

Trabalho de Otimização de Consultas no Cassandra em Aplicações Analíticas

Pedro Igor, Voltan, Delfino, Façanha

1

1. Query 1.1

1.1. Query para keyspace normalizado

```
1 | select sum(o.extendedprice*o.discount) as revenue
2 | from cnssb_cf.orderFact o
3 | inner join cnssb_cf.date d on o.dblinenumber = d.dblinenumber
4 | where d.year = 1993
5 | and o.discount between 1 and 3
6 | and o.quantity < 25;
```

Tempos das primeiras 5 consultas (em segundos)

1. 16.287
2. 15.015
3. 14.709
4. 14.272
5. 14.689

1.2. Query não otimizada para keyspace desnormalizado

```
1 | select sum(extendedprice*discount) as revenue
2 | from nlineorder
3 | where year = 1993
4 | and discount between 1 and 3
5 | and quantity < 25;
```

Tempos das primeiras 5 consultas (em segundos)

1. 7.056
2. 7.502
3. 6.744
4. 6.844
5. 7.167

1.3. Tratamento da query no keyspace desnormalizado

```
1 | create materialized view v1 as
2 | select discount, year, quantity, orderkey, linenumber,
   | extendedprice
3 | from nlineorder
4 | where year = 1993
5 | and quantity < 25
6 | and orderkey is not null
7 | and linenumber is not null
8 | and discount is not null
9 | primary key (discount, quantity, orderkey, linenumber, year);
```

Criou-se uma visão materializada em que antecipam-se as restrições *where year = 1993* e *where quantity > 25* da query 1.1. A coluna *discount* foi adicionada como primeira partition key devido a restrição *where discount between 1 and 3*.

Além disso, como a keyword *between* não é aceita em CQL, a query foi adaptada para uma equivalente que faz uso da keyword *in*.

1.4. Query otimizada para keyspace desnormalizado

```
1 | select sum(extendedprice*discount) as revenue
2 | from v1
3 | where discount in (1,2,3);
```

Tempos das primeiras 5 consultas (em segundos)

1. 0.874
2. 0.637
3. 0.587
4. 0.578
5. 0.592

2. Query 2.1

2.1. Query para keyspace normalizado

```
1 | select sum(o.revenue), d.year, p.brand1
2 | from cnssb_cf.orderFact o
3 | inner join cnssb_cf.date d on o.dblinenum = d.dblinenum
4 | inner join cnssb_cf.part p on o.dblinenum = p.dblinenum
5 | inner join cnssb_cf.supplier s on o.dblinenum = s.dblinenum
6 | where p.category = 'MFGR#12'
7 | and s.supregion = 'AMERICA'
8 | group by d.year, p.brand1
9 | order by d.year, p.brand1;
```

Tempos das primeiras 5 consultas (em segundos)

1. 35.303
2. 33.849
3. 33.796
4. 33.934
5. 33.483

2.2. Query não otimizada para keyspace desnormalizado

```
1 | select sum(revenue), year, brand1
2 | from nlineorder
3 | where category = 'MFGR#12'
4 | and supregion = 'AMERICA'
5 | group by year, brand1
6 | order by year, brand1;
```

Tempos das primeiras 5 consultas (em segundos)

1. 47.462
2. 46.744
3. 47.026
4. 47.478
5. 45.609

2.3. Tratamento da query no keyspace desnormalizado

```
1 | create materialized view v2 as
2 | select revenue, year, quantity, orderkey, linenumber, brand1
3 | from nlineorder
4 | where quantity is not null
5 | and year is not null
6 | and orderkey is not null
7 | and linenumber is not null
8 | and brand1 is not null
9 | and category = 'MFGR#12'
10 | and suppreion = 'AMERICA'
11 | primary key ((year, brand1), quantity, orderkey, linenumber);
```

Criou-se uma visão materializada em que antecipam-se as restrições *where category = 'MFGR#12'* e *where suppreion = 'AMERICA'* da query 2.1. As colunas *year* e *brand1* foram selecionadas como composite partition key para que em cada operação *group by* seja feita apenas uma agregação por partição.

2.4. Query otimizada para keyspace desnormalizado

```
1 | select sum(revenue), year, brand1
2 | from v2
3 | group by year, brand1
4 | order by year, brand1;
```

Tempos das primeiras 5 consultas (em segundos)

1. 1.076
2. 1.079
3. 1.029
4. 0.974
5. 1.005

3. Query 3.1

3.1. Query para keyspace normalizado

```
1 | select c.nation, s.suppnation, d.year, sum(o.revenue)
2 | as revenue
3 | from cnssb_cf.orderFact o
4 | inner join cnssb_cf.date d on o.dblinenumbr = d.dblinenumbr
5 | inner join cnssb_cf.customer c on o.dblinenumbr = c.dblinenumbr
6 | inner join cnssb_cf.supplier s on o.dblinenumbr = s.dblinenumbr
7 | where c.region = 'ASIA' and s.suppreion = 'ASIA'
8 | and d.year >= 1992 and d.year <= 1997
9 | group by c.nation, s.suppnation, d.year
10 | order by d.year asc, revenue desc;
```

Tempos das primeiras 5 consultas (em segundos)

1. 43.002
2. 39.162
3. 38.912
4. 40.469
5. 39.406

3.2. Query não otimizada para keyspace desnormalizado

```
1 | select nation, suppnation, year, sum(revenue) as revenue
2 | from nlineorder
3 | where region = 'ASIA' and suppreregion = 'ASIA'
4 | and year >= 1992 and year <= 1997
5 | group by nation, suppnation, year
6 | order by year asc, revenue desc;
```

Tempos das primeiras 5 consultas (em segundos)

1. 60.097
2. 45.778
3. 40.963
4. 40.619
5. 41.057

3.3. Tratamento da query no keyspace desnormalizado

```
1 | create materialized view v3 as
2 | select nation, suppnation, year, quantity, orderkey, linenumbr,
   | revenue
3 | from nlineorder
4 | where quantity is not null
5 | and year is not null
6 | and orderkey is not null
7 | and linenumbr is not null
8 | and nation is not null
9 | and suppnation is not null
10 | and region = 'ASIA'
11 | and suppreregion = 'ASIA'
12 | and year >= 1992
13 | and year <= 1997
14 | primary key ((nation,year), quantity, orderkey, linenumbr);
```

Criou-se uma visão materializada em que antecipam-se as restrições *where region = 'ASIA' and suppreregion = 'ASIA' and year >= 1992 and year <= 1997* da query 3.1. As colunas *nation* e *year* foram selecionadas como composite partition key para que em cada operação *group by* sejam feitas agregações apenas para a coluna *suppnation* dentro de uma mesma partição.

3.4. Query otimizada para keyspace desnormalizado

```
1 | select nation, suppnation, year, sum(revenue) as revenue
2 | from v3
3 | group by nation, year, suppnation
4 | order by year asc, revenue desc;
```

Tempos das primeiras 5 consultas (em segundos)

1. 1.778
2. 1.349
3. 1.452
4. 1.329
5. 1.706

4. Query 4.1

4.1. Query para keyspace normalizado

```
1 | select d.year, c.nation, sum(o.revenue - o.supplycost) as profit
2 | from cnssb_cf.orderFact o
3 | inner join cnssb_cf.date d on o.dblinenum = d.dblinenum
4 | inner join cnssb_cf.customer c on o.dblinenum = c.dblinenum
5 | inner join cnssb_cf.supplier s on o.dblinenum = s.dblinenum
6 | inner join cnssb_cf.part p on o.dblinenum = p.dblinenum
7 | where c.region = 'AMERICA'
8 | and s.suppreion = 'AMERICA'
9 | and (p.mfgr = 'MFGR#1' or p.mfgr = 'MFGR#2')
10 | group by d.year, c.nation
11 | order by d.year, c.nation;
```

Tempos das primeiras 5 consultas (em segundos)

1. 60.830
2. 59.294
3. 60.005
4. 59.204
5. 57.493

4.2. Query não otimizada para keyspace desnormalizado

```
1 | select year, nation, sum(revenue - supplycost) as profit
2 | from nlineorder
3 | where region = 'AMERICA'
4 | and suppreion = 'AMERICA'
5 | and (mfgr = 'MFGR#1' or mfgr = 'MFGR#2')
6 | group by year, nation
7 | order by year, nation;
```

Tempos das primeiras 5 consultas (em segundos)

1. 46.181
2. 42.990
3. 44.039
4. 41.293
5. 41.012

4.3. Tratamento da query no keyspace desnormalizado

```
1 | create materialized view v4 as
2 | select nation, year, quantity, orderkey, linenum, revenue,
   | supplycost
3 | from nlineorder
4 | where quantity is not null
5 | and year is not null
6 | and orderkey is not null
7 | and linenum is not null
8 | and nation is not null
9 | and region = 'AMERICA'
10 | and suppreion = 'AMERICA'
11 | and mfgr <= 'MFGR#2'
12 | primary key ((year, nation), quantity, orderkey, linenum);
```

Criou-se uma visão materializada em que antecipam-se as restrições *where region = 'AMERICA' and suppreregion = 'AMERICA' and (mfgr = 'MFGR#1' or mfgr = 'MFGR#2')* da query 4.1. As colunas *nation* e *year* foram selecionadas como composite partition key para que em cada operação *group by* seja feita apenas uma agregação por partição.

4.4. Query otimizada para keyspace desnormalizado

```
1 | select year, nation, sum(revenue - supplycost) as profit
2 | from v4
3 | group by year, nation
4 | order by year, nation;
```

Tempos das primeiras 5 consultas (em segundos)

1. 0.985
2. 0.851
3. 0.794
4. 0.802
5. 0.810

5. Comparação dos tempos obtidos

Em todas as consultas realizadas, foi possível diminuir de vinte a sessenta vezes o tempo das consultas otimizadas sobre visões materializadas no modelo desnormalizado em relação às consultas não otimizadas. Vale destacar que nem sempre as consultas não otimizadas sobre o modelo desnormalizado obtiveram tempos menores do que as consultas sobre o modelo normalizado.

6. Observações finais

- Mesmo com as otimizações, não são todas as queries que podem ser executadas tanto diretamente via CQL quanto via Spark devido a existência de agregações que fazem uso de *group by* ou devido ao uso de *order by*.
- Para consistência das comparações de tempo de consulta, todas elas foram realizadas pelo Spark.
- Vale ressaltar que uma query otimizada para o keyspace desnormalizado pode ter resultados diferentes se realizada no spark ou diretamente em cql. Isso ocorre porque os resultados podem sofrer overflow no tipo int (4 bytes) quando realizadas no Cassandra, enquanto o Spark faz um tratamento automático do overflow, exibindo o resultado correto.
- Não foram utilizados índices para as otimizações devido a elevada cardinalidade de algumas colunas, o que resultaria em um elevado overhead para executar as queries. Além disso, cada uma das consultas envolveriam o uso de múltiplos índices, o que também prejudica a performance. Por fim, para otimização, pode-se aproveitar a estrutura do modelo colunar e criar visões materializadas tais que as colunas selecionadas como partition key e clustering key ordenam os dados de maneira conveniente para cada query, o que substitui a função do índice.