before statement level ---- not available in mysql

before row level trigger on delete

delete statement

after row level trigger on delete

after statement level ------ not available in mysql

procedure----- doesnot return a value

procedure cannot be called in select statement

in , out, inout type parameters can be used in procedure

functions ---return a value

functions can be called in select statement

or can be used in procedure

in , out, inout type parameters can be used in procedure

|  |  |
| --- | --- |
| SQL database | NoSQL |
| Structured | Unstructured |
| Transactions are supported so more secure | No transaction support so less secure |
| Vertical scaling | Horizontal scaling |
| Centralized data | Distributed data |
| Data retrieval is slower as compared NOSQL | Data retrieval is faster as compared to SQL when data is large |
| Cost involved is more | Cost involved is less |
| Data stored in table format | Data stored in JSON format |
| Availability is less because if server is down retrieval of data may become a proble | Availability is high because of replica |

Sharding---- Data is stored in distributed manner on multiple nodes is called as sharding

------find all movies with name kahani

db.movie.find({name:'kahani'},{ticket\_num:0,\_id:0}).pretty();

----find all movies with rating =3

db.movie.find({rating:3})

----to find all movies with rating=4 and price =250

db.movie.find({rating:4,price:250}).pretty()

-----find all movies with rating > 3

$gt,$lt,$eq,$ne,$lte,$gte

db.movie.find({rating:{$gt:3}},{name:1,price:1,rating:1})

------to display all movies with reating>3 and price>250

> db.movie.find({rating:{$gt:3},price:{$gt:250}},{name:1,price:1,rating:1}).pretty();

------find all movies with name kahani and rating >4

db.movie.find({name:’kahani’,rating:{$gt:4}},{name:1,rating:1,price:1})

---- remove document from collection

db.movie.remove(); ----- this removes all documents

db.movie.remove({name:’kahani’})

----to modify the document in mongoDB

use $unset to remove a key -value pair from document

db.movie.update({name;’kahani’},{$unset:{name:””,price:””}},{multi:true})

-------to modify rating to 4

db.movie.update({name;’kahani’},{$set:{name:”Kahani”,price:200}},{multi:true})

-------to find all movies with rating >=3 and <5

db.movie.find({rating:{$gte:3,$lt:5}},{name:1,rating:1,price:1,\_id:0})

------to find all movies with rating >=3 and <5 and price>250

rating:{$gte:3,$lt:5}

price:{$gt:250}

db.movie.find({rating:{$gte:3,$lt:5} , price:{$gt:250}},{name:1,rating:1,price:1})

------to find all movies with rating >=3 and <5 or price>250

rating:{$gte:3,$lt:5}

price:{$gt:250}

$or:[{ rating:{$gte:3,$lt:5}},{ price:{$gt:250}}]

db.movie.find({$or:[{ rating:{$gte:3,$lt:5}},{ price:{$gt:250}}]})

------ find all movies with rating <5 or price >200 or ticket\_num>150

rating:{$lt:5}

price:{$gt:200}

ticket\_num:{$gt:150}

$or:[{ rating:{$lt:5}},{ price:{$gt:200}},{ ticket\_num:{$gt:150} }]

db.movie.find({$or:[{ rating:{$lt:5}},{ price:{$gt:200}},{ ticket\_num:{$gt:150} }]})

------- find all movies with rating=3 or 4 or 5

rating:{$in:[3,4,5]}

db.movie.find({rating:{$in:[3,4,5]}})

---- to find all movies with price 200,300,400

price:{$in:[200,300,400]}

db.movie.find({price:{$in:[200,300,400]}})

--------to find all movies with price 200,300,400 or rating 4

rating:4

price:{$in:[200,300,400]}

$or:[{ rating:4},{ price:{$in:[200,300,400]}}]

db.movie.find({ $or:[{ rating:4},{ price:{$in:[200,300,400]}}] })

-------find all movies in which actor Amitabh is acted

db.movie.find({‘actor.0’:’Amitabh’})

-------- to find all documents which has rating key

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

-------to find all movies with rating is null

db.movie.find({rating:{$in:[null]}}).pretty()

-------to find all movies with rating is null and rating key exists

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

--------- to find all movies in which actor array size =3

db.movie.find({actor:{$size:3}})

------- to find movies whose name starts with A

db.movie.find({name:/^A/})

-----to find all movies with name ends with t

db.movie.find({name:/t$/})

------to find all movies with name starts with either S or k

db.movie.find({name:/^[sk]/})

-------- to find all movies with rating divisible by 2

db.movie.find({rating :{$mod:[2,0]}})

------ to find all with price not equal to either 200, 300,400

db.movie.find({price:{$nin:[200,300,400]}})

-------to increase rating by 2 for kahani movie

> db.movie.update({name:'kahani'},{$inc:{rating:2}},{multi:true})

----to create capped collection

it will allow you store maximum 2 documents in the collection

> db.createCollection("testcollection",{capped:true,max:2,size:40000})

----to create capped collection

> db.createCollection("testcollection")