where Dimension_Name = 'GEOGRAPHY'
   and Hierarchy_Name = 'COUNTRY_ROLLUP';

LEVEL_NAME                     CHILD_JOIN_COLUMN
------------------------------ ------------------------------
CONTINENT_ID                   CONTINENT
```

You can see the levels of a dimension by querying the USER_DIM_LEVELS data dictionary view, and see the key columns for the levels via USER_DIM_LEVEL_KEY. Attribute information for dimensions is accessible via USER_DIM_ATTRIBUTES.

There are "ALL" and "DBA" views of all of the data dictionary views related to dimensions. Because the "USER" views for dimensions contain an Owner column, there are no additional columns found in the corresponding "ALL" and "DBA" views.

Space Allocation and Usage, Including Partitions and Subpartitions

You can query the data dictionary to determine the space that is available and allocated for database objects. In the following sections, you will see how to determine the default storage parameters for objects, your space usage quota, available free space, and the way in which objects are physically stored. For information on Oracle's methods of storing data, see Chapters 20 and 46.

Tablespaces: USER_TABLESPACES

You can query the USER_TABLESPACES data dictionary view to determine which tablespaces you have been granted access to and the default storage parameters in each. A tablespace's default storage parameters will be used for each object stored within that tablespace unless the **create** or **alter** command for that object specifies its own storage parameters. The Storage-Related columns in USER_TABLESPACES, listed in Table 42-11, are very similar to the Storage-Related columns in USER_TABLES. See the "Storage" entry of the Alphabetical Reference for further details.

There is no "ALL" version of this view. DBA_TABLESPACES shows the storage parameters for all tablespaces.

Column Name	Description
Tablespace_Name	The name of the tablespace.
Block_Size	Database block size in use for this tablespace.
Initial_Extent	The default INITIAL parameter for objects in the tablespace.
Next_Extent	The default NEXT parameter for objects in the tablespace.
Min_Extents	The default MINEXTENTS parameter for objects in the tablespace.
Max_Extents	The default MAXEXTENTS parameter for objects in the tablespace.
Pct_Increase	The default PCTINCREASE parameter for objects in the tablespace.
Min_Extlen	The minimum extent size for objects in the tablespace.
Status	The tablespace's status ('ONLINE', 'OFFLINE', 'INVALID', 'READ ONLY'). An "invalid" tablespace is one that has been dropped; its record is still visible via this view.
Contents	A flag to indicate whether the tablespace is used to store permanent objects ('PERMANENT') or only temporary segments ('TEMPORARY').
Logging	A flag to indicate the default LOGGING/NOLOGGING parameter value for objects in the tablespace.
Extent_Management	Where extent management is performed for the tablespace ('DICTIONARY' or 'LOCAL').
Allocation_Type	Type of extent allocation in effect.
Segment_Space_Management	Flag to indicate if free space is managed via free lists ('MANUAL') or bitmaps ('AUTO').
Retention	Undo tablespace retention ('GUARANTEE','NO GUARANTEE', or 'NOT APPLY').
Bigfile	Indicates if the tablespace is a bigfile tablespace ('YES') or a smallfile tablespace ('NO').
Force_Logging	Yes/No flag to indicate if tablespace is in force logging mode.

TABLE 42-11. *Columns in USER_TABLESPACES*

Space Quotas: USER_TS_QUOTAS

USER_TS_QUOTAS is a very useful view for determining the amount of space you have currently allocated and the maximum amount of space available to you by tablespace. A sample query of USER_TS_QUOTAS is shown in the following listing:

```
select * from USER_TS_QUOTAS;

TABLESPACE_NAME       BYTES  MAX_BYTES      BLOCKS MAX_BLOCKS
--------------- ---------- ---------- ---------- ----------
USERS                 67584          0         33          0
```

USER_TS_QUOTAS contains one record for each Tablespace_Name. The Bytes column reflects the number of bytes allocated to objects owned by the user. Max_Bytes is the maximum number of bytes the user can own in that tablespace; if there is no quota for that tablespace, then Max_Bytes will display a value of 0. The Bytes and Max_Bytes columns are translated into Oracle blocks in the Blocks and Max_Blocks columns, respectively.

There is no "ALL" version of this view. DBA_TS_QUOTAS shows the storage quotas for all users for all tablespaces and is a very effective way to list space usage across the entire database.

Segments and Extents: USER_SEGMENTS and USER_EXTENTS

As described in Chapter 20, space is allocated to objects (such as tables, clusters, and indexes) in *segments*, the physical counterparts to the logical objects created in the database. You can query USER_SEGMENTS to see the current storage parameters and space usage in effect for your segments. USER_SEGMENTS is very useful when you are in danger of exceeding one of the storage limits; its columns are listed in Table 42-12.

Column Name	Description
Segment_Name	The name of the segment
Partition_Name	**NULL** if the object is not partitioned; otherwise, the segment's partition name
Segment_Type	The type of segment ('TABLE', 'CLUSTER', 'INDEX', 'ROLLBACK', and so on)
Tablespace_Name	The name of the tablespace in which the segment is stored
Bytes	The number of bytes allocated to the segment
Blocks	The number of Oracle blocks allocated to the segment
Extents	The number of extents in the segment
Initial_Extent	The size of thse initial extent in the segment
Next_Extent	The value of the NEXT parameter for the segment
Min_Extents	The minimum number of extents in the segment
Max_Extents	The value of the MAXEXTENTS parameter for the segment
Pct_Increase	The value of the PCTINCREASE parameter for the segment
Freelists	The number of process freelists (lists of data blocks in the segment that can be used during inserts) allocated to the segment; if a segment has multiple freelists, then contention for free blocks during concurrent **inserts** will be lessened
Freelist_Groups	The number of freelist groups allocated to the segment
Buffer_Pool	Buffer pool into which the segment will be read ('DEFAULT', 'KEEP', or 'RECYCLE') if you have defined multiple buffer pools

TABLE 42-12. *Columns in USER_SEGMENTS*

Segments consist of contiguous sections called *extents*. The extents that constitute segments are described in USER_EXTENTS. In USER_EXTENTS, you will see the actual size of each extent within the segment; this is very useful for tracking the impact of changes to the **next** and **pctincrease** settings. In addition to the Segment_Name, Segment_Type, and Tablespace_Name columns, USER_EXTENTS has three new columns: Extent_ID (to identify the extent within the segment), Bytes (the size of the extent, in bytes), and Blocks (the size of the extent, in Oracle blocks).

Both USER_SEGMENTS and USER_EXTENTS have "DBA" versions, which are useful for listing the space usage of objects across owners. Both DBA_SEGMENTS and DBA_EXTENTS have an additional Owner column. If you want to list all of the owners who own segments in a tablespace, you can query based on the Tablespace_Name column in DBA_SEGMENTS, and list all of the owners of segments in that tablespace. There are no "ALL" versions of these views.

Partitions and Subpartitions

A single table's data can be stored across multiple partitions. See Chapter 17 for examples of partitions and descriptions of the indexing options available. To see how a table is partitioned, you should query the USER_PART_TABLES data dictionary view, whose columns are listed, by category, in Table 42-13.

Most of the columns of USER_PART_TABLES define the default storage parameters for the table partitions. When a partition is added to the table, it will by default use the storage parameters shown in USER_PART_TABLES. USER_PART_TABLES also shows the number of partitions in the table (Partition_Count), the number of columns in the partition key (Partitioning_Key_Count), and the type of partitioning (Partitioning_Type).

Identification	Storage-Related
Table_Name	Def_Tablespace_Name
Partitioning_Type	Def_Pct_Free
Subpartitioning_Type	Def_Pct_Used
Partition_Count	Def_Ini_Trans
Def_Subpartition_Count	Def_Max_Trans
Partitioning_Key_Count	Def_Initial_Extent
Subpartitioning_Key_Count	Def_Next_Extent
Def_Logging	Def_Min_Extents
Def_Buffer_Pool	Def_Max_Extents
	Def_Pct_Increase
	Def_Freelists
	Def_Freelist_Groups
	Def_Compression

TABLE 42-13. *Columns in USER_PART_TABLES*

Identification	Storage-Related	Statistics-Related
Table_Name	Tablespace_Name	Num_Rows
Composite	Pct_Free	Blocks
Partition_Name	Pct_Used	Empty_Blocks
Subpartition_Count	Ini_Trans	Avg_Space
High_Value	Max_Trans	Chain_Cnt
High_Value_Length	Initial_Extent	Avg_Row_Len
Partition_Position	Next_Extent	Sample_Size
Logging	Min_Extent	Last_Analyzed
Buffer_Pool	Max_Extent	Global_Stats
	Pct_Increase	User_Stats
	Freelists	
	Freelist_Groups	
	Compression	

TABLE 42-14. *Columns in USER_TAB_PARTITIONS*

USER_PART_TABLES stores a single row for each table that has been partitioned. To see information about each of the individual partitions that belong to the table, you can query USER_TAB_PARTITIONS. In USER_TAB_PARTITIONS, you will see one row for each of the table's partitions. The columns of USER_TAB_PARTITIONS are shown, by category, in Table 42-14.

USER_TAB_PARTITIONS contains columns that identify the table to which the partition belongs and shows the partition's storage parameters and the statistics for the partition. The Statistics-Related columns are populated when the table is analyzed. The Identification columns show the high value for the range used to define the partition (High_Value) and the position of the partition within the table (Partition_Position).

The columns used for the partition key are accessible via the USER_PART_KEY_COLUMNS data dictionary view. USER_PART_KEY_COLUMNS contains only four columns:

Name	The name of the partitioned table or index
Object_Type	Object type (TABLE or INDEX)
Column_Name	The name of the column that is part of the partition key
Column_Position	The position of the column within the partition key

Statistics for the partition columns are accessible via the USER_PART_COL_STATISTICS data dictionary view. The columns in USER_PART_COL_STATISTICS closely mirror those in USER_TAB_COL_STATISTICS.

Data histogram information for partitions is accessible via the USER_PART_HISTOGRAMS data dictionary view. The columns of this view display the endpoint values for each of the buckets in the histogram. The columns are Table_Name, Partition_Name, Column_Name, Bucket_Number, Endpoint_Value, and Endpoint_Actual_Value.

Since indexes may be partitioned, there is a USER_IND_PARTITIONS data dictionary view available. The columns in USER_IND_PARTITIONS can be grouped into three categories, as shown in Table 42-15.

The columns in USER_IND_PARTITIONS parallel those in USER_INDEXES, with a few modifications to the Identification columns. The Identification columns for partitioned indexes include the name of the partition, the high value for the partition, and the position of the partition within the table. The Statistics-Related columns are populated when the partition is analyzed. The Space-Related columns describe the space allocation for the index; see the "Storage" entry in the Alphabetical Reference for information on storage parameters.

If a partition has subpartitions, you can see the details for the subpartitions via data dictionary views. USER_IND_SUBPARTITIONS contains the Space-Related and Statistics-Related columns of USER_IND_PARTITIONS, along with columns to identify the subpartition (Subpartition_Name and Subpartition_Position). Similarly, USER_TAB_SUBPARTITIONS contains the Space-Related and Statistics-Related columns of USER_TAB_PARTITIONS, along with Subpartition_Name and Subpartition_Position columns. Thus, you can determine the space definitions of each of your partitions and subpartitions.

As shown earlier in this section, you can query the USER_PART_COL_STATISTICS and USER_PART_HISTOGRAMS data dictionary views for statistical information regarding partitions. For

Identification	Space-Related	Statistics-Related
Index_Name	Tablespace_Name	Blevel
Composite	Ini_Trans	Leaf_Blocks
Partition_Name	Max_Trans	Distinct_Keys
Subpartition_Count	Initial_Extent	Avg_Leaf_Blocks_Per_Key
High_Value	Next_Extent	Avg_Data_Blocks_Per_Key
High_Value_Length	Min_Extent	Clustering_Factor
Partition_Position	Max_Extent	Num_Rows
Status	Pct_Increase	Sample_Size
Logging	Pct_Free	Last_Analyzed
Buffer_Pool	Freelists	User_Stats
Compression	Freelist_Groups	Pct_Direct_Access
Domidx_Opstatus		Global_Stats
Parameters		

TABLE 42-15. *Columns in USER_IND_PARTITIONS*

subpartitions, you can query USER_SUBPART_COL_STATISTICS and USER_SUBPART_HISTOGRAMS, whose structures mirror that of the partition statistics views. To see the subpartition key columns, you can query USER_SUBPART_KEY_COLUMNS, whose column structure is identical to that of USER_PART_KEY_COLUMNS.

Free Space: USER_FREE_SPACE

In addition to viewing the space you have used, you can also query the data dictionary to see how much space is currently marked as "free" space. USER_FREE_SPACE lists the free extents in all tablespaces accessible to the user. It lists by Tablespace_Name the File_ID, Block_ID, and relative file number of the starting point of the free extent. The size of the free extent is listed in both bytes and blocks. DBA_FREE_SPACE is frequently used by DBAs to monitor the amount of free space available and the degree to which it has become fragmented.

Users and Privileges

Users and their privileges are recorded within the data dictionary. In the following sections, you will see how to query the data dictionary for information about user accounts, resource limits, and user privileges.

Users: USER_USERS

You can query USER_USERS to list information about your account. The USER_USERS view includes your Username, User_ID (a number assigned by the database), Default_Tablespace, Temporary_Tablespace, and Created date (when your account was created). The Account_Status column in USER_USERS displays the status of your account, whether it is locked, unlocked ('OPEN'), or expired. If the account is locked, the Lock_Date column will display the date the account was locked and the Expiry_Date column will display the date of expiration. You can also query information about the resource consumer group (Initial_Rsrc_Consumer_Group) assigned to you by the DBA, and your External_Name value.

ALL_USERS contains only the Username, User_ID, and Created columns from USER_USERS, but it lists that information for all accounts in the database. ALL_USERS is useful when you need to know the usernames that are available (for example, during **grant** commands). DBA_USERS contains all of the columns in USER_USERS, plus two additional columns: Password (the encrypted password for the account) and Profile (the user's resource profile). DBA_USERS lists this information for all users in the database.

Resource Limits: USER_RESOURCE_LIMITS

In Oracle, *profiles* can be used to place limits on the amount of system and database resources available to a user. If no profiles are created in a database, the default profile, which specifies unlimited resources for all users, will be used. The resources that can be limited are described in the **create profile** entry of the Alphabetical Reference. Profiles enforce additional security measures, such as expiration dates on accounts and the minimum length of passwords.

To view the limits that are in place for your current session, you can query USER_RESOURCE_LIMITS. Its columns are the following:

Resource_Name	The name of the resource (for example, SESSIONS_PER_USER)
Limit	The limit placed on this resource

USER_PASSWORD_LIMITS describes the password profile parameters for the user. It has the same columns as USER_RESOURCE_LIMITS.

There is no "ALL" or "DBA" version of this view; it is limited to the user's current session. To see the cost associated with each available resource, you can query the RESOURCE_COST view. DBAs can access the DBA_PROFILES view to see the resource limits for all profiles. The Resource_Type column of DBA_PROFILES indicates whether the resource profile is a 'PASSWORD' or 'KERNEL' profile.

Table Privileges: USER_TAB_PRIVS

To view grants for which you are the grantee, the grantor, or the object owner, query USER_TAB_PRIVS (user table privileges). In addition to its Grantee, Grantor, and Owner columns, this view contains columns for the Table_Name, Hierarchy, Privilege, and a flag (set to 'YES' or 'NO') to indicate whether the privilege was granted **with admin option** (Grantable).

USER_TAB_PRIVS_MADE displays the USER_TAB_PRIVS records for which the user is the owner (it therefore lacks an Owner column). USER_TAB_PRIVS_RECD (user table privileges received) displays the USER_TAB_PRIVS records for which the user is the grantee (it therefore lacks a Grantee column). Since both USER_TAB_PRIVS_MADE and USER_TAB_PRIVS_RECD are simply subsets of USER_TAB_PRIVS, you can duplicate their functionality simply by querying against USER_TAB_PRIVS with an appropriate **where** clause to view the subset you want.

There are "ALL" versions available for USER_TAB_PRIVS, USER_TAB_PRIVS_MADE, and USER_TAB_PRIVS_RECD. The "ALL" versions list those objects for which either the user or PUBLIC is the grantee or grantor. There is a "DBA" version of USER_TAB_PRIVS named DBA_TAB_PRIVS that lists all object privileges granted to all users in the database. DBA_TAB_PRIVS has the same column definitions as USER_TAB_PRIVS. Instead of an Owner column, ALL_TAB_PRIVS has a Table_Schema column that displays the schema of the object.

Column Privileges: USER_COL_PRIVS

In addition to granting privileges on tables, you can also grant privileges at the column level. For example, you can grant users the ability to **update** only certain columns in a table. See Chapter 18 and the **grant** command in the Alphabetical Reference for further details.

The data dictionary views used to display column privileges are almost identical in design to the table privileges views described in the previous section. There is an additional Column_Name column in each of the COL views, and the Hierarchy column is not found there. USER_COL_PRIVS is analogous to USER_TAB_PRIVS, USER_COL_PRIVS_MADE is analogous to USER_TAB_PRIVS_MADE, and USER_COL_PRIVS_RECD is analogous to USER_TAB_PRIVS_RECD.

"ALL" versions are available for all of the column privileges views. DBA_COL_PRIVS lists all column privileges that have been granted to users in the database (just as DBA_TAB_PRIVS lists all table privileges granted to users).

System Privileges: USER_SYS_PRIVS

USER_SYS_PRIVS lists the system privileges that have been granted to the user. Its columns are Username, Privilege, and Admin_Option (a flag set to 'YES' or 'NO' to indicate whether the privilege

was granted **with admin option**). All system privileges directly granted to a user are displayed via this view. System privileges granted to a user via a role are not displayed here. The following query shows sample output for the PRACTICE account:

```
select Username, Privilege * from USER_SYS_PRIVS;

USERNAME                       PRIVILEGE
------------------------------ ----------------------
PRACTICE                       CREATE DIMENSION
PRACTICE                       UNLIMITED TABLESPACE
```

There is no "ALL" version of this view. To see the system privileges granted to all users in the database, query DBA_SYS_PRIVS, which has nearly the same column definitions as USER_SYS_PRIVS (the DBA version has a Grantee column, while the USER version has a Username column).

Roles

In addition to privileges granted directly to users, sets of privileges may be grouped into roles. Roles may be granted to users or to other roles, and may consist of both object and system privileges. For information on the use and management of roles, see Chapter 18.

To see which roles have been granted to you, query the USER_ROLE_PRIVS data dictionary view. Any roles that have been granted to PUBLIC will also be listed here. The available columns for USER_ROLE_PRIVS are as follows:

Username	The username (may be 'PUBLIC')
Granted_Role	The name of the role granted to the user
Admin_Option	A flag to indicate whether the role was granted **with admin option** ('YES' or 'NO')
Default_Role	A flag to indicate whether the role is the user's default role ('YES' or 'NO')
OS_Granted	A flag to indicate whether the operating system is being used to manage roles ('YES' or 'NO')

To list all of the roles available in the database, you need to have DBA authority; you can then query DBA_ROLES to list all roles. DBA_ROLE_PRIVS lists the assignment of those roles to all of the users in the database.

Roles may receive three different types of grants, and each has a corresponding data dictionary view:

Table/column grants	ROLE_TAB_PRIVS. Similar to USER_TAB_PRIVS and USER_COL_PRIVS, except that it has a Role column instead of a Grantee column and does not have a Grantor column.
System privileges	ROLE_SYS_PRIVS. Similar to USER_SYS_PRIVS, except that it has a Role column instead of a Username column.
Role grants	ROLE_ROLE_PRIVS. Lists all roles that have been granted to other roles.

NOTE
If you are not a DBA, these data dictionary views list only those roles that have been granted to you.

In addition to these views, there are two views, each with a single column, that list the privileges and roles enabled for the current session:

SESSION_PRIVS	The Privilege column lists all system privileges available to the session, whether granted directly or via roles.
SESSION_ROLES	The Role column lists all roles that are currently enabled for the session.

SESSION_PRIVS and SESSION_ROLES are available to all users.

NOTE
See Chapter 18 for information on enabling and disabling roles and the setting of default roles.

Auditing

As a non-DBA user within an Oracle database, you cannot enable the database's auditing features. If auditing has been enabled, there are data dictionary views that anyone can use to view the audit trail.

Many different audit trail data dictionary views are available. Most of these views are based on a single audit trail table in the database (SYS.AUD$). The most generic of the audit trail views available is named USER_AUDIT_TRAIL. Its columns are described in Table 42-16. Since this view

Column Name	Description
OS_Username	The audited user's operating system account
Username	The Oracle username of the audited user
UserHost	A numeric ID for the instance used by the audited user
Terminal	The user's operating-system terminal identifier
TimeStamp	The date and time the audit record was created
Owner	The owner of the object affected by an action (for action audits)
Obj_Name	The name of the object affected by an action (for action audits)
Action	The numeric code for the audited action
Action_Name	The name of the audited action
New_Owner	The owner of the object named in the New_Name column

TABLE 42-16. *Columns in USER_AUDIT_TRAIL*

Column Name	Description
New_Name	The new name of an object that has been **rename**d
Obj_Privilege	The object privilege that has been **grant**ed or **revoke**d
Sys_Privilege	The system privilege that has been **grant**ed or **revoke**d
Admin_Option	A flag to indicate whether the role or system privilege was **grant**ed **with admin option** ('Y' or 'N')
Grantee	The username specified in a **grant** or **revoke** command
Audit_Option	The auditing options set via an **audit** command
Ses_Actions	A string of characters serving as a session summary, recording success and failure for different actions
Logoff_Time	The date and time the user logged off
Logoff_LRead	The number of logical reads performed during the session
Logoff_PRead	The number of physical reads performed during the session
Logoff_LWrite	The number of logical writes performed during the session
Logoff_DLock	The number of deadlocks detected during the session
Comment_Text	A text comment on the audit trail entry
SessionID	The numeric ID for the session
EntryID	A numeric ID for the audit trail entry
StatementID	The numeric ID for each command that was executed
ReturnCode	The return code for each command that was executed; if the command was successful, then the ReturnCode will be 0
Priv_Used	The system privilege used to execute the action
Client_ID	Client ID
Session_CPU	Session CPU used
Extended_Timestamp	Timestamp of the creation of the audit trail entry in session's time zone
Proxy_SessionID	Proxy session serial number, if enterprise user has logged through a proxy mechanism
Global_UID	Global user identifier for the user, if the user had logged in as an enterprise user
Instance_Number	Instance number as specified in the initialization parameter file
OS_Process	Operating-system process identifier of the Oracle server process
TransactionID	Transaction identifier of the transaction in which the object is accessed or modified
SCN	SCN of the query
SQL_Bind	Bind variable data of the query
SQL_Text	SQL text of the query

TABLE 42-16. *Columns in USER_AUDIT_TRAIL* (continued)

shows the audit records for many different types of actions, many of the columns may be inapplicable for any given row. The "DBA" version of this view, DBA_AUDIT_TRAIL, lists all entries from the audit trail table; USER_AUDIT_TRAIL lists only those that are relevant to the user.

As of Oracle Database 10*g*, a number of new columns are available in USER_AUDIT_TRAIL, including SQL_Text, SQL_Bind, OS_Process, and SCN. See the chapters related to flashback query usage for details on the uses of SCN (System Change Number) values.

As shown in Table 42-16, a vast array of auditing capabilities is available (see the **audit** command in the Alphabetical Reference for a complete listing). Each type of audit can be accessed via its own data dictionary view. The following are the available views:

USER_AUDIT_OBJECT	For statements concerning objects
USER_AUDIT_SESSION	For connections and disconnections
USER_AUDIT_STATEMENT	For **grant**, **revoke**, **audit**, **noaudit**, and **alter system** commands issued by the user

There are "DBA" versions available for each of the "USER" views in the previous list; the "DBA" versions show all of the audit trail records that fit into the view's category.

You can view the auditing options that are currently in effect for your objects by querying USER_OBJ_AUDIT_OPTS. For each object listed in USER_OBJ_AUDIT_OPTS, the audit options for each command that may be performed on that object (identified by the Object_Name and Object_Type columns) are listed in USER_OBJ_AUDIT_OPTS. The column names of USER_OBJ_AUDIT_OPTS correspond to the first three letters of the command (for example, Alt for **alter**, Upd for **update**, and so on). Each column will record whether that command is audited for that object when the command is successful ('S'), unsuccessful ('U'), or both. The default auditing options in effect for any new objects in the database can be displayed via the ALL_DEF_AUDIT_OPTS view, which has the same column naming conventions as USER_OBJ_AUDIT_OPTS. USER_OBJ_AUDIT_OPTS includes Object_Name and Object_Type columns.

The commands that can be audited are stored in a reference table named AUDIT_ACTIONS, which has two columns: Action (the numeric code for the action) and Name (the name of the action/command). Action and Name correspond to the Action and Action_Name columns of USER_AUDIT_TRAIL.

DBAs can use several additional auditing views that do not have "USER" counterparts, including DBA_AUDIT_EXISTS, DBA_PRIV_AUDIT_OPTS, DBA_STMT_AUDIT_OPTS, and STMT_AUDIT_OPTION_MAP. See the *Oracle Database Reference* for details on these DBA-only views.

Miscellaneous

In addition to the data dictionary views described earlier in this chapter, several miscellaneous views and tables may be available within your data dictionary. These views and tables include DBA-only views and the table used when the **explain plan** command is executed. In the following sections, you will see brief descriptions for each of the miscellaneous view types.

Monitoring: The V$ Dynamic Performance Tables

The views that monitor the database environment performance are called the *dynamic performance views*. The dynamic performance views are commonly referred to as the *V$* (pronounced "Vee-Dollar") tables, because they all begin with the letter *V* followed by the dollar sign ($).

The definitions and usage of columns within the monitoring views are subject to change with each version of the database that is released. Correctly interpreting the results of ad hoc queries against views usually requires referring to the *Oracle Database Administrator's Guide*. The V$ tables are normally used only by the DBA. Consult the *Oracle Database Reference* and the *Oracle DBA Handbook* for details on the use of the V$ views.

CHAINED_ROWS

The **analyze** command can be used to generate a listing of the chained or migrated rows within a table. This listing of chained rows can be stored in a table called CHAINED_ROWS. To create the CHAINED_ROWS table in your schema, run the utlchain.sql script (usually found in the /rdbms/admin subdirectory under the Oracle home directory).

To populate the CHAINED_ROWS table, use the **list chained rows into** clause of the **analyze** command, as shown in the following listing:

```
analyze TABLE BIRTHDAY list chained rows into CHAINED_ROWS;
```

The CHAINED_ROWS table lists the Owner_Name, Table_Name, Cluster_Name (if the table is in a cluster), Partition_Name (if the table is partitioned), Subpartition_Name (if the table contains subpartitions), Head_RowID (the ROWID for the row), and an Analyze_TimeStamp column that shows the last time the table or cluster was analyzed. You can query the table based on the Head_RowID values in CHAINED_ROWS, as shown in the following example:

```
select * from BIRTHDAY
 where RowID in
       (select Head_RowID
          from CHAINED_ROWS
         where Table_Name = 'BIRTHDAY');
```

If the chained row is short in length, then it may be possible to eliminate the chaining by deleting and reinserting the row.

PLAN_TABLE

When tuning SQL statements, you may want to determine the steps that the optimizer will take to execute your query. To view the query path, you must first create a table in your schema named PLAN_TABLE. The script used to create this table is called utlxplan.sql, and is usually stored in the /rdbms/admin subdirectory of the Oracle software home directory.

After you have created the PLAN_TABLE table in your schema, you can use the **explain plan** command, which will generate records in your PLAN_TABLE, tagged with the Statement_ID value you specify for the query you want to have explained. See Chapter 43 for details on the generation of the explain plan listings.

The ID and Parent_ID columns in PLAN_TABLE establish the hierarchy of steps (Operations) that the optimizer will follow when executing the query. See Chapter 43 for details on the Oracle optimizer and the interpretation of PLAN_TABLE records.

Interdependencies: USER_DEPENDENCIES and IDEPTREE

Objects within Oracle databases can depend upon each other. For example, a stored procedure may depend upon a table, or a package may depend upon a package body. When an object within the database changes, any procedural object that depends upon it will have to be recompiled. This recompilation can take place either automatically at run time (with a consequential performance penalty) or manually (see Chapter 31 for details on compiling procedural objects).

Two sets of data dictionary views are available to help you track dependencies. The first is USER_DEPENDENCIES, which lists all *direct* dependencies of objects. However, this only goes one level down the dependency tree. To fully evaluate dependencies, you must create the recursive dependency-tracking objects in your schema. To create these objects, run the utldtree.sql script (usually located in the /rdbms/admin subdirectory of the Oracle home directory). This script creates two objects you can query: DEPTREE and IDEPTREE. They contain identical information (although their column definitions differ), but IDEPTREE is indented based on the pseudo-column Level, and is thus easier to read and interpret.

DBA-Only Views

Since this chapter is intended for use by developers and end users, the data dictionary views available only to DBAs are not covered here. The DBA-only views are used to provide information about distributed transactions, lock contention, rollback segments, and other internal database functions. For information on the use of the DBA-only views, see the *Oracle Database Administrator's Guide*.

Oracle Label Security

Users of Oracle Label Security can view additional data dictionary views, including ALL_SA_GROUPS, ALL_SA_POLICIES, ALL_SA_USERS, and ALL_SA_USER_PRIVS. For details on the usage of these views, see the *Oracle Label Security Administrator's Guide*.

SQL*Loader Direct Load Views

To manage the direct load option within SQL*Loader, Oracle maintains a number of data dictionary views. These generally are only queried for debugging purposes, upon request from Oracle Customer Support. The SQL*Loader direct load option is described under the "SQLLDR" entry in the Alphabetical Reference; its supporting data dictionary views are listed here:

- LOADER_COL_INFO
- LOADER_CONSTRAINT_INFO
- LOADER_FILE_TS
- LOADER_PARAM_INFO
- LOADER_PART_INFO
- LOADER_REF_INFO
- LOADER_TAB_INFO
- LOADER_TRIGGER_INFO

For details on the use of these views, see the catldr.sql script, usually located in the /rdbms/admin subdirectory of the Oracle home directory.

Globalization Support Views

Three data dictionary views are used to display information about the Globalization Support parameters currently in effect in the database. Nonstandard values for the NLS parameters (such as NLS_DATE_FORMAT and NLS_SORT) can be set via the database's parameter file or via the **alter session** command. (See the **alter session** command in the Alphabetical Reference for further information on NLS settings.) To see the current NLS settings for your session, instance, and database, query NLS_SESSION_PARAMETERS, NLS_INSTANCE_PARAMETERS, and NLS_DATABASE_PARAMETERS, respectively.

Libraries

Your PL/SQL routines (see Chapter 31) can call external C programs. To see which external C program libraries are owned by you, you can query USER_LIBRARIES, which displays the name of the library (Library_Name), the associated file (File_Spec), whether or not the library is dynamically loadable (Dynamic), and the library's status (Status). ALL_LIBRARIES and DBA_LIBRARIES are also available; they include an additional Owner column to indicate the owner of the library. For further information on libraries, see the entry for the **create library** command in the Alphabetical Reference.

Heterogeneous Services

To support the management of heterogeneous services, Oracle provides a number of data dictionary views. All of the views in this category begin with the letters "HS" instead of "DBA." In general, these views are used primarily by DBAs. For details on the "HS" views, see the *Oracle Database Reference.*

Indextypes and Operators

Operators and indextypes are closely related. You can use the **create operator** command to create a new operator and define its bindings. You can reference operators in indextypes and in SQL statements. The operators, in turn, reference functions, packages, types, and other user-defined objects.

You can query the USER_OPERATORS view to see each operator's Owner, Operator_Name, and Number_of_Binds values. Ancillary information for operators is accessible via USER_OPANCILLARY, and you can query USER_OPARGUMENTS to see the operator arguments. You can query USER_OPBINDINGS to see the operator bindings.

USER_INDEXTYPE_OPERATORS lists the operators supported by indextypes. Indextypes, in turn, are displayed via USER_INDEXTYPES. There are "ALL" and "DBA" views of all the operator and indextype views. As of Oracle Database 10*g*, you can see the indextype arraytypes via the USER_INDEXTYPE_ARRAYTYPES view.

Outlines

When you use stored outlines, you can retrieve the names of, and details for, the outlines via the USER_OUTLINES data dictionary views. To see the hints that make up the outlines, query USER_OUTLINE_HINTS. There are "ALL" and "DBA" versions of USER_OUTLINES and USER_OUTLINE_HINTS.

Advisors

As of Oracle Database 10*g*, users can access data dictionary views related to tuning recommendations. These views begin with "USER_ADVISOR_" and there is a matching set of views at the "DBA_" level, and there are DBA-only advisor views as well (such as DBA_ADVISOR_DEFINITIONS).

For example, USER_ADVISOR_ACTIONS contains data about the actions associated with all recommendations in the database. Each action is specified in the Command column, with six related attribute columns. USER_ADVISOR_LOG displays information about the current state of all the tasks, as well as execution-specific data such as progress monitoring and completion status. Parameters for the advisors are accessible via USER_ADVISOR_PARAMETERS; you can see the rationale for recommendations via the USER_ADVISOR_RATIONALE view.

For the results of an analysis of all recommendations in the database, see the USER_ADVISOR_RECOMMENDATIONS view. A recommendation can have multiple actions associated with it (see USER_ADVISOR_ACTIONS) based on the rationales used.

Schedulers

As of Oracle Database 10*g*, you can access a set of views related to Scheduler jobs in the database. These include USER_SCHEDULER_JOBS (all Scheduler jobs owned by the user), USER_SCHEDULER_PROGRAMS (the scheduler programs), and USER_SCHEDULER_PROGRAM_ARGS (the arguments for the scheduler programs). You can see the execution details for the jobs via the USER_SCHEDULER_JOB_LOG, USER_SCHEDULER_JOB_RUN_DETAILS, and USER_SCHEDULER_RUNNING_JOBS views. You can see the schedules via USER_SCHEDULER_SCHEDULES.