AMIS

COMPARING MONGODB SEARCH WITH ORACLE SQL QUERIES

MongoDB Find & Aggregate vs comparable Oracle SQL queries

Lucas Jellema

mongoDB

TOPICS

- Filter & Sort (find, sort)
- Aggregation ($group, $project, $match, $sort)
- Lookup & Outer Join ($lookup, $arrayElemAt)
- Facet Search ($facet, $bucket, $sortByCount)
- Update (findAndModify, forEach, save, update, upsert, $set, $unset)
- Date and Time operations
- Materialized View ($out)
- Nested documents/tables ($unwind, $reduce)
- Geospatial (ensureIndex, 2dsphere, $near, $geoNear)
- Text Search (createIndex, text, $text, $search)
- Stored Procedures (db.system.js.save, $where)

AMIS

DATA SET

- HRM
  - Collections emp and dept – JSON documents in MongoDB database
  - Tables EMP and DEPT – relational records in Oracle Database

DEPT	
DEPTNO	DNAME
10	ACCOUNTS
20	RESEARCH
30	SALES

EMP		
EMPNO	DEPTNO	ENAME
7782	10	CLARK
7934	10	MILLER
7876	20	ADAMS
7902	20	FORD
7900	30	JAMES

AMIS

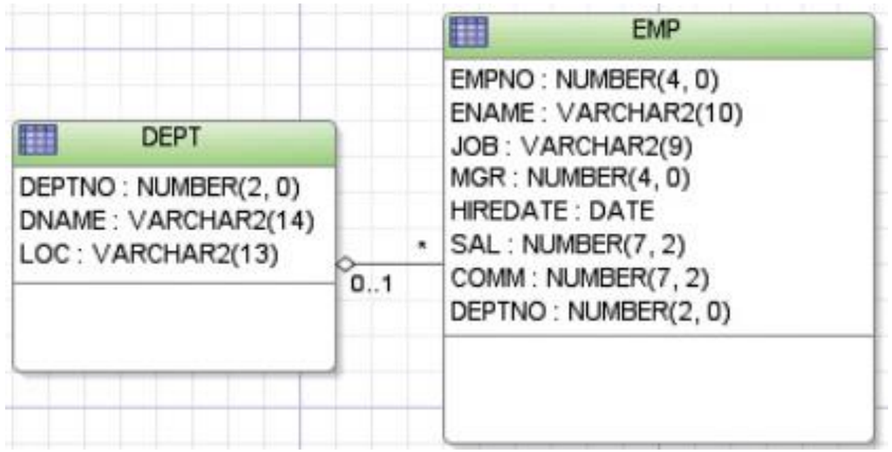
HRM DATA SET
JSON DOCUMENT COLLECTIONS EMP AND DEPT

```
{ "deptno" : 10, "dname" : "ACCOUNTING", "loc" : "NEW YORK" }
{ "deptno" : 20, "dname" : "RESEARCH", "loc" : "DALLAS" }
{ "deptno" : 30, "dname" : "SALES", "loc" : "CHICAGO" }
{ "deptno" : 40, "dname" : "OPERATIONS", "loc" : "BOSTON" }
```

```
{ "EMPNO" : 7369, "ENAME" : "SMITH", "JOB" : "CLERK", "MGR" : 7902, "SAL" : 800, "COMM" : "", "DEPTNO" : 20 }
{ "EMPNO" : 7499, "ENAME" : "ALLEN", "JOB" : "SALESMAN", "MGR" : 7698, "HIREDATE" : "20-02-81", "SAL" : 1600, "COMM" : 300, "DEPTNO" : 30 }
{ "EMPNO" : 7521, "ENAME" : "WARD", "JOB" : "SALESMAN", "MGR" : 7698, "HIREDATE" : "22-02-81", "SAL" : 1250, "COMM" : 500, "DEPTNO" : 30 }
{ "EMPNO" : 7566, "ENAME" : "JONES", "JOB" : "MANAGER", "MGR" : 7839, "HIREDATE" : "02-04-81", "SAL" : 2975, "COMM" : "", "DEPTNO" : 20 }
{ "EMPNO" : 7654, "ENAME" : "MARTIN", "JOB" : "SALESMAN", "MGR" : 7698, "HIREDATE" : "28-09-81", "SAL" : 1250, "COMM" : 1400, "DEPTNO" : 30 }
{ "EMPNO" : 7782, "ENAME" : "CLARK", "JOB" : "MANAGER", "MGR" : 7839, "HIREDATE" : "09-06-81", "SAL" : 2450, "COMM" : "", "DEPTNO" : 10 }
{ "EMPNO" : 7788, "ENAME" : "SCOTT", "JOB" : "ANALYST", "MGR" : 7566, "HIREDATE" : "09-12-82", "SAL" : 3000, "COMM" : "", "DEPTNO" : 20 }
{ "EMPNO" : 7839, "ENAME" : "KING", "JOB" : "PRESIDENT", "MGR" : "", "HIREDATE" : "17-11-81", "SAL" : 5000, "COMM" : "", "DEPTNO" : 10 }
{ "EMPNO" : 7844, "ENAME" : "TURNER", "JOB" : "SALESMAN", "MGR" : 7698, "HIREDATE" : "08-09-81", "SAL" : 1500, "COMM" : 0, "DEPTNO" : 30 }
{ "EMPNO" : 7876, "ENAME" : "ADAMS", "JOB" : "CLERK", "MGR" : 7788, "HIREDATE" : "12-01-83", "SAL" : 1100, "COMM" : "", "DEPTNO" : 20 }
{ "EMPNO" : 7900, "ENAME" : "JAMES", "JOB" : "CLERK", "MGR" : 7698, "HIREDATE" : "03-12-81", "SAL" : 950, "COMM" : "", "DEPTNO" : 30 }
{ "EMPNO" : 7902, "ENAME" : "FORD", "JOB" : "ANALYST", "MGR" : 7566, "HIREDATE" : "03-12-81", "SAL" : 3000, "COMM" : "", "DEPTNO" : 20 }
{ "EMPNO" : 7934, "ENAME" : "MILLER", "JOB" : "CLERK", "MGR" : 7782, "HIREDATE" : "23-01-82", "SAL" : 1300, "COMM" : "", "DEPTNO" : 10 }
{ "EMPNO" : 7698, "ENAME" : "BLAKE", "JOB" : "MANAGER", "MGR" : 7839, "HIREDATE" : "01-05-81", "SAL" : 2850, "COMM" : "", "DEPTNO" : 30 }
```

AMIS

HRM DATA SET
TABLES EMP AND DEPT



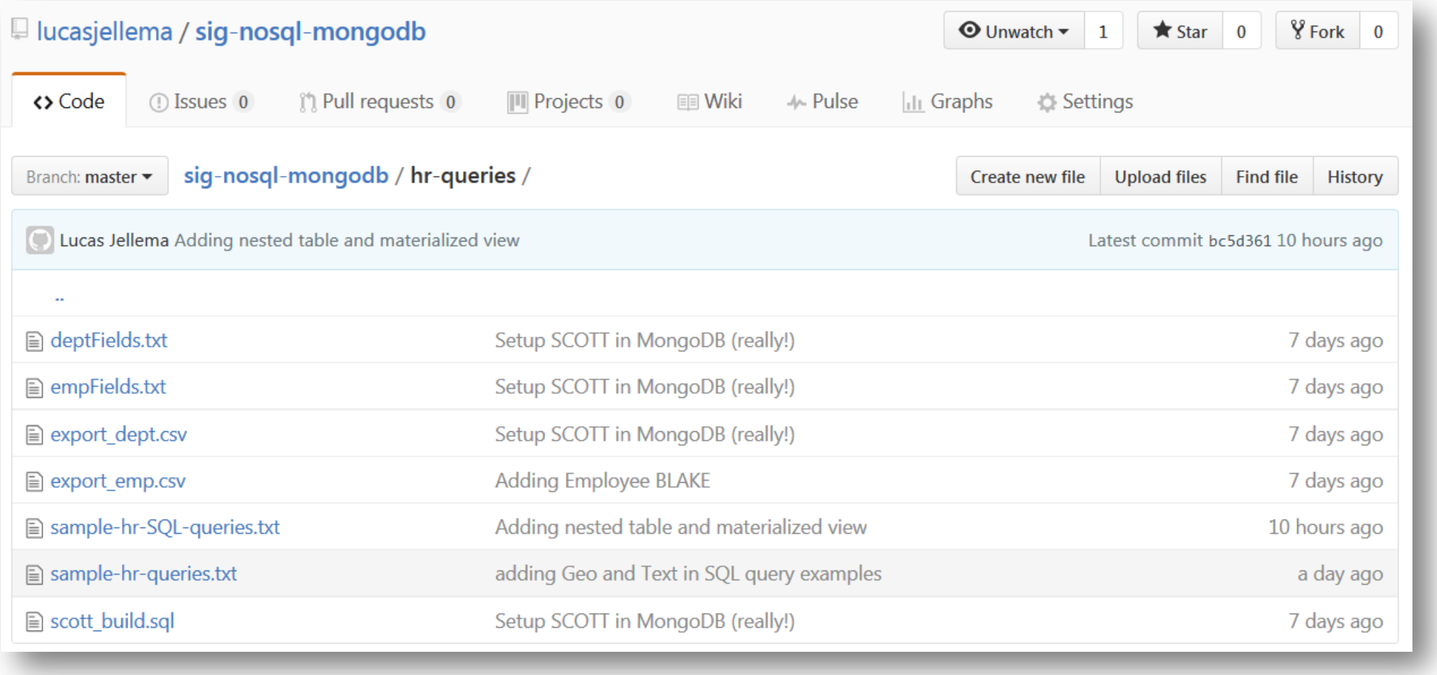
DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	12/17/1980	800	-	20
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	02/22/1981	1250	500	30
7566	JONES	MANAGER	7839	04/02/1981	2975	-	20
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30
7698	BLAKE	MANAGER	7839	05/01/1981	2850	-	30
7782	CLARK	MANAGER	7839	06/09/1981	2450	-	10
7788	SCOTT	ANALYST	7566	04/19/1987	3000	-	20
7839	KING	PRESIDENT	-	11/17/1981	5000	-	10
7844	TURNER	SALESMAN	7698	09/08/1981	1500	0	30
7876	ADAMS	CLERK	7788	05/23/1987	1100	-	20
7900	JAMES	CLERK	7698	12/03/1981	950	-	30
7902	FORD	ANALYST	7566	12/03/1981	3000	-	20
7934	MILLER	PA	7782	01/23/1982	1300	-	10

AMIS

ALL SOURCES ARE AVAILABLE ON GITHUB

- https://github.com/lucasjellema/sig-nosql-mongodb



AMIS

FIND THE NAMES OF ALL MANAGERS

```
db.emp.find
( {"JOB":"MANAGER"}
, {ENAME:1}
)
```

```
select ename
from   emp
where  job = 'MANAGER'
```

AMIS

FIND NAME AND SALARY OF SALESMEN ORDERED BY SALARY FROM HIGH TO LOW

```
db.emp.find
( {"JOB":"SALESMAN"}
, { ENAME:1
  , SAL:1}
)
.sort
( {'SAL':-1})
```

```
select ename
,      sal
from   emp
where  job = 'SALESMAN'
order
by     sal desc
```

AMIS

FIND NAME AND SALARY OF TWO HIGHEST EARNING SALESMEN – BEST PAID FIRST

```
db.emp.find
( {"JOB":"SALESMAN"}
, { ENAME:1
  , SAL:1}
)
.sort
( {'SAL':-1})
.limit(2)
```

```
select ename
,      sal
from   emp
where  job = 'SALESMAN'
order
by     sal desc
FETCH FIRST 2 ROWS ONLY
```

AMIS

FIND EMPLOYEES WITH ‘AR’ IN THEIR NAME – IN ALPHABETICAL ORDER BY NAME

```
db.emp.find
( {"ENAME":{$regex: "AR"} }
, { ENAME:1
  , SAL:1}
)
.sort
( {'ENAME':1})
```

```
select ename
,      sal
from   emp
where  ename like '%AR%'
order
by     ename
```

AMIS

FIND EMPLOYEES NOT IN DEPARTMENT 10, NAME AND SALARY AND SORTED ALPHABETICALLY BY NAME

```
db.emp.find
( { "DEPTNO":{$ne: 10} }
, { ENAME:1
  , SAL:1
  , DEPTNO:1
  }
)
.sort
( {'ENAME':1})
```

```
select ename
,      sal
,      deptno
from   emp
where  deptno != 10
order
by     ename
```

AMIS

SET DATE TYPE PROPERTY STARTDATE DERIVED FROM STRING TYPE PROPERTY HIREDATE

```
db.emp.find().forEach
( function (elem) {
    elem.startdate =
     new Date( "19“
         + elem.HIREDATE.substring(6)
         + "-"
         + elem.HIREDATE.substring(3,5)
         + "-"
         + elem.HIREDATE.substring(0,2)
         );
    db.emp.save(elem);
  }
)
```

```
alter table emp
add   (startdate date)

update emp
set    startdate =
         to_date( hiredate, 'DD-MM-RR')
```

AMIS

FIND SALESMEN WITH A TOTAL INCOME HIGHER THAN 2000

```
-- use $where with embedded JavaScript
db.emp.find
( {"JOB":"SALESMAN"
, $where :
    " this.SAL +
        (this.COMM != null? this.COMM: 0)
      > 2000"
  }
)
```

```
select *
from   emp
where  sal + nvl(comm, 0) > 2000
```

CREATE STORED FUNCTION SALARY CAP TO RETURN MAX SALARY PER JOB; USE SALARY CAP TO FIND EMPLOYEES EARNING OVER THEIR CAP

```
db.system.js.save({
   "_id": "salaryCap",
   "value": function(job) {
      return job=='CLERK'?1000
           :(job=='ANALYST'?3500
           :(job=='SALESMAN'?2000
           :(job=='MANAGER'?3000
           :10000
           )));
    }
})

-- load function in current database
db.loadServerScripts();



db.emp.find(
{ $where : " this.SAL >
                    salaryCap(this.JOB)"}
, {ENAME:1, SAL:1, JOB:1})
```

```
create or replace
function salary_cap
(p_job in varchar2)
return number
is
begin
  return
  case p_job
  when 'CLERK' then 1000
  when 'ANALYST' then 3500
  when 'SALESMAN' then 2000
  when 'MANAGER' then 3000
  else  10000
  end;
end salary_cap;


select ename
,      sal
,      job
from   emp
where  sal > salary_cap( job)
```

SELECT NAME, STARTMONTH AND STARTYEAR FOR ALL EMPLOYEES

```
db.emp.aggregate(
[{$project: {
    "ENAME": 1,
    "startmonth": { $month: "$startdate"},
    "startyear": { $year: "$startdate"}
    }
 }
]
)
```

```
select ename
,      extract (month from startdate)
           as startmonth
,      extract (year from startdate)
           as startyear
from   emp
```

TOTAL SALARY SUM, TOTAL NUMBER OF EMPLOYEES, THE HIGHEST SALARY AND THE EARLIEST STARTDATE

```
db.emp.aggregate(
[{$group: {
     _id: null,
     total_salary_sum: { $sum: "$SAL" },
     total_staff_count: { $sum: 1 },
     max_sal: { $max: "$SAL" },
     min_startdate: { $min: "$startdate" }
     }
 }
]
)
```

```
select sum(sal) total_salary_sum
,      count(*) total_staff_count
,      max(sal)  max_sal
,      min(startdate) min_startdate
from   emp
```

AMIS

TOTAL SALARY SUM, TOTAL NUMBER OF EMPLOYEES, THE HIGHEST SALARY AND THE EARLIEST STARTDATE *PER DEPARTMENT*

```
db.emp.aggregate(
[{$group: {
     _id: "$DEPTNO",
     total_salary_sum: { $sum: "$SAL" },
     total_staff_count: { $sum: 1 },
     max_sal: { $max: "$SAL" },
     min_startdate: { $min: "$startdate" }
     }
 }
]
)
```

```
select deptno
,      extract (year from startdate)
                                  hireyear
,      sum(sal) total_salary_sum
,      count(*) total_staff_count
,      max(sal)  max_sal
,      min(startdate) min_startdate
from   emp
group
by     deptno
,      extract (year from startdate)
```

AMIS

TOTAL SALARY SUM, NUMBER OF EMPLOYEES, HIGHEST SALARY AND EARLIEST STARTDATE PER DEPARTMENT AND HIREYEAR
WITH NUMBER OF EMPLOYEES TWO OR MORE

```
db.emp.aggregate(
[{$group: {
     _id: { deptno: "$DEPTNO“
          , hireyear :
              {  $year: "$startdate"}
          },
     total_salary_sum: { $sum: "$SAL" },
     total_staff_count: { $sum: 1 },
     max_sal: { $max: "$SAL" },
     min_startdate: { $min: "$startdate" }
     }
 }
,{$match: {
     total_staff_count: { $gt: 1 }
     }
 }
]
)
```

```
select deptno
,      extract (year from startdate)
                                  hireyear
,      sum(sal) total_salary_sum
,      count(*) total_staff_count
,      max(sal)  max_sal
,      min(startdate) min_startdate
from   emp
having count(*) > 1
group
by     deptno
,      extract (year from startdate)
```

ALL EMPLOYEES WITH THEIR DEPARTMENT DETAILS (WHEN AVAILABLE)

```
db.emp.aggregate(
[{$lookup: {
     from:"dept",
     localField:"DEPTNO",
     foreignField:"deptno",
     as:"dept"
     }
,{$project: {
     "EMPNO": 1,
     "ENAME": 1,
     "DEPT": { $arrayElemAt:["$dept", 0]}
     }
 }
]
)
```

```
select e.*
,      d.*
from   emp e
       left outer join
       dept d
       on (e.deptno = d.deptno)
```

ALL DEPARTMENTS WITH A LIST OF THE NAMES OF THEIR EMPLOYEES

```
db.dept.aggregate(
[{$lookup: {
   from:"emp",
   localField:"deptno",
   foreignField:"DEPTNO",
   as:"emps"
   }
 }
,{$project: {
   "deptno": 1,
   "dname": 1,
   "staff": {
      $reduce: {
        input: "$emps",
        initialValue: "",
        in: { $concat : ["$$value"
                         , ", "
                         ,"$$this.ENAME"]
            }
               } // reduce
            } // staff
    } // project
 }
])
```

```
select d.deptno, d.dname
,      listagg( ename, ',')
       within group (order by ename)
       as "staff"
from   dept d
       left outer join
       emp e
       on (d.deptno = e.deptno)
group
by     d.deptno
,      d.dname
```

ALL EMPLOYEES WHO WORK IN NEW YORK

```
db.emp.aggregate(
[{$lookup: {
     from:"dept",
     localField:"DEPTNO",
     foreignField:"deptno",
     as:"dept"
     }
,{$project: {
     "EMPNO": 1,
     "ENAME": 1,
     "DEPT": { $arrayElemAt:["$dept", 0]}
     }
 }
, {$match: {  "DEPT.loc" :"NEW YORK"} }
]
)
```

```
select e.*
,      d.*
from   emp e
       left outer join
       dept d
       on (e.deptno = d.deptno)
where  d.loc = 'NEW YORK'
```

EMPLOYEE NAMED KING
WITH ALL EMPLOYEES WHO WORK UNDER HER OR HIM
AND A NEAT LIST OF THE NAMES OF THESE SUBORDINATE STAFF

```
db.emp.aggregate(
[{$match: { ENAME: "KING"}}
,{$lookup: {
         from:"emp",
         localField:"EMPNO",
         foreignField:"MGR",
         as:"subordinates"
     }
 }
,{$project: {
   "EMPNO": 1,
   "ENAME": 1,
   "subordinates": 1,
   "staff": { $reduce: {
               input: "$subordinates",
               initialValue: "",
               in: { $concat :
                  ["$$value", ", ","$$this.ENAME"]
                   } //in
                       } // reduce
            },
    }
 }
]
)
```

```
select e.*
,      cursor( select *
               from emp s
               where s.mgr = e.empno
             ) subordinates
,      ( select listagg( s.ename, ',')
                within group
                (order by ename)
         from   emp s
         where  s.mgr = e.empno
       ) as "staff"
from   emp e
where  e.ename = 'KING'
```

FACET AGGREGATION: # EMPLOYEES BY JOB, BY SALARY BUCKET, BY DEPARTMENT AND BY STARTDATE (1)

```
db.emp.aggregate(
[{$facet: {
   "categorizedByJob": [
        { $sortByCount: "$JOB" }
      ],
   "categorizedBySalary": [
        {$bucket: {
            groupBy: "$SAL",
            boundaries: [0, 1000, 2000 ,3000 ,10000 ],
            default: "Other",
            output: {
              "count": { $sum: 1 },
              "employees": { $push: "$ENAME" }
            } // output
            }// bucket
        }
      ],
   "categorizedByDepartment": [
        { $sortByCount: "$DEPTNO" }
      ],
   "categorizedByHiredate(Auto)": [
        {
          $bucketAuto: {
            groupBy: "$startdate",
            buckets: 4
          }
        }
      ]
    }
  }
])
```

```
-- categorizedByJob
select job
,      count(*) as "count"
from   emp
group
by     job
order
by     "count" desc
```

```
-- categorizedByDepartment
select deptno
,      count(*) as "count"
from   emp
group
by     deptno
order
by     "count" desc
```

FACET AGGREGATION: # EMPLOYEES BY JOB, BY SALARY BUCKET, BY DEPARTMENT AND BY STARTDATE (2)

```
db.emp.aggregate(
[{$facet: {
   "categorizedByJob": [
        { $sortByCount: "$JOB" }
      ],
   "categorizedBySalary": [
        {$bucket: {
            groupBy: "$SAL",
            boundaries: [0, 1000, 2000 ,3000 ,10000 ],
            default: "Other",
            output: {
              "count": { $sum: 1 },
              "employees": { $push: "$ENAME" }
            } // output
            }// bucket
        }
      ],
   "categorizedByDepartment": [
        { $sortByCount: "$DEPTNO" }
      ],
   "categorizedByHiredate(Auto)": [
        {
          $bucketAuto: {
            groupBy: "$startdate",
            buckets: 4
          }
        }
      ]
    }
  }
])
```

```
-- categorizedBySalary
with bucket_boundaries as
( select 10000 lower_boundary from dual
  union all
  select 2000 lower_boundary from dual
  union all
  select 3000 lower_boundary from dual
  union all
  select 10000 lower_boundary from dual
)
, buckets as
( select lower_boundary
  ,      lead(lower_boundary) over
         (order by lower_boundary)-1 upper_boundary
  from   bucket_boundaries
)
select lower_boundary
,      count(*)
from   emp
       left outer join
       buckets
       on (sal between lower_boundary
                   and upper_boundary)
group
by     lower_boundary
```

FACET AGGREGATION: # EMPLOYEES BY JOB, BY SALARY BUCKET, BY DEPARTMENT AND BY STARTDATE (3)

```
db.emp.aggregate(
[{$facet: {
   "categorizedByJob": [
        { $sortByCount: "$JOB" }
      ],
   "categorizedBySalary": [
        {$bucket: {
            groupBy: "$SAL",
            boundaries: [0, 1000, 2000 ,3000 ,10000 ],
            default: "Other",
            output: {
              "count": { $sum: 1 },
              "employees": { $push: "$ENAME" }
            } // output
            }// bucket
        }
      ],
   "categorizedByDepartment": [
        { $sortByCount: "$DEPTNO" }
      ],
   "categorizedByHiredate(Auto)": [
        {
          $bucketAuto: {
            groupBy: "$startdate",
            buckets: 4
          }
        }
      ]
    }
  }
])
```

```
-- categorizedByHiredate
with tiled as
( select ename
  ,      startdate
  ,      ntile(4)  over (order by startdate asc)
         as size_bucket
  from   emp
)
select size_bucket
,      count(*) as "count"
,      listagg( ename, ',')
       within group (order by startdate) employees
from   tiled
group
by     size_bucket
order
by     size_bucket
```

CREATE MATERIALIZED COLLECTION FROM QUERY DEPARTMENTS WITH NESTED EMPLOYEES

```
db.dept.aggregate(
[  {$lookup:
      {
         from:"emp",
         localField:"deptno",
         foreignField:"DEPTNO",
         as:"emps"
     }
   }
, {$out: "departments"
  }
]
)
```

```
create or replace type emp_t as object
( EMPNO             NUMBER(4)
, ENAME             VARCHAR2(10)
, JOB               VARCHAR2(9)
, MGR               NUMBER(4)
, SAL               NUMBER(7,2)
, COMM              NUMBER(7,2)
, STARTDATE         DATE
)

create or replace type emp_tbl_t as  table of emp_t

-- create materialized view
-- with nested table
create materialized view departments
BUILD IMMEDIATE
REFRESH FORCE
ON DEMAND
as
select deptno
,      dname
,      loc
,      cast ( multiset ( select empno, ename, job
                         ,    mgr, sal, comm, hiredate
                         from   emp e
                         where  e.deptno = d.deptno
                       )
              as emp_tbl_t) staff
from   dept d
```

FIND DEPARTMENT THAT CONTAINS EMPLOYEE NAMED KING FROM DEPARTMENTS COLLECTION WITH NESTED EMPLOYEES

```
db.departments.find
( {"emps.ENAME":"KING"}
)
```

```
select d.deptno
,      d.dname
,      d.loc
from   departments d
where  ( select count(*)
         from table(d.staff)
         where ename ='KING'
       ) > 0
```

ONLY FIND EMPLOYEE KING (AND NOT ALL EMPLOYEES IN THE DEPARTMENT) FROM DEPARTMENTS COLLECTION WITH NESTED EMPLOYEES

```
db.departments.aggregate(
[
  {$unwind: {path:"$emps"}}
, {$match: { "emps.ENAME": "KING"}}
, {$project: {
        "EMPNO": "$emps.EMPNO",
        "JOB": "$emps.JOB",
        "ENAME": "$emps.ENAME",
        "STARTDATE": "$emps.startdate",
        "DNAME": 1
    }
   }
]
)
```

```
select d.deptno
,      d.dname
,      d.loc
,      staff.*
from   departments d
,      table(d.staff) staff
where  staff.ename = 'KING'
```

FIND NAMES OF ALL MANAGERS FROM DEPARTMENTS COLLECTION WITH NESTED EMPLOYEES

```
db.departments.aggregate(
[
  {$unwind: {path:"$emps"}}
, {$match: { "emps.JOB": "MANAGER"}}
, {$project: {
        "ENAME": "$emps.ENAME",
    }
   }
]
)
```

```
select staff.ename
from   departments d
,      table(d.staff) staff
where  staff.job = 'MANAGER'
```

FIND ALL EMPLOYEES WHO ARE NOT IN DEPARTMENT 10, WITH THEIR NAME AND SALARY AND SORTED ALPHABETICALLY BY NAME FROM DEPARTMENTS COLLECTION WITH NESTED EMPLOYEES

```
db.departments.aggregate(
[
   {$match: { "DEPTNO": {$ne:10}}}
 , {$unwind: {path:"$emps"}}
 , {$project: {
        "ENAME": "$emps.ENAME",
        "SAL": "$emps.SAL",
        "DEPTNO": 1,
    }
   }
 , {$sort : {"ENAME":1}}
]
)
```

```
select staff.ename
,      staff.sal
,      d.deptno
from   departments d
,      table(d.staff) staff
where  d.deptno != 10
order
by     staff.ename
```

TOTAL SALARY SUM, TOTAL NUMBER OF EMPLOYEES, THE HIGHEST SALARY AND THE EARLIEST STARTDATE, PER DEPARTMENT FROM DEPARTMENTS COLLECTION WITH NESTED EMPLOYEES

```
db.departments.aggregate(
[
   {$unwind: {path:"$emps"}}
,  {$group:{ _id: '$deptno'
           , total_salary_sum : {$sum: "$emps.SAL"}
           , total_staff_count : {$sum: 1}
           , max_sal : {$max: "$emps.SAL"}
           , min_startdate : {$min: "$emps.startdate"}
           }
   }
]
)
```

```
select d.deptno
,      sum(staff.sal) total_salary_sum
,      count(staff.empno) total_staff_count
,      max(staff.sal) max_sal
,      min(staff.startdate) min_startdate
from   departments d
,      table(d.staff) staff
group
by     d.deptno
```

ADDING GEO LOCATIONS AND CREATE GEO INDEX

```
db.dept.findAndModify({
    query: { loc: "NEW YORK" },
    update: { $set: { "location" : {
                        "type" : "Point",
                        "coordinates" : [ -73.9352, 40.7306 ]
                        }
              } },
    upsert: true
})
db.dept.findAndModify({
    query: { loc: "DALLAS" },
    update: { $set: { "location" : {
                        "type" : "Point",
                        "coordinates" : [ -96.8005, 32.7801 ]
                        }
              } },
    upsert: true
})
db.dept.findAndModify({
    query: { loc: "BOSTON" },
    update: { $set: { "location" : {
                        "type" : "Point",
                        "coordinates" : [ -71.0598, 42.3584 ]
                        }
              } },
    upsert: true
})
...
-- create spatial index
db.dept.ensureIndex( { location : "2dsphere" } );
```

```
-- add column geo_location to hold SDO_GEOMETRY
alter table dept
add (geo_location SDO_GEOMETRY)

-- add geo location to each department
update dept
set    geo_location = SDO_GEOMETRY(2001, 8307,
                   SDO_POINT_TYPE (-73.935242, 40.730610,NULL)
                                  , NULL, NULL)
where  loc = 'NEW YORK'

update dept
set    geo_location = SDO_GEOMETRY(2001, 8307,
          SDO_POINT_TYPE (-96.8005, 32.7801,NULL), NULL, NULL)
where  loc = 'DALLAS'

-- insert dimensional meta information for the  spatial column
INSERT INTO USER_SDO_GEOM_METADATA
(TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('DEPT', 'GEO_LOCATION',
   SDO_DIM_ARRAY
    (SDO_DIM_ELEMENT('LONG', -180.0, 180.0, 0.5),
     SDO_DIM_ELEMENT('LAT', -90.0, 90.0, 0.5)
    ),
     8307
);

-- create spatial index
CREATE INDEX dept_spatial_idx
ON dept(geo_location)
INDEXTYPE IS mdsys.spatial_index;
```

FIND DEPARTMENTS WITHIN 500 KM FROM WASHINGTON DC ([-77.0364, 38.8951])

```
db.dept.find(
  {
    location : {
      $near : {
        $geometry : {
          type : "Point" ,
          coordinates : [ -77.0364, 38.8951 ]
        },
        $maxDistance : 500000
      }
    }
  }
 )
```

```
with d as
( SELECT loc
  ,      SDO_GEOM.SDO_DISTANCE
         ( SDO_GEOMETRY(2001, 8307
         , SDO_POINT_TYPE ( -77.0364, 38.8951,NULL)
         , NULL, NULL
         )
         , geo_location
         , 0.005
         , 'unit=KM'
         ) distance
  from   dept
)
select d.*
from   d
where  d.distance < 500
```

ALL DEPARTMENTS, THE DISTANCE FOR EACH DEPARTMENT IN KILOMETER FROM WASHINGTON DC, ORDERED BY THAT DISTANCE

```
db.dept.aggregate([
    { "$geoNear": {
        "near": {
            "type": "Point",
            "coordinates": [ -77.0364, 38.8951 ]
        },
        "spherical": true,
        "distanceField": "distanceFromTarget",
        "distanceMultiplier": 0.001 // from meter to km
     }}
 , {$sort : {"distanceFromTarget":1}}
,  {$project: {
        _id: 0,
        dname: 1,
        loc: 1,
        "distance from Washington DC":
                            { $trunc : "$distanceFromTarget"},
    }
   }
])
```

```
with d as
( SELECT loc
  ,      dname
  ,      SDO_GEOM.SDO_DISTANCE
         ( SDO_GEOMETRY(2001, 8307
         , SDO_POINT_TYPE ( -77.0364, 38.8951,NULL)
         , NULL, NULL
         )
         , geo_location
         , 0.005
         , 'unit=KM'
         ) distance
  from   dept
)
select d.dname
,      d.loc
,      d.distance “distance from Washington DC”
from   d
order
by     d.distance
```

ADD BIOGRAPHIES TO EMPLOYEES (PREPARING FOR TEXT INDEX AND SEARCH) AND CREATE TEXT INDEX

```
db.emp.findAndModify({
    query: { ENAME: "KING" },
    update: { $set: { "bio" : "Gerald Ford was born ...." }
            },
    upsert: true
})
db.emp.findAndModify({
    query: { ENAME: "BLAKE" },
    update: { $set: { "bio" : "Jamaican sprinter Yohan Blake
                                ..." }
            },
    upsert: true
})
db.emp.findAndModify({
    query: { ENAME: "FORD" },
    update: { $set: { "bio" : "Harrison Ford is one of
                                            ...Han Solo." }
            },
    upsert: true
})

-- create text index, allowing use of text search
db.emp.createIndex(
{ ENAME:'text',
  JOB:'text',
  BIO:'text',
}
,{  weights: { ENAME:10, JOB:5, bio:1}
  , name: 'employee_text_index'
}
)
```

```
update emp
set    bio = q'[Gerald Ford was born ... in 2006.]'
where  ename = 'KING‘

update emp
set    bio = q'[Jamaican sprinter Yohan Blake holds
...]'
where  ename = 'BLAKE'

-- create a multi column text index
exec ctx_ddl.create_preference
     ( 'my_mcds', 'multi_column_datastore' )
exec ctx_ddl.set_attribute
     ( 'my_mcds', 'columns', 'bio, ename, job' )
-- to support stemming
exec ctxsys.ctx_ddl.create_preference
     ('my_lexer','BASIC_LEXER');
exec ctxsys.ctx_ddl.set_attribute
     ('my_lexer','index_stems','1');
exec ctxsys.ctx_ddl.create_preference
     ('my_wordlist','BASIC_WORDLIST');
exec ctxsys.ctx_ddl.set_attribute
     ('my_wordlist','stemmer','ENGLISH');

create index emp_txt_idx on emp( ename )
indextype is ctxsys.context
parameters( 'datastore my_mcds WORDLIST my_wordlist
LEXER my_lexer' )
```

WHICH EMPLOYEES ARE FOUND WHEN LOOKING FOR SOMEONE TO LEAD?

```
db.emp.find(
{$text: {$search: 'lead'}}
,{ENAME:1}
)
```

```
-- leveraging the multicolumn text index
-- on ename, bio and job
SELECT ename
,      SCORE(1)
FROM   emp
WHERE  CONTAINS(ename, 'lead', 1) > 0
```

TEXT SEARCH INCLUDING SCORING - APPLYING WEIGHT AND DERIVING APPLICABILITY

```
db.emp.aggregate(
[{ $match: { $text: { $search: 'managing' } } }
,{ $project: {
    _id:0,
    ENAME: 1,
    score: { $meta: 'textScore' } }
 }
,{ $sort: {score:-1} }
]
)
```

```
-- leveraging stemming and the multicolumn text index
-- on ename, bio and job
SELECT ename
,      SCORE(1) score
FROM   emp
WHERE  CONTAINS(ename, '$manage', 1) > 0
order
By     score desc
```