KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University Established Under Section 3 of UGC Act 1956)

Coimbatore - 641021.

(For the candidates admitted from 2018 onwards)

DEPARTMENT OF COMPUTER SCIENCE, CA & IT

SUBJECT : **DATABASE ADMINISTRATION**
SEMESTER : **IV** **L T P C**
SUBJECT CODE : **17CAP404D** **CLASS** : **II MCA** **4 0 0 4**

LECTURE PLAN

STAFF NAME: A.JENNETH

S.No	Lecture Duration (Hr)	Topics	Support Materials
		UNIT-I	
1.	1	Oracle DBA's: The Oracle DBA's Role- Oracle Database 10g Architecture: Oracle Databases and instances-	T1: 3-6
2.	1	Oracle Logical Storage structures ,Oracle Logical Database structures	T1:6-24
3.	1	Oracle Physical Storage structures- Multiplexing Database Files ,Oracle Memory Structures	T1:24-32
4.	1	Oracle Backup and Recovery, Security Capabilities	T1:38-41
5.	1	Tablespace Architecture, Oracle Tablespace installation	T1: 61-75
6.	1	Traditional Disk Space Storage	T1: 76-98
7.	1	Automatic Storage Management Management	T1:98-123
8.	1	Recapitulation and Discussion of important questions	
Text Book	1. **T1→** Bob Bryla, Kevin Loney (2008), "Oracle Database 11g DBA Handbook", McGraw-		

Reference Book	Hill Osborne. **T2→** Saikat Basak.(2010), Oracle DBA Concise Handbook ,Ensel Software
Websites	**W1→** ww.oracle.com/technology/software/products/database/oracle10g/index.html **W2→** www.oracle-base.com/articles/10g/ **W3→** www.adp-gmbh.ch/ora/misc/10g.html

Total No of Hours Planned For Unit – I : 8 Hours			
UNIT-II			
1.	1	Common Space Management Problems	T1:163-166
2.	1	Oracle Segments, Extents and Blocks	T1:166-171
3.	1	Space Management Methodologies	T1:175-182
4.	1	SYSAUX monitoring and usage , Archived Redo Log File Management	T1:182-184
5.	1	Built in Space Management Tools: Segment Advisor , Undo Advisor and the Automatic Workload Repository , Index usage, Space Uasge Warning Levels , Reusable space allocation, Managing lert and Trace Files with ADR.	T1:184-197
6.	1	Transaction Basics ,Undo Basics, Managing Undo Tablespaces	T1:207-223
7.	1	Flashback features	T1:223-231
8.	1	Recapitulation and Discussion of important questions	

Text Book	**T1→** Bob Bryla, Kevin Loney (2008), "Oracle Database 11g DBA Handbook", McGraw-Hill Osborne.
Reference Book	**T2→** Saikat Basak.(2010), Oracle DBA Concise Handbook ,Ensel Software
Websites	**W1 →** www.oracle.com/technology/software/products/database/oracle10g/index.html **W2→** www.oracle-base.com/articles/10g/ **W3→** www.adp-gmbh.ch/ora/misc/10g.html

Total No of Hours Planned For Unit – II : 8 Hours			
UNIT-III			
1.	1	Tuning Application Design , Tuning SQL	T1:241-252
2.	1	Tuning Memory Usage	T1:252-257
3.	1	Tuning Data Access	T1:257-262
4.	1	Tuning Data Manipulation	T1:262-267
5.	1	Tuning Physical Storage –	T1:267-268
6.	1	Reducing Network Security	T1:268-279
7.	1	Database Authentication Methods	T1:279-282
8.	1	Recapitulation and Discussion of important questions	
Text Book **Reference Book** **Websites**	**T1→** Bob Bryla, Kevin Loney (2008), "Oracle Database 11g DBA Handbook", McGraw-Hill Osborne. **T2→** Saikat Basak.(2010), Oracle DBA Concise Handbook ,Ensel Software **W1** → www.oracle.com/technology/software/products/database/oracle10g/index.html **W2→** www.oracle-base.com/articles/10g/ **W3→** www.adp-gmbh.ch/ora/misc/10g.html		
Total No of Hours Planned For Unit – III : 8 Hours			
UNIT-IV			
1.	1	Database Authorization Methods	T1:292-331
2.	1	Auditing: Auditing Locations Statement Auditing – Privilege Auditing ,Schema Object Auditing , Auditing Related Data Dictionary Views	T1:331-339
3.	1	Logical Backups , Physical Backups	T1:391-395
4.	1	Using Data Pump Export and Import –	T1:395-403
5.	1	Data Pump Import Options – Integration of Backup Procedures	T1:403-417

6.	1	Overview of Oracle Net, Using the Oracle Net Configuration Assistant , Using the Oracle Net Manager	T1:511-527
7.	1	Starting the Listener Server Process – Controlling the Listener Server Process ,Using Database links	T1:527-540
8.	1	Recapitulation and Discussion of important questions	
Text Book **Reference Book** **Websites**	**T1→** Bob Bryla, Kevin Loney (2008), "Oracle Database 11g DBA Handbook", McGraw-Hill Osborne. **T2→** Saikat Basak.(2010), Oracle DBA Concise Handbook ,Ensel Software **W1 →** www.oracle.com/technology/software/products/database/oracle10g/index.html **W2→** www.oracle-base.com/articles/10g/ **W3→** www.adp-gmbh.ch/ora/misc/10g.html		
Total No of Hours Planned For Unit – IV 8 Hours			
UNIT-V			
1.	1	Creating Table spaces in a VLDB Environment: Bigfile Table space Basics , Creating and Modifying Bigfile Tablespace Bigfile Tablespace ROWID format DBMS_ROWID and Bigfile Tablespaces	T1:543:550
2.	1	Advanced Oracle Table Types	T1:552:589
3.	1	Using Bitmap Indexes – Oracle Data Pump	T1:589-599
4.	1	Remote queries , Remote Data Manipulation: Two Phase Commit	T1:601-604
5.	1	Managing Distributed Data–	T1:604-630
6.	1	Managing Distributed Transactions	T1:630-632
7.	1	Monitoring Distributed Database	T1:632,W1
8.	1	Tuning Distributed Database	T1:632,W1
9.	1	Recapitulation and Discussion of important questions	
10.	1	**Discussion of previous ESE Question**	

		papers	
11.	1	**Discussion of previous ESE Question papers**	
12.	1	**Discussion of previous ESE Question papers**	
Text Book **Reference Book**	**T1→** Bob Bryla, Kevin Loney (2008), "Oracle Database 11g DBA Handbook", McGraw-Hill Osborne. **T2→** Saikat Basak.(2010), Oracle DBA Concise Handbook ,Ensel Software **W1** → www.oracle.com/technology/software/products/database/oracle10g/index.html **W2→** www.oracle-base.com/articles/10g/ **W3→** www.adp-gmbh.ch/ora/misc/10g.html		
Total No of Hours Planned For Unit – V 12 Hours			
Total No. of Hours Planned: 44			

Text Book:

T1→ Bob Bryla, Kevin Loney (2008), "Oracle Database 11g DBA Handbook", McGraw-Hill Osborne.

T2→ Saikat Basak.(2010), Oracle DBA Concise Handbook ,Ensel Software

WEBSITES

W1 → www.oracle.com/technology/software/products/database/oracle10g/index.html
W2→ www.oracle-base.com/articles/10g/
W3→ www.adp-gmbh.ch/ora/misc/10g.html

UNIT-I

SYLLABUS

Oracle DBA's: The Oracle DBA's Role- Oracle Database 10g Architecture: Oracle Databases and instances- Oracle Logical Storage structures – Oracle Logical Database structures – Oracle Physical Storage structures- Multiplexing Database Files - Oracle Memory Structures-Oracle Backup and Recovery – Security Capabilities – Tablespace Architecture – Oracle Tablespace installation – Traditional Disk Space Storage – Automatic Storage Management

Role of Database Administrator (DBA)

As the primary manager and custodian of the data resource the Oracle DBA has many job duties. Many of the duties of the Oracle DBA have been clouded by the explosive growth in Oracle servers during the 1990's, when UNIX minicomputers caused IT shops to have hundreds of Oracle servers, each with their own instance, and copy of the Oracle software.

Historically the senior Oracle DBA had total responsibility for the Oracle database and had the following job duties:

- Oracle DBA job duty: Install, patch and maintain all Oracle software
- Oracle DBA job duty: Tune all Oracle instance components including SQL and PL/SQL
- Oracle DBA job duty: Approve all production schema changes
- Oracle DBA job duty: Approve changes to database design
- Oracle DBA job duty: Control all migrations of Oracle schema objects
- Oracle DBA job duty: Design and implement a backup & recovery system
- Oracle DBA job duty: Implement Oracle failover technology

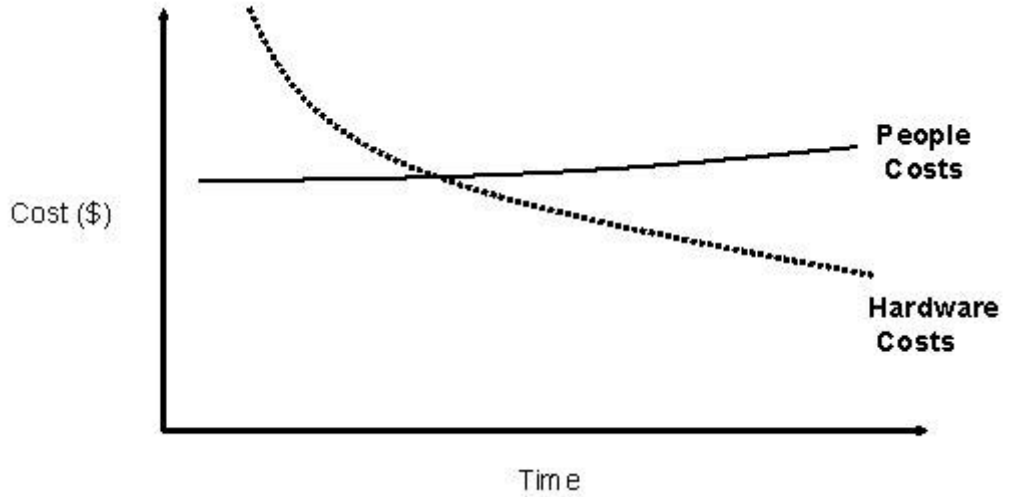
Remember, in many shops the Oracle DBA is a senior manager with high responsibilities and specialized job duties.

The Changing Role of the Oracle DBA

In the late 20th century, shops had dozens of Oracle DBA staff and important tasks were still overlooked because DBAs said "It's not my job," or "I don't have time." Changing technology mandated that the 21st century DBA would have more overall responsibility for the whole operation of their Oracle database.
So, what did this mean to the Oracle DBA of the early 21st century? Clearly, less time was spent installing and maintaining multiple copies of Oracle, and this freed-up time for the DBA to pursue more advanced tasks such as SQL tuning and database performance optimization.

But the sad reality of server consolidation was that thousands of mediocre Oracle DBAs lost their jobs to this trend.

The best DBAs continued to find work, but DBAs who were used for the repetitive tasks of installing upgrades on hundreds of small servers were displaced:



The changing dynamics of human and hardware costs

The surviving Oracle DBAs found that they were relieved of the tedium of applying patches to multiple servers, constantly re-allocating server resources with Oracle Grid control, and constantly monitoring and tuning multiple systems. The DBA job role became far more demanding, and many companies started to view the DBA as a technical management position, encompassing far more responsibility than the traditional DBA.

New Oracle DBA job duties

Consequently, many computer professionals and Oracle DBAs were faced with a new requirement to have degrees in both computer science and business administration. The business administration allowed them to understand the working of internal systems and helped them to design the corporate database.

By 2005, the automation of many of the Oracle DBA functions led to the Oracle professional accepting responsibility for a whole new set of duties:

- Data modeling and Oracle database design
- Data interface protocols
- Managing data security
- Managing development projects
- Predicting future Oracle trends for hardware usage and user load

DBA Responsibilities

The job of the **DBA** seems to be everything that everyone else either doesn't want to do, or doesn't have the ability to do. DBAs get the enviable task of figuring out all of the things no one else can figure out. More seriously though, here is a list of typical DBA responsibilities:

- Installation, configuration and upgrading of Oracle server software and related products
- Evaluate Oracle features and Oracle related products
- Establish and maintain sound backup and recovery policies and procedures
- Take care of the Database design and implementation
- Implement and maintain database security (create and maintain users and roles, assign privileges)
- Perform database tuning and performance monitoring
- Perform application tuning and performance monitoring
- Setup and maintain documentation and standards
- Plan growth and changes (capacity planning)
- Work as part of a team and provide 24x7 support when required
- Perform general technical trouble shooting and give consultation to development teams
- Interface with Oracle Corporation for technical support.
- Patch Management and Version Control
-

DBA Skills Required

- Good understanding of the Oracle database, related utilities and tools
- A good understanding of the underlying operating system
- A good knowledge of the physical database design
- Ability to perform both Oracle and operating system performance tuning and monitoring
- Knowledge of ALL Oracle backup and recovery scenarios
- A good knowledge of Oracle security management
- A good knowledge of how Oracle acquires and manages resources
- A good knowledge Oracle data integrity
- Sound knowledge of the implemented application systems
- Experience in code migration, database change management and data management through the various stages of the development life cycle
- A sound knowledge of both database and system performance tuning
- A DBA should have sound communication skills with management, development teams, vendors and systems administrators
- Provide a strategic database direction for the organisation
- A DBA should have the ability to handle multiple projects and deadlines
- A DBA should possess a sound understanding of the business

DBA Qualifications

- May be certified as an Oracle DBA.
- Preferably a BS in computer science or related engineering field
- Lots and lots of EXPERIENCE
-

Application Database Administrator (ADBA)

Application DBA's or ADBA's are responsible for looking after the application tasks pertaining to a specific application. This includes the creation of database objects, snapshots, SQL tuning, etc.

Typical ADBA responsibilities:

- Implement and maintain the database design
- Create database objects (tables, indexes, etc.)
- Write database procedures, functions and triggers
- Assist developers with database activities
- Tune database queries
- Monitor application related jobs and data replication activities

ORACLE 10G ARCHITECTURE & DIAGRAM

What is a Database?

The Oracle database has a logical layer and a physical layer. The physical layer consists of the files that reside on the disk; the components of the logical layer map the data to these physical components.

The Physical Layer

The physical layer of the database consists of three types of files:
One or more datafiles–Datafiles store the information contained in the database. You can have as few as one datafile or as many as hundreds of datafiles. The information for a single table can span many datafiles or many tables can share a set of datafiles. Spreading tablespaces over many datafiles can have a significant positive effect on performance. The number of datafiles that can be configured is limited by the Oracle parameter MAXDATAFILES.
Two or more redo log files–Redo log files hold information used for recovery in the event of a system failure. Redo log files, known as the redo log, store a log of all changes made to the database. This information is used in the event of a system failure to reapply changes that have been made and committed but that might not have been made to the datafiles. The redo log files must perform well and be protected against hardware failures (through software or hardware fault tolerance). If redo log information is lost, you cannot recover the system.
One or more control files–Control files contain information used to start an instance, such as the location of datafiles and redo log files; Oracle needs this information to start the database instance. Control files must be protected. Oracle provides a mechanism for storing multiple copies of control files.

The Logical Layer

The logical layer of the database consists of the following elements:
One or more tablespaces.
The database schema, which consists of items such as tables, clusters, indexes, views, stored procedures, database triggers, sequences, and so on.

Tablespaces and Datafiles

New Term: The database is divided into one or more logical pieces known as tablespaces. A tablespace is used to logically group data together.

Table–A table, which consists of a tablename and rows and columns of data, is the basic logical storage unit in the Oracle database. Columns are defined by name and data type. A table is stored within a tablespace; often, many tables share a tablespace.

Cluster–A cluster is a set of tables physically stored together as one table that shares a common column. If data in two or more tables is frequently retrieved together based on data in the common column, using a clustered table can be quite efficient. Tables can be accessed separately even though they are part of a clustered table. Because of the structure of the cluster, related data requires much less I/O overhead if accessed simultaneously.

Index–An index is a structure created to help retrieve data more quickly and efficiently (just as the index in this book allows you to find a particular section more quickly). An index is declared on a column or set of columns. Access to the table based on the value of the indexed column(s) (as in a WHERE clause) will use the index to locate the table data.

View–A view is a window into one or more tables. A view does not store any data; it presents table data. A view can be queried, updated, and deleted as a table without restriction. Views are typically used to simplify the user's perception of data access by providing limited information from one table, or a set of information from several tables transparently. Views can also be used to prevent some data from being accessed by the user or to create a join from multiple tables.

Stored procedure–A stored procedure is a predefined SQL query that is stored in the data dictionary. Stored procedures are designed to allow more efficient queries. Using stored procedures, you can reduce the amount of information that must be passed to the RDBMS and thus reduce network traffic and improve performance.

Database trigger–A database trigger is a procedure that is run automatically when an event occurs. This procedure, which is defined by the administrator or developer, triggers, or is run whenever this event occurs. This procedure could be an insert, a deletion, or even a selection of data from a table.

Sequence–The Oracle sequence generator is used to automatically generate a unique sequence of numbers in cache. By using the sequence generator you can avoid the steps necessary to create this sequence on your own such as locking the record that has the last value of the sequence, generating a new value, and then unlocking the record

Segments, Extents, and Data Blocks

Within Oracle, the space used to store data is controlled by the use of logical structures. These structures consist of the following:

Data blocks–A block is the smallest unit of storage in an Oracle database. The database block contains header information concerning the block itself as well as the data.
Extents–Extents consist of data blocks.
Segments–A segment is a set of extents used to store a particular type of data
Segments
An Oracle database can use four types of segments:

Data segment–Stores user data within the database.
Index segment–Stores indexes.
Rollback segment–Stores rollback information used when data must be rolled back.
Temporary segment–Created when a SQL statement needs a temporary work area; these segments are destroyed when the SQL statement is finished. These segments are used during various database operations, such as sorts.

Extents
Extents are the building blocks of segments; in turn, they consist of data blocks. An extent is used to minimize the amount of wasted (empty) storage. As more and more data is entered into tablespaces in your database, the extents used to store that data can grow or shrink as necessary. In this manner, many tablespaces can share the same storage space without preallocating the divisions between those tablespaces.

At tablespace-creation time, you can specify the minimum number of extents to allocate as well as the number of extents to add at a time when that allocation has been used. This arrangement gives you efficient control over the space used in your database.

Data Blocks

Data blocks are the smallest pieces of an Oracle database; they are physically stored on disk. Although the data block in most systems is 2KB (2,048 bytes), you can change this size for efficiency depending on your application or operating system.

Segments
An Oracle database can use four types of segments:
Data segment–Stores user data within the database.

Index segment–Stores indexes.

Rollback segment–Stores rollback information used when data must be rolled back.
Temporary segment–Created when a SQL statement needs a temporary work area; these

segments are destroyed when the SQL statement is finished. These segments are used during various database operations, such as sorts.

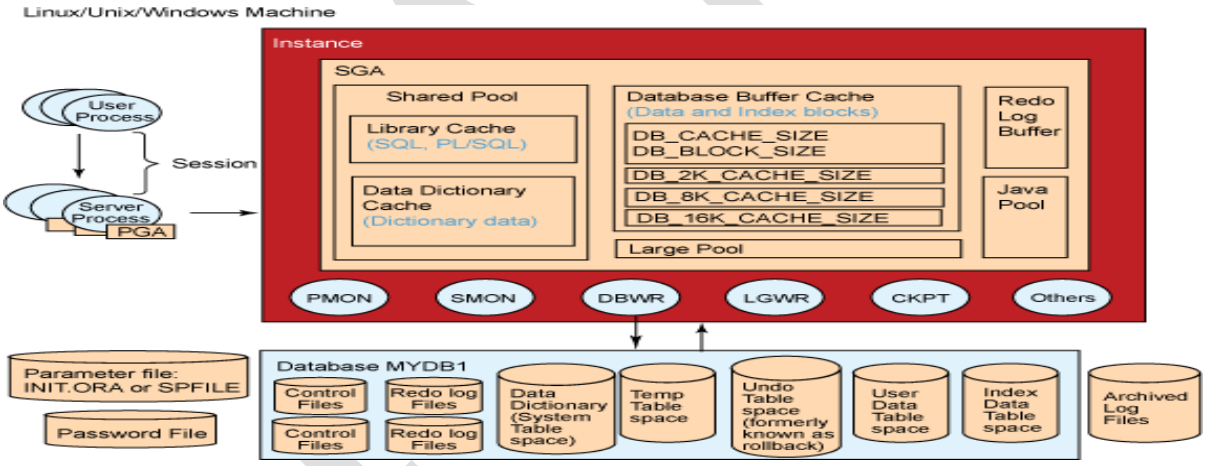
Extents

Extents are the building blocks of segments; in turn, they consist of data blocks. An extent is used to minimize the amount of wasted (empty) storage. As more and more data is entered into tablespaces in your database, the extents used to store that data can grow or shrink as necessary. In this manner, many tablespaces can share the same storage space without preallocating the divisions between those tablespaces.

At tablespace-creation time, you can specify the minimum number of extents to allocate as well as the number of extents to add at a time when that allocation has been used. This arrangement gives you efficient control over the space used in your database.

Data Blocks

Data blocks are the smallest pieces of an Oracle database; they are physically stored on disk. Although the data block in most systems is 2KB (2,048 bytes), you can change this size for efficiency depending on your application or operating system.



The Oracle Instance

The Oracle instance consists of the Oracle processes and shared memory necessary to access information in the database. The instance is made up of the user processes, the Oracle background processes, and the shared memory used by these processes (see Figure Below).

The Oracle Memory Structure

New Term: Oracle uses shared memory for several purposes, including caching of data and indexes as well as storing shared program code. This shared memory is broken into various pieces, or memory structures. The basic memory structures associated with Oracle are the System Global Area (SGA) and the Program Global Area (PGA).

The System Global Area (SGA)

The SGA is a shared memory region that Oracle uses to store data and control information for one Oracle instance. The SGA is allocated when the Oracle instance starts and deallocated when the Oracle instance shuts down. Each Oracle instance that starts has its own SGA. The information in the SGA consists of the following elements, each of which has a fixed size and is created at instance startup:

The database buffer cache–This stores the most recently used data blocks. These blocks can contain modified data that has not yet been written to disk (sometimes known as dirty blocks), blocks that have not been modified, or blocks that have been written to disk since modification (sometimes known as clean blocks). Because the buffer cache keeps blocks based on a most recently used algorithm, the most active buffers stay in memory to reduce I/O and improve performance.

The redo log buffer–This stores redo entries, or a log of changes made to the database. The redo log buffers are written to the redo log as quickly and efficiently as possible. Remember that the redo log is used for instance recovery in the event of a system failure.
The shared pool–This is the area of the SGA that stores shared memory structures such as shared SQL areas in the library cache and internal information in the data dictionary. The shared pool is important because an insufficient amount of memory allocated to the shared pool can cause performance degradation. The shared pool consists of the library cache and the data-dictionary cache.

The Library Cache

The library cache is used to store shared SQL. Here the parse tree and the execution plan for every unique SQL statement are cached. If multiple applications issue the same SQL statement, the shared SQL area can be accessed by each to reduce the amount of memory needed and to reduce the processing time used for parsing and execution planning.

The Data-Dictionary Cache

The data dictionary contains a set of tables and views that Oracle uses as a reference to the database. Oracle stores information here about the logical and physical structure of the database.

The data dictionary contains information such as the following:

- User information, such as user privileges
- Integrity constraints defined for tables in the database

- Names and data types of all columns in database tables
- Information on space allocated and used for schema objects
- The data dictionary is frequently accessed by Oracle for the parsing of SQL statements.
- This access is essential to the operation of Oracle; performance bottlenecks in the data dictionary affect all Oracle users.
- Because of this, you should make sure that the data-dictionary cache is large enough to cache this data.
- If you do not have enough memory for the data-dictionary cache, you see a severe performance degredation.
- If you ensure that you have allocated sufficient memory to the shared pool where the data-dictionary cache resides, you should see no performance problems.

The Program Global Area (PGA)

The PGA is a memory area that contains data and control information for the Oracle server processes. The size and content of the PGA depends on the Oracle server options you have installed. This area consists of the following components:

Stack space–This is the memory that holds the session's variables, arrays, and so on.
Session information–If you are not running the multithreaded server, the session information is stored in the PGA. If you are running the multithreaded server, the session information is stored in the SGA.

Private SQL area–This is an area in the PGA where information such as binding variables and runtime buffers is kept.
Processes
New Term: In many operating systems, traditional processes have been replaced by threads or lightweight processes. The term process is used in this book to describe a thread of execution, or

a mechanism that can execute a set of code; process refers to the mechanism of execution and can refer to a traditional process or a thread.

The Oracle RDBMS uses two types of processes: user processes and Oracle processes (also known as background processes). In some operating systems (such as Windows NT), these processes are actually threads; for the sake of consistency, I will refer to them as processes.

User Processes

User, or client, processes are the user's connections to the RDBMS system. The user process manipulates the user's input and communicates with the Oracle server process through the Oracle program interface. The user process is also used to display the information requested by the user and, if necessary, can process this information into a more useful form.

Oracle Processes

Oracle processes perform functions for users. Oracle processes can be split into two groups: server processes (which perform functions for the invoking process) and background processes (which perform functions on behalf of the entire RDBMS).

Server Processes (Shadow Processes)

Server processes, also known as shadow processes, communicate with the user and interact with Oracle to carry out the user's requests. For example, if the user process requests a piece of data not already in the SGA, the shadow process is responsible for reading the data blocks from the datafiles into the SGA. There can be a one-to-one correlation between user processes and shadow processes (as in a dedicated server configuration); although one shadow process can connect to multiple user processes (as in a multithreaded server configuration), doing so reduces the utilization of system resources.

Background Processes

Background processes are used to perform various tasks within the RDBMS system. These tasks vary from communicating with other Oracle instances and performing system maintenance and cleanup to writing dirty blocks to disk. Following are brief descriptions of the nine Oracle background processes:

DBWR (Database Writer)–DBWR is responsible for writing dirty data blocks from the database block buffers to disk. When a transaction changes data in a data block, that data block need not be immediately written to disk. Therefore, the DBWR can write this data to disk in a manner that is more efficient than writing when each transaction completes. The DBWR usually writes only when the database block buffers are needed for data to be read. Data is written in a least recently

used fashion. For systems in which asynchronous I/O (AIO) is available, there should be only one DBWR process. For systems in which AIO is not available, performance can be greatly enhanced by adding more DBWR processes.

LGWR (Log Writer)–The LGWR process is responsible for writing data from the log buffer to the redo log.

CKPT (Checkpoint)–The CKPT process is responsible for signaling the DBWR process to perform a checkpoint and to update all the datafiles and control files for the database to indicate the most recent checkpoint. A checkpoint is an event in which all modified database buffers are written to the datafiles by the DBWR. The CKPT process is optional. If the CKPT process is not present, the LGWR assumes these responsibilities.

PMON (Process Monitor)–PMON is responsible for keeping track of database processes and cleaning up if a process prematurely dies (PMON cleans up the cache and frees resources that might still be allocated). PMON is also responsible for restarting any dispatcher processes that might have failed.

SMON (System Monitor)–SMON performs instance recovery at instance startup. This includes cleaning temporary segments and recovering transactions that have died because of a system crash. The SMON also defragments the database by coalescing free extents within the database.

RECO (Recovery)–RECO is used to clean transactions that were pending in a distributed database. RECO is responsible for committing or rolling back the local portion of the disputed transactions.

ARCH (Archiver)–ARCH is responsible for copying the online redo log files to archival storage when they become full. ARCH is active only when the RDBMS is operated in ARCHIVELOG mode. When a system is not operated in ARCHIVELOG mode, it might not be possible to recover after a system failure. It is possible to run in NOARCHIVELOG mode under certain circumstances, but typically should operate in ARCHIVELOG mode.

LCKn (Parallel Server Lock)–Up to 10 LCK processes are used for interinstance locking when the Oracle Parallel Server option is used.
Dnnn (Dispatcher)–When the Multithreaded Server option is used, at least one Dispatcher process is used for every communications protocol in use. The Dispatcher process is responsible for routing requests from the user processes to available shared server processes and back.

https://anandoracle.wordpress.com/2012/03/28/oracle-architecture-10g/

Possible Questions

(Each Question Carries 6 Marks)

1. Explain the Role of DBA with suitable diagram.
2. Discuss Oracle database instances with suitable example
3. Explain how to monitor and tune distributed transaction of a student information database.
4. Discuss the architecture of Oracle 11g architecture with suitable diagram.
5. Explain the logical storage structures with suitable example.
6. Explain the types of Oracle Database users with example.
7. Write about the tasks of a Database Administrator.
8. Explain the types of Oracle Database users with example.
9. Discuss the tasks of a Database Administrator.
10. Explain the overview of i)User Global Area ii) Program Global Area.
11. Discuss the Recovery Manager(RMAN) architecture with suitable example.

KARPAGAM ACADEMY OF HIGHER EDUCATION

COIMBATORE - 21

DEPARTMENT: CS,CA & IT

CLASS : II MCA

BATCH : 2018-2020

Part -A Online Examinations **(1 mark questions)**

SUBJECT: Database Administration **SUBJECT CODE: 17CAP404D**

UNIT I

S.no	Questions	opt1	opt 2	opt 3	opt 4	Answer
1	_____ means that the DBA heads off potential trouble by careful monitoring	proactive tuning	active tuning	postactive tuning	infer tuning	proactive tuning
2	The DBA is responible for planning for future growth and potential changes in the applications , is known as ____	implement planning	capacity planning	forward planning	external planning	capacity planning
3	_____ is the process of properly migrating new code	role management	process management	change management	default management	change management
4	_____ refers to database administrators in charge of production of databases	production	development	implement	excution	production
5	DBAs who are involved in the preproduction design and development of database is known as _____	production	development	implement	excution	development
6	Decision support sytem is also known as _____	OLTP	OLAP	OLDP	OLTM	OLAP
7	_____- databse are characterized by heavy transaction volume	OLTP	OLAP	OLDP	OLTM	OLTP
8	_____ range from small database to large database	OLTP	DAM	DSS	OLTM	DSS
9	The _____ are designed to simulate actual production databases and are used to test the functionality of code	production database	test databases	implement database	execution database	test databases
10	An _____ consists of files, both data files and Oracle system files.	production database	test databases	Oracle database	execution database	Oracle database
11	A _____consists of a number of bytes of disk space in the operating system's storage system	extends	data block	segments	tablespaces	data block
12	An _____ is two or more contiguous Oracle data blocks	extends	data block	segments	tablespaces	extends
13	A _____ is a set of extents that you allocate to a logical structure like a table or an index	extends	data block	segments	tablespaces	segments
14	A _____ is a set of one or more data files, and usually consists of related segments.	extends	data block	segments	tablespaces	tablespaces
15	The smallest logical component of an Oracle database is the _____	extends	data block	segments	tablespaces	data block
16	The _____ of data blocks contains the data stored in the tables or their indexes.	free space section	row data section	database section	segment section	free space section
17	The _____ is a purely logical construct and is the primary logical storage structure of an Oracle database.	extends	data block	segments	tablespaces	tablespaces
18	_____ tablespaces are the default in Oracle Database 10g.	locally managed	dictionary managed	database managed	uniform managed	locally managed
19	_____ tablespaces are tablespaces with a single large data file	Bigfile	smallfile	temporary	permanent	Bigfile
20	_____tablespaces can contain multiple data files	Bigfile	smallfile	temporary	permanent	smallfile
21	_____ files store the table and index data.	Bigfile	Data files	temporary	permanent	Data files
22	_____files record changes to all database structures.	Bigfile	Data files	Control files	permanent	Control files
23	_____online files contain the changes made to table data.	Bigfile	Data files	Control files	Redo log files	Redo log files
24	The is an optional file in which you can specify the names of database users who have been granted the special SYSDBA or SYSOPER administrative privileges,	Bigfile	Data files	password file	Redo log files	password file
25	A is essentially a connection or thread to the operating system that performs a task or job.	action	process	compliation	execution	process

26	_____ are responsible for running the application that connects the user to the database instance.	oracle processes	server processes	User processes	database processes	User processes
27	The ____ process is the process that services an individual user process.	server	oracle	background	database	server
28	_____ process Cleans up after finished and failed processes	system monitor	database writer	Process monitor	memory manager	Process monitor
29	The job of the _____ process is to transfer the contents of the redo log buffer to disk.	system monitor	database writer	Process monitor	log writer	log writer
30	The_____ process performs system-monitoring tasks for the Oracle instance,	database writer	Process monitor	system monitor	log writer	system monitor
31	The _____ process collects several types of statistics to help the database manage itself.	database writer	manageability monitor	system monitor	log writer	manageability monitor
32	The _____ purpose is to speed up query performance and to enable a high amount of concurrent database activity.	SGA	PGA	MMA	MMON	SGA
33	_____ are data buffers that are currently in active use by user sessions.	shared buffer	Pinned buffers	dirty buffers	free buffers	Pinned buffers
34	The _____ process coordinates the sizing of the memory components.	SGA	MMAN	PGA	MMON	MMAN
35	The _____ process coordinates disk rebalancing activity when you use an automatic storage management storage system	rebalance master	ASM rebalance	MMON	ASM background	rebalance master
36	_____ is your database name.	ORACLE_HOME	ORACLE_SID	ORACLE_BASE	ORACLE_PATH	ORACLE_SID
37	The _____ parameter turns auditing on or off for the database	AUDIT_SET	AUDIT_FILE	AUDIT_TRAIL	AUDIT_DEST	AUDIT_TRAIL
38	The _____ parameter sets the name of the database.	DB_DOMAIN	DB_INSTANCE	DB_NAME	DB_UNIQUE_NAM	DB_NAME
39	The _____ parameter allows you to use the latest oracle database release.	SERVICE_NAME	INSTANCE_NAME	INSTANCE_TYPE	COMPATIBLE	COMPATIBLE
40	You can use the _____ parameter to embed another initialization file in it.	IFILE	CONTROL_FILES	DB_FILES	USER_FILE	IFILE
41	The _____ parameter specifies the maximum number of database files that can be opened for a database.	IFILE	CONTROL_FILES	DB_FILES	USER_FILE	DB_FILES
42	The _____ parameter specifies the size of the redo log buffer.	LOG_POOL_SIZE	LARGE_POOL_SIZE	LOG_BUFFER	BUFFER_SIZE	LOG_BUFFER
43	_____ parameter specifies the default filename format for the archieved redo log files.	LOG_ARCHIVE_FORM	LOG_FORMAT	LOG_ARCHIVE	LOG_BUFFER	LOG_ARCHIVE_FORMAT
44	_____ parameter specifies the maximum number of users you can create in your database	LICENSE_USERS	LICENSE_DB	LICENSE_MAX_US	LICENSE_LEVEL	LICENSE_MAX_USERS
45	The file is called _____ file because it is always maintained on the machine where the oracle database server is located.	oracle	server	init	orcl	server
46	Which of the following is a package?	put_line	serveroutput	dbms_output	dbms_input	dbms_output
47	You can use the _____ parameter to embed another initialization file in it.	ifile	db_domain	compatible	dispatchers	ifile
48	You can create a new init.ora file from the SPFILE in the default location by using the following command:	create init.ora from	create pfile from spfile;	create init from spfile	create pfile from	create pfile from spfile;
49	_____ is the command to suspend the database.	alter system quiesce	alter database quiesce	alter database	Alter system	Alter system suspend
50	____ is the continuation character in SQL.	hypen	comma	semicolon	dot	hypen
51	_____ commands are executed locally and are not sent to the server.	server executed	SQL*Plus	Local	Create table	Local
52	____ is used to see the cureently logged in user name.	show user	show sga	show recyclebin	show errors	show user
53	____ command enables you to save the output of one or more SQL statements to an OS file.	prompt	Execute	Spool	Pause	Spool
54	_____ command enables you to create your own variables.	declare	Variable	Create	Define	Define

55	____ parameter specifies both the username and the password of the user in the database .	userid	userpassword	userpw	userlogin	userid

UNIT-II

SYLLABUS

Common Space Management Problems – Oracle Segments, Extents and Blocks – Space Management Methodologies – SYSAUX monitoring and usage – Archived Redo Log File Management – Built in Space Management Tools: Segment Advisor – Undo Advisor and the Automatic Workload Repository – Index usage – Space Usage Warning Levels – Reusable space allocation – Managing alert and Trace Files with ADR – Transaction Basics – Undo Basics – Managing Undo Tablespaces – Flashback features

Common Space Management Problems

Space management problems generally fall into one of three categories: running out of space in a regular tablespace, not having enough undo space for long-running queries that need a consistent "before" image of the tables, and insufficient space for temporary segments. Although we may still have some fragmentation issues within a database object such as a table or index, locally managed tablespaces solve the problem of tablespace fragmentation.

Running Out of Free Space in a Tablespace

If a tablespace is not defined with the AUTOEXTEND attribute, then the total amount of space in all the datafiles that compose the tablespace limits the amount of data that can be stored in the tablespace. If the AUTOEXTEND attribute is defined, then one or more of the datafiles that compose the tablespace will grow to accommodate the requests for new segments or the growth of existing segments. Even with the AUTOEXTEND attribute, the amount of space in the tablespace is ultimately limited by the amount of disk space on the physical disk drive or storage group.

The AUTOEXTEND attribute is the default if you don't specify the SIZE parameter in the **create tablespace** command and you are using OMF, so you'll actually have to go out of your way to prevent a datafile from autoextending. In Oracle Database 11*g* with the initialization parameter DB_CREATE_FILE_DEST set to an ASM or file system location, you can run a **create tablespace** command like this:

create tablespace bi_02;

In this case, the tablespace BI_02 is created with a size of 100MB in a single datafile, AUTOEXTEND is on, and the next extent is 100MB when the first datafile fills up. In addition,extent management is set to LOCAL, space allocation is AUTOALLOCATE, and segment space management set to AUTO.

The conclusion to be reached here is that we want to monitor the free and used space within a tablespace to detect trends in space usage over time, and as a result be proactive in making sure that enough space is available for future space requests. As of Oracle Database 10*g*, you can use the DBMS_SERVER_ALERT package to automatically notify you when a tablespace reaches a warning or critical space threshold level, either at a percent used, space remaining, or both.

Insufficient Space for Temporary Segments

A *temporary segment* stores intermediate results for database operations such as sorts, index builds, **distinct** queries, **union** queries, or any other operation that necessitates a sort/merge operation that cannot be performed in memory. Under no circumstances should the SYSTEM tablespace be used for temporary segments; when the database is created, a non-SYSTEM tablespace should be specified as a default

temporary tablespace for users who are not otherwise assigned a temporary tablespace. If the SYSTEM tablespace is locally managed, a default temporary tablespace must be defined when the database is created.

When there is not enough space available in the user's default temporary tablespace, and either the tablespace cannot be autoextended or the tablespace's AUTOEXTEND attribute is disabled, the user's query or DML statement fails.

Too Much or Too Little Undo Space Allocated

As of Oracle9*i*, undo tablespaces have simplified the management of rollback information by managing undo information automatically within the tablespace. The DBA no longer has to define the number and size of the rollback segments for the kinds of activity occurring in the database.

As of Oracle 10*g*, manual rollback management has been deprecated. Not only does an undo segment allow a rollback of an uncommitted transaction, it provides for read consistency of long-running queries that begin before inserts, updates, and deletes occur on a table. The amount of undo space available for providing read consistency is under the control of the DBA and is specified as the number of seconds that Oracle will attempt to guarantee that "before" image data is available for long-running queries.

As with temporary tablespaces, we want to make sure we have enough space allocated in an undo tablespace for peak demands without allocating more than is needed. As with any tablespace, we can use the AUTOEXTEND option when creating the tablespace to allow for unexpected growth of the tablespace without reserving too much disk space up front.

Fragmented Tablespaces and Segments

As of Oracle8*i*, a tablespace that is locally managed uses bitmaps to keep track of free space, which,in addition to eliminating the contention on the data dictionary, eliminates wasted space because all extents are either the same size (with uniform extent allocation) or are multiples of the smallest size (with autoallocation). For migrating from a dictionary-managed tablespace, we will review an example that converts a dictionary-managed tablespace to a locally managed tablespace. In a default installation of Oracle Database 10*g* or Oracle Database 11*g* using the Database Creation Assistant (DBCA), all tablespaces, including the SYSTEM and SYSAUX tablespaces, are created as locally managed tablespaces.

Even though locally managed tablespaces with automatic extent management (using the **autoallocate** clause) are created by default when you use **create tablespace**, you still need to specify **extent management local** if you need to specify **uniform** for the extent management type in the **create tablespace** statement:

```
SQL> create tablespace USERS4
2 datafile '+DATA'
3 size 250M autoextend on next 250M maxsize 2000M
4 extent management local uniform size 8M
5 segment space management auto;
Tablespace created.
```

This tablespace will be created with an initial size of 250MB, and it can grow as large as 2000MB (2GB); extents will be locally managed with a bitmap, and every extent in this tablespace will be exactly 8MB in size. Space within each segment (table or index) will be managed automatically with a bitmap instead of freelists.

Even with efficient extent allocation, table and index segments may eventually contain a lot of free space due to **update** and **delete** statements. As a result, a lot of unused space can be reclaimed by using some of the scripts I provide later in this chapter, as well as by using the Oracle Segment Advisor.

Oracle Segments, Extents, and Blocks

At the lowest level, a tablespace segment consists of one or more extents, each extent comprising one or more data blocks.

Data Blocks

A *data block* is the smallest unit of storage in the database. Ideally, an Oracle block is a multiple of the operating system block to ensure efficient I/O operations. The default block size for the database is specified with the DB_BLOCK_SIZE initialization parameter; this block size is used for the SYSTEM, TEMP, and SYSAUX tablespaces at database creation and cannot be changed without re-creating the database.

Every data block contains a *header* that specifies what kind of data is in the block—table rows or index entries. The *table directory* section has information about the table with rows in the block;a block can have rows from only one table or entries from only one index, unless the table is a clustered table, in which case the table directory identifies all the tables with rows in this block.

The *row directory* provides details of the specific rows of the table or index entries in the block.

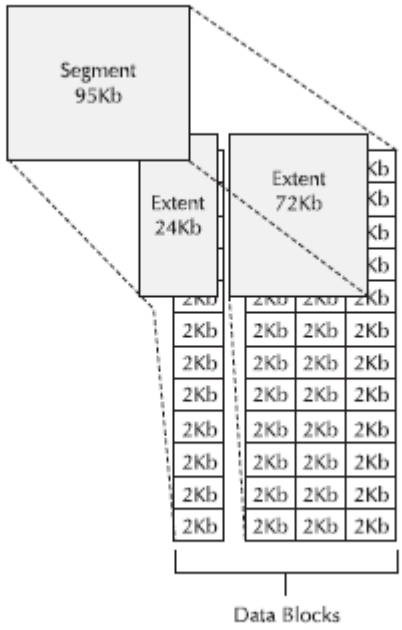


FIGURE 6-1 *Oracle segments, extents, and blocks*

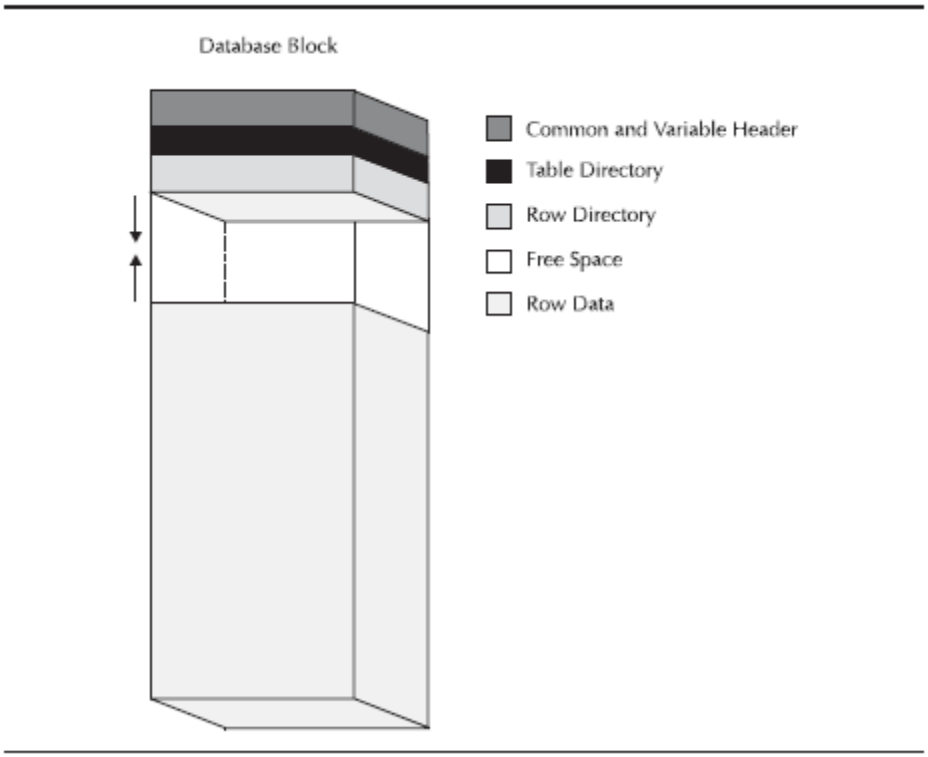


FIGURE 6-2 *Contents of an Oracle data block*

The space for the header, table directory, and row directory is a very small percentage of the space allocated for a block; our focus, then, is on the *free space* and *row data* within the block.

Within a newly allocated block, free space is available for new rows and updates to existing rows; the updates may increase or decrease the space allocated for the row if there are varyinglength columns in the row or a non-NULL value is changed to a NULL value, or vice versa. Space is available within a block for new inserts until there is less than a certain percentage of space available in the block defined by the PCTFREE parameter, specified when the segment is created.

Once there is less than PCTFREE space in the block, no inserts are allowed. If freelists are used to manage space within the blocks of a segment, then new inserts are allowed on the table when used space within the block falls below PCTUSED.

A row may span more than one block if the row size is greater than the block size or an updated row no longer fits into the original block. In the first case, a row that is too big for a block is stored in a *chain* of blocks; this may be unavoidable if a row contains columns that exceed even the largest block size allowed, which in Oracle 11*g* is 32KB.

Oracle will *migrate* the data for the entire row to a new block and leave a pointer in the first block to point to the location in the second block where the updated row is stored. As you may infer, a segment with many migrated rows may cause I/O performance problems because the number of blocks required to satisfy a query can double. In some cases, adjusting the value of PCTFREE or rebuilding the table may result in better space utilization and I/O performance.

Starting with Oracle9*i* Release 2, you can use Automatic Segment Space Management (ASSM) to manage free space within blocks; you enable ASSM in locally managed tablespaces by using the **segment space management auto** keywords in the **create tablespace** command (although this is the default for locally managed tablespaces).

Using ASSM reduces segment header contention and improves simultaneous insert concurrency;this is because the free space map in a segment is spread out into a bitmap block within each extent of the

segment. As a result, you dramatically reduce waits because each process performing **insert**, **update**, or **delete** operations will likely be accessing different blocks instead of one freelist or one of a few freelist groups. In addition, each extent's bitmap block lists each block within the extent along with a four-bit "fullness" indicator defined as follows (with room for future expansion from values 6–15):

0000 Unformatted block
0001 Block full
0010 Less than 25 percent free space available
0011 25 percent to 50 percent free space
0100 50 percent to 75 percent free space
0101 Greater than 75 percent free space

In a RAC database environment, using ASSM segments means you no longer need to create multiple freelist groups. In addition, you no longer need to specify PCTUSED, FREELISTS, or FREELIST GROUPS parameters when you create a table; if you specify any of these parameters, they are ignored.

Extents

An *extent* is the next level of logical space allocation in a database; it is a specific number of blocks allocated for a specific type of object, such as a table or index. An extent is the minimum number of blocks allocated at one time; when the space in an extent is full, another extent is allocated.

When a table is created, an *initial* extent is allocated. Once the space is used in the initial extent, *incremental* extents are allocated. In a locally managed tablespace, these subsequent extents can either be the same size (using the UNIFORM keyword when the tablespace is created) or optimally sized by Oracle (AUTOALLOCATE). For extents that are optimally sized, Oracle starts with a minimum extent size of 64KB and increases the size of subsequent extents as multiples of the initial extent as the segment grows. In this way, fragmentation of the tablespace is virtually eliminated.

When the extents are sized automatically by Oracle, the storage parameters INITIAL, NEXT, PCTINCREASE, and MINEXTENTS are used as a guideline, along with Oracle's internal algorithm, to determine the best extent sizes. In the following example, a table created in the USERS tablespace (during installation of a new database, the USERS tablespace is created with AUTOALLOCATE enabled) does not use the storage parameters specified in the **create table** statement:

```
SQL> create table t_autoalloc (c1 char(2000))
2 storage (initial 1m next 2m pctincrease 50)
3 tablespace users;
Table created.
SQL> begin
2 for i in 1..3000 loop
3 insert into t_autoalloc values ('a');
4 end loop;
5 end;
6 /
PL/SQL procedure successfully completed.
SQL> select segment_name, extent_id, bytes, blocks
2 from user_extents where segment_name = 'T_AUTOALLOC';
SEGMENT_NAME EXTENT_ID BYTES BLOCKS
------------ ---------- ---------- ----------
T_AUTOALLOC 0 65536 8
T_AUTOALLOC 1 65536 8
. . .
T_AUTOALLOC 15 65536 8
```

T_AUTOALLOC 16 1048576 128
. . .
T_AUTOALLOC 22 1048576 128
23 rows selected.

Unless a table is truncated or the table is dropped, any blocks allocated to an extent remain allocated for the table, even if all rows have been deleted from the table. The maximum number of blocks ever allocated for a table is known as the *high-water mark (HWM).*

Segments

Groups of extents are allocated for a single *segment.* A segment must be wholly contained within one and only one tablespace. Every segment represents one and only one type of database object, such as a table, a partition of a partitioned table, an index, or a temporary segment. For partitioned tables, every partition resides in its own segment; however, a cluster (with two or more tables) resides within a single segment. Similarly, a partitioned index consists of one segment for each index partition.

Temporary segments are allocated in a number of scenarios. When a sort operation cannot fit in memory, such as a **select** statement that needs to sort the data to perform a **distinct**, **group by**, or **union** operation, a temporary segment is allocated to hold the intermediate results of the sort. Index creation also typically requires the creation of a temporary segment. Because allocation and deallocation of temporary segments occurs often, it is highly desirable to create a tablespace specifically to hold temporary segments. This helps to distribute the I/O required for a given operation, and it reduces the possibility that fragmentation may occur in other tablespaces due to the allocation and deallocation of temporary segments. When the database is created, a *default temporary tablespace* can be created for any new users who do not have a specific temporary tablespace assigned; if the SYSTEM tablespace is locally managed, a separate temporary tablespace must be created to hold temporary segments.

How space is managed within a segment depends on how the tablespace containing the block is created. If the tablespace is dictionary managed, the segment uses freelists to manage space within the data blocks; if the tablespace is locally managed, space in segments can be managed with either freelists or bitmaps. Oracle strongly recommends that all new tablespaces be created as locally managed and that free space within segments be managed automatically with bitmaps. Automatic segment space management allows more concurrent access to the bitmap lists in a segment compared to freelists; in addition, tables that have widely varying row sizes make more efficient use of space in segments that are automatically managed.

As I mentioned earlier, in the section titled "Data Blocks," if a segment is created with automatic segment space management, bitmaps are used to manage the space within the segment. As a result, the **pctused**, **freelist**, and **freelist groups** keywords within a **create table** or **create index** statement are ignored. The three-level bitmap structure within the segment indicates whether blocks below the HWM are full (less than **pctfree**), 0 to 25 percent free, 25 to 50 percent free, 50 to 75 percent free, 75 to 100 percent free, or unformatted.

Space Management Methodologies

In the following sections, we will consider various features of Oracle 11*g* to facilitate the efficient use of disk space in the database. Locally managed tablespaces offer a variety of advantages to the DBA, improving the performance of the objects within the tablespace, as well as easing administration of the tablespace—fragmentation of a tablespace is a thing of the past. Another feature introduced in Oracle9*i*, Oracle Managed Files, eases datafile maintenance by automatically removing files at the operating system level when a tablespace or other database object is dropped. Bigfile tablespaces, introduced in Oracle 10*g*, simplify datafile management because one and only one datafile is associated with a bigfile tablespace. This moves the maintenance point up one level, from the datafile to the tablespace. We'll also review a couple other features introduced in

Oracle9*i*—undo tablespaces and multiple block sizes.

Locally Managed Tablespaces

Prior to Oracle8*i*, there was only one way to manage free space within a tablespace—by using data dictionary tables in the SYSTEM tablespace. If a lot of insert, delete, and update activity occurs anywhere in the database, there is the potential for a "hot spot" to occur in the SYSTEM tablespace where the space management occurs. Oracle removed this potential bottleneck by introducing *locally managed tablespaces (LMTs).* A locally managed tablespace tracks free space in the tablespace with bitmaps, as discussed in Chapter 1. These bitmaps can be managed very efficiently because they are very compact compared to a freelist of available blocks. Because they are stored within the tablespace itself, instead of in the data dictionary tables, contention in the SYSTEM tablespace is reduced.

As of Oracle 10*g*, by default, all tablespaces are created as locally managed tablespaces, including the SYSTEM and SYSAUX tablespaces. When the SYSTEM tablespace is locally managed, you can no longer create any dictionary-managed tablespaces in the database that are read/write. A dictionary-managed tablespace may still be plugged into the database from an earlier version of Oracle, but it is read-only.

An LMT can have objects with one of two types of extents: automatically sized or all of a uniform size. If extent allocation is set to UNIFORM when the LMT is created, all extents, as expected, are the same size. Because all extents are the same size, there can be no fragmentation. Gone is the classic example of a 51MB segment that can't be allocated in a tablespace with two free 50MB extents because the two 50MB extents are not adjacent.

On the other hand, automatic segment extent management within a locally managed tablespace allocates space based on the size of the object. Initial extents are small, and if the object stays small, very little space is wasted. If the table grows past the initial extent allocated for the segment, subsequent extents to the segment are larger. Extents in an autoallocated LMT have sizes of 64KB, 1MB, 8MB, and 64MB, and the extent size increases as the size of the segment increases, up to a maximum of 64MB. In other words, Oracle is specifying what the values of INITIAL, NEXT, and PCTINCREASE are automatically, depending on how the object grows. Although it seems like fragmentation can occur in a tablespace with autoallocation, in practice the fragmentation is minimal because a new object with a 64KB initial segment size will fit nicely in a 1MB, 4MB, 8MB, or 64MB block preallocated for all other objects with an initial 64KB extent size.

Given an LMT with either automatically managed extents or uniform extents, the free space within the segment itself can be AUTO or MANUAL. With AUTO segment space management, a bitmap is used to indicate how much space is used in each block. The parameters PCTUSED, FREELISTS, and FREELIST GROUPS no longer need to be specified when the segment is created. In addition, the performance of concurrent DML operations is improved because the segment's bitmap allows concurrent access. In a freelist-managed segment, the data block in the segment header that contains the freelist is locked out to all other writers of the block when a single writer is looking for a free block in the segment. Although allocating multiple freelists for very active segments does somewhat solve the problem, it is another structure that the DBA has to manage.

Another advantage of LMTs is that rollback information is reduced or eliminated when any LMT space-related operation is performed. Because the update of a bitmap in a tablespace is not recorded in a data dictionary table, no rollback information is generated for this transaction.

Other than third-party applications, such as older versions of SAP that require dictionarymanaged tablespaces, there are no other reasons for creating new dictionary-managed tablespaces in Oracle 10*g.* As mentioned earlier, compatibility is provided in part to allow dictionary-managed tablespaces from previous versions of Oracle to be "plugged into" an Oracle 11*g* database, although if the SYSTEM tablespace is locally managed, any dictionary-managed tablespaces must be opened read-only. Later in this chapter, you'll see some examples where we can optimize space and

performance by moving a tablespace from one database to another and allocating additional data buffers for tablespaces with different sizes.

Migrating a dictionary-managed tablespace to a locally managed tablespace is very straightforward using the DBMS_SPACE_ADMIN built-in package:

execute sys.dbms_space_admin.tablespace_migrate_to_local('USERS')

After upgrading a database to either Oracle9*i*, Oracle 10*g*, or Oracle 11*g*, you may also want to consider migrating the SYSTEM tablespace to an LMT; if so, a number of prerequisites are in order:

Before starting the migration, shut down the database and perform a cold backup of the database.

Any non-SYSTEM tablespaces that are to remain read/write should be converted to LMTs.

The default temporary tablespace must not be SYSTEM.

If automatic undo management is being used, the undo tablespace must be online.

For the duration of the conversion, all tablespaces except for the undo tablespace must be set to read-only.

The database must be started in RESTRICTED mode for the duration of the conversion.

If any of these conditions are not met, the TABLESPACE_MIGRATE_TO_LOCAL procedure will not perform the migration.

Using OMF to Manage Space

In a nutshell, *Oracle-Managed Files (OMF)* simplifies the administration of an Oracle database. At database-creation time, or later by changing a couple parameters in the initialization parameter file, the DBA can specify a number of default locations for database objects such as datafiles, redo log files, and control files. Prior to Oracle9*i*, the DBA had to remember where the existing datafiles were stored by querying the DBA_DATA_FILES and DBA_TEMP_FILES views. On many occasions, a DBA would drop a tablespace, but would forget to delete the underlying datafiles, thus wasting space and the time it took to back up files that were no longer used by the database.

Using OMF, Oracle not only automatically creates and deletes the files in the specified directory location, it ensures that each filename is unique. This avoids corruption and database downtime in a non-OMF environment due to existing files being overwritten by a DBA inadvertently creating a new datafile with the same name as an existing datafile, and using the REUSE clause.

In a test or development environment, OMF reduces the amount of time the DBA must spend on file management and lets him or her focus on the applications and other aspects of the testdatabase.

OMF has an added benefit for packaged Oracle applications that need to create tablespaces: The scripts that create the new tablespaces do not need any modification to include a datafile name, thus increasing the likelihood of a successful application deployment.

Migrating to OMF from a non-OMF environment is easy, and it can be accomplished over a longer time period. Non-OMF files and OMF files can coexist indefinitely in the same database. When the appropriate initialization parameters are set, all new datafiles, control files, and redo log files can be created as OMF files, while the previously existing files can continue to be managed manually until they are converted to OMF, if ever.

The OMF-related initialization parameters are detailed in Table 6-1. Note that the operating system path specified for any of these initialization parameters must already exist; Oracle will not create the directory. Also, these directories must be writable by the operating system account that owns the Oracle software (which on most platforms is oracle).

Initialization Parameter	Description
DB_CREATE_FILE_DEST	The default operating system file directory where datafiles and tempfiles are created if no pathname is specified in the **create tablespace** command. This location is used for redo log files and control files if DB_CREATE_ONLINE_LOG_DEST_*n* is not specified.
DB_CREATE_ONLINE_LOG_DEST_*n*	Specifies the default location to store redo log files and control files when no pathname is specified for redo log files or control files at database-creation time. Up to five destinations can be specified with this parameter, allowing up to five multiplexed control files and five members of each redo log group.
DB_RECOVERY_FILE_DEST	Defines the default pathname in the server's file system where RMAN backups, archived redo logs, and flashback logs are located. Also used for redo log files and control files if neither DB_CREATE_FILE_DEST nor DB_CREATE_ONLINE_LOG_DEST_*n* is specified.

TABLE 6-1 *OMF-Related Initialization Parameters*

Bigfile Tablespaces

Bigfile tablespaces, introduced in Oracle 10*g*, take OMF files to the next level; in a bigfile tablespace, a single datafile is allocated, and it can be up to 8EB (exabytes, a million terabytes) in size. Bigfile tablespaces can only be locally managed with automatic segment space management. If a bigfile tablespace is used for automatic undo or for temporary segments, then segment space management must be set to MANUAL.

Bigfile tablespaces can save space in the System Global Area (SGA) and the control file because fewer datafiles need to be tracked; similarly, all **alter tablespace** commands on bigfile tablespaces need not refer to datafiles because one and only one datafile is associated with each bigfile tablespace. This moves the maintenance point from the physical (datafile) level to the logical (tablespace) level, simplifying administration. One downside to bigfile tablespaces is that a backup of a bigfile tablespace uses a single process; a number of smaller tablespaces, however, can be backed up using parallel processes and will most likely take less time to back up than a single bigfile tablespace.

Creating a bigfile tablespace is as easy as adding the **bigfile** keyword to the **create tablespace** command:

```
SQL> create bigfile tablespace whs01
2 datafile '/u06/oradata/whs01.dbf' size 10g;
Tablespace created.
```

If you are using OMF, then the **datafile** clause can be omitted. To resize a bigfile tablespace, you can use the **resize** clause:

```
SQL> alter tablespace whs01 resize 80g;
Tablespace altered.
```

In this scenario, even 80GB is not big enough for this tablespace, so we will let it autoextend 20GB at a time:

```
SQL> alter tablespace whs01 autoextend on next 20g;
Tablespace altered.
```

Notice in both cases that we do not need to refer to a datafile; there is only one datafile, and once the tablespace is created, we no longer need to worry about the details of the underlying datafile

and how it is managed.
Bigfile tablespaces are intended for use with Automatic Storage Management, discussed in the next section.

Automatic Storage Management

Using *Automatic Storage Management (ASM)* can significantly reduce the administrative overhead of managing space in a database because a DBA need only specify an ASM *disk group* when allocating space for a tablespace or other database object. Database files are automatically distributed among all available disks in a disk group, and the distribution is automatically updated whenever the disk configuration changes. For example, when a new disk volume is added to an existing disk group in an ASM instance, all datafiles within the disk group are redistributed to use the new disk volume. I introduced ASM in Chapter 4. In this section, I'll revisit some other key ASM concepts from a storage management point of view and provide more examples.
Because ASM automatically places datafiles on multiple disks, performance of queries and DML statements is improved because the I/O is spread out among several disks. Optionally, the disks in an ASM group can be mirrored to provide additional redundancy and performance benefits. Using ASM provides a number of other benefits. In many cases, an ASM instance with a number of physical disks can be used instead of a third-party volume manager or network-attached storage (NAS) subsystem. As an added benefit over volume managers, ASM maintenance operations do not require a shutdown of the database if a disk needs to be added or removed from a disk group.

SYSAUX Monitoring and Usage

The SYSAUX tablespace, introduced in Oracle 10*g*, is an auxiliary tablespace to the SYSTEM tablespace, and it houses data for several components of the Oracle database that either required their own tablespace or used the SYSTEM tablespace in previous releases of Oracle. These components include the Enterprise Manager Repository, formerly in the tablespace OEM_ REPOSITORY, as well as LogMiner, Oracle Spatial, and Oracle Text, all of which formerly used the SYSTEM tablespace for storing configuration information. The current occupants of the SYSAUX tablespace can be identified by querying the V$SYSAUX_OCCUPANTS view:

```
SQL> select occupant_name, occupant_desc, space_usage_kbytes
2 from v$sysaux_occupants;

OCCUPANT_NAME OCCUPANT_DESC                                    SPACE_USAGE_KBYTES
------------- ---------------------------------------- ------------------
LOGMNR        LogMiner                                                  7744
LOGSTDBY      Logical Standby                                            960
SMON_SCN_TIME Transaction Layer - SCN to TIME mapping                   3328
PL/SCOPE      PL/SQL Identifier Collection                               384
STREAMS       Oracle Streams                                            1024
XDB           XDB                                                      98816
AO            Analytical Workspace Object Table                        38208
XSOQHIST      OLAP API History Tables                                  38208
XSAMD         OLAP Catalog                                             15936
SM/AWR        Server Manageability - Automatic Workloa                131712
              d Repository
SM/ADVISOR    Server Manageability - Advisor Framework                  13248
SM/OPTSTAT    Server Manageability - Optimizer Statist                  52672
              ics History
SM/OTHER      Server Manageability - Other Components                    6016
STATSPACK     Statspack Repository                                         0
SDO           Oracle Spatial                                           47424
WM            Workspace Manager                                         7296
ORDIM         Oracle interMedia ORDSYS Components                      11200
```

```
ORDIM/PLUGINS Oracle interMedia ORDPLUGINS Components                0
ORDIM/SQLMM   Oracle interMedia SI_INFORMTN_SCHEMA Com               0
              ponents
EM            Enterprise Manager Repository                     155200
TEXT          Oracle Text                                         5568
ULTRASEARCH   Oracle Ultra Search                                  7616
ULTRASEARCH_D Oracle Ultra Search Demo User                       12288
EMO_USER
EXPRESSION_FI Expression Filter System                            3968
LTER
EM_MONITORING Enterprise Manager Monitoring User                  1536
_USER
TSM           Oracle Transparent Session Migration User            256
SQL_MANAGEMEN SQL Management Base Schema                          1728
T_BASE
AUTO_TASK     Automated Maintenance Tasks                          320
JOB_SCHEDULER Unified Job Scheduler                                576

29 rows selected.
```

components of the Oracle database will be unavailable; the core functionality of the database will be unaffected. In any case, the SYSAUX tablespace helps to take the load off of the SYSTEM tablespace during normal operation of the database.

To monitor the usage of the SYSAUX tablespace, you can query the column SPACE_USAGE_ KBYTES on a routine basis, and it can alert the DBA when the space usage grows beyond a certain level. If the space usage for a particular component requires a dedicated tablespace to be allocated for the component, such as for the EM Repository, the procedure identified in the MOVE_ PROCEDURE column of the V$SYSAUX_OCCUPANTS view will move the application to another tablespace:

SQL> **select occupant_name, move_procedure from v$sysaux_occupants**
2 **where occupant_name = 'EM';**
OCCUPANT_NAME MOVE_PROCEDURE
--- ---
EM emd_maintenance.move_em_tblspc

In the following scenario, we know that we will be adding several hundred nodes to our management repository in the near future. Because we want to keep the SYSAUX tablespace from growing too large, we decide to create a new tablespace to hold only the Enterprise Manager data. In the following example, we'll create a new tablespace and move the Enterprise Manager schema into the new tablespace:

SQL> **create tablespace EM_REP**
2> **datafile '+DATA' size 250m autoextend on next 100m;**
Tablespace created.
SQL> **execute sysman.emd_maintenance.move_em_tblspc('EM_REP');**
PL/SQL procedure successfully completed.
SQL> **select occupant_name, occupant_desc, space_usage_kbytes**
2> **from v$sysaux_occupants**
3> **where occupant_name = 'EM';**
OCCUPANT_NAME OCCUPANT_DESC SPACE_USAGE_KBYTES
--- --- ---
EM Enterprise Manager Repository 0
1 row selected.

Since the current space allocation for the EM tools is about 150MB, a tablespace starting at a size of 250MB with additional extents of 100MB each should be sufficient for most environments. Note that the row for Enterprise Manager is still in V$SYSAUX_OCCUPANTS; even though it is

not taking up any space in the SYSAUX tablespace, we may want to move its metadata back into SYSAUX at some point in the future. Therefore, we may need to query V$SYSAUX_OCCUPANTS again to retrieve the move procedure. We use the same procedure for moving the application into and out of SYSAUX:

SQL> **execute sysman.emd_maintenance.move_em_tblspc('SYSAUX');**

PL/SQL procedure successfully completed.

If a component is not being used in the database at all, such as Ultra Search, a negligible amount of space is used in the SYSAUX tablespace.

Archived Redo Log File Management

It is important to consider space management for objects that exist outside of the database, such as archived redo log files. In ARCHIVELOG mode, an online redo log file is copied to the destination(s) specified by LOG_ARCHIVE_DEST_*n* (where *n* is a number from 1 to 10) or by DB_RECOVERY_ FILE_DEST (the flash recovery area) if none of the LOG_ARCHIVE_DEST_*n* values are set. The redo log being copied must be copied successfully to at least one of the destinations before it can be reused by the database. The LOG_ARCHIVE_MIN_SUCCEED_DEST parameter defaults to 1 and must be at least 1. If none of the copy operations are successful, the database will be suspended until at least one of the destinations receives the log file. Running out of disk space is one possible reason for this type of failure.

If the destination for the archived log files is on a local file system, an operating system shell script can monitor the space usage of the destination, or it can be scheduled with DBMS_ SCHEDULER or with OEM.

Built-in Space Management Tools

Oracle 10*g* provides a number of built-in tools that a DBA can use on demand to determine if there are any problems with disk space in the database. Most, if not all, of these tools can be manually configured and run by calling the appropriate built-in package. In this section, we'll cover the packages and procedures used to query the database for space problems or advice on space management. In addition, I'll show you the new initialization parameter used by the Automatic Diagnostic Repository to identify the alert and trace file location. Later in this chapter, you'll see how some of these tools can be automated to notify the DBA via e-mail or pager when a problem is imminent; many, if not all, of these tools are available on demand via the EM Database Control web interface.

Segment Advisor

Frequent inserts, updates, and deletes on a table may, over time, leave the space within a table fragmented. Oracle can perform *segment shrink* on a table or index. Shrinking the segment makes the free space in the segment available to other segments in the tablespace, with the potential to improve future DML operations on the segment because fewer blocks may need to be retrieved for the DML operation after the segment shrink. Segment shrink is very similar to online table redefinition in that space in a table is reclaimed. However, segment shrink can be performed in place without the additional space requirements of online table redefinition.

To determine which segments will benefit from segment shrink, you can invoke *Segment Advisor* to perform growth trend analysis on specified segments. In this section, we'll invoke Segment Advisor on some candidate segments that may be vulnerable to fragmentation.

In the example that follows, we'll set up Segment Advisor to monitor the HR.EMPLOYEES table. In recent months, there has been high activity on this table; in addition, a new column, WORK_RECORD, has been added to the table, which HR uses to maintain comments about the employees:

SQL> **alter table hr.employees add (work_record varchar2(4000));**

Table altered.

SQL> alter table hr.employees enable row movement;

Table altered.

We have enabled ROW MOVEMENT in the table so that shrink operations can be performed on the table if recommended by Segment Advisor.

After Segment Advisor has been invoked to give recommendations, the findings from Segment Advisor are available in the DBA_ADVISOR_FINDINGS data dictionary view. To show the potential benefits of shrinking segments when Segment Advisor recommends a shrink operation, the view DBA_ADVISOR_RECOMMENDATIONS provides the recommended shrink operation along with the potential savings, in bytes, for the operation.

To set up Segment Advisor to analyze the HR.EMPLOYEES table, we will use an anonymous PL/SQL block, as follows:

```
-- begin Segment Advisor analysis for HR.EMPLOYEES
-- rev. 1.1 RJB 07/07/2007
--
-- SQL*Plus variable to retrieve the task number from Segment Advisor
variable task_id number
-- PL/SQL block follows
declare
name varchar2(100);
descr varchar2(500);
obj_id number;
begin
name := ''; -- unique name generated from create_task
descr := 'Check HR.EMPLOYEE table';
dbms_advisor.create_task
('Segment Advisor', :task_id, name, descr, NULL);
dbms_advisor.create_object
(name, 'TABLE', 'HR', 'EMPLOYEES', NULL, NULL, obj_id);
dbms_advisor.set_task_parameter(name, 'RECOMMEND_ALL', 'TRUE');
dbms_advisor.execute_task(name);
end;
PL/SQL procedure successfully completed.
SQL> print task_id
TASK_ID
----------
384
SQL>
```

The procedure DBMS_ADVISOR.CREATE_TASK specifies the type of advisor; in this case, it is Segment Advisor. The procedure will return a unique task ID and an automatically generated name to the calling program; we will assign our own description to the task.

Within the task, identified by the uniquely generated name returned from the previous procedure, we identify the object to be analyzed with DBMS_ADVISOR.CREATE_OBJECT. Depending on the type of object, the second through the sixth arguments vary. For tables, we only need to specify the schema name and the table name.

Using DBMS_ADVISOR.SET_TASK_PARAMETER, we tell Segment Advisor to give all possible recommendations about the table. If we want to turn off recommendations for this task, we would specify FALSE instead of TRUE for the last parameter.

Finally, we initiate the Segment Advisor task with the DBMS_ADVISOR.EXECUTE_TASK procedure. Once it is done, we display the identifier for the task so we can query the results in the appropriate data dictionary views.

Now that we have a task number from invoking Segment Advisor, we can query DBA_ ADVISOR_FINDINGS to see what we can do to improve the space utilization of the HR.EMPLOYEES table:

```
SQL> select owner, task_id, task_name, type,
2 message, more_info from dba_advisor_findings
3 where task_id = 384;
OWNER TASK_ID TASK_NAME TYPE
---------- ------- --------- ------
RJB 6 TASK_00003 INFORMATION
MESSAGE
--------------------------------------------------
Perform shrink, estimated savings is 107602 bytes.
MORE_INFO
--------------------------------------------------------------------
Allocated Space:262144: Used Space:153011: Reclaimable Space :107602:
```

The results are fairly self-explanatory. We can perform a segment shrink operation on the table to reclaim space from numerous insert, delete, and update operations on the HR.EMPLOYEES table. Because the WORK_RECORD column was added to the HR.EMPLOYEES table after the table was already populated, we may have created some chained rows in the table; in addition, since the WORK_RECORD column can be up to 4000 bytes long, updates or deletes of rows with big WORK_RECORD columns may create blocks in the table with free space that can be reclaimed.

The view DBA_ADVISOR_RECOMMENDATIONS provides similar information:

```
SQL> select owner, task_id, task_name, benefit_type
2 from dba_advisor_recommendations
3 where task_id = 384;
OWNER TASK_ID TASK_NAME
---------- ------- ----------
RJB 384 TASK_00003
BENEFIT_TYPE
--------------------------------------------------
Perform shrink, estimated savings is 107602 bytes.
```

In any case, we will shrink the segment HR.EMPLOYEES to reclaim the free space. As an added time-saving benefit to the DBA, the SQL needed to perform the shrink is provided in the view DBA_ADVISOR_ACTIONS:

```
SQL> select owner, task_id, task_name, command, attr1
2 from dba_advisor_actions where task_id = 384;
OWNER TASK_ID TASK_NAME COMMAND
---------- ------- ---------- -----------------
RJB 6 TASK_00003 SHRINK SPACE
ATTR1
--------------------------------------------------
alter table HR.EMPLOYEES shrink space
1 row selected.
SQL> alter table HR.EMPLOYEES shrink space;
Table altered.
```

As mentioned earlier, the shrink operation does not require extra disk space and does not prevent access to the table during the operation, except for a very short period of time at the end of the process to free the unused space. All indexes are maintained on the table during the operation. In addition to freeing up disk space for other segments, there are other benefits to shrinking a

segment. Cache utilization is improved because fewer blocks need to be in the cache to satisfy SELECT or other DML statements against the segment. Also, because the data in the segment is more compact, the performance of full table scans is improved.

There are a couple of caveats and minor restrictions. First, segment shrink will not work on LOB segments if you are using Oracle Database 10*g*. Online table reorganization is a more appropriate method in this case. Also, segment shrink is not allowed on a table that contains any function-based indexes regardless of whether you are using Oracle Database 10*g* or 11*g*.

Undo Advisor and the Automatic Workload Repository

New to Oracle 10*g*, the *Undo Advisor* provides tuning information for the undo tablespace, whether it's sized too large, it's too small, or the undo retention (via the initialization parameter UNDO_RETENTION) is not set optimally for the types of transactions that occur in the database. Using the Undo Advisor is similar to using the Segment Advisor in that we will call the DBMS_ADVISOR procedures and query the DBA_ADVISOR_* data dictionary views to see the results of the analysis.

The Undo Advisor, however, relies on another feature new to Oracle 10*g*—the *Automatic Workload Repository (AWR).* The Automatic Workload Repository, built into every Oracle database, contains snapshots of all key statistics and workloads in the database at 60-minute intervals by default. The statistics in the AWR are kept for seven days, after which the oldest statistics are dropped. Both the snapshot intervals and the retention period can be adjusted to suit your environment, however. The AWR maintains the historical record of how the database is being used over time and helps to diagnose and predict problems long before they can cause a database outage.

To set up Undo Advisor to analyze undo space usage, we will use an anonymous PL/SQL block similar to what we used for Segment Advisor. Before we can use Segment Advisor, however, we need to determine the timeframe to analyze. The data dictionary view DBA_HIST_SNAPSHOT contains the snapshot numbers and date stamps; we will look for the snapshot numbers from 8:00 P.M. Saturday, July 21, 2007 through 9:30 P.M. Saturday, July 21, 2007:

```
SQL> select snap_id, begin_interval_time, end_interval_time
2 from DBA_HIST_SNAPSHOT
3 where begin_interval_time > '21-Jul-07 08.00.00 PM' and
4 end_interval_time < '21-Jul-07 09.31.00 PM'
5 order by end_interval_time desc;
SNAP_ID BEGIN_INTERVAL_TIME END_INTERVAL_TIME
---------- --------------------------- ---------------------------
8 21-JAN-07 09.00.30.828 PM 21-JAN-07 09.30.14.078 PM
7 21-JAN-07 08.30.41.296 PM 21-JAN-07 09.00.30.828 PM
6 21-JAN-07 08.00.56.093 PM 21-JAN-07 08.30.41.296 PM
```

Given these results, we will use a SNAP_ID range from 6 to 8 when we invoke Undo Advisor. The PL/SQL anonymous block is as follows:

```
-- begin Undo Advisor analysis
-- rev. 1.1 RJB 7/16/2007
--
-- SQL*Plus variable to retrieve the task number from Segment Advisor
variable task_id number
declare
task_id number;
name varchar2(100);
descr varchar2(500);
obj_id number;
```

```
begin
name := ''; -- unique name generated from create_task
descr := 'Check Undo Tablespace';
dbms_advisor.create_task
('Undo Advisor', :task_id, name, descr);
dbms_advisor.create_object
(name, 'UNDO_TBS', NULL, NULL, NULL, 'null', obj_id);
dbms_advisor.set_task_parameter(name, 'TARGET_OBJECTS', obj_id);
dbms_advisor.set_task_parameter(name, 'START_SNAPSHOT', 6);
dbms_advisor.set_task_parameter(name, 'END_SNAPSHOT', 8);
dbms_advisor.set_task_parameter(name, 'INSTANCE', 1);
dbms_advisor.execute_task(name);
end;
PL/SQL procedure successfully completed.
SQL> print task_id
TASK_ID
-------
527
```

As with the Segment Advisor, we can review the DBA_ADVISOR_FINDINGS view to see the problem and the recommendations.

```
SQL> select owner, task_id, task_name, type,
2 message, more_info from dba_advisor_findings
3 where task_id = 527;
OWNER TASK_ID TASK_NAME TYPE
---------- ------- ---------- -------------
RJB 527 TASK_00003 PROBLEM
MESSAGE
-----------------------------------------------------
The undo tablespace is OK.
MORE_INFO
-----------------------------------------------------------------
```

In this particular scenario, Undo Advisor indicates that there is enough space allocated in the undo tablespace to handle the types and volumes of queries run against this database.

Index Usage

Although indexes provide a tremendous benefit by speeding up queries, they can have an impact on space usage in the database. If an index is not being used at all, the space occupied by an index can be better used elsewhere; if we don't need the index, we also can save processing time for insert, update, and delete operations that have an impact on the index. Index usage can be monitored with the dynamic performance view V$OBJECT_USAGE. In our HR schema, we suspect that the index on the JOB_ID column of the EMPLOYEES table is not being used. We turn on monitoring for this index as follows:

```
SQL> alter index hr.emp_job_ix monitoring usage;
Index altered.
```

We take a quick look at the V$OBJECT_USAGE view to make sure this index is being monitored:

```
SQL> select * from v$object_usage;
INDEX_NAME TABLE_NAME MON USED START_MONITORING
--------------- --------------- --- ---- -------------------
EMP_JOB_IX EMPLOYEES YES NO 07/24/2007 10:04:55
```

The column USED will tell us if this index is accessed to satisfy a query. After a full day of

typical user activity, we check V$OBJECT_USAGE again and then turn off monitoring:

```
SQL> alter index hr.emp_job_ix nomonitoring usage;
Index altered.
SQL> select * from v$object_usage;
INDEX_NAME TABLE_NAME MON USED START_MONITORING END_MONITORING
---------- --------------- --- ---- ------------------- ---------------
EMP_JOB_IX EMPLOYEES NO YES 07/24/2007 10:04:55 07/25/2007 11:39:45
```

Sure enough, the index appears to be used at least once during a typical day.

On the other end of the spectrum, an index may be accessed too frequently. If key values are inserted, updated, and deleted frequently, an index can become less efficient in terms of space usage. The following commands can be used as a baseline for an index after it is created, and then run periodically to see if the space usage becomes inefficient:

```
SQL> analyze index hr.emp_job_ix validate structure;
Index analyzed.
SQL> select pct_used from index_stats where name = 'EMP_JOB_IX';
PCT_USED
----------
78
```

The PCT_USED column indicates the percentage of the allocated space for the index in use. Over time, the EMPLOYEES table is heavily used, due to the high turnover rate of employees at the company, and this index, among others, is not using its space efficiently, as indicated by the following **analyze** command and **select** query, so we decide that a rebuild is in order:

```
SQL> analyze index hr.emp_job_ix validate structure;
Index analyzed.
SQL> select pct_used from index_stats where name = 'EMP_JOB_IX';
PCT_USED
----------
26
SQL> alter index hr.emp_job_ix rebuild online;
Index altered.
```

Notice the inclusion of the **online** option in the **alter index . . . rebuild** statement. The indexed table can remain online with minimal overhead while the index is rebuilding. In rare circumstances, such as on longer key lengths, you may not be able to use the **online** option.

Space Usage Warning Levels

Earlier in this chapter, we reviewed the data dictionary view DBA_THRESHOLDS, which contains a list of the active metrics to measure a database's health. In a default installation of Oracle 11*g*, use the following **select** statement to see some of the 22 built-in thresholds:

```
SQL> select metrics_name, warning_operator warn, warning_value wval,
2 critical_operator crit, critical_value cval,
3 consecutive_occurrences consec
4 from dba_thresholds;
```

```
METRICS_NAME                   WARN WVAL          CRIT CVAL          CONSEC
------------------------------ ---- ------------ ---- ------------ ------
Average Users Waiting Counts   GT   10           NONE               3
. . .
Blocked User Session Count     GT   0            NONE               15
Current Open Cursors Count     GT   1200         NONE               3
Database Time Spent Waiting (%) GT  30           NONE               3
. . .
Logons Per Sec                 GE   100          NONE               2
Session Limit %                GT   90           GT   97            3
Tablespace Bytes Space Usage   LE   0            LE   0             1
Tablespace Space Usage         GE   85           GE   97            1

22 rows selected.
```

In terms of space usage, we see that the warning level for a given tablespace is when the tablespace is 85 percent full, and the space is at a critical level when it reaches 97 percent full. In addition, this condition need only occur during one reporting period, which by default is one minute. For the other conditions in this list, the condition must be true anywhere between 2 and 15 consecutive reporting periods before an alert is issued.

To change the level at which an alert is generated, we can use the DBMS_SERVER_ALERT. SET_THRESHOLD procedure. In this example, we want to be notified sooner if a tablespace is running out of space, so we will update the warning threshold for alert notification from 85 percent down to 60 percent:

```
--
-- PL/SQL anonymous procedure to update the Tablespace Space Usage threshold
--
declare
/* OUT */
warning_operator number;
warning_value varchar2(100);
critical_operator number;
critical_value varchar2(100);
observation_period number;
consecutive_occurrences number;
/* IN */
metrics_id number;
instance_name varchar2(50);
object_type number;
object_name varchar2(50);
new_warning_value varchar2(100) := '60';
begin
metrics_id := DBMS_SERVER_ALERT.TABLESPACE_PCT_FULL;
object_type := DBMS_SERVER_ALERT.OBJECT_TYPE_TABLESPACE;
instance_name := 'dw';
object_name := NULL;
-- retrieve the current values with get_threshold
dbms_server_alert.get_threshold(
metrics_id, warning_operator, warning_value,
critical_operator, critical_value,
observation_period, consecutive_occurrences,
instance_name, object_type, object_name);
-- update the warning threshold value from 85 to 60
```

```
dbms_server_alert.set_threshold(
metrics_id, warning_operator, new_warning_value,
critical_operator, critical_value,
observation_period, consecutive_occurrences,
instance_name, object_type, object_name);
end;
```

PL/SQL procedure successfully completed.

Checking DBA_THRESHOLDS again, we see the warning level has been changed to 60 percent:

SQL> **select metrics_name, warning_operator warn, warning_value wval**
2 **from dba_thresholds;**

```
METRICS_NAME                   WARN WVAL
------------------------------ ---- -------------
Average Users Waiting Counts   GT   10
. . .
Blocked User Session Count     GT   0
Current Open Cursors Count     GT   1200
Database Time Spent Waiting (%) GT   30
. . .
Logons Per Sec                 GE   100
Session Limit %                GT   90
Tablespace Bytes Space Usage   LE   0
Tablespace Space Usage         GE   60

22 rows selected.
```

Managing Alert and Trace Files with ADR

New to Oracle Database 11*g*, the Automatic Diagnostic Repository (ADR) is a system-managed repository for storing database alert logs, trace files, and any other diagnostic data previously controlled by several other initialization parameters.

The initialization parameter DIAGNOSTIC_DEST sets the base location for all diagnostic directories; in the dw database I use throughout this chapter, the value of the parameter DIAGNOSTIC_DEST is /u01/app/oracle. Figure 6-4 shows a typical directory structure starting with the subdirectory /u01/app/oracle/diag.

Notice that there are separate directories for the ASM databases and the database (rdbms) instances; within the rdbms directory, you can see the dw directory twice: the first-level directory is the database dw, and the second dw is the instance dw. If this were a Real Application Clusters (RAC) database, you would see each instance of the dw database under the first-level dw directory. In fact, Oracle strongly recommends that all instances within a RAC database have the same value for DIAGNOSTIC_DEST.

Because the location of all logging and diagnostic information is controlled by the initialization parameter DIAGNOSTIC_DEST, the following initialization parameters are ignored:

BACKGROUND_DUMP_DEST
USER_DUMP_DEST
CORE_DUMP_DEST

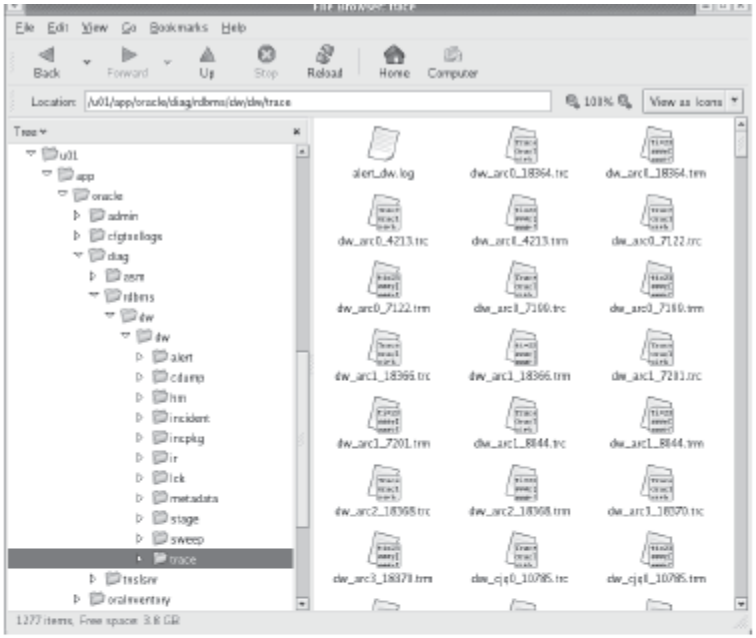


FIGURE 6-4 *ADR directory structure*

For backward compatibility, however, you can still use these as read-only parameters to determine the location of the alert log, trace files, and core dumps:

```
SQL> show parameter dump_dest

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
background_dump_dest                 string      /u01/app/oracle/diag/rdbms/dw/
                                                 dw/trace
core_dump_dest                       string      /u01/app/oracle/diag/rdbms/dw/
                                                 dw/cdump
user_dump_dest                       string      /u01/app/oracle/diag/rdbms/dw/
                                                 dw/trace
```

You can still alter the values for these parameters, but they are ignored by ADR. Alternatively, you can use the view V$DIAG_INFO to find all diagnostic-related directories for the instance:

```
SQL> select name, value from v$diag_info;
NAME VALUE
---------------------- ----------------------------------------
Diag Enabled TRUE
ADR Home /u01/app/oracle/diag/rdbms/dw/dw
Diag Trace /u01/app/oracle/diag/rdbms/dw/dw/trace
Diag Alert /u01/app/oracle/diag/rdbms/dw/dw/alert
Diag Incident /u01/app/oracle/diag/rdbms/dw/dw/incident
Diag Cdump /u01/app/oracle/diag/rdbms/dw/dw/cdump
Health Monitor /u01/app/oracle/diag/rdbms/dw/dw/hm
Default Trace File /u01/app/oracle/diag/rdbms/dw/dw/trace/dw_ora
_28810.trc
Active Problem Count 0
Active Incident Count 0
11 rows selected.
```

Transaction Basics

A *transaction* is a collection of SQL DML statements that is treated as a logical unit; the failure of

any of the statements in the transaction implies that none of the other changes made to the database in the transaction should be permanently saved to the database. Once the DML statements in the transaction have successfully completed, the application or SQL*Plus user will issue a **commit** to make the changes permanent. In the classic banking example, a transaction that transfers a dollar amount from one account to another is successful only if both the debit of one account (an **update** of the savings account balance) and the credit of another account (an **update** of the checking account balance) are both successful. Failure of either or both statements invalidates the entire transaction. When the application or SQL*Plus user issues a **commit**, if only one or the other **update** statement is successful, the bank will have some very unhappy customers!

A transaction is initiated implicitly. After a **commit** of a previous transaction is completed, and at least one row of a table is inserted, updated, or deleted, a new transaction is implicitly created. Also, any DDL commands such as **create table** and **alter index** will commit an active transaction and begin a new transaction. You can name a transaction by using the **set transaction . . . name '*transaction_name*'** command. Although this provides no direct benefit to the application, the name assigned to the transaction is available in the dynamic performance view V$TRANSACTION and allows a DBA to monitor long-running transactions; in addition, the transaction name helps the DBA resolve in-doubt transactions in distributed database environments. The **set transaction** command, if used, must be the first statement within the transaction.

Within a given transaction, you can define a *savepoint.* A savepoint allows the sequence of DML commands within a transaction to be partitioned so that it is possible to roll back one or more of the DML commands after the savepoint, and subsequently submit additional DML commands or commit the DML commands performed before the savepoint. Savepoints are created with the **savepoint *savepoint_name*** command. To undo the DML commands since the last savepoint, you use the command **rollback to savepoint *savepoint_name.***

A transaction is implicitly committed if a user disconnects from Oracle normally; if the user process terminates abnormally, the most recent transaction is rolled back.

Undo Basics

Undo tablespaces facilitate the rollback of logical transactions. In addition, undo tablespaces support a number of other features, including read consistency, various database-recovery operations, and Flashback functions.

Rollback

As described in the previous section, any DML command within a transaction—whether the transaction is one or one hundred DML commands—may need to be rolled back. When a DML command makes a change to a table, the old data values changed by the DML command are recorded in the undo tablespace within a system-managed undo segment or a rollback segment. When an entire transaction is rolled back (that is, a transaction without any savepoints), Oracle undoes all the changes made by DML commands since the beginning of the transaction using the corresponding undo records, releases the locks on the affected rows, if any, and the transaction ends.

If part of a transaction is rolled back to a savepoint, Oracle undoes all changes made by DML commands after the savepoint. All subsequent savepoints are lost, all locks obtained after the savepoint are released, and the transaction remains active.

Read Consistency

Undo provides *read consistency* for users who are reading rows that are involved in a DML transaction by another user. In other words, all users who are reading the affected rows will see no changes in the rows until they issue a new query after the DML user commits the transaction. Undo segments are used to reconstruct the datablocks back to a read-consistent version and, as a result, provide the previous values of the rows to any user issuing a **select** statement before the transaction commits.

For example, user CLOLSEN begins a transaction at 10:00 that is expected to commit at 10:15, with various updates and insertions to the EMPLOYEES table. As each **insert**, **update**, and **delete** occurs on the EMPLOYEES table, the old values of the table are saved in the undo tablespace. When the user SUSANP issues a **select** statement against the EMPLOYEES table at 10:08, none of the changes made by CLOLSEN are visible to anyone except CLOLSEN; the undo tablespace provides the previous values of CLOLSEN's changes for SUSANP and all other users. Even if the query from SUSANP does not finish until 10:20, the table still appears to be unchanged until a new query is issued after the changes are committed. Until CLOLSEN performs a **commit** at 10:15, the data in the table appears unchanged as of 10:00.

If there is not enough undo space available to hold the previous values of changed rows, the user issuing the **select** statement may receive an "ORA-01555: Snapshot Too Old" error. Later in this chapter, we will discuss ways in which we can address this issue.

Database Recovery

Undo tablespaces are also a key component of instance recovery. The online redo logs bring both committed and uncommitted transactions forward to the point in time of the instance crash; the undo data is used to roll back any transactions that were not committed at the time of the crash or instance failure.

Flashback Operations

The data in the undo tablespace is used to support the various types of Flashback options: Flashback Table, Flashback Query, and the package DBMS_FLASHBACK. Flashback Table will restore a table as of a point of time in the past, Flashback Query lets you view a table as of an SCN or time in the past, and DBMS_FLASHBACK provides a programmatic interface for Flashback operations. Flashback Data Archive, new to Oracle Database 11*g*, stores and tracks all transactions on a specified table for a specified time period; in a nutshell, Flashback Data Archive stores undo data for a specific table in a specific tablespace outside of the global undo tablespace. Also new to Oracle Database 11*g* is Flashback Transaction Backout that can roll back an already committed transaction and its dependent transactions while the database is online. All these Flashback options are covered in more detail at the end of this chapter.

Managing Undo Tablespaces

Creating and maintaining undo tablespaces is a "set it and forget it" operation once the undo requirements of the database are understood. Within the undo tablespace, Oracle automatically creates, sizes, and manages the undo segments, unlike previous versions of Oracle in which the DBA would have to manually size and constantly monitor rollback segments.

In the next couple sections, we'll review the processes used to create and manage undo tablespaces, including the relevant initialization parameters. In addition, we'll review some scenarios where we may create more than one undo tablespace and how to switch between undo tablespaces.

Creating Undo Tablespaces

Undo tablespaces can be created in two ways: at database creation or with the **create tablespace** command after the database is created. As with any other tablespace in Oracle 10*g*, the undo tablespace can be a bigfile tablespace, further easing the maintenance of undo tablespaces.

Creating an Undo Tablespace with CREATE DATABASE

A database may have more than one undo tablespace, although only one can be active at a time. Here's what creating an undo tablespace at database creation looks like:

```
create database ord
user sys identified by ds88dkw2
user system identified by md78s233
sysaux datafile '/u02/oradata/ord/sysaux001.dbf' size 1g
default temporary tablespace temp01
```

tempfile '/u03/oradata/ord/temp001.dbf' size 150m
undo tablespace undotbs01
datafile '/u01/oradata/ord/undo001.dbf' size 500m;

If the undo tablespace cannot be successfully created in the **create database** command, the entire operation fails. The error must be corrected, any files remaining from the operation must be deleted, and the command must be reissued.

Although the **undo tablespace** clause in the **create database** command is optional, if it is omitted and Automatic Undo Management is enabled, an undo tablespace is still created with an autoextensible datafile with an initial size of 10MB and the default name SYS_UNDOTBS.

Creating an Undo Tablespace with CREATE TABLESPACE

Any time after the database is created, a new undo tablespace can be created. An undo tablespace is created just as any other tablespace with the addition of the **undo** keyword:

create **undo** tablespace undotbs02
datafile '/u01/oracle/rbdb1/undo0201.dbf'
size 500m reuse autoextend on;

Depending on the volatility of the database or the expectation that the undo needs of the database may increase dramatically in the future, we start out this tablespace at only 500MB and allow it to grow.

Extents in an undo tablespace must be system managed; in other words, you can only specify **extent management** as **local autoallocate**.

Creating an Undo Tablespace Using EM Database Control

Creating an undo tablespace is straightforward using Enterprise Manager Database Control. From the Server tab on the home page, click the Tablespaces link. You will be presented with a list of existing tablespaces; click the Create button.

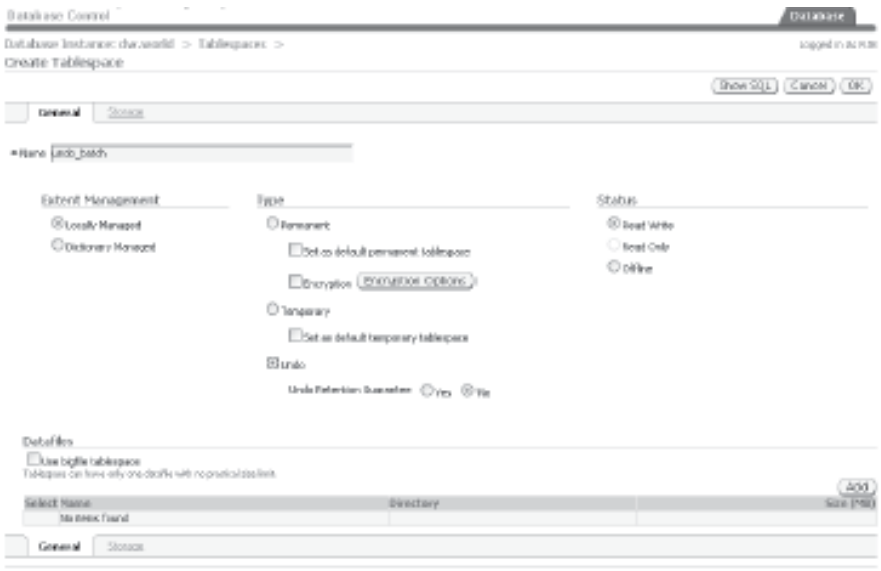


FIGURE 7-1 *Using EM Database Control to create an undo tablespace*

FIGURE 7-2 *Specifying a datafile for a new undo tablespace*

Modifying Undo Tablespaces

The following operations are allowed on undo tablespaces:

- Adding a datafile to an undo tablespace
- Renaming a datafile in an undo tablespace
- Changing an undo tablespace's datafile to online or offline
- Beginning or ending an open tablespace backup (**alter tablespace undotbs begin backup**)
- Enabling or disabling the undo retention guarantee

Using OMF for Undo Tablespaces

In addition to using a bigfile tablespace for undo tablespaces, you can also use OMF to automatically name (and locate, if you're not using ASM) an undo tablespace; the initialization parameter DB_CREATE_FILE_DEST contains the location where an undo tablespace will be created if the **datafile** clause is not specified in the **create undo tablespace** command. In the following example, we create an undo tablespace using OMF in an ASM disk group:

```
SQL> show parameter db_create_file_dest
NAME TYPE VALUE
------------------------------------ ----------- ------------------------------
db_create_file_dest string +DATA
SQL> create undo tablespace undo_bi;
Tablespace created.
SQL> select ts.name ts_name, df.name df_name, bytes
2 from v$tablespace ts join v$datafile df using(ts#)
3 where ts.name = 'UNDO_BI';
TS_NAME DF_NAME BYTES
------------ ------------------------------------------ ----------
UNDO_BI +DATA/dw/datafile/undo_bi.275.629807457 104857600
SQL>
```

Because we did not specify a datafile size either, the tablespace defaults to a size of 100MB; in addition, the datafile is autoextensible with an unlimited maximum size, limited only by the file system.

Undo Tablespace Dynamic Performance Views

A number of dynamic performance views and data dictionary views contain information about undo tablespaces, user transactions, and undo segments.

Undo Tablespace Initialization Parameters

In the following sections, we'll describe the initialization parameters needed to specify the undo tablespace for the database as well as control how long Oracle will retain undo information in the database.

UNDO_MANAGEMENT

The parameter UNDO_MANAGEMENT defaults to MANUAL in Oracle Database 10*g*, and AUTO in Oracle Database 11*g*. Setting the parameter UNDO_MANAGEMENT to AUTO places the database in Automatic Undo Management mode. At least one undo tablespace must exist in the database for this parameter to be valid, whether UNDO_TABLESPACE is specified or not. UNDO_MANAGEMENT is not a dynamic parameter; therefore, the instance must be restarted whenever UNDO_MANAGEMENT is changed from AUTO to MANUAL, or vice versa.

UNDO_TABLESPACE

The UNDO_TABLESPACE parameter specifies which undo tablespace will be used for Automatic Undo Management. If UNDO_MANAGEMENT is not specified or set to MANUAL, and UNDO_TABLESPACE is specified, the instance will not start.

View	Description
DBA_TABLESPACES	Tablespace names and characteristics, including the CONTENTS column, which can be PERMANENT, TEMPORARY, or UNDO; the undo RETENTION column is NOT APPLY, GUARANTEE, or NOGUARANTEE.
DBA_UNDO_EXTENTS	All undo segments in the database, including their size, their extents, the tablespace where they reside, and current status (EXPIRED or UNEXPIRED).
V$UNDOSTAT	The amount of undo usage for the database at ten-minute intervals; contains at most 1008 rows (7 days).
V$ROLLSTAT	Rollback segment statistics, including size and status.
V$TRANSACTION	Contains one row for each active transaction for the instance.

TABLE 7-1 *Undo Tablespace Views*

Conversely, if UNDO_MANAGEMENT is set to AUTO and there is no undo tablespace in the database, the instance will start, but then the SYSTEM rollback segment will be used for all undo operations, and a message is written to the alert log. Any user DML that attempts to make changes in non-SYSTEM tablespaces will, in addition, receive the error message "ORA-01552: cannot use system rollback segment for non-system tablespace 'USERS,'" and the statement fails.

UNDO_RETENTION

UNDO_RETENTION specifies a minimum amount of time that undo information is retained for queries. In automatic undo mode, UNDO_RETENTION defaults to 900 seconds. This value is valid only if there is enough space in the undo tablespace to support read-consistent queries; if active transactions require additional undo space, an unexpired undo may be used to satisfy the active transactions and may cause "ORA-01555: Snapshot Too Old" errors.

The column TUNED_UNDORETENTION of the dynamic performance view V$UNDOSTAT gives the tuned undo retention time for each time period; the status of the undo tablespace usage is updated in V$UNDOSTAT every ten minutes:

```
SQL> show parameter undo_retention
NAME TYPE VALUE
------------------------------------ ----------- ------------
undo_retention integer 900
SQL> select to_char(begin_time,'yyyy-mm-dd hh24:mi'),
2 undoblks, txncount, tuned_undoretention
3 from v$undostat where rownum = 1;
TO_CHAR(BEGIN_TI UNDOBLKS TXNCOUNT TUNED_UNDORETENTION
---------------- ---------- ---------- -------------------
2007-08-05 16:07 9 89 900
1 row selected.
SQL>
```

Because the transaction load is very light during the most recent time period, and the instance has just recently started up, the tuned undo retention value is the same as the minimum specified in the UNDO_RETENTION initialization parameter: 900 seconds (15 minutes).

Multiple Undo Tablespaces

As mentioned earlier in this chapter, a database can have multiple undo tablespaces, but only one of them can be active for a given instance at any one time. In this section, we'll show an example of switching to a different undo tablespace while the database is open.

In our **dw** database, we have three undo tablespaces:

```
SQL> select tablespace_name, status from dba_tablespaces
2 where contents = 'UNDO';
TABLESPACE_NAME STATUS
```

---------- ---------
UNDOTBS1 ONLINE
UNDO_BATCH ONLINE
UNDO_BI ONLINE
2 rows selected.
But only one of the undo tablespaces is active:
SQL> **show parameter undo_tablespace**
NAME TYPE VALUE
---------- ---------- ----------
undo_tablespace string UNDOTBS1
For overnight processing, we change the undo tablespace from UNDOTBS1 to the tablespace UNDO_BATCH, which is much larger to support higher DML activity. The disk containing the daytime undo tablespace is much faster but has a limited amount of space; the disk containing the overnight undo tablespace is much larger, but slower. As a result, we use the smaller undo tablespace to support OLTP during the day, and the larger undo tablespace for our data mart and data warehouse loads, as well as other aggregation activities, at night when response time is not as big of an issue.
About the time the undo tablespace is going to be switched, the user HR is performing some maintenance operations on the HR.EMPLOYEES table, and she has an active transaction in the current undo tablespace:
SQL> **connect hr/hr@dw;**
Connected.
SQL> **set transaction name 'Employee Maintenance';**
Transaction set.
SQL> **update employees set commission_pct = commission_pct * 1.1;**
107 rows updated.
SQL>
Checking V$TRANSACTION, you see HR's uncommitted transaction:
SQL> **select t.status, t.start_time, t.name**
2 **from v$transaction t join v$session s on t.ses_addr = s.saddr**
3 **where s.username = 'HR';**
STATUS START_TIME NAME
---------- ---------- ----------
ACTIVE 08/05/07 17:41:50 Employee Maintenance
1 row selected.
You change the undo tablespace as follows:
SQL> **alter system set undo_tablespace=undo_batch;**
System altered.
HR's transaction is still active, and therefore the old undo tablespace still contains the undo information for HR's transaction, leaving the undo segment still available with the following status until the transaction is committed or rolled back:
SQL> **select r.status**
2 **from v$rollstat r join v$transaction t on r.usn=t.xidusn**
3 **join v$session s on t.ses_addr = s.saddr**
4 **where s.username = 'HR';**
STATUS

PENDING OFFLINE
1 row selected.

Even though the current undo tablespace is UNDO_BATCH, the daytime tablespace UNDOTBS1 cannot be taken offline or dropped until HR's transaction is committed or rolled back:

```
SQL> show parameter undo_tablespace
NAME TYPE VALUE
------------------------ ----------- ---------------------
undo_tablespace string UNDO_BATCH
SQL> alter tablespace undotbs1 offline;
alter tablespace undotbs1 offline
*
ERROR at line 1:
ORA-30042: Cannot offline the undo tablespace
```

The error message ORA-30042 applies if you try to offline an undo tablespace that is in use— either it is the current undo tablespace or it still has pending transactions. Note that if we switch back to the daytime tablespace before HR commits or rolls back the original transaction, the status of HR's rollback segment reverts back to ONLINE:

```
SQL> alter system set undo_tablespace=undotbs1;
System altered.
SQL> select r.status
2 from v$rollstat r join v$transaction t on r.usn=t.xidusn
3 join v$session s on t.ses_addr = s.saddr
4 where s.username = 'HR';
STATUS
---------------
ONLINE
1 row selected.
```

Flashback Features

Flashback Query,
Flashback Table,
Flashback Version Query, and
Flashback Transaction Query.

Flashback Database uses Flashback logs in the Flash Recovery Area instead of undo in an undo tablespace to provide the Flashback functionality; Flashback Drop places dropped tables into a virtual recycle bin within the tablespace and they remain there until the user retrieves it with **flashback table . . . to before drop** command or empties the recycle bin, or else until the space is needed by new permanent objects in the tablespace.

To further extend the self-service capabilities of Oracle10*g* and Oracle 11*g*, the DBA can grant system and object privileges to users to allow them to fix their own problems, usually without any DBA intervention. In the following example, we're enabling the user SCOTT to perform Flashback operations on specific tables and to access transaction metadata across the database:

```
SQL> grant insert, update, delete, select on hr.employees to scott;
Grant succeeded.
SQL> grant insert, update, delete, select on hr.departments to scott;
Grant succeeded.
SQL> grant flashback on hr.employees to scott;
Grant succeeded.
SQL> grant flashback on hr.departments to scott;
Grant succeeded.
SQL> grant select any transaction to scott;
Grant succeeded.
```

Flashback Query

Starting with Oracle9*i* Release 2, the **as of** clause is available in a **select** query to retrieve the state of a table as of a given timestamp or SCN. You might use this to find out which rows in a table were deleted since midnight, or you might want to just do a comparison of the rows in a table today versus what was in the table yesterday.

In the following example, HR is cleaning up the EMPLOYEES table and deletes two employees who no longer work for the company:

```
SQL> delete from employees
2 where employee_id in (195,196);
2 rows deleted.
SQL> commit;
Commit complete.
SQL>
```

Normally, HR will copy these rows to the EMPLOYEES_ARCHIVE table first, but she forgot to do that this time; HR doesn't need to put those rows back into the EMPLOYEES table, but she needs to get the two deleted rows and put them into the archive table. Because HR knows she deleted the rows less than an hour ago, we can use a relative timestamp value with Flashback Query to retrieve the rows:

```
SQL> insert into hr.employees_archive
2 select * from hr.employees
3 as of timestamp systimestamp - interval '60' minute
4 where hr.employees.employee_id not in
5 (select employee_id from hr.employees);
2 rows created.
SQL> commit;
Commit complete.
```

Because we know that EMPLOYEE_ID is the primary key of the table, we can use it to retrieve the employee records that existed an hour ago but do not exist now. Note also that we didn't have to know which records were deleted; we essentially compared the table as it existed now versus an hour ago and inserted the records that no longer exist into the archive table.

Although we could use Flashback Table to get the entire table back, and then archive and delete the affected rows, in this case it is much simpler to merely retrieve the deleted rows and insert them directly into the archive table.

Another variation of Flashback Table is to use Create Table As Select (CTAS) with the subquery being a Flashback Query:

```
SQL> delete from employees where employee_id in (195,196);
2 rows deleted.
SQL> commit;
Commit complete.
SQL> create table employees_deleted as
2 select * from employees
3 as of timestamp systimestamp - interval '60' minute
4 where employees.employee_id not in
5 (select employee_id from employees);
Table created.
SQL> select employee_id, last_name from employees_deleted;
EMPLOYEE_ID LAST_NAME
----------- -------------------------
195 Jones
```

196 Walsh
2 rows selected.

This is known as an *out-of-place restore* (in other words, restoring the table or a subset of the table to a different location than the original). This has the advantage of being able to further manipulate the missing rows, if necessary, before placing them back in the table; for example, after reviewing the out-of-place restore, an existing referential integrity constraint may require that you insert a row into a parent table before the restored row can be placed back in the child table.

One of the disadvantages of an out-of-place restore using CTAS is that neither constraints nor indexes are rebuilt automatically.

Possible Questions
(Each question carries 6 Marks)

1. Explain extents and blocks with suitable example.
2. Discuss reusable space allocation with suitable example
3. Discuss segment advisor with suitable example.
4. Explain the automatic work load repository with suitable example.
5. Write notes on i)Multiplexing Redo log files ii)Controlling Archive Log.
6. Explain how to manage tablespace alert with suitable example.
7. Explain the components of Data Blocks with suitable example.
8. Discuss the composite indexes.
9. Discuss Managing Undo Tablespaces.
10. Describe Flashback features.

KARPAGAM ACADEMY OF HIGHER EDUCATION

COIMBATORE - 21

DEPARTMENT: CS,CA & IT

CLASS : II MCA

BATCH : 2018-2020

Part -A Online Examinations **(1 mark questions)**

SUBJECT: Database Administration **SUBJECT CODE: 17CAP404D**

UNIT II

S.no	Questions	opt1	opt 2	opt 3	opt 4	Answer
1	_____tablespaces are used to store objects for the duration of a user's session only	undo	temporary	permanent	locally	temporary
2	_____ tablespaces are a type of permanent tablespace that are used to store undo data.	undo	temporary	permanent	locally	undo
3	_____ tablespace have to constantly check the data dictionary during the course of extend management	temporary	permanent	locally managed	dictionary managed	dictionary managed
4	Oracle keepd track of how much free space is in its data blocks by maintaining _____	buffers	freelists	freespace	freeorder	freelists
5	Every table and index maintains a list of all its data blocks with freespace greater than	PCTFREE	PCTUSED	PCTNULL	PCTSPACE	PCTUSED
6	The _____ parameter lets you reserve a percentage of space in each data block for future updates to existing data	PCTFREE	PCTUSED	PCTNULL	PCTSPACE	PCTFREE
7	_____ is the default for extend management	ALLOCATE	EXTEND	AUTOALLOCATE	RESIZE	AUTOALLOCATE
8	To remove the tablepace from the database use the command_____	REMOVE	DROP	DELETE	ALTER	DROP
9	You can use the new _____ data dictionary view to manage the temporary tablespace groups in your database	DBA_TABLESPA	DBA_TABLESPAC	DBA_TABLES	DBA_GROUPS	DBA_TABLESPACE_GROUPS
10	_____ provides speedy access to table rows by stroing sorted values of specified columns	oracle processes	oracle files	oracle indexes	oracle tables	oracle indexes
11	_____ indexes are the unique indexes in a table that must always posses a value	secondary	nonunique	primary	composite	primary
12	_____ indexes are indexes that contain two or more columns from the same table.	secondary	nonunique	primary	composite	composite
13	_____ indexes are also known as concatenated indexes	secondary	nonunique	primary	composite	composite
14	_____ indexes are structured in the form of an inverse tree.	primary	composite	B-tree	h-tree	B-tree
15	_____ is a virtual table.	view	index	functions	tables	view
16	_____ are static objects that derive their data from the underlying base tables.	view	materialized views	index	functions	materialized views
17	Oracle will automatically rewrite the queries using the techniques called _____	query writable	query rearrange	query rewrite	query change	query rewrite
18	Materialized view is automatically refreshed when _____ mode is used	ON DEMAND	ON END	ON COMMIT	ON EXIT	ON COMMIT
19	Under the _____ mechanism, oracle will use a materialized view log to log all changes to the master table.	COMPLETE	FAST	FORCE	NEVER	FAST
20	A _____ is a logical unit of work soncisting of one or more SQL statements	order	transaction	recovery	backup	transaction
21	The majority of _____ statements retrive data from the database.	DML	DDL	Functional	indexed	DML
22	If a transaction encounters a _____ statement, all the changes to that point are made permanent in the database.	ROLLBACK	COMMIT	RECOVERY	BACKUP	COMMIT
23	If a transaction encouters a _____ statement, all changes made up to that point are cancelled	ROLLBACK	COMMIT	RECOVERY	BACKUP	ROLLBACK
24	With the _____ option, in the commit statement, the log wirter writes the redo log records for the committing transaction immediately to disk.	IMMEDIATE	BATCH	WAIT	NOWAIT	IMMEDIATE
25	You can partially rollback the effects of a transaction by using _____	SAVEPOINT	BACK	WAIT	NOWAIT	SAVEPOINT

26	A _____ occurs when a transaction reads data that has been updated by an ongoing transaction but has not been committed permanently to the database	dirty read	phantom read	lost update	fuzzy read	dirty read
27	The _____ level, allows a transaction to read another transaction's intermediate values before it commits	serializable	repeatable read	read uncommited	read committed	read uncommited
28	Oracle uses _____ granularity to lock objects	row level	table level	column level	serial level	row level
29	An _____ only permits queries on a table, and prevents users from perfroming any other activity on it.	exclusive lock	inclusive lock	import lock	export lock	exclusive lock
30	_____ are internal mechanisms that protect shared data structures in the SGA.	internal lock	distributed lock	latches	DDL locks	latches
31	_____ locks are used by oracle whenever the dictionary objects are being modified.	internal lock	distributed lock	latches	data dictionary	data dictionary
32	_____ are used by oracle to protect access to structure such as data files, tablespaces and rollback segments	internal lock	distributed lock	latches	data dictionary	internal lock
33	_____ locks are used in oracle real application clusters	internal lock	distributed lock	latches	data dictionary	distributed lock
34	Oracles uses _____ to provide automatic statement level read consistency	undo records	undo read	undo locks	undo buffer	undo records
35	The _____ parameter lets you control the reuse of the committed undospace.	expired	unexpired	undo_retention	undo_time	undo_retention
36	The default retention time is _____ seconds	1000	1200	900	800	900
37	_____ retrives data from a past point in time	flashback query	flashback data	flashback table	flashback statements	flashback query
38	The independent child transaction is called as _____ transaction	parent	autonomous	discrete	non autonomous	autonomous
39	_____keyword indicates the percentage of space reserved in the index blocks	PCTFREE	PCTTHRESHOLD	THRESHOLD	PCTFREE THRESHOLD	PCTTHRESHOLD
40	In Indexed Organised structure the data is stored in _____index structure.	B-tree Index	Hybrid Structure	Logical Structure	Network structure	B-tree Index Structure
41	_____is used to store two or more tables together	Cluster	Group	Joins	Composite	Cluster
42	_____provides speedy access to table rows by storing sorted values of specified columns.	Joins	Sorting	Indexing	Primary key	Indexing
43	_____indexes are based on unique column.	Unique Indexes	Secondary Indexes	Composite Indexes	NonUnique Indexes	Unique Indexes
44	_____package is used to estimate the size of a new index.	DBMS_SPACE	INDEX_COST1	DBMS_SIZE	INDEX_COST1	DBMS_SPACE
45	Views can be deleted by using _____command	Delete	Remove	Replace	Drop	Drop
46	_____view takes up physical space in our database.	Materialized view	Updatable view	Cursor view	Physical View	Materialized view
47	_____problems are caused by the appearance of new data in between two database operations in a transaction.	Lost-Update problem	Dirt-Read Problem	Phantom-Read Problem	Nonrepeatable-Read Problem	Phantom-Read Problem
48	_____isolation level will lock all the tables	Repeatable read	Serializable	Read uncommitted	Read Commited	Serializable
49	_____isolation level guarantees that the row data won't change while we are accessing a particular row in an Oracle table.	Repeatable read	Serializable	Read uncommitted	Read Commited	Read Commited
50	Serializable transactions are more prone to	deadlocks	locks	throughput	Phanthom	deadlocks
51	____ specifies the name and the location of the bad file.	log	bad	undo	dirty	bad
52	_____ is used for the current date.	date	sysdate	today	timestamp	sysdate
53	Direct path loading method doesn't use the _____ statement to put data into the tables.	SQL insert	delete	drop	update	SQL insert
54	_____parameter is used to show a value of LOCAL or DICTIONARY managed tablespace.	Initial Extent	Next Extent	Extent Management	Segment Space Management	Extent Management
55	_____parameter is used to set the limits of the tablespaces	LIMIT	MAXSIZE	UNLIMITED	AUTOEXTEND	MAXSIZE
56	_____tablespace contains only one very large data file.	Bigfile tablespace	Large file tablespace	Voulme tablespace	Default tablespace	Bigfile tablespace

57	Which clause is used to drop the constraints permanently?	DELETE CONSTRAINT	REMOVE CONSTRAINT	CASCADE CONSTRAINT	PURGE CONSTRAINT	CASCADE CONSTRAINT
58	_____keyword indicates the percentage of space reserved in the index blocks	PCTFREE	PCTTHRESHOLD	THRESHOLD	PCTFREE THRESHOLD	PCTTHRESHOLD

UNIT-II

SYLLABUS

Common Space Management Problems – Oracle Segments, Extents and Blocks – Space Management Methodologies – SYSAUX monitoring and usage – Archived Redo Log File Management – Built in Space Management Tools: Segment Advisor – Undo Advisor and the Automatic Workload Repository – Index usage – Space Usage Warning Levels – Reusable space allocation – Managing alert and Trace Files with ADR – Transaction Basics – Undo Basics – Managing Undo Tablespaces – Flashback features

Common Space Management Problems

Space management problems generally fall into one of three categories: running out of space in a regular tablespace, not having enough undo space for long-running queries that need a consistent "before" image of the tables, and insufficient space for temporary segments. Although we may still have some fragmentation issues within a database object such as a table or index, locally managed tablespaces solve the problem of tablespace fragmentation.

Running Out of Free Space in a Tablespace

If a tablespace is not defined with the AUTOEXTEND attribute, then the total amount of space in all the datafiles that compose the tablespace limits the amount of data that can be stored in the tablespace. If the AUTOEXTEND attribute is defined, then one or more of the datafiles that compose the tablespace will grow to accommodate the requests for new segments or the growth of existing segments. Even with the AUTOEXTEND attribute, the amount of space in the tablespace is ultimately limited by the amount of disk space on the physical disk drive or storage group.

The AUTOEXTEND attribute is the default if you don't specify the SIZE parameter in the **create tablespace** command and you are using OMF, so you'll actually have to go out of your way to prevent a datafile from autoextending. In Oracle Database 11*g* with the initialization parameter DB_CREATE_FILE_DEST set to an ASM or file system location, you can run a **create tablespace** command like this:

create tablespace bi_02;

In this case, the tablespace BI_02 is created with a size of 100MB in a single datafile, AUTOEXTEND is on, and the next extent is 100MB when the first datafile fills up. In addition,extent management is set to LOCAL, space allocation is AUTOALLOCATE, and segment space management set to AUTO.

The conclusion to be reached here is that we want to monitor the free and used space within a tablespace to detect trends in space usage over time, and as a result be proactive in making sure that enough space is available for future space requests. As of Oracle Database 10*g*, you can use the DBMS_SERVER_ALERT package to automatically notify you when a tablespace reaches a warning or critical space threshold level, either at a percent used, space remaining, or both.

Insufficient Space for Temporary Segments

A *temporary segment* stores intermediate results for database operations such as sorts, index builds, **distinct** queries, **union** queries, or any other operation that necessitates a sort/merge operation that cannot be performed in memory. Under no circumstances should the SYSTEM tablespace be used for temporary segments; when the database is created, a non-SYSTEM tablespace should be specified as a default

temporary tablespace for users who are not otherwise assigned a temporary tablespace. If the SYSTEM tablespace is locally managed, a default temporary tablespace must be defined when the database is created.

When there is not enough space available in the user's default temporary tablespace, and either the tablespace cannot be autoextended or the tablespace's AUTOEXTEND attribute is disabled, the user's query or DML statement fails.

Too Much or Too Little Undo Space Allocated

As of Oracle9*i*, undo tablespaces have simplified the management of rollback information by managing undo information automatically within the tablespace. The DBA no longer has to define the number and size of the rollback segments for the kinds of activity occurring in the database.

As of Oracle 10*g*, manual rollback management has been deprecated. Not only does an undo segment allow a rollback of an uncommitted transaction, it provides for read consistency of long-running queries that begin before inserts, updates, and deletes occur on a table. The amount of undo space available for providing read consistency is under the control of the DBA and is specified as the number of seconds that Oracle will attempt to guarantee that "before" image data is available for long-running queries.

As with temporary tablespaces, we want to make sure we have enough space allocated in an undo tablespace for peak demands without allocating more than is needed. As with any tablespace, we can use the AUTOEXTEND option when creating the tablespace to allow for unexpected growth of the tablespace without reserving too much disk space up front.

Fragmented Tablespaces and Segments

As of Oracle8*i*, a tablespace that is locally managed uses bitmaps to keep track of free space, which,in addition to eliminating the contention on the data dictionary, eliminates wasted space because all extents are either the same size (with uniform extent allocation) or are multiples of the smallest size (with autoallocation). For migrating from a dictionary-managed tablespace, we will review an example that converts a dictionary-managed tablespace to a locally managed tablespace. In a default installation of Oracle Database 10*g* or Oracle Database 11*g* using the Database Creation Assistant (DBCA), all tablespaces, including the SYSTEM and SYSAUX tablespaces, are created as locally managed tablespaces.

Even though locally managed tablespaces with automatic extent management (using the **autoallocate** clause) are created by default when you use **create tablespace**, you still need to specify **extent management local** if you need to specify **uniform** for the extent management type in the **create tablespace** statement:

```
SQL> create tablespace USERS4
2 datafile '+DATA'
3 size 250M autoextend on next 250M maxsize 2000M
4 extent management local uniform size 8M
5 segment space management auto;
Tablespace created.
```

This tablespace will be created with an initial size of 250MB, and it can grow as large as 2000MB (2GB); extents will be locally managed with a bitmap, and every extent in this tablespace will be exactly 8MB in size. Space within each segment (table or index) will be managed automatically with a bitmap instead of freelists.

Even with efficient extent allocation, table and index segments may eventually contain a lot of free space due to **update** and **delete** statements. As a result, a lot of unused space can be reclaimed by using some of the scripts I provide later in this chapter, as well as by using the Oracle Segment Advisor.

Oracle Segments, Extents, and Blocks

At the lowest level, a tablespace segment consists of one or more extents, each extent comprising one or more data blocks.

Data Blocks

A *data block* is the smallest unit of storage in the database. Ideally, an Oracle block is a multiple of the operating system block to ensure efficient I/O operations. The default block size for the database is specified with the DB_BLOCK_SIZE initialization parameter; this block size is used for the SYSTEM, TEMP, and SYSAUX tablespaces at database creation and cannot be changed without re-creating the database.

Every data block contains a *header* that specifies what kind of data is in the block—table rows or index entries. The *table directory* section has information about the table with rows in the block;a block can have rows from only one table or entries from only one index, unless the table is a clustered table, in which case the table directory identifies all the tables with rows in this block.

The *row directory* provides details of the specific rows of the table or index entries in the block.

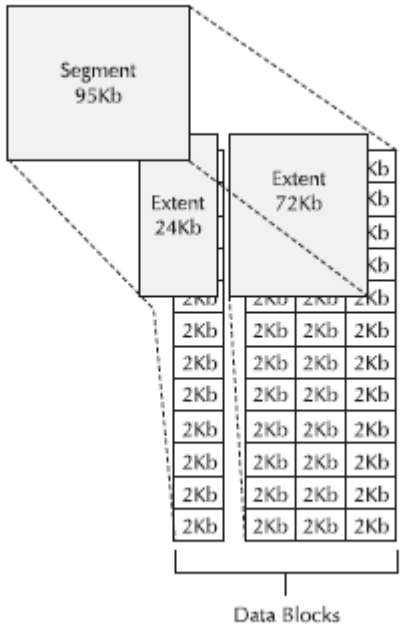


FIGURE 6-1 *Oracle segments, extents, and blocks*

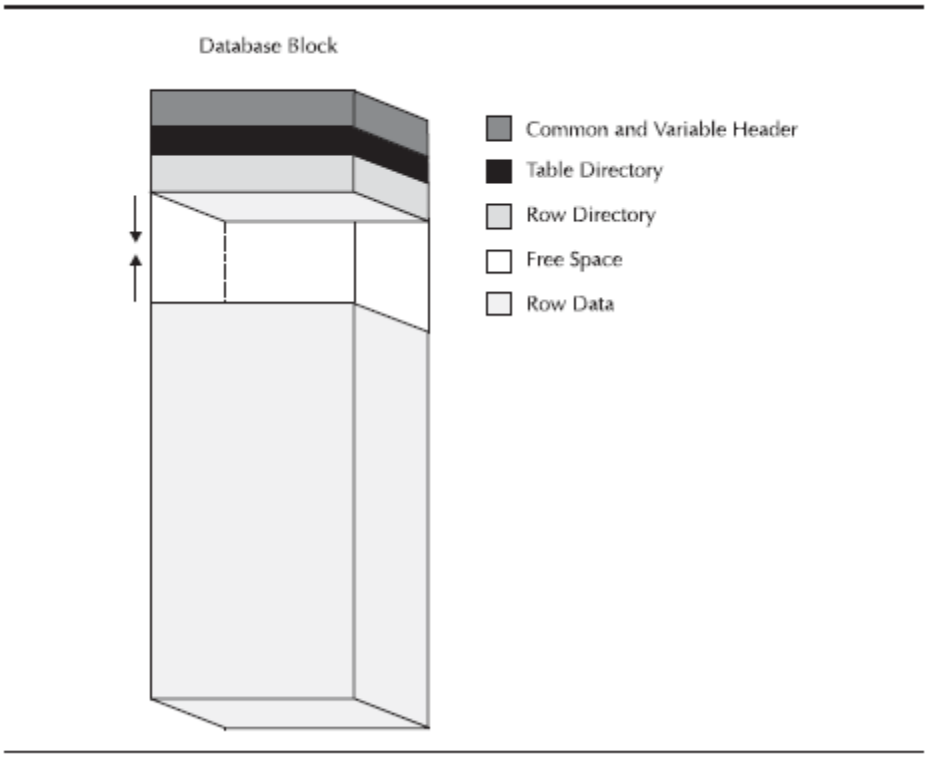


FIGURE 6-2 *Contents of an Oracle data block*

The space for the header, table directory, and row directory is a very small percentage of the space allocated for a block; our focus, then, is on the *free space* and *row data* within the block.

Within a newly allocated block, free space is available for new rows and updates to existing rows; the updates may increase or decrease the space allocated for the row if there are varyinglength columns in the row or a non-NULL value is changed to a NULL value, or vice versa. Space is available within a block for new inserts until there is less than a certain percentage of space available in the block defined by the PCTFREE parameter, specified when the segment is created.

Once there is less than PCTFREE space in the block, no inserts are allowed. If freelists are used to manage space within the blocks of a segment, then new inserts are allowed on the table when used space within the block falls below PCTUSED.

A row may span more than one block if the row size is greater than the block size or an updated row no longer fits into the original block. In the first case, a row that is too big for a block is stored in a *chain* of blocks; this may be unavoidable if a row contains columns that exceed even the largest block size allowed, which in Oracle 11*g* is 32KB.

Oracle will *migrate* the data for the entire row to a new block and leave a pointer in the first block to point to the location in the second block where the updated row is stored. As you may infer, a segment with many migrated rows may cause I/O performance problems because the number of blocks required to satisfy a query can double. In some cases, adjusting the value of PCTFREE or rebuilding the table may result in better space utilization and I/O performance.

Starting with Oracle9*i* Release 2, you can use Automatic Segment Space Management (ASSM) to manage free space within blocks; you enable ASSM in locally managed tablespaces by using the **segment space management auto** keywords in the **create tablespace** command (although this is the default for locally managed tablespaces).

Using ASSM reduces segment header contention and improves simultaneous insert concurrency;this is because the free space map in a segment is spread out into a bitmap block within each extent of the

segment. As a result, you dramatically reduce waits because each process performing **insert**, **update**, or **delete** operations will likely be accessing different blocks instead of one freelist or one of a few freelist groups. In addition, each extent's bitmap block lists each block within the extent along with a four-bit "fullness" indicator defined as follows (with room for future expansion from values 6–15):

0000 Unformatted block
0001 Block full
0010 Less than 25 percent free space available
0011 25 percent to 50 percent free space
0100 50 percent to 75 percent free space
0101 Greater than 75 percent free space

In a RAC database environment, using ASSM segments means you no longer need to create multiple freelist groups. In addition, you no longer need to specify PCTUSED, FREELISTS, or FREELIST GROUPS parameters when you create a table; if you specify any of these parameters, they are ignored.

Extents

An *extent* is the next level of logical space allocation in a database; it is a specific number of blocks allocated for a specific type of object, such as a table or index. An extent is the minimum number of blocks allocated at one time; when the space in an extent is full, another extent is allocated.

When a table is created, an *initial* extent is allocated. Once the space is used in the initial extent, *incremental* extents are allocated. In a locally managed tablespace, these subsequent extents can either be the same size (using the UNIFORM keyword when the tablespace is created) or optimally sized by Oracle (AUTOALLOCATE). For extents that are optimally sized, Oracle starts with a minimum extent size of 64KB and increases the size of subsequent extents as multiples of the initial extent as the segment grows. In this way, fragmentation of the tablespace is virtually eliminated.

When the extents are sized automatically by Oracle, the storage parameters INITIAL, NEXT, PCTINCREASE, and MINEXTENTS are used as a guideline, along with Oracle's internal algorithm, to determine the best extent sizes. In the following example, a table created in the USERS tablespace (during installation of a new database, the USERS tablespace is created with AUTOALLOCATE enabled) does not use the storage parameters specified in the **create table** statement:

```
SQL> create table t_autoalloc (c1 char(2000))
2 storage (initial 1m next 2m pctincrease 50)
3 tablespace users;
Table created.
SQL> begin
2 for i in 1..3000 loop
3 insert into t_autoalloc values ('a');
4 end loop;
5 end;
6 /
PL/SQL procedure successfully completed.
SQL> select segment_name, extent_id, bytes, blocks
2 from user_extents where segment_name = 'T_AUTOALLOC';
SEGMENT_NAME EXTENT_ID BYTES BLOCKS
------------ ---------- ---------- ----------
T_AUTOALLOC 0 65536 8
T_AUTOALLOC 1 65536 8
. . .
T_AUTOALLOC 15 65536 8
```

T_AUTOALLOC 16 1048576 128
. . .
T_AUTOALLOC 22 1048576 128
23 rows selected.

Unless a table is truncated or the table is dropped, any blocks allocated to an extent remain allocated for the table, even if all rows have been deleted from the table. The maximum number of blocks ever allocated for a table is known as the *high-water mark (HWM).*

Segments

Groups of extents are allocated for a single *segment.* A segment must be wholly contained within one and only one tablespace. Every segment represents one and only one type of database object, such as a table, a partition of a partitioned table, an index, or a temporary segment. For partitioned tables, every partition resides in its own segment; however, a cluster (with two or more tables) resides within a single segment. Similarly, a partitioned index consists of one segment for each index partition.

Temporary segments are allocated in a number of scenarios. When a sort operation cannot fit in memory, such as a **select** statement that needs to sort the data to perform a **distinct**, **group by**, or **union** operation, a temporary segment is allocated to hold the intermediate results of the sort. Index creation also typically requires the creation of a temporary segment. Because allocation and deallocation of temporary segments occurs often, it is highly desirable to create a tablespace specifically to hold temporary segments. This helps to distribute the I/O required for a given operation, and it reduces the possibility that fragmentation may occur in other tablespaces due to the allocation and deallocation of temporary segments. When the database is created, a *default temporary tablespace* can be created for any new users who do not have a specific temporary tablespace assigned; if the SYSTEM tablespace is locally managed, a separate temporary tablespace must be created to hold temporary segments.

How space is managed within a segment depends on how the tablespace containing the block is created. If the tablespace is dictionary managed, the segment uses freelists to manage space within the data blocks; if the tablespace is locally managed, space in segments can be managed with either freelists or bitmaps. Oracle strongly recommends that all new tablespaces be created as locally managed and that free space within segments be managed automatically with bitmaps. Automatic segment space management allows more concurrent access to the bitmap lists in a segment compared to freelists; in addition, tables that have widely varying row sizes make more efficient use of space in segments that are automatically managed.

As I mentioned earlier, in the section titled "Data Blocks," if a segment is created with automatic segment space management, bitmaps are used to manage the space within the segment. As a result, the **pctused**, **freelist**, and **freelist groups** keywords within a **create table** or **create index** statement are ignored. The three-level bitmap structure within the segment indicates whether blocks below the HWM are full (less than **pctfree**), 0 to 25 percent free, 25 to 50 percent free, 50 to 75 percent free, 75 to 100 percent free, or unformatted.

Space Management Methodologies

In the following sections, we will consider various features of Oracle 11*g* to facilitate the efficient use of disk space in the database. Locally managed tablespaces offer a variety of advantages to the DBA, improving the performance of the objects within the tablespace, as well as easing administration of the tablespace—fragmentation of a tablespace is a thing of the past. Another feature introduced in Oracle9*i*, Oracle Managed Files, eases datafile maintenance by automatically removing files at the operating system level when a tablespace or other database object is dropped. Bigfile tablespaces, introduced in Oracle 10*g*, simplify datafile management because one and only one datafile is associated with a bigfile tablespace. This moves the maintenance point up one level, from the datafile to the tablespace. We'll also review a couple other features introduced in

Oracle9*i*—undo tablespaces and multiple block sizes.

Locally Managed Tablespaces

Prior to Oracle8*i*, there was only one way to manage free space within a tablespace—by using data dictionary tables in the SYSTEM tablespace. If a lot of insert, delete, and update activity occurs anywhere in the database, there is the potential for a "hot spot" to occur in the SYSTEM tablespace where the space management occurs. Oracle removed this potential bottleneck by introducing *locally managed tablespaces (LMTs).* A locally managed tablespace tracks free space in the tablespace with bitmaps, as discussed in Chapter 1. These bitmaps can be managed very efficiently because they are very compact compared to a freelist of available blocks. Because they are stored within the tablespace itself, instead of in the data dictionary tables, contention in the SYSTEM tablespace is reduced.

As of Oracle 10*g*, by default, all tablespaces are created as locally managed tablespaces, including the SYSTEM and SYSAUX tablespaces. When the SYSTEM tablespace is locally managed, you can no longer create any dictionary-managed tablespaces in the database that are read/write. A dictionary-managed tablespace may still be plugged into the database from an earlier version of Oracle, but it is read-only.

An LMT can have objects with one of two types of extents: automatically sized or all of a uniform size. If extent allocation is set to UNIFORM when the LMT is created, all extents, as expected, are the same size. Because all extents are the same size, there can be no fragmentation. Gone is the classic example of a 51MB segment that can't be allocated in a tablespace with two free 50MB extents because the two 50MB extents are not adjacent.

On the other hand, automatic segment extent management within a locally managed tablespace allocates space based on the size of the object. Initial extents are small, and if the object stays small, very little space is wasted. If the table grows past the initial extent allocated for the segment, subsequent extents to the segment are larger. Extents in an autoallocated LMT have sizes of 64KB, 1MB, 8MB, and 64MB, and the extent size increases as the size of the segment increases, up to a maximum of 64MB. In other words, Oracle is specifying what the values of INITIAL, NEXT, and PCTINCREASE are automatically, depending on how the object grows. Although it seems like fragmentation can occur in a tablespace with autoallocation, in practice the fragmentation is minimal because a new object with a 64KB initial segment size will fit nicely in a 1MB, 4MB, 8MB, or 64MB block preallocated for all other objects with an initial 64KB extent size.

Given an LMT with either automatically managed extents or uniform extents, the free space within the segment itself can be AUTO or MANUAL. With AUTO segment space management, a bitmap is used to indicate how much space is used in each block. The parameters PCTUSED, FREELISTS, and FREELIST GROUPS no longer need to be specified when the segment is created. In addition, the performance of concurrent DML operations is improved because the segment's bitmap allows concurrent access. In a freelist-managed segment, the data block in the segment header that contains the freelist is locked out to all other writers of the block when a single writer is looking for a free block in the segment. Although allocating multiple freelists for very active segments does somewhat solve the problem, it is another structure that the DBA has to manage.

Another advantage of LMTs is that rollback information is reduced or eliminated when any LMT space-related operation is performed. Because the update of a bitmap in a tablespace is not recorded in a data dictionary table, no rollback information is generated for this transaction.

Other than third-party applications, such as older versions of SAP that require dictionarymanaged tablespaces, there are no other reasons for creating new dictionary-managed tablespaces in Oracle 10*g.* As mentioned earlier, compatibility is provided in part to allow dictionary-managed tablespaces from previous versions of Oracle to be "plugged into" an Oracle 11*g* database, although if the SYSTEM tablespace is locally managed, any dictionary-managed tablespaces must be opened read-only. Later in this chapter, you'll see some examples where we can optimize space and

performance by moving a tablespace from one database to another and allocating additional data buffers for tablespaces with different sizes.

Migrating a dictionary-managed tablespace to a locally managed tablespace is very straightforward using the DBMS_SPACE_ADMIN built-in package:

execute sys.dbms_space_admin.tablespace_migrate_to_local('USERS')

After upgrading a database to either Oracle9*i*, Oracle 10*g*, or Oracle 11*g*, you may also want to consider migrating the SYSTEM tablespace to an LMT; if so, a number of prerequisites are in order:

Before starting the migration, shut down the database and perform a cold backup of the database.

Any non-SYSTEM tablespaces that are to remain read/write should be converted to LMTs.

The default temporary tablespace must not be SYSTEM.

If automatic undo management is being used, the undo tablespace must be online.

For the duration of the conversion, all tablespaces except for the undo tablespace must be set to read-only.

The database must be started in RESTRICTED mode for the duration of the conversion.

If any of these conditions are not met, the TABLESPACE_MIGRATE_TO_LOCAL procedure will not perform the migration.

Using OMF to Manage Space

In a nutshell, *Oracle-Managed Files (OMF)* simplifies the administration of an Oracle database. At database-creation time, or later by changing a couple parameters in the initialization parameter file, the DBA can specify a number of default locations for database objects such as datafiles, redo log files, and control files. Prior to Oracle9*i*, the DBA had to remember where the existing datafiles were stored by querying the DBA_DATA_FILES and DBA_TEMP_FILES views. On many occasions, a DBA would drop a tablespace, but would forget to delete the underlying datafiles, thus wasting space and the time it took to back up files that were no longer used by the database.

Using OMF, Oracle not only automatically creates and deletes the files in the specified directory location, it ensures that each filename is unique. This avoids corruption and database downtime in a non-OMF environment due to existing files being overwritten by a DBA inadvertently creating a new datafile with the same name as an existing datafile, and using the REUSE clause.

In a test or development environment, OMF reduces the amount of time the DBA must spend on file management and lets him or her focus on the applications and other aspects of the testdatabase.

OMF has an added benefit for packaged Oracle applications that need to create tablespaces: The scripts that create the new tablespaces do not need any modification to include a datafile name, thus increasing the likelihood of a successful application deployment.

Migrating to OMF from a non-OMF environment is easy, and it can be accomplished over a longer time period. Non-OMF files and OMF files can coexist indefinitely in the same database. When the appropriate initialization parameters are set, all new datafiles, control files, and redo log files can be created as OMF files, while the previously existing files can continue to be managed manually until they are converted to OMF, if ever.

The OMF-related initialization parameters are detailed in Table 6-1. Note that the operating system path specified for any of these initialization parameters must already exist; Oracle will not create the directory. Also, these directories must be writable by the operating system account that owns the Oracle software (which on most platforms is oracle).

Initialization Parameter	Description
DB_CREATE_FILE_DEST	The default operating system file directory where datafiles and tempfiles are created if no pathname is specified in the **create tablespace** command. This location is used for redo log files and control files if DB_CREATE_ONLINE_LOG_DEST_*n* is not specified.
DB_CREATE_ONLINE_LOG_DEST_*n*	Specifies the default location to store redo log files and control files when no pathname is specified for redo log files or control files at database-creation time. Up to five destinations can be specified with this parameter, allowing up to five multiplexed control files and five members of each redo log group.
DB_RECOVERY_FILE_DEST	Defines the default pathname in the server's file system where RMAN backups, archived redo logs, and flashback logs are located. Also used for redo log files and control files if neither DB_CREATE_FILE_DEST nor DB_CREATE_ONLINE_LOG_DEST_*n* is specified.

TABLE 6-1 *OMF-Related Initialization Parameters*

Bigfile Tablespaces

Bigfile tablespaces, introduced in Oracle 10*g*, take OMF files to the next level; in a bigfile tablespace, a single datafile is allocated, and it can be up to 8EB (exabytes, a million terabytes) in size. Bigfile tablespaces can only be locally managed with automatic segment space management. If a bigfile tablespace is used for automatic undo or for temporary segments, then segment space management must be set to MANUAL.

Bigfile tablespaces can save space in the System Global Area (SGA) and the control file because fewer datafiles need to be tracked; similarly, all **alter tablespace** commands on bigfile tablespaces need not refer to datafiles because one and only one datafile is associated with each bigfile tablespace. This moves the maintenance point from the physical (datafile) level to the logical (tablespace) level, simplifying administration. One downside to bigfile tablespaces is that a backup of a bigfile tablespace uses a single process; a number of smaller tablespaces, however, can be backed up using parallel processes and will most likely take less time to back up than a single bigfile tablespace.

Creating a bigfile tablespace is as easy as adding the **bigfile** keyword to the **create tablespace** command:

```
SQL> create bigfile tablespace whs01
2 datafile '/u06/oradata/whs01.dbf' size 10g;
Tablespace created.
```

If you are using OMF, then the **datafile** clause can be omitted. To resize a bigfile tablespace, you can use the **resize** clause:

```
SQL> alter tablespace whs01 resize 80g;
Tablespace altered.
```

In this scenario, even 80GB is not big enough for this tablespace, so we will let it autoextend 20GB at a time:

```
SQL> alter tablespace whs01 autoextend on next 20g;
Tablespace altered.
```

Notice in both cases that we do not need to refer to a datafile; there is only one datafile, and once the tablespace is created, we no longer need to worry about the details of the underlying datafile

and how it is managed.
Bigfile tablespaces are intended for use with Automatic Storage Management, discussed in the next section.

Automatic Storage Management

Using *Automatic Storage Management (ASM)* can significantly reduce the administrative overhead of managing space in a database because a DBA need only specify an ASM *disk group* when allocating space for a tablespace or other database object. Database files are automatically distributed among all available disks in a disk group, and the distribution is automatically updated whenever the disk configuration changes. For example, when a new disk volume is added to an existing disk group in an ASM instance, all datafiles within the disk group are redistributed to use the new disk volume. I introduced ASM in Chapter 4. In this section, I'll revisit some other key ASM concepts from a storage management point of view and provide more examples.
Because ASM automatically places datafiles on multiple disks, performance of queries and DML statements is improved because the I/O is spread out among several disks. Optionally, the disks in an ASM group can be mirrored to provide additional redundancy and performance benefits. Using ASM provides a number of other benefits. In many cases, an ASM instance with a number of physical disks can be used instead of a third-party volume manager or network-attached storage (NAS) subsystem. As an added benefit over volume managers, ASM maintenance operations do not require a shutdown of the database if a disk needs to be added or removed from a disk group.

SYSAUX Monitoring and Usage

The SYSAUX tablespace, introduced in Oracle 10*g*, is an auxiliary tablespace to the SYSTEM tablespace, and it houses data for several components of the Oracle database that either required their own tablespace or used the SYSTEM tablespace in previous releases of Oracle. These components include the Enterprise Manager Repository, formerly in the tablespace OEM_ REPOSITORY, as well as LogMiner, Oracle Spatial, and Oracle Text, all of which formerly used the SYSTEM tablespace for storing configuration information. The current occupants of the SYSAUX tablespace can be identified by querying the V$SYSAUX_OCCUPANTS view:

```
SQL> select occupant_name, occupant_desc, space_usage_kbytes
2 from v$sysaux_occupants;

OCCUPANT_NAME OCCUPANT_DESC                                    SPACE_USAGE_KBYTES
------------- ---------------------------------------- ------------------
LOGMNR        LogMiner                                                  7744
LOGSTDBY      Logical Standby                                            960
SMON_SCN_TIME Transaction Layer - SCN to TIME mapping                   3328
PL/SCOPE      PL/SQL Identifier Collection                               384
STREAMS       Oracle Streams                                            1024
XDB           XDB                                                      98816
AO            Analytical Workspace Object Table                        38208
XSOQHIST      OLAP API History Tables                                  38208
XSAMD         OLAP Catalog                                             15936
SM/AWR        Server Manageability - Automatic Workloa                131712
              d Repository
SM/ADVISOR    Server Manageability - Advisor Framework                  13248
SM/OPTSTAT    Server Manageability - Optimizer Statist                  52672
              ics History
SM/OTHER      Server Manageability - Other Components                    6016
STATSPACK     Statspack Repository                                         0
SDO           Oracle Spatial                                           47424
WM            Workspace Manager                                         7296
ORDIM         Oracle interMedia ORDSYS Components                      11200
```

```
ORDIM/PLUGINS Oracle interMedia ORDPLUGINS Components                0
ORDIM/SQLMM   Oracle interMedia SI_INFORMTN_SCHEMA Com               0
              ponents
EM            Enterprise Manager Repository                     155200
TEXT          Oracle Text                                         5568
ULTRASEARCH   Oracle Ultra Search                                  7616
ULTRASEARCH_D Oracle Ultra Search Demo User                       12288
EMO_USER
EXPRESSION_FI Expression Filter System                            3968
LTER
EM_MONITORING Enterprise Manager Monitoring User                  1536
_USER
TSM           Oracle Transparent Session Migration User            256
SQL_MANAGEMEN SQL Management Base Schema                          1728
T_BASE
AUTO_TASK     Automated Maintenance Tasks                          320
JOB_SCHEDULER Unified Job Scheduler                                576

29 rows selected.
```

components of the Oracle database will be unavailable; the core functionality of the database will be unaffected. In any case, the SYSAUX tablespace helps to take the load off of the SYSTEM tablespace during normal operation of the database.

To monitor the usage of the SYSAUX tablespace, you can query the column SPACE_USAGE_ KBYTES on a routine basis, and it can alert the DBA when the space usage grows beyond a certain level. If the space usage for a particular component requires a dedicated tablespace to be allocated for the component, such as for the EM Repository, the procedure identified in the MOVE_ PROCEDURE column of the V$SYSAUX_OCCUPANTS view will move the application to another tablespace:

SQL> **select occupant_name, move_procedure from v$sysaux_occupants**
2 **where occupant_name = 'EM';**
OCCUPANT_NAME MOVE_PROCEDURE
--- ---
EM emd_maintenance.move_em_tblspc

In the following scenario, we know that we will be adding several hundred nodes to our management repository in the near future. Because we want to keep the SYSAUX tablespace from growing too large, we decide to create a new tablespace to hold only the Enterprise Manager data. In the following example, we'll create a new tablespace and move the Enterprise Manager schema into the new tablespace:

SQL> **create tablespace EM_REP**
2> **datafile '+DATA' size 250m autoextend on next 100m;**
Tablespace created.
SQL> **execute sysman.emd_maintenance.move_em_tblspc('EM_REP');**
PL/SQL procedure successfully completed.
SQL> **select occupant_name, occupant_desc, space_usage_kbytes**
2> **from v$sysaux_occupants**
3> **where occupant_name = 'EM';**
OCCUPANT_NAME OCCUPANT_DESC SPACE_USAGE_KBYTES
--- --- ---
EM Enterprise Manager Repository 0
1 row selected.

Since the current space allocation for the EM tools is about 150MB, a tablespace starting at a size of 250MB with additional extents of 100MB each should be sufficient for most environments. Note that the row for Enterprise Manager is still in V$SYSAUX_OCCUPANTS; even though it is

not taking up any space in the SYSAUX tablespace, we may want to move its metadata back into SYSAUX at some point in the future. Therefore, we may need to query V$SYSAUX_OCCUPANTS again to retrieve the move procedure. We use the same procedure for moving the application into and out of SYSAUX:

SQL> **execute sysman.emd_maintenance.move_em_tblspc('SYSAUX');**

PL/SQL procedure successfully completed.

If a component is not being used in the database at all, such as Ultra Search, a negligible amount of space is used in the SYSAUX tablespace.

Archived Redo Log File Management

It is important to consider space management for objects that exist outside of the database, such as archived redo log files. In ARCHIVELOG mode, an online redo log file is copied to the destination(s) specified by LOG_ARCHIVE_DEST_*n* (where *n* is a number from 1 to 10) or by DB_RECOVERY_ FILE_DEST (the flash recovery area) if none of the LOG_ARCHIVE_DEST_*n* values are set. The redo log being copied must be copied successfully to at least one of the destinations before it can be reused by the database. The LOG_ARCHIVE_MIN_SUCCEED_DEST parameter defaults to 1 and must be at least 1. If none of the copy operations are successful, the database will be suspended until at least one of the destinations receives the log file. Running out of disk space is one possible reason for this type of failure.

If the destination for the archived log files is on a local file system, an operating system shell script can monitor the space usage of the destination, or it can be scheduled with DBMS_ SCHEDULER or with OEM.

Built-in Space Management Tools

Oracle 10*g* provides a number of built-in tools that a DBA can use on demand to determine if there are any problems with disk space in the database. Most, if not all, of these tools can be manually configured and run by calling the appropriate built-in package. In this section, we'll cover the packages and procedures used to query the database for space problems or advice on space management. In addition, I'll show you the new initialization parameter used by the Automatic Diagnostic Repository to identify the alert and trace file location. Later in this chapter, you'll see how some of these tools can be automated to notify the DBA via e-mail or pager when a problem is imminent; many, if not all, of these tools are available on demand via the EM Database Control web interface.

Segment Advisor

Frequent inserts, updates, and deletes on a table may, over time, leave the space within a table fragmented. Oracle can perform *segment shrink* on a table or index. Shrinking the segment makes the free space in the segment available to other segments in the tablespace, with the potential to improve future DML operations on the segment because fewer blocks may need to be retrieved for the DML operation after the segment shrink. Segment shrink is very similar to online table redefinition in that space in a table is reclaimed. However, segment shrink can be performed in place without the additional space requirements of online table redefinition.

To determine which segments will benefit from segment shrink, you can invoke *Segment Advisor* to perform growth trend analysis on specified segments. In this section, we'll invoke Segment Advisor on some candidate segments that may be vulnerable to fragmentation.

In the example that follows, we'll set up Segment Advisor to monitor the HR.EMPLOYEES table. In recent months, there has been high activity on this table; in addition, a new column, WORK_RECORD, has been added to the table, which HR uses to maintain comments about the employees:

SQL> **alter table hr.employees add (work_record varchar2(4000));**

Table altered.

SQL> alter table hr.employees enable row movement;

Table altered.

We have enabled ROW MOVEMENT in the table so that shrink operations can be performed on the table if recommended by Segment Advisor.

After Segment Advisor has been invoked to give recommendations, the findings from Segment Advisor are available in the DBA_ADVISOR_FINDINGS data dictionary view. To show the potential benefits of shrinking segments when Segment Advisor recommends a shrink operation, the view DBA_ADVISOR_RECOMMENDATIONS provides the recommended shrink operation along with the potential savings, in bytes, for the operation.

To set up Segment Advisor to analyze the HR.EMPLOYEES table, we will use an anonymous PL/SQL block, as follows:

```
-- begin Segment Advisor analysis for HR.EMPLOYEES
-- rev. 1.1 RJB 07/07/2007
--
-- SQL*Plus variable to retrieve the task number from Segment Advisor
variable task_id number
-- PL/SQL block follows
declare
name varchar2(100);
descr varchar2(500);
obj_id number;
begin
name := ''; -- unique name generated from create_task
descr := 'Check HR.EMPLOYEE table';
dbms_advisor.create_task
('Segment Advisor', :task_id, name, descr, NULL);
dbms_advisor.create_object
(name, 'TABLE', 'HR', 'EMPLOYEES', NULL, NULL, obj_id);
dbms_advisor.set_task_parameter(name, 'RECOMMEND_ALL', 'TRUE');
dbms_advisor.execute_task(name);
end;
PL/SQL procedure successfully completed.
SQL> print task_id
TASK_ID
----------
384
SQL>
```

The procedure DBMS_ADVISOR.CREATE_TASK specifies the type of advisor; in this case, it is Segment Advisor. The procedure will return a unique task ID and an automatically generated name to the calling program; we will assign our own description to the task.

Within the task, identified by the uniquely generated name returned from the previous procedure, we identify the object to be analyzed with DBMS_ADVISOR.CREATE_OBJECT. Depending on the type of object, the second through the sixth arguments vary. For tables, we only need to specify the schema name and the table name.

Using DBMS_ADVISOR.SET_TASK_PARAMETER, we tell Segment Advisor to give all possible recommendations about the table. If we want to turn off recommendations for this task, we would specify FALSE instead of TRUE for the last parameter.

Finally, we initiate the Segment Advisor task with the DBMS_ADVISOR.EXECUTE_TASK procedure. Once it is done, we display the identifier for the task so we can query the results in the appropriate data dictionary views.

Now that we have a task number from invoking Segment Advisor, we can query DBA_ ADVISOR_FINDINGS to see what we can do to improve the space utilization of the HR.EMPLOYEES table:

```
SQL> select owner, task_id, task_name, type,
2 message, more_info from dba_advisor_findings
3 where task_id = 384;
OWNER TASK_ID TASK_NAME TYPE
---------- ------- --------- ------
RJB 6 TASK_00003 INFORMATION
MESSAGE
--------------------------------------------------
Perform shrink, estimated savings is 107602 bytes.
MORE_INFO
--------------------------------------------------------------------
Allocated Space:262144: Used Space:153011: Reclaimable Space :107602:
```

The results are fairly self-explanatory. We can perform a segment shrink operation on the table to reclaim space from numerous insert, delete, and update operations on the HR.EMPLOYEES table. Because the WORK_RECORD column was added to the HR.EMPLOYEES table after the table was already populated, we may have created some chained rows in the table; in addition, since the WORK_RECORD column can be up to 4000 bytes long, updates or deletes of rows with big WORK_RECORD columns may create blocks in the table with free space that can be reclaimed.

The view DBA_ADVISOR_RECOMMENDATIONS provides similar information:

```
SQL> select owner, task_id, task_name, benefit_type
2 from dba_advisor_recommendations
3 where task_id = 384;
OWNER TASK_ID TASK_NAME
---------- ------- ----------
RJB 384 TASK_00003
BENEFIT_TYPE
--------------------------------------------------
Perform shrink, estimated savings is 107602 bytes.
```

In any case, we will shrink the segment HR.EMPLOYEES to reclaim the free space. As an added time-saving benefit to the DBA, the SQL needed to perform the shrink is provided in the view DBA_ADVISOR_ACTIONS:

```
SQL> select owner, task_id, task_name, command, attr1
2 from dba_advisor_actions where task_id = 384;
OWNER TASK_ID TASK_NAME COMMAND
---------- ------- ---------- -----------------
RJB 6 TASK_00003 SHRINK SPACE
ATTR1
--------------------------------------------------
alter table HR.EMPLOYEES shrink space
1 row selected.
SQL> alter table HR.EMPLOYEES shrink space;
Table altered.
```

As mentioned earlier, the shrink operation does not require extra disk space and does not prevent access to the table during the operation, except for a very short period of time at the end of the process to free the unused space. All indexes are maintained on the table during the operation. In addition to freeing up disk space for other segments, there are other benefits to shrinking a

segment. Cache utilization is improved because fewer blocks need to be in the cache to satisfy SELECT or other DML statements against the segment. Also, because the data in the segment is more compact, the performance of full table scans is improved.

There are a couple of caveats and minor restrictions. First, segment shrink will not work on LOB segments if you are using Oracle Database 10*g*. Online table reorganization is a more appropriate method in this case. Also, segment shrink is not allowed on a table that contains any function-based indexes regardless of whether you are using Oracle Database 10*g* or 11*g*.

Undo Advisor and the Automatic Workload Repository

New to Oracle 10*g*, the *Undo Advisor* provides tuning information for the undo tablespace, whether it's sized too large, it's too small, or the undo retention (via the initialization parameter UNDO_RETENTION) is not set optimally for the types of transactions that occur in the database. Using the Undo Advisor is similar to using the Segment Advisor in that we will call the DBMS_ADVISOR procedures and query the DBA_ADVISOR_* data dictionary views to see the results of the analysis.

The Undo Advisor, however, relies on another feature new to Oracle 10*g*—the *Automatic Workload Repository (AWR).* The Automatic Workload Repository, built into every Oracle database, contains snapshots of all key statistics and workloads in the database at 60-minute intervals by default. The statistics in the AWR are kept for seven days, after which the oldest statistics are dropped. Both the snapshot intervals and the retention period can be adjusted to suit your environment, however. The AWR maintains the historical record of how the database is being used over time and helps to diagnose and predict problems long before they can cause a database outage.

To set up Undo Advisor to analyze undo space usage, we will use an anonymous PL/SQL block similar to what we used for Segment Advisor. Before we can use Segment Advisor, however, we need to determine the timeframe to analyze. The data dictionary view DBA_HIST_SNAPSHOT contains the snapshot numbers and date stamps; we will look for the snapshot numbers from 8:00 P.M. Saturday, July 21, 2007 through 9:30 P.M. Saturday, July 21, 2007:

```
SQL> select snap_id, begin_interval_time, end_interval_time
2 from DBA_HIST_SNAPSHOT
3 where begin_interval_time > '21-Jul-07 08.00.00 PM' and
4 end_interval_time < '21-Jul-07 09.31.00 PM'
5 order by end_interval_time desc;
SNAP_ID BEGIN_INTERVAL_TIME END_INTERVAL_TIME
---------- --------------------------- ---------------------------
8 21-JAN-07 09.00.30.828 PM 21-JAN-07 09.30.14.078 PM
7 21-JAN-07 08.30.41.296 PM 21-JAN-07 09.00.30.828 PM
6 21-JAN-07 08.00.56.093 PM 21-JAN-07 08.30.41.296 PM
```

Given these results, we will use a SNAP_ID range from 6 to 8 when we invoke Undo Advisor. The PL/SQL anonymous block is as follows:

```
-- begin Undo Advisor analysis
-- rev. 1.1 RJB 7/16/2007
--
-- SQL*Plus variable to retrieve the task number from Segment Advisor
variable task_id number
declare
task_id number;
name varchar2(100);
descr varchar2(500);
obj_id number;
```

```
begin
name := ''; -- unique name generated from create_task
descr := 'Check Undo Tablespace';
dbms_advisor.create_task
('Undo Advisor', :task_id, name, descr);
dbms_advisor.create_object
(name, 'UNDO_TBS', NULL, NULL, NULL, 'null', obj_id);
dbms_advisor.set_task_parameter(name, 'TARGET_OBJECTS', obj_id);
dbms_advisor.set_task_parameter(name, 'START_SNAPSHOT', 6);
dbms_advisor.set_task_parameter(name, 'END_SNAPSHOT', 8);
dbms_advisor.set_task_parameter(name, 'INSTANCE', 1);
dbms_advisor.execute_task(name);
end;
PL/SQL procedure successfully completed.
SQL> print task_id
TASK_ID
-------
527
```

As with the Segment Advisor, we can review the DBA_ADVISOR_FINDINGS view to see the problem and the recommendations.

```
SQL> select owner, task_id, task_name, type,
2 message, more_info from dba_advisor_findings
3 where task_id = 527;
OWNER TASK_ID TASK_NAME TYPE
---------- ------- ---------- -------------
RJB 527 TASK_00003 PROBLEM
MESSAGE
-----------------------------------------------------
The undo tablespace is OK.
MORE_INFO
-----------------------------------------------------------------
```

In this particular scenario, Undo Advisor indicates that there is enough space allocated in the undo tablespace to handle the types and volumes of queries run against this database.

Index Usage

Although indexes provide a tremendous benefit by speeding up queries, they can have an impact on space usage in the database. If an index is not being used at all, the space occupied by an index can be better used elsewhere; if we don't need the index, we also can save processing time for insert, update, and delete operations that have an impact on the index. Index usage can be monitored with the dynamic performance view V$OBJECT_USAGE. In our HR schema, we suspect that the index on the JOB_ID column of the EMPLOYEES table is not being used. We turn on monitoring for this index as follows:

```
SQL> alter index hr.emp_job_ix monitoring usage;
Index altered.
```

We take a quick look at the V$OBJECT_USAGE view to make sure this index is being monitored:

```
SQL> select * from v$object_usage;
INDEX_NAME TABLE_NAME MON USED START_MONITORING
--------------- --------------- --- ---- -------------------
EMP_JOB_IX EMPLOYEES YES NO 07/24/2007 10:04:55
```

The column USED will tell us if this index is accessed to satisfy a query. After a full day of

typical user activity, we check V$OBJECT_USAGE again and then turn off monitoring:

```
SQL> alter index hr.emp_job_ix nomonitoring usage;
Index altered.
SQL> select * from v$object_usage;
INDEX_NAME TABLE_NAME MON USED START_MONITORING END_MONITORING
---------- --------------- --- ---- ------------------- ---------------
EMP_JOB_IX EMPLOYEES NO YES 07/24/2007 10:04:55 07/25/2007 11:39:45
```

Sure enough, the index appears to be used at least once during a typical day.

On the other end of the spectrum, an index may be accessed too frequently. If key values are inserted, updated, and deleted frequently, an index can become less efficient in terms of space usage. The following commands can be used as a baseline for an index after it is created, and then run periodically to see if the space usage becomes inefficient:

```
SQL> analyze index hr.emp_job_ix validate structure;
Index analyzed.
SQL> select pct_used from index_stats where name = 'EMP_JOB_IX';
PCT_USED
----------
78
```

The PCT_USED column indicates the percentage of the allocated space for the index in use. Over time, the EMPLOYEES table is heavily used, due to the high turnover rate of employees at the company, and this index, among others, is not using its space efficiently, as indicated by the following **analyze** command and **select** query, so we decide that a rebuild is in order:

```
SQL> analyze index hr.emp_job_ix validate structure;
Index analyzed.
SQL> select pct_used from index_stats where name = 'EMP_JOB_IX';
PCT_USED
----------
26
SQL> alter index hr.emp_job_ix rebuild online;
Index altered.
```

Notice the inclusion of the **online** option in the **alter index . . . rebuild** statement. The indexed table can remain online with minimal overhead while the index is rebuilding. In rare circumstances, such as on longer key lengths, you may not be able to use the **online** option.

Space Usage Warning Levels

Earlier in this chapter, we reviewed the data dictionary view DBA_THRESHOLDS, which contains a list of the active metrics to measure a database's health. In a default installation of Oracle 11*g*, use the following **select** statement to see some of the 22 built-in thresholds:

```
SQL> select metrics_name, warning_operator warn, warning_value wval,
2 critical_operator crit, critical_value cval,
3 consecutive_occurrences consec
4 from dba_thresholds;
```

```
METRICS_NAME                   WARN WVAL          CRIT CVAL          CONSEC
------------------------------ ---- ------------ ---- ------------ ------
Average Users Waiting Counts   GT   10           NONE               3
. . .
Blocked User Session Count     GT   0            NONE               15
Current Open Cursors Count     GT   1200         NONE               3
Database Time Spent Waiting (%) GT  30           NONE               3
. . .
Logons Per Sec                 GE   100          NONE               2
Session Limit %                GT   90           GT   97            3
Tablespace Bytes Space Usage   LE   0            LE   0             1
Tablespace Space Usage         GE   85           GE   97            1

22 rows selected.
```

In terms of space usage, we see that the warning level for a given tablespace is when the tablespace is 85 percent full, and the space is at a critical level when it reaches 97 percent full. In addition, this condition need only occur during one reporting period, which by default is one minute. For the other conditions in this list, the condition must be true anywhere between 2 and 15 consecutive reporting periods before an alert is issued.

To change the level at which an alert is generated, we can use the DBMS_SERVER_ALERT. SET_THRESHOLD procedure. In this example, we want to be notified sooner if a tablespace is running out of space, so we will update the warning threshold for alert notification from 85 percent down to 60 percent:

```
--
-- PL/SQL anonymous procedure to update the Tablespace Space Usage threshold
--
declare
/* OUT */
warning_operator number;
warning_value varchar2(100);
critical_operator number;
critical_value varchar2(100);
observation_period number;
consecutive_occurrences number;
/* IN */
metrics_id number;
instance_name varchar2(50);
object_type number;
object_name varchar2(50);
new_warning_value varchar2(100) := '60';
begin
metrics_id := DBMS_SERVER_ALERT.TABLESPACE_PCT_FULL;
object_type := DBMS_SERVER_ALERT.OBJECT_TYPE_TABLESPACE;
instance_name := 'dw';
object_name := NULL;
-- retrieve the current values with get_threshold
dbms_server_alert.get_threshold(
metrics_id, warning_operator, warning_value,
critical_operator, critical_value,
observation_period, consecutive_occurrences,
instance_name, object_type, object_name);
-- update the warning threshold value from 85 to 60
```

```
dbms_server_alert.set_threshold(
metrics_id, warning_operator, new_warning_value,
critical_operator, critical_value,
observation_period, consecutive_occurrences,
instance_name, object_type, object_name);
end;
```

PL/SQL procedure successfully completed.

Checking DBA_THRESHOLDS again, we see the warning level has been changed to 60 percent:

SQL> **select metrics_name, warning_operator warn, warning_value wval**
2 **from dba_thresholds;**

```
METRICS_NAME                   WARN WVAL
------------------------------ ---- -------------
Average Users Waiting Counts   GT   10
. . .
Blocked User Session Count     GT   0
Current Open Cursors Count     GT   1200
Database Time Spent Waiting (%) GT   30
. . .
Logons Per Sec                 GE   100
Session Limit %                GT   90
Tablespace Bytes Space Usage   LE   0
Tablespace Space Usage         GE   60

22 rows selected.
```

Managing Alert and Trace Files with ADR

New to Oracle Database 11*g*, the Automatic Diagnostic Repository (ADR) is a system-managed repository for storing database alert logs, trace files, and any other diagnostic data previously controlled by several other initialization parameters.

The initialization parameter DIAGNOSTIC_DEST sets the base location for all diagnostic directories; in the dw database I use throughout this chapter, the value of the parameter DIAGNOSTIC_DEST is /u01/app/oracle. Figure 6-4 shows a typical directory structure starting with the subdirectory /u01/app/oracle/diag.

Notice that there are separate directories for the ASM databases and the database (rdbms) instances; within the rdbms directory, you can see the dw directory twice: the first-level directory is the database dw, and the second dw is the instance dw. If this were a Real Application Clusters (RAC) database, you would see each instance of the dw database under the first-level dw directory. In fact, Oracle strongly recommends that all instances within a RAC database have the same value for DIAGNOSTIC_DEST.

Because the location of all logging and diagnostic information is controlled by the initialization parameter DIAGNOSTIC_DEST, the following initialization parameters are ignored:

BACKGROUND_DUMP_DEST
USER_DUMP_DEST
CORE_DUMP_DEST

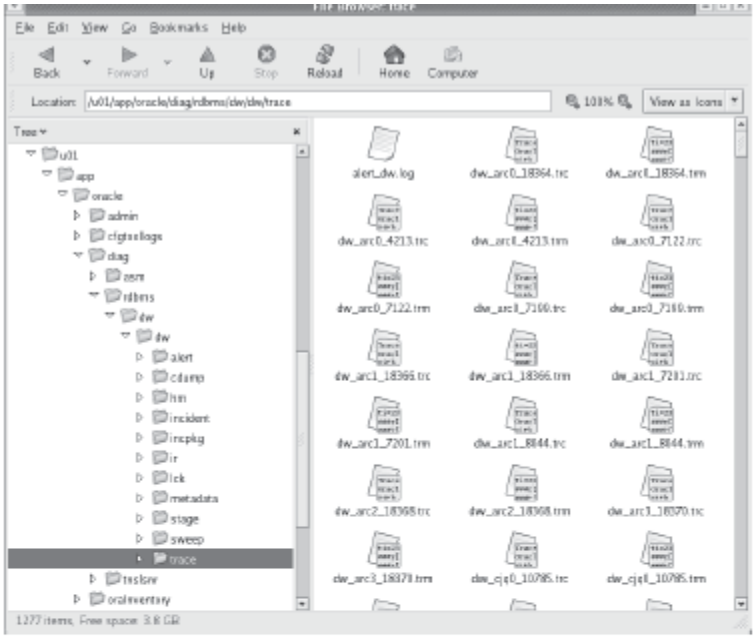


FIGURE 6-4 *ADR directory structure*

For backward compatibility, however, you can still use these as read-only parameters to determine the location of the alert log, trace files, and core dumps:

```
SQL> show parameter dump_dest

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
background_dump_dest                 string      /u01/app/oracle/diag/rdbms/dw/
                                                 dw/trace
core_dump_dest                       string      /u01/app/oracle/diag/rdbms/dw/
                                                 dw/cdump
user_dump_dest                       string      /u01/app/oracle/diag/rdbms/dw/
                                                 dw/trace
```

You can still alter the values for these parameters, but they are ignored by ADR. Alternatively, you can use the view V$DIAG_INFO to find all diagnostic-related directories for the instance:

```
SQL> select name, value from v$diag_info;
NAME VALUE
---------------------- ----------------------------------------
Diag Enabled TRUE
ADR Home /u01/app/oracle/diag/rdbms/dw/dw
Diag Trace /u01/app/oracle/diag/rdbms/dw/dw/trace
Diag Alert /u01/app/oracle/diag/rdbms/dw/dw/alert
Diag Incident /u01/app/oracle/diag/rdbms/dw/dw/incident
Diag Cdump /u01/app/oracle/diag/rdbms/dw/dw/cdump
Health Monitor /u01/app/oracle/diag/rdbms/dw/dw/hm
Default Trace File /u01/app/oracle/diag/rdbms/dw/dw/trace/dw_ora
_28810.trc
Active Problem Count 0
Active Incident Count 0
11 rows selected.
```

Transaction Basics

A *transaction* is a collection of SQL DML statements that is treated as a logical unit; the failure of

any of the statements in the transaction implies that none of the other changes made to the database in the transaction should be permanently saved to the database. Once the DML statements in the transaction have successfully completed, the application or SQL*Plus user will issue a **commit** to make the changes permanent. In the classic banking example, a transaction that transfers a dollar amount from one account to another is successful only if both the debit of one account (an **update** of the savings account balance) and the credit of another account (an **update** of the checking account balance) are both successful. Failure of either or both statements invalidates the entire transaction. When the application or SQL*Plus user issues a **commit**, if only one or the other **update** statement is successful, the bank will have some very unhappy customers!

A transaction is initiated implicitly. After a **commit** of a previous transaction is completed, and at least one row of a table is inserted, updated, or deleted, a new transaction is implicitly created. Also, any DDL commands such as **create table** and **alter index** will commit an active transaction and begin a new transaction. You can name a transaction by using the **set transaction . . . name '*transaction_name*'** command. Although this provides no direct benefit to the application, the name assigned to the transaction is available in the dynamic performance view V$TRANSACTION and allows a DBA to monitor long-running transactions; in addition, the transaction name helps the DBA resolve in-doubt transactions in distributed database environments. The **set transaction** command, if used, must be the first statement within the transaction.

Within a given transaction, you can define a *savepoint.* A savepoint allows the sequence of DML commands within a transaction to be partitioned so that it is possible to roll back one or more of the DML commands after the savepoint, and subsequently submit additional DML commands or commit the DML commands performed before the savepoint. Savepoints are created with the **savepoint *savepoint_name*** command. To undo the DML commands since the last savepoint, you use the command **rollback to savepoint *savepoint_name.***

A transaction is implicitly committed if a user disconnects from Oracle normally; if the user process terminates abnormally, the most recent transaction is rolled back.

Undo Basics

Undo tablespaces facilitate the rollback of logical transactions. In addition, undo tablespaces support a number of other features, including read consistency, various database-recovery operations, and Flashback functions.

Rollback

As described in the previous section, any DML command within a transaction—whether the transaction is one or one hundred DML commands—may need to be rolled back. When a DML command makes a change to a table, the old data values changed by the DML command are recorded in the undo tablespace within a system-managed undo segment or a rollback segment. When an entire transaction is rolled back (that is, a transaction without any savepoints), Oracle undoes all the changes made by DML commands since the beginning of the transaction using the corresponding undo records, releases the locks on the affected rows, if any, and the transaction ends.

If part of a transaction is rolled back to a savepoint, Oracle undoes all changes made by DML commands after the savepoint. All subsequent savepoints are lost, all locks obtained after the savepoint are released, and the transaction remains active.

Read Consistency

Undo provides *read consistency* for users who are reading rows that are involved in a DML transaction by another user. In other words, all users who are reading the affected rows will see no changes in the rows until they issue a new query after the DML user commits the transaction. Undo segments are used to reconstruct the datablocks back to a read-consistent version and, as a result, provide the previous values of the rows to any user issuing a **select** statement before the transaction commits.

For example, user CLOLSEN begins a transaction at 10:00 that is expected to commit at 10:15, with various updates and insertions to the EMPLOYEES table. As each **insert**, **update**, and **delete** occurs on the EMPLOYEES table, the old values of the table are saved in the undo tablespace. When the user SUSANP issues a **select** statement against the EMPLOYEES table at 10:08, none of the changes made by CLOLSEN are visible to anyone except CLOLSEN; the undo tablespace provides the previous values of CLOLSEN's changes for SUSANP and all other users. Even if the query from SUSANP does not finish until 10:20, the table still appears to be unchanged until a new query is issued after the changes are committed. Until CLOLSEN performs a **commit** at 10:15, the data in the table appears unchanged as of 10:00.

If there is not enough undo space available to hold the previous values of changed rows, the user issuing the **select** statement may receive an "ORA-01555: Snapshot Too Old" error. Later in this chapter, we will discuss ways in which we can address this issue.

Database Recovery

Undo tablespaces are also a key component of instance recovery. The online redo logs bring both committed and uncommitted transactions forward to the point in time of the instance crash; the undo data is used to roll back any transactions that were not committed at the time of the crash or instance failure.

Flashback Operations

The data in the undo tablespace is used to support the various types of Flashback options: Flashback Table, Flashback Query, and the package DBMS_FLASHBACK. Flashback Table will restore a table as of a point of time in the past, Flashback Query lets you view a table as of an SCN or time in the past, and DBMS_FLASHBACK provides a programmatic interface for Flashback operations. Flashback Data Archive, new to Oracle Database 11*g*, stores and tracks all transactions on a specified table for a specified time period; in a nutshell, Flashback Data Archive stores undo data for a specific table in a specific tablespace outside of the global undo tablespace. Also new to Oracle Database 11*g* is Flashback Transaction Backout that can roll back an already committed transaction and its dependent transactions while the database is online. All these Flashback options are covered in more detail at the end of this chapter.

Managing Undo Tablespaces

Creating and maintaining undo tablespaces is a "set it and forget it" operation once the undo requirements of the database are understood. Within the undo tablespace, Oracle automatically creates, sizes, and manages the undo segments, unlike previous versions of Oracle in which the DBA would have to manually size and constantly monitor rollback segments.

In the next couple sections, we'll review the processes used to create and manage undo tablespaces, including the relevant initialization parameters. In addition, we'll review some scenarios where we may create more than one undo tablespace and how to switch between undo tablespaces.

Creating Undo Tablespaces

Undo tablespaces can be created in two ways: at database creation or with the **create tablespace** command after the database is created. As with any other tablespace in Oracle 10*g*, the undo tablespace can be a bigfile tablespace, further easing the maintenance of undo tablespaces.

Creating an Undo Tablespace with CREATE DATABASE

A database may have more than one undo tablespace, although only one can be active at a time. Here's what creating an undo tablespace at database creation looks like:

```
create database ord
user sys identified by ds88dkw2
user system identified by md78s233
sysaux datafile '/u02/oradata/ord/sysaux001.dbf' size 1g
default temporary tablespace temp01
```

tempfile '/u03/oradata/ord/temp001.dbf' size 150m
undo tablespace undotbs01
datafile '/u01/oradata/ord/undo001.dbf' size 500m;

If the undo tablespace cannot be successfully created in the **create database** command, the entire operation fails. The error must be corrected, any files remaining from the operation must be deleted, and the command must be reissued.

Although the **undo tablespace** clause in the **create database** command is optional, if it is omitted and Automatic Undo Management is enabled, an undo tablespace is still created with an autoextensible datafile with an initial size of 10MB and the default name SYS_UNDOTBS.

Creating an Undo Tablespace with CREATE TABLESPACE

Any time after the database is created, a new undo tablespace can be created. An undo tablespace is created just as any other tablespace with the addition of the **undo** keyword:

create **undo** tablespace undotbs02
datafile '/u01/oracle/rbdb1/undo0201.dbf'
size 500m reuse autoextend on;

Depending on the volatility of the database or the expectation that the undo needs of the database may increase dramatically in the future, we start out this tablespace at only 500MB and allow it to grow.

Extents in an undo tablespace must be system managed; in other words, you can only specify **extent management** as **local autoallocate**.

Creating an Undo Tablespace Using EM Database Control

Creating an undo tablespace is straightforward using Enterprise Manager Database Control. From the Server tab on the home page, click the Tablespaces link. You will be presented with a list of existing tablespaces; click the Create button.

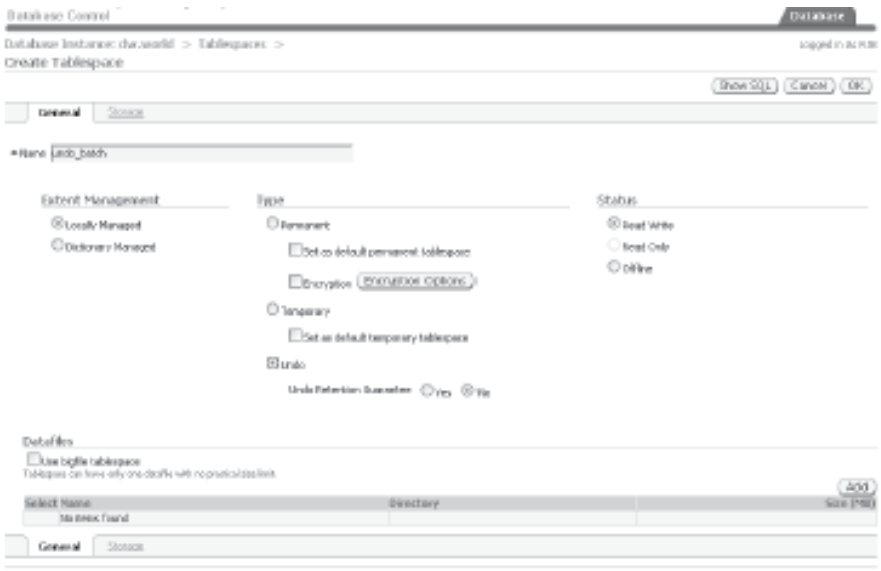


FIGURE 7-1 *Using EM Database Control to create an undo tablespace*

FIGURE 7-2 *Specifying a datafile for a new undo tablespace*

Modifying Undo Tablespaces

The following operations are allowed on undo tablespaces:

- Adding a datafile to an undo tablespace
- Renaming a datafile in an undo tablespace
- Changing an undo tablespace's datafile to online or offline
- Beginning or ending an open tablespace backup (**alter tablespace undotbs begin backup**)
- Enabling or disabling the undo retention guarantee

Using OMF for Undo Tablespaces

In addition to using a bigfile tablespace for undo tablespaces, you can also use OMF to automatically name (and locate, if you're not using ASM) an undo tablespace; the initialization parameter DB_CREATE_FILE_DEST contains the location where an undo tablespace will be created if the **datafile** clause is not specified in the **create undo tablespace** command. In the following example, we create an undo tablespace using OMF in an ASM disk group:

```
SQL> show parameter db_create_file_dest
NAME TYPE VALUE
------------------------------------ ----------- ------------------------------
db_create_file_dest string +DATA
SQL> create undo tablespace undo_bi;
Tablespace created.
SQL> select ts.name ts_name, df.name df_name, bytes
2 from v$tablespace ts join v$datafile df using(ts#)
3 where ts.name = 'UNDO_BI';
TS_NAME DF_NAME BYTES
------------ ------------------------------------------ ----------
UNDO_BI +DATA/dw/datafile/undo_bi.275.629807457 104857600
SQL>
```

Because we did not specify a datafile size either, the tablespace defaults to a size of 100MB; in addition, the datafile is autoextensible with an unlimited maximum size, limited only by the file system.

Undo Tablespace Dynamic Performance Views

A number of dynamic performance views and data dictionary views contain information about undo tablespaces, user transactions, and undo segments.

Undo Tablespace Initialization Parameters

In the following sections, we'll describe the initialization parameters needed to specify the undo tablespace for the database as well as control how long Oracle will retain undo information in the database.

UNDO_MANAGEMENT

The parameter UNDO_MANAGEMENT defaults to MANUAL in Oracle Database 10*g*, and AUTO in Oracle Database 11*g*. Setting the parameter UNDO_MANAGEMENT to AUTO places the database in Automatic Undo Management mode. At least one undo tablespace must exist in the database for this parameter to be valid, whether UNDO_TABLESPACE is specified or not. UNDO_MANAGEMENT is not a dynamic parameter; therefore, the instance must be restarted whenever UNDO_MANAGEMENT is changed from AUTO to MANUAL, or vice versa.

UNDO_TABLESPACE

The UNDO_TABLESPACE parameter specifies which undo tablespace will be used for Automatic Undo Management. If UNDO_MANAGEMENT is not specified or set to MANUAL, and UNDO_TABLESPACE is specified, the instance will not start.

View	Description
DBA_TABLESPACES	Tablespace names and characteristics, including the CONTENTS column, which can be PERMANENT, TEMPORARY, or UNDO; the undo RETENTION column is NOT APPLY, GUARANTEE, or NOGUARANTEE.
DBA_UNDO_EXTENTS	All undo segments in the database, including their size, their extents, the tablespace where they reside, and current status (EXPIRED or UNEXPIRED).
V$UNDOSTAT	The amount of undo usage for the database at ten-minute intervals; contains at most 1008 rows (7 days).
V$ROLLSTAT	Rollback segment statistics, including size and status.
V$TRANSACTION	Contains one row for each active transaction for the instance.

TABLE 7-1 *Undo Tablespace Views*

Conversely, if UNDO_MANAGEMENT is set to AUTO and there is no undo tablespace in the database, the instance will start, but then the SYSTEM rollback segment will be used for all undo operations, and a message is written to the alert log. Any user DML that attempts to make changes in non-SYSTEM tablespaces will, in addition, receive the error message "ORA-01552: cannot use system rollback segment for non-system tablespace 'USERS,'" and the statement fails.

UNDO_RETENTION

UNDO_RETENTION specifies a minimum amount of time that undo information is retained for queries. In automatic undo mode, UNDO_RETENTION defaults to 900 seconds. This value is valid only if there is enough space in the undo tablespace to support read-consistent queries; if active transactions require additional undo space, an unexpired undo may be used to satisfy the active transactions and may cause "ORA-01555: Snapshot Too Old" errors.

The column TUNED_UNDORETENTION of the dynamic performance view V$UNDOSTAT gives the tuned undo retention time for each time period; the status of the undo tablespace usage is updated in V$UNDOSTAT every ten minutes:

```
SQL> show parameter undo_retention
NAME TYPE VALUE
------------------------------------ ----------- ------------
undo_retention integer 900
SQL> select to_char(begin_time,'yyyy-mm-dd hh24:mi'),
2 undoblks, txncount, tuned_undoretention
3 from v$undostat where rownum = 1;
TO_CHAR(BEGIN_TI UNDOBLKS TXNCOUNT TUNED_UNDORETENTION
---------------- ---------- ---------- -------------------
2007-08-05 16:07 9 89 900
1 row selected.
SQL>
```

Because the transaction load is very light during the most recent time period, and the instance has just recently started up, the tuned undo retention value is the same as the minimum specified in the UNDO_RETENTION initialization parameter: 900 seconds (15 minutes).

Multiple Undo Tablespaces

As mentioned earlier in this chapter, a database can have multiple undo tablespaces, but only one of them can be active for a given instance at any one time. In this section, we'll show an example of switching to a different undo tablespace while the database is open.

In our **dw** database, we have three undo tablespaces:

```
SQL> select tablespace_name, status from dba_tablespaces
2 where contents = 'UNDO';
TABLESPACE_NAME STATUS
```

---------- ---------
UNDOTBS1 ONLINE
UNDO_BATCH ONLINE
UNDO_BI ONLINE
2 rows selected.
But only one of the undo tablespaces is active:
SQL> **show parameter undo_tablespace**
NAME TYPE VALUE
---------- ---------- ----------
undo_tablespace string UNDOTBS1
For overnight processing, we change the undo tablespace from UNDOTBS1 to the tablespace UNDO_BATCH, which is much larger to support higher DML activity. The disk containing the daytime undo tablespace is much faster but has a limited amount of space; the disk containing the overnight undo tablespace is much larger, but slower. As a result, we use the smaller undo tablespace to support OLTP during the day, and the larger undo tablespace for our data mart and data warehouse loads, as well as other aggregation activities, at night when response time is not as big of an issue.
About the time the undo tablespace is going to be switched, the user HR is performing some maintenance operations on the HR.EMPLOYEES table, and she has an active transaction in the current undo tablespace:
SQL> **connect hr/hr@dw;**
Connected.
SQL> **set transaction name 'Employee Maintenance';**
Transaction set.
SQL> **update employees set commission_pct = commission_pct * 1.1;**
107 rows updated.
SQL>
Checking V$TRANSACTION, you see HR's uncommitted transaction:
SQL> **select t.status, t.start_time, t.name**
2 **from v$transaction t join v$session s on t.ses_addr = s.saddr**
3 **where s.username = 'HR';**
STATUS START_TIME NAME
---------- ---------- ----------
ACTIVE 08/05/07 17:41:50 Employee Maintenance
1 row selected.
You change the undo tablespace as follows:
SQL> **alter system set undo_tablespace=undo_batch;**
System altered.
HR's transaction is still active, and therefore the old undo tablespace still contains the undo information for HR's transaction, leaving the undo segment still available with the following status until the transaction is committed or rolled back:
SQL> **select r.status**
2 **from v$rollstat r join v$transaction t on r.usn=t.xidusn**
3 **join v$session s on t.ses_addr = s.saddr**
4 **where s.username = 'HR';**
STATUS

PENDING OFFLINE
1 row selected.

Even though the current undo tablespace is UNDO_BATCH, the daytime tablespace UNDOTBS1 cannot be taken offline or dropped until HR's transaction is committed or rolled back:

```
SQL> show parameter undo_tablespace
NAME TYPE VALUE
------------------------ ----------- ---------------------
undo_tablespace string UNDO_BATCH
SQL> alter tablespace undotbs1 offline;
alter tablespace undotbs1 offline
*
ERROR at line 1:
ORA-30042: Cannot offline the undo tablespace
```

The error message ORA-30042 applies if you try to offline an undo tablespace that is in use— either it is the current undo tablespace or it still has pending transactions. Note that if we switch back to the daytime tablespace before HR commits or rolls back the original transaction, the status of HR's rollback segment reverts back to ONLINE:

```
SQL> alter system set undo_tablespace=undotbs1;
System altered.
SQL> select r.status
2 from v$rollstat r join v$transaction t on r.usn=t.xidusn
3 join v$session s on t.ses_addr = s.saddr
4 where s.username = 'HR';
STATUS
---------------
ONLINE
1 row selected.
```

Flashback Features

Flashback Query,
Flashback Table,
Flashback Version Query, and
Flashback Transaction Query.

Flashback Database uses Flashback logs in the Flash Recovery Area instead of undo in an undo tablespace to provide the Flashback functionality; Flashback Drop places dropped tables into a virtual recycle bin within the tablespace and they remain there until the user retrieves it with **flashback table . . . to before drop** command or empties the recycle bin, or else until the space is needed by new permanent objects in the tablespace.

To further extend the self-service capabilities of Oracle10*g* and Oracle 11*g*, the DBA can grant system and object privileges to users to allow them to fix their own problems, usually without any DBA intervention. In the following example, we're enabling the user SCOTT to perform Flashback operations on specific tables and to access transaction metadata across the database:

```
SQL> grant insert, update, delete, select on hr.employees to scott;
Grant succeeded.
SQL> grant insert, update, delete, select on hr.departments to scott;
Grant succeeded.
SQL> grant flashback on hr.employees to scott;
Grant succeeded.
SQL> grant flashback on hr.departments to scott;
Grant succeeded.
SQL> grant select any transaction to scott;
Grant succeeded.
```

Flashback Query

Starting with Oracle9*i* Release 2, the **as of** clause is available in a **select** query to retrieve the state of a table as of a given timestamp or SCN. You might use this to find out which rows in a table were deleted since midnight, or you might want to just do a comparison of the rows in a table today versus what was in the table yesterday.

In the following example, HR is cleaning up the EMPLOYEES table and deletes two employees who no longer work for the company:

```
SQL> delete from employees
2 where employee_id in (195,196);
2 rows deleted.
SQL> commit;
Commit complete.
SQL>
```

Normally, HR will copy these rows to the EMPLOYEES_ARCHIVE table first, but she forgot to do that this time; HR doesn't need to put those rows back into the EMPLOYEES table, but she needs to get the two deleted rows and put them into the archive table. Because HR knows she deleted the rows less than an hour ago, we can use a relative timestamp value with Flashback Query to retrieve the rows:

```
SQL> insert into hr.employees_archive
2 select * from hr.employees
3 as of timestamp systimestamp - interval '60' minute
4 where hr.employees.employee_id not in
5 (select employee_id from hr.employees);
2 rows created.
SQL> commit;
Commit complete.
```

Because we know that EMPLOYEE_ID is the primary key of the table, we can use it to retrieve the employee records that existed an hour ago but do not exist now. Note also that we didn't have to know which records were deleted; we essentially compared the table as it existed now versus an hour ago and inserted the records that no longer exist into the archive table.

Although we could use Flashback Table to get the entire table back, and then archive and delete the affected rows, in this case it is much simpler to merely retrieve the deleted rows and insert them directly into the archive table.

Another variation of Flashback Table is to use Create Table As Select (CTAS) with the subquery being a Flashback Query:

```
SQL> delete from employees where employee_id in (195,196);
2 rows deleted.
SQL> commit;
Commit complete.
SQL> create table employees_deleted as
2 select * from employees
3 as of timestamp systimestamp - interval '60' minute
4 where employees.employee_id not in
5 (select employee_id from employees);
Table created.
SQL> select employee_id, last_name from employees_deleted;
EMPLOYEE_ID LAST_NAME
----------- -------------------------
195 Jones
```

196 Walsh
2 rows selected.

This is known as an *out-of-place restore* (in other words, restoring the table or a subset of the table to a different location than the original). This has the advantage of being able to further manipulate the missing rows, if necessary, before placing them back in the table; for example, after reviewing the out-of-place restore, an existing referential integrity constraint may require that you insert a row into a parent table before the restored row can be placed back in the child table.

One of the disadvantages of an out-of-place restore using CTAS is that neither constraints nor indexes are rebuilt automatically.

Possible Questions
(Each question carries 6 Marks)

1. Explain extents and blocks with suitable example.
2. Discuss reusable space allocation with suitable example
3. Discuss segment advisor with suitable example.
4. Explain the automatic work load repository with suitable example.
5. Write notes on i)Multiplexing Redo log files ii)Controlling Archive Log.
6. Explain how to manage tablespace alert with suitable example.
7. Explain the components of Data Blocks with suitable example.
8. Discuss the composite indexes.
9. Discuss Managing Undo Tablespaces.
10. Describe Flashback features.

KARPAGAM ACADEMY OF HIGHER EDUCATION
COIMBATORE - 21
DEPARTMENT: CS,CA & IT
CLASS : II MCA
BATCH : 2018-2020

Part -A Online Examinations **(1 mark questions)**
SUBJECT: Database Administration **SUBJECT CODE: 17CAP404D**

UNIT III

S.no	Questions	opt1	opt 2	opt 3	opt 4	Answer
1	_____ is the process of placing the data in the database tables.	loading	extraction	transformation	insertion	loading
2	_____ provides a way to merge the laoding and transformation processes	multitable inserts	external tables	upserts	table functions	external tables
3	_____ produce a set of rows as output.	multitable inserts	external tables	upserts	table functions	table functions
4	The term _____ implies that a given table structure is mapped to a data file that's located in an operating system file	multitable inserts	external tables	upserts	table functions	external tables
5	Using _____ driver, you can only load data into a table from an external table.	ORACLE_LOADER	ORACLE_DATAPUMP	ORACLE_PUMP	ORACLE_OBJECT	ORACLE_LOADER
6	The _____ access driver lets you both load and unload data using external dump files	ORACLE_LOADER	ORACLE_DATAPUMP	ORACLE_PUMP	ORACLE_OBJECT	ORACLE_DATAPUMP
7	_____ data means reading data from an external table and loading it into a regualr oracle table.	loading	unloading	copying	inserting	loading
8	Using _____ statement , it can perform highly expressive computations using sets of interrelated formulas	MODEL	EXTRACT	SQL	DML	MODEL
9	_____ refers to the direct transmission of results from one process to the other without any intermediate steps.	parallel execution	pipelining	streaming	loading	streaming
10	You can use the DBMS_ADVISOR package to create an ADDM report by using the ___ procedure.	create_report	create_task	create_work	create_load	create_report
11	Data Pump Export which is invoked with the_____ command	impdp	expdp	exp	export	expdp
12	Data Pump Import which is invoked with the____ command	impdp	import	impdp	import dump	impdp
13	The_____ technology enables very high transfer of data from one database to another	data pump	data load	loader	directory object	data pump
14	Data pump is a_____ technology	client side	server side	complier	interface	server side
15	_____ the main engine for driving data dictionary metadata loading and unloading	dbms_pump	dbms_datapump -	dbms_data	dbms_load	dbms_datapump -
16	_____used to extract the appropriate metadata	dbms_pump	dbms_datapump -	dbms_metadata	dbms_load	dbms_metadata
17	_____lets oracle read data from and write data to operating system files that lie outside the database	direct path	external path	external loader	internal path	external path
18	_____ holds the data and metadata	dump files	sql files	log files	redo files	dump files
19	_____contain the DDL statements describing the objects included in the job but can contain data	dump files	sql files	log files	redo files	sql files
20	A _____maps a name to a directory path on the file system.	index	command	directory object	execute	directory object
21	The _____ Process , creates jobs and monitors them.	Master Control	slave	shadow	client	Master Control
22	The _____ process updates the master table with the various job status	Master Control	slave	shadow	work	work
23	The _____processes are the expdp and impdp commands	Master Control	slave	client	work	client
24	interactive datapump export can be performed by using _____ command	ATTACH	ALTER	ALIGN	ARRANGE	ATTACH
25	In _____ modes datapump exports the entire database in one export session	schema modes	Full export modes	table mode	tablespace modes	Full export modes
26	mode is used only for a single user's data or objects only	schema modes	Full export modes	table mode	tablespace modes	schema modes

27	_____ parameter lets you specify more than a single active executive thread for your export job	parallel	orderby	sortby	linear	parallel
28	Use _____parameters to list the objects that you wish to import	EXCLUDE	IMPORT	INCLUDE	INTER	INCLUDE
29	The_____ view shows summary information of all currently running datapump jobs.	DBA_DATAPUMP	DBA_DATAPUMP_JOBS	DBA_DATAPUMP_FILES	DBA_DATAPUMP_TABLES	
30	_____ view identifies the user sessions currently attached to a Data Pump Export or import job	DATAPUMP_SESSIO	DBA_SESSIONS	DBA_ACTIVE_Sessi	DBA_DATAPUMP_SESSIO	DBA_DATAPUMP_SESSIONS
31	_____ facilitates spot analysis of both foreground and background sessions	ASH	AWR	ARR	ARW	ASH
32	The _____ package lets you transfer operating system files directly through the database	DBMS_TRANSFER	DBMS_FILE_TRNSFER	DBMS_FULL_FILE	DBMS_TRANSFER_FILE	DBMS_FILE_TRNSFER
33	_____ are the accumulated total value of particular statistics since the start of an oracle isntance	sample data	baseline data	cumulative statistics	database metrics	cumulative statistics
34	The ASH automatically collects the _____	session sample data	baseline data	cumulative statistics	database metrics	session sample data
35	The statistics from the period when the database performed well are called _____	session sample data	baseline data	cumulative statistics	database metrics	baseline data
36	_____ measures the rate of change in a cumulative performance statistics	session sample data	baseline data	cumulative statistics	database metrics	database metrics
37	_____ procedure is used to define threshold settings for a database metric	SET_THRESHOLG	GET_THRESHOLD	GIVE_THRESHOLD	PASS_THRESHOLD	SET_THRESHOLG
38	The _____ package lets you manage alert notifications	DBMS_QUERY	DBMS_AQADM	DBMS_ASH	DBMS_ALERT	DBMS_AQADM
39	_____ shows the mapping of metric names to metric IDs	V$ALERT_TYPES	V$METRICNAME	V$DATABASE	V$DBNAME	V$METRICNAME
40	The _____ view provides information about all system alert types.	V$ALERT_TYPES	V$METRICNAME	V$DATABASE	V$DBNAME	V$ALERT_TYPES
41	The _____ automatically collects and stores database performance statistics	ASH	AWR	ATH	SGA	AWR
42	Expand MMNL	manageability monitor	manageability monitor link	manageability	manageability monitor	manageability monitor light
43	The _____ view is where the database stores a sample of all active session data	V$ALERT_TYPES	V$METRICNAME	V$ACTIVE_SESSION_HISTORY	V$DBNAME	V$ACTIVE_SESSION_HISTORY
44	_____ script is used to produce a ASH report	ashrpt.sql	ashrrt.sql	ashreport.sql	ashret.sql	ashrpt.sql
45	The _____ ensures that the external data processing matches the description of the external table.	access driver	directory objects	tables	functional mode	access driver
46	the older export/import technology was ____ .	client-based	server-based	DBMS_STATS	terminate	client-based
47	___ is / are the components of data pump technology. .	dbms_datapump package	dbms_metadata package	cluster	terminate	terminate
48	Data pump operations uses _____ files.	dump files	log	abort immediate	drop diskgroup	drop diskgroup
49	Data pump job creates all its dump files on the _____ .	server machine	client machine	abort immediate	remote machine	server machine
50	__ ___ command causes all the open Oracle connections to terminate automatically.	abort immediate	terminate	shutdown abort	close imemdiate	shutdown abort

KARPAGAM ACADEMY OF HIGHER EDUCATION
COIMBATORE - 21
DEPARTMENT: CS,CA & IT
CLASS : II MCA
BATCH : 2018-2020

Part -A Online Examinations **(1 mark questions)**
SUBJECT: Database Administration **SUBJECT CODE: 17CAP404D**

UNIT IV

S.no	Questions	opt1	opt 2	opt 3	opt 4	Answer
1	The _____ clause gives the user a 500MB space allocation in the users tablespace	GRANT QUOTA	GRANT SPACE	GRANT USER	GRANT MEMORY	GRANT QUOTA
2	The _____ statement is used to alter a user in the database.	ALTER SYSTEM	ALTER SET	ALTER USER	ALTER USER SPACE	ALTER USER
3	To drop a user, use the _____ statement	DROP	DROP USER	DROP SYSTEM	DELETE USER	DROP USER
4	A _____ is a collection of resource usage and password related attributes that you can assign to a user	roles	profiles	user	groups	profiles
5	_____ parameters are purely concerned with limiting resource usage.	resource parameters	password parameters	system parameters	memory parameters	resource parameters
6	_____ specifies the toal time a session may remain connected to the database	CPU_TIME	LOGICAL_TIME	CONNECT_TIME	WHOLE_TIME	CONNECT_TIME
7	A _____ is used to group together similar users based on their resource needs.	group user	resource plan	resource consumer group	resource cluster	resource consumer group
8	The _____ lays out how resource consumer groups are allocated resouces	resource group	resource parameters	resource plan	resource cluster	resource plan
9	The _____ privilege has to been granted to enable database resource manager	DBMS_RESOURCE	ADMINISTER_RESOURCE_MA	ADMINISTER_MANAGER	DBMS_MANAGER	ADMINISTER_RESOURCE_MANAGER
10	_____ is used to validate the changes before their implementation	submitting area	validate area	pending area	query area	pending area
11	_____ is used to assign resources to the various resource consumer group	resource plan	resource group	resource plan directives	resource plan	resource plan directives
12	The _____ view shows all currently active resource plan.	V$SESSION	V$RSRC_PLAN	V$CURRENT	V$ACTIVE	V$RSRC_PLAN
13	A_____ is the right to execute a particular type of SQL statement or to access a database object owned by anote	roles	privilege	grant	accept	privilege
14	The SYSDBA system privilege includes the _____ privilege and has all system privileges with ADMIN OPTION	VALIDATED SESSION	ENQUIRY SESSION	QUERYREWRITE	RESTRICTED SESSION	RESTRICTED SESSION
15	_____ are privileges on the various types of database objects.	system privileges	object privileges	query privileges	sysdba privileges	object privileges
16	ALTER and SELECT comes under _____ privileges	view privileges	system privileges	object privileges	sequence privileges	sequence privileges
17	_____ is based on statement, privilege and object level auditing	fine grained auditing	standard auditing	user auditing	database auditing	standard auditing
18	A _____ specifies the auditing of all actions on any type of object	statement level audit	system level audit	object level audit	privilege level audit	statement level audit
19	_____ involves the copying of database files	physical backups	logical backups	server backups	client backups	physical backups
20	A _____ needs a recovery	physical backups	logical backups	inconsistent backups	consistent backups	inconsistent backups
21	An _____ backup is a backup in which the files contain data from different points in time	physical backups	logical backups	inconsistent backups	consistent backups	inconsistent backups
22	_____ are backups made using the datapump export utility	physical backups	logical backups	inconsistent backups	consistent backups	logical backups
23	_____ can create and maage backups on disk and on tape devices.	RMAN	ASH	AWR	ACC	RMAN
24	_____ script is kept in the RMAN recovery catalog	text	stored	valid	table	stored
25	_____ script is kept in regular text files	text	stored	valid	table	text
26	The _____option lets you to specify how many copies of the backups you want to retain.	COPY	REDUNDANCY	RETENTION WINDOW	BACKUP COPY	REDUNDANCY
27	The _____ option ensures that RMAN doesn't perform a file backup if it has already backed up identical versions o	BACKUP OPTIMIZATIO	BACKUP RETENTION	BACKUP PARALLELISM	BACKUP COPY	BACKUP OPTIMIZATION
28	The _____ view displays all the online redo logs for the database	V$LOG	V$BACKUP	V$REDO	V$ACTIVE	V$LOG
29	_____ corruption occurs when you have inconsistent data in tables or indexes	repair	data block	block	files	data block
30	_____ tool if used from the operating system to check the structural integrity of the database files for corruption	DBVERIFY	DBATTACH	DBANALYSE	DBREPAIR	DBVERIFY
31	You can use the_____ command to validate backup sets before you use them from a recovery.	VALIDATE BACKUP	VALIDATE BACKUPSET	VALIDATE VOLUME	VALIDATE CHECKUP	VALIDATE BACKUPSET
32	_____are the means by which RMAN conducts its backup and recovery operations	Channels	volumes	triggers	functions	Channels
33	Thedefault value of degree of parallelism is _____	2	4	5	1	1
34	The _____ command will remove the recovery catalog:	REMOVE CATALOG	DELETE CATALOG	DROP CATALOG	ESC CATALOG	DROP CATALOG
35	The _____ option tells the RMAN to finish the backup as fast as it can.	PARTIAL	MINIMIZE TIME	MINIMIZE LOAD	DURATION	MINIMIZE TIME
36	The _____file, is used to track the physical location of all database block changes.	modify tracking	track files	track database	change-tracking	change-tracking
37	_____contains the Oracle databases that are backed up by Oracle Backup.	Media server	Client host server	Administrative server	client server	Client host server

38	_____is the frequent writing of the dirty database buffers in the cache to disk by the database writer	Fast checking	Fast-Start Checkpointing	start checkpointing	fast start recovery	Fast-Start Checkpointing
39	Bringing the data files up to date using backed-up data files and archived redo log files is called _____	restoring	recovery.	backing	blocking	recovery.
40	The process of applying the contents of both the archived and redo log files to bring the data files up to date is called _____	transaction recovery	open recovery	cache recovery	closed recovery	cache recovery
41	A _____ is a recovery for which you need to shut down the database completely.	transaction recovery	open recovery	cache recovery	closed recovery	closed recovery
42	when you use the _____command, RMAN restores a data file from an image backup	RESTORE	RECOVER	BACKING	BLOCKING	RESTORE
43	if your redo logs are lost or damaged, you need to specify _____ in the CREATE CONTROLFILE statement.	RECOVERLOGS	BACKUPLOGS	RESETLOGS	RESTORELOGS	RESETLOGS
44	_____allows you to view old row data based on a point in time or an SCN	Flashback Versions Query	Flashback Query	Flashback Transaction Query	Flashback Drop	Flashback Query
45	A user can permanently remove the objects from the Recycle Bin using the _____ command	DROP	RESTORE	DELETE	PURGE	PURGE
46	You can use ____ package to manage alerts	dbms_alert	alert_manage	dbms_server	alert_server	dbms_server
47	_____ tablespaces are used to store objects for the duration of a users session only.	Temporary tablespaces	Undo tablespaces	Databases	Tablespaces	Temporary tablespaces
48	SQL Access Advisor to help determine _____	Cursor view	Physical View	Materialized view	Updatable view	Materialized view

UNIT-V

SYLLABUS

Creating Table spaces in a VLDB Environment: Bigfile Table space Basics – Creating and Modifying Bigfile Tablespace – Bigfile Tablespace ROWID format – DBMS_ROWID and Bigfile Tablespaces.- Advanced Oracle Table Types – Using Bitmap Indexes – Oracle Data Pump

Remote queries – Remote Data Manipulation: Two Phase Commit – Managing Distributed Data – Managing Distributed Transactions – Monitoring and Tuning Distributed Database

Creating Tablespaces in a VLDB Environment

Creating a Bigfile Tablespace

Bigfile tablespaces are tablespaces with a single large data file, whose size can range from 8 to 128 terabytes, depending on the database block size.

To create a bigfile tablespace, specify the BIGFILE keyword of the CREATE TABLESPACE statement (CREATE BIGFILE TABLESPACE ...). Oracle Database automatically creates a locally managed tablespace with automatic segment-spec management. You can, but need not, specify EXTENT MANAGEMENT LOCAL and SEGMENT SPACE MANAGEMENT AUTO in this statement. However, the database returns an error if you specify EXTENT MANAGEMENT DICTIONARY or SEGMENT SPACE MANAGEMENT MANUAL. The remaining syntax of the statement is the same as for the CREATE TABLESPACE statement, but you can only specify one datafile. For example:

CREATE BIGFILE TABLESPACE bigtbs

DATAFILE '/u02/oracle/data/bigtbs01.dbf' SIZE 50G

...

You can specify SIZE in kilobytes (K), megabytes (M), gigabytes (G), or terabytes (T).

If the default tablespace type was set to BIGFILE at database creation, you need not specify the keyword BIGFILE in the CREATE TABLESPACE statement. A bigfile tablespace is created by default. If the default tablespace type was set to BIGFILE at database creation, but you want to create a traditional (smallfile) tablespace, then specify a CREATE SMALLFILE TABLESPACE statement to override the default tablespace type for the tablespace that you are creating

Creating and Modifying Bigfile Tablespace

Two clauses of the ALTER TABLESPACE statement support datafile transparency when you are using bigfile tablespaces:

- RESIZE: The RESIZE clause lets you resize the single datafile in a bigfile tablespace to an absolute size, without referring to the datafile. For example:

- ALTER TABLESPACE bigtbs RESIZE 80G;

- AUTOEXTEND (used outside of the ADD DATAFILE clause):

With a bigfile tablespace, you can use the AUTOEXTEND clause outside of the ADD DATAFILE clause. For example:

ALTER TABLESPACE bigtbs AUTOEXTEND ON NEXT 20G;

An error is raised if you specify an ADD DATAFILE clause for a bigfile tablespace.

Identifying a Bigfile Tablespace

The following views contain a BIGFILE column that identifies a tablespace as a bigfile tablespace:

- DBA_TABLESPACES

- USER_TABLESPACES

- V$TABLESPACE

Managing the Sysaux Tablespace

The SYSAUX tablespace is always created at database creation. The SYSAUX tablespace serves as an auxiliary tablespace to the SYSTEM tablespace. Because it is the default tablespace for

many Oracle Database features and products that previously required their own tablespaces, it reduces the number of tablespaces required by the database and that you, as a DBA, must maintain. Other functionality or features that previously used the SYSTEM tablespace can now use the SYSAUX tablespace, thus reducing the load on the SYSTEM tablespace.

You can specify only datafile attributes for the SYSAUX tablespace, using the SYSAUX DATAFILE clause in the CREATE DATABASE statement. Mandatory attributes of the SYSAUX tablespace are set by Oracle Database and include:

- PERMANENT
- READ WRITE
- EXTENT MANAGMENT LOCAL
- SEGMENT SPACE MANAGMENT AUTO

You cannot alter these attributes with an ALTER TABLESPACE statement, and any attempt to do so will result in an error. You cannot drop or rename the SYSAUX tablespace.

Oracle Data Pump Remote queries

- To unload data, you use the ORACLE_DATAPUMP access driver. The data stream that is unloaded is in a proprietary format and contains all the column data for every row being unloaded.
- An unload operation also creates a metadata stream that describes the contents of the data stream. The information in the metadata stream is required for loading the data stream. Therefore, the metadata stream is written to the datafile and placed before the data stream.

Parallel Access with ORACLE_LOADER

- The ORACLE_LOADER access driver attempts to divide large datafiles into chunks that can be processed separately.
- The following file, record, and data characteristics make it impossible for a file to be processed in parallel:
- Sequential data sources (such as a tape drive or pipe)
- Data in any multibyte character set whose character boundaries cannot be determined starting at an arbitrary byte in the middle of a string

- This restriction does not apply to any datafile with a fixed number of bytes per record.
- Records with the VAR format
- Specifying a PARALLEL clause is of value only when large amounts of data are involved.

Parallel Access with ORACLE_DATAPUMP

- When you use the ORACLE_DATAPUMP access driver to unload data, the data is unloaded in parallel when the PARALLEL clause or parallel hint has been specified and when multiple locations have been specified for the external table.
- Each parallel process writes to its own file. Therefore, the LOCATION clause should specify as many files as there are degrees of parallelism

External Table Restrictions

- An external table does not describe any data that is stored in the database.
- An external table does not describe how data is stored in the external source. This is the function of the access parameters
- An external table cannot load data into a LONG column.
- When identifiers (for example, column or table names) are specified in the external table access parameters, certain values are considered to be reserved words by the access parameter parser. If a reserved word is used as an identifier, it must be enclosed in double quotation marks.

Remote Data Manipulation

In most cases, especially in data warehousing environments, the data you're loading needs to be transformed to make it more meaningful for analysis. Oracle10g helps to efficiently transform the data within the database itself.The techniques used are:

1. Derive the tables from existing tables – use join or aggregations
2. Use SQL to transform data – MERGE, table functions, multiple table inserts.

3. Use oracle database 10g's – MODEL perform highly expressive computations using sets of interrelated formulas)

Deriving the data from Existing tables

2 methods

- 1. creating newly – use CTAS
- 2. table exists already – use INSERT /*APPEND */ INTO … SELECT method

Using CTAS method means u can create a new table from an existing table.

Example showing the use of CTAS method

Example showing how to load data into an existing table from another table.

```
Sql> INSERT/*APPEND NOLOGGING PARALLEL*/
INTO sales_data
SELECT product_id, customer_id, TRUNC(sales_date),
Discount_rate, sales_quantity, sale_price
FROM sales_history;
```

NOLOGGING and PARALLEL makes the bulk run extremely fast.

You must use the following statement so any DML statements you issue can be considered for parallel execution:

```
Sql> ALTER SESSION ENABLE PARALLEL DML;
```

Session altered.

Using SQL to transform data

Common ways – MERGE statement, multitable inserts and table functions

Using MERGE statement

- It is a powerful means of transforming data because it provides the functionality of checking data to see if an update is indeed required for a given row.
- It is actually an UPDATE-ELSE-INSERT operation performed by a single SQL statement
- In the first pass, you update all rows(that have the matching) and in second pass, you insert all rows (that don't have the matching)

Eg:

```
Sql> UPDATE catalog c SET
(catalog_name, catalog_desc, catalog_category, catalog_price) =
SELECT (catalog_name, catalog_desc, catalog_category, catalog_price)
FROM catalog_data d
WHERE c.catalog_id=d.catalog_id;
```

Second, the insert

```
Sql> INSERT INTO  catalog c
SELECT * FROM catalog_data d
WHERE c.catalog_id NOT IN
(selectcatalog_id from catalog_data);
```

Multi Table INSERT in Oracle

Most INSERT statements are the single-table variety, but Oracle also supports a multiple-table INSERT statement. With a multitable insert, you can make a single pass through the source data and load the data into more than one table.

- [ALL | FIRST]

WHEN condition THEN insert_into_clause [values_clause]

[insert_into_clause [values_clause]]...

[WHEN condition THEN insert_into_clause [values_clause]

[insert_into_clause [values_clause]]...

]...

[ELSE insert_into_clause [values_clause]

[insert_into_clause [values_clause]]...

]

Simple multi-table insert

To begin, we will unconditionally INSERT ALL the source data into every target table. The source records and target tables are all of the same structure so we will omit the VALUES clause from each INSERT.

SQL> SELECT COUNT(*) FROM all_objects;

COUNT(*)

28981

1 row selected.

```
SQL> INSERT ALL
  2    INTO t1
  3    INTO t2
  4    INTO t3
  5    INTO t4
6  SELECT owner
7  ,object_type
8  ,object_name
9  ,object_id
10 ,     created
11  FROMall_objects;

115924 rows created.

SQL> SELECT COUNT(*) FROM t1;

COUNT(*)
----------
     28981

1 row selected.
```

Using table functions for data transformation

It produces a collection of transformed rows that can be queried just like a regular table's data. Table functions can take a set of rows as input and return a transformed set of rows.

Three features make table functions a powerful means of performing fast transformation of data sets

1. streaming – direct transmission of results from one process to the other without any intermediate steps.

2. parallel execution – concurrent execution of the functions on multiprocessor systems

3. pipelining – lets you see the results of a query iteratively, instead of waiting for the entire result set to be batched and returned.

Using Data Pump Export and Import

Introduction

Oracle Database 10g offers the new Data Pump technology, a server-side infrastructure for fast data movement between Oracle databases. The Data Pump technology enables DBAs to transfer large amounts of data and metadata at very high speeds compared with the older export/import technology. Data Pump manages multiple parallel streams of data to achieve maximum throughput. Data Pump is a superset of the original export and import utilities, offering several new capabilities. Data Pump lets you estimate job times, perform fine-grained object selection, monitor jobs effectively, and directly load a database from a remote instance via the network. Oracle Data Pump technology consists of two components: the Data Pump Export utility, to unload data objects from a database, and the Data Pump Import utility, to load data objects into a database.

Here's how you invoke the two utilities:

$ expdp username/password (various parameters here)

$ impdp username/password (various parameters here)

Benefits of Data Pump Technology

Advantages using data pump are

- ability to estimate jobs times
- ability to restart failed jobs
- perform fine-grained object selection
- monitor running jobs
- directly load a database from a remote instance via the network
- remapping capabilities
- improved performance using parallel executions

Data Pump Uses

You can use data pump for the following

- migrating databases
- copying databases
- transferring oracle databases between different operating systems
- backing up important tables before you change them
- moving database objects from one tablespace to another
- transporting tablespace's between databases
- reorganizing fragmented table data
- extracting the DDL for tables and other objects such as stored procedures and packages

Data Pump components

Data pump technology consists of three major components

- dbms_datapump - the main engine for driving data dictionary metadata loading and unloading
- dbms_metadata - used to extract the appropriate metadata
- command-line - expdp and impdp are the import/export equivalents

Data Access methods

Data pump has two methods for loading data, direct path or external table path you as a dba have no control with what data pump uses, normally simple structures such as heap tables without triggers will use direct path more complex tables will use the external path, oracle will always try and use the direct-path method.

Direct Path bypasses the database buffer cache and writes beyond the high water mark when finished adjusts the high water mark, No undo is generated and can switch off redo as well, minimal impact to users as does not use SGA. Must disable triggers on tables before use.

External Path Uses the database buffer cache acts as a SELECT statement into a dump file, during import reconstructs statements into INSERT statements, so whole process is like a normal SELECT/INSERT job. Both undo and redo are generated and uses a normal COMMIT just like a DML statement would.

In the following cases oracle will use the external path if any of the below are in use

- clustered tables
- active triggers in the table
- a single partition in a table with a global index
- referential integrity constraints
- domain indexes on LOB columns
- tables with fine-grained access control enabled in the insert mode
- tables with BFILE or opaque type columns

Data Pump files

You will use three types's of files when using data pump, all files will be created on the server.

- dump files - holds the data and metadata
- log files - the resulting output from the data pump command
- sql files - contain the DDL statements describing the objects included in the job but can contain data
- Master data pump tables - when using datapump it will create tables within the schema, this is used for controlling the datapump job, the table is removed when finished.

Data Pump privileges

In order to advance features of data pump you need exp_full_database and imp_full_database privileges.

How Data Pump works

The Master Control Process (MCP), has the process name DMnn, only one master job runs per job which controls the whole Data Pump job, it performs the following

- create jobs and controls them
- creates and manages the worker processes
- monitors the jobs and logs the process
- maintains the job state and restart information in the master table (create in the users schema running the job)
- manages the necessary files including the dump file set

The master process creates a master table which contains job details (state, restart info), this table is created in the users schema who is running the Data Pump job. Once the job has finished it dumps the table contents into the data pump file and deletes the table. When you import the data pump file it re-creates the table and reads it to verify the correct sequence in which the it should import the various database objects.

The worker process is named DWnn and is the process that actually performs the work, you can have a number of worker process running on the same job (parallelism). The work process updates the master table with the various job status.

The shadow process is created when the client logs in to the oracle server it services data pump API requests, it creates the job consisting of the master table and the master process.

The client processes are the expdp and impdp commands.

Performing Data Pump Exports and Imports

- The data pump export utility loads row data from database tables, as well as object metadata, into dump file sets in a proprietary format that only the dat pump import utility can read.
- Dump files usually refer to a single file such as the default export dump file exptdat.dmp

Data pump methods

1. Using the command line

eg: $ expdp system/manager DIRECTORY=dpump_dir1 DUMPFILE=expdat1.dmp

2. Using Parameter file

eg: SCHEMAS =HR

DIRECTORY=dpump_dir1

DUMPFILE=system1.dmp

SCHEMAS=hr

Once you create the parameter file, all you need to do in order to export the HR schema is invoke expdp with the PARFILE parameter

$ expdp PARFILE=myfile.txt

3. Using interactivedatapump export- can use interactive mode for one purpose : to change some export parameters midstream while the job is still running. 1st way to perform is (Ctrl + c, which interrupts the running job and displays export prompt, you can insert the command). 2nd way is (using ATTACH command and parameter)

Data pump export modes

- Full export modes – export the entire database in one export session
- Schema modes – used only for a single user's data or objects only
- Tablespace mode – can export all the tables in one or more tablespaces
- Table mode – can export one or more tables

Data pump export parameters

1. File and directory related parameters – DIRECTORY, DUMPFILE, DILESIZE, PARFILE, LOGFILE, NOLOGFILE, NOLOGFILE, COMPRESSION

2. Export mode related parameters – FULL, SCHEMAS, TABLES, TABLESPACES, TRANSPORT_TABLESPACES, TRANSPORT_FULL_CHECK

3. Export filtering parameters –

(i) CONTENT – ALL, DATA_ONLY, METADATA_ONLY

(ii) EXCLUDE & INCLUDE – can perform metadata filtering, helps to omit/include specific database object types from an export or import operation.

- EXCLUDE =object_type(:name_clause)
- INCLUDE=object_type[:name_clause)
- EXCLUDE=TABLE: ”LIKE ’EMP%’ ”

(iii) QUERY- it lets to selectively export table row data with the help of a SQL statement. SAMPLE parameter can alos be used. (lets you to specify a percentage value ranging from 0.000001 to 100)

Syntax:

SAMPLE =[[schema_name.]table_name:]sample_percent

Eg: SAMPLE=“HR”.”EMPLOYEE”:50

4. Estimation Parameters- estimates how much physical space your export job will consume:

(i) ESTIMATE – tells you how much space ur new export job is going to consume. It is estimated in terms of bytes.

Syntax:

ESTIMATE ={BLOCKS | STATISTICS}

(ii) ESTIAMTE_ONLY – the estimate parameter is operative only during an actual export job, you can use the ESTIMATE_ONLY parameter without starting an export job.

5.. The Network Link parameter – provides a way to initiate a network export. You can initiate an export job from your server and have data pump export data from a remote database to dump files located on the instance from which you initiate the Data pump export job - ENCRYPTION parameter.

6. Job related parameters

JOB_NAME,STATUS,FLASHBACK_SCN,FLASHBACK_TIME,

PARALLEL,ATTACH.

JOB_NAME is used to give an explicit name to the export job.

STATUS is useful while you are running the long jobs

FLASHBACK_SCN specifies the SCN that Datapump export wil use to enable the flashback utility

FLASHBACK_TIME –gives the time limit

PARALLEL – lets you specify more than a single active executive thread for your export job

ATTACH – attaches your data pump client session to a running job and palces you in an interactive mode.

1. File and directory related parameters

- it uses the PARFILE, DIRECTORY , DUMPFILE, LOGFILE AND NOLOGFILE and SQLFILE. The SQLFILE parameter is similar to the old import INDEXFILE parameter. When you perform a DATA pump Import job, you may sometimes wish to extract the DDL from the export dump file. The SQLFILE parameter enables you to do this easily.

2. Filtering parameters

- use CONTENT parameter

- CONTENT=DATA_ONLY

-CONTENT=ALL

-CONTENT=METADATA_ONLY

Use INCLUDE parameters to list the objects that you wish to import.Use the EXCLUDE parameters to list the objects you don't want to import

3. Job related parameters – JOB_NAME, STATUS PARALLEL

4. Import mode related parameters- can perform datapump import in various modes using TABLE, SCHEMAS, TABLESPACES, FULL, TRANSPORT_FULL_CHECK. TRANSPORT_DATAFILES import parameter is used during a transportable tablespaces operation to specify the list of data fiels the job should import into the target databases.

5. Remapping parameters – it remaps the objects during the data import process. The parameters are REMAP_SCHEMA, REMAP_DATAFILE and REMAP_TABLESPACE.

REMAP_SCHEMA – you can move objects from one schema to another.

$impdp system/manager DUMPFILE=newdump.dmp REMAP_SCHEMA=hr:oe

REMAP_DATAFILE – helps to change the filenames, when the databases are moved from one platform to another platform.

REMAP_TABLESPACE – enables you to move objects from one tablespace into a different tablespace into different tablespace during an import statement.

REMAP_DATAFILE – when you are moving databses between two different paltforms, each with a separate file-nameing convention, the REMAP_DATFILE parameter comes

6. Transform parameter- lets you specify that yourdatapup import job should not import certain stroage and other attributes. Using this, you can exclude the STORAGE and TABLESPACE clauses. Syntax:

TRANSFORM = transform_name:value[:object_type]

7. Network Link Parameters – enables the Datapump import utility to connect directly to the source database and transfer data to the target database.

Eg: 1. create a database link in the remote database

sql>create database link remote connect to system identified by sammy1 using ‘remote.world’;

2. If there isn’t one already create a datapump directory object

Sql> create directory remote_dir1 AS ‘/u01/app/oracle/dp_dir’;

3. set the new directory as your default directory, by exporting the directory value:

$export DATA_PUMP_DIR=remote=dir1

4. perform the network import from the database named remote, using the following data pump import command

[local]$impdp system/sammy1 SCHEMAS =scott NETWORK_LINK=remote.

8. Flashback parameters – enables you to import data at specified time. The FLASHBACK_SCN parameter is similar to the FLASHBACK_TIME parameter, except that you directly specify the SCN.

;

Transportable Tablespaces

Oracle's transportable tablespaces feature offers you an easy way to move large amounts of data between databases efficiently by simply moving data files from one database to the other. Transporting tablespaces involves copying all the data files belonging to the source database to the target database and importing the data dictionary information about the tablespaces from the source database to the target database.

Uses for Transportable Tablespaces

The following are some of the important uses of the transportable tablespaces feature:

• Moving data from a source database (usually OLTP) to a data warehouse

• Moving data from a staging database into a data warehouse

• Moving data from a data warehouse to a data mart

• Performing tablespace point-in-time recovery (PITR)

• Archiving historical data

Transporting a Tablespace

Transporting a tablespace between two databases involves the following main steps:

1. Select the tablespace to be transported (and make sure there are no dependencies with objects in other tablespaces).

2. Generate the transportable tablespace set.

3. Perform the tablespace import. This involves copying data files to the target server and importing related metadata into the target database.

Selecting the Tablespaces to Be Transported

The primary condition you must meet for transporting tablespaces is that the set of candidate tablespaces must be self-contained. For example, if the tables in the tablespaces have any

indexes, they should be contained in one of the tablespaces in the set you're transporting. One way to verify that your set of tablespaces meets the self-contained criteria is by using the DBMS_TTS package, as follows:

```
SQL> EXECUTE sys.dbms_tts.transport_set_check('sales01,sales02',true);

PL/SQL procedure successfully completed.
```

You must have the EXECUTE_CATALOG_ROLE role to execute the TRANSPORT_SET_CHECK procedure.

You can further confirm this by querying the transport_set_violation table, which table lists all the partially contained tables in a tablespace and any references between objects belonging to different tablespaces.

```
SQL> SELECT * FROM sys.transport_set_violation

no rows selected
```

Generating the Transportable Tablespace Set

Before you can transport your tablespaces to the target database, you must generate a transportable tablespace set. The transportable tablespace set consists of all the data files in the tablespaces plus the export dump file, which contains the structural data dictionary information about the tablespaces. You can then transport this new tablespace to a different database.

```
SQL> ALTER TABLESPACE sales01 READ ONLY;

Tablespace altered.

SQL> ALTER TABLESPACE sales02 READ ONLY;

Tablespace altered.
```

Exporting the Dictionary Information (Metadata) for the Tablespaces

The first step in creating the transportable tablespaces set is to export the metadata that describes the objects that are part of the tablespaces you want to export. Listing 7

shows the export of the metadata for the pair of tablespaces.

Server Generated Alerts

- Server generated alerts automatically alert you when a problem conditions occur.
- The database generates alerts based on the occurrence of specific events, or when certain database metrics exceed their threshold values.
- Oracle calls the threshold based alerts stateful alerts, and they can be set off at either a warning threshold or a critical threshold.

Examples for stateless alerts:

- 1. recovery area space usage exceeded
- 2. resumable session suspended
- Snapshot too old
- There are 3 situations when a database can send an alert
- 1. a metric crosses a critical threshold value
- A metric crosses a warning threshold values
- A nonthreshold type of alert occurs
- Default server generated alerts

The server generated default alerts could be either threshold based or problem alerts. Some of them are

1. Snapshot too old
2. Tablespace space usage

3. Resumable session suspended

4. Recovery session running out of free space

In addition to default alerts, u can choose to use other alerts, and u can also change the thresholds for the default alerts. When the database issues an alerts, you can see it in the database control alerts table, which is located at the bottom of the database control homepage. The alert data is, by default updated every 60 seconds. To get the details of an alert, click on the alert message in the message column of the alerts table.

Managing alerts

- best way to manage database alerts is to use the OEM database control.
- Oracle automatically sends an alert message to a persistent queue named ALERT_QUE and OEM reads this queue and sends out notification about the outstanding server alerts.
- It is very easy to set your own warning and critical thresholds for any database metric.
- To set alert thresholds go to database control home page and click the manage metrics link which will find under the related links groups.

For each metric on the edit thresholds page, you can set the following:

- 1. warning and critical threshold
- 2. Response action
- There are notification rules which enable you to control the conditions under which you want to receive a message from the OEM. (Preferences Link)

Using the DBMS_SERVER_ALERT package to manage alerts

- It provides an easy way to manage database alerts.
- It has two main procedures : GET_THRESHOLD
- SET_THRESHOLD-define threshold settings for a database metric.

Eg: SQL>DESC DBMS_SERVER_ALERT.SET_THRESHOLD

PROCEDURE dbms_server_alert.set_threshold

- DBMS_AQ & DBMS_AQADM packages can also be used to directly access and read alert messages.
- DBMS_AQADM packages lets you subscribe to the alert queue, set threshold and display alert modifications using various procedures.
- The DBMS_AQ packages lets you manage alert notifications.

Proactive tablespace alerts

- 10g tablespaces have built-in alerts that will notify you if there is free space drops below a set threshold.
- Two default thresholds are critical and warning
- The MMON background process monitors, the free space in each tablespace and sends out the alerts.
- Oracle will send alert with a warning when your tablespace is at 85% capacity and will send a critical alert when the tablespace is at 97% capacity.
- To view information on your thresholds, see the DBA_THRESHOLDS view.

Example:

1. create a small tablespace to use for testing oracle alert mechanism:

SQL>create tablespace test datafile ‘test01.dbf’ size 10M

Extend management local uniform size 3M;

Tablespace created

Data dictionary view related to metrics and alerts

- V$METRICNAME – show mapping of metric names to metric IDs
- V$ALERT_TYPES – dispalys information about server alert types
- DBA_HIST_SYSMETRIC_HISTORY – contains snapshots of V$SYSMETRIC_HISTORY
- DBS_ALERT_HISTORY – provides a history of alerts of alerts that are no longer outstanding that is, all alerts that you have already resolved
- DBA_THRESHOLDS – shows the names as well as the critical and warning values for all thresholds in the databases.

V$ALERT TYPES

- It provides information about all system alert types.

1. STATE – stateful or stateless alerts. Stateful alerts are those alerts that clear automatically when the alert threshold that prompted the alert is cleared. It first appears in the DBA_OUTSTANDING_ALERTS view and then goes to DBA_ALERT_HISTORY Whereas the stateless alert goes straight to DBA_ALERT_HISTORY.

2. SCOPE – classifies alerts into database-wide and instance –wide

3.GROUP_NAME – oracle aggregates the various database alerts into some common groups:space, performance and configuration.

The pair of snapshots associated with a baseline are retained until the baseline is explicitly deleted:

BEGIN

DBMS_WORKLOAD_REPOSITORY.drop_baseline (

```
baseline_name => 'batch baseline',

        cascade => FALSE); -- Deletes associated
snapshots if TRUE.

END; /
```

Baseline information can be queried from the DBA_HIST_BASELINE view.

Workload Repository Views

The following workload repository views are available:

- V$ACTIVE_SESSION_HISTORY - Displays the active session history (ASH) sampled every second.
- V$METRIC - Displays metric information.
- V$METRICNAME - Displays the metrics associated with each metric group.
- V$METRIC_HISTORY - Displays historical metrics.
- V$METRICGROUP - Displays all metrics groups
- DBA_HIST_ACTIVE_SESS_HISTORY - Displays the history contents of the active session history.
- DBA_HIST_BASELINE - Displays baseline information.
- DBA_HIST_DATABASE_INSTANCE - Displays database environment information.
- DBA_HIST_SNAPSHOT - Displays snapshot information.
- DBA_HIST_SQL_PLAN - Displays SQL execution plans.
- DBA_HIST_WR_CONTROL - Displays AWR settings.

Workload Repository Reports

Oracle provides a script named awrrpt.sql to generate summary reports about the statistics collected by the AWR facility. when you run the awrrpt.sql script, you'll need to make the following choices:

- Choose between an HTML or plain text report
- Specify the beginning and ending snap id's

The two reports give essential the same output but the awrrpti.sql allows you to select a single instance. The reports can be generated as follows:

@$ORACLE_HOME/rdbms/admin/awrrpt.sql @$ORACLE_HOME/rdbms/admin/awrrpti.sql

The AWR reports include voluminous information, including the following:

- Load profile
- Top five timed events
- Wait events and latch activity
- Time-model statistics
- Operating system statistics
- Operating system statistics
- SQL ordered by elapsed time
- Tablespace and file I/O statistics
- Buffer pool and PGA statistics and advisors

Active Session History

It is a new source of Oracle database performance data in 10g.

- An active session is one which is in a user call

– Parse

– Execute

– Fetch

– On the CPU

- Provides historical information about recently sampled "active" sessions

- ASH = V$SESSION_WAIT++ with History

– Note: In 10g V$SESSION_WAIT is integrated with V$SESSION

- It facilitates spot analysis of both foreground and background sessions

Why should you use ASH?

- Great for performance diagnostics

– Logs wait events along with SQL details and session-specific context in a circular buffer in memory

- Fixed session sampling algorithm uses < 0.1% of 1 CPU

– Can be modified by the use of an _ parameter

- Primary data provider for the Automatic Database Diagnostic Monitor (ADDM)

– ADDM supports proactive performance diagnostics within the Oracle Kernel

–

Components of ASH

- Memory buffers in the fixed areas
- New Oracle Background Process

– MMNL – MMON Lite

- V$ACTIVE_SESSION_HISTORY
- X$ASH
- DBA_HIST_ACTIVE_SESS_HISTORY

– Based on WRH$_ACTIVE_SESSION_HISTORY

ASH Architecture

ASH Details - General

- No installation or setup required
- Intended 30-min circular buffer in the SGA
- In memory ASH contains as much history as it can store.

– Circular buffer not cleared when written to disk

- ASH on Disk (1 of 10 in memory samples)
- Init.ora

– STATISTICS_LEVEL = TYPICAL (Default)

- Master Switch

– _ACTIVE_SESSION_HISTORY = TRUE (Default)

- 30-minute circular buffer in the SGA - GOAL

– May scale down to smaller duration on large systems

- Circular Buffer Sizing Formula:

Max(Min (# of CPUs * 2MB, 5% of SHARED_POOL_SIZE, 30MB), 1MB)

- If SHARED_POOL_SIZE is not explicitly set

– Formula changes to 2% of SGA target

- Assumptions for MAX Size - 30MB

– 100 active sessions

– Sampled at once per second (60 samples in 1 minute)

– Assume 17 minutes of non-stop collection

– Assume 300 bytes per sample

– Size = 100*60*17*300 bytes ~ 29.18MB

- Fudge Factor of 0.82 MB
- History flushed to Automatic Workload Repository (AWR) every 30 minutes
- Part of the AWR snapshot
- Database metrics
- Session Wait Information
- Sampling done every second
- Can support sub-second sampling
- _ash_sampling_interval = 1000 (milliseconds by default)
- Can dump to process trace (if required)
- Estimated 2500 CPU Instructions per active session per sample
- 400 active sessions on a 1 Ghz processor consumes < 1 millisecond
- The sampler (MMNL) does not take any latches
- It supports dirty reads
- Can write to the in-memory buffer without any issues

Working with the Undo and the MTTR Advisors

Using the Undo Advisor

You can get to the Undo Advisor by following these steps:

1. From the Database Control home page, click on the Advisor Central link.

2. On the Advisor Central page, click on the Undo Management link.

3. Click the Undo Advisor button at the top of the page.

Using the Undo Advisor, you can do the following:

• Set the low threshold value for undo retention

• Figure out the size of the undo tablespace size you'll need for a new undo retention setting

• Use different analysis time periods representing different levels of system activity to get recommendations, in the form of a graph, about the right undo tablespace size for varying undo retention length.

Two-Phase Commit Mechanism

Unlike a transaction on a local database, a distributed transaction involves altering data on multiple databases. Consequently, distributed transaction processing is more complicated, because the database must coordinate the committing or rolling back of the changes in a transaction as a self-contained unit. In other words, the entire transaction commits, or the entire transaction rolls back.

The database ensures the integrity of data in a distributed transaction using the two-phase commit mechanism. In the prepare phase, the initiating node in the transaction asks the other participating nodes to promise to commit or roll back the transaction. During the commit phase, the initiating node asks all participating nodes to commit the transaction. If this outcome is not possible, then all nodes are asked to roll back.

All participating nodes in a distributed transaction should perform the same action: they should either all commit or all perform a rollback of the transaction. The database automatically controls and monitors the commit or rollback of a distributed transaction and maintains the integrity of the global database (the collection of databases participating in the transaction) using the two-phase commit mechanism. This mechanism is completely transparent, requiring no programming on the part of the user or application developer.

The commit mechanism has the following distinct phases, which the database performs automatically whenever a user commits a distributed transaction:

Prepare Phase

The first phase in committing a distributed transaction is the prepare phase. In this phase, the database does not actually commit or roll back the transaction. Instead, all nodes referenced in a distributed transaction (except the commit point site, described in the "Commit Point Site") are told to prepare to commit. By preparing, a node:

• Records information in the redo logs so that it can subsequently either commit or roll back the transaction, regardless of intervening failures

- Places a distributed lock on modified tables, which prevents reads

When a node responds to the global coordinator that it is prepared to commit, the prepared node promises to either commit or roll back the transaction later, but does not make a unilateral decision on whether to commit or roll back the transaction. The promise means that if an instance failure occurs at this point, the node can use the redo records in the online log to recover the database back to the prepare phase.

Advanced Oracle Table Types

Creating Bitmap Indexes

Conventional wisdom holds that bitmap indexes are most appropriate for columns having low distinct values--such as GENDER, MARITAL_STATUS, and RELATION. This assumption is not completely accurate, however. In reality, a bitmap index is always advisable for systems in which data is not frequently updated by many concurrent systems. In fact, as I'll demonstrate here, a bitmap index on a column with 100-percent unique values (a column candidate for primary key) is as efficient as a B-tree index.

In this article I'll provide some examples, along with optimizer decisions, that are common for both types of indexes on a low-cardinality column as well as a high-cardinality one. These examples will help DBAs understand that the usage of bitmap indexes is not in fact cardinality dependent but rather application dependent.

Comparing the Indexes

There are several disadvantages to using a bitmap index on a unique column--one being the need for sufficient space (and Oracle does not recommend it). However, the size of the bitmap index depends on the cardinality of the column on which it is created as well as the data distribution. Consequently, a bitmap index on the GENDER column will be smaller than a B-tree index on the same column. In contrast, a bitmap index on EMPNO (a candidate for primary key) will be much larger than a B-tree index on this column. But because fewer users access decision-support

systems (DSS) systems than would access transaction-processing (OLTP) ones, resources are not a problem for these applications.

To illustrate this point, I created two tables, TEST_NORMAL and TEST_RANDOM. I inserted one million rows into the TEST_NORMAL table using a PL/SQL block, and then inserted these rows into the TEST_RANDOM table in random order:

```
Create table test_normal (empnonumber(10), ename varchar2(30), sal number(10));

Begin
For i in 1..1000000
Loop
  Insert into test_normal
values(i, dbms_random.string('U',30), dbms_random.value(1000,7000));
  If mod(i, 10000) = 0 then
  Commit;
 End if;
End loop;
End;
/
```

Managing Distributed Data

The database with the highest commit point strength determines which node commits first in a distributed transaction. When specifying a commit point strength for each node, ensure that the most critical server will be non-blocking if a failure occurs during a prepare or commit phase. The COMMIT_POINT_STRENGTH initialization parameter determines the commit point strength of a node.

The default value is operating system-dependent. The range of values is any integer from 0 to 255. For example, to set the commit point strength of a database to 200, include the following line in the database initialization parameter file:

COMMIT_POINT_STRENGTH = 200

The commit point strength is only used to determine the commit point site in a distributed transaction.

When setting the commit point strength for a database, note the following considerations:

- Because the commit point site stores information about the status of the transaction, the commit point site should not be a node that is frequently unreliable or unavailable in case other nodes need information about transaction status.

Using Global Names for Database

A global database name is formed from two components: a database name and a domain. The database name and the domain name are determined by the following initialization parameters at database creation:

The name that you give to a link on the local database depends on whether the remote database that you want to access enforces global naming. If the remote database enforces global naming, then you must use the remote database global database name as the name of the link. For example, if you are connected to the localhq server and want to create a link to the remote mfg database, and mfg enforces global naming, then you must use the mfg global database name as the link name.

The DB_DOMAIN initialization parameter is only important at database creation time when it is used, together with the DB_NAME parameter, to form the database global name. At this point, the database global name is stored in the data dictionary. You must change the global name using an ALTER DATABASE statement, not by altering the DB_DOMAIN parameter in the initialization parameter file. It is good practice, however, to change theDB_DOMAIN parameter to reflect the change in the domain name before the next database startup.

Monitoring and Tuning Distributed Database

The most effective way of optimizing distributed queries is to access the remote databases as little as possible and to retrieve only the required data.

For example, assume you reference five remote tables from two different remote databases in a distributed query and have a complex filter (for example, WHERE r1.salary + r2.salary > 50000). You can improve the performance of the query by rewriting the query to access the

remote databases once and to apply the filter at the remote site. This rewrite causes less data to be transferred to the query execution site.

Rewriting your query to access the remote database once is achieved by using collocated inline views. The following terms need to be defined:

- Collocated

Two or more tables located in the same database.

- Inline view

A SELECT statement that is substituted for a table in a parent SELECT statement. The embeddedSELECT statement, shown within the parentheses is an example of an inline view:

- Collocated inline view

An inline view that selects data from multiple tables from a single database only. It reduces the amount of times that the remote database is accessed, improving the performance of a distributed query.

Oracle recommends that you form your distributed query using collocated inline views to increase the performance of your distributed query. Oracle Database cost-based optimization can transparently rewrite many of your distributed queries to take advantage of the performance gains offered by collocated inline views.

Using Cost-Based Optimization

In addition to rewriting your queries with collocated inline views, the cost-based optimization method optimizes distributed queries according to the gathered statistics of the referenced tables and the computations performed by the optimizer.

For example, cost-based optimization analyzes the following query. The example assumes that table statistics are available. Note that it analyzes the query inside a CREATE TABLE statement:

```
CREATE TABLE AS (
        SELECT l.a, l.b, r1.c, r1.d, r1.e, r2.b, r2.c
        FROM local l, remote1 r1, remote2 r2
          WHERE l.c = r.c
```

Possible Questions

(Each question carries 6 Marks)

1. Explain big file table space with suitable example.
2. Discuss Oracle data pump with query example.
3. Explain how to use bitmaps and indexes with suitable example.
4. Write about Remote Queries with suitable example.
5. Explain Bigfile tablespace overview and benefits with suitable example.
6. Explain the accessing of remote database by using collocated inline views..
7. Explain Bigfile tablespace overview and benefits with suitable example.
8. Explain the accessing of remote database by using collocated inline views.
9. Explain the use of Bit Map indexes in Data Warehouse with suitable example.
10. Discuss distributed SQL statements with query example.
11. Explain how to perform transaction processing for distributed employee database in Oracle 11G with suitable example.

Reg. No.....

[14CAP405D]

KARPAGAM UNIVERSITY

Karpagam Academy of Higher Education
(Established Under Section 3 of UGC Act 1956)
COIMBATORE - 641 021
(For the candidates admitted from 2014 onwards)

MCA DEGREE EXAMINATION, APRIL 2016

Fourth Semester

COMPUTER APPLICATIONS

DATABASE ADMINISTRATION

Time: 3 hours Maximum: 60 marks

PART – A (20 x 1 = 20 Marks) (30 Minutes)
(Question Nos. 1 to 20 Online Examinations)

PART B (5 x 8 = 40 Marks) (2 ½ Hours)
Answer ALL the Questions

21 a. Explain the responsibilities of an Oracle DBA.

Or

b. Elucidate the structure of Oracle Physical storage.

22 a. Give an example to explain the concept of Oracle Segments, Extents and Blocks.

Or

b. Brief note on the various features of Flashback data archive.

23 a. Give an example to explain the Tuning of SQL statements.

Or

b. Implement Bulk Inserts and Bulk Deletes.

24 a. Brief note on Schema object auditing.

Or

b. How to control the Listener Server Process? Explain.

25 a. Give an example for Crating and Modifying Bigfile Tablspace.

Or

b. Brief about the Monitoring and Tuning of Distributed Database.

Reg. No.....

[12CAP405D]

KARPAGAM UNIVERSITY

(Under Section 3 of UGC Act 1956)
COIMBATORE - 641 021
(For the candidates admitted from 2012 onwards)

MCA DEGREE EXAMINATION, APRIL 2014

Fourth Semester

COMPUTER APPLICATIONS

DATABASE ADMINISTRATION

Time: 3 hours Maximum: 100 marks

PART – A (15 x 2 = 30 Marks)
Answer ALL the Questions

1. List down the tasks involved in monitoring the system.
2. Define OLTP.
3. Define DSS.
4. Briefly explain how AUTOALLOCATE option works.
5. Differentiate between regular oracle tables and index organized tables.
6. List down the main characteristics of temporary table space group.
7. List down the uses of data pump tools.
8. List down the capabilities of the SQL* loader.
9. Brief on loading and unloading in the context on the external table.
10. List down the steps involved in starting database resource manager.
11. Define role. List down the important roles of Oracle database.
12. What does RMAN offer during database recovery?
13. Write a short note on heuristic query processing strategies.
14. What are the joining methods used by CBO?
15. Give out the guidelines regarding joining tables wisely.

PART – B (5 X 14= 70 Marks)
Answer ALL the Questions

16. a. Discuss the roles of DBA?

Or

b. Elaborately explain the following.
   i. With the help of an example explain how oracle process transactions.
   ii. List down and mention the role of oracle data structures that ensures recovery of database after a problem.
   iii. Write short notes on various shutdown command option.

17. a. Explain how to create and manage table space.

Or

b. Write short note on the following.
   i. Transaction properties ii. Transaction concurrency control
   iii. Oracle's isolation level iv. Discrete and Autonomous transactions

18. a. Discuss the various instructions given by control file to SQL* loader.

Or

b. Explain the Data pump import parameters.

19. a. Give guidelines followed to perform backup.

Or

b. Write short notes on :
   i. Archive log and Non-archive log modes
   ii. Consistent and Inconsistent backups
   iii. Open and Closed backups
   iv. Physical and logical backups

20. a. Explain how to evaluate system performance?

Or

b. Explain how DBA can help to improve SQL processing.

Reg. No.....

[15CAP404D]

KARPAGAM UNIVERSITY
Karpagam Academy of Higher Education
(Established Under Section 3 of UGC Act 1956)
COIMBATORE - 641 021
(For the candidates admitted from 2015 onwards)

MCA DEGREE EXAMINATION, APRIL 2017
Fourth Semester

COMPUTER APPLICATIONS

DATABASE ADMINISTRATION

Time: 3 hours Maximum : 60 marks

PART – A (20 x 1 = 20 Marks) (30 Minutes)
(Question Nos. 1 to 20 Online Examinations)

PART B (5 x 6 = 30 Marks)
Answer ALL the Questions

21. a. Explain the Role of DBA with suitable diagram
Or
b. Discuss Oracle database instances with suitable example.

22. a. Explain extents and blocks with suitable example.
Or
b. Discuss reusable space allocation with suitable example.

23. a. Write in detail on tuning application with a neat Diagram
Or
b. Explain about database authentication methods with example.

24. a. Discuss Privilege Auditing Schema with example
Or
b. Explain the steps for integration of backup procedures with suitable example.

25. a. Explain big file table space with suitable example
Or
b. Discuss Oracle data pump with query example.

PART C (1 x 10 = 10 Marks)
CASE STUDY (Compulsory)

26. Explain how to monitor and tune distributed transaction of a student information database

---