Instituto Federal Goiano - Campus Ceres
Bacharelado em Sistemas de Informação
Prof. Me. Ronneesley Moura Teles

Daniel Moreira Cardoso
Gusttavo Nunes Gomes
Ianka Talita Bastos de Assis
Ígor Justino Rodrigues
Thalia Santos de Santana

Diferenças de desempenho do uso de BLOBs no banco de dados

Novembro
2017

Sumário

Diferenças de desempenho do uso de BLOBs no banco de dados

1 Introdução

O BLOB *(Binary Large Object)*, ou em português, Grande Objeto Binário, refere-se a campos criados para armazenar qualquer tipo de informação em formato binário, dentro de um banco de dados [1]. Geralmente, são arquivos multimídia, como imagens, áudios, vídeos, etc. Normalmente, grande parte dos bancos de dados (BDs) dão suporte aos tipos básicos, como datas, *strings*, números e assim, para aqueles que não fazem parte dessa linha, utiliza-se BLOBs [2].

O MySQL também faz uso de BLOBs e são divididos em quatro tipos [1]:

- TINYBLOB: campo BLOB de armazenamento máximo igual a 255 caracteres (8 bits) mais 1 de controle;
- BLOB: o mesmo que o Tinyblob, porém armazenando até 16535 caracteres (16 bits) mais 2 de controle;
- MEDIUMBLOB: o mesmo que o Tinyblob, porém armazenando até 16777216 caracteres (24 bits) mais 3 de controle;
- LONGBLOB: o mesmo que o Tinyblob, porém armazenando até 4294967295 caracteres (32 bits) mais 4 de controle.

De acordo com o site *ITPro*, trabalhar com armazenamento de dados usando BLOB faz com que todas as informações se mantenham em uma entidade de maneira conjunta, o que acarreta em facilidades de busca e de recuperação de dados. Mas infelizmente, isso acaba gerando um banco de dados com tamanho elevado [3].

Quando trata-se de armazenar separadamente, com uso de URLs (Localizador Uniforme de Recursos), os bancos são menores comparativamente, além de trazer rapidez, pois o processo de leitura de dados sobrecarrega muito menos do que ler no próprio BD. Mas é necessário sempre realizar mantenimento manual do *link*, ao qual, liga o BD e o arquivo externo [3].

O autor ainda afirma que é possível guardar arquivos binários fora do BD e incluí-los como URL para o objeto, unindo as técnicas apresentadas.

Em alguns fóruns, como *yiiframework*, alguns participantes sugerem BLOBs devido ao quesito segurança. Porém, no quesito desempenho, os mesmos concordam que URL é a melhor forma de trabalho. Outro correspondente afirma, inclusive, que essa escolha deve depender de como aplicação é estruturada, mas ele prefere fazer uso de caminhos do arquivo [4].

Já no *Stack Exchange*, há concordância que depende do caso específico. Para um dos usuários, BLOBs traz gerenciabilidade, ajudando em situações que envolvam restauração do BD e *backups* [5].

Segundo ainda o Blog *SQL Authority*, a autor faz a sugestão em prol do armazenamento da localização das imagens (por exemplo), no banco de dados usando o tipo de dados VARCHAR em vez de qualquer BLOB ou outro tipo de dados binário. Sua explicação está ligada ao fato de que armazenar a localização ao invés do BLOB reduz muito o tamanho do BD, bem como atualizar ou substituir a imagem são métodos mais simples, visto que refere-se apenas a uma operação de arquivo em vez de uma atualização efetiva de *inserts* e *delets* no BD [6].

Ademais, há um artigo intitulado *"To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem?"* [7], que descreve que ao realizar a decisão por se armazenar em um sistema de arquivos ou em uma base de dados, é feita através de atributos como simplicidade de aplicação ou capacidade de gerenciamento, sabendo que o desempenho do sistema acaba afetando esses fatores. Os autores recordam que é de conhecimento que BDs lidam eficientemente com grande número de objetos pequenos, enquanto sistemas de arquivos são mais eficiente para objetos grandes. O estudo traz a informação que BLOBs menores do que 256KB são mais eficientemente gerenciados por um BD, enquanto um sistema de arquivos é mais eficiente para aqueles maiores do que 1 MB, com variações decorrentes de diferentes BDs e sistemas de arquivos.

Além disso, os mesmos afirmam que o desempenho muda ao longo do tempo conforme o armazenamento torna-se fragmentado, sendo este tema, parte importante das contextações do estudo, concentrando-se em problemas de fragmentação de armazenamento. Verifica-se que, em síntese, os sistemas de arquivos parecem ter uma melhor fragmentação e manipulação do que BDs e isso impulsiona o ponto de equilíbrio de cerca de 1 MB para aproximadamente 256KB.

2 Aplicação

Em busca de avaliar o desempenho do uso de BLOBs, criou-se a mesma tabela (Figura 1), uma contendo o BLOB e outra com a URL do arquivo.

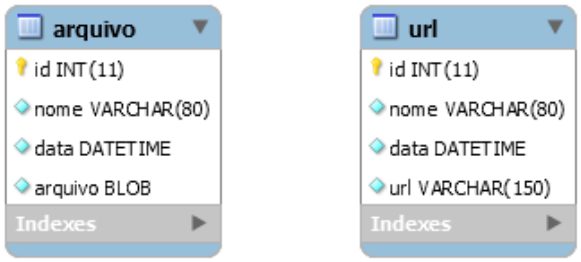

Figura 1. Tabela com uso de BLOB e tabela com URL.

Em cada uma das tabelas foram inseridos 49000 registros, igualmente. A partir destes, avaliou-se a diferença de tempo de 1000 consultas. Abaixo, tem-se o arquivo *.sql* referente a criação das tabelas no SGBD (Sistema Gerenciador de Banco de Dados) *MySQL Workbench*.

```
1  CREATE SCHEMA `redesocial_teste`;
2  USE `redesocial_teste`;
3
4  CREATE  TABLE `redesocial_teste`.`url` (
5    `id` INT NOT NULL AUTO_INCREMENT ,
6    `nome` VARCHAR(80) NOT NULL ,
7    `data` DATETIME NOT NULL ,
8    `url` VARCHAR(150) NOT NULL ,
9   PRIMARY KEY (`id`) );
10
11 CREATE  TABLE `redesocial_teste`.`arquivo` (
12   `id` INT NOT NULL AUTO_INCREMENT ,
13   `nome` VARCHAR(80) NOT NULL ,
14   `data` DATETIME NOT NULL ,
```

```
15    `arquivo` BLOB NOT NULL ,
16   PRIMARY KEY (`id`) );
```

recursos/codigos/banco.sql

3 Resultados

Avaliando-se o desempenho, algumas consultas foram realizadas com 1000 repetições, selecionando por data, nome e por todos os campos, a qual essa última, informa se há ou não real diferença em relação ao uso de BLOBs. Todos os tempos de cada consulta foram salvos, gerando uma média final referente a cada *select*. No caso do BLOB, o *sql* retrata o uso de *limit 100*, visto que o BD não conseguiu com lidar com os dados, e o máximo alcançado justamente trata-se de 100 registros por vez.

```
1000 select count(*) from arquivo where data < '2017-01-01 00:00:00';

1 - Tempo Gasto: 0.01757 seg
2 - Tempo Gasto: 0.01746 seg
3 - Tempo Gasto: 0.0171 seg
/*[...]1000 Testes*/
998 - Tempo Gasto: 0.01709 seg
999 - Tempo Gasto: 0.01711 seg
1000 - Tempo Gasto: 0.01712 seg


/*Resultados:*/
Tempo Total: 17.65624 seg
Media de Tempo: 0.01765624 seg
```

recursos/codigos/Select_Data_Blob.txt

```
1000 select count(*) from url where data < '2017-01-01 00:00:00';

1 - Tempo Gasto: 0.01939 seg
2 - Tempo Gasto: 0.01818 seg
3 - Tempo Gasto: 0.01832 seg
/*[...]1000 Testes*/
998 - Tempo Gasto: 0.01715 seg
999 - Tempo Gasto: 0.01737 seg
1000 - Tempo Gasto: 0.01719 seg


/*Resultados:*/
Tempo Total: 17.53472 seg
Media de Tempo: 0.01753472 seg
```

recursos/codigos/Select_Data_URL.txt

```
1000 select count(*) from redesocial_teste.arquivo where nome like '
    Igor';

1 - Tempo Gasto: 0.53173 seg
2 - Tempo Gasto: 0.03177 seg
3 - Tempo Gasto: 0.03301 seg
/*[...]1000 Testes*/
```

```
998 - Tempo Gasto: 0.03144 seg
999 - Tempo Gasto: 0.03114 seg
1000 - Tempo Gasto: 0.03088 seg


/*Resultados:*/
Tempo Total: 35.49721 seg
Media de Tempo: 0.03549721 seg
```

recursos/codigos/Select_Nome_Blob.txt

```
1000 select count(*) from redesocial_teste.url where nome like 'Igor';

1 - Tempo Gasto: 0.10893 seg
2 - Tempo Gasto: 0.03169 seg
3 - Tempo Gasto: 0.03149 seg
/*[...]1000 Testes*/
997 - Tempo Gasto: 0.03219 seg
998 - Tempo Gasto: 0.03107 seg
999 - Tempo Gasto: 0.0315 seg
1000 - Tempo Gasto: 0.03121 seg

/*Resultados:*/
Tempo Total: 32.40123 seg
Media de Tempo: 0.03240123 seg
```

recursos/codigos/Select_Nome_URL.txt

```
1000 select * from redesocial_teste.arquivo limit 100;

1 - Tempo Gasto: 1.40533 seg
2 - Tempo Gasto: 1.25419 seg
3 - Tempo Gasto: 1.26168 seg
/*[...]1000 Testes*/
998 - Tempo Gasto: 1.44957 seg
999 - Tempo Gasto: 1.43912 seg
1000 - Tempo Gasto: 1.42797 seg


/*Resultados:*/
Tempo Total: 1353.09537 seg
Media de Tempo: 1.35309537 seg
```

recursos/codigos/Select_Todos_Blob.txt

```
1000 select * from redesocial_teste.url;

1 - Tempo Gasto: 0.2262 seg
2 - Tempo Gasto: 0.18478 seg
3 - Tempo Gasto: 0.1828 seg
/*[...]1000 Testes*/
997 - Tempo Gasto: 0.20113 seg
998 - Tempo Gasto: 0.18631 seg
999 - Tempo Gasto: 0.20277 seg
1000 - Tempo Gasto: 0.17124 seg

/*Resultados:*/
Tempo Total: 178.3877 seg
Media de Tempo: 0.1783877 seg
```

recursos/codigos/Select_Todos_URL.txt

4 Considerações finais

Levando em conta o objetivo deste, ao realizar a comparação de desempenho nas consultas, fica claro que ao fazer uso de URLs, há maior rapidez nas consultas do que usando BLOBs (0.1783877 seg $<$ 1.35309537 seg).

5 Referências Bibliográficas

[**1**] https://pt.stackoverflow.com/questions/100802/como-funciona-o-campo-blob

[**2**] https://pt.wikipedia.org/wiki/BLOB

[**3**] http://www.itprotoday.com/cloud-data-center/storing-blobs-database-or-file-system

[**4**] http://www.yiiframework.com/forum/index.php/topic/33244-best-practice-to-store-images-in-database-blob-or-url-based/

[**5**] https://dba.stackexchange.com/questions/736/is-it-better-to-store-images-in-a-blob-or-just-the-url

[**6**] https://blog.sqlauthority.com/2007/12/13/sql-server-do-not-store-images-in-database-store-location-of-images-url/

[**7**] SEARS, Russell; VAN INGEN, Catharine; GRAY, Jim. **To blob or not to blob: Large object storage in a database or a filesystem?.** arXiv preprint cs/0701168, 2007.