shahidul034 /
Database-Lab-Tutorial

Code Issues Pull requests 2 Actions Projects Security Insights

MIT license

29 stars 36 forks 3 watching Branches Activity
Tags

Public repository

1 Branch 0 Tags Go to file t Go to file + Add file Code

shahidul034 updated readme 648a10a · 2 months ago

DIAGRAM PIC	Add files via upload	11 months ago
Components of DBMS.ppt	Slide added as a study materials	last year
LICENSE	Create LICENSE	2 months ago
Procedure & function.pptx	Add files via upload	last year
README.md	updated readme	2 months ago
Transaction Management.ppt	Slide added as a study materials	last year
ch3.pptx	Add files via upload	last year
ch4.pptx	Add files via upload	last year
complex datatypes.pptx	Slide added as a study materials	last year

README MIT license

Contents

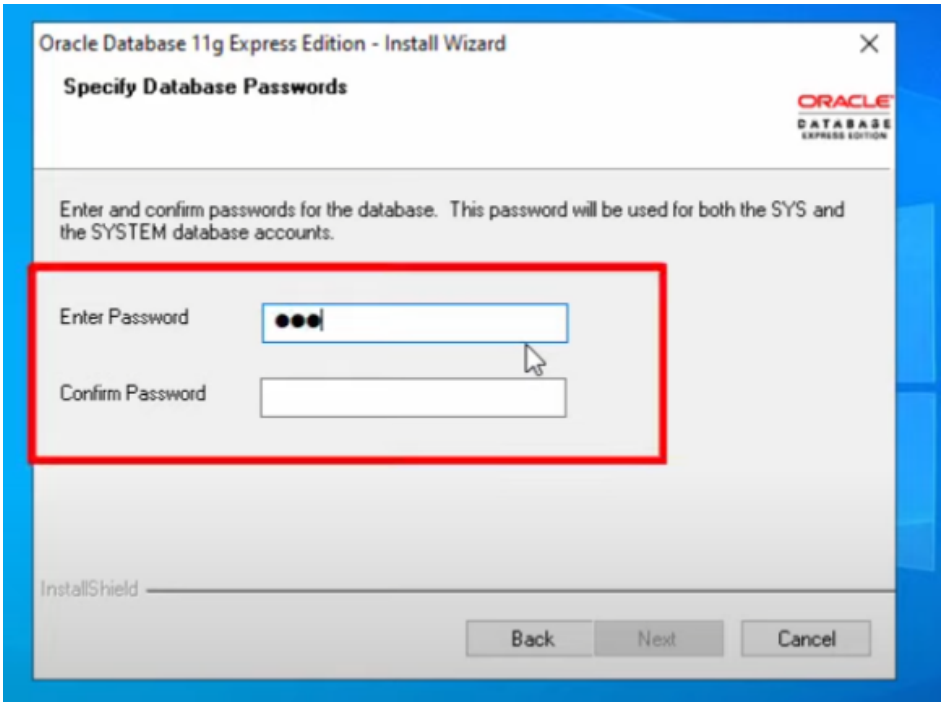
#Lab	Topics name
1	✔ Lab 1 - Installation, creating user and giving system privileges
2	✔ Lab 2 - DDL
3	✔ Lab 3 - DML
4	✔ Lab 4 - Aggregate function, group by, nested subquery, set
5	✔ Lab 5 - String operations
6	✔ Lab 6 - Join, view and referential integrity
7	✔ Lab 7 - PL/SQL (Declaration, insert, set value, rowtype and cursor)
8	✔ Lab 8 - PL/SQL (Loop, array and IF-ELSE)
9	✔ Lab 9 - Procedure and function
10	✔ Lab 10 - Trigger

🎯 Lab 1

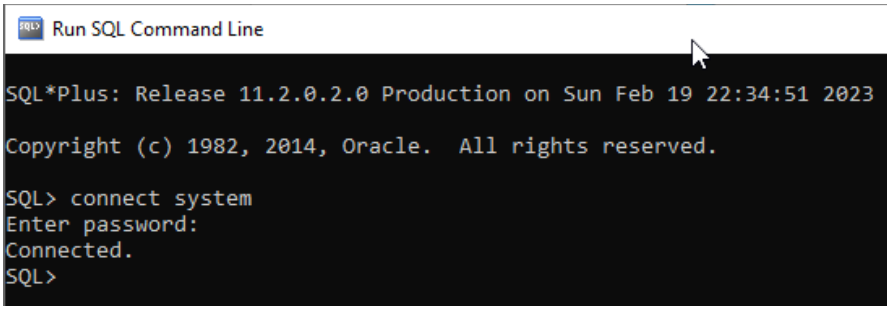
🚀 Installation process

> Download the (Oracle Database 21c Express Edition) or (Oracle Database 11g Express Edition) and install the

> Download the (Oracle Database 21c Express Edition) or (Oracle Database 11g Express Edition) and install the software. Remember the password during the installation because this password is used for connecting the database account.



> Open the SQL Plus. Write 'connect system' and use the password that you set in the installation process. Follow the below figure.



> You can create a new user because we use the system(administrator) as a user. Then, we will give the new user all privileges to perform all SQL tasks. Follow the figure. Change the password of the admin(system)

```
connect / as sysdba
alter user system identified by [password]
```

Creating user and giving system privileges

```
create user shakib034 identified by shakib034;
```

> Delete any user

```
DROP USER username;
```

> Granting system privileges Here, we give all the privileges to the user.

```
grant all privileges to shakib034;
```

> Reboke all the privileges from the user

```
revoke all privileges from user_name
```

```
SQL*Plus: Release 11.2.0.2.0 Production on Sun Feb 19 22:40:56 2023

Copyright (c) 1982, 2014, Oracle.  All rights reserved.

SQL> connect system
Enter password:
Connected.
SQL> create user shakib034 identified by shakib034;

User created.

SQL> grant all privileges to shakib034;

Grant succeeded.

SQL> disconnect
Disconnected from Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
SQL> connect shakib034
Enter password:
Connected.
SQL>
```

> We can give specific privileges to the user.

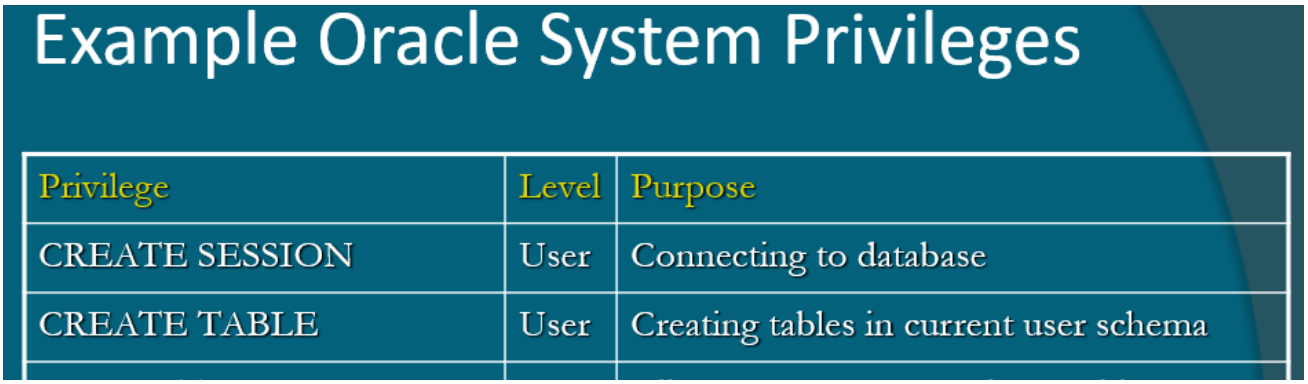
```
grant create session to shakib034;
```

> Creating a role(Assigned system privileges)

```
create role cse2k15;
grant create table, create session to cse2k15;
grant cse2k15 to shakib034;
```

> Revoking System Privileges

```
revoke create table from shakib034;
```



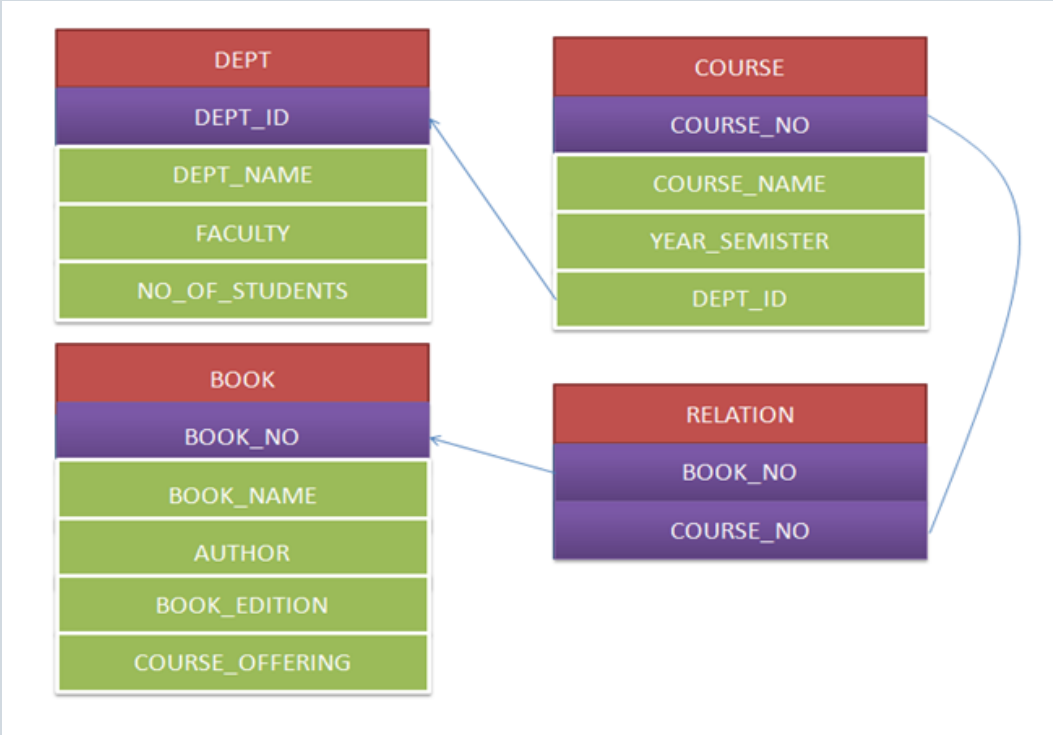
UNLIMITED TABLESPACE	User	Allows user to create schema objects using as much space as needed
CREATE USER	DBA	Creating new users
GRANT ANY PRIVILEGE	DBA	Granting system privileges to users
CREATE ANY TABLE	DBA	Creating tables in any user schema
DROP ANY TABLE	DBA	Dropping tables in any user schema

Set line size and page size

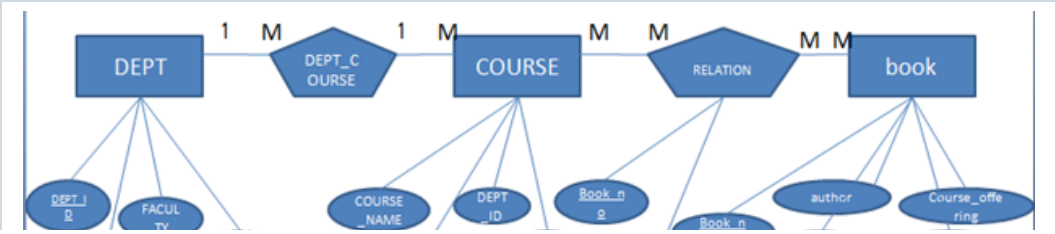
```
show pagesize
show linesize
```

```
set pagesize 100
set linesize 200
```

Database project demo



SCHEMA



ERD

Run the SQL command

> You can run SQL script by using the SQL command line. Besides, you can write your SQL command in a txt file and save the txt file as a SQL extension. Then, type the below command in the SQL command line.

```
start C:\Users\andromeda\Desktop\file.sql
```

Oracle data types

> number(Precision,Scale)

Precision: total number of digits on either side of the decimal point

Scale: number of digits to the right of the decimal point

> char vs varchar2 (varchar2: Variable-length character strings and char: Fixed-length character data)

> Date format: DD-MON-YY

```
create table test(
name varchar2(30),
name2 char(3),
roll number(3),
GPA number(3,2),
roll2 number,
s_dob DATE
);
```

Here is a link about the [oracle data types.]

Checking current user name

```
show user
```

Checking the existing table in database.

```
select table_name from user_tables;
```

🎯 Lab 2

```
SQL> select table_name from user_tables;
```

```
TABLE_NAME
------------------------------
DEPARTMENT
COURSE
```

> It shows that we already have two tables in my database. So we need to drop those tables from the database for a fresh start. You can use this command to drop any table.

```
drop table department;
drop table course;
```

> First, we create the table "dept". Here, we can see that the primary key is dept_id because it uniquely identifies each row in the table.

```
Create table dept(
dept_id number(20),
dept_name varchar(30),
faculty varchar(30),
no_of_student number(20),
primary key(dept_id)
);
```

> To see the last/immediate created table.

```
describe dept
```

> We also create a course, book and relation table.

```
Create table course(
course_no varchar(20),
course_name varchar(50),
year_semister number(3),
credit number(20,4),
dept_id number(20),
primary key(course_no),
foreign key(dept_id) references dept(dept_id)
);
create table book(
book_no number(20),
book_name varchar(50),
author varchar(50),
book_edition number(4),
course_offering number(6),
primary key(book_no)
);
create table relation(
book_no number(20),
course_no varchar(20),
primary key(book_no,course_no),
foreign key (book_no) references book(book_no),
foreign key (course_no) references course(course_no)
```

```
);
```

> When you observe the course table, they have another foreign key keyword referencing the dept_id from the "dept" table. We insert something in the course table. We must check whether this dept_id exists in the "dept" table.

DDL

Add column in the table

> We add a column in the dept table which is location.
>
> Command structure: alter table **table_name** add **column_name column_definition**;

```
alter table dept add location char(20);
```

Modify column definition in the table

> Command structure: alter table **table_name** modify **column_name column_definition**;
>
> We modify the location data types char(20) to varchar(23);

```
alter table dept modify location varchar(23);
```

Rename the column name

> Command structure: alter table **table_name** rename **column_name** to **column_name**;

```
alter table dept rename column location to location2;
```

Drop the column from table

> Command structure: alter table **table_name** drop column **column_name**;

```
alter table dept drop column location2;
```

🎯 Lab 3

DML

Insert the data in our table

```
insert into dept(dept_id,dept_name,faculty,no_of_student)values(7,'CSE','EE',120);
```

> Here, "dept" refers to the table name, and we also insert values according to table columns. We also insert values into the other tables.

Insert the data in the table

```
insert into dept(dept_id,dept_name,faculty,no_of_student)values(3,'EEE','EE',120);
insert into dept(dept_id,dept_name,faculty,no_of_student)values(1,'CE','CE',120);
insert into dept(dept_id,dept_name,faculty,no_of_student)values(5,'ME','ME',120);
insert into dept(dept_id,dept_name,faculty,no_of_student)values(2,'ECE','EE',60);

insert into course(course_no,course_name,year_semister,credit,dept_id)values('CSE1101','discreate
math',11,3.00,7);
insert into course(course_no,course_name,year_semister,credit,dept_id)values('CSE3105','database
systems',31,3.00,7);
insert into course(course_no,course_name,year_semister,credit,dept_id)values('EEE1101','Basic
electrical engineering',11,3.00,3);
insert into course(course_no,course_name,year_semister,credit,dept_id)values('ME3101','solid
mechanics',31,3.00,5);

insert into book(book_no,book_name,author,book_edition,course_offering)values(12,'discreate
math','rosen',4,2);
insert into book(book_no,book_name,author,book_edition,course_offering)values(13,'database
systems','korth',5,1);
insert into
book(book_no,book_name,author,book_edition,course_offering)values(14,'data_communication','willim
stallings',6,3);
insert into book(book_no,book_name,author,book_edition,course_offering)values(15,'solid
mechanics','john abraham',3,2);
insert into book(book_no,book_name,author,book_edition,course_offering)values(16,'electrical
engineering','boylsted',8,4);

insert into relation(book_no,course_no)values(12,'CSE1101');
insert into relation(book_no,course_no)values(16,'EEE1101');
insert into relation(book_no,course_no)values(15,'ME3101');
insert into relation(book_no,course_no)values(13,'CSE3105');
```

Dept

dept_id	dept_name	faculty	no_of_student
7	CSE	EE	120
3	EEE	EE	120
1	CE	CE	120
5	ME	ME	120
2	ECE	EE	60

Course

course_no	course_name	year_semister	credit	dept_id
CSE1101	discreate math	11	3	7
CSE3105	database systems	31	3	7
EEE1101	Basic electrical engineering	11	3	3
ME3101	solid mechanics	31	3	5

Book

book_no	book_name	author	book_edition	course_offering

12	discreate math	rosen	4	2
13	database systems	korth	5	1
14	data_communication	willim stallings	6	3
15	solid mechanics	john abraham	3	2
16	electrical engineering	boylsted	8	4

Relation

book_no	course_no
12	CSE1101
16	EEE1101
15	ME3101
13	CSE3105

> All the data are inserted. Now, we do some experiments here. We run the command below.

```
insert into course(course_no,course_name,year_semister,credit,dept_id)values('MME1101','Basic structure of materials',11,3.00,9);
```

> We inserted the course_no "MME1101" and course_id "9" in the course table. After running the command, we found this error. The course table refers to the dept table, and course id "9" does not exist in the dept table. So, it shows this error.

```
SQL> insert into course(course_no,course_name,year_semister,credit,dept_id)values('M
ME1101','Basic structure of materials',11,3.00,9);
insert into course(course_no,course_name,year_semister,credit,dept_id)values('MME110
1','Basic structure of materials',11,3.00,9)
*
ERROR at line 1:
ORA-02291: integrity constraint (S1507034.SYS_C007228) violated - parent key
not found
```

Displaying table data using SELECT command

> Now, we find the rows from the "dept" table, which have 120 students using the select command.

```
select * from dept where no_of_student=120;
```

```
SQL> select * from dept where no_of_student=120;

   DEPT_ID DEPT_NAME                      FACULTY
---------- ------------------------------ --------
NO_OF_STUDENT
-------------
         7 CSE                            EE
          120

         3 EEE                            EE
          120
```

```
         1 CE                            CE
          120


   DEPT_ID DEPT_NAME                     FACULTY
---------- ------------------------------ --------
NO_OF_STUDENT
-------------
         5 ME                            ME
          120
```

> We find the dept_name, which course name is "database systems".

```
select * from dept where dept_id=(select dept_id from course where course_name='database systems');
```

```
SQL> select dept_name from dept where
  2  dept_id=(select dept_id from course where
  3  course_no=(select course_no from relation where
  4  book_no=(select book_no from book where author='rosen')));

DEPT_NAME
------------------------------
CSE
```

> Here, we add extra select command by adding an additional condition to find the dept_id in the "dept" table. This type of query is called a subquery.

Updating the data in a table

> Now we want to update the value of the course name from the course table where course_no is "EEE1101";
> Command structure: update table_name set column_name=value where condition;

```
update course set course_name='Digital Electronics' where course_no='EEE1101';
```

Deleting row from a table

> We add an extra row to perform the delete operation in the dept table.

```
insert into dept(dept_id,dept_name,faculty,no_of_student)values(12,'URP','CE',60);
```

> Now we delete the row from the dept table where dept_id is 12.

> Command structure: delete from **table_name** where **condition**;

```
delete from  dept where dept_id=12;
```

union, intersect, and except

```
select dept_name from dept where dept_name like 'E%' union select dept_name from dept where
dept_name like '%M%';
```

With clause

Calculates the maximum value of the no_of_student column from the dept table. The result of this subquery is a single row with a single column (val) containing the maximum value.

```
with max_student(val) as (select max(no_of_student) from dept)
select * from dept,max_student where dept.no_of_student=max_student.val;
```

Save the SQL command output

You can save the SQL command in different format.

Save the SQL command output in CSV file

> Simply change the folder path and your sql command.

```
SET COLSEP ","
SET HEADING OFF
SET PAGESIZE 0
SET FEEDBACK OFF
SPOOL C:\Users\andromeda\Desktop\file.csv
SELECT *
FROM dept;
SPOOL OFF
```

Save the SQL command output in txt file

> Simply change the folder path and your sql command.

```
SPOOL C:\Users\andromeda\Desktop\file.txt
SELECT *
FROM dept;
SPOOL OFF
```

🎯 Lab 4

Aggregate function

We count how many row exist in dept table.

```
select count(*) from dept;
```

We also give alias name to any output in select command.

```
select count(dept_name) as number_of_dept from dept;
```

We can count distinct department name in dept table.

```
select count(distinct dept_name) as number_of_dept from dept;
```

We can count average and total no. of students in dept table.

```
select avg(no_of_student) from dept;
select sum(no_of_student) from dept;
```

We can find max and min no. of students of any department from dept table.

```
select max(no_of_student) from dept;
select min(no_of_student) from dept;
```

Group by and Having

> Find the average of student according to faculty.

```
select faculty,avg(no_of_student) from dept group by faculty;
```

```
FACULTY                        AVG(NO_OF_STUDENT)
------------------------------ ------------------
EE                                            100
ME                                            120
CE                                            120
```

> Find the average of student according to faculty where average of student is greater than 60.

```
select faculty,avg(no_of_student) from dept group by faculty having avg(no_of_student)>60;
```

```
SQL> select faculty,avg(no_of_student) from dept group by faculty having avg(no_of_student)>60;

FACULTY                        AVG(NO_OF_STUDENT)
------------------------------ ------------------
EE                                            100
ME                                            120
CE                                            120
```

Nested subquery

> Find the department name where the “rosen”(author) book is taught.

```
 select dept_name from dept where dept_id=(select dept_id from course where course_no=(select
course_no from relation where book_no=(select book_no from book where author='rosen')));
```

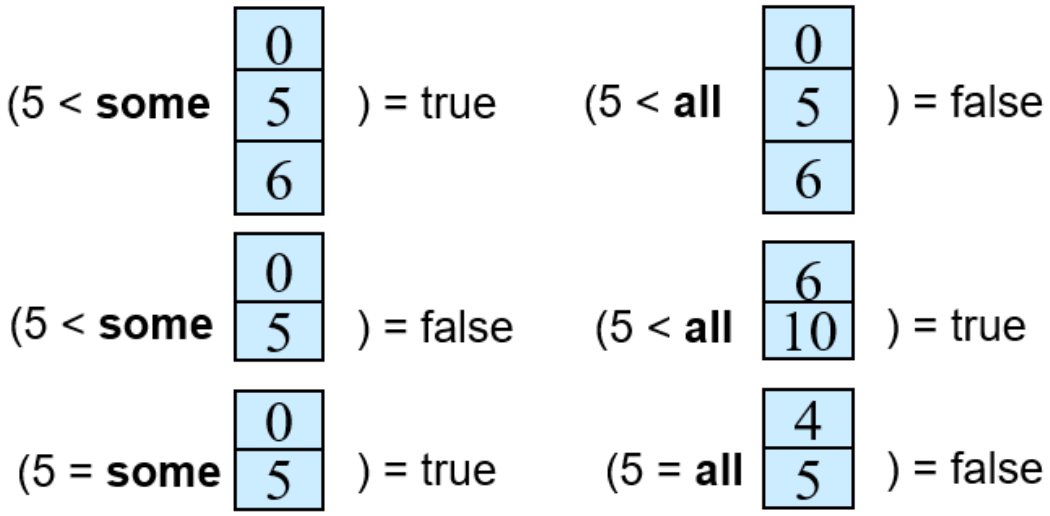
```
SQL> select dept_name from dept where
  2  dept_id=(select dept_id from course where
  3  course_no=(select course_no from relation where
  4  book_no=(select book_no from book where author='rosen')));

DEPT_NAME
------------------------------
CSE
```

Set Membership(AND, OR,NOT)

Find the rows where faculty is "EE" and "CSE" string exists in course_no.

```
select * from dept where faculty='EE' and dept_id in (select dept_id from course where course_no like '%CSE%')
```

some/all/exists/unique




```
select * from book where book_no> some(select book_no from book where book_no>=14);
select * from book where book_no> all(select book_no from book where book_no>=14);
```

The exists construct returns the value true if the argument subquery is nonempty.

```
select * from course where year_semister>=11 and exists(select * from dept where faculty like '%EE%');
```

```
SQL> select * from course where year_semister>=11 and
  2  exists(select * from dept where faculty like '%EE%');

COURSE_NO            COURSE_NAME
-------------------- ------------------------------------
YEAR_SEMISTER      CREDIT     DEPT_ID
```

```
YEAR_SEMISTER     CREDIT    DEPT_ID
------------- ---------- ----------
CSE1101                  discreate math
           11          3          7

CSE3105                  database systems
           31          3          7

EEE1101                  Basic electrical engineering
           11          3          3


COURSE_NO                COURSE_NAME
-------------------- ------------------------------------
YEAR_SEMISTER     CREDIT    DEPT_ID
------------- ---------- ----------
ME3101                   solid mechanics
           31          3          5
```

🎯 Lab 5

String operations

> percent (%). The % character matches any substring.

> underscore (_). The _ character matches any character.

> Demo table.

NAME
EEE
CSE
BME
ECE
MSE
URP
ME
CIVIL
ECEC
EE

> 'E%' matches any string beginning with “E”.

```
SELECT * FROM TEST WHERE NAME LIKE 'E%';
```

NAME
EEE
ECE
ECEC
EE

> '%E' matches any string ending with “E”

> '%E' matches any string ending with 'E'.

```
SELECT * FROM TEST WHERE NAME LIKE '%E';
```

NAME
EEE
CSE
BME
ECE
MSE
ME
EE

> '%E%E%' contains with 'EE'.

```
SELECT * FROM TEST WHERE NAME LIKE '%E%E%';
```

> '_ _ _' matches any string of exactly three characters.

```
SELECT * FROM TEST WHERE NAME LIKE '___';
```

> Below command matches any string of at least three characters and at most five characters.

```
SELECT * FROM TEST WHERE NAME LIKE '___' or NAME LIKE '____' or NAME LIKE '_____';
```

🎯 Lab 6

Join operations

```
select * from dept natural join course where dept_id=7;
```

DEPT_ID	DEPT_NAME	FACULTY	NO_OF_STUDENT	COURSE_NO	COURSE_NAME	YEAR_SEMISTER	CREDIT
7	CSE	EE	120	CSE1101	discreate math	11	3
7	CSE	EE	120	CSE3105	database systems	31	3

```
select * from dept natural join course;
```

DEPT_ID	DEPT_NAME	FACULTY	NO_OF_STUDENT	COURSE_NO	COURSE_NAME	YEAR_SEMISTER	CREDIT
7	CSE	EE	120	CSE1101	discreate math	11	3
7	CSE	EE	120	CSE3105	database systems	31	3
3	EEE	EE	120	EEE1101	electrical engineering	11	3
5	ME	ME	120	ME3101	solid mechanics	31	3

```
select dept_name,course_name from dept join course using(dept_id);
select dept_name,course_name from dept join course on dept.dept_id=course.dept_id;
```

DEPT_NAME	COURSE_NAME
CSE	discreate math
CSE	database systems

	EEE	electrical engineering
	ME	solid mechanics

```
select dept_name,course_name from dept left outer join course using(dept_id);
select dept_name,course_name from dept right outer join course using(dept_id);
select dept_name,course_name from dept full outer join course using(dept_id);
select dept_name,course_name from dept left outer join course on dept.dept_id=course.dept_id;
```

Views

View definition is not the same as creating a new relation by evaluating the query expression.

A view of dept without their faculty,no_of_student.

```
create view dept_details as select dept_id,dept_name from dept;
```

Find all course in the CSE department.

```
create view CSE_DEPT_COURSE as select course_name from course where dept_id=(select dept_id from
dept where dept_name='CSE');
```

Views Defined Using Other Views

```
create view  custom as select * from dept_details where dept_id>=3;
```

Cascading Actions in Referential Integrity

```
Create table dept2(
dept_id number(20),
dept_name varchar(30),
faculty varchar(30),
no_of_student number(20),
primary key(dept_id)
);
Create table course2(
course_no varchar(20),
course_name varchar(50),
year_semister number(3),
credit number(20,4),
dept_id number(20),
primary key(course_no),
foreign key(dept_id) references dept2(dept_id)
on delete cascade
);
insert into dept2(dept_id,dept_name,faculty,no_of_student)values(7,'CSE','EE',120);
insert into dept2(dept_id,dept_name,faculty,no_of_student)values(3,'EEE','EE',120);
insert into dept2(dept_id,dept_name,faculty,no_of_student)values(1,'CE','CE',120);
insert into dept2(dept_id,dept_name,faculty,no_of_student)values(5,'ME','ME',120);
insert into dept2(dept_id,dept_name,faculty,no_of_student)values(2,'ECE','EE',60);

insert into
```

```
course2(course_no,course_name,year_semister,credit,dept_id)values('CSE1101','discreate
math',11,3.00,7);
insert into course2(course_no,course_name,year_semister,credit,dept_id)values('CSE3105','database
systems',31,3.00,7);
insert into course2(course_no,course_name,year_semister,credit,dept_id)values('EEE1101','Basic
electrical engineering',11,3.00,3);
insert into course2(course_no,course_name,year_semister,credit,dept_id)values('ME3101','solid
mechanics',31,3.00,5);


delete from dept2 where dept_id=5;
```

Constraints on a Single Relation

```
CREATE TABLE my_table (
    id INTEGER PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    age INTEGER CHECK (age >= 18)
);
CREATE TABLE my_table2 (
    id INTEGER PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    age INTEGER CHECK (age >= 18 AND age <= 120),
    status VARCHAR(10) CHECK (status IN ('active', 'inactive', 'pending')),
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    CONSTRAINT check_age_status CHECK (
        (status = 'active' AND age >= 18 AND age <= 65) OR
        (status = 'inactive' AND age >= 18 AND age <= 120) OR
        (status = 'pending' AND age >= 18 AND age <= 100) OR
        (end_date > start_date)
    )
);
```

> Insert data

```
insert into my_table values(1,'ss','ss@gmail.com',25);
insert into my_table2 values(3,'ss','ss@gmail.com',25,'active','03-APR-2007','04-APR-2009');
```

🎯 Lab 7

PL/SQL

Toad for oracle download

Download the (Toad) and install the software.

PL/SQL variable declaration and print value

```
set serveroutput on
```

```
set serveroutput on
declare
dept_id dept.dept_id%type;
dept_name DEPT.DEPT_NAME%type;
no_of_student number;
begin
select dept_id,dept_name,no_of_student into dept_id,dept_name,no_of_student from dept where
dept_id=7;
dbms_output.put_line('DEPT_id: '||dept_id|| ' DEPT_name: '||dept_name || ' no_of_student: '||
no_of_student);
end;
/
```

Insert and set default value

```
set serveroutput on
declare
dept_id dept.dept_id%type:=9;
dept_name DEPT.DEPT_NAME%type:='MME';
faculty dept.faculty%type:='ME';
no_of_student number:=30;
begin
insert into dept values(dept_id,dept_name,faculty,no_of_student);
end;
/
```

Row type

```
set serveroutput on
declare
dept_row dept%rowtype;
begin
select dept_id,dept_name,no_of_student into
dept_row.dept_id,dept_row.dept_name,dept_row.no_of_student from dept where dept_id=7;
end;
/
```

Cursor and row count

```
set serveroutput on
declare
cursor dept_cursor is select * from dept;
dept_row dept%rowtype;
begin
open dept_cursor;
fetch dept_cursor into
dept_row.dept_id,dept_row.dept_name,dept_row.faculty,dept_row.no_of_student;
while dept_cursor%found loop
dbms_output.put_line('DEPT_id: '||dept_row.dept_id|| ' DEPT_name: '||dept_row.dept_name || '
faculty: ' ||dept_row.faculty|| ' no_of_student: '||dept_row.no_of_student);
dbms_output.put_line('Row count: '|| dept_cursor%rowcount);
fetch dept_cursor into
dept_row.dept_id,dept_row.dept_name,dept_row.faculty,dept_row.no_of_student;
end loop;
close dept_cursor;
```

```
end;
/
```

🎯 Lab 8

FOR LOOP/WHILE LOOP/ARRAY with extend() function

```
TYPE NAMEARRAY IS VARRAY(5) OF book.book_name%type;
```

> It defines a user-defined collection type named NAMEARRAY as a variable array of 5 elements, with each element being of the same data type as the book_name column in the book table.

```
A_NAME NAMEARRAY:=NAMEARRAY();
```

> It initializes a variable A_NAME of type NAMEARRAY.

```
set serveroutput on
declare
  counter number;
  book_name2 book.book_name%type;
  TYPE NAMEARRAY IS VARRAY(5) OF book.book_name%type;
  A_NAME NAMEARRAY:=NAMEARRAY();
begin
  counter:=1;
  for x in 12..16
  loop
    select book_name into book_name2 from book where book_no=x;
    A_NAME.EXTEND();
    A_NAME(counter):=book_name2;
    counter:=counter+1;
  end loop;
  counter:=1;
  WHILE counter<=A_NAME.COUNT
    LOOP
    DBMS_OUTPUT.PUT_LINE(A_NAME(counter));
    counter:=counter+1;
  END LOOP;
end;
/
```

ARRAY without extend() function

```
DECLARE
   counter NUMBER := 1;
   book_name2 book.book_name%TYPE;
   TYPE NAMEARRAY IS VARRAY(5) OF book.book_name%TYPE;
   A_NAME NAMEARRAY:=NAMEARRAY('Book 1', 'Book 2', 'Book 3', 'Book 4', 'Book 5');
   -- VARRAY with a fixed size of 5 elements and initialized with book names
BEGIN
   counter := 1;
   FOR x IN 12..16
   LOOP
```

```
      SELECT book_name INTO book_name2 FROM book WHERE book_no=x;
      A_NAME(counter) := book_name2;
      counter := counter + 1;
   END LOOP;
   counter := 1;
   WHILE counter <= A_NAME.COUNT
   LOOP
      DBMS_OUTPUT.PUT_LINE(A_NAME(counter));
      counter := counter + 1;
   END LOOP;
END;
```

ARRAY with extend() and without extend() function

The A_NAME.EXTEND() method is used to dynamically increase the size of the varray A_NAME at runtime by one element. When you use A_NAME.EXTEND(), you don't need to know the size of the varray in advance, and you can add elements to the varray as needed.

On the other hand, if you declare the varray with a fixed size and initialize it with values at the beginning, you can assign values to the elements of the varray using the varray_name(index) := value syntax, as shown in the previous example. However, if you try to assign a value to an index that is beyond the fixed size of the varray, you will get a PLS-00302: component 'index' must be declared error.

So the main difference between using and not using A_NAME.EXTEND() is that with A_NAME.EXTEND(), you can dynamically resize the varray as needed, while without it, the varray has a fixed size and you cannot add more elements than the declared size.

IF /ELSEIF /ELSE

```
DECLARE
   counter NUMBER := 1;
   book_name2 book.book_name%TYPE;
   TYPE NAMEARRAY IS VARRAY(5) OF book.book_name%TYPE;
   A_NAME NAMEARRAY:=NAMEARRAY('Book 1', 'Book 2', 'Book 3', 'Book 4', 'Book 5');
   -- VARRAY with a fixed size of 5 elements and initialized with book names
BEGIN
   counter := 1;
   FOR x IN 12..16
   LOOP
      SELECT book_name INTO book_name2 FROM book WHERE book_no=x;
      if book_name2='discreate math'
        then
        dbms_output.put_line(book_name2||' is a '||'CSE course');
      elsif book_name2='electrical engineering'
        then
        dbms_output.put_line(book_name2||' is a '||'EEE course');
      else
        dbms_output.put_line(book_name2||' is a '||'other dept course');
        end if;
   END LOOP;
END;
```

🎯 Lab 9

Procedure

```
CREATE OR REPLACE PROCEDURE proc2(
  var1 IN NUMBER,
  var2 OUT VARCHAR2,
  var3 IN OUT NUMBER
)
AS
  t_show CHAR(30);
BEGIN
  t_show := 'From procedure: ';
  SELECT course_name INTO var2 FROM course WHERE course_no IN (SELECT course_no FROM relation
WHERE book_no = var1);
  var3 := var1 + 1;
  DBMS_OUTPUT.PUT_LINE(t_show || var2 || ' code is ' || var1 || ' In out parameter: ' || var3);
END;
/
```

```
set serveroutput on
declare
book_no book.book_no%type:=12;
course_name course.course_name%type;
extra number;
begin
proc2(book_no,course_name,extra);
end;
/
```

Function

```
set serveroutput on
create or replace function fun(var1 in varchar) return varchar AS
value dept.dept_name%type;
begin
  select dept_name into value from dept where dept_id=var1;
   return value;
end;
/
```

```
set serveroutput on
declare
value varchar(20);
begin
value:=fun(5);
end;
/
```

> A function must return a value and procedure cannot return a value.

drop procedure and function

> Stucture: drop procedure procedure_name Stucture: drop function function_name

```
drop procedure proc2;
drop function fun;
```

🎯 Lab 10

Trigger

Trigger body

```
create trigger [trigger_name]
[before | after]
{insert | update | delete}
on [table_name]
[for each row]
[trigger_body]
```

> This trigger is named "try" and is set to execute before each row is deleted from the "relation" table The REFERENCING OLD AS o NEW AS n clause specifies that the trigger will reference the "old" values (i.e. the values before the deletion) as "o" and the "new" values (which do not exist in this case because it's a delete trigger) as "n".

> Inside the trigger, there are two delete statements. The first statement deletes any rows from the "book" table where the "book_no" matches the value of "o.book_no" (i.e. the book number of the row being deleted from "relation"). The second statement deletes any rows from the "course" table where the "course_no" matches the value of "o.course_no" (i.e. the course number of the row being deleted from "relation").

```
SET SERVEROUTPUT ON
CREATE OR REPLACE TRIGGER try
BEFORE delete ON relation
REFERENCING OLD AS o NEW AS n
FOR EACH ROW
BEGIN
delete from book where book_no=:o.book_no;
delete from course where course_no=:o.course_no;
END;
/
```

```
delete from relation where book_no=16;
```

> The trigger is set to fire after an update operation is performed on the "course" table. For each row being updated, the trigger fires and updates the "book_name" column in the "book" table with the new course name (:n.course_name) where the "book_no" is present in the "relation" table for the corresponding old course number (:o.course_no).

```
SET SERVEROUTPUT ON
CREATE OR REPLACE TRIGGER trigger2
after update ON course
REFERENCING OLD AS o NEW AS n
```

```
REFERENCING OLD AS o NEW AS n
FOR EACH ROW
Enable
BEGIN
update book set book_name=:n.course_name where book_no in (select book_no from relation where
course_no=:o.course_no);
END;
/
```

```
update course set course_name='discrete math2' where course_no='CSE1101';
```

> This trigger is designed to execute after an insertion occurs in the "relation" table. It will update a related "book" table by incrementing the value of the "course_offering" field by 1, where the "book_no" matches the "book_no" of the newly inserted row.

> The ":n.book_no" syntax refers to the "book_no" field in the newly inserted row (referenced as "n"). The "update" statement will modify the "book" table by incrementing the "course_offering" field by 1 for the row where the "book_no" matches the "book_no" of the newly inserted row.

```
SET SERVEROUTPUT ON
CREATE OR REPLACE TRIGGER trigger_new
after insert ON relation
REFERENCING OLD AS o NEW AS n
FOR EACH ROW
Declare
bok integer;
BEGIN
update book set course_offering=course_offering+1 where book_no=:n.book_no;
END;
/
```

```
show errors;
select * from user_triggers;
drop trigger TRIGGER_NEW;
```

Reference

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2023). Database System Concepts (7th ed.).

Releases

No releases published

Packages

No packages published

Contributors 2


shahidul034 Shahidul Shakib


tashib11 Md Tashibul Islam