A **database** is an organized collection of data. **A general-purpose database management system** (DBMS) is a software system designed to allow the definition, creation, querying, update, and administration of databases.

A DBMS is responsible for maintaining the integrity and security of stored data, and for recovering information if the system fails.

**Data dictionary**

The Oracle data dictionary is one of the most important components of the Oracle DBMS. **It contains all information about the structures and objects of the database** such as

**• tables,**

**• columns,**

**• users,**

**• data files etc.**

**The data stored in the data dictionary are also often called metadata**. Although it is usually the domain of database administrators (DBAs), the data dictionary is a valuable source of information for end users and developers.

Example: USER, ALL , and DBA

A **relational database** is a type of database that stores and provides access to data points that are related to one another. Relational databases are based on the relational model, an intuitive, straightforward way of representing data in tables. In a relational database, each row in the table is a record with a unique ID called the key. The columns of the table hold attributes of the data, and each record usually has a value for each attribute, making it easy to establish the relationships among data points.

Two type:

Logical: what content is required?

Physical: how things should be done.

**Type of databases:**

* **In memory:** An in-memory database is a database that primarily resides in main memory but is typically backed-up by non-volatile computer data storage. Main memory databases are faster than disk databases, and so are often used where response time is critical, such as in telecommunications network equipment
* **An active database** includes an event-driven architecture which can respond to conditions both inside and outside the database. Possible uses include security monitoring, alerting, statistics gathering and authorization. Many databases provide active database features in the form of database triggers.
* **A cloud database** relies on cloud technology. Both the database and most of its DBMS reside remotely, "in the cloud," while its applications are both developed by programmers and later maintained and utilized by (application's) end-users through a web browser and Open APIs.
* **Data warehouses** archive data from operational databases and often from external sources such as market research firms.
* A **distributed database** is one in which both the data and the DBMS span multiple computers.
* A **document-oriented database** is designed for storing, retrieving, and managing document-oriented, or semi Structured data, information. Document-oriented databases are one of the main categories of NoSQL databases
* An **embedded database** system is a DBMS which is tightly integrated with an application software that requires access to stored data in such a way that the DBMS is hidden from the application’s end-users and requires little or no ongoing maintenance.

**Database models:** A database model is a type of data model that determines the logical structure of a database and fundamentally determines in which manner data can be stored, organized, and manipulated. The most popular example of a database model is the relational model (or the SQL approximation of relational), which uses a table-based format.

Common logical data models for databases include:

• Hierarchical database model

• Network model

• Relational model

• Entity–relationship model

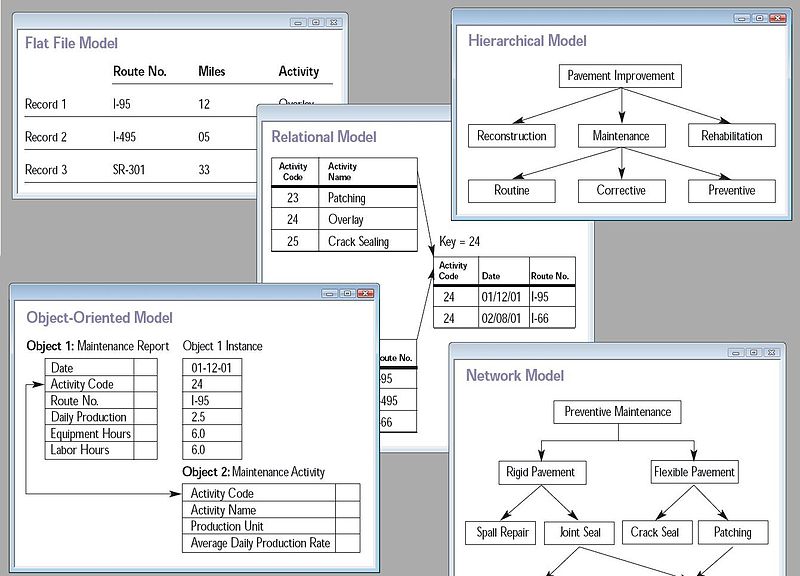
• Enhanced entity–relationship model

• Object model

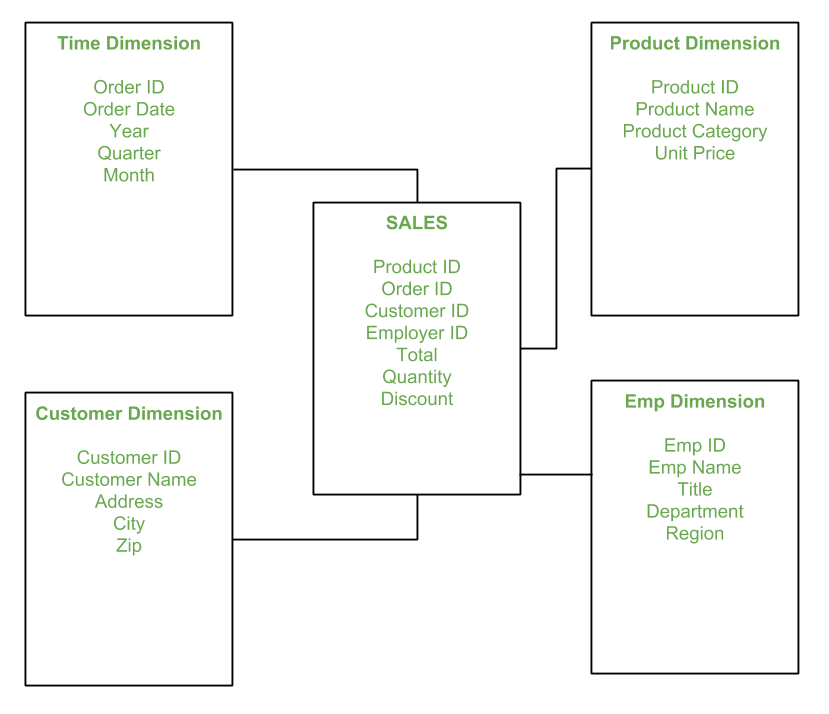
• Document model

• Entity–attribute–value model

• Star schema

[](https://upload.wikimedia.org/wikipedia/commons/3/3b/Database_models.jpg)

Star Schema:



**External, conceptual, and internal views**

Traditional view of data A database management system provides **three** views of the database data:

• The **Internal level** The internal schema defines the physical storage structure of the database. The internal schema is a very low-level representation of the entire database. It contains multiple occurrences of multiple types of internal record. In the ANSI term, it is also called "stored record'.

• The **conceptual level** The conceptual schema describes the Database structure of the whole database for the community of users. This schema hides information about the physical storage structures and focuses on describing data types, entities, relationships, etc

• **The external level** (or physical level) An external schema describes the part of the database which specific user is interested in. It hides the unrelated details of the database from the user. There may be "n" number of external views for each database. Occasionally it stores data of individual views (**materialized views**), computed from generic data, if performance justification exists for such redundancy. It balances all the external views' performance requirements, possibly conflicting, in an attempt to optimize overall performance across all activities.

A **database schema** is a logical container for data structures, called schema objects.

A schema is a namespace - a logical thing. It is used to organize the names of database objects. It has nothing to do with the way the data is stored.

**Tablespaces**: Oracle stores data logically in tablespaces and physically in datafiles associated with the corresponding tablespace. A tablespace is a physical thing. It's a container for data and has nothing to do with the logical organization of the database objects.

Char vs Nchar:

It takes 1 byte of data and the maximum storage capacity is of 8000 byte

nchar stands for national character. It takes up two bytes to store the data and can store upto 4000 chars.

|  |  |
| --- | --- |
| **Char** | **n-char** |
| A char stores fixed-length, **non-Unicode** characters. | A n-char stores fixed-length **Unicode** characters. |
| **One byte** to store the data. | **Two bytes** to store data. |
| char stores upto **8000** bytes. | n-char stores upto **4000** bytes. |
| It is widely preferred datatype. | It is preferred less. |
| It uses **Unicode** standards for storing the data. | It uses **ASCII** standards to store the data. |

**Integrity constraints** are named rules that restrict the values for one or more columns in a table. These rules prevent invalid data entry into tables. Also, constraints can prevent the deletion of a table when certain dependencies exist.

1. **Primary Key** : that helps to identify every record present in that table uniquely
2. **Candidate Key**: Candidate keys are those attributes that uniquely identify rows of a table. The Primary Key of a table is selected from one of the candidate keys.
3. **Super Key**: Super Key is the set of all the keys which help to identify rows in a table uniquely
4. **Foreign Key**: Foreign Key is used to establish relationships between two tables. A foreign key will require each value in a column or set of columns to match the Primary Key of the referential table.
5. **Composite Key**: Composite Key is a set of two or more attributes that help identify each tuple in a table uniquely. The attributes in the set may not be unique when considered separately
6. **Alternate Key**
7. **Unique Key:** Unique Key is a column or set of columns that uniquely identify each record in a table. All values will have to be unique in this Key. A unique Key differs from a primary key because it can have only one null value.

**Normalization in DBMS**

Normalization is the process of minimizing redundancy from a relation or set of relations. Redundancy in relation may cause insertion, deletion and updation anomalies. So, it helps to minimize the redundancy in relations. Normal forms are used to eliminate or reduce redundancy in database tables.

**1. First Normal Form –**

If a relation contain composite or multi-valued attribute, it violates first normal form or a relation is in first normal form if it does not contain any composite or multi-valued attribute. A relation is in first normal form if every attribute in that relation is singled valued attribute.

**Below is not in 1NF**

ID Name Courses

------------------

1 A c1, c2 (multi valued attributes)

2 E c3

3 M C2, c3

**2. Second Normal Form** - a relation must be in first normal form and relation must not contain any partial dependency. A relation is in 2NF if it has No Partial Dependency, i.e., no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table.

A **partial dependency** occurs when a non-key attribute is dependent on only a *part* of a composite primary key, rather than the entire key. This can lead to redundancy and inconsistencies in the data.

Candidate Key is AB

Example 2 – Consider following functional dependencies in relation R (A, B , C, D )

AB -> C [A and B together determine C]

BC -> D [B and C together determine D]

In the above relation, AB is the only candidate key and there is no partial dependency, i.e., any proper subset of AB doesn’t determine any non-prime attribute.

**3. Third Normal Form –**

A relation is in third normal form, if there is no transitive dependency for non-prime attributes as well as it is in second normal form. A relation is in 3NF if at least one of the following condition holds in every non-trivial function dependency X –> Y

X is a super key.

Y is a prime attribute (each element of Y is part of some candidate key).

**Data warehouse**:

Dimensional Modelling (DM) is a data structure technique optimized for data storage in a Data warehouse. The purpose of dimensional modelling is to optimize the database for faster retrieval of data. The concept of Dimensional Modelling was developed by Ralph Kimball and consists of “fact” and “dimension” tables.

A dimensional model in data warehouse is designed to read, summarize, analyse numeric information like values, balances, counts, weights, etc. in a data warehouse. In contrast, relation models are optimized for addition, updating and deletion of data in a real-time Online Transaction System.

These dimensional and relational models have their unique way of data storage that has specific advantages.

For instance, in the relational mode, normalization and ER models reduce redundancy in data. On the contrary, dimensional model in data warehouse arranges data in such a way that it is easier to retrieve information and generate reports.

**Elements of Dimensional Data Model**

**Fact**

Facts are the measurements/metrics or facts from your business process. For a Sales business process, a measurement would be quarterly sales number

**Dimension**

Dimension provides the context surrounding a business process event. In simple terms, they give who, what, where of a fact. In the Sales business process, for the fact quarterly sales number, dimensions would be

Who – Customer Names

Where – Location

What – Product Name

In other words, a dimension is a window to view information in the facts.

In another word

***Fact Table****: Stores* ***quantitative data*** *(measurable metrics, like revenue, sales, quantity).*

***Dimension Tables****: Store* ***descriptive attributes*** *related to facts (e.g., time, product, customer).*

**Attributes**

The Attributes are the various characteristics of the dimension in dimensional data modeling.

In the Location dimension, the attributes can be

State

Country

Zipcode etc.

Attributes are used to search, filter, or classify facts. Dimension Tables contain Attributes.

**Fact Table**

A fact table is a primary table in dimension modelling. A Fact Table contains

Measurements/facts

Foreign key to dimension table

**Dimension Table**

A dimension table contains dimensions of a fact.

They are joined to fact table via a foreign key.

Dimension tables are de-normalized tables.

The Dimension Attributes are the various columns in a dimension table

Dimensions offers descriptive characteristics of the facts with the help of their attributes

No set limit set for given for number of dimensions

The dimension can also contain one or more hierarchical relationships

**Types of Dimensions in Data Warehouse**

Following are the Types of Dimensions in Data Warehouse:

* Conformed Dimension
* Outrigger Dimension
* Shrunken Dimension
* Role-playing Dimension
* Dimension to Dimension Table
* Junk Dimension
* Degenerate Dimension
* Swappable Dimension
* Step Dimension

**Steps of Dimensional Modelling**

The accuracy in creating your Dimensional modeling determines the success of your data warehouse implementation. Here are the steps to create Dimension Model

* Identify Business Process
* Identify Grain (level of detail)
* Identify Dimensions
* Identify Facts
* Build Star

**Partitioning of Table:**

There are so many aspects which are important in improving the performance of SQL. Partition allows tables, indexes and indexes organized tables to be subdivided into smaller pieces. Table partition is used to reduce the cost and improving performance of the application. There are some partition mechanisms using which one can divide a table into smaller pieces. Partitions can be used in so many applications where we need to improve the performance. Each partition has its own name, and it has own memory storage.

**What is mean by Table Partition?**

Following are Advantages of Partition:

1. Increase Performance

2. Increases availability

3. Enable storage cost optimization

4. Enables Simpler management

**When to partition the table?**

1. Table should be greater than 2 GB

2. Tables which contains historical data in which new data will be added in to newest partition. The real-life example of this is historical table which contains updatable data for one year other data is read only.

3. When contents of the table need to be distributed in different storage devices.

4. When table performance is weak and we need to improve performance of application.

### Types Of Table partitioning:

There are following types of Table partition:

1.Range Partition

2.List Partition

3.Hash Partition

Create table Employee(emp\_no number(2),Salary number(2))

**partition by range(Salary)**

(partition p1 values less than(10000),

partition p2 values less than(20000),

partition p3 values less than(30000),

partition p4 values less than(maxvalue));

Select \* from Employee partition(p1);

Alter table Employee add partition p5 values less than(50000);

Alter table Employee drop partition p1;

You can tell Oracle to compute the statistics by looking at all rows:

ANALYZE TABLE scott.emp COMPUTE STATISTICS FOR TABLE;

You can tell Oracle to use a sample of 1064 rows(Sampled)

ANALYZE TABLE scott.emp ESTIMATE STATISTICS FOR TABLE;

ANALYZE TABLE scott.emp ESTIMATE STATISTICS FOR TABLE SAMPLE 10 PERCENT;

## **Database Sharding in Oracle**

With Oracle 12cR1, Oracle introduced “Global Data Service/Global Service Manager” (GDS/DSM) with the primary goal is to provide Routing, Load Balancing, and Service failover capabilities.

Oracle introduced Oracle Database Sharding as a new feature in 12cR2, which uses the GDS framework as the Shard Director and provided some basic sharding capabilities. With the release of Oracle 18cR1, Oracle added some significant capabilities within sharding such as User Defined Sharding, PDB aware sharding, RAC aware as well as mid-tier sharding. With the release of 19c, Oracle has introduced further enhancements to the PDB Aware sharding capabilities, mid-tier sharding capabilities.

### The Basic Architecture

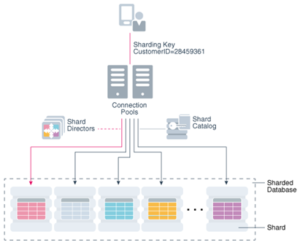
The diagram below (taken from Oracle’s documentation) is a very good starting point to understand the architecture of the Oracle Database sharding solution. The key components of an Oracle SDB are:

**Sharded database (SDB)** – a single logical Oracle Database that is horizontally partitioned across a pool of physical Oracle Databases (shards) that share no hardware or software

**Shards** – independent physical Oracle Databases that hosts a subset of the sharded database

* **Global service** – database services that provide access to data in an SDB
* **Shard catalog (SCAT)–** An Oracle Database with the express purpose of being the persistent store for SDB configuration metadata.
* **Shard directors –** network listeners that enable high-performance connection routing based on a sharding key
* **Connection pools** – at runtime, act as shard directors by routing database requests across pooled connections
* **Management interfaces** – GDSCTL (command-line utility) and Oracle Enterprise Manager (GUI)

Each shard is a self-contained Oracle database with its own resource pool of CPUs, Storage, and networking. There is no requirement of having shared resources amongst shards. Shards within an SDB can be located in network proximity or be geographically separated. In Cloud terminology, that would be within a region or across regions. The current version of Oracle Data Guard is shard aware. For HA, the standby shards can be placed in the same datacenter (or region) where the primary shards are placed. For DR, the standby shards are located in another region.



In oracle two type optimizers are there.

Rule based

Cost based

**Rule Based Optimization**: This is an old technique. Basically, the RBO used a set of rules to determine how to execute a query. E.g. If an index is available on a table, the RBO rules can be to always use that index (a RBO for our travel analogy can be avoid all routes with speed brakers). As it turns out that this is simpler to implement but not the best strategy always and can backfire.

**Cost Based Optimization**: Motivation behind CBO is to come up with the cheapest execution plan available for each SQL statement. The cheapest plan is the one that will use the least amount of resources (in relation to our travel analogy this can be Petrol, time, etc.).

**What is MongoDB?**

* [MongoDB](https://www.geeksforgeeks.org/mongodb/what-is-mongodb-working-and-features/)is a popular open-source, [NoSQL](https://www.geeksforgeeks.org/dbms/introduction-to-nosql/)database that is designed to store and manage large volumes of unstructured or semi-structured data.
* MongoDB stores data in flexible, JSON-like documents.
* MongoDB includes an aggregation framework(sort, filter, group etc on data pipeline) for processing data and transforming it in a way that allows us to perform complex data manipulations (e.g. grouping, filtering, and sorting).
* MongoDB uses replica sets, which provide high availability by replicating data across multiple servers.
* Uses a document-oriented model, where data is stored in flexible, schema-less BSON (Binary JSON) documents. Each document can have its own structure, which allows for dynamic schema design.
* Provides horizontal scalability and high availability. It uses a distributed architecture where data is partitioned across multiple nodes using sharding.
* Optimized for high read/write performance with large volumes of unstructured data.

**What is PostgreSQL?**

* [PostgreSQL](https://www.geeksforgeeks.org/postgresql/postgresql-tutorial/) is an advanced, open-source relational database management system ([RDBMS](https://www.geeksforgeeks.org/dbms/rdbms-full-form/)) known for its robustness, extensibility, and standards compliance.
* PostgreSQL follows the [SQL](https://www.geeksforgeeks.org/sql/sql-tutorial/)standard but also includes many advanced features that make it one of the most versatile and powerful relational databases available.
* PostgreSQL is often used to power the backend of web and mobile applications, especially for applications requiring **complex queries** and **data integrity.**
* MongoDB is good at storing and retrieving large amounts of unstructured data while PostgreSQL is designed to handle structured data with complex relationships.
* Follows a tabular relational model, with structured tables that enforce schema constraints. Relationships between data entities are managed using primary and foreign keys.
* It allows monolithic architecture with strong **ACID** (Atomicity, Consistency, Isolation, Durability) compliance. Its vertical scalability supports complex transactions, stored procedures and data integrity constraints.
* Excels in managing complex queries and ensuring data consistency in high-concurrency environments.

| **Feature** | **MongoDB** | **PostgreSQL** |
| --- | --- | --- |
| Data Model | Document-oriented (NoSQL) | Relational (SQL) |
| Schema | Flexible (schema-less) | Rigid (schema-enforced) |
| Scalability | Horizontal (sharding) | Vertical |
| Query Language | JavaScript-like query syntax | Standard SQL |
| Performance | Optimized for unstructured data | Optimized for complex queries |
| ACID Compliance | Limited | Full |
| Use Cases | IoT, real-time analytics, CMS | Financial systems, ERP, CRM |

| **Feature** | **Oracle** | **PostgreSQL** |
| --- | --- | --- |
| **Recursive Queries** | **CONNECT BY (proprietary syntax)** | **Standard CTEs with WITH RECURSIVE** |
| **Full-text Search** | **❌** | **✅ Built-in support** |
| **JSON support** | **Limited (improving in recent versions)** | **✅ Native JSON and JSONB** |
| **Geometric types** | **❌** | **✅ POINT, LINE, POLYGON** |
| **Network address types** | **❌** | **✅ INET, CIDR** |
| **Range types (e.g., date/number)** | **❌** | **✅ Built-in support** |
| **B-tree indexes** | **✅ Default** | **✅ Default** |
| **Bitmap indexes** | **✅** | **❌** |
| **Hash indexes** | **❌** | **✅** |
| **Autonomous transactions** | **✅** | **❌** |
| **Clustering** | **✅ Real Application Clusters (RAC)** | **❌ No built-in clustering (use Patroni, Stolon, etc.)** |
| **Result cache** | **✅ Query + PL/SQL function result cache** | **❌** |
| **In-Memory Column Store** | **✅** | **❌ (can use extensions like Citus or TimescaleDB)** |
| **Automatic memory management** | **✅** | **⚠️ Manual tuning required** |
| **Memory management** | **✅ Automatic** | **⚠️ Manual tuning required** |
| **Storage management** | **✅ Automatic Storage Management (ASM)** | **⚠️ Manual configuration** |
| **Performance monitoring** | **✅ Automatic Workload Repository (AWR)** | **⚠️ Extensions like pg\_stat\_statements** |
| **Workload management** | **✅ Database Resource Manager** | **⚠️ Manual or third-party tools** |
| **Backup and recovery** | **⚠️ Complex procedures** | **✅ Simple tools (pg\_dump, pg\_restore)** |

An autonomous transaction is a separate, independent mini transaction inside a larger transaction.

oracle savepoint vs an autonomous transaction

 **Savepoint**: Like playing a video game and setting a **checkpoint** — if something goes wrong, you go **back to the checkpoint**, not the very beginning.

 **Autonomous Transaction**: Like opening a **new window/tab** to do something separately (like send an email), then closing it and returning to your main work — no effect on each other.

CREATE OR REPLACE PROCEDURE log\_error(p\_msg IN VARCHAR2) IS

PRAGMA AUTONOMOUS\_TRANSACTION;

BEGIN

INSERT INTO error\_log (error\_message, log\_time)

VALUES (p\_msg, SYSDATE);

COMMIT; -- Must commit, since it’s independent

END;

BEGIN

-- some business logic

UPDATE accounts SET balance = balance - 100 WHERE account\_id = 101;

-- simulate error logging

log\_error('Account updated successfully');

ROLLBACK; -- will NOT affect the log\_error insert

END;

B-tree: is of order m is an mway tree( tree where Each node may have up to m children ).

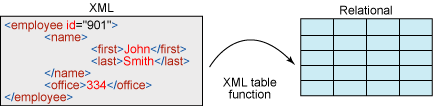
RAC is sharing one database between many instances.

Oracle redefined the term "grid" to refer to the adding and removal of small rack-mounted blade servers to Oracle:

Since Oracle 10g, Oracle has added new functions XQuery and XMLTable to its arsenal of XML processing APIs.

XMLQuery lets you construct XML data and query XML and relational data using the XQuery language.

XMLTable lets you create relational tables and columns from XQuery query results.



**Example:**

CREATE TABLE EMPLOYEES

(

   id     NUMBER,

   data   XMLTYPE

);

INSERT INTO EMPLOYEES

VALUES (1, xmltype ('<Employees>

<Employee emplid="1111" type="admin">

<firstname>John</firstname>

<lastname>Watson</lastname>

<age>30</age>

<email>johnwatson@sh.com</email>

</Employee>

<Employee emplid="2222" type="admin">

<firstname>Sherlock</firstname>

<lastname>Homes</lastname>

<age>32</age>

<email>sherlock@sh.com</email>

</Employee>

<Employee emplid="3333" type="user">

<firstname>Jim</firstname>

<lastname>Moriarty</lastname>

<age>52</age>

<email>jim@sh.com</email>

</Employee>

<Employee emplid="4444" type="user">

<firstname>Mycroft</firstname>

<lastname>Holmes</lastname>

<age>41</age>

<email>mycroft@sh.com</email>

</Employee>

</Employees>'));

**XMLTable Systax:**

XMLTable('<XQuery>'

PASSING <xml column>

COLUMNS <new column name> <column type> PATH <XQuery path>)

--print firstname, lastname and age of all employees

SELECT t.id, x.\*

FROM employees t,

XMLTABLE ('/Employees/Employee'

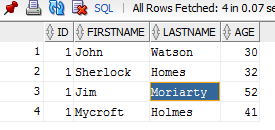
PASSING t.data

COLUMNS firstname VARCHAR2(30) PATH 'firstname',

lastname VARCHAR2(30) PATH 'lastname',

age int PATh 'age') x

WHERE t.id = 1;



--print firstname of all employees

SELECT t.id, x.\*

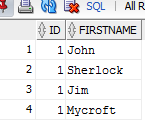
FROM employees t,

XMLTABLE ('/Employees/Employee/firstname'

PASSING t.data

COLUMNS firstname VARCHAR2 (30) PATH 'text()') x

WHERE t.id = 1;



--print employee type of all employees

   SELECT emp.id, x.\*

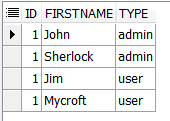
     FROM employees emp,

          XMLTABLE ('/Employees/Employee'

                    PASSING emp.data

                    COLUMNS firstname VARCHAR2(30) PATH 'firstname',

                            type VARCHAR2(30) PATH '@type') x;

**Output:**  


--print firstname and lastname of employee with id 2222

   SELECT t.id, x.\*

     FROM employees t,

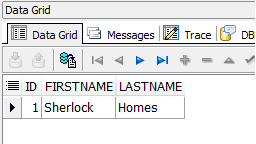
          XMLTABLE ('/Employees/Employee[@emplid=2222]'

                    PASSING t.data

                    COLUMNS firstname VARCHAR2(30) PATH 'firstname',

                            lastname VARCHAR2(30) PATH 'lastname') x

    WHERE t.id = 1;

**Output:**  


--print firstname and lastname of employees who are admins

   SELECT t.id, x.\*

     FROM employees t,

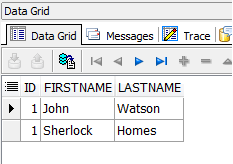
          XMLTABLE ('/Employees/Employee[@type="admin"]'

                    PASSING t.data

                    COLUMNS firstname VARCHAR2(30) PATH 'firstname',

                            lastname VARCHAR2(30) PATH 'lastname') x

    WHERE t.id = 1;



--print firstname and lastname of employees having age > 40

   SELECT t.id, x.\*

     FROM employees t,

          XMLTABLE ('/Employees/Employee[age>40]'

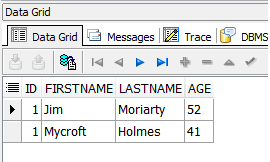
                    PASSING t.data

                    COLUMNS firstname VARCHAR2(30) PATH 'firstname',

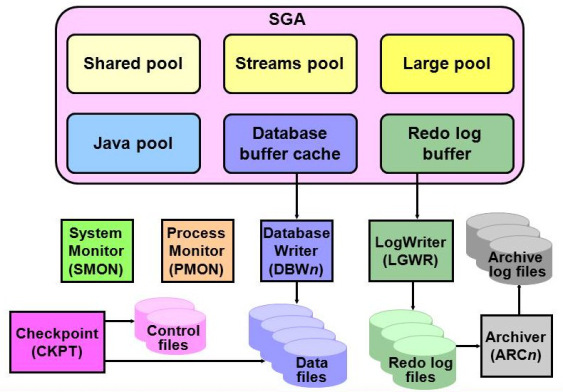
                            lastname VARCHAR2(30) PATH 'lastname',

                            age VARCHAR2(30) PATH 'age') x

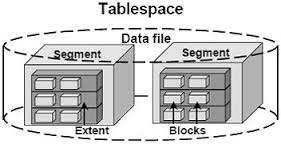
    WHERE t.id = 1;



Oracle Architecture- PFiles, SPFiles, Control Files, Redo Logs and Datafiles



Each database instance is madeup of 1 datafile, 1 control file and 2 redo log files.

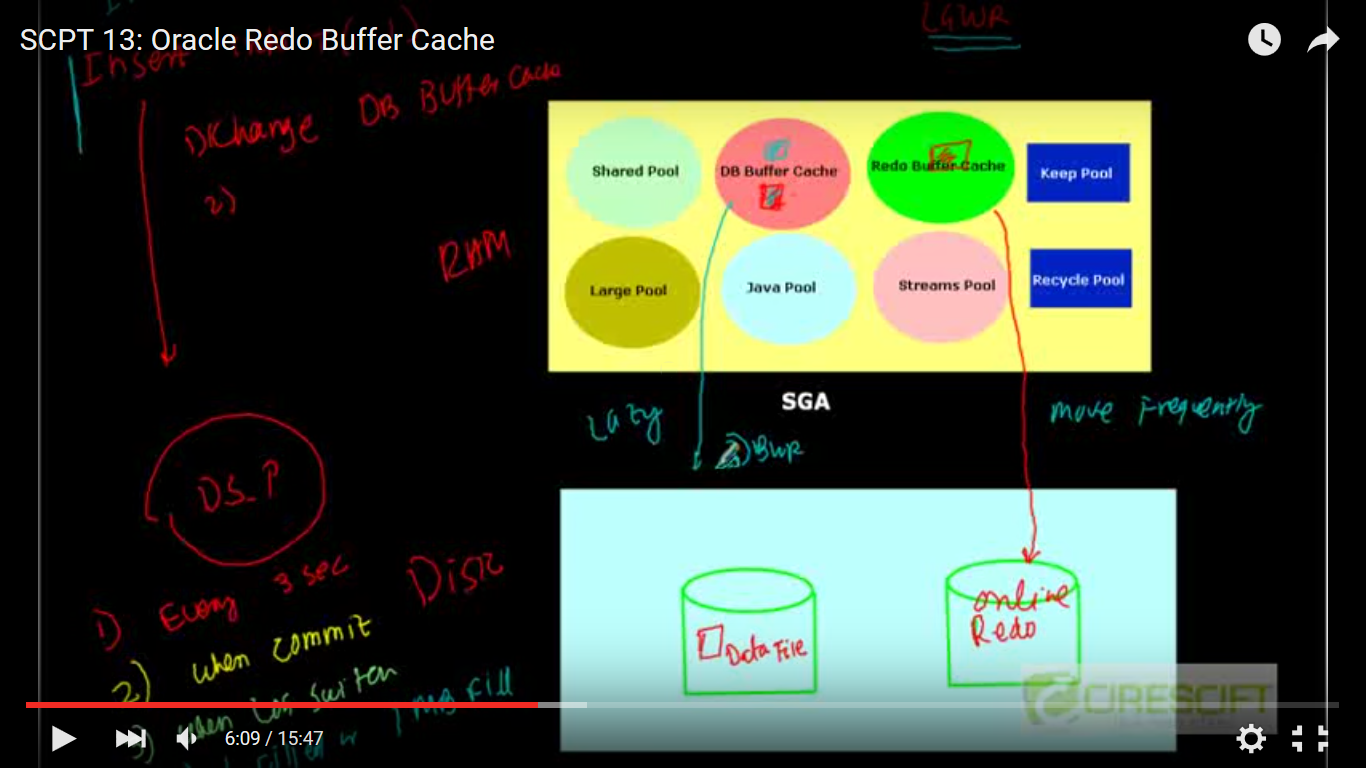


alter system set db\_block\_size=4096 scope=spfile;

Atomicity

Atomicity requires that each transaction is "all or nothing":

if one part of the transaction fails, the entire transaction fails, and the database state is left unchanged.



1. Every 3 sec
2. When commit
3. Log switch
4. 1 MB full

Select substr(‘ORACLE’,level,1) from dual

Connect by level <= length(‘ORACLE’);

**SQL Profile**

A **SQL profile** is a database object that contains auxiliary statistics specific to a SQL statement.