

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

1	Introdução	2
2	Aplicação	3
3	Resultados	3
4	Conclusão	5
5	Referências Bibliográficas	5

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 manutenção 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].

2 Aplicação

Em busca de avaliar o desempenho do uso de BLOBs, criou-se a mesma tabela em dois bancos separados (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.

```
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

```
1 1000 select count(*) from arquivo where data < '2017-01-01 00:00:00';
2
3 1 - Tempo Gasto: 0.01757 seg
4 2 - Tempo Gasto: 0.01746 seg
5 3 - Tempo Gasto: 0.0171 seg
6 /*[...]1000 Testes*/
7 998 - Tempo Gasto: 0.01709 seg
8 999 - Tempo Gasto: 0.01711 seg
9 1000 - Tempo Gasto: 0.01712 seg
10
11
12 /*Resultados:*/
13 Tempo Total: 17.65624 seg
14 Media de Tempo: 0.01765624 seg
```

recursos/codigos/Select_Data_Blob.txt

```

1 1000 select count(*) from url where data < '2017-01-01 00:00:00';
2
3 1 - Tempo Gasto: 0.01939 seg
4 2 - Tempo Gasto: 0.01818 seg
5 3 - Tempo Gasto: 0.01832 seg
6 /*[...]1000 Testes*/
7 998 - Tempo Gasto: 0.01715 seg
8 999 - Tempo Gasto: 0.01737 seg
9 1000 - Tempo Gasto: 0.01719 seg
10
11
12 /*Resultados:*/
13 Tempo Total: 17.53472 seg
14 Media de Tempo: 0.01753472 seg

```

recursos/codigos/Select_Data_URL.txt

```

1 1000 select count(*) from redesocial_teste.arquivo where nome like '
    Igor';
2
3 1 - Tempo Gasto: 0.53173 seg
4 2 - Tempo Gasto: 0.03177 seg
5 3 - Tempo Gasto: 0.03301 seg
6 /*[...]1000 Testes*/
7 998 - Tempo Gasto: 0.03144 seg
8 999 - Tempo Gasto: 0.03114 seg
9 1000 - Tempo Gasto: 0.03088 seg
10
11
12 /*Resultados:*/
13 Tempo Total: 35.49721 seg
14 Media de Tempo: 0.03549721 seg

```

recursos/codigos/Select_Nome_Blob.txt

```

1 1000 select count(*) from redesocial_teste.url where nome like 'Igor';
2
3 1 - Tempo Gasto: 0.10893 seg
4 2 - Tempo Gasto: 0.03169 seg
5 3 - Tempo Gasto: 0.03149 seg
6 /*[...]1000 Testes*/
7 997 - Tempo Gasto: 0.03219 seg
8 998 - Tempo Gasto: 0.03107 seg
9 999 - Tempo Gasto: 0.0315 seg
10 1000 - Tempo Gasto: 0.03121 seg
11
12 /*Resultados:*/
13 Tempo Total: 32.40123 seg
14 Media de Tempo: 0.03240123 seg

```

recursos/codigos/Select_Nome_URL.txt

```

1 1000 select * from redesocial_teste.arquivo limit 100;
2
3 1 - Tempo Gasto: 1.40533 seg
4 2 - Tempo Gasto: 1.25419 seg
5 3 - Tempo Gasto: 1.26168 seg
6 /*[...]1000 Testes*/
7 998 - Tempo Gasto: 1.44957 seg
8 999 - Tempo Gasto: 1.43912 seg

```

```

9 1000 - Tempo Gasto: 1.42797 seg
10
11
12 /*Resultados:*/
13 Tempo Total: 1353.09537 seg
14 Media de Tempo: 1.35309537 seg

```

recursos/codigos/Select_Todos_Blob.txt

```

1 1000 select * from redesocial_teste.url;
2
3 1 - Tempo Gasto: 0.2262 seg
4 2 - Tempo Gasto: 0.18478 seg
5 3 - Tempo Gasto: 0.1828 seg
6 /*[...]1000 Testes*/
7 997 - Tempo Gasto: 0.20113 seg
8 998 - Tempo Gasto: 0.18631 seg
9 999 - Tempo Gasto: 0.20277 seg
10 1000 - Tempo Gasto: 0.17124 seg
11
12 /*Resultados:*/
13 Tempo Total: 178.3877 seg
14 Media de Tempo: 0.1783877 seg

```

recursos/codigos/Select_Todos_URL.txt

4 Conclusão

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-a-blob>

[5] <https://dba.stackexchange.com/questions/736/is-it-better-to-store-images-in-a-blob>

[6] <https://blog.sqlauthority.com/2007/12/13/sql-server-do-not-store-images-in-database>