

[Home \(/\)](#) / [Database \(/en/database/database.html\)](#) / [Oracle Database Online Documentation 12c Release 1 \(12.1\) \(../index.html\)](#) / [Database Administration \(../nav/portal_4.htm\)](#)

Database Concepts

1 Introduction to Oracle Database

This chapter provides an overview of Oracle Database.

This chapter contains the following sections:

- [About Relational Databases \(intro.htm#GUID-166C1E31-CDBC-47D9-867A-3D4C9AAC837D\)](#)
- [Schema Objects \(intro.htm#GUID-35C20601-E266-486E-987B-7F355DB10DD4\)](#)
- [Data Access \(intro.htm#GUID-BCDCCCA1-D0E2-4A04-BD7D-AD573DC75151\)](#)
- [Transaction Management \(intro.htm#GUID-ACACFFBC-BCB9-4BA9-A2A3-1B9B2003B3B4\)](#)
- [Oracle Database Architecture \(intro.htm#GUID-CF765A7D-9429-4901-BF33-36E0B0220293\)](#)
- [Oracle Database Documentation Roadmap \(intro.htm#GUID-C756D70F-75D4-4234-BEB5-CB05A4742128\)](#)

About Relational Databases

Every organization has information that it must store and manage to meet its requirements. For example, a corporation must collect and maintain human resources records for its employees. This information must be available to those who need it.

An **information system** ([glossary.htm#GUID-3A7A1F9C-9CA9-45E5-A5C5-6D578E68F4D](#)) is a formal system for storing and processing information. An information system could be a set of cardboard boxes containing manila folders along with rules for how to store and retrieve the folders. However, most companies today use a database to automate their information systems. A database is an organized collection of information treated as a unit. The purpose of a database is to collect, store, and retrieve related information for use by database applications.

Database Management System (DBMS)

A **database management system (DBMS)** is software that controls the storage, organization, and retrieval of data.

Typically, a DBMS has the following elements:

- Kernel code

This code manages memory and storage for the DBMS.

- Repository of metadata

This repository is usually called a **data dictionary** ([glossary.htm#GUID-D6A3934F-BA6D-464F-9612-683E20514A4](#)).

- Query language

This language enables applications to access the data.

A **database application** (glossary.htm#GUID-ED5708AA-D3BF-4CEF-AA1910F6705D75F1) is a software program that interacts with a database to access and manipulate data.

The first generation of database management systems included the following types:

- Hierarchical

A **hierarchical database** (glossary.htm#GUID-10A60173-6F11-4372-BB70-A1E3794C65D) organizes data in a tree structure. Each parent record has one or more child records, similar to the structure of a file system.

- Network

A **network database** (glossary.htm#GUID-2934B37E-9E0F-4A47-BBCC-32A408E385F) is similar to a hierarchical database, except records have a many-to-many rather than a one-to-many relationship.

The preceding database management systems stored data in rigid, predetermined relationships. Because no data definition language existed, changing the structure of the data was difficult. Also, these systems lacked a simple query language, which hindered application development.

Relational Model

In his seminal 1970 paper "A Relational Model of Data for Large Shared Data Banks," E. F. Codd defined a relational model based on mathematical set theory. Today, the most widely accepted database model is the relational model.

A **relational database** (glossary.htm#GUID-C76D1B18-B273-4163-A12A-88AC06E8057) is a database that conforms to the relational model. The relational model has the following major aspects:

- Structures

Well-defined objects store or access the data of a database.

- Operations

Clearly defined actions enable applications to manipulate the data and structures of a database.

- Integrity rules

Integrity rules govern operations on the data and structures of a database.

A relational database stores data in a set of simple relations. A **relation** (glossary.htm#GUID-F81FA4BC-AFAF-4A1B-BDFA-0BDF42512B7A) is a set of tuples. A **tuple** (glossary.htm#GUID-C7F27053-7E3C-4F7A-945A-66598050C70) is an unordered set of attribute values.

A **table** (glossary.htm#GUID-DA8F7E11-B6AF-4ED3-B2A9-B5741E9AE2D4) is a two-dimensional representation of a relation in the form of rows (tuples) and columns (attributes). Each row in a table has the same set of columns. A relational database is a database that stores data in relations (tables). For example, a relational database could store information about company employees in an employee table, a department table, and a salary table.

See Also:

<http://portal.acm.org/citation.cfm?id=362685> (<http://portal.acm.org/citation.cfm?id=362685>) for an abstract and link to Codd's paper

Relational Database Management System (RDBMS)

The relational model is the basis for a **relational database management system (RDBMS)**. An RDBMS moves data into a database, stores the data, and retrieves it so that applications can manipulate it.

An RDBMS distinguishes between the following types of operations:

- Logical operations

In this case, an application specifies *what* content is required. For example, an application requests an employee name or adds an employee record to a table.

- Physical operations

In this case, the RDBMS determines *how* things should be done and carries out the operation. For example, after an application queries a table, the database may use an index to find the requested rows, read the data into memory, and perform many other steps before returning a result to the user. The RDBMS stores and retrieves data so that physical operations are transparent to database applications.

Oracle Database is an RDBMS. An RDBMS that implements object-oriented features such as user-defined types, inheritance, and polymorphism is called an **object-relational database management system (ORDBMS)** (glossary.htm#GUID-B5A50116-D8BD-431E-93E6-4C6516297756). Oracle Database has extended the relational model to an object-relational model, making it possible to store complex business models in a relational database.

Brief History of Oracle Database

The current version of Oracle Database is the result of over 35 years of innovative development.

Highlights in the evolution of Oracle Database include the following:

- Founding of Oracle

In 1977, Larry Ellison, Bob Miner, and Ed Oates started the consultancy Software Development Laboratories, which became Relational Software, Inc. (RSI). In 1983, RSI became Oracle Systems Corporation and then later Oracle Corporation.

- First commercially available RDBMS

In 1979, RSI introduced Oracle V2 (Version 2) as the first commercially available **SQL** (glossary.htm#GUID-30430C74-B2C1-43A0-B7FF-64111B786BC5)-based RDBMS, a landmark event in the history of relational databases.

- Portable version of Oracle Database

Oracle Version 3, released in 1983, was the first relational database to run on mainframes, minicomputers, and PCs. The database was written in C, enabling the database to be ported to multiple platforms.

- Enhancements to concurrency control, data distribution, and scalability

Version 4 introduced multiversion **read consistency** (glossary.htm#GUID-A2D92FBD-9C36-432E-A44F-062DB2E5527). Version 5, released in 1985, supported client/server computing and **distributed database** (glossary.htm#GUID-DB7296DF-74E0-45E1-9BC2-5052DD543214) systems. Version 6 brought enhancements to disk I/O, row locking, scalability, and backup and recovery. Also, Version 6 introduced the first version of the **PL/SQL** (glossary.htm#GUID-68532EDC-3324-48DE-9AD5-5CD2EC671EA) language, a proprietary procedural extension to SQL.

- PL/SQL stored program units

Oracle7, released in 1992, introduced PL/SQL stored procedures and triggers.

- Objects and partitioning

Oracle8 was released in 1997 as the object-relational database, supporting many new data types. Additionally, Oracle8 supported partitioning of large tables.

- Internet computing

Oracle8i Database, released in 1999, provided native support for internet protocols and server-side support for Java. Oracle8i was designed for internet computing, enabling the database to be deployed in a multitier environment.

- Oracle Real Application Clusters (Oracle RAC)

Oracle9i Database introduced Oracle RAC in 2001, enabling multiple instances to access a single database simultaneously. Additionally, Oracle XML Database (**Oracle XML DB** (glossary.htm#GUID-FFC80925-7152-41DA-8D78-47D8201FA2CA)) introduced the ability to store and query XML.

- Grid computing

Oracle Database 10g introduced **grid computing** (glossary.htm#GUID-EF507F26-F59A-415E-8C3E-3B95BFBA2F7C) in 2003. This release enabled organizations to virtualize computing resources by building a **grid infrastructure** (glossary.htm#GUID-B9983AA6-0B7F-4FC3-B5FC-3DD4F4E1997) based on low-cost commodity servers. A key goal was to make the database self-managing and self-tuning. **Oracle Automatic Storage Management (Oracle ASM)** (glossary.htm#GUID-8E45BEB5-F8AF-4896-B04A-016812BD96D) helped achieve this goal by virtualizing and simplifying database storage management.

- Manageability, diagnosability, and availability

Oracle Database 11g, released in 2007, introduced a host of new features that enabled administrators and developers to adapt quickly to changing business requirements. The key to adaptability is simplifying the information infrastructure by consolidating information and using automation wherever possible.

- Plugging In to the Cloud

Oracle Database 12c, released in 2013, was designed for the Cloud, featuring a new Multitenant architecture, In-Memory column store, and support for JSON documents. Oracle Database 12c helps customers make more efficient use of their IT resources, while continuing to reduce costs and improve service levels for users.

Schema Objects

One characteristic of an RDBMS is the independence of physical data storage from logical data structures.

In Oracle Database, a database **schema** ([glossary.htm#GUID-D55ED7A6-3BC4-4A16-981F-2E7E905A64D](#)) is a collection of logical data structures, or schema objects. A database user owns a database schema, which has the same name as the **user name** ([glossary.htm#GUID-E141F64B-73ED-43DC-B22A-DA0210B8EBEF](#)).

Schema objects are user-created structures that directly refer to the data in the database. The database supports many types of schema objects, the most important of which are tables and indexes.

A schema object is one type of **database object** ([glossary.htm#GUID-785767E0-DB47-4579-AA4D-F46C8680ACCA](#)). Some database objects, such as profiles and roles, do not reside in schemas.

See Also:

"Introduction to Schema Objects ([tablecls.htm#GUID-72E247B5-F39A-47F1-9445-72D9221F57E3](#))"

Tables

A table describes an entity such as employees. You define a table with a table name, such as employees, and set of columns. In general, you give each **column** ([glossary.htm#GUID-F1D9AA5F-C66A-4D5F-A84C8D7360DE3BE5](#)) a name, a **data type** ([glossary.htm#GUID-3873B26C-657D-4508-B13D-955F1D5D8F4](#)), and a width when you create the table.

A table is a set of rows. A column identifies an attribute of the entity described by the table, whereas a **row** ([glossary.htm#GUID-14228D8A-2E66-49A7-B041-DB6CA759D8C](#)) identifies an instance of the entity. For example, attributes of the employees entity correspond to columns for employee ID and last name. A row identifies a specific employee.

You can optionally specify a rule, called an **integrity constraint** ([glossary.htm#GUID-67F8FE8C-EBA5-4796-820A-8919982A1411](#)), for a column. One example is a NOT NULL integrity constraint. This constraint forces the column to contain a value in every row.

See Also:

- "Overview of Tables ([tablecls.htm#GUID-F845B1A7-71E3-4312-B66D-BC16C198ECE5](#))"
- Data Integrity ([datainte.htm#GUID-6A89FF39-AD42-4399-BD1B-E51ECEE50B4E](#))

Indexes

An **index** is an optional data structure that you can create on one or more columns of a table. Indexes can increase the performance of data retrieval.

When processing a request, the database can use available indexes to locate the requested rows efficiently. Indexes are useful when applications often query a specific row or range of rows.

Indexes are logically and physically independent of the data. Thus, you can drop and create indexes with no effect on the tables or other indexes. All applications continue to function after you drop an index.

See Also:

"Introduction to Indexes ([indexiot.htm#GUID-DE7A95BC-6E4A-47EA-9FC5-B85B54F8CF41](#))"

Data Access

A general requirement for a DBMS is to adhere to accepted industry standards for a data access language.

Structured Query Language (SQL)

SQL is a set-based declarative language that provides an interface to an RDBMS such as Oracle Database. In contrast to procedural languages such as C, which describe *how* things should be done, SQL is nonprocedural and describes *what* should be done.

SQL is the ANSI standard language for relational databases. All operations on the data in an Oracle database are performed using SQL statements. For example, you use SQL to create tables and query and modify data in tables.

A SQL statement can be thought of as a very simple, but powerful, computer program or instruction. Users specify the result that they want (for example, the names of employees), not how to derive it. A SQL statement is a string of SQL text such as the following:

```
SELECT first_name, last_name FROM employees;
```

SQL statements enable you to perform the following tasks:

- Query data
- Insert, update, and delete rows in a table
- Create, replace, alter, and drop objects
- Control access to the database and its objects
- Guarantee database consistency and integrity

SQL unifies the preceding tasks in one consistent language. **Oracle SQL** ([glossary.htm#GUID-48DAF4AC-49FB-453A-A48B-AF225FEA600B](#)) is an implementation of the ANSI standard. Oracle SQL supports numerous features that extend beyond standard SQL.

See Also:

"SQL ([sqlangu.htm#GUID-DA48618A-A6BB-421A-A10A-02859D8ED9AD](#))"

PL/SQL and Java

PL/SQL is a procedural extension to Oracle SQL. PL/SQL is integrated with Oracle Database, enabling you to use all of the Oracle Database SQL statements, functions, and data types. You can use PL/SQL to control the flow of a SQL program, use variables, and write error-handling procedures.

A primary benefit of PL/SQL is the ability to store application logic in the database itself. A **PL/SQL procedure** ([glossary.htm#GUID-D2210901-18F2-43C0-B0CC-653E94CD7EF](#)) or **function** ([glossary.htm#GUID-5611828A-3621-4527-AEB1-12017A454E94](#)) is a schema object that consists of a set of SQL statements and other PL/SQL constructs, grouped together, stored in the database, and run as a unit to solve a specific problem or to perform a set of related tasks. The principal benefit of server-side programming is that built-in functionality can be deployed anywhere.

Oracle Database can also store program units written in Java. A Java stored procedure is a Java method published to SQL and stored in the database for general use. You can call existing PL/SQL programs from Java and Java programs from PL/SQL.

See Also:

"Server-Side Programming: PL/SQL and Java ([srvrside.htm#GUID-D4A154D2-DF56-45DA-863C-BED5DA6BDA34](#))" and

"Client-Side Database Programming ([cncptdev.htm#GUID-87F23274-B506-486B-8775-80016095EF5](#))"

Transaction Management

Oracle Database is designed as a multiuser database. The database must ensure that multiple users can work concurrently without corrupting one another's data.

Transactions

A **transaction** is a logical, atomic unit of work that contains one or more SQL statements. An RDBMS must be able to group SQL statements so that they are either all committed, which means they are applied to the database, or all rolled back, which means they are undone.

An illustration of the need for transactions is a funds transfer from a savings account to a checking account. The transfer consists of the following separate operations:

1. Decrease the savings account.
2. Increase the checking account.
3. Record the transaction in the transaction journal.

Oracle Database guarantees that all three operations succeed or fail as a unit. For example, if a hardware failure prevents a statement in the transaction from executing, then the other statements must be rolled back.

Transactions are one feature that set Oracle Database apart from a file system. If you perform an atomic operation that updates several files, and if the system fails halfway through, then the files will not be consistent. In contrast, a transaction moves an Oracle database from one consistent state to another. The basic principle of a transaction is "all or nothing": an atomic operation succeeds or fails as a whole.

See Also:

"Transactions ([transact.htm#GUID-B97790CB-DF82-442D-B9D5-50CCE6BF9FBD](#))"

Data Concurrency

A requirement of a multiuser RDBMS is the control of **data concurrency**, which is the simultaneous access of the same data by multiple users.

Without concurrency controls, users could change data improperly, compromising **data integrity** ([glossary.htm#GUID-9DE527B3-8901-4F4C-A18F-D2C307AFE1](#)). For example, one user could update a row while a different user simultaneously updates it.

If multiple users access the same data, then one way of managing concurrency is to make users wait. However, the goal of a DBMS is to reduce wait time so it is either nonexistent or negligible. All SQL statements that modify data must proceed with as little interference as possible. Destructive interactions, which are interactions that incorrectly update data or alter underlying data structures, must be avoided.

Oracle Database uses locks to control concurrent access to data. A **lock** (glossary.htm#GUID-6D016291-A487-4F88-BE0B-ACF8FA2AE72C) is a mechanism that prevents destructive interaction between transactions accessing a shared resource. Locks help ensure data integrity while allowing maximum concurrent access to data.

See Also:

"Overview of the Oracle Database Locking Mechanism (consist.htm#GUID-AD0CEE83-2F33-4906-94E1-3D1022924C63)"

Data Consistency

In Oracle Database, each user must see a consistent view of the data, including visible changes made by a user's own transactions and committed transactions of other users.

For example, the database must prevent the **lost update** (glossary.htm#GUID-F9DA6F63-B518-44C5-B98B-225680EB498) problem, which occurs when one transaction sees uncommitted changes made by another concurrent transaction.

Oracle Database always enforces statement-level **read consistency** (glossary.htm#GUID-A2D92FBD-9C36-432E-A44F-0462DB2E5527), which guarantees that the data that a single query returns is committed and consistent for a single point in time. Depending on the transaction isolation level, this point is the time at which the statement was opened or the time the transaction began. The Flashback Query feature enables you to specify this point in time explicitly.

The database can also provide read consistency to all queries in a transaction, known as transaction-level read consistency. In this case, each statement in a transaction sees data from the same point in time, which is the time at which the transaction began.

See Also:

- Data Concurrency and Consistency (consist.htm#GUID-E8CBA9C5-58E3-460F-A82A-850E0152E95C)
- "Oracle Database Development Guide (../ADFNS/adfns_flashback.htm#ADFNS01003)" to learn about Flashback Query

Oracle Database Architecture

A **database server** is the key to information management.

In general, a **server** (glossary.htm#GUID-A3046A66-CC14-47F6-8F91-BDF097F1CA05) reliably manages a large amount of data in a multiuser environment so that users can concurrently access the same data. A database server also prevents unauthorized access and provides efficient solutions for failure recovery.

Database and Instance

An Oracle database server consists of a database and at least one **database instance**, commonly referred to as simply an *instance*. Because an instance and a database are so closely connected, the term **Oracle database** is sometimes used to refer to both instance and database.

In the strictest sense the terms have the following meanings:

- Database

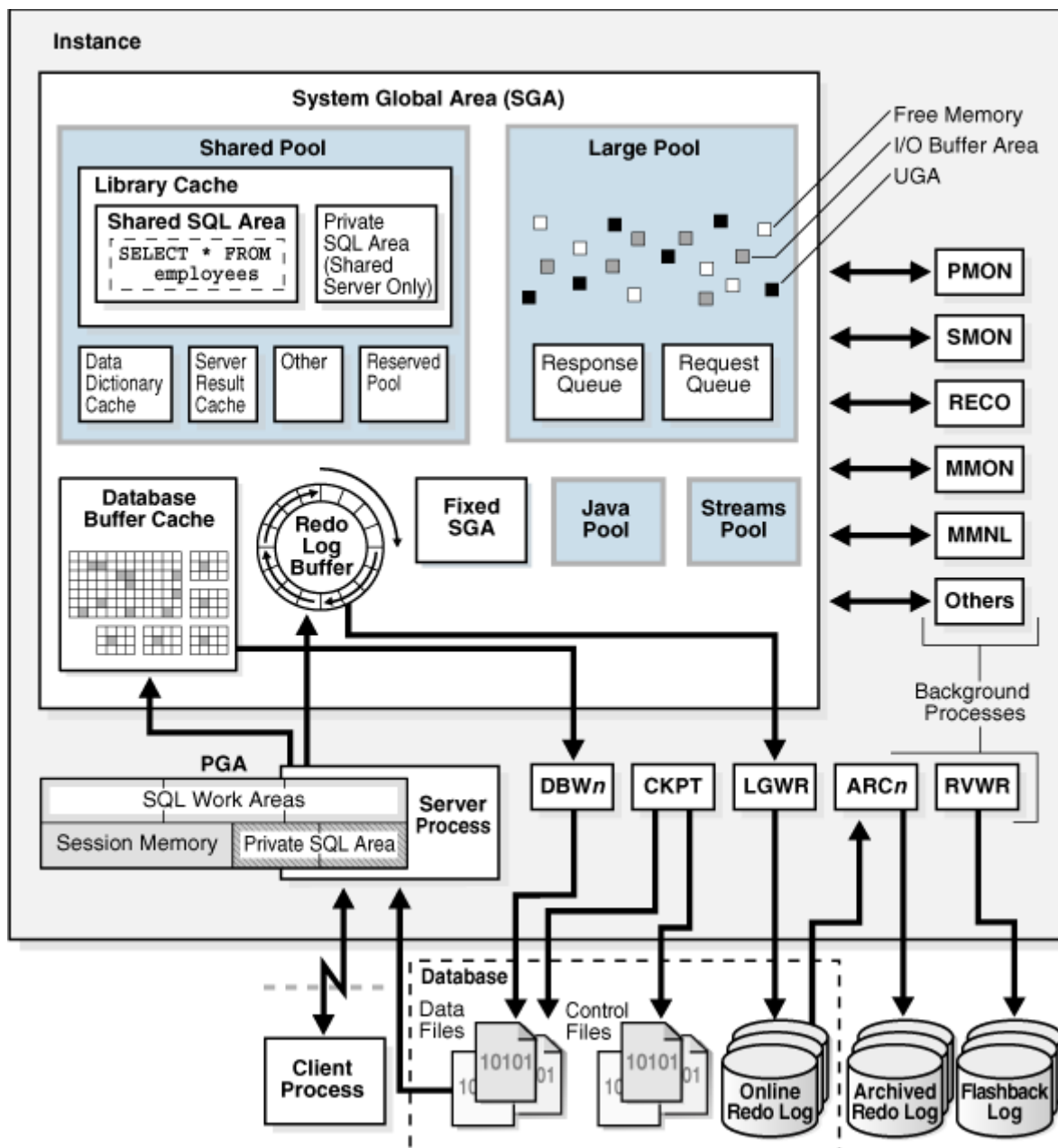
A database is a set of files, located on disk, that store data. These files can exist independently of a database instance.

- Database instance

An instance is a set of memory structures that manage database files. The instance consists of a shared memory area, called the **system global area (SGA)** (glossary.htm#GUID-78C0E867-233A-4857-B9FE-A852A9B7BDF), and a set of background processes. An instance can exist independently of database files.

Figure 1-1 (intro.htm#GUID-8F2EEEC8-0372-4419-88FF-7D77A9C0FCAD__CEGDIFEFF) shows a database and its instance. For each user connection to the instance, a **client process** (glossary.htm#GUID-6F3BBCE8-F471-41E8-8C98-F4C62A85862B) runs the application. Each client process is associated with its own **server process** (glossary.htm#GUID-E660AC1C-B704-4DC1-A35A-DB49EFB34F4A). The server process has its own private session memory, known as the **program global area (PGA)** (glossary.htm#GUID-76021F69-AC7A-4D8D-A308-AB8AC072EEF).

Figure 1-1 Oracle Instance and Database



Description of "Figure 1-1 Oracle Instance and Database" (img_text/GUID-A30AF38A-7D55-4987-A0FB-02BAAF079AC5-print.htm)

A database can be considered from both a physical and logical perspective. Physical data is data viewable at the operating system level. For example, operating system utilities such as the Linux `ls` and `ps` can list database files and processes. Logical data such as a table is meaningful only for the database. A SQL statement can list the tables in an Oracle database, but an operating system utility cannot.

The database has physical structures and logical structures. Because the physical and logical structures are separate, you can manage the physical storage of data without affecting access to logical storage structures. For example, renaming a physical database file does not rename the tables whose data is stored in this file.

See Also:

"Oracle Database Instance ([startup.htm#GUID-67247052-CE3F-44D2-BA3E-7067DEF4B6D5](#))"

Database Storage Structures

An essential task of a relational database is data storage. This section briefly describes the physical and logical storage structures that Oracle Database uses.

Physical Storage Structures

The physical database structures are the files that store the data.

When you execute a `CREATE DATABASE` statement, the following files are created:

- Data files

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

- Control files

Every Oracle database has a **control file** ([glossary.htm#GUID-DF498605-D186-451B-87AA-73DF396AD599](#)). A control file contains metadata specifying the physical structure of the database, including the database name and the names and locations of the database files.

- Online redo log files

Every Oracle Database has an **online redo log** ([glossary.htm#GUID-2A8BC112-AB70-4B06-9F85-7E975861CEE0](#)), which is a set of two or more online redo log files. An online **redo log** ([glossary.htm#GUID-F24A4593-5E47-4D06-B9D6-0061D288373C](#)) is made up of redo entries (also called redo log records), which record all changes made to data.

Many other files are important for the functioning of an Oracle database server. These include parameter files and networking files. Backup files and archived redo log files are offline files important for backup and recovery.

See Also:

"Physical Storage Structures ([physical.htm#GUID-FFA872E1-7F63-4DC5-8A35-F21394AB4595](#))"

Logical Storage Structures

Logical storage structures enable Oracle Database to have fine-grained control of disk space use.

This topic discusses logical storage structures:

- Data blocks

At the finest level of granularity, Oracle Database data is stored in data blocks. One **data block** (glossary.htm#GUID-2141C31D-6752-4260-91CE-9B4CC557247) corresponds to a specific number of bytes on disk.

- Extents

An **extent** (glossary.htm#GUID-C56D833A-B3D3-4B85-AAB4-34F7CF3F5E9) is a specific number of logically contiguous data blocks, obtained in a single allocation, used to store a specific type of information.

- Segments

A **segment** (glossary.htm#GUID-EC12AA68-8C89-43B3-B1F9-3AABF7CAEB9F) is a set of extents allocated for a user object (for example, a table or index), **undo data** (glossary.htm#GUID-297B963A-989C-4720-B061-A252FF72892), or temporary data.

- Tablespaces

A database is divided into logical storage units called tablespaces. A **tablespace** (glossary.htm#GUID-AA66891C-71B2-4D55-8F64-0E427AE24E88) is the logical container for a segment. Each tablespace consists of at least one data file.

See Also:

"Logical Storage Structures (logical.htm#GUID-13CE5EDA-8C66-4CA0-87B5-4069215A368D)"

Database Instance Structures

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 RDBMS.

When applications connect to an Oracle database, they connect to a **database instance** (glossary.htm#GUID-CFB1A30E-76B6-44EA-839E-9E63C8DA31AC). The instance services applications by allocating other memory areas in addition to the SGA, and starting other processes in addition to background processes.

Oracle Database Processes

A **process** is a mechanism in an operating system that can run a series of steps. Some operating systems use the terms *job*, *task*, or *thread*.

For the purposes of this topic, a thread is equivalent to a process. An Oracle database instance has the following types of processes:

- Client processes

These processes are created and maintained to run the software code of an application program or an Oracle tool. Most environments have separate computers for client processes.

- Background processes

These processes consolidate functions that would otherwise be handled by multiple Oracle Database programs running for each client process. Background processes asynchronously perform I/O and monitor other Oracle Database processes to provide increased parallelism for better performance and reliability.

- Server processes

These processes communicate with client processes and interact with Oracle Database to fulfill requests.

Oracle processes include server processes and background processes. In most environments, Oracle processes and client processes run on separate computers.

See Also:

"Process Architecture ([process.htm#GUID-85D9852E-5BF1-4AC0-9E5A-49F0570DBD7A](#))"

Instance Memory Structures

Oracle Database creates and uses memory structures for program code, data shared among users, and private data areas for each connected user.

The following memory structures are associated with a database instance:

- System Global Area (SGA)

The SGA is a group of shared memory structures that contain data and control information for one database instance. Examples of SGA components include the database buffer cache and shared SQL areas. Starting in Oracle Database 12c Release 1 (12.1.0.2), the SGA can contain an optional **In-Memory Column Store** ([glossary.htm#GUID-A19D33EA-7BAA-42C0-9E13-0AF686E2A976](#)) (IM column store), which enables data to be populated in memory in a **columnar format** ([glossary.htm#GUID-DDC39B24-BE95-406A-986F-D760308CA26D](#)).

- Program Global Areas (PGA)

A PGA is a memory region that contains data and control information for a server or background process. Access to the PGA is exclusive to the process. Each server process and background process has its own PGA.

See Also:

"Memory Architecture ([memory.htm#GUID-913335DF-050A-479A-A653-68A064DCCA41](#))"

Application and Networking Architecture

To take full advantage of a given computer system or network, Oracle Database enables processing to be split between the database server and the client programs. The computer running the RDBMS handles the database server responsibilities while the computers running the applications handle the interpretation and display of data.

Application Architecture

The application architecture is the computing environment in which a database application connects to an Oracle database. The two most common database architectures are client/server and multitier.

- In a **client/server architecture** ([glossary.htm#GUID-6F0D8417-F8ED-4D91-BB81-F86499ADD69](#)), the client application initiates a request for an operation to be performed on the database server.

The server runs Oracle Database software and handles the functions required for concurrent, shared data access. The server receives and processes requests that originate from clients.

- In a traditional **multitier architecture** ([glossary.htm#GUID-A8812375-43CA-4C1A-B376-A59C6FADF50B](#)), one or more application servers perform parts of the operation.

An **application server** (glossary.htm#GUID-1774BB0F-837E-4651-B1A3-~~BC~~468B027B8F) contains a large part of the application logic, provides access to the data for the client, and performs some query processing. In this way, the load on the database decreases. The application server can serve as an interface between clients and multiple databases and provide an additional level of security.

A **service-oriented architecture (SOA)** (glossary.htm#GUID-2CB01B71-A23D-4127-B54A-~~B~~46C23BEDFC) is a multitier architecture in which application functionality is encapsulated in services. SOA services are usually implemented as Web services. Web services are accessible through HTTP and are based on XML-based standards such as Web Services Description Language (WSDL) and SOAP.

Oracle Database can act as a Web service provider in a traditional multitier or SOA environment.

See Also:

- "Overview of Multitier Architecture (dist_pro.htm#GUID-A65D8A2C-2DE0-4438-A042-54A3C07FF006)"
- *Oracle XML DB Developer's Guide* (../ADXDB/xdm_web_services.htm#ADXDB3900) for more information about using Web services with the database

Networking Architecture

Oracle Net Services is the interface between the database and the network communication protocols 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.

Oracle Net (glossary.htm#GUID-4F4232CE-6A82-4F39-8565-28AF563F0AE), a component of Oracle Net Services, establishes and maintains a network session from a client application to a database server. After a network session is established, Oracle Net acts as the data courier for both the client application and the database server, exchanging messages between them. Oracle Net can perform these jobs because it is located on each computer in the network.

An important component of Net Services is the **Oracle Net Listener** (glossary.htm#GUID-504800FA-2159-495B-8E4F-B2D4D938EA85) (called the *listener*), which is a process that runs on the database or elsewhere in the network. Client applications send connection requests to the listener, which manages the traffic of these requests to the database. When a connection is established, the client and database communicate directly.

The most common ways to configure an Oracle database to service client requests are:

- Dedicated server architecture

Each client process connects to a **dedicated server** (glossary.htm#GUID-8B14C804-9D68-471C-A581-~~5~~AEE673A9FCD) process. The server process is not shared by any other client for the duration of the client's session. Each new session is assigned a dedicated server process.

- Shared server architecture

The database uses a pool of **shared server** (glossary.htm#GUID-E94CE0E3-CC86-4F46-B8EF-~~5~~1945F026326) processes for multiple sessions. A client process communicates with a **dispatcher** (glossary.htm#GUID-D8288F03-1F84-4766-AF5B-B341A714BD3F), which is a process that enables many clients to connect to the same database instance without the need for a dedicated server process for each client.

See Also:

- "Overview of Oracle Networking Architecture ([dist_pro.htm#GUID-136C7637-89B5-4C5A-B186-E06F86CE2359](#))"
- *Oracle Database Net Services Administrator's Guide* ([../NETAG/net_arch.htm#NETAG1511](#)) to learn more about Oracle Net architecture
- *Oracle XML DB Developer's Guide* ([../ADXDB/xdb22pro.htm#ADXDB2500](#)) for information about using WebDAV with the database

Multitenant Architecture

Starting in Oracle Database 12c, the **multitenant architecture** enables an Oracle database to be a multitenant container database (CDB).

A **CDB** ([glossary.htm#GUID-135FF536-DE9B-40CF-9F42-C246762BD77F](#)) is a single physical database that contains zero, one, or many user-created pluggable databases. A **pluggable database (PDB)** ([glossary.htm#GUID-15825EC2-B512-4FD1-A734-AE4DAB3B3AEC](#)) is a portable collection of schemas, schema objects, and nonschema objects that appears to an **Oracle Net** ([glossary.htm#GUID-4F4232CE-6A82-4F39-8565-281AF563F0AE](#)) client as a non-CDB. A **non-CDB** ([glossary.htm#GUID-B2710CAA-8F00-40B3-97AB-4521D2147EE8](#)) is a traditional Oracle database that cannot contain PDBs.

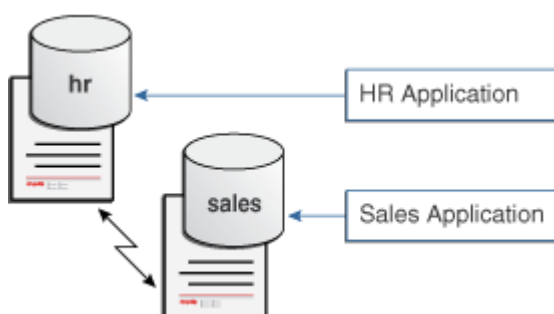
Starting in Oracle Database 12c, you must create a database as either a CDB or non-CDB. You can plug a non-CDB into a CDB as a PDB. To move a PDB to a non-CDB, you must use Oracle Data Pump.

By consolidating multiple physical databases on separate computers into a single database on a single computer, the multitenant architecture provides the following benefits:

- Cost reduction for hardware
- Easier and more rapid movement of data and code
- Easier management and monitoring of the physical database
- Separation of data and code
- Separation of duties between a **PDB administrator** ([glossary.htm#GUID-6E44D9F6-9FE1-4FEA-998A-51ACA5775AF5](#)), who manages only the PDBs to which she or he is granted privileges, and the **CDB administrator** ([glossary.htm#GUID-24233CA1-CF2F-4E02-B202-83DB0571DDB9](#)), who manages the entire CDB

Figure 1-2 ([intro.htm#GUID-AB84D6C9-4BBE-4D36-992F-2BB85739329F__BHCFDEGI](#)) shows two separate non-CDBs: hr and sales. Each non-CDB has its own memory and set of database files, and resides on its own computer. Each non-CDB has its own dedicated user application.

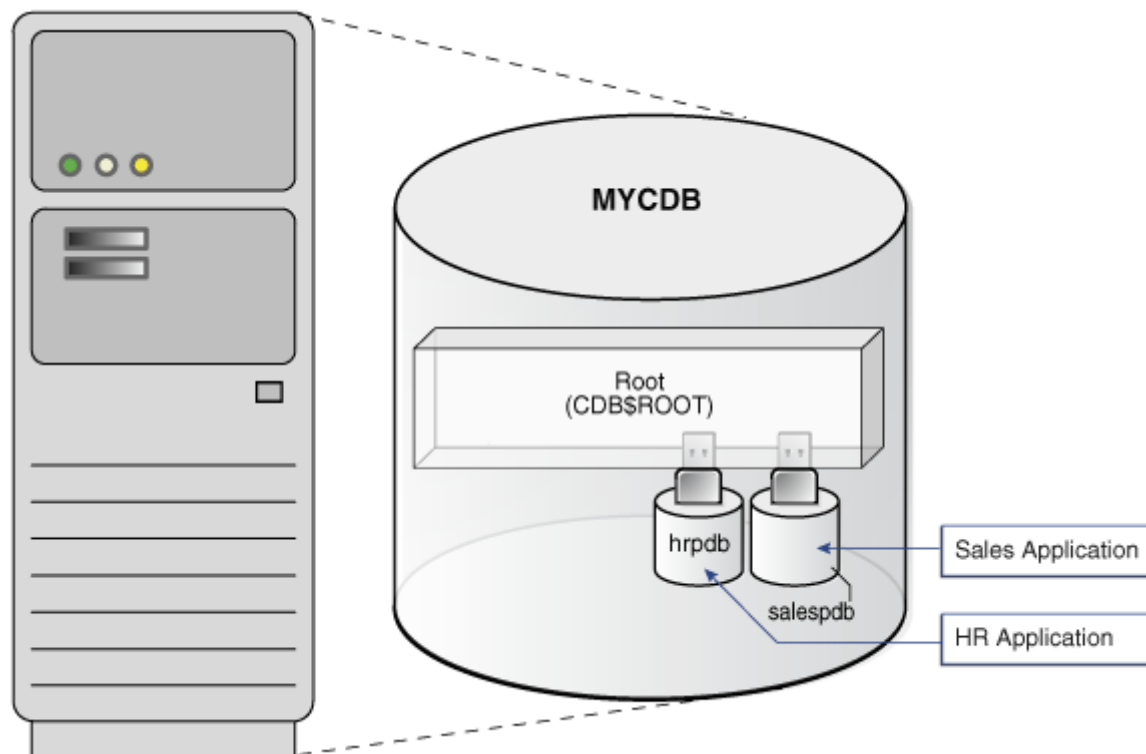
Figure 1-2 Non-CDBs



Description of "Figure 1-2 Non-CDBs" ([img_text/GUID-5983CD37-26F6-492A-98F8-720D2F1056E1-print.htm](#))

Figure 1-3 (intro.htm#GUID-AB84D6C9-4BBE-4D36-992F-2BB85739329F__BHCIEHAI) shows the same data after being consolidated into the CDB named MYCDB.

Figure 1-3 CDB



Description of "Figure 1-3 CDB" (img_text/GUID-8AEFA5D-EE9D-457D-96C8-67780B040656-print.htm)

Physically, MYCDB is an Oracle database. MYCDB has one database instance (although multiple instances are possible in Oracle Real Application Clusters) and one set of database files, just like a non-CDB.

MYCDB contains two PDBs: hrpdb and salespdb. As shown in Figure 1-3 (intro.htm#GUID-AB84D6C9-4BBE-4D36-992F-2BB85739329F__BHCIEHAI), these PDBs appear to their respective applications just as they did before database consolidation. To administer the CDB itself or any PDB within it, a CDB administrator can connect to the **CDB root** (glossary.htm#GUID-1C63428A-905A-4918-95AA-F3BBA3D2B825), which is a collection of schemas, schema objects, and nonschema objects to which all PDBs belong.

CDBs and non-CDBs have architectural differences. This manual assumes the architecture of a non-CDB unless otherwise indicated.

See Also:

"Introduction to the Multitenant Architecture (cdbovrww.htm#GUID-267F7D12-D33F-4AC9-AA45E9CD671B6F22)" to learn more about CDBs

Oracle Database Documentation Roadmap

To a new user, the Oracle Database documentation library can seem daunting. Not only are there are over 200 manuals, but many of these manuals are several hundred pages long. However, the documentation is designed with specific access paths to ensure that users are able to find the information they need as efficiently as possible.

The documentation set is divided into three layers or groups: basic, intermediate, and advanced. Users begin with the manuals in the basic group (*Oracle Database 2 Day DBA* (./ADMQS/toc.htm), *Oracle Database 2 Day Developer's Guide* (./TDDDG/toc.htm), or this manual), proceed to the manuals in the intermediate group (the *2 Day + series*), and finally to the advanced manuals, which include the remainder of the documentation set.

Oracle Database Documentation: Basic Group

Technical users who are new to Oracle Database begin by reading one or more manuals in the basic group from cover to cover. Each manual in this group is designed to be read in two days.

In addition to this manual, the basic group includes:

- *Oracle Database 2 Day DBA* ([../ADMQS/toc.htm](#))

A database administrator (DBA) is responsible for the overall operation of Oracle Database. This task-based quick start teaches DBAs how to perform daily database administrative tasks using Oracle Enterprise Manager Database Express (EM Express). The manual teaches DBAs how to perform all typical administrative tasks needed to keep the database operational, including how to perform basic troubleshooting and performance monitoring activities.

- *Oracle Database 2 Day Developer's Guide* ([../TDDDG/toc.htm](#))

This task-based quick start guide explains how to use the basic features of Oracle Database through SQL and PL/SQL.

The manuals in the basic group are closely related, which is reflected in the number of cross-references. For example, *Oracle Database Concepts* frequently sends users to a *2 Day* manual to learn how to perform a task based on a concept. The *2 Day* manuals frequently reference *Oracle Database Concepts* for conceptual background about a task.

Oracle Database Documentation: Intermediate Group

The next step up from the basic group is the intermediate group.

Manuals in the intermediate group are prefixed with the word *2 Day +* because they expand on and assume information contained in the *2 Day* manuals. The *2 Day +* manuals cover topics in more depth than was possible in the basic manuals, or cover topics of special interest.

As shown in the following table, the *2 Day +* manuals are divided into manuals for DBAs and developers.

Table 1-1 Intermediate Group: 2 Day + Guides

Database Administrators	Database Developers
<i>Oracle Database 2 Day + Performance Tuning Guide</i> (../TDPPT/toc.htm)	<i>Oracle Database 2 Day + Application Express Developer's Guide</i> (../TDPAX/toc.htm)
<i>Oracle Database 2 Day + Real Application Clusters Guide</i> (../TDPRC/toc.htm)	<i>Oracle Database 2 Day + Java Developer's Guide</i> (../TDPJD/toc.htm)
<i>Oracle Database 2 Day + Security Guide</i> (../TDPSPG/tdpsg_intro.htm#TDPSPG94426)	<i>Oracle Database 2 Day + PHP Developer's Guide</i> (../TDPPIH/toc.htm)

Advanced Group

The advanced group manuals are intended for expert users who require more detailed information about a particular topic than can be provided by the 2 Day + manuals.

Essential reference manuals in the advanced group include:

- *Oracle Database SQL Language Reference* (../SQLRF/toc.htm)

This manual is the definitive source of information about Oracle SQL.

- *Oracle Database Reference* (../REFRN/toc.htm)

The manual is the definitive source of information about initialization parameters, data dictionary views, and dynamic performance views.

The advanced guides are too numerous to list in this section. Table 1-2 (intro.htm#GUID-7BDBD26D-AD11-4599-8C7B-EAB9AAEAA558__ADVANCEDGROUP-185DC652)lists guides that the majority of expert DBAs and developers use.

Table 1-2 Advanced Group

Database Administrators	Database Developers
<i>Oracle Database Administrator's Guide</i> (../ADMIN/toc.htm)	<i>Oracle Database Development Guide</i> (../ADFNS/toc.htm)
<i>Oracle Database Performance Tuning Guide</i> (../TGDBA/toc.htm) and <i>Oracle Database SQL Tuning Guide</i> (../TGSQL/toc.htm)	<i>Oracle Database PL/SQL Language Reference</i> (../LNPLS/toc.htm)
<i>Oracle Database Backup and Recovery User's Guide</i> (../BRADV/toc.htm)	<i>Oracle Database PL/SQL Packages and Types Reference</i> (../ARPLS/toc.htm)
<i>Oracle Real Application Clusters Administration and Deployment Guide</i> (../RACAD/toc.htm)	

Other advanced guides required by a particular user depend on the area of responsibility of this user. For example, a security officer naturally refers to the *Oracle Database Security Guide* (../DBSEG/toc.htm).