1818
2018

DALHOUSIE UNIVERSITY

Data Management Warehouse & Analytics

Ocean Tracking Network
Case Study

Shivam Gupta
B00810723

- # Task A

  - Read the document on http://oceantrackingnetwork.org/about/ - oceanmonitoring
  - Explored the Ocean dataset on Brightspace and checked all the tables in the data set.
  - Different Dataset and Attributes I discovered:
  - There are 8 tables in the dataset provided in the ocean dataset provided in Brightspace.
    - Animal
    - Data Center
    - Detections
    - Manmade Platform
    - Project Attributes
    - Receivers
    - Recovery
    - Tag Releases

  - All the tables have project reference attribute which is referring to Project attribute table, which stores all the reference code of all the projects.
  - All the tables have data center reference code as well, which is linked to Data Center table. I noticed that Project Attribute table has data center code as well. So, we can say, that no table requires data center code in the table fields as we can get that though project attributes table.

- # Dataset Cleaning and Transformation:

  - Columns were re-arranged.
    Animal Table:
    - Removed the second row which has empty values.
    - Removed the column “TaxonRank”
    - Replaced null values if lifestage, sex and stock column as “not defined” by using formula. (=IF(N215="","Not defined", N215))
    - Removed age column as it doesn’t have any values.
    - Removed NaN value from weight and replaced then with 0.
    - I noticed animal_guid is made form datacenter_reference, animal_projet_reference, and animal_reference_id. I didn’t do anything on this column to not remove this data. But it can be removed.

    Data Center table:
    - Replaced NaN with 0 where ever the values were present.

    Detections table:
    - Removed 2nd row and added the data to column where ever needed.

- Deleted columns : receiver_log_id, depth, uncertainty_in_latitude, uncertainty_in_longitude, depth_data_source, uncertainty_in_depth, other_position_data, dataset_quality
- Replaced sensor data unit and senson data's null values as 0 by the formula given above.
-

Manmade Platform table:

- Removed 2nd row and added the data to column where ever needed.
- Replaced NaN values with 0 in latitude, longitude and platform_depth.
- Platform name has some data which is coming in date format; changed that to text format.

Project Attribute Table:

- Removed 2nd row and added the data to column where ever needed.
- Removed column having null values: project_references, project_doi, project_linestring, geospatial_vertical_min, geospatial_vertical_max, geospatial_vertical_positive, time_coverage_start, time_coverage_end, project_date_modified, project_distribution_statement
- Project_pi_organisation is there which can be normalized into another table. I didn't normalise that as data fetching would take more time in my web page.

Receivers Table:

- Removed 2nd row and added the data to column where ever needed
- Removed all columns having blank values
- Replaced NaN with 0.
- Replaced null values as 'not defined'

Recovery Table:

- Removed 2nd row and added the data to column where ever needed
- Removed all columns having blank values
- Replaced NaN with 0
- Replaced null values as 'not defined'

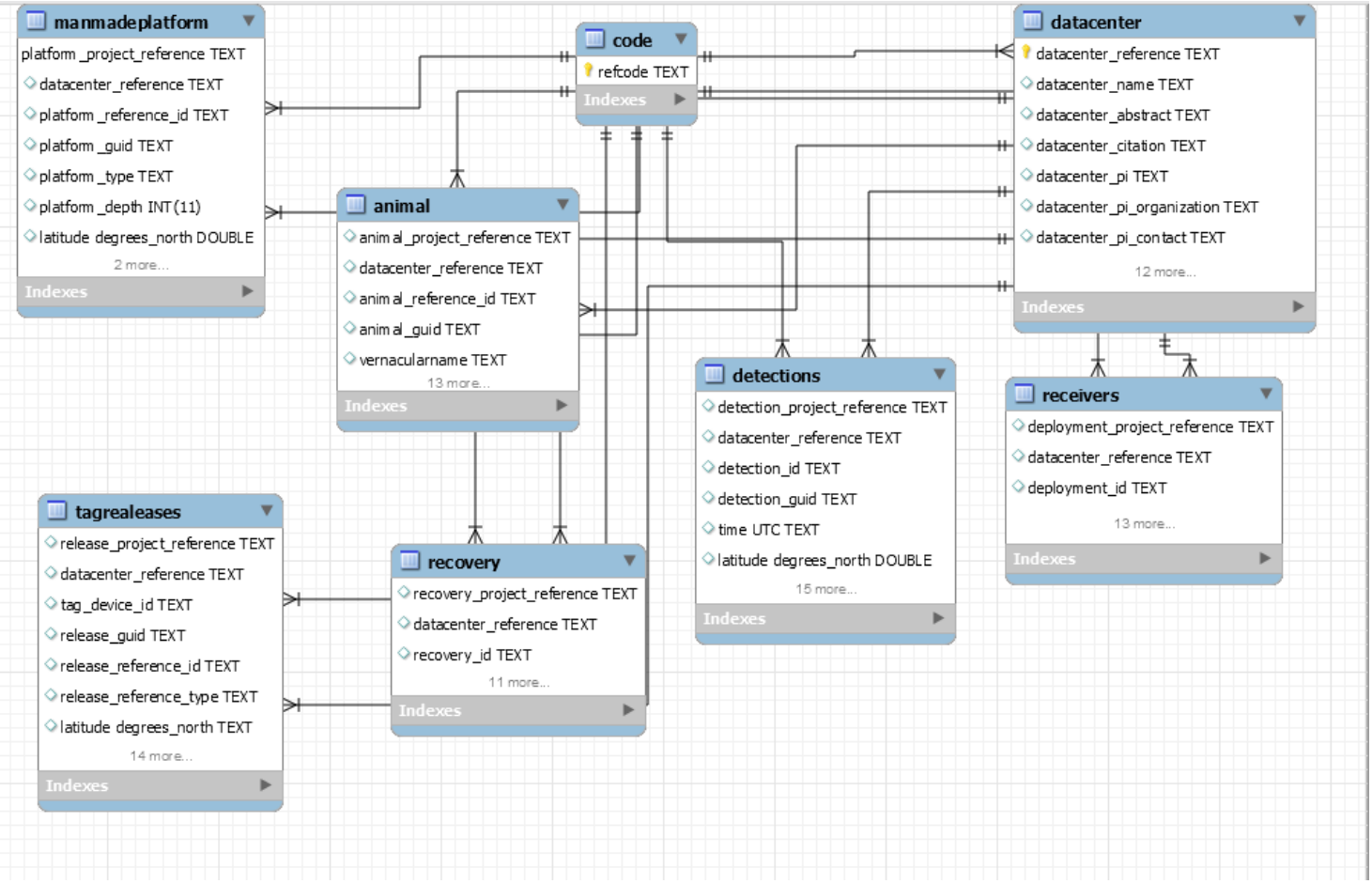
Tag Releases Table:

- Removed 2nd row and added the data to column where ever needed
- Removed all columns having blank values
- Replaced NaN with 0
- Replaced null values as 'not defined'
- Cleaned the date format to store it in database by using formula: =LEFT(J3,10)

Code Table:

- A new table code is created which is containing the reference code.

- **Relational Schema and ERD:**

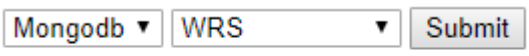


Every table has 1-n cardinality with code table. Which means there are many values of code table in all the table. And refcode is primary key.

Data center is also connected to all the tables with 1-n cardinality, which means, every table has many values for data center. And datacenter_reference is primary key.

- **Task B**

  - Web application is created using JavaScript, html and node.js.
  - I have created a server file for node.js which will do all the connection to both the databases and send the result of the queries to the local host server.
  - Then my JavaScript file will read the local host and print the data in the webpage.
  - Web application simple page with 2 drop downs.



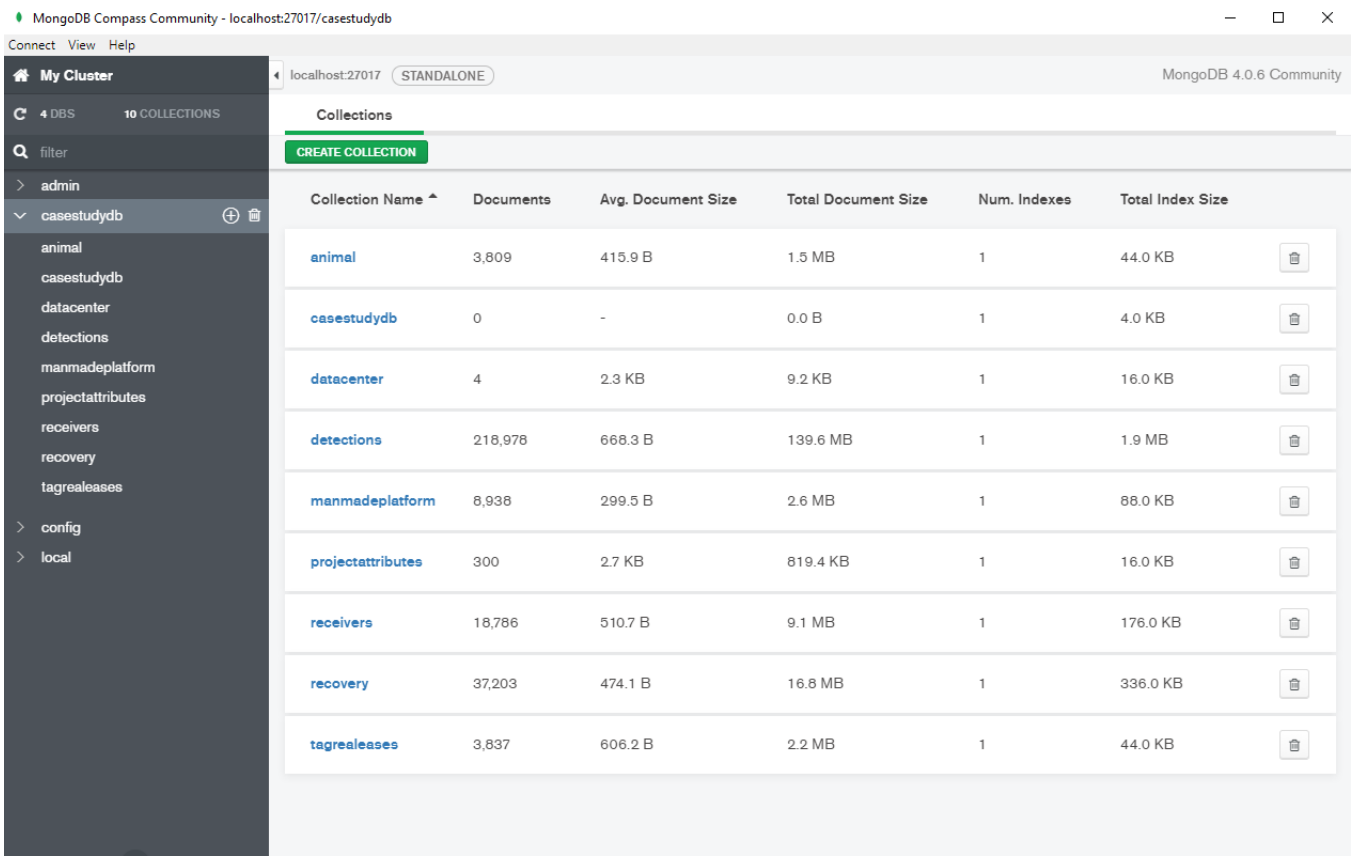
mysql ▾ WRS ▾ Submit

animal_project_reference	datacenter_reference	animal_reference_id	animal_guid	vernacularname	scientificname	aphiaid	tsn	animal_origin	stock	length	length_type	weight	life_stage	age	sex
WRS	OTN-Global	WRS-1073397	OTN-GlobalWRSWRS-1073397	Atlantic salmon	Salmo salar	127186	161996	W	West River, Sheet Harbour	0.204	FORK	0.09	SMOLT	Not Given	U
WRS	OTN-Global	WRS-1073398	OTN-GlobalWRSWRS-1073398	Atlantic salmon	Salmo salar	127186	161996	W	West River, Sheet Harbour	0.214	FORK	0.083	SMOLT	Not Given	U
WRS	OTN-Global	WRS-1073399	OTN-GlobalWRSWRS-1073399	Atlantic salmon	Salmo salar	127186	161996	W	West River, Sheet Harbour	0.211	FORK	0.081	SMOLT	Not Given	U
WRS	OTN-Global	WRS-1073400	OTN-GlobalWRSWRS-1073400	Atlantic salmon	Salmo salar	127186	161996	W	West River, Sheet Harbour	0.186	FORK	0.066	SMOLT	Not Given	U
WRS	OTN-Global	WRS-1073401	OTN-GlobalWRSWRS-1073401	Atlantic salmon	Salmo salar	127186	161996	W	West River, Sheet Harbour	0.188	FORK	0.075	SMOLT	Not Given	U
WRS	OTN-Global	WRS-1073402	OTN-GlobalWRSWRS-1073402	Atlantic salmon	Salmo salar	127186	161996	W	West River, Sheet Harbour	0.181	FORK	0.057	SMOLT	Not Given	U
WRS	OTN-Global	WRS-1073403	OTN-GlobalWRSWRS-1073403	Atlantic salmon	Salmo salar	127186	161996	W	West River, Sheet Harbour	0.188	FORK	0.059	SMOLT	Not Given	U
WRS	OTN-Global	WRS-1073404	OTN-GlobalWRSWRS-1073404	Atlantic salmon	Salmo salar	127186	161996	W	West River, Sheet Harbour	0.193	FORK	0.076	SMOLT	Not Given	U
WRS	OTN-Global	WRS-1073405	OTN-GlobalWRSWRS-1073405	Atlantic salmon	Salmo salar	127186	161996	W	West River, Sheet Harbour	0.216	FORK	0.102	SMOLT	Not Given	U
WRS	OTN-Global	WRS-1073406	OTN-GlobalWRSWRS-1073406	Atlantic salmon	Salmo salar	127186	161996	W	West River, Sheet Harbour	0.202	FORK	0.08	SMOLT	Not Given	U
WRS	OTN-Global	WRS-1073407	OTN-GlobalWRSWRS-1073407	Atlantic salmon	Salmo salar	127186	161996	W	West River, Sheet Harbour	0.185	FORK	0.059	SMOLT	Not Given	U

Mongodb ▾ NSBS ▾ Submit

_id	animal_project_reference	datacenter_reference	animal_reference_id	animal_guid	vernacularname	scientificname	aphiaid	tsn	animal_origin	stock	length	length_type	weight	life_stage	age	sex
5c71edeaab094442a10da3fb	NSBS	OTN-Global	NSBS-ALI	OTN-GlobalNSBSNSBS-ALI	blue shark	Prionace glauca	105801	160424	W	NW Atlantic	1.5748	FORK		JUVENILE	Not Given	F
5c71edeaab094442a10da3fc	NSBS	OTN-Global	NSBS-1248829-2017-08-22-11270	OTN-GlobalNSBSNSBS-1248829-2017-08-22-11270	blue shark	Prionace glauca	105801	160424	W	NW Atlantic	1.04	FORK		JUVENILE	Not Given	F
5c71edeaab094442a10da3fd	NSBS	OTN-Global	NSBS-1248828-2017-08-27-11268	OTN-GlobalNSBSNSBS-1248828-2017-08-27-11268	blue shark	Prionace glauca	105801	160424	W	NW Atlantic	1.37	FORK		JUVENILE	Not Given	F
5c71edeaab094442a10da3fe	NSBS	OTN-Global	NSBS-1248827-2017-08-22-11266	OTN-GlobalNSBSNSBS-1248827-2017-08-22-11266	blue shark	Prionace glauca	105801	160424	W	NW Atlantic	1.77	FORK		JUVENILE	Not Given	F
5c71edeaab094442a10da3ff	NSBS	OTN-Global	NSBS-1248826-2017-08-25-11264	OTN-GlobalNSBSNSBS-1248826-2017-08-25-11264	blue shark	Prionace glauca	105801	160424	W	NW Atlantic	1.52	FORK		JUVENILE	Not Given	F
5c71edeaab094442a10da400	NSBS	OTN-Global	NSBS-1248825-2017-08-22-11262	OTN-GlobalNSBSNSBS-1248825-2017-08-22-11262	blue shark	Prionace glauca	105801	160424	W	NW Atlantic	1.68	FORK		JUVENILE	Not Given	F
5c71edeaab094442a10da401	NSBS	OTN-Global	NSBS-1248829-2017-08-22-11269	OTN-GlobalNSBSNSBS-1248829-2017-08-22-11269	blue shark	Prionace glauca	105801	160424	W	NW Atlantic	1.04	FORK		JUVENILE	Not Given	F
5c71edeaab094442a10da402	NSBS	OTN-Global	NSBS-1248828-2017-08-27-	OTN-GlobalNSBSNSBS-1248828-2017-08-27-	blue shark	Prionace	105801	160424	W	NW	1.37	FORK		JUVENILE	Not	F

- **Task C**
  - Mongo dB compass is installed
  - Imported the csv files to MongoDB using mongoimport command:

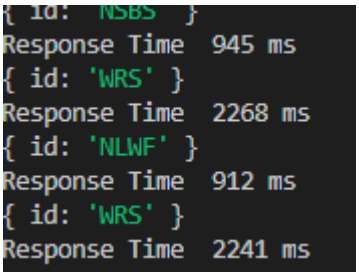


- Mongoimport db —casestudydb –collection animal –type csv –file "C:\Users\sgshi\Desktop\case study data\otn_public_data_dump\animal.csv" –headerline
- Mongoimport db —casestudydb –collection datacenter –type csv –file "C:\Users\sgshi\Desktop\case study data\otn_public_data_dump\datacenter.csv" --headerline
- Mongoimport db —casestudydb –collection detections –type csv –file "C:\Users\sgshi\Desktop\case study data\otn_public_data_dump\detections.csv" –headerline
- And so on.

- Additional method is defined in the code to access mongo DB. Please refer to code provided below.

• Task D

- **Response Time**: For SQL the response time is coming less: 945ms is for SQL and 2268 is for mongo DB. So, SQL is better as it follows a schema.



HTML code:

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://code.jquery.com/jquery-3.3.1.min.js"
        integrity="sha256-FgpCb/KJQlLNfOu91ta32o/NMZxltwRo8QtmkMRdAu8=" crossorigin="anonymous"></script>
    <script src="index.js"></script>
<body onload="getXMLData()">
    <select id="databaseSelect">
        <option value="mongo">Mongodb</option>
        <option value="mysql">mysql</option>
    </select>
    <select id="select">
    </select>
    <button onclick="submit()">Submit</button>
</body>
</html>
```

JavaScript code:

```
var jsonData;
function getXMLData() {
    $.get("http://localhost:9999/code/", function (data) {
        $(".result").html(data);
        console.log("Load was performed.", data);
        $.each(data, function () {
            $("#select").append($("<option/>").val(this.code).text(this.code));
        });
    });
}
function submit() {
    var table = document.getElementsByTagName('table');
    if (table.length > 0) {
        document.getElementsByTagName('body')[0].removeChild(table[0]);
    }
    var e = document.getElementById("databaseSelect");
    var strDatabase = e.options[e.selectedIndex].value;
    var e = document.getElementById("select");
    var strProjectReference = e.options[e.selectedIndex].value;
    if (strDatabase == "mongo") {
        $.get("http://localhost:9999/mongodb/" + strProjectReference, function (data) {
            bulidUI(data[0]);
        });
    }
    if (strDatabase == "mysql") {
        $.get("http://localhost:9999/sql/" + strProjectReference, function (data) {
            bulidUI(data[0]);
            console.log("Load was performed.", data);
        });
    }
}
function bulidUI(data) {

    var col = [];
    for (var i = 0; i < data.length; i++) {
        for (var key in data[i]) {
            if (col.indexOf(key) === -1) {
                col.push(key);
            }
        }
    }
    // CREATE DYNAMIC TABLE.
```

```
    var table = document.createElement("table");

    // CREATE HTML TABLE HEADER ROW USING THE EXTRACTED HEADERS ABOVE.

    var tr = table.insertRow(-1);
    for (var i = 0; i < col.length; i++) {
        var th = document.createElement("th");
        th.innerHTML = col[i];
        tr.appendChild(th);
    }

    // ADD JSON DATA TO THE TABLE AS ROWS.
    for (var i = 0; i < data.length; i++) {

        tr = table.insertRow(-1);

        for (var j = 0; j < col.length; j++) {
            var tabCell = tr.insertCell(-1);
            tabCell.innerHTML = data[i][col[j]];
        }
    }
    document.getElementsByTagName('body')[0].append(table);

}
```

Server connection code/ node js:

```
var express = require('express');
var mysql = require('mysql');
var mongo = require('mongodb');
var server = express();
var port = 9999
server.listen(port, () => {
    console.log('server started', port);
});
server.use(function (req, res, next) {
    res.header("Access-Control-Allow-Origin", "*");
    res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
    next();
});
var con = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: "Shivam1!",
    database: 'casestudydb',
    multipleStatements: true
```

```
});
var url = "mongodb://localhost:27017/casestudydb";
con.connect(function (err) {
    if (err) throw err;
    console.log("Connected to MYSQL!");
});
server.get('/code', (req, res) => {
    console.log(req.params)
    var sql = 'select *  from code';
    con.query(sql, function (err, data) {
        if (err) throw err;
        var result = data
        // console.log("Result: " + JSON.stringify(result));
        res.send(result)
    });
});
server.get('/sql/:id', (req, res) => {
    var inDate = new Date().getTime();
    console.log(req.params)
    var sql = 'select * from animal where animal_project_reference=\'' + req.params.id + '\'; ' +
        'select * from detections where detection_project_reference=\'' + req.params.id + '\'; ' +
        'select * from manmadeplatform where platform_project_reference=\'' + req.params.id + '\'; ' +
        'select * from receivers where deployment_project_reference=\'' + req.params.id + '\'; ' +
        'select * from recovery where recovery_project_reference=\'' + req.params.id + '\'; ' +
        'select * from tagrealeases where release_project_reference=\'' + req.params.id + '\'; ';
    con.query(sql, function (err, data) {
        if (err) throw err;
        var result = data
        console.log('Response Time ', new Date().getTime() - inDate, 'ms');
        res.send(result)
    });
});
server.get('/mongodb/:id', (req, res) => {
    var inDate = new Date().getTime();
    console.log(req.params)
    mongo.connect(url, function (err, db) {
        console.log("Database connected!");
        var dbo = db.db("casestudydb");
        var query = { animal_project_reference: req.params.id };
        var array = [];
        dbo.collection("animal").find(query).toArray(function (err, result) {
            if (err) throw err;
            array.push(result);
        });
        var query1 = { detection_project_referecnce: req.params.id };
        dbo.collection("detections").find(query1).toArray(function (err, result) {
            if (err) throw err;
```

```
            array.push(result);
        });
        var query2 = { platform_project_reference: req.params.id };
        dbo.collection("manmadeplatforms").find(query2).toArray(function (err, result) {
            if (err) throw err;
            array.push(result);
        });
        var query3 = { deployment_project_reference: req.params.id };
        dbo.collection("receivers").find(query3).toArray(function (err, result) {
            if (err) throw err;
            array.push(result);
        });
        var query4 = { recovery_project_reference: req.params.id };
        dbo.collection("recovery").find(query4).toArray(function (err, result) {
            if (err) throw err;
            array.push(result);
        });
        var query5 = { release_project_reference: req.params.id };
        dbo.collection("tagrealeases").find(query5).toArray(function (err, result) {
            if (err) throw err;
            array.push(result);
            db.close();
            console.log('response Time ' + new Date().getTime() - inDate);
            res.send(array)
        });
    });
});
```

- **References**

  - Learn JavaScript: https://www.w3schools.com/js/default.asp
  - Learn Node js: https://www.w3schools.com/nodejs/default.asp
  - Learn HTML: https://www.w3schools.com/html/default.asp
  - Stack overflow: took help from various questions over stack overflow: www.stackoverflow.com
  - YouTube Videos.