create trigger delete\_emp before delete on emp

for each row

insert into emp\_history values(OLD.empno,OLD.ename,OLD.sal,null,current\_user(),now(),’delete’);

create trigger update\_emp before update on emp

for each row

insert into emp\_history values(OLD.empno,OLD.ename,OLD.sal,NEW.sal,current\_user(),now(),’update’);

---------exception handling

declare exit handler for 1062 select ‘duplicate value’

declare continue handler for 1062 select ‘error occurred’

declare continue handler for NOT FOUND set cset=1

declare exit handler for SQLEXCEPTION select ‘error occurred’

create procedure myproc1(in pno int,out pnm varchar(20),out pjob varchar(20))

begin

declare exit handle for SQLWarning select “warning msg”

select empno,ename into pno,pnm

from emp

where empno=pno;

end//

NOSQL ---MongoDB

{

empid:123,

ename:'kishori',

skills:['java','python','mongodb','spring boot','hibernate'],

joining\_dt: ISODate('2000-04-27'),

dept:{deptno:11,dname:'hr',dloc:'pune'},

experience:[{name:'hsbc',years:3},{name:'igate',years:4},{name:'Capgemini',years:5}],

maritalStatus:null

}

“dept.dname”

“experience.1.name” igate

“skills.3” 'spring boot'

skills

CRUD ----- create , read, update, delete

db.createCollection(“mycoll111”)

db.mycoll.insert({name:’xxx’,desg:’yyy’})

-----capped collection

---- restrict number of document

----- you cannot delete data from capped collction

blog----5 years

comments ------- lots comments

latest 100 comments

db.createCollection(“mycoll111”,{capped :true,max:2,size:40000 })

RDBMS indexes for primary key gets created automatically

db.mycoll111.insert({name:”Rajan”,comment:”good blog”})

db.mycoll111.insert({name:”Revati”,comment:”excellent”})

db.mycoll111.insert({name:”Rajesh”,comment:”good blog”})

--------------- Read data

db.emp.find()------to find all record select \* from emp;

db.emp.findOne() ----- to find one record select \* from emp limit 1

db.movie.find({},{name:1,rating:1,\_id:0}).sort({rating:-1,name:1}).limit(1).skip(3).pretty();

db.movie.find({},{ticket\_no:0,\_id:0}).sort({rating:-1,name:1}).limit(1).skip(3).pretty();

Query --🡪operators

$in,$nin,$eq,$ne,$gt,$lt,$and,$or,$not,$exists,$size,$mod,$elemMatch,$gte,$lte

--------list all the movies with name=’padmavat’

db.movie.find({name:'padmavat'}).pretty();

-----to list all movies with rating = 3

> db.movie.find({rating:3},{name:1,rating:1,price:1,\_id:0}).sort({price:1})

------to find all movies with rating=3 and price=260

rating:3

price:260

db.movie.find({ rating:3, price:260},{name:1,rating:1,ticke\_no:1,\_id:0})

$and:[{ rating:3},{ price:260}]

db.movie.find({ $and:[{ rating:3},{ price:260}]},{name:1,rating:1,ticke\_no:1,\_id:0})

------to find all movies with price=200

db.movie.find({price:200},{name:1,rating:1,price:1}).pretty()

-------to find all movies with price >200

db.movie.find({price:{$gte:200}},{name:1,rating:1,price:1}).pretty()

-------to find movie with rating < 5

> db.movie.find({rating:{$lt:5}},{name:1,rating:1,price:1}).pretty()

------to find all movies with rating < 5 and price > 300

rating:{$lt:5}

price:{$gt:300}

db.movie.find({ rating:{$lt:5}, price:{$gt:300}})

$and:[{ rating:{$lt:5} },{ price:{$gt:300}}]

db.movie.find({ $and:[{ rating:{$lt:5} },{ price:{$gt:300}}]})

------to find all movies with rating < 5 or price > 300

rating:{$lt:5}

price:{$gt:300}

$or:[{ rating:{$lt:5}},{ price:{$gt:300}}]

db.movie.find({ $or:[{ rating:{$lt:5} },{ price:{$gt:300}}]})

-----list all movies with price =260 or 300 or 450

price:260

price:300

price:450

$or:[{ price:260},{ price:300},{ price:450}]

same as

price:{$in:[260,300,450]}

db.movie.find({price:{$in:[260,300,450]}})

-----list all movies with price not equal 260 or 300 or 450

price:{$nin:[260,300,450]}

db.movie.find({price:{$nin:[260,300,450]}})

-----to list all movies with price = either 260 or 300 or 450

and rating> 3

price:{$in:[260,300,450]}

rating:{$gt:3}

$and:[{ price:{$in:[260,300,450]}},{ rating:{$gt:3}}]

db.movie.find({$and:[{ price:{$in:[260,300,450]}},{ rating:{$gt:3}}]})

or

db.movie.find({ price:{$in:[260,300,450]}, rating:{$gt:3}})

--------to find all movies which has rating key and the value is null

db.movie.find({rating:{$in:[null],$exists:true}})

------------to find all movies which do not have rating key

db.movie.find({rating:{$exists:false}})