Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут”
Кафедра АСОІУ

ЗВІТ

про виконання контрольної роботи
з дисципліни
“Програмне застосування баз данних”
Тема: Аналітичні запити

Прийняв:
Клименко О.М.

Виконав:
студент 2-го курсу
гр. ІП-52 ФІОТ
Набоков Е.М

Київ 2017

Постановка завдання

1 Перепроектування схеми БД та доповнення даними

```
-- create postgres table and fill with data
CREATE TABLE auto_brands (
  id SERIAL PRIMARY KEY,
  brand VARCHAR(80) UNIQUE
);

COPY auto_brands (brand) FROM '/var/lib/postgresql/csvfiles/brands.csv'
  WITH (FORMAT csv);

CREATE TABLE garages (
  id SERIAL PRIMARY KEY,
  num INT NOT NULL UNIQUE
);

COPY garages (num) FROM '/var/lib/postgresql/csvfiles/garages.csv'
  WITH (FORMAT csv);

CREATE TABLE mending_type (
  id SERIAL PRIMARY KEY,
  types_code INT UNIQUE,
  types_name VARCHAR(15) UNIQUE
);

COPY mending_type (types_code, types_name)
  FROM '/var/lib/postgresql/csvfiles/mend_type.csv' WITH (FORMAT csv);

CREATE TABLE trailers_park (
  id SERIAL PRIMARY KEY,
  auto_num VARCHAR(10),
  auto_brand INT references auto_brands(id),
  release_date DATE,
  garage_number INT references garages(id)
);

COPY trailers_park (auto_num, auto_brand, release_date, garage_number)
  FROM '/var/lib/postgresql/csvfiles/trailers_park_data.csv' WITH (FORMAT csv);

CREATE TABLE staff (
  id SERIAL PRIMARY KEY,
  mechanic_id VARCHAR(15) UNIQUE,
  surname VARCHAR(15),
  IPB VARCHAR(20),
  garage_number INT references garages(id),
  start_date DATE,
  end_date DATE,
  salary REAL
);

COPY staff (mechanic_id, surname, ipb, garage_number, start_date, end_date, salary)
  FROM '/var/lib/postgresql/csvfiles/staff.csv' WITH (FORMAT csv);

CREATE TABLE repairs (
  id SERIAL PRIMARY KEY,
  rdate DATE,
  auto_num INT references trailers_park(id),
  mechanic_id INT references staff(id),
  types_code INT references mending_type(id)
);

COPY repairs (rdate, auto_num, mechanic_id, types_code)
  FROM '/var/lib/postgresql/csvfiles/repairs.csv' WITH (FORMAT csv);
```

Копіювання усіх данних до csv з Postgresql

```
COPY (
        SELECT * FROM auto_brands
) TO '/var/lib/postgresql/mydata/auto_brands_copy.csv' CSV HEADER DELIMITER ',';

COPY (
        SELECT * FROM garages
) TO '/var/lib/postgresql/mydata/garages_copy.csv' CSV HEADER DELIMITER ',';

COPY (
        SELECT * FROM mending_type
) TO '/var/lib/postgresql/mydata/mending_type_copy.csv' CSV HEADER DELIMITER ',';

COPY (
        SELECT * FROM trailers_park
) TO '/var/lib/postgresql/mydata/trailers_park_copy.csv' CSV HEADER DELIMITER ',';

COPY (
        SELECT * FROM staff
) TO '/var/lib/postgresql/mydata/staff_copy.csv' CSV HEADER DELIMITER ',';

COPY (
        SELECT * FROM repairs
) TO '/var/lib/postgresql/mydata/repairs_copy.csv' CSV HEADER DELIMITER ',';
```

Копіювання усіх данних з csv до MongoDB

```
-- IMPORT FROM CSV TO MONGO
mongoimport -d admin -c auto_brand_from_csv --type csv --file auto_brands_copy.csv --headerline
mongoimport -d admin -c mending_type_from_csv --type csv --file mending_type_copy.csv --headerline
mongoimport -d admin -c repairs_from_csv --type csv --file repairs_copy.csv --headerline
mongoimport -d admin -c staff_from_csv --type csv --file staff_copy.csv --headerline
mongoimport -d admin -c trailers_park_from_csv --type csv --file trailers_park_copy.csv --headerline
mongoimport -d admin -c garages_from_csv --type csv --file garages_copy.csv --headerline
```

Денормалізація бази данних для спрощенного задання наступних запитів

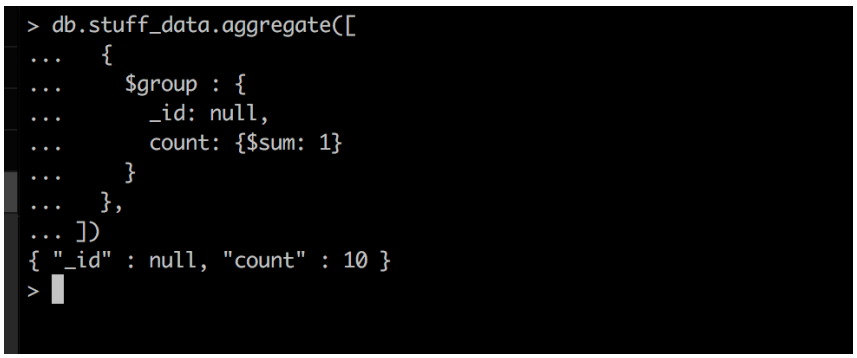
```
-- DENORMALIZATION
db.trailers_park_from_csv.find().snapshot().forEach(
  function (e) {
    // update document, using its own properties
    var new_brand = db.auto_brand_from_csv.findOne({'id': e.auto_brand}).brand;
    e.auto_brand = new_brand;

    var gar_num = db.garages_from_csv.findOne({'id': e.garage_number}).num;
    e.garage_number = gar_num;

    // save the updated document
    db.trailers_park_from_csv.save(e);
  }
)

db.repairs_from_csv.find().snapshot().forEach(
  function (e) {
    // update document, using its own properties
    var type_code = db.mending_type_from_csv.findOne({'id': e.types_code}).types_name;
    e.types_code = type_code;

    // save the updated document
    db.repairs_from_csv.save(e);
  }
)
```

```
db.repairs_from_csv.find().snapshot().forEach(
  function(e) {
    var type_codes = ['Small fixes', 'Middle fixes', 'Large fixes', 'Repair wheels', 'Recolor', 'Improve handle', 'Paint
wheels', 'Paint base', 'Inside cleaning',
    'Outside cleaning', 'Full cleaning'];

    var c1 = Math.floor(Math.random() * type_codes.length);
    var c2 = Math.floor(Math.random() * type_codes.length);
    var c3 = Math.floor(Math.random() * type_codes.length);
    current_code = type_codes[c1];

    e.types_code = [type_codes[c1], type_codes[c2], type_codes[c3]];
    db.repairs_from_csv.save(e);
  }
)
```

2 Групування та агрегування в MongoDB

- **Порахувати кількість документів**

```
db.stuff_data.aggregate([
  {
    $group : {
      _id: null,
     count: {$sum: 1}
    }
  },
])
```



- **Згрупувати за призвіщем та порахувати зустрічаємих (push them to the list)**

```
db.stuff_data.aggregate([
  {
    $group : {
     _id: "$surname",
     ids: {$push: "$_id"}
    }
  },
  {
    $sort : {
      _id: -1
    }
  }
])
```

```
> db.stuff_data.aggregate([
...   {
...     $group : {
...      _id: "$surname",
...      ids: {$push: "$_id"}
...     }
...   },
...   {
...     $sort : {
...       _id: -1
...     }
...   }
... ])
{ "_id" : "Sofie", "ids" : [ 23 ] }
{ "_id" : "Nick", "ids" : [ 22 ] }
{ "_id" : "Mitchel", "ids" : [ 21, 24 ] }
{ "_id" : "Mark", "ids" : [ 20 ] }
{ "_id" : "Louise", "ids" : [ 29 ] }
{ "_id" : "Frank", "ids" : [ 27 ] }
{ "_id" : "Clark", "ids" : [ 26 ] }
{ "_id" : "Bruce", "ids" : [ 25 ] }
{ "_id" : "Brad", "ids" : [ 28 ] }
>
```

- **Зробити обьеднання двох документів**

```
db.repairs_data.aggregate([
  {
   $lookup:
     {
      from: "trailers_park_data",
      localField: "auto_num_id",
      foreignField: "_id",
      as: "auto_num_id"
     }
  },
  {
   $project:
     {
      "_id": "$_id",
      "repair_date": "$rdate",
      "auto_brand": "$auto_num_id.auto_brand",
      "auto_num": "$auto_num_id.auto_num",
      "repairs": "$types_code"
     }
  },
  {$unwind: "$auto_brand"},
  {$unwind: "$auto_num"},
  {
   $sort: { "auto_brand": 1 }
  }
])
```

```
> db.repairs_data.aggregate([
...   {
...     $lookup:
...       {
...         from: "trailers_park_data",
...         localField: "auto_num_id",
...         foreignField: "_id",
...         as: "auto_num_id"
...       }
...   },
...   {
...     $project:
...       {
...         "_id": "$_id",
...         "repair_date": "$rdate",
...         "auto_brand": "$auto_num_id.auto_brand",
...         "auto_num": "$auto_num_id.auto_num",
...         "repairs": "$types_code"
...       }
...   },
...   {$unwind: "$auto_brand"},
...   {$unwind: "$auto_num"},
...   {
...     $sort: { "auto_brand": 1 }
...   }
... ])
{ "_id" : 30, "repair_date" : "19.4.61", "auto_brand" : "BMW", "auto_num" : 6426, "repairs" : [ "Paint wheels", "Large fixes", "Improve handle" ] }
{ "_id" : 36, "repair_date" : "7.3.13", "auto_brand" : "BMW", "auto_num" : 6084, "repairs" : [ "Full cleaning", "Large fixes", "Inside cleaning" ] }
{ "_id" : 35, "repair_date" : "25.6.16", "auto_brand" : "Ferrari", "auto_num" : 1149, "repairs" : [ "Small fixes", "Full cleaning", "Paint base" ] }
{ "_id" : 34, "repair_date" : "17.2.13", "auto_brand" : "Ford", "auto_num" : 6812, "repairs" : [ "Middle fixes", "Paint wheels", "Recolor" ] }
{ "_id" : 31, "repair_date" : "20.1.56", "auto_brand" : "Mercedes", "auto_num" : 9806, "repairs" : [ "Outside cleaning", "Repair wheels", "Inside cleaning" ] }
{ "_id" : 38, "repair_date" : "9.1.51", "auto_brand" : "Mercedes", "auto_num" : 5322, "repairs" : [ "Full cleaning", "Outside cleaning", "Middle fixes" ] }
{ "_id" : 33, "repair_date" : "9.11.12", "auto_brand" : "Schkoda", "auto_num" : 6362, "repairs" : [ "Large fixes", "Improve handle", "Full cleaning" ] }
{ "_id" : 39, "repair_date" : "15.5.9", "auto_brand" : "Subaru", "auto_num" : 1334, "repairs" : [ "Outside cleaning", "Outside cleaning", "Outside cleaning" ] }
{ "_id" : 37, "repair_date" : "9.2.88", "auto_brand" : "Suzuki", "auto_num" : 7293, "repairs" : [ "Repair wheels", "Paint base", "Large fixes" ] }
{ "_id" : 32, "repair_date" : "12.3.50", "auto_brand" : "Volkswagen", "auto_num" : 6533, "repairs" : [ "Middle fixes", "Recolor", "Small fixes" ] }
>
```

- **Вивести механіка та авто. Вивести, які автомобілі були почищені.Порахувати загальну зарплату працівників та розбити на підгрупи**

```
db.repairs_from_csv.aggregate([
  {
   $lookup: {
     from: "trailers_park_from_csv",
     localField: "auto_num",
     foreignField: "id",
     as: "auto"
   }
  },
  {
   $lookup: {
     from: "staff_from_csv",
     localField: "mechanic_id",
     foreignField: "id",
     as: "mechanic"
   }
  },
  {
   $project: {
     "mechanic": "$mechanic.surname",
     "salary": "$mechanic.salary",
     "car": "$auto.auto_brand",
     "was_fully_cleaned": {
       $in : ["Full cleaning", "$types_code"]
     }
   }
  },
  {$unwind: "$mechanic"},
  {$unwind: "$car"},
  {$unwind: "$salary"},
  {
   $bucket:  {
     groupBy: "$salary",
     boundaries: [0, 200, 300, 600, 750, 800, 1000, 1300, 1500, 2000],
     default: "Other",
     output: {
       "count": { $sum: 1 },
```

```
      "mechanics": { $addToSet: "$mechanic" },
      "cars": { $addToSet: "$car" },
      "total_salary": { $sum: "$salary"}
    }
   }
  }
])
```

```
> db.repairs_data.aggregate([
...   {
...     $lookup: {
...       from: "trailers_park_data",
...       localField: "auto_num_id",
...       foreignField: "_id",
...       as: "auto_num_id"
...     }
...   },
...   {
...     $lookup: {
...       from: "stuff_data",
...       localField: "mechanic_id",
...       foreignField: "_id",
...       as: "mechanic_id"
...     }
...   },
...   {
...     $project: {
...       "mechanic": "$mechanic_id.surname",
...       "salary": "$mechanic_id.salary",
...       "car": "$auto_num_id.auto_brand",
...       "was_fully_cleaned": {
...         $in : ["Full cleaning", "$types_code"]
...       }
...     }
...   },
...   {$unwind: "$mechanic"},
...   {$unwind: "$car"},
...   {$unwind: "$salary"},
...   {
...     $bucket:  {
...       groupBy: "$salary",
...       boundaries: [0, 5000, 10000, 20000],
...       default: "Other",
...       output: {
...         "count": { $sum: 1 },
...         "mechanics": { $push: "$mechanic" },
...         "cars": { $push: "$car" },
...         "total_salary": { $sum: "$salary"}
...       }
...     }
...   }
... ])
{ "_id" : 0, "count" : 2, "mechanics" : [ "Mitchel", "Frank" ], "cars" : [ "Ford", "Suzuki" ], "total_salary" : 7853 }
{ "_id" : 5000, "count" : 4, "mechanics" : [ "Mark", "Mitchel", "Brad", "Louise" ], "cars" : [ "BMW", "Mercedes", "Mercedes", "Subaru" ], "total_salary" : 30929 }
{ "_id" : 10000, "count" : 4, "mechanics" : [ "Nick", "Sofie", "Bruce", "Clark" ], "cars" : [ "Volkswagen", "Schkoda", "Ferrari", "BMW" ], "total_salary" : 47615 }
```

3 Виконання аналогу JOIN за допомогою MapReduce

- **Порахувати загальну зарплатню кожного механіка**

```
var mapFunction = function() {
   emit(this._id, this.salary);
};

var reduceFunction = function(id, salary) {
   return Array.sum(salary);
};

db.staff_from_csv.mapReduce(
                mapFunction,
                reduceFunction,
                { out: "map_reduce_result" }
              )
```

```
> var mapFunction = function() {
...     emit(this._id, this.salary);
... };
>
> var reduceFunction = function(id, salary) {
...     return Array.sum(salary);
... };
>
> db.stuff_data.mapReduce(
...                     mapFunction,
...                     reduceFunction,
...                     { out: "map_reduce_result" }
...                   )
{
        "result" : "map_reduce_result",
        "timeMillis" : 48,
        "counts" : {
                "input" : 10,
                "emit" : 10,
                "reduce" : 0,
                "output" : 10
        },
        "ok" : 1
}
>
> db.map_reduce_result.find()
{ "_id" : 20, "value" : 5397 }
{ "_id" : 21, "value" : 8772 }
{ "_id" : 22, "value" : 11838 }
{ "_id" : 23, "value" : 12440 }
{ "_id" : 24, "value" : 3602 }
{ "_id" : 25, "value" : 10682 }
{ "_id" : 26, "value" : 12655 }
{ "_id" : 27, "value" : 4251 }
{ "_id" : 28, "value" : 9147 }
{ "_id" : 29, "value" : 7613 }
>
```

```
var mapperFoo = function(){
  for (var i = 0; i < 50; i++){
    var key = this.id;
    var value = {
       count: 1,
       salary: this.salary
    }
    emit(key, value);
  }
}

var reducerFoo = function(key, values){
  var reduced = {count: 0, salary: 0};

  for (var i = 0; i < values.length; i++){
```

```
    reduced.count += values[i].count;
    reduced.salary += values[i].salary;
  }

  return reduced;
};

var finalizer = function(key, reducer) {
  return reducer;
}

db.staff_from_csv.mapReduce(
  mapperFoo,
  reducerFoo,
  {
    out: {merge: "map_reduce_salary" },
    finalize: finalizer
  }
)
```

```
> var mapFoo = function() {
...   emit(this._id, this.released_date);
... };
>
> var reduceFoo = function(id, date){
...   date_splitted = date.split('.');
...   var day, month, year;
...   day = date_splitted[0];
...   month = date_splitted[1];
...   year = date_splitted[2];
...
...   return (year > 50) ? year : null
... };
>
>
> db.trailers_park_data.mapReduce(
...                     mapFoo,
...                     reduceFoo,
...                     { out: "map_reduce_date" }
...                   )
{
        "result" : "map_reduce_date",
        "timeMillis" : 52,
        "counts" : {
                "input" : 10,
                "emit" : 10,
                "reduce" : 0,
                "output" : 10
        },
        "ok" : 1
}
> db.map_reduce_date.find()
{ "_id" : 0, "value" : "23.1.51" }
{ "_id" : 1, "value" : "14.11.17" }
{ "_id" : 2, "value" : "15.1.22" }
{ "_id" : 3, "value" : "11.1.60" }
{ "_id" : 4, "value" : "14.6.2" }
{ "_id" : 5, "value" : "4.6.72" }
{ "_id" : 6, "value" : "21.8.63" }
{ "_id" : 7, "value" : "28.11.70" }
{ "_id" : 8, "value" : "27.5.30" }
{ "_id" : 9, "value" : "4.5.73" }
>
```