Friday 3/22

DAO design pattern for project?

https://www.tutorialspoint.com/design\_pattern/factory\_pattern.htm

factory method pattern

<https://www.tutorialspoint.com/design_pattern/factory_pattern.htm>

Paypal factorymethoddemo java

-create employee, increment id, never reuse id

-if applications close, count goes back to zero if static variable initialized to 0

auto increment SQL

create sequence empIdGenerator

start with 101

increment by 1

--maxvalue 10000000

no cycle/ cycle will cycle again

create sequence empIdGenerator

start with 101

increment by 1

nocycle;

//you cannot add the sequence after using “alter”

insert into emp values (empIdGenerator.nextval, ename)

//select emIdGenerator.curval from dual // will give back current value

dual used for output

select 2\* 5 from dual

select sysdate dual;

--select the emp with max salary

--select max(salary) from product;

--anonymous block

<http://www.oracletutorial.com/plsql-tutorial/plsql-anonymous-block/>

declare

empid number(10) :=101;

begin //ned to always

select ename from emp where empid = empid

exception //catch any exceptions

set serveroutput on;

--anonymous block

declare

num1 number := 100;

num2 number :=200;

being

debms\_output.put\_line(num1+num2);

end;

delcare

empId number := 101;

ename varchar2(100);

begin

select name into ename from emp where empno= empId;

dbms\_out.putline(‘employee name is’ || enam)’

end;

//the variables and the data type needs to match

--insert a new emp record into db and id is auto generated

-- after insert and then return id

anonymous block you can put multiple statements in in the begin block

create exception and catch exception and have a statement

declare

esal number (7,2);

empIdException exception;

begin

select sal insto esal from emp where ename= “Jim”

if esal < 1000 then

raise empIdexception;

end if;

exception

when empIdException then

update emp set sal= sal+500 where ename= ‘Jim’

end;

//name

//constant does not allow manipulation of that variable, similar to “final”

declare

name constant varchar2(100)= ‘Jim;

esal number (7,2) := %esal;//ask input from user

job not null default ‘Training’ //if job is not then default training value

hasPassport boolean not null := True

empIdException exception;

begin

select sal into esal ,job into job from into esal from emp where ename= name

if esal < 1000 then

raise empIdexception;

end if;

exception

when empIdException then

update emp set sal= sal+500 where ename= ‘Jim’

end;

use “%” to ask for user input

default keyword, assign a value to default to use default

https://ramkedem.com/en/sql-server-create-table/

for project, only do junit testing for one class not whole project

anonymous blocks cannot return anything and cannot be stored

function block can only return exactly ONE value

procedure block can return multiple values or no values at all

procedure block

create or replace procedure proc1

create procedure proc1

is

begin

update emp set emp.sal = emp.sal + 500 where emp.sal<1000;

end proc1;

exec proc1;

create or replace procedure proc1(x in number, y in number, z out number) //y is return value

is

begin

if x > y then

z :=x;

else

z:=y;

end if;

end;

declare

z number;

begin

proc1(2,3,z);

dbms\_output.put\_line(z);

end;

//cannot use execute if argument if an output type

in class

set serveroutput on;

create or replace procedure proc1(x in number, y in number, z out number)

is

begin

if x > y then

z :=x;

else

z:=y;

end if;

end;

declare

z number;

begin

proc1(2,10,z);

dbms\_output.put\_line(z);

end;

q) --emp: sal<1000 => increment the salary by 500Rs

--sal is blank: raise the exception: salar missing for emp

--exception: emp\_audit table: name sal is null

--create emp\_audit : name is blank

create or replace procedure proc1(empNo number)

is

eSal emp.sal%type;

salmissingException exception;

ename emp.ename%type;

begin

select ename into name,sal into eSal from emp where empId=empNo;

if eSal is null then

raise salmissingException;

elsif eSal < 1000 then

update emp set sal= sal+500 where sal<1000;

end if;

exception

when salmissingException then

insert into emp\_audit values (name);

end proc1;

create or replace procedure proc1(empNo number)

is

eRecord emp%rowtype //represent the whole row query

eSal emp.sal%type;

salmissingException exception;

ename emp.ename%type;

begin

select \* into eRecord from emp where empId=empNo;

if eSal is null then

raise salmissingException;

elsif eSal < 1000 then

update emp set sal= sal+500 where sal<1000;

end if;

exception

when salmissingException then

insert into emp\_audit values (name);

end proc1;

ename : eRecord.ename;

create or replace function func1

return number is

empId number(2) :54;

begin

select sal into salary from emp where empNo= empId;

return salary

end;

declare

sal number;

begin

sal= func1();

dbms\_outout.put\_line(sal);

end;

rowtype captures rows,

record type captures columns

set serveroutput on;

declare

type empDetail is record

(

ename emp.ename%type,

sal emp.sal%type

);

e1 empDetail ;

begin

e1.ename := 'JIM';

e1.sal := 5000;

dbms\_output.put\_line(e1.name);

end;

//constant needs to be a set a value

DECLARE

V\_Sample1 NUMBER(2);

V\_Sample2 CONSTANT NUMBER(2) ;

V\_Sample3 NUMBER(2) NOT NULL ;

V\_Sample4 NUMBER(2) := 50;

V\_Sample5 NUMBER(2) DEFAULT 25;

//labeling blocks and calling variables based on the block

<<outer>>

DECLARE --outer block

var\_num1 NUMBER := 5;

BEGIN

<<inneri>>

DECLARE --inner block

var\_num1 NUMBER := 10;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Value for var\_num1:' ||inneri.var\_num1);

--Can outer block variable (var\_num1) be printed here.IfYes,Print the same.

END;

DBMS\_OUTPUT.PUT\_LINE('Value for var\_num1:' ||var\_num1);

--Can inner block variable(var\_num1) be printed here.IfYes,Print the same.

END;

q)

create table employee

(

empId number(10) primary key,

ename varchar2(100) not null,

sal number(20,2) check (sal>10000),

projectName varchar2(100),

noOfEmployees number(10),

mgrId number(10),

etype char(1)

);

insert into employee values(101, 'Amit', 1000000, 'Training', 12, 201,'M')

insert into employee (empId, ename, sal, etype) values (11, 'kirsti', 100000, 'E');

select ename from employee where mgrId = (

select mgrId from employee where ename = 'Amit');

for inner join

q)

select empid from employee address where city = city

SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate

FROM Orders

INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;

emp1

empid, name, address, city

emp2

empid, name, address, city

select emp1.name,emp2.name, emp1.city from emp1,emp2 where emp1.city =emp2.city and emp1.empid<>emp2.empid

if just one table for self join

//must have alias to reference two rows

select emp1.name,emp2.name, emp1.city from emp emp1, emp emp2 where emp1.city =emp2.city and emp1.empid<>emp2.empid

emp: id, name, mgrid

101 amit 102

11 krist 101

12 puja 101

select name with manager so expect output is

kristi amit

puja ami

amit amit’s manager

select emp1.name, emp2.name from emp emp1, emp emp2 where emp1.mgrid = emp2.id

//that is showing employee first then the manager

//flip if you want manager first then employee

view

join on more than 1 table where...groupby...having...order by…

select mgrid, count(empid) from employee group by mgrid;

emid name, mgrid

mgrid count

101 12

102 10

emid name, mgrid sal

1 K 101 5k

2 P 101 10k

3 A 102 20k

select name, max(salary) from employee where salary = (

select max(salary) from employee;

q) --select 5 records from emp

select \* from employees where empId in (

select empId from emp where rownum <=5 order by empId desc)

--find all employees whose names are either amit or kirti

select \* from employee where lower(ename) in (‘amit”, “kirti”)

--find employees with names other than amit and kirti

select \* from employee where lower(ename) not in (‘amit”, “kirti”)

--display employees which have age greater >40 and name ascending order

select \* from employee where age >40 order by name asc

select \* from employee where age >40 order by name asc

group by age

order by age

when using group by can only use that column in also the select column unless aggregate functions

**views** save your query

create or replace view view1 as

select\* from view1; // view is like a virtual table which stores subsets data in some other tables...It does not save the data itself

CREATE VIEW [Products Above Average Price] AS

SELECT ProductName, Price

FROM Products

WHERE Price > (SELECT AVG(Price) FROM Products);

SELECT \* FROM [Products Above Average Price];

drop viewname, functionname, proceduresname

materialized view will save data, only want data immediately, not while it is updated

<http://www.java67.com/2012/11/what-is-difference-between-view-vs-materialized-view-database-sql.html?m=1>

order by column1, column2 //this will organize asc column1 then column2, reminder it’s asc implicitly

3:15 PM airplane reservation

make service static but only need one object anyways,

throw the exception in DButil

best practices: project should no warning

serialVersionUID prevents changes from classes affect the original object when deserializes, some variables do not exist anymore

//therefore save version

check serialization demo

use logger to log into a file and not a console, you may not always have a console

different loggers

slf4j: specification (low level api)

log4j must add on dependency from maven Apache log4j 1.2.17

log4j2/logback

“Logger” class

.getLogger factor method and pass it into the class

Levels

All

Trace

Debug: logger.debug(“”); debugging (Production: disable)

Info: logger.info(“”); //productions

Error: logger.error(“”)(“Exception occured for flight id” + flightid +e.getMessage());

Fatal: logger.fatal()

create log4j.properties to log types of levels

//looks for this files by default

log4j.rootLogger=”level”=set to INFO

log4j.rootLogger=INFO, console // file

appender.ConsoleAppender

appender.FIleAppender

for pattern

<https://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/PatternLayout.html>

skype payal\_bnsl

flightreservation

flight repository file

“prepared statements” for better performance and also prevent SQL injection

CallableStatement call procedures from sql

for both type of statements “settype” of value to restrict variable type

https://www.tutorialspoint.com/jdbc/jdbc-statements.htm