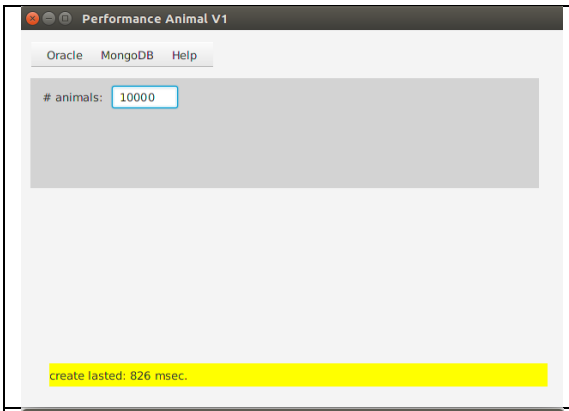
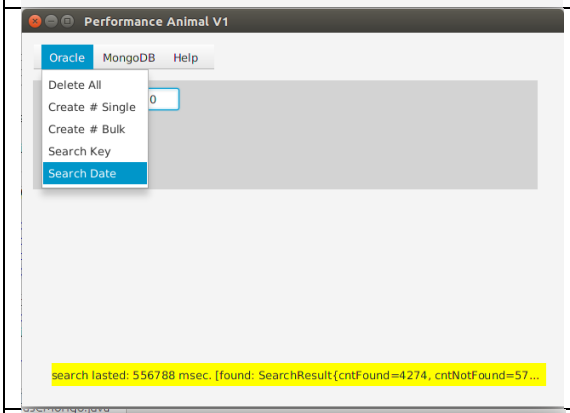
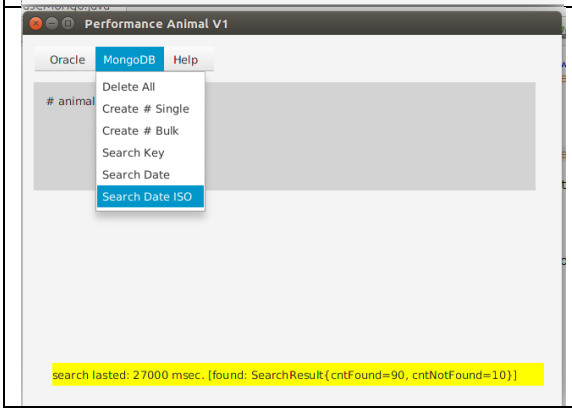
Overview

Simulation of read/write - accesses on a table/collection of animals; both databases are installed on the identical server

User's View

	creating 10000 documents lasts 826 msec
	searching in 100000 documents for a random date needs more than 9 min.
	searching after 100 (non indexed) ISO-Date – values lasts 27000 sec.

Developer's Hints

```
public class Animal {
    private int aid;
    private String aname;
    private LocalDate adate;
    private String adetails;
```

```java
public class MyMath {

    private static final int MONTH_UPPER = 12;
    private static final int DAY_UPPER = 28;
    private static final int YEAR_UPPER = 2100;
    private static final int YEAR_LOWER = 1500;

    /**...7 lines */
    static public long random(long lower, long upper) throws Exception {
        long interval = (long) ((upper - lower) * Math.random());

        return (lower + interval);
    }

    /** return msecs of timespan ...5 lines */
    static public long getMiliSecsBetween(LocalTime start, LocalTime end) throws Exception {
        return Math.abs((start.toNanoOfDay() - end.toNanoOfDay()) / 1000000);
    }

    /**...5 lines */
    static public String getRandomDate() throws Exception {
        String day = Long.toString(MyMath.random(1, DAY_UPPER)+100).substring(1);
        String month = Long.toString(MyMath.random(1, MONTH_UPPER)+100).substring(1);
        String year = Long.toString(MyMath.random(YEAR_LOWER, YEAR_UPPER));
        return day + "." + month + "." + year;
    }
```

MongoDB

```
C:\Program Files\MongoDB\Server\3.4\bin\mongo.exe
{ "_id" : ObjectId("5c3f4281a467e0153f82c1f3"), "aid" : -99, "aname" : "o-animal
", "adate" : { "year" : 2006, "month" : 4, "day" : 26 }, "aDateISO" : ISODate("2
006-04-26T00:00:00Z") }
{ "_id" : ObjectId("5c3f4281a467e0153f82c1f4"), "aid" : -99, "aname" : "l-animal
", "adate" : { "year" : 2041, "month" : 7, "day" : 21 }, "aDateISO" : ISODate("2
041-07-21T00:00:00Z") }
{ "_id" : ObjectId("5c3f4281a467e0153f82c1f5"), "aid" : -99, "aname" : "e-animal
", "adate" : { "year" : 2033, "month" : 11, "day" : 12 }, "aDateISO" : ISODate("
2033-11-12T00:00:00Z") }
{ "_id" : ObjectId("5c3f4281a467e0153f82c1f6"), "aid" : -99, "aname" : "w-animal
", "adate" : { "year" : 2097, "month" : 3, "day" : 16 }, "aDateISO" : ISODate("2
097-03-16T00:00:00Z") }
{ "_id" : ObjectId("5c3f4281a467e0153f82c1f7"), "aid" : -99, "aname" : "n-animal
", "adate" : { "year" : 1975, "month" : 10, "day" : 19 }, "aDateISO" : ISODate("
1975-10-19T00:00:00Z") }
{ "_id" : ObjectId("5c3f4281a467e0153f82c1f8"), "aid" : -99, "aname" : "k-animal
", "adate" : { "year" : 2009, "month" : 2, "day" : 4 }, "aDateISO" : ISODate("20
09-02-04T00:00:00Z") }
{ "_id" : ObjectId("5c3f4281a467e0153f82c1f9"), "aid" : -99, "aname" : "u-animal
", "adate" : { "year" : 1982, "month" : 8, "day" : 15 }, "aDateISO" : ISODate("1
982-08-15T00:00:00Z") }
Type "it" for more
> db.Animals.find().count()
10000
>
```

Working with Date

Insert some persons with ISODate:

```
> db.person.insert({
     firstname: "gerald",
     lastname: "ortner",
     birthdate: ISODate("2018-02-28")})
> db.person.insert({     firstname: "astrid",     lastname: "ortner",     birthdate:
ISODate("2017-02-28")})
➔
> db.person.find()
{ "_id" : ObjectId("5c363cefd526352604ddcbf5"), "firstname" : "gerald", "lastname" : "ortner",
"birthdate" : ISODate("2018-02-28T00:00:00Z") }
{ "_id" : ObjectId("5c363d16d526352604ddcbf6"), "firstname" : "astrid", "lastname" : "ortner",
"birthdate" : ISODate("2017-02-28T00:00:00Z") }
```

Java: convert LocalDate ➔ ISODate:

```
doc.append("aDateISO",
        new BsonDateTime(animal.getAdate().atStartOfDay().toInstant(ZoneOffset.UTC).getEpochSecond() * 1000));
```

Get datespecific information:

```
db.person.aggregate(
   [
     {
       $project:
         {
          _id: 0,
           year: { $year: "$birthdate" },
          month: { $month: "$birthdate" },
           day: { $dayOfMonth: "$birthdate" },
           hour: { $hour: "$birthdate" },
           minutes: { $minute: "$birthdate" },
           seconds: { $second: "$birthdate" },
           milliseconds: { $millisecond: "$birthdate" },
           dayOfYear: { $dayOfYear: "$birthdate" },
           dayOfWeek: { $dayOfWeek: "$birthdate" },
           week: { $week: "$birthdate" }
         }
     }
   ]
)
➔
{ "year" : 2018, "month" : 2, "day" : 28, "hour" : 0, "minutes" : 0, "seconds" :
 0, "milliseconds" : 0, "dayOfYear" : 59, "dayOfWeek" : 4, "week" : 8 }
{ "year" : 2017, "month" : 2, "day" : 28, "hour" : 0, "minutes" : 0, "seconds" :
 0, "milliseconds" : 0, "dayOfYear" : 59, "dayOfWeek" : 3, "week" : 9 }
```

Find datespecific information:

```
db.person.aggregate(
   [
     {
       $project:
         {
          _id: 0,
           year: { $year: "$birthdate" },
          month: { $month: "$birthdate" },
           day: { $dayOfMonth: "$birthdate" }
         }
     },
     {
        $match:
        {  $and:
                [{year: 2018},
                  {month: 2}
                ]
        }
     }
   ]
)
```

Best Results Oracle

	10000 entries	100000 entries	
delete all entries	150 msec	890 msec	
create single entries	4600 msec	46700 msec	each add-call is sent to db
create with bulk	350 msec	4800 msec	bulk => 1 call to db
search key	4400 msec	4500 msec	10000x searches =>size of table does not matter
search date	15100 msec	47700 msec	date-field not indexed

Best Results MongoDB

	10000 entries	100000 entries	
delete all entries	160 msec	1390 msec	
create single entries	5760 msec	49100 msec	each add-call is sent to db
create with bulk	220 msec	1900 msec	bulk => 1 call to db
search key	5300 msec	5300 msec	10000x searches =>size of table does not matter
search date	78300 msec	557000 msec	10000x searches date-field not indexed
search date ISO	270000 msec	2600000 msec	10000x searches ISO-Date