EDUCATALYSTS

Introduction to Oracle Database

Introduction to the Oracle Database

This chapter provides an overview of the Oracle database server. The topics include:

- Oracle Database Architecture
- Oracle Database Features
- Oracle Database Application Development

> **Note:** This book contains information relating to both Oracle Database Standard Edition and Oracle Database Enterprise Edition. Some of the features and options documented in this chapter are available only if you have purchased the Oracle Database Enterprise Edition. See Oracle Database New Features for information about the differences between Oracle Database Standard Edition and Oracle Database Enterprise Edition.

Oracle Database Architecture

An Oracle database is a collection of data treated as a unit. The purpose of a database is to store and retrieve related information. A database server is the key to solving the problems of information management. In general, a server reliably manages a large amount of data in a multiuser environment so that many users can concurrently access the same data. All this is accomplished while delivering high performance. A database server also prevents unauthorized access and provides efficient solutions for failure recovery.

Oracle Database is the first database designed for enterprise grid computing, the most flexible and cost effective way to manage information and applications. Enterprise grid computing creates large pools of industry-standard, modular storage and servers. With this architecture, each new system can be rapidly

provisioned from the pool of components. There is no need for peak workloads, because capacity can be easily added or reallocated from the resource pools as needed.

The database has logical structures and physical structures. Because the physical and logical structures are separate, the physical storage of data can be managed without affecting the access to logical storage structures.

The section contains the following topics:

- Overview of Oracle Grid Architecture - XXX
- Overview of Application Architecture
- Overview of Physical Database Structures
- Overview of Logical Database Structures
- Overview of Schemas and Common Schema Objects
- Overview of the Oracle Data Dictionary
- Overview of the Oracle Instance
- Overview of Accessing the Database
- Overview of Oracle Utilities

Overview of Oracle Grid Architecture - XXX

The Oracle grid architecture pools large numbers of servers, storage, and networks into a flexible, on-demand computing resource for enterprise computing needs. The grid computing infrastructure continually analyzes demand for resources and adjusts supply accordingly.

For example, you could run different applications on a grid of several linked database servers. When reports are due at the end of the month, the database administrator could automatically provision more servers to that application to handle the increased demand.

Grid computing uses sophisticated workload management that makes it possible for applications to share resources across many servers. Data processing capacity can be added or removed on demand, and resources within a location can be dynamically provisioned. Web services can quickly integrate applications to create new business processes.

Grid computing offers high performance and scalability, because all computing resources can be flexibly allocated to applications as needed.

Oracle Database enables enterprise grid computing in the following ways:

- Performance and scalability with low cost hardware clusters, like Itanium and Linux.
- Reliability: Continuous availability of data and applications
- Security and privacy: security features that lets you share enterprise grid resources with confidence that privacy is maintained
- Self-management: Oracle infrastructure automates many functions so that a single administrator can manage hundreds of servers.
- Distributed computing: Oracle has advanced integration features that allow applications and data to run anywhere in the network.

Difference between a cluster and a grid: Clustering is one technology used to create a grid infrastructure. Simple clusters have static resources for specific applications by specific owners. Grids, which can consist of multiple clusters, are dynamic resource pools shareable among many different applications and users. A grid does not assume that all servers in the grid are running the same set of applications. Applications can be scheduled and migrated across servers in the grid. Grids share resources from and among independent system owners.

At the highest level, the idea of grid computing is computing as a utility. In other words, you should not care where your data resides, or what computer processes your request. You should be able to request information or computation and have it delivered - as much as you want, and whenever you want. This is analogous to the way electric utilities work, in that you don't know where the generator is, or how the electric grid is wired, you just ask for electricity, and you get it. The goal is to make computing a utility, a commodity, and ubiquitous. Hence the name, The Grid. This view of utility computing is, of course, a "client side" view.

From the "server side", or behind the scenes, the grid is about resource allocation, information sharing, and high availability. Resource allocation ensures that all those that need or request resources are getting what they need, that resources are not standing idle while requests are going unserviced. Information sharing makes sure that the information users and applications need is available where and when it is needed. High availability features guarantee all the data and computation is always there, just like a utility company always provides electric power.

Oracle provides substantial grid computing technology, available today, that can help you capitalize on the grid. Oracle is the best on commodity clusters, a hardware platform that many believe would come to dominate the grid. Oracle possesses the key technology differentiators -- Oracle Real Application Clusters, Oracle Streams, Oracle Transportable Tablespaces -- for building the grid. Only

Oracle delivers the operational characteristics -- portability, RAS, security, and scalability -- necessary for the grid.

Recently, ideas behind grid computing have evolved, due in large part to advances in hardware and networking technologies and the drop in costs associated with these technologies. New high-volume processors and inexpensive blade servers, for example, are extremely affordable compared to their high-end SMP predecessors. Instead of scavenging resources as in the older model of grid computing, a business could relatively cheaply build a blade server farm whose resources could be dynamically and automatically allocated to the areas of the enterprise that required the computing power.

In addition to operating on low-cost, off-the-shelf hardware that can be quickly assembled to create a relatively large-scale operation, this vision of grid computing allows for a high level of flexibility in meeting existing and future computing needs.

Oracle Technologies that Enable the Grid

Oracle has been working for years on technologies that support and enable grid computing.

Computing resource provisioning is one of the most important capabilities of a grid. This enables computing resources to be dynamically provisioned to applications as required. Resources must be appropriately allocated based on business priorities and demand. Oracle provides a number features for computing resource provisioning, including:

Real Application Clusters. RAC is a cluster database with a shared cache architecture that runs on multiple machines, attached through a cluster interconnect and a shared storage subsystem. An Oracle RAC database not only appears like a single standard Oracle Database to users, but the same maintenance tools and practices used for a single Oracle Database can be used on the entire cluster. All standard backup and recovery operations, including the use of Recovery Manager, work transparently with RAC. All SQL operations, including data definition language and integrity constraints, are also identical for both configurations. The most important part of RAC, however, is the ability to manage your workload—to add nodes or relinquish nodes on demand—based on your business processing needs.

Automatic Storage Management. Oracle recommends using Automatic Storage Management (ASM) for your database files and a cluster file system for the Oracle home. ASM simplifies the administration of Oracle database files. Instead of managing many database files, ASM requires you to manage only a small number

of disk groups. You can define a particular disk group as the default disk group for a database.

Oracle Resource Manager. Though Oracle Database is largely a self-managing database, Database Resource Manager allows resource administrators to influence how the Oracle database resources are allocated to users.

Oracle Scheduler. Oracle Scheduler provides many capabilities to schedule and perform business and IT tasks, called jobs, in a grid.

Information provisioning means delivering information to users whenever they need it, regardless of where it resides on the grid. To process information on any available resource, the grid must efficiently share information across distributed systems. The grid must also provide access to data residing on heterogeneous systems—database systems from multiple vendors and file systems. Oracle provides a broad set of features and tools for information provisioning on a grid, including the following:

Oracle Transportable Tablespaces. Transportable Tablespaces allows Oracle datafiles to be unplugged from a database, moved, or copied to another location, and then plugged into another database. Unplugging or plugging a datafile involves reading or loading only a small amount of metadata. Transportable Tablespaces also supports simultaneous mounting of read-only tablespaces by two or more databases.

Oracle Streams. Some data needs to be shared as it is created or changed, rather than occasionally shared in bulk. Oracle Streams can stream data between databases, nodes, or blade farms in a grid and can keep two or more copies in sync as updates are applied. It also provides a unified framework for information sharing, combining message queuing, replication, events, data warehouse loading, notifications, and publish/subscribe into a single technology.

A combination Streams/Transportable Tablespace feature enables a self-propelled database. With a single command, you can take a tablespace from one database, ship the tablespace to another database, reformat the tablespace if the second database is on a different operating system, plug this tablespace into the second database, and start syncing the tablespace with the changes happening in the first database. If the second database is on a grid, then you have just migrated your application to a grid with a single command.

Easy OCI/JDBC Install. With Oracle Database, any application using Oracle Call Interface (OCI) can easily install a small footprint version of the Oracle client files without installing and configuring the entire client. These applications include Java database connectivity (JDBC) type-2 driver applications. OCI and the JDBC type-2 driver (which requires OCI) can easily be installed by downloading a small subset

of the Oracle client files and updating a few environment variables to point to the location of the downloaded libraries. This means your grid client doesn't need to install any Oracle software and yet you get easy, secure access to data from an Oracle database running on a grid.

Distributed SQL and Distributed Transactions. Oracle Distributed SQL allows grid users to efficiently access and integrate data stored in multiple Oracle and non-Oracle databases. Transparent remote data access with Distributed SQL allows grid users to run their applications against any other database without making any code change to the applications. While integrating data and managing transactions across multiple data stores, the Oracle database intelligently optimizes the execution plans to access data in the most efficient manner.

Ultra Large Database Support. For enabling Ultra Large Databases (ULDB), the Bigfile Tablespace feature allows Oracle Database to contain tablespaces made up of single large files rather than numerous smaller ones. This allows Oracle Database to utilize the ability of 64-bit systems to create and manage ultralarge files. The consequence of this is that Oracle Database can now scale up to 8 exabytes in size.

Managing the Grid

Because some of the key goals of a grid are to provide high availability, scalability, and service performance optimization with minimal costs and complexity, an integrated approach to grid management is necessary. These tools and features include:

Grid Management with Oracle Enterprise Manager. Enterprise Manager provides a simplified, centralized management framework for managing enterprise resources and analyzing a grid's performance. With Enterprise Manager, administrators can manage the grid environment through a Web browser throughout the system lifecycle, front to back, from any location on the network. With Oracle Database, this includes integrated management and monitoring of RAC databases as well as standby Data Guard systems, for failover or switchover scenarios.

Managing Security in the Grid The dynamic environment in a grid makes security extremely important. Oracle makes managing security easy for you by centralizing security management for a distributed enterprise using the Lightweight Directory Access Protocol (LDAP)-compliant Oracle Internet Directory (OID). In this regard, Enterprise User Security provides the ability to create and manage privileges of a user globally—across all enterprise databases. The enterprise user privilege administration is done in the OID, thus avoiding the need to create the same user in multiple databases across a grid. Additionally, you now can store a Secure Sockets Layer (SSL) Certificate in a smart card, for roaming access to the grid.

Virtual Private Database. VPD provides server-enforced, fine-grained access control and a secure application context that can be used in a grid setting to enable multiple customers, partners, or departments utilizing the same database to have secure access to mission-critical data. VPD enables per-user and per-customer data access within a single database, with the assurance of physical data separation. VPD is enabled by associating one or more security policies with tables or views.

Oracle Label Security. Oracle Label Security gives administrators an out-of-the-box row-level and column-level security solution for controlling access to data based on its sensitivity, eliminating the need to manually write such policies. Using the GUI tool Oracle Policy Manager, administrators can quickly create and assign Oracle Label Security policies to rows and columns within application tables. Moreover, Oracle Database adds integration of Oracle Label Security with OID, allowing policies to be managed centrally within a dynamically changing grid setting.

Overview of Application Architecture

There are two common ways to architect a database: client/server or multitier. As internet computing becomes more prevalent in computing environments, many database management systems are moving to a multitier environment.

Client/Server Architecture

Multiprocessing uses more than one processor for a set of related jobs. Distributed processing reduces the load on a single processor by allowing different processors to concentrate on a subset of related tasks, thus improving the performance and capabilities of the system as a whole.

An Oracle database system can easily take advantage of distributed processing by using its client/server architecture. In this architecture, the database system is divided into two parts: a front-end or a client, and a back-end or a server.

The Client The client is a database application that initiates a request for an operation to be performed on the database server. It requests, processes, and presents data managed by the server. The client workstation can be optimized for its job. For example, it might not need large disk capacity, or it might benefit from graphic capabilities.

Often, the client runs on a different computer than the database server, generally on a PC. Many clients can simultaneously run against one server.

The Server The server runs Oracle software and handles the functions required for concurrent, shared data access. The server receives and processes the SQL and

PL/SQL statements that originate from client applications. The computer that manages the server can be optimized for its duties. For example, it can have large disk capacity and fast processors.

Multitier Architecture: Application Servers

A multitier architecture has the following components:

- A client or initiator process that starts an operation
- One or more application servers that perform parts of the operation. An application server provides access to the data for the client and performs some of the query processing, thus removing some of the load from the database server. It can serve as an interface between clients and multiple database servers, including providing an additional level of security.
- An end or database server that stores most of the data used in the operation

This architecture enables use of an application server to do the following:

- Validate the credentials of a client, such as a Web browser
- Connect to an Oracle database server
- Perform the requested operation on behalf of the client

If proxy authentication is being used, then the identity of the client is maintained throughout all tiers of the connection.

Overview of Physical Database Structures

The following sections explain the physical database structures of an Oracle database, including datafiles, redo log files, and control files.

Datafiles

Every Oracle database has one or more physical datafiles. The datafiles contain all the database data. The data of logical database structures, such as tables and indexes, is physically stored in the datafiles allocated for a database.

The characteristics of datafiles are:

- A datafile can be associated with only one database.
- Datafiles can have certain characteristics set to let them automatically extend when the database runs out of space.
- One or more datafiles form a logical unit of database storage called a tablespace.

Data in a datafile is read, as needed, during normal database operation and stored in the memory cache of Oracle. For example, assume that a user wants to access some data in a table of a database. If the requested information is not already in the memory cache for the database, then it is read from the appropriate datafiles and stored in memory.

Modified or new data is not necessarily written to a datafile immediately. To reduce the amount of disk access and to increase performance, data is pooled in memory and written to the appropriate datafiles all at once, as determined by the database writer process (DBWn) background process.

> **See Also:** "Overview of the Oracle Instance" on page 1-16 for more information about Oracle's memory and process structures

Control Files

Every Oracle database has a control file. A control file contains entries that specify the physical structure of the database. For example, it contains the following information:

- Database name
- Names and locations of datafiles and redo log files
- Time stamp of database creation

Oracle can multiplex the control file, that is, simultaneously maintain a number of identical control file copies, to protect against a failure involving the control file.

Every time an instance of an Oracle database is started, its control file identifies the database and redo log files that must be opened for database operation to proceed. If the physical makeup of the database is altered (for example, if a new datafile or redo log file is created), then the control file is automatically modified by Oracle to reflect the change. A control file is also used in database recovery.

> **See Also:** Chapter 3, "Tablespaces, Datafiles, and Control Files"

Redo Log Files

Every Oracle database has a set of two or more redo log files. The set of redo log files is collectively known as the redo log for the database. A redo log is made up of redo entries (also called redo records).

The primary function of the redo log is to record all changes made to data. If a failure prevents modified data from being permanently written to the datafiles, then the changes can be obtained from the redo log, so work is never lost.

To protect against a failure involving the redo log itself, Oracle allows a multiplexed redo log so that two or more copies of the redo log can be maintained on different disks.

The information in a redo log file is used only to recover the database from a system or media failure that prevents database data from being written to the datafiles. For example, if an unexpected power outage terminates database operation, then data in memory cannot be written to the datafiles, and the data is lost. However, lost data can be recovered when the database is opened, after power is restored. By applying the information in the most recent redo log files to the database datafiles, Oracle restores the database to the time at which the power failure occurred.

The process of applying the redo log during a recovery operation is called rolling forward.

> **See Also:** "Overview of Database Backup and Recovery Features" on page 1-28

Archive Log Files

You can enable automatic archiving of the redo log. Oracle automatically archives log files when the database is in `ARCHIVELOG` mode.

Parameter Files

Parameter files contain a list of configuration parameters for that instance and database.

Oracle recommends that you create a server parameter file (SPFILE) as a dynamic means of maintaining initialization parameters. A server parameter file lets you store and manage your initialization parameters persistently in a server-side disk file.

> **See Also:**
>
> - "Initialization Parameter Files and Server Parameter Files" on page 12-3
> - *Oracle Database Administrator's Guide* for information on creating and changing parameter files

Alert and Trace Log Files

Each server and background process can write to an associated trace file. When an internal error is detected by a process, it dumps information about the error to its trace file. Some of the information written to a trace file is intended for the database

administrator, while other information is for Oracle Support Services. Trace file information is also used to tune applications and instances.

The alert file, or alert log, is a special trace file. The alert file of a database is a chronological log of messages and errors.

See Also: *Oracle Database Administrator's Guide*

Backup Files

To restore a file is to replace it with a backup file. Typically, you restore a file when a media failure or user error has damaged or deleted the original file.

User-managed backup and recovery requires you to actually restore backup files before you can perform a trial recovery of the backups.

Server-managed backup and recovery manages the backup process, such as scheduling of backups, as well as the recovery process, such as applying the correct backup file when recovery is needed.

See Also:

- Chapter 15, "Backup and Recovery"
- *Oracle Database Backup and Recovery Advanced User's Guide*

Overview of Logical Database Structures

The logical storage structures, including data blocks, extents, and segments, enable Oracle to have fine-grained control of disk space use.

Tablespaces

A database is divided into logical storage units called tablespaces, which group related logical structures together. For example, tablespaces commonly group together all application objects to simplify some administrative operations.

Each database is logically divided into one or more tablespaces. One or more datafiles are explicitly created for each tablespace to physically store the data of all logical structures in a tablespace. The combined size of the datafiles in a tablespace is the total storage capacity of the tablespace.

Every Oracle database contains a `SYSTEM` tablespace and a `SYSAUX` tablespace. Oracle creates them automatically when the database is created. The system default is to create a smallfile tablespace, which is the traditional type of Oracle tablespace. The `SYSTEM` and `SYSAUX` tablespaces are created as smallfile tablespaces.

Oracle also lets you create bigfile tablespaces up to 8 exabytes (8 million terabytes) in size. With Oracle-managed files, bigfile tablespaces make datafiles completely transparent for users. In other words, you can perform operations on tablespaces, rather than the underlying datafiles.

See Also: "Overview of Tablespaces" on page 3-6

Online and Offline Tablespaces A tablespace can be online (accessible) or offline (not accessible). A tablespace is generally online, so that users can access the information in the tablespace. However, sometimes a tablespace is taken offline to make a portion of the database unavailable while allowing normal access to the remainder of the database. This makes many administrative tasks easier to perform.

Oracle Data Blocks

At the finest level of granularity, Oracle database data is stored in data blocks. One data block corresponds to a specific number of bytes of physical database space on disk. The standard block size is specified by the `DB_BLOCK_SIZE` initialization parameter. In addition, you can specify up to five other block sizes. A database uses and allocates free database space in Oracle data blocks.

Extents

The next level of logical database space is an extent. An extent is a specific number of contiguous data blocks, obtained in a single allocation, used to store a specific type of information.

Segments

Above extents, the level of logical database storage is a segment. A segment is a set of extents allocated for a certain logical structure. The following table describes the different types of segments.

Segment	Description
Data segment	Each nonclustered table has a data segment. All table data is stored in the extents of the data segment. For a partitioned table, each partition has a data segment. Each cluster has a data segment. The data of every table in the cluster is stored in the cluster's data segment.
Index segment	Each index has an index segment that stores all of its data. For a partitioned index, each partition has an index segment.

Segment	Description
Temporary segment	Temporary segments are created by Oracle when a SQL statement needs a temporary database area to complete execution. When the statement finishes execution, the extents in the temporary segment are returned to the system for future use.
Rollback segment	If you are operating in automatic undo management mode, then the database server manages undo space using tablespaces. Oracle recommends that you use automatic undo management. Earlier releases of Oracle used rollback segments to store undo information. The information in a rollback segment was used during database recovery for generating read-consistent database information and for rolling back uncommitted transactions for users. Space management for these rollback segments was complex, and Oracle has deprecated that method. This book discusses the undo tablespace method of managing undo; this eliminates the complexities of managing rollback segment space, and lets you exert control over how long undo is retained before being overwritten. Oracle does use a `SYSTEM` rollback segment for performing system transactions. There is only one `SYSTEM` rollback segment and it is created automatically at `CREATE DATABASE` time and is always brought online at instance startup. You are not required to perform any operations to manage the `SYSTEM` rollback segment.

Oracle dynamically allocates space when the existing extents of a segment become full. In other words, when the extents of a segment are full, Oracle allocates another extent for that segment. Because extents are allocated as needed, the extents of a segment may or may not be contiguous on disk.

See Also:

- Chapter 2, "Data Blocks, Extents, and Segments"
- Chapter 3, "Tablespaces, Datafiles, and Control Files"
- "Introduction to Automatic Undo Management" on page 2-19
- "Read Consistency" on page 1-23
- "Overview of Database Backup and Recovery Features" on page 1-28

Overview of Schemas and Common Schema Objects

A schema is a collection of database objects. A schema is owned by a database user and has the same name as that user. Schema objects are the logical structures that directly refer to the database's data. Schema objects include structures like tables, views, and indexes. (There is no relationship between a tablespace and a schema. Objects in the same schema can be in different tablespaces, and a tablespace can hold objects from different schemas.)

Some of the most common schema objects are defined in the following section.

Tables

Tables are the basic unit of data storage in an Oracle database. Database tables hold all user-accessible data. Each table has columns and rows. A table that has an employee database, for example, can have a column called employee number, and each row in that column is an employee's number.

Indexes

Indexes are optional structures associated with tables. Indexes can be created to increase the performance of data retrieval. Just as the index in this manual helps you quickly locate specific information, an Oracle index provides an access path to table data.

When processing a request, Oracle can use some or all of the available indexes to locate the requested rows efficiently. Indexes are useful when applications frequently query a table for a range of rows (for example, all employees with a salary greater than 1000 dollars) or a specific row.

Indexes are created on one or more columns of a table. After it is created, an index is automatically maintained and used by Oracle. Changes to table data (such as adding new rows, updating rows, or deleting rows) are automatically incorporated into all relevant indexes with complete transparency to the users.

Views

Views are customized presentations of data in one or more tables or other views. A view can also be considered a stored query. Views do not actually contain data. Rather, they derive their data from the tables on which they are based, referred to as the base tables of the views.

Like tables, views can be queried, updated, inserted into, and deleted from, with some restrictions. All operations performed on a view actually affect the base tables of the view.

Views provide an additional level of table security by restricting access to a predetermined set of rows and columns of a table. They also hide data complexity and store complex queries.

Clusters

Clusters are groups of one or more tables physically stored together because they share common columns and are often used together. Because related rows are physically stored together, disk access time improves.

Like indexes, clusters do not affect application design. Whether a table is part of a cluster is transparent to users and to applications. Data stored in a clustered table is accessed by SQL in the same way as data stored in a nonclustered table.

Synonyms

A synonym is an alias for any table, view, materialized view, sequence, procedure, function, package, type, Java class schema object, user-defined object type, or another synonym. Because a synonym is simply an alias, it requires no storage other than its definition in the data dictionary.

> **See Also:** Chapter 5, "Schema Objects" for more information on these and other schema objects

Overview of the Oracle Data Dictionary

Each Oracle database has a data dictionary. An Oracle data dictionary is a set of tables and views that are used as a read-only reference about the database. For example, a data dictionary stores information about both the logical and physical structure of the database. A data dictionary also stores the following information:

- The valid users of an Oracle database
- Information about integrity constraints defined for tables in the database
- The amount of space allocated for a schema object and how much of it is in use

A data dictionary is created when a database is created. To accurately reflect the status of the database at all times, the data dictionary is automatically updated by Oracle in response to specific actions, such as when the structure of the database is altered. The database relies on the data dictionary to record, verify, and conduct ongoing work. For example, during database operation, Oracle reads the data dictionary to verify that schema objects exist and that users have proper access to them.

See Also: Chapter 7, "The Data Dictionary"

Overview of the Oracle Instance

An Oracle database server consists of an Oracle database and an Oracle instance. Every time a database is started, a system global area (SGA) is allocated and Oracle background processes are started. The combination of the background processes and memory buffers is called an Oracle instance.

Real Application Clusters: Multiple Instance Systems

Some hardware architectures (for example, shared disk systems) enable multiple computers to share access to data, software, or peripheral devices. Real Application Clusters (RAC) takes advantage of such architecture by running multiple instances that share a single physical database. In most applications, RAC enables access to a single database by users on multiple machines with increased performance.

An Oracle database server uses memory structures and processes to manage and access the database. All memory structures exist in the main memory of the computers that constitute the database system. Processes are jobs that work in the memory of these computers.

See Also: *Oracle Real Application Clusters Administrator's Guide*

Instance Memory Structures

Oracle creates and uses memory structures to complete several jobs. For example, memory stores program code being run and data shared among users. Two basic memory structures are associated with Oracle: the system global area and the program global area. The following subsections explain each in detail.

System Global Area

The System Global Area (SGA) is a shared memory region that contains data and control information for one Oracle instance. Oracle allocates the SGA when an instance starts and deallocates it when the instance shuts down. Each instance has its own SGA.

Users currently connected to an Oracle database share the data in the SGA. For optimal performance, the entire SGA should be as large as possible (while still fitting in real memory) to store as much data in memory as possible and to minimize disk I/O.

The information stored in the SGA is divided into several types of memory structures, including the database buffers, redo log buffer, and the shared pool.

Database Buffer Cache of the SGA Database buffers store the most recently used blocks of data. The set of database buffers in an instance is the database buffer cache. The buffer cache contains modified as well as unmodified blocks. Because the most recently (and often, the most frequently) used data is kept in memory, less disk I/O is necessary, and performance is improved.

Redo Log Buffer of the SGA The redo log buffer stores redo entries—a log of changes made to the database. The redo entries stored in the redo log buffers are written to an online redo log, which is used if database recovery is necessary. The size of the redo log is static.

Shared Pool of the SGA The shared pool contains shared memory constructs, such as shared SQL areas. A shared SQL area is required to process every unique SQL statement submitted to a database. A shared SQL area contains information such as the parse tree and execution plan for the corresponding statement. A single shared SQL area is used by multiple applications that issue the same statement, leaving more shared memory for other uses.

> **See Also:** "SQL Statements" on page 1-43 for more information about shared SQL areas

Statement Handles or Cursors A cursor is a handle or name for a private SQL area in which a parsed statement and other information for processing the statement are kept. (Oracle Call Interface, OCI, refers to these as statement handles.) Although most Oracle users rely on automatic cursor handling of Oracle utilities, the programmatic interfaces offer application designers more control over cursors.

For example, in precompiler application development, a cursor is a named resource available to a program and can be used specifically to parse SQL statements embedded within the application. Application developers can code an application so it controls the phases of SQL statement execution and thus improves application performance.

Program Global Area

The Program Global Area (PGA) is a memory buffer that contains data and control information for a server process. A PGA is created by Oracle when a server process is started. The information in a PGA depends on the Oracle configuration.

See Also: Chapter 8, "Memory Architecture"

Oracle Background Processes

An Oracle database uses memory structures and processes to manage and access the database. All memory structures exist in the main memory of the computers that constitute the database system. Processes are jobs that work in the memory of these computers.

The architectural features discussed in this section enable the Oracle database to support:

- Many users concurrently accessing a single database
- The high performance required by concurrent multiuser, multiapplication database systems

Oracle creates a set of background processes for each instance. The background processes consolidate functions that would otherwise be handled by multiple Oracle programs running for each user process. They asynchronously perform I/O and monitor other Oracle process to provide increased parallelism for better performance and reliability.

There are numerous background processes, and each Oracle instance can use several background processes.

> **See Also:** "Background Processes" on page 9-5 for more information on some of the most common background processes

Process Architecture

A process is a "thread of control" or a mechanism in an operating system that can run a series of steps. Some operating systems use the terms job or task. A process generally has its own private memory area in which it runs.

An Oracle database server has two general types of processes: user processes and Oracle processes.

User (Client) Processes User processes are created and maintained to run the software code of an application program (such as an OCI or OCCI program) or an Oracle tool (such as Enterprise Manager). User processes also manage communication with the server process through the program interface, which is described in a later section.

Oracle Processes Oracle processes are invoked by other processes to perform functions on behalf of the invoking process.

Oracle creates server processes to handle requests from connected user processes. A server process communicates with the user process and interacts with Oracle to carry out requests from the associated user process. For example, if a user queries some data not already in the database buffers of the SGA, then the associated server process reads the proper data blocks from the datafiles into the SGA.

Oracle can be configured to vary the number of user processes for each server process. In a dedicated server configuration, a server process handles requests for a single user process. A shared server configuration lets many user processes share a small number of server processes, minimizing the number of server processes and maximizing the use of available system resources.

On some systems, the user and server processes are separate, while on others they are combined into a single process. If a system uses the shared server or if the user and server processes run on different machines, then the user and server processes must be separate. Client/server systems separate the user and server processes and run them on different machines.

See Also: Chapter 9, "Process Architecture"

Overview of Accessing the Database

This section describes Oracle Net Services, as well as how to start up the database.

Network Connections

Oracle Net Services is Oracle's mechanism for interfacing with the communication protocols used by the networks that facilitate distributed processing and distributed databases.

Communication protocols define the way that data is transmitted and received on a network. Oracle Net Services supports communications on all major network protocols, including TCP/IP, HTTP, FTP, and WebDAV.

Using Oracle Net Services, application developers do not need to be concerned with supporting network communications in a database application. If a new protocol is used, then the database administrator makes some minor changes, while the application requires no modifications and continues to function.

Oracle Net, a component of Oracle Net Services, enables a network session from a client application to an Oracle database server. Once a network session is established, Oracle Net acts as the data courier for both the client application and

the database server. It establishes and maintains the connection between the client application and database server, as well as exchanges messages between them. Oracle Net can perform these jobs because it is located on each computer in the network.

> **See Also:** *Oracle Net Services Administrator's Guide*

Starting Up the Database

The three steps to starting an Oracle database and making it available for systemwide use are:

1. Start an instance.
2. Mount the database.
3. Open the database.

A database administrator can perform these steps using the SQL*Plus `STARTUP` statement or Enterprise Manager. When Oracle starts an instance, it reads the server parameter file (SPFILE) or initialization parameter file to determine the values of initialization parameters. Then, it allocates an SGA, and creates background processes.

> **See Also:** Chapter 12, "Database and Instance Startup and Shutdown"

How Oracle Works

The following example describes the most basic level of operations that Oracle performs. This illustrates an Oracle configuration where the user and associated server process are on separate machines (connected through a network).

1. An instance has started on the computer running Oracle (often called the host or database server).
2. A computer running an application (a local machine or client workstation) runs the application in a user process. The client application attempts to establish a connection to the server using the proper Oracle Net Services driver.
3. The server is running the proper Oracle Net Services driver. The server detects the connection request from the application and creates a dedicated server process on behalf of the user process.
4. The user runs a SQL statement and commits the transaction. For example, the user changes a name in a row of a table.

5. The server process receives the statement and checks the shared pool for any shared SQL area that contains a similar SQL statement. If a shared SQL area is found, then the server process checks the user's access privileges to the requested data, and the previously existing shared SQL area is used to process the statement. If not, then a new shared SQL area is allocated for the statement, so it can be parsed and processed.
6. The server process retrieves any necessary data values from the actual datafile (table) or those stored in the SGA.
7. The server process modifies data in the system global area. The DBW*n* process writes modified blocks permanently to disk when doing so is efficient. Because the transaction is committed, the LGWR process immediately records the transaction in the redo log file.
8. If the transaction is successful, then the server process sends a message across the network to the application. If it is not successful, then an error message is transmitted.
9. Throughout this entire procedure, the other background processes run, watching for conditions that require intervention. In addition, the database server manages other users' transactions and prevents contention between transactions that request the same data.

See Also: Chapter 9, "Process Architecture" for more information about Oracle configuration

Overview of Oracle Utilities

Oracle provides several utilities for data transfer, data maintenance, and database administration, including Data Pump Export and Import, SQL*Loader, and LogMiner.

See Also: Chapter 11, "Oracle Utilities"

Oracle Database Features

This section contains the following topics:

- Overview of Scalability and Performance Features
- Overview of Manageability Features
- Overview of Database Backup and Recovery Features

- Overview of High Availability Features
- Overview of Business Intelligence Features
- Overview of Content Management Features
- Overview of Security Features
- Overview of Data Integrity and Triggers
- Overview of Information Integration Features

Overview of Scalability and Performance Features

Oracle includes several software mechanisms to fulfill the following important requirements of an information management system:

- Data concurrency of a multiuser system must be maximized.
- Data must be read and modified in a consistent fashion. The data a user is viewing or changing is not changed (by other users) until the user is finished with the data.
- High performance is required for maximum productivity from the many users of the database system.

This contains the following sections:

- Concurrency
- Read Consistency
- Locking Mechanisms
- Quiesce Database
- Real Application Clusters
- Portability

Concurrency

A primary concern of a multiuser database management system is how to control concurrency, which is the simultaneous access of the same data by many users. Without adequate concurrency controls, data could be updated or changed improperly, compromising data integrity.

One way to manage data concurrency is to make each user wait for a turn. The goal of a database management system is to reduce that wait so it is either nonexistent or negligible to each user. All data manipulation language statements should proceed

with as little interference as possible, and destructive interactions between concurrent transactions must be prevented. Destructive interaction is any interaction that incorrectly updates data or incorrectly alters underlying data structures. Neither performance nor data integrity can be sacrificed.

Oracle resolves such issues by using various types of locks and a multiversion consistency model. These features are based on the concept of a transaction. It is the application designer's responsibility to ensure that transactions fully exploit these concurrency and consistency features.

Read Consistency

Read consistency, as supported by Oracle, does the following:

- Guarantees that the set of data seen by a statement is consistent with respect to a single point in time and does not change during statement execution (statement-level read consistency)
- Ensures that readers of database data do not wait for writers or other readers of the same data
- Ensures that writers of database data do not wait for readers of the same data
- Ensures that writers only wait for other writers if they attempt to update identical rows in concurrent transactions

The simplest way to think of Oracle's implementation of read consistency is to imagine each user operating a private copy of the database, hence the multiversion consistency model.

Read Consistency, Undo Records, and Transactions To manage the multiversion consistency model, Oracle must create a read-consistent set of data when a table is queried (read) and simultaneously updated (written). When an update occurs, the original data values changed by the update are recorded in the database undo records. As long as this update remains part of an uncommitted transaction, any user that later queries the modified data views the original data values. Oracle uses current information in the system global area and information in the undo records to construct a read-consistent view of a table's data for a query.

Only when a transaction is committed are the changes of the transaction made permanent. Statements that start *after* the user's transaction is committed only see the changes made by the committed transaction.

The transaction is key to Oracle's strategy for providing read consistency. This unit of committed (or uncommitted) SQL statements:

- Dictates the start point for read-consistent views generated on behalf of readers
- Controls when modified data can be seen by other transactions of the database for reading or updating

Read-Only Transactions By default, Oracle guarantees statement-level read consistency. The set of data returned by a single query is consistent with respect to a single point in time. However, in some situations, you might also require transaction-level read consistency. This is the ability to run multiple queries within a single transaction, all of which are read-consistent with respect to the same point in time, so that queries in this transaction do not see the effects of intervening committed transactions. If you want to run a number of queries against multiple tables and if you are not doing any updating, you prefer a read-only transaction.

Locking Mechanisms

Oracle also uses locks to control concurrent access to data. When updating information, the data server holds that information with a lock until the update is submitted or committed. Until that happens, no one else can make changes to the locked information. This ensures the data integrity of the system.

Oracle provides unique non-escalating row-level locking. Unlike other data servers that "escalate" locks to cover entire groups of rows or even the entire table, Oracle always locks only the row of information being updated. Because Oracle includes the locking information with the actual rows themselves, Oracle can lock an unlimited number of rows so users can work concurrently without unnecessary delays.

Automatic Locking Oracle locking is performed automatically and requires no user action. Implicit locking occurs for SQL statements as necessary, depending on the action requested. Oracle's lock manager automatically locks table data at the row level. By locking table data at the row level, contention for the same data is minimized.

Oracle's lock manager maintains several different types of row locks, depending on what type of operation established the lock. The two general types of locks are exclusive locks and share locks. Only one exclusive lock can be placed on a resource (such as a row or a table); however, many share locks can be placed on a single resource. Both exclusive and share locks always allow queries on the locked resource but prohibit other activity on the resource (such as updates and deletes).

Manual Locking Under some circumstances, a user might want to override default locking. Oracle allows manual override of automatic locking features at both the

row level (by first querying for the rows that will be updated in a subsequent statement) and the table level.

Quiesce Database

Database administrators occasionally need isolation from concurrent non-database administrator actions, that is, isolation from concurrent non-database administrator transactions, queries, or PL/SQL statements. One way to provide such isolation is to shut down the database and reopen it in restricted mode. You could also put the system into quiesced state without disrupting users. In quiesced state, the database administrator can safely perform certain actions whose executions require isolation from concurrent non-DBA users.

See Also: Chapter 13, "Data Concurrency and Consistency"

Real Application Clusters

Real Application Clusters (RAC) comprises several Oracle instances running on multiple clustered machines, which communicate with each other by means of a so-called interconnect. RAC uses cluster software to access a shared database that resides on shared disk. RAC combines the processing power of these multiple interconnected computers to provide system redundancy, near linear scalability, and high availability. RAC also offers significant advantages for both OLTP and data warehouse systems and all systems and applications can efficiently exploit clustered environments.

You can scale applications in RAC environments to meet increasing data processing demands without changing the application code. As you add resources such as nodes or storage, RAC extends the processing powers of these resources beyond the limits of the individual components.

See Also: *Oracle Real Application Clusters Administrator's Guide*

Portability

Oracle provides unique portability across all major platforms and ensures that your applications run without modification after changing platforms. This is because the Oracle code base is identical across platforms, so you have identical feature functionality across all platforms, for complete application transparency. Because of this portability, you can easily upgrade to a more powerful server as your requirements change.

Overview of Manageability Features

People who administer the operation of an Oracle database system, known as database administrators (DBAs), are responsible for creating Oracle databases, ensuring their smooth operation, and monitoring their use. In addition to the many alerts and advisors Oracle provides, Oracle also offers the following features:

Self-Managing Database

Oracle Database provides a high degree of self-management - automating routine DBA tasks and reducing complexity of space, memory, and resource administration. Oracle self-managing database features include the following: automatic undo management, dynamic memory management, Oracle-managed files, mean time to recover, free space management, multiple block sizes, and Recovery Manager (RMAN).

Oracle Enterprise Manager

Enterprise Manager is a system management tool that provides an integrated solution for centrally managing your heterogeneous environment. Combining a graphical console, Oracle Management Servers, Oracle Intelligent Agents, common services, and administrative tools, Enterprise Manager provides a comprehensive systems management platform for managing Oracle products.

From the client interface, the Enterprise Manager Console, you can perform the following tasks:

- Administer the complete Oracle environment, including databases, *i*AS servers, applications, and services
- Diagnose, modify, and tune multiple databases
- Schedule tasks on multiple systems at varying time intervals
- Monitor database conditions throughout the network
- Administer multiple network nodes and services from many locations
- Share tasks with other administrators
- Group related targets together to facilitate administration tasks
- Launch integrated Oracle and third-party tools
- Customize the display of an Enterprise Manager administrator

SQL*Plus

SQL*Plus is a tool for entering and running ad-hoc database statements. It lets you run SQL statements and PL/SQL blocks, and perform many additional tasks as well.

See Also: *SQL*Plus User's Guide and Reference*

Automatic Storage Management

Automatic Storage Management automates and simplifies the layout of datafiles, control files, and log files. Database files are automatically distributed across all available disks, and database storage is rebalanced whenever the storage configuration changes. It provides redundancy through the mirroring of database files, and it improves performance by automatically distributing database files across all available disks. Rebalancing of the database's storage automatically occurs whenever the storage configuration changes.

The Scheduler

To help simplify management tasks, as well as providing a rich set of functionality for complex scheduling needs, Oracle provides a collection of functions and procedures in the `DBMS_SCHEDULER` package. Collectively, these functions are called the Scheduler, and they are callable from any PL/SQL program.

The Scheduler lets database administrators and application developers control when and where various tasks take place in the database environment. For example, database administrators can schedule and monitor database maintenance jobs such as backups or nightly data warehousing loads and extracts.

Database Resource Manager

Traditionally, the operating systems regulated resource management among the various applications running on a system, including Oracle databases. The Database Resource Manager controls the distribution of resources among various sessions by controlling the execution schedule inside the database. By controlling which sessions run and for how long, the Database Resource Manager can ensure that resource distribution matches the plan directive and hence, the business objectives.

See Also: Chapter 14, "Manageability"

Overview of Database Backup and Recovery Features

In every database system, the possibility of a system or hardware failure always exists. If a failure occurs and affects the database, then the database must be recovered. The goals after a failure are to ensure that the effects of all committed transactions are reflected in the recovered database and to return to normal operation as quickly as possible while insulating users from problems caused by the failure.

Oracle provides various mechanisms for the following:

- Database recovery required by different types of failures
- Flexible recovery operations to suit any situation
- Availability of data during backup and recovery operations so users of the system can continue to work

Types of Failures

Several circumstances can halt the operation of an Oracle database. The most common types of failure are described in the following table.

Failure	Description
User error	Requires a database to be recovered to a point in time before the error occurred. For example, a user could accidentally drop a table. To enable recovery from user errors and accommodate other unique recovery requirements, Oracle provides exact point-in-time recovery. For example, if a user accidentally drops a table, the database can be recovered to the instant in time before the table was dropped.
Statement failure	Occurs when there is a logical failure in the handling of a statement in an Oracle program. When statement failure occurs, any effects of the statement are automatically undone by Oracle and control is returned to the user.
Process failure	Results from a failure in a user process accessing Oracle, such as an abnormal disconnection or process termination. The background process PMON automatically detects the failed user process, rolls back the uncommitted transaction of the user process, and releases any resources that the process was using.