Verificação da otimização das consultas pela utilização de índices

Para tirar proveito dos índices extensivamente, devemos considerar como escrever as nossas consultas e conferir se o servidor MySQL realmente está a otimizá-las, usando por exemplo a declaração EXPLAIN.

No EXPLAIN, são dadas dicas importantes para a otimização das queries, também designado como plano de execução da querie.

como usar explain no mysql

```
SELECT designacao, preco_unitario
from medicamento
where preco_unitario in (select min(preco_unitario) from medicamento)
```

designacao	preco_unitario
Aspirina	9.67

A sua consulta SQL foi executada com êxito.

```
EXPLAIN SELECT designacao, preco_unitario from medicamento where preco_unitario in (select min(preco_unitario) from medicamento)
```

[Edit inline] [Edita] [Saltar Explicar SQL] [Analyze Explain at mariadb.org] [

+ Opções

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	medicamento	*NULL*	ALL	*NULL*	*NULL*	*NULL*	*NULL*	6	100.00	Using where
2	SUBQUERY	medicamento	*NULL*	ALL	*NULL*	*NULL*	*NULL*	*NULL*	6	100.00	*NULL*

Linhas examinadas=6x6

Colunas do retorno do EXPLAIN SELECT

Id: Número sequencial que identifica as consultas dentro do select.

Select_type: Tipo de cláusula SQL:

- SIMPLE (Select simples)
- PRIMARY (Select mais externa)
- UNION (Segunda Select ou Select proveniente do UNION)
- DEPENDING UNION (segunda select ou select proveniente do UNION)
- SUBQUERY (primeiro select encadeado- subquery)
- DEPENDEND SUBQUERY (primeiro select encadeado- subquery)
- DERIVES (select da tabela derivada – sunquery da CLAUSULA FROM)

TABLE: Tabela examinada.

TYPE: Descreve como as tabelas são unidas (tipo de JOIN):

(Do melhor valor para o pior)

- SYSTEM (tabela que só tem uma linha)
- CONST (tabela que tem no máximo uma linha coincidente. São constantes)
- EQ_REF (todas as partes das chaves são usadas para combinação de registos)
- REF (idem ao EQ_REF, mas com índices não nulos)
- REF_OR_NULL (idem ao REF, mas com busca IS NULL)
- RANGE (faixa de busca quando o campo é comparado a uma constante)
- INDEX (quando a consulta só usa colunas que são parte de um índice)
- ALL ("varredura" completa na tabela para a busca de registos)

POSSIBLE_KEYS: Sugestão de índices a serem utilizados

(se o valor retornado é NULL, não há índices possíveis, e o MySQL terá que buscar na tabela toda)

KEY: Chave em utilização na consulta

(key mostra qual das possible_keys foi escolhida para acelerar a query. Se é NULL, não há índice possível para otimizar esta query.)

KEY_LEN: Tamanho da chave do campo KEY-> o mysql pode usar índices em colunas com mais eficiência se forem declarados com o mesmo tipo e tamanho

REF: Colunas utilizadas pela chave do campo KEY

ROWS: Quantidade de linhas que o sql acredita serem analisadas para gerar a consulta (Quanto maior, pior)

FILTERED: Percentagem relativa ao total de linhas filtradas/examinadas

No exemplo da página anterior filtered é de 100% , logo 6x100%=6 , no total das duas linhas foram examinadas 6x6, correspondente ao produto da coluna row, de cada linha. Se filtered , na 1.ª linha fosse de 50% teríamos 6x50% =6 de linhas examinadas na 1.ª linha e 6x100%=6 na 2.ª linha , o que dava um total de 3x6=18 linhas examinadas.

EXTRA: Sugestão de índices a serem utilizados (diversas informações adicionais).

- Distinct (Termina a busca quando encontra o 1.º registo coincidente)
- Nor exists (idem ao distinct, mas com LEFT JOIN)
- Range checked for eack record (index map: #)

  (O Mysql não encontrou um bom índice para usar)
- Using filesort (pesquisa extra na tabela apra realizar a ordem de classificação)
- Using index (Recuperação feita apenas com índices)

- Using temporary (Utyilização de tabelas temporárias para realizar a busca)
- Using where (Tipo de restrição na busca de registos)

Tentar evitar ao máximo using filesort e using temporary, típicos de queries com ORDER BY e GROUP BY.

Dicas importantes para SELECT

Cardinalidade do índice - É o número de valores únicos que o índice consegue apontar

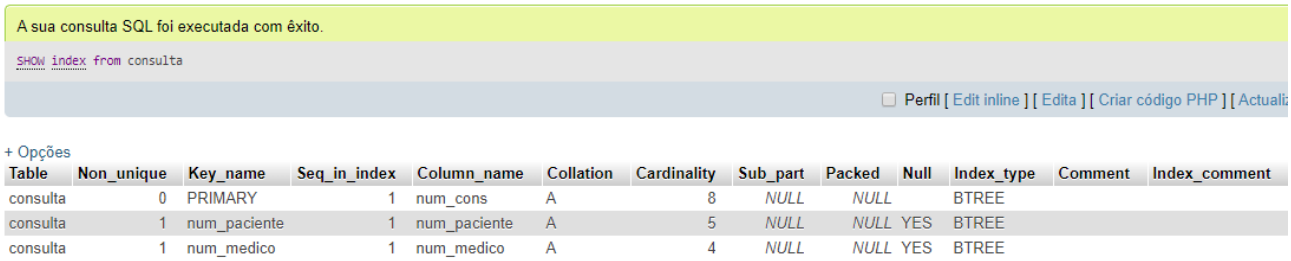
- Quanto maior a cardinalidade, mais provável será encontrar os valores de interesse com menos leituras do índice, pois ele aponta para os registros com maior precisão;
- A chave primária é o índice de maior cardinalidade possível, pois cada entrada no índice aponta para um registo em particular;
- Utilizar SHOW INDEX FROM para verificar se a referência de cardinalidade (coluna cardinality) está atualizada.

Sintaxe:

Show index from <nome_tabela>

Exemplo:

SHOW INDEX FROM consulta



A sua consulta SQL foi executada com êxito.

```
SHOW index from consulta
```

Perfil [Edit inline] [Edita] [Criar código PHP] [Actuali

+ Opções

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment
consulta	0	PRIMARY	1	num_cons	A	8	NULL	NULL		BTREE		
consulta	1	num_paciente	1	num_paciente	A	5	NULL	NULL	YES	BTREE		
consulta	1	num_medico	1	num_medico	A	4	NULL	NULL	YES	BTREE		

ANALYZE TABLE: Função que atualizará as estatísticas sobre a tabela. Tais estatísticas são utilizadas pelo MySQL para seleção de como e qual índice pode ser utilizado.

Dicas importantes para SELECT

- Se não forem utilizadas colunas de todas as tabelas , o MySQL irá parar a "varredura" das tabelas não usadas logo que encontrar a 1.ª correspondência.

```
SELECT DISTINCT t1.a
   FROM t1,t2
   WHERE t1.a=t2.a;
```

- Se estivermos a unir muitas tabelas e as colunas relativas ao ORDER BY não são todas da 1.ª tabela.
- O ideal é que os campos do ORDER BY pertençam à 1.ª tabela.

- Tentar usar campos no ORDER BY que façam parte de índices. Isso evita um processo de ordenação por parte do MySQL.
- Internamente, o MySQL ordena as consultas GROUP BY como se fosse o ORDER BY. Para que só o agrupamento aconteça, incluir no código um ORDER BY NULL.

Dicas importantes para LIMIT

- O MySQL vai buscar a quantidade de registos estipulados no LIMIT e só depois vai executar outras funções (ORDER BY ou GROUP BY, por exemplo).

Dicas importantes para INSERT

- Na importação de um grande volume de dados, também é válido desativar os índices com:

ALTER TABLE < nome:tabela > DISABLE/ENABLE KEYS

Dicas importantes para UPDATE

- Deixar para alterar todo o registo de uma só vez

Dicas importantes para DELETE

Se for para "limpar" uma tabela, usar:
TRUNCATE TABLE

OUTRAS DICAS IMPORTANTES

Se nas buscas usamos uma determinada ordem dos campos, mas na tabela esses capos estão numa ordem diferente, devemos mudar a ordem dos campos com:
ALTER TABLE…ORDER BY c1, c2…

Relembrar que: Os índices ocupam espaço em disco, e em memória, ao serem utilizados. Se um índice fica muito grande, o MySQL pode escolher não utilizá-lo, pois o peso de carregá-lo na memória é maior do que varrer a tabela inteira!
Cada inserção, remoção ou atualização da tabela gera a necessidade de atualização dos índices.
Supondo que a aplicação que usa uma tabela de clientes faz bastantes buscas por nome.

Vamos criar um índice na coluna nome, indexando apenas os 6 primeiros carateres, número suficiente de carateres para uma pesquisa efetiva (isto porque indexar o nome inteiro geraria um índice muito grande - quanto mais carateres indexados, maior o índice, e mais memória necessária para carregá-lo).

```
ALTER TABLE Clientes ADD INDEX(nome( 6));
Query OK, 0 rows affected ( 3 min 7.77 sec)
```

Suponhamos que queremos selecionar todos os clientes cujo nome é "Joana"

```
SELECT * FROM Clientes WHERE nome LIKE "Joana%"
```

- **Sem o índice:**

```
EXPLAIN SELECT * FROM Clientes WHERE nome LIKE 'Joana%';
+----+-------------+----------+------+---------------+------+---------+------+
| id |select_type |table |type |possible_keys |key |key_len |ref |rows |Extra|
+----+-------------+----------+------+---------------+------+---------+------+
| 1  |SIMPLE |Clientes |ALL |NULL |NULL |NULL |NULL |39173588 |Using where |
+----+-------------+----------+------+---------------+------+---------+------+
```

- **Com o índice:**

```
EXPLAIN SELECT * FROM Clientes WHERE nome LIKE 'Joana%';
+----+-------------+----------+-------+---------------+-----+---------+
| id |select_type|table|type|possible_keys|key|key_len|ref|rows|Extra|
+----+-------------+----------+-------+---------------+-----+---------+
| 1  |SIMPLE|Clientes|range|nome|nome|8|NULL|233130|Using where|
+----+-------------+----------+-------+---------------+-----+---------+
```

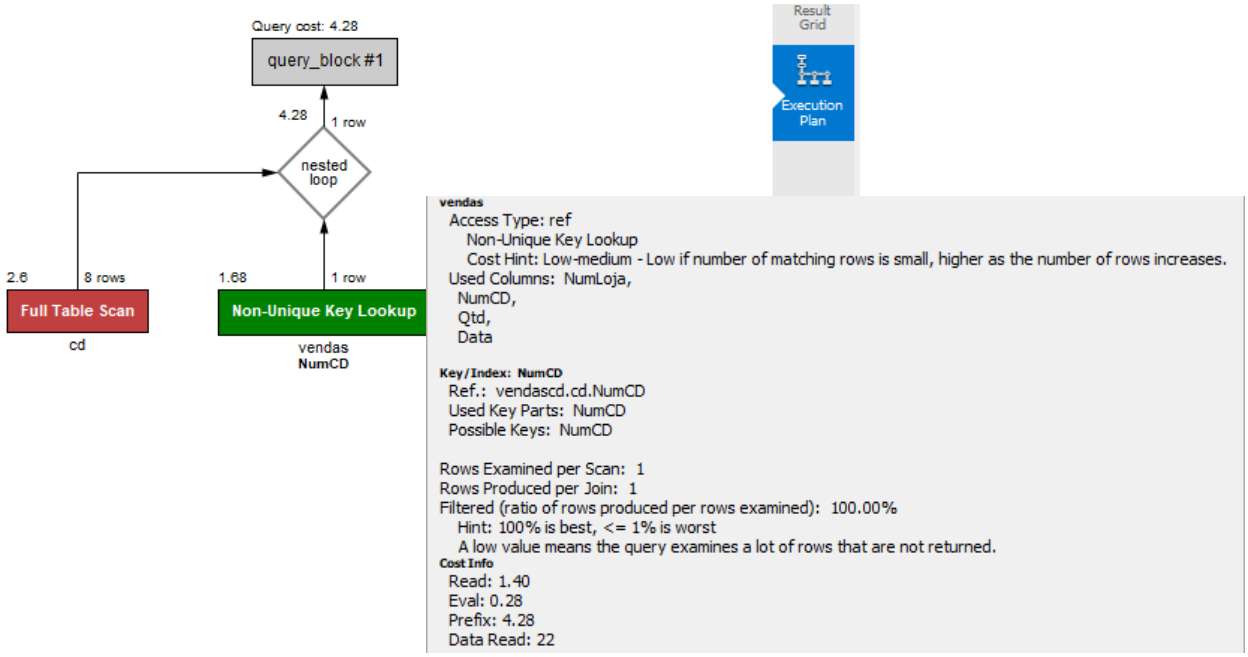
Execução sem o índice:

```
SELECT * FROM Clientes WHERE nome LIKE 'Joana%';
---
119200 rows in set (40.75 sec)
```

Execução com o índice:

```
SELECT * FROM Clientes WHERE nome LIKE 'Joana%';
---
119200 rows in set (9.61 sec)
```

Plano de execução (visual explain) de uma querie no MySQLWorkBench



EXERCÍCIOS

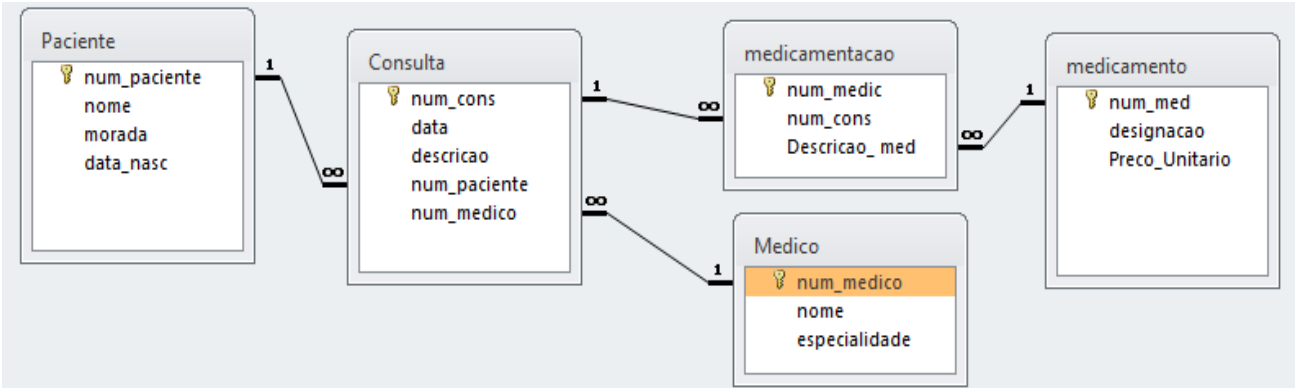


Fig. 1 – *Tabelas e relações da base de dados*

Tendo em conta que:
num_med, num_cons, num_medico, num_paciente, descrição_med: Números inteiros

descricao, nome, designacao, morada, especialidade: Texto

quantidade: Número inteiro ; **Preco_unitario:** real ; **data_nasc, data:** Data/Hora

1. Escreve o(s) comando(s) SQL que te permita:

    1. Quais os pacientes do médico com o nome 'Alberto Martelo'?

    2. Executa o comando EXPLAIN e tenta interpretar o resultado.

    3. Cria um índice para o nome do paciente, do tipo FULLTEXT, na tabela paciente.

    4. Executa a querie/consulta do exercício 1, com o comando EXPLAIN e tenta interpretar o resultado.

    5. Utiliza as cláusulas MATCH e AGAINST para:

        a. Procurar o nome referente ao exercício 1.
        b. Procurar o nome que contenha 'Rosa' e que não tenha 'Maria'
        c. Os nomes iniciados por "Mar"
        d. Os nomes que contenham 'Rosa'

    6. Quais as consultas que têm pacientes de nome António?
        a. Executa usando o EXPLAIN
        b. Usa a sintaxe *show index from* aplicada à tabela paciente.

        Em ambas as alíneas, analisa o resultado.

    Utiliza a base de dados **vendascd** para realizares os exercícios seguintes:

    7. Usa o operador LIKE para obter todas as vendas do CD cujo título começa pela palavra 'Meu'
        a. Executa usando o EXPLAIN, observa e retém os dados essenciais do plano de execução.

    8. Cria um índice fulltext para o título do CD.

9. Usa a pesquisa booleana para obter todas as vendas do CD cujo título começa pela palavra 'Meu'

    a. Executa usando o EXPLAIN, observa e retém os dados essenciais do plano de execução.

10. Interpreta os planos de execução de 7.a e 9.a em termos de otimização da querie.

11. Executa a seguinte consulta: Quais o total de vendas de CD cada Editora (nomeEditora)?

    a. Aplica o EXPLAIN e retém os dados essenciais do plano de execução.

    b. Cria um índice para o campo qtd, da tabela vendas.

    c. Executa de novo a consulta, mas obrigando o MySql a usar o índice criado na alínea anterior (aplicar na consulta … USE INDEX (qtd)).

    d. Aplica de novo o EXPLAIN e compara os dados essenciais do plano de execução com os da alínea a.

Bibliografia

Tavares, F. (2015). MySql. Lisboa: FCA
https://pt.slideshare.net/helderfredlopes/melhorando-o-desempenho-de-suas-consultas-no-mysql
https://dev.mysql.com/doc/refman/5.6/en/explain-output.html#explain-join-types
https://georgemoura.com.br/explicando-mysql-explain/
https://docs.microsoft.com/pt-br/azure/mysql/howto-troubleshoot-query-performance
http://bit.ly/2rrwqdg