O que são índices?

São estruturas constituídas por um campo ou conjuntos de campos que permitem localizar e classificar registos com maior rapidez. É a ferramenta principal utilizada para a otimização de consultas, evitando a perda de desempenho de um SGBD. Os índices criados são (por defeito) do tipo Btree, ou seja, são armazenados segundo uma estrutura em árvore, o que permite a sua otimização. Quando um índice é criado, é criada uma estrutura de dados com as informações que fazem parte do índice. É esta estrutura que é percorrida para localizar os dados.

A indexação usa-se em campos que se pesquisam com frequência, que se ordenem e que se associem a campos noutras tabelas ou em consultas de várias tabelas. Opcionalmente permite também garantir a exclusividade dos valores introduzidos.

Os índices podem agilizar pesquisas e consultas, mas podem prejudicar o desempenho quando se adicionam ou atualizam dados. Quando os dados são inseridos numa tabela que contém um ou mais campos indexados, o SGBD deve atualizar os índices de cada vez que um registo é adicionado ou alterado, o que pode ter o efeito reverso do pretendido!

Só tem interesse criar índices para os campos/colunas cuja consulta está prevista, caso contrário, serão usados recursos e espaço em disco que não terão qualquer vantagem.

É possível criar um máximo de 16 índices por tabela e mais do que um por campo. Os índices são mais importantes em *storages engines* transacionais (tipo InnoDB) do que não transacionais (tipo MyISAM).

Tipos de índices*

KEY ou INDEX — Refere-se a um índice que não é único, permitindo valores não distintos, isto é, podem existir linhas com valores idênticos em todas as colunas do índice, pelo que o seu objetivo é apenas acelerar as consultas efetuadas. Podem ser indexadas neste tipo as colunas onde sejam aplicadas consultas usando as condições do tipo WHERE ou ORDER BY;
UNIQUE — índice em que cada linha adquire um valor único, distinto, não podendo ter valores duplicados. Não se trata de uma otimização ao nível das consultas efetuadas, mas antes para evitar duplicação de valores (pode admitir o valor NULL).

Notas:

- Estes índices, aplicados a mais do que uma coluna, garantem que cada combinação de valores na chave de índice é única. Por exemplo, se o índice for criado com a combinação dos campos Apelido e Nome, não poderão existir, na tabela associada, duas linhas (dois registos) com a mesma combinação de valores, digamos "Ana Duarte".
- Este índice não poderá ser criado se se refere a um único campo e tendo este o valor NULL em mais do que um registo. Da mesma forma, não se poderá criar o índice, se este se referir a mais do que um campo, se a combinação de campos tiver o valor NULL, em

mais do que um registo. Estas situações serão interpretadas como valores duplicados para fins de indexação.

PRIMARY — Atua da mesma forma que o índice único, mas apenas pode existir um por tabela (tipo clustered). O objetivo é identificar cada linha da tabela, pelo que não deve ser usado em situações onde possa ser inserido um valor do tipo NULL e é utilizado em situações de autoincremento. No MySQL para a chave primária de uma tabela é criado automaticamente um índice PRIMARY;

FULLTEXT — É diferente dos índices anteriores permitindo a pesquisa de texto. Constitui uma alternativa ao índice comum ou nas situações em que o operador LIKE apresenta limitações. Neste caso a pesquisa é feita usando a condição MATCH/AGAINST.

Sintaxe geral de consulta usando o índice FULLTEXT:

```
SELECT * FROM nome_tabela
WHERE  MATCH (nome_campo)
AGAINST ('palavra1 palavra2 palavra3…')
```

Exemplo(*): Selecionar o nome das lojas com o nome Adega ou Viperina.

SELECT * FROM Lojas WHERE MATCH (Nome) AGAINST ('Adega Viperina')

Criar Índices em Tabelas

Sintaxe(**):

```
CREATE [UNIQUE|FULLTEXT] INDEX <nome_índice>
ON <nome_tabela> (campo1 [(tamanho)][ASC|DESC],campo2[(tam)][ [ ASC|DESC ],...)
```

Onde **tam** corresponde ao tamanho do valor do campo indexado

NOTAS:

A palavra reservada UNIQUE não permite que o índice tenha valores duplicados;
ASC|DESC ordem de valores dos campos ASC – ascendente (valor pré-determinado), DESC-descendente.
Não se deve utilizar a palavra reservada PRIMARY na criação de um novo índice numa tabela que já tenha uma chave primária.

Exemplo: Criar um índice único sobre o campo **CodC** da Tabela **Comissoes**, cujos valores estejam ordenados por ordem descendente.

CREATE UNIQUE INDEX InCodC **ON** Comissoes (CodC **desc**)

Apagar um Índice

Síntaxe:

```
DROP INDEX <Nome_Index> ON <Nome_Tabela>
```

Exemplo: Apagar o índice anteriormente criado.

DROP INDEX InCodC **ON** Comissoes

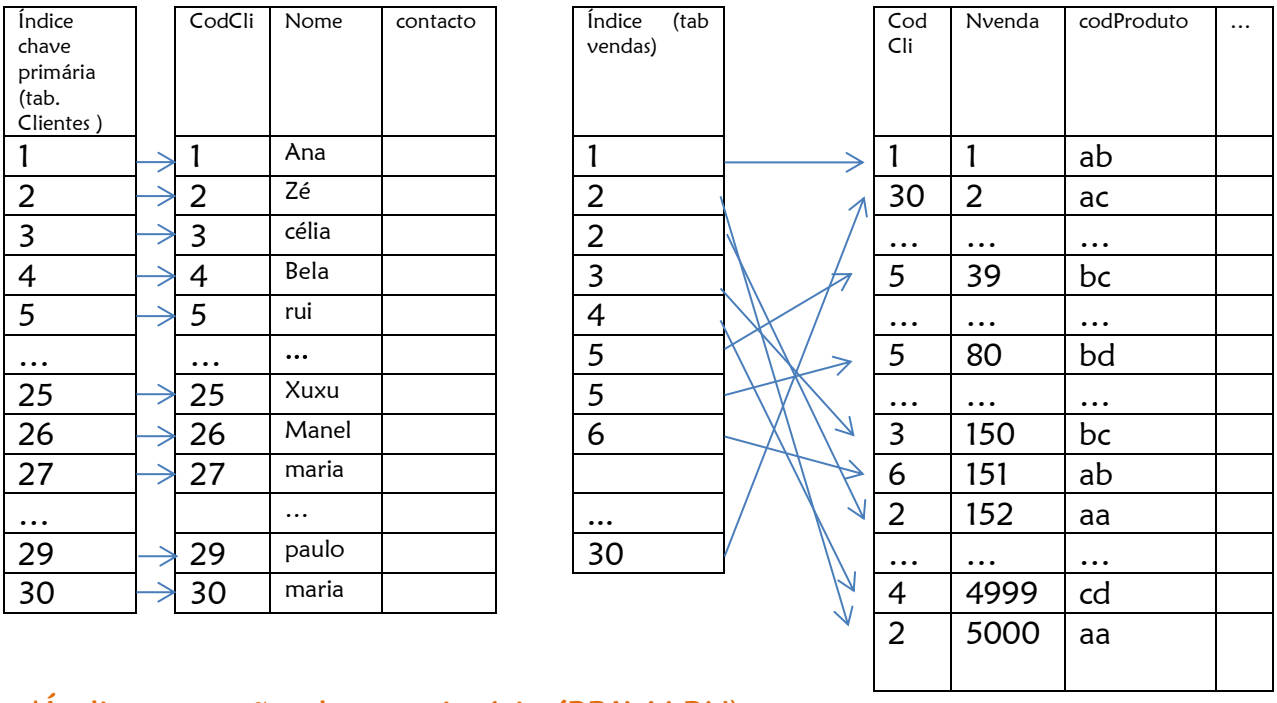
Nota: Esta cláusula só permite eliminar um índice de cada vez!
Para eliminar um campo que seja um índice é necessário proceder em primeiro lugar à eliminação do índice e só depois executar o SQL que elimina o campo!

Quando se faz uma consulta a uma tabela com vista a encontrar um registo (linha), ou conjunto de registos específicos, é necessário percorrer todos os registos até se encontrar o que se pretende. Se a tabela tiver milhares de registos e se considerarmos obter resultados dependentes de várias tabelas o número de registos a percorrer aumenta enormemente.
Supõe que temos duas tabelas, CLIENTE e VENDAS e que pretendemos obter todas as vendas por nome de cliente:

```
SELECT nome, vendas.*
FROM cliente c, vendas v
WHERE a.codCliente = v.codCliente
```

Imaginemos que a tabela cliente contém 30 registos e que a tabela Vendas contenha 5.000 registos. Teríamos o seguinte produto cartesiano para abordar apenas os dados necessários: 30 x 5.000 = 150.000. Este valor são todos os registos distintos que podem ser encontrados na consulta, tendo a base de dados, no total, apenas de 5.040 registos!

O ganho de eficiência obtido com o uso de índices permite determinar precisamente onde as linhas "casadas" acabam e ignorar o resto das linhas. Outro ganho de eficiência é obtido pelo uso de algoritmos de posicionamento para encontrar a primeira entrada "casada", sem ter que fazer uma varredura linear desde o início do índice.



Índice chave primária (tab. Clientes)
1
2
3
4
5
…
25
26
27
…
29
30

CodCli	Nome	contacto
1	Ana	
2	Zé	
3	célia	
4	Bela	
5	rui	
…	…	
25	Xuxu	
26	Manel	
27	maria	
	…	
29	paulo	
30	maria	

Índice (tab vendas)
1
2
2
3
4
5
5
6
…
30

Cod Cli	Nvenda	codProduto	…
1	1	ab	
30	2	ac	
…	…	…	
5	39	bc	
…	…	…	
5	80	bd	
…	…	…	
3	150	bc	
6	151	ab	
2	152	aa	
…	…	…	
4	4999	cd	
2	5000	aa	

*Índices que são chave primária (PRIMARY)

O índice é montado na própria tabela, criando a estrutura ORDENADA de árvore para facilitar as buscas.

Exemplo clássico: Dicionário (Localizamos a palavra e junto com ela está a sua definição).

*Índices que não são chave primária

O índice é uma estrutura ORDENADA à parte, que contém apenas a coluna indexada e uma tabela pode ter N índices deste tipo. Se for necessário consultar alguma informação que não está neste índice, a informação é localizada utilizando o índice de chave primária da tabela (Key Lookup).

Exemplo clássico: Índice de um livro (A partir do índice de um livro localizamos a página onde está o capítulo e depois abrimos a página para ver as informações pretendidas).

Acelerar a pesquisa usando determinado índice

```
SELECT … USE INDEX (Nome_do_indice)
FROM …
```

Em que campos/colunas usar índices no MySQL?

Devemos indexar campos usados para procurar, ordenar ou agrupar, e não campos que serão apenas exibidos no resultado da execução de uma consulta (os campos que seguem o SELECT).

Os melhores campos candidatos para índices são:

- Os campos que estão em condições da cláusula WHERE (qualquer índice que não cubra todos os níveis de AND da cláusula Where não será utilizado para otimizar a consulta);
- Campos que façam parte das junções (linhas que "casam" com linhas de outras tabelas)
- Campos usados ma função de agregação MIN ou MAX;
- Campos onde se executem operações de ordenação e agrupamento (ORDER BY e GROUP BY);
- Campos com cardinalidade relativamente alta em relação ao número de linhas da tabela (isto é, que têm muitos valores distintos e poucos valores duplicados)[1]
- Campos, preferencialmente, do tipo inteiro, exclusivos e não nulos;
- Campos que não sofram mudanças frequentes;
- Chaves pequenas (sem muitos campos);

[1]A cardinalidade de uma coluna é o número de valores distintos que contém. Por exemplo, uma coluna que contém os valores 1, 3, 7, 4, 7 e 3, tem cardinalidade quatro. Se uma coluna contiver muitos valores de idade diferentes, um índice diferenciará linhas prontamente. Por outro lado, um índice não será frutífero para registrar o género e só contém os valores 'M' e 'F'. Se os valores ocorrem aproximadamente com a mesma frequência, obteremos aproximadamente metade das linhas, qualquer que seja o valor procurado. Dadas estas circunstâncias, o índice poderia nunca ser usado, porque o otimizador de consulta geralmente ignora um índice em favor de uma varredura completa da tabela quando determinar que certo valor aconteça numa grande percentagem das linhas de uma tabela. O valor convencional para esta percentagem era de 30%.

Devemos, na criação de um índice, ter em consideração se numa tabela são efetuadas muitas operações de inserção, eliminação e alteração de dados, INSERT, DELETE e UPDATE, respetivamente, os índices podem ficar ineficientes e gerar uma tarefa extra para o SGBD, que deverá tentar atualizar os índices a cada operação.

EXERCÍCIOS

Inicializa o serviço de **Wamp**, e através do PHPMyAdmin elabora as seguintes operações, no **phpmyadmin**, copiando o resultado, após execução, para um ficheiro do notepad++.

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

1. Criar um índice na tabela Postal, para o campo Localidade, ordenado por ordem descendente.

2. Adiciona na tabela Postal o campo Endereco, com 45 carateres e não nulo.

3. Escrever o comando SQL que te permita criar um índice na tabela Empregados2 utilizando o campo Id, obrigando a que o índice não contenha valores repetidos.

4. Criar um índice, que não permita valores duplicados, na tabela Empregados para o campo telemóvel e que limite na pesquisa os 3 primeiros carateres.

5. Criar um índice, do tipo FULLTEXT, na tabela Empregados para o campo Nome.

6. Eliminar, da tabela Postal, o campo Enderecos (toma atenção que o campo Endereco é um índice!)

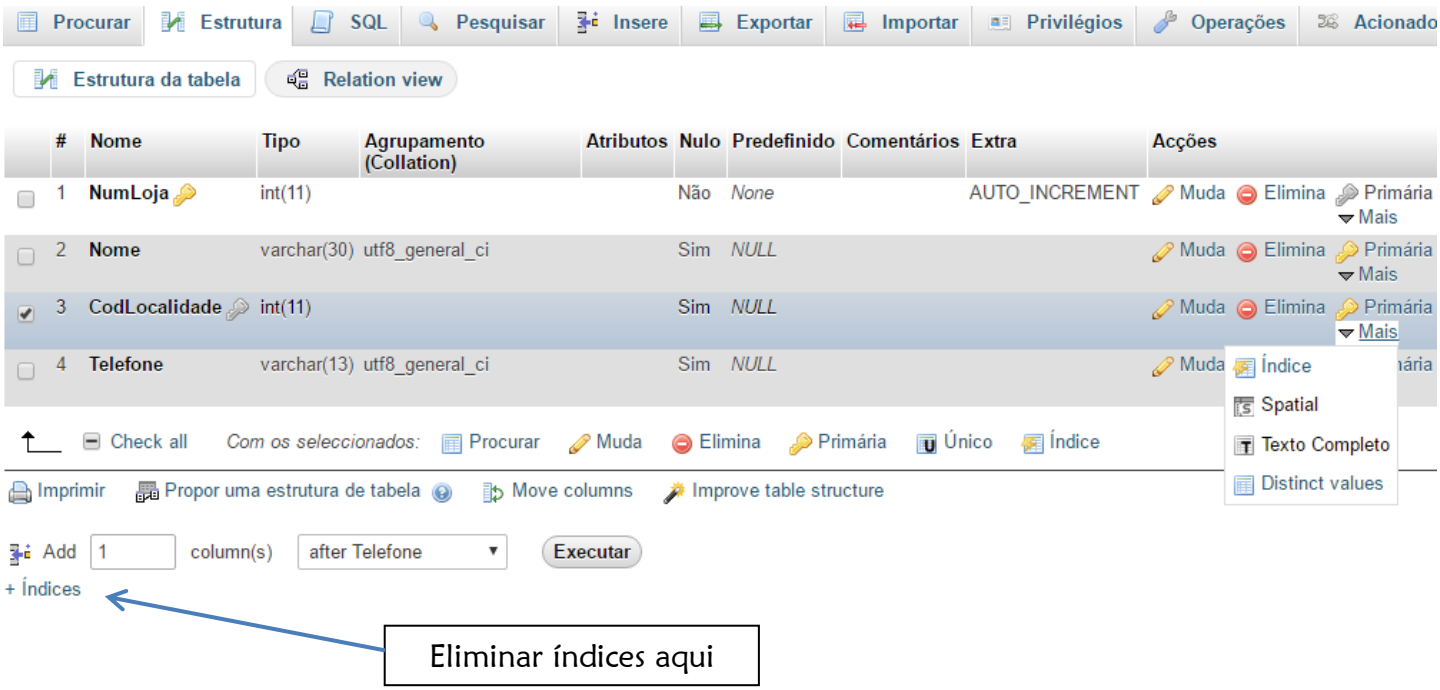
7. Exporta a base de dados com o nome **Empresa1-indices.sql** e envia-a também para a classroom, ***Disciplina de PSD, no tópico UFCD 0814 - Programação em linguagem SQL avançada,*** do Google Apps.

PROCEDIMENTOS para Adição de índice para a chave externa CodLocalidade

- Selecionar a tabela Lojas e clicar no separador estrutura;

- Selecionar o campo CodLocalidade;

- Em **Mais** selecionar índice e executar.



A- Volta a tentar estabelecer as relações entre as tabelas da base de dados.

(*) ALTER TABLE <nome_tabela>

ADD [UNIQUE|FULLTEXT] INDEX|KEY <nome_índice> (campo1 [(tamanho)][ASC|DESC]…)

Exemplo:

ALTER TABLE empregados2
ADD UNIQUE INDEX ID_NOME (NOME (3) DESC)

Ou

ALTER TABLE empregados2
ADD INDEX (NOME (3) DESC)

Neste caso o índice fica com o nome igual ao do campo. KEY é sinónimo de INDEX.

(**)
```
SELECT titulo, descricao
 FROM websites
 WHERE MATCH (titulo, descricao)
 AGAINST ('"Microsoft Brasil"' IN BOOLEAN MODE)
```

Qual o resultado deste select?

Pesquisa as vantagens de utilização do AGAINST em BOOLEAN MODE no link (a)

Bibliografia

Damas, L. (2005). SQL. Lisboa: FCA
Tavares, F. (2015). MySql. Lisboa: FCA
Marcelo. Otimizando consultas SQL em mysql.2007. [S.I.] Disponível em: https://www.devmedia.com.br/otimizando-consultas-sql-em-mysql/5257
Paul. Optimização de consultas no MySql. 2007.[S.I.]. Disponível em: https://www.devmedia.com.br/otimizacao-de-consultas-no-mysql/6178
https://pt.slideshare.net/helderfredlopes/melhorando-o-desempenho-de-suas-consultas-no-mysql

(a) https://www.devmedia.com.br/indices-fulltext-no-mysql/7631

RESPONDE ÀS SEGUINTES QUESTÕES:

RESPONDE ÀS SEGUINTES QUESTÕES:

(Escreve as questões e respostas num documento, com o nome índicesFULLTEXT.docx, para a classroom, a adicionar à entrega da ficha de trabalho n.º2)

Como criar um índice fulltext?

Como efetuar uma pesquisa através de um índice *fulltext* ?

Quais os parâmetros de um índice fulltext?

O que é o valor de relevância?

Explicita o que é o grau de relevância 0 e o que o faz aumentar.

Como exibir, num resultado o valor de relevância?

O mysql desconsidera palavras pequenas na pesquisa fulltext?

Qual a base das pesquisas *fulltext* em modo booleano?

Quais os operadores usados em modo boleano e para que servem?

Qual o significado de:
"Usar "~" para diminuir o valor de relevância da frase usada no AGAINST"