## Definition:

## “Database is collection of all schema objects.”

## “A ****user**** is a database account that allows you to log into the Oracle database.”

## “A schema allows you to logically group objects in an Oracle database.”

## Hierarchy:

## Database management system

## Database server:

## A database server may have many databases. (Usually one)

## A database has many users. (Users categories on basis of role like dba, operators, end users)

## A user has one associated schema.

## (In Oracle one schema is created when we create a user)

## (In PostgreSQL we can create as many schema for a user)

## A schema is a group of logical objects.

## Objects are table, views, sequences, synonyms, indexes, constraints and triggers.

## Note: if a user has special privileges it can access objects in different schema. (Like user as dba can access other objects create by other user but with proper Schema. Objects)

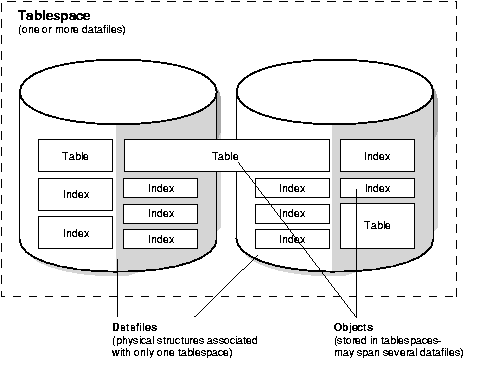
## Connect to the database:

sqlplus username/password@database-name

//sqlplus practice/practice@orcl

**Tablespace:**

Oracle stores data logically in **tablespaces** and physically in **datafiles** associated with the corresponding tablespace.



**Create Tablespace:**

* A **permanent tablespace** contains persistent schema objects. Objects in permanent tablespaces are stored in datafiles.
* An **undo tablespace** is a type of permanent tablespace used by Oracle Database to manage undo data if you are running your database in automatic undo management mode. Oracle strongly recommends that you use automatic undo management mode rather than using rollback segments for undo.
* A **temporary tablespace** contains schema objects only for the duration of a session. Objects in temporary tablespaces are stored in tempfiles.

**Creating a Bigfile Tablespace**

CREATE BIGFILE TABLESPACE bigtbs\_01 DATAFILE 'bigtbs\_f1.dat' SIZE 20M AUTOEXTEND ON;

Tablespace name: Bigtbs\_01

Control file name: 'bigtbs\_f1.dat'

While creating users we can define the Tablespace for those users.

All schema objects created by this user stored to this Tablespace.

**How is the default Tablespace determined when creating a table?**

select \* from database\_properties where property\_name like 'DEFAULT%TABLESPACE';

alter database default tablespace BIGTBS\_01; // its alter database.

alter database default temporary tablespace BIGTBS\_01Temp;

**Create user**

Create user practice identified by practice ;( It’s also create default schema name practice)

CREATE USER username

IDENTIFIED BY password

DEFAULT TABLESPACE tbs\_perm\_01

TEMPORARY TABLESPACE tbs\_temp\_01

QUOTA 20M on tbs\_perm\_01;

**Assign SYSTEM privileges to new user**

GRANT create session TO username;

GRANT create table TO username;

GRANT create view TO username;

GRANT create any trigger TO username;

GRANT create any procedure TO username;

GRANT create sequence TO username;

GRANT create synonym TO username;

**Create Schema**

Use the **CREATE** **SCHEMA** statement to create multiple tables and views and perform multiple grants in your own schema in a single transaction.

This statement does not actually create a schema (as other database provider do). **Oracle Database automatically creates a schema when you create a user.** This statement lets you populate your schema with tables and views and grant privileges on those objects without having to issue multiple SQL statements in multiple transactions.

(User with special privileges can access other user schema object with prefix with schema name to objects)

**Schema Objects:**

**Create table**

Create table employee (emp\_id number(10),emp\_name varchar2(10),hiredate date default sysdate);

**Insert into**

Insert into employee(emp\_id,emp\_name) values(101,’Akash’);

Insert into employee(emp\_id,emp\_name) values(102,’Akash’);

Insert into employee(emp\_id,emp\_name) values(103,’Akash’);

Insert into practice.employee(emp\_id,emp\_name) values(101,’Akash’);

*Note: if we are not providing column name with table name then we need to put all values.*

**Select**

Select \* from employee;

Select emp\_id,emp\_name form employee;

**Create table from query result(or another table)**

Create table emp\_backup as select emp\_id,emp\_name from employee;

Create table emp\_backup as select emp\_id,emp\_name from employee where emp\_id<104;

Create table emp\_backup as select emp\_id e\_id, emp\_name e\_name from employee where emp\_id<104;

Create table emp\_backup(e\_id,e\_name) as select emp\_id,emp\_name from employee where emp\_id<104;

*Note: Not only column type is same as of another table but data is also copied to new table*

**Drop table**

Drop table emp\_backup;

**Delete**

Delete emp\_backup;

**Desc**

Desc emp\_backup;

**Alter table**

* Add a new column
* Modify an existing column
  + Increase or decrease the width of a column
  + Changing the datatype of existing column
  + Changing a default value of a column
  + Changing the null or not null property of a column
* Rename and drop a column
* Rename a table;

Create table emp as Select \* from employee;

Alter table emp add(email varchar(20));

Alter table emp modify(email varchar(10));

*Note:*

1. *Iincrease in size is always allowed*
2. *For decrease in size of numeric column it should be empty*
3. *For decrease in size of char column new width should be greater or equal to all existing values.*
4. *For datatype change column should be empty.*

Alter table emp rename column email to email\_id;

*Note: At a time only one column can be renamed*

Alter table emp drop column email\_id;

Alter table emp drop (emp\_name,hiredate);

Alter table employee remane to employeetb;

**Create view**

Create view emp\_view(emp\_id) as select emp\_id from employeetb;

Select \* from emp\_view;

Drop view emp\_view;

*Note:*

1. *If we insert/delete data from table then it’s also reflected on view data.*
2. *If table is drop no meaning of view exists. View has shown error.*
3. *If we modify/remane the table then also view shows errors.*

**Create Index**

Create index emp\_index1 on employeetb(emp\_name);

Create index emp\_index2 on employeetb(emp\_name,hiredate desc);

Drop index emp\_index1;

*Note:*

1. *Rename a table/or column name have no effect on view*
2. *Modification of column participating in index cannot be happened*

**Create Sequence**

Create sequence employee\_empid\_sequence maxvalue 500 increment by 1 start with 100;

Insert into employeetb(emp\_id,emp\_name) values(employee\_empid\_sequence.nextval,’Rajeev’);

Insert into employeetb(emp\_id,emp\_name) values(employee\_empid\_sequence.nextval,’Nitin’);

Select \* from employeetb;

**Create synonym**

Create synonym e for employeetb;

Select \* from e;

*Note:*

1. *After rename a table error is shown while using synonym “synonym translation is no longer valid”*

**How to List All Tables and Describe Tables in Oracle:**

To list all tables owned by the current user, type:

select tablespace\_name, table\_name from user\_tables;

To list all tables in a database:

select tablespace\_name, table\_name from dba\_tables;

To list all tables accessible to the current user, type:

select tablespace\_name, table\_name from all\_tables;

**Relative link between below entity:**

User: Access purpose

Schema: Grouping purpose

Tablespace: storage purpose.

Database is top entity contains users /schema/Tablespace

# Insert a date/time value into an Oracle table (TO\_DATE )

insert into table\_name (date\_field)values(TO\_DATE(‘2003/05/03 21:02:44’, ‘yyyy/mm/dd hh24:mi:ss));

update table\_name set date\_field=to\_date(‘2012/08/04’,’yyyy/mm/dd’) where date\_field=todate((‘2003/05/03 21:02:44’, ‘yyyy/mm/dd hh24:mi:ss));

insert into table\_name (timestamp\_field)

values (TO\_TIMESTAMP(‘2012/08/04’,’yyyy/mm/dd’));

To use Current Time use “values (CURRENT\_TIMESTAMP)”

**JOINS:**

The JOIN (INNER JOIN) keyword selects all rows from both tables as long as there is a match between the columns in both tables.

The LEFT JOIN (LEFT OUTER JOIN) keyword returns all rows from the left table (table1), with the matching rows in the right table (table2). The result is NULL in the right side when there is no match.

The RIGHT JOIN (RIGHT OUTER JOIN) keyword returns all rows from the right table (table2), with the matching rows in the left table (table1). The result is NULL in the left side when there is no match.

SELECT column name(s) FROM table1 Join Type table2 ON table1.column\_name=table2.column\_name;

## SQL Constraints

SQL constraints are used to specify rules for the data in a table.

Constraints can be specified when the table is created (inside the CREATE TABLE statement) or after the table is created (inside the ALTER TABLE statement).

 In SQL, we have the following constraints:

* **NOT NULL** - Indicates that a column cannot store NULL value
* **UNIQUE** - Ensures that each row for a column must have a unique value
* **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Ensures that a column (or combination of two or more columns) have an unique identity which helps to find a particular record in a table more easily and quickly
* **FOREIGN KEY** - Ensure the referential integrity of the data in one table to match values in another table (A FOREIGN KEY in one table points to a PRIMARY KEY in another table.)
* **CHECK** - Ensures that the value in a column meets a specific condition
* **DEFAULT** - Specifies a default value when specified none for this column

## 2nd highest/smallest salary:Nth

SELECT \*

FROM Employee Emp1

WHERE (N-1) = (

SELECT COUNT (DISTINCT (Emp2.Salary))

FROM Employee Emp2

WHERE Emp2.Salary > Emp1.Salary)

**Using Rank:**

select \* FROM (select EmpID, Salary, rank() over (order by Salary DESC) ranking from Employee) WHERE ranking = N;

**Using row\_number;**

select \* FROM (select newemp.\*, row\_number() over (order by Salary DESC) rownumb from newemp) WHERE rownumb = 2;

select \* FROM (select EmpID, Salary, row\_number() over (order by Salary DESC) rownumb from newemp) WHERE rownumb = 2;