Análise da base de dados Denver

Daniel P. Campagna[1]

[1]Instituto de Computação – Universidade Federal Fluminense (UFF) – Niterói – RJ – Brazil

danielcampagna@id.uff.br

1. Agrupamento por classes de GEO_LAT, GEO_LON definidas pelo valor do meio

A primeira maneira de como podemos agrupar os dados é levando em conta as coordenadas GEO_LAT e GEO_LON. Contudo, devido à natureza contínua, e por possuir múltiplos valores distintos na base, tal agrupamento não trará resultados relevantes para analise. Portanto, faz-se necessário uma tarefa, anterior, de classificar tais dados.

Nesse primeiro instante a classificação considerou os valores do meio de cada campo, i.e. entre o maior e o menor existentes na base. Os resultados foram 19.95215585 e -57.73203075 para GEO_LAT e GEO_LON, respectivamente.

Em seguida, foram geradas as classes A, B, C e D, com valores limites conforme mostra a Tabela 1. Durante a classificação 4239 registros foram removidos por ter valores inválidos (i.e. *String* vazia ou nulo) em GEO_LAT ou GEO_LON.

Classe	Limites
A	GEO_LAT < 19.95215585 e GEO_LON < -57.73203075
B	GEO_LAT < 19.95215585 e GEO_LON >= -57.73203075
C	GEO_LAT >= 19.95215585 e GEO_LON < -57.73203075
D	GEO_LAT >= 19.95215585 e GEO_LON >= -57.73203075

Tabela 1: Definição dos limites das classes A, B, C e D divindo GEO_LAT e GEO_LON no meio.

Por fim, a quantidade de crimes cometidos em cada área pôde ser calculado, e é apresentado na Tabela 2.

Classe	Quantidade de crimes

A	0
B	174
C	508241
D	0

Tabela 2: Quantidade de crimes nas classes A, B, C e D divindo GEO_LAT e GEO_LON no meio.

Os números apresentados na Tabela 2 refletem os resultados obtidos tanto na execução no SQLite3 quanto no MongoDB.

Os comandos usado no MongoDB é apresentado a seguir:

```
use denver

db.crimestst01.drop()

db.createCollection('crimestst01')

db.crimes.find({GEO_LAT: {$nin: ["", null]}, GEO_LON: {$nin: ["", null]}}).forEach(function(doc) { db.crimestst01.insert(doc) });

db.adminCommand('top').totals["denver.crimestst01"]

db.crimes.find().count() - db.crimestst01.find().count()

db.adminCommand('top').totals["denver.crimestst01"]

const max_lat = db.crimestst01.find({}, {"GEO_LAT": 1}).sort({"GEO_LAT":-1}).limit(1)[0]

const min_lat = db.crimestst01.find({}, {"GEO_LAT": 1}).sort({"GEO_LAT":1}).limit(1)[0]

const max_lon = db.crimestst01.find({}, {"GEO_LON": 1}).sort({"GEO_LON":-1}).limit(1)[0]

const min_lon = db.crimestst01.find({}, {"GEO_LON": 1}).sort({"GEO_LON":1}).limit(1)[0]
```

```
const mean_lat = (max_lat['GEO_LAT'] + min_lat['GEO_LAT']) / 2

const mean_lon = (max_lon['GEO_LON'] + min_lon['GEO_LON']) / 2

db.adminCommand('top').totals["denver.crimestst01"]

db.crimestst01.update(
    {},
    { $set: {'region': '' }},
    { multi: true }
)

db.crimestst01.update({
    GEO_LAT: { $lt: mean_lat }, GEO_LON: { $lt: mean_lon }
    }, { $set: {'region':'A'}},
    { multi: true }
)

db.crimestst01.update({
    GEO_LAT: { $lt: mean_lat }, GEO_LON: { $gte: mean_lon }
    }, { $set: {'region':'B'}},
    { multi: true }
)

db.crimestst01.update({
    GEO_LAT: { $gte: mean_lat }, GEO_LON: { $lt: mean_lon }
    }, { $set: {'region':'C'}},
    { multi: true }
)

db.crimestst01.update({
    GEO_LAT: { $gte: mean_lat }, GEO_LON: { $gte: mean_lon }
    }, { $set: {'region':'D'}},
    { multi: true }
)

db.adminCommand('top').totals["denver.crimestst01"]
```

```
db.crimestst01.aggregate({
    $group: {
       _id: '$region',
       count: { $sum: 1 }
    }
});

db.adminCommand('top').totals["denver.crimestst01"]
```

A seguir, o bloco com os comandos usados para classificar e extrair os dados do SQLite3.

```
.open denverdb

.timer on

DROP TABLE IF EXISTS crimestst01;

-- Julian time to Epoch MS
SELECT CAST((julianday('now') - 2440587.5)*86400000 AS INTEGER);

CREATE TABLE crimestst01 AS SELECT * FROM crimes WHERE GEO_LAT IS NOT NULL AND
GEO_LAT != "" AND GEO_LON IS NOT NULL AND GEO_LON != "";

-- Julian time to Epoch MS
SELECT CAST((julianday('now') - 2440587.5)*86400000 AS INTEGER);

SELECT (SELECT count(*) FROM crimes) - (SELECT count(*) FROM crimestst01);

-- Julian time to Epoch MS
SELECT CAST((julianday('now') - 2440587.5)*86400000 AS INTEGER);

BEGIN;

PRAGMA temp_store = 2;

CREATE TEMP TABLE _variables(name TEXT PRIMARY KEY, value REAL);
```

```
INSERT INTO _variables (name) VALUES ('max_lat');

UPDATE _variables SET value = (SELECT (SELECT MAX(GEO_LAT) FROM crimestst01 LIMIT
1)) WHERE name = 'max_lat';


INSERT INTO _variables (name) VALUES ('min_lat');

UPDATE _variables SET value = (SELECT (SELECT MIN(GEO_LAT) FROM crimestst01 LIMIT
1)) WHERE name = 'min_lat';


INSERT INTO _variables (name) VALUES ('max_lon');

UPDATE _variables SET value = (SELECT (SELECT MAX(GEO_LON) FROM crimestst01 LIMIT
1)) WHERE name = 'max_lon';


INSERT INTO _variables (name) VALUES ('min_lon');

UPDATE _variables SET value = (SELECT (SELECT MIN(GEO_LON) FROM crimestst01 LIMIT
1)) WHERE name = 'min_lon';


INSERT INTO _variables (name) VALUES ('mean_lat');

UPDATE _variables SET value = (SELECT ((SELECT value FROM _variables WHERE name =
'max_lat' LIMIT 1) + (SELECT value FROM _variables WHERE name = 'min_lat' LIMIT
1)) / 2) WHERE name = 'mean_lat';


INSERT INTO _variables (name) VALUES ('mean_lon');

UPDATE _variables SET value = (SELECT ((SELECT value FROM _variables WHERE name =
'max_lon' LIMIT 1) + (SELECT value FROM _variables WHERE name = 'min_lon' LIMIT
1)) / 2) WHERE name = 'mean_lon';


-- Julian time to Epoch MS

SELECT CAST((julianday('now') - 2440587.5)*86400000 AS INTEGER);


ALTER TABLE crimestst01 ADD COLUMN region text default "";


UPDATE crimestst01

    SET region = "A"

    WHERE GEO_LAT < (SELECT value FROM _variables WHERE name = "mean_lat" limit
1)

        AND GEO_LON < (SELECT value FROM _variables WHERE name = "mean_lon" limit
1);
```

```
UPDATE crimestst01
    SET region = "B"
    WHERE GEO_LAT < (SELECT value FROM _variables WHERE name = "mean_lat" limit
1)
        AND GEO_LON >= (SELECT value FROM _variables WHERE name = "mean_lon"
limit 1);


UPDATE crimestst01
    SET region = "C"
    WHERE GEO_LAT >= (SELECT value FROM _variables WHERE name = "mean_lat" limit
1) AND GEO_LON < (SELECT value FROM _variables WHERE name = "mean_lon" limit 1);


UPDATE crimestst01
    SET region = "D"
    WHERE GEO_LAT >= (SELECT value FROM _variables WHERE name = "mean_lat" limit
1)
        AND GEO_LON >= (SELECT value FROM _variables WHERE name = "mean_lon"
limit 1);


-- Julian time to Epoch MS
SELECT CAST((julianday('now') - 2440587.5)*86400000 AS INTEGER);


SELECT region, count(*) from crimestst01 group by region;


-- Julian time to Epoch MS
SELECT CAST((julianday('now') - 2440587.5)*86400000 AS INTEGER);


select * from _variables;


END;
```

Os tempos de execução de cada abordagem, em cada etapa é apresentado na Tabela 3.

Etapa	**MongoDB (seg)**	**SQLite3 (seg)**
Importação	0,372	6,351
Cálculo dos pontos de corte	2,568047	0,422
Classificação	25,709819	7,070
Contagem das classes	0,000182	0,339

Tabela 3: Tempo de execução de cada etapa em cada SGBD

2. Agrupamento por classes de GEO_LAT, GEO_LON definidas pelas mediana

É notável, na Tabela 2, que a classificação levando em conta os valores do meio não gerou uma boa distribuição dos dados entre as classes. Portanto, nesta seção o mesmo processamento será feito, contudo considerando os valores medianos de GEO_LAT e GEO_LON, que foram calculados, respectivamente, 39,7397729 e -104.9824294.

Classe	Limites
A	GEO_LAT < 39,7397729 e GEO_LON < -104.9824294
B	GEO_LAT < 39,7397729 e GEO_LON >= -104.9824294
C	GEO_LAT >= 39,7397729 e GEO_LON < -104.9824294
D	GEO_LAT >= 39,7397729 e GEO_LON >= -104.9824294

Tabela 4: Definição dos limites das classes A, B, C e D divindo GEO_LAT e GEO_LON pela mediana

A quantidade de crimes cometidos em cada área é apresentado na Tabela 5.

Classe	Quantidade de crimes
A	137943
B	116234
C	139497
D	114741

Tabela 5: Quantidade de crimes nas classes A, B, C e D divindo GEO_LAT e GEO_LON pela mediana.

Os comandos usado no MongoDB é apresentado a seguir:

```
use denver


db.crimestst02.drop()


db.createCollection('crimestst02')


db.crimes.find({GEO_LAT:    {$nin:    ["",    null]},    GEO_LON:    {$nin:    ["",
null]}}).forEach(function(doc) { db.crimestst02.insert(doc) });
```

```
db.adminCommand('top').totals["denver.crimestst02"]


db.crimes.find().count() - db.crimestst02.find().count()


db.adminCommand('top').totals["denver.crimestst02"]


db.crimestst02.createIndex({ GEO_LAT:1 })


db.crimestst02.createIndex({ GEO_LON:1 })


const        median_lat        =        db.crimestst02.find({},        {GEO_LAT:
1}).sort( {"GEO_LAT":1} ).skip(db.crimestst02.count() / 2 - 1).limit(1)[0]


const        median_lon        =        db.crimestst02.find({},        {GEO_LON:
1}).sort( {"GEO_LON":1} ).skip(db.crimestst02.count() / 2 - 1).limit(1)[0]


db.adminCommand('top').totals["denver.crimestst02"]


db.crimestst02.update(
    {},
    { $set: {'region': '' }},
    { multi: true }
)


db.crimestst02.update({
    GEO_LAT:    {    $lt:    median_lat['GEO_LAT']    },    GEO_LON:    {    $lt:
median_lon['GEO_LON'] }
    }, { $set: {'region':'A'}},
    { multi: true }
)


db.crimestst02.update({
    GEO_LAT:    {    $lt:    median_lat['GEO_LAT']    },    GEO_LON:    {    $gte:
median_lon['GEO_LON'] }
    }, { $set: {'region':'B'}},
    { multi: true }
)
```

```
db.crimestst02.update({
    GEO_LAT:    {    $gte:    median_lat['GEO_LAT']    },    GEO_LON:    {    $lt:
median_lon['GEO_LON'] }
    }, { $set: {'region':'C'}},
    { multi: true }
)


db.crimestst02.update({
    GEO_LAT:    {    $gte:    median_lat['GEO_LAT']    },    GEO_LON:    {    $gte:
median_lon['GEO_LON'] }
    }, { $set: {'region':'D'}},
    { multi: true }
)


db.adminCommand('top').totals["denver.crimestst02"]


db.crimestst02.aggregate({
    $group: {
       _id: '$region',
       count: { $sum: 1 }
    }
});


db.adminCommand('top').totals["denver.crimestst02"]
```

A seguir, o bloco com os comandos usados para classificar e extrair os dados do SQLite3.

```
.open denverdb


.timer on


DROP TABLE IF EXISTS crimestst02;


-- Julian time to Epoch MS
SELECT CAST((julianday('now') - 2440587.5)*86400000 AS INTEGER);


CREATE TABLE crimestst02 AS SELECT * FROM crimes WHERE GEO_LAT IS NOT NULL AND
GEO_LAT != "" AND GEO_LON IS NOT NULL AND GEO_LON != "";
```

```
-- Julian time to Epoch MS
SELECT CAST((julianday('now') - 2440587.5)*86400000 AS INTEGER);


SELECT (SELECT count(*) FROM crimes) - (SELECT count(*) FROM crimestst02);


BEGIN;


PRAGMA temp_store = 2;


CREATE TEMP TABLE _variables(name TEXT PRIMARY KEY, value REAL);


INSERT INTO _variables (name) VALUES ('median_lat');
UPDATE _variables SET value = (SELECT GEO_LAT FROM crimestst02 ORDER BY GEO_LAT
LIMIT 1 OFFSET (SELECT COUNT(*) FROM crimestst02) / 2) WHERE name = 'median_lat';


INSERT INTO _variables (name) VALUES ('median_lon');
UPDATE _variables SET value = (SELECT GEO_LON FROM crimestst02 ORDER BY GEO_LON
LIMIT 1 OFFSET (SELECT COUNT(*) FROM crimestst02) / 2) WHERE name = 'median_lon';


-- Julian time to Epoch MS
SELECT CAST((julianday('now') - 2440587.5)*86400000 AS INTEGER);


ALTER TABLE crimestst02 ADD COLUMN region text default "";


UPDATE crimestst02
    SET region = "A"
    WHERE GEO_LAT < (SELECT value FROM _variables WHERE name = "median_lat" limit
1)
        AND GEO_LON < (SELECT value FROM _variables WHERE name = "median_lon"
limit 1);


UPDATE crimestst02
    SET region = "B"
    WHERE GEO_LAT < (SELECT value FROM _variables WHERE name = "median_lat" limit
1)
        AND GEO_LON >= (SELECT value FROM _variables WHERE name = "median_lon"
limit 1);
```

```
UPDATE crimestst02
    SET region = "C"
    WHERE GEO_LAT >= (SELECT value FROM _variables WHERE name = "median_lat"
limit 1) AND GEO_LON < (SELECT value FROM _variables WHERE name = "median_lon"
limit 1);


UPDATE crimestst02
    SET region = "D"
    WHERE GEO_LAT >= (SELECT value FROM _variables WHERE name = "median_lat"
limit 1)
        AND GEO_LON >= (SELECT value FROM _variables WHERE name = "median_lon"
limit 1);


-- Julian time to Epoch MS
SELECT CAST((julianday('now') - 2440587.5)*86400000 AS INTEGER);


SELECT region, count(*) from crimestst02 group by region;


-- Julian time to Epoch MS
SELECT CAST((julianday('now') - 2440587.5)*86400000 AS INTEGER);


select * from _variables;


END;
```

Os tempos de execução de cada abordagem, em cada etapa é apresentado na Tabela 3.

Etapa	MongoDB (seg)	SQLite3 (seg)
Importação	0,372	6,351
Cálculo dos pontos de corte	4,773865	2,259
Classificação	35,044058	8,314
Contagem das classes	0,000144	0,288

Tabela 6: Tempo de execução de cada etapa em cada SGBD

3. Agrupando pela coluna DISTRICT_ID

Ao realizar um agrupamento pelo campo DISTRICT_ID o seguinte resultado é obtido.

DISTRICT_ID	Quantidade de crimes
1	90223
2	75085
3	113404
4	72442
5	52311
6	103229
7	5960
Total	512654

Tabela 7: Quantidade de crimes agrupando por DISTRICT_ID.

Nesse agrupamento, o distrito de ID 3 teve mais crimes.

4. Agrupando pela coluna PRECINCT_ID

Ao realizar um agrupamento pelo campo PRECINCT_ID o seguinte resultado é obtido.

PRECINCT_ID	Quantidade de crimes
111	14221
112	10445
113	14047
121	12541
122	18440
123	20500
211	15357
212	11492
213	10579
221	11160
222	9976
223	16550
311	23421

(continua)

(continuação)

312	16046
313	15006
314	8792
321	10795
322	13225
323	14724
324	11360
411	12789
412	19703
421	13333
422	12797
423	13820
511	10390
512	15267
521	13860
522	6827
523	5967
611	34794
612	19765
621	17230
622	13980
623	17495
759	5960
Total	512654

Tabela 7: Quantidade de crimes agrupando por PRECINCT_ID.

Nesse agrupamento, o arredor com ID 611 teve mais crimes (34794).

5. Agrupando pela coluna NEIGHBORHOOD_ID

Ao realizar um agrupamento pelo campo NEIGHBORHOOD_ID o seguinte resultado é obtido.

NEIGHBORHOOD_ID	Quantidade de crimes
athmar-park	7279
auraria	5089
baker	14042
barnum	6077
barnum-west	3250
bear-valley	3497
belcaro	2583
berkeley	4981
capitol-hill	17697
cbd	19024
chaffee-park	2607
cheesman-park	7546
cherry-creek	5981
city-park	2686
city-park-west	6459
civic-center	12324
clayton	3611
cole	4020
college-view-south-platte	6609
congress-park	5499
cory-merrill	2644
country-club	1229
dia	7807

(continua)

(continuação)

east-colfax	13374
elyria-swansea	7699
five-points	26930
fort-logan	2376
gateway-green-valley-ranch	11003
globeville	7898
goldsmith	4331
hale	3730
hampden	8146
hampden-south	8898
harvey-park	5987
harvey-park-south	4147
highland	9263
hilltop	3064
indian-creek	569
jefferson-park	4691
kennedy	2255
lincoln-park	13755
lowry-field	4114
mar-lee	7248
marston	3800
montbello	17441
montclair	4290
north-capitol-hill	9434
north-park-hill	3756

(continua)

(continuação)

northeast-park-hill	9055
overland	4753
platt-park	3265
regis	2698
rosedale	1732
ruby-hill	6118
skyland	2067
sloan-lake	3991
south-park-hill	4003
southmoor-park	2454
speer	7918
stapleton	22072
sun-valley	4432
sunnyside	6438
union-station	12093
university	4136
university-hills	4277
university-park	3564
valverde	3658
villa-park	7214
virginia-village	5645
washington-park	3131
washington-park-west	4365
washington-virginia-vale	7434
wellshire	628

(continua)

(continuação)

west-colfax	10464
west-highland	4785
westwood	12601
whittier	3093
windsor	3830
Total	512654

Tabela 7: Quantidade de crimes agrupando por NEIGHBORHOOD_ID.

Nesse agrupamento, o distrito de ID five-points teve mais crimes, 26930.

6. Análise e discussão dos resultados

Ingestão dos dados - conforme se observa nas Tabela 3 e 6, nenhuma das abordagens, nos sistemas usados, apresentou diferença durante a fase de ingestão. Ambas ferramentas apresentam comandos para importar os dados no formato *csv*.

Linguagem de Consulta - devido maior familiaridade com a linguagem SQL, a elaboração das consultas no SQLite3 se deu de maneira mais rápida, fácil e compreensível. Contudo, a linguagem utilizada pelo MongoDB facilita a criação, a compreensão e o uso de variáveis, que foi útil para a etapa de classificação.

Tempo de Processamento - nota-se que ambos sistemas apresentam vantagens e desvantagens, dependendo da etapa à qual se está processando. As etapas de Importação e Contagem dos dados foi resolvida mais rapidamente no sistema MongoDB. Já as etapas de Cálculo dos pontos de corte e Classificação somam para o SQLite3. Vale ressaltar, ainda, que o MongoDB criou tabelas de índices para que pudesse processar estas duas últimas etapas, dando problema de caso no isso não fosse feito. Como consequência, uma análise dos resultados mais justa levaria em conta futuras inserções (que contaria com operações tanto para inserir o novo dado quanto para gerenciar as tabelas de índice. Outro ponto que poderia ser considerado era o custo de espaço utilizado por ambas abordagens.