

# Desvendando o SELECT

 Entendendo o gStat - parte 2



Entrevista João Prado Maia

Review do livro Firebird  
Essencial



Checklist de Performance

## Editorial

O lançamento da primeira edição da DB FreeMagazine foi um sucesso! Em um mês tivemos cerca de **5.000 downloads!**

Gostaria de lembrar que por se tratar de uma “revista eletrônica”, a DB FreeMagazine possui algumas funcionalidades que não são encontradas em mídias impressas. Entre elas estão os hiperlinks que, quando clicados, executam ações como posicionar o cursor em determinado capítulo ou parágrafo no texto, abrir uma página web, enviar um email, etc.

Um recurso muito importante para nos ajudar a direcionar a revista de acordo com a preferência do leitor são os links para **avaliação**, que podem ser encontrados no final de cada artigo. Clicando na opção escolhida (*Ótimo*, *Bom* ou *Ruim*) seu voto estará sendo captado pelo servidor web da revista, e chegará até nós. A única informação armazenada é o seu número IP, para dificultar fraudes na votação. **Não deixem de avaliar os artigos de cada edição!**

O índice localizado na parte superior da revista permite o posicionamento instantâneo em qualquer artigo, bastando clicar com o mouse sobre o título do artigo desejado.

**Atenção usuários de Linux!** Sugerimos que seja utilizado o leitor de PDF da própria Adobe (Adobe Acrobat Reader para Linux). Alguns leitores não oficiais tem problemas para renderizar a revista corretamente.

O site da revista DB FreeMagazine está sendo constantemente atualizado com notícias e novidades do mundo dos bancos de dados. Os usuários de **RSS** também podem contar com um *feeder* para receber as notícias do site. Aproveite também e vote nas pesquisas que estão online no site da DB FreeMagazine!

### Informações

DB FreeMagazine nº 002 - Ano I  
Maio/2005  
Contato geral:  
webmaster@dbfreemagazine.com.br

### Equipe editorial

Carlos H. Cantu  
(cantu@dbfreemagazine.com.br)

Luiz Paulo de Oliveira Santos  
(lpaulo@dbfreemagazine.com.br)

### Contribuíram nessa edição

Luiz Paulo de Oliveira Santos  
Carlos H. Cantu  
Artur Anjos  
Josh Berkus

*É proibida a reprodução de qualquer parte do conteúdo dessa publicação sem autorização prévia por escrito.*

Boa leitura, e bom divertimento!

Carlos H. Cantu  
Editor DB FreeMagazine

## ANUNCIE NA DB FreeMagazine

Valorize seu produto ou serviço!

anuncios@dbfreemagazine.com.br

### Dica para melhor visualização:

Utilize a resolução 1024x768 pixels e configure o *Acrobat Reader* para Zoom de 100%. Feche todas as abas laterais e esconda as barras de ferramentas, liberando o máximo de área útil na tela, ou simplesmente rode a revista em modo *fullscreen*.

## Saindo do Forno...

### Firebird 2.0 Alpha2

Mais um release alpha do Firebird 2.0 foi disponibilizado para testes e pode ser baixado gratuitamente em [http://www.firebirdsql.org/index.php?op=files&id=fb2\\_alpha02](http://www.firebirdsql.org/index.php?op=files&id=fb2_alpha02). Essa versão corrige vários bugs encontrados no Alpha1.

Fonte: <http://www.firebirdsql.org>

### JBUILDER Open Source?

Um "ruído" muito grande foi ouvido na net recentemente, quando o site "The Register" publicou uma notícia sobre uma possível abertura do código do JBuilder pela Borland. Depois de alguns dias, a notícia foi desmentida pela própria Borland. O boato surgiu devido ao fato da Borland estar investindo fortemente no desenvolvimento da IDE Eclipse (Open Source).

Fonte: <http://bdn.borland.com/article/0,1410,33055,00.html>

### Certificação MySQL

O site Database Journal lançou uma série de artigos sobre o processo de certificação do MySQL, onde é mostrado os exames

e assuntos abordados, bem como links interessantes relacionados a obtenção de conhecimento em cada um dos tópicos.

Fonte: <http://www.databasejournal.com/features/mysql/article.php/3492896>

### PostgreSQL lança pacotes com correções de falhas de segurança

Os novos pacotes se referem às Security releases 8.0.3, 7.4.8, 7.3.10, 7.2.8. As falhas corrigidas já haviam sido reportadas por Tom Lane, bem como os procedimentos para corrigi-las manualmente.

Fonte: <http://www.postgresql.org/about/news.322>

### Oracle e Mozilla juntos

A Oracle e a Mozilla estão trabalhando juntos em um software livre que promete bater de frente com o Outlook da Microsoft. O novo produto iria integrar os projetos Sunbird e Thunderbird, ambos da Mozilla.

Fonte: <http://www.rittman.net/archives/001255.html>

### Vulnerabilidades no Oracle

O CERT (U.S. Computer Emergency Response Team) recentemente anunciou a descoberta de diversas falhas de segurança em diversos produtos da Oracle. Patches estão sendo disponibilizados para a correção das falhas e devem ser aplicados o mais rápido possível.

Fonte: <http://databases.about.com/gi/dynamic/offsite.htm?site=http://www.us%2Dcert.gov/cas/techalerts/TA05%2D117A.html>

### Conferência internacional de Firebird

Está marcada a data para a realização da conferência internacional de Firebird. Esse ano a conferência será em Praga, na República Checa, de 13 a 15 de Novembro, no Hotel Olsanka. Mais detalhes serão divulgados em breve.

Fonte: <http://www.FireBase.com.br>

Continua na próxima página...



**Seja um AUTOR da Editora Ciência Moderna!**

Se você possui algum manuscrito, original ou livro pronto, submeta à nossa apreciação enviando um resumo do seu trabalho para o e-mail: [lcm@lcm.com.br](mailto:lcm@lcm.com.br). Teremos o maior prazer em avaliá-lo e, se aprovado, publicá-lo em forma de livro.

## Evento brasileiro sobre Firebird (FDD)

O site da segunda edição do **Firebird Developers Day** já está online. Esse ano o evento contará com a presença de **Jim Starkey** (criador do InterBase e do Firebird Vulcan) e **Ann Harrison** (IBPhoenix). O evento será realizado no dia 16 de Julho, e todas as informações sobre o evento estão disponíveis no site oficial <http://www.FirebirdDevelopers-Day.com.br>

## Errata da Edição #001

Na edição #001 no título do artigo sobre buscal textual no PostgreSQL, o nome do banco saiu com um S a mais. Portanto onde se lê PostgresSQL, deve-se ler PostgreSQL. Falha nossa!

## Como avaliar os artigos da revista

A avaliação pode, e deve ser feita em todos os artigos, tanto na edição #001 como agora na edição #002, e nas futuras edições da DB FreeMagazine. Para você que ainda não usou esse dispositivo, observe no final de cada artigo uma box conforme a demonstrada abaixo:

Avalie esse artigo		
ÓTIMO	BOM	RUIM

Uma vez que você leu o artigo, ajude-nos a preparar a próxima edição avaliando os artigos da revista. Para isso, clique com o mouse sobre a opção que reflete sua avaliação.

**Nota:** Para que possa avaliar, você deverá estar conectado à internet.

Obviamente acompanhamos semanalmente as avaliações de cada edição para direcionarmos o conteúdo das próximas edições de acordo com a preferência do leitor.

# Entendendo o gStat – parte 2

Carlos Henrique Cantu

No primeiro artigo da série, analisamos as informações retornadas pelo gStat sobre o header de um Banco de Dados. Nessa segunda parte, analisaremos as informações sobre as páginas de dados e índices, que podem nos ajudar até mesmo a detectar possíveis problemas de desempenho.

## Dica

O gStat permite que você especifique qual (ou quais) tabelas devem ser analisadas. Para isso, passe na linha de comando o parâmetro `-t` seguido do nome da(s) tabela(s) que você deseja que sejam analisadas, separadas por espaço, e escritas com a caixa adequada. Exemplo:

```
gstat -d employee.fdb -t JOB
```

## Páginas de dados (-d)

O gStat também retorna informações sobre as páginas de dados de um BD. Um arquivo de banco de dados (vulgo .FDB) consiste de diversas páginas, com o tamanho definido no momento da criação do banco (geralmente 4096 bytes), ou na restauração de um backup. Existem páginas de diversos tipos: *páginas de índice*, *páginas de blob*, *páginas de dados*, *páginas de ponteiros*, etc.

As páginas de dados, como o próprio nome sugere, armazenam os dados das linhas (registros) das tabelas que compõem o banco

de dados.

Abaixo temos parte do retorno gerado pela execução do comando `gstat -d` no banco de dados `Employee.fdb`, especificamente sobre a tabela `JOB`:

```
JOB (129)
Primary pointer page: 157,
Index root page: 158
Data pages: 3, data page slots: 3,
average fill: 78%
Fill distribution:
 0 - 19% = 0
20 - 39% = 0
40 - 59% = 1
60 - 79% = 0
80 - 99% = 2
```

Vejamos o que cada informação quer dizer:

**Primary pointer page:** Indica a primeira página de ponteiros dessa tabela. Essa página contém os ponteiros apontando para as páginas de dados que armazenam os dados da tabela;

**Index root page:** Indica o número da primeira página de ponteiros de índices da tabela;

**Data pages:** O número total de páginas de dados já alocadas para a tabela;

**Data page slots:** O número de ponteiros atualmente alocados para páginas de dados dessa tabela (deve ser o mesmo valor de "Data Pages");

**Average fill:** a média geral de ocupação das páginas de dados alocadas;

**Fill distribution:** No exemplo acima, temos duas páginas de dados com ocupação entre 80% e 99%, e uma página de dados com ocupação entre 40% e 59%;

## Informações de registros (-r)

O parâmetro `-r` foi disponibilizado a partir do Firebird 1.5, e permite a extração de algumas informações sobre os registros do BD, como tamanho e estatísticas de versão. Na listagem abaixo, podemos visualizar essas informações referente à tabela `JOB`, em negrito:

```
JOB (129)
Primary pointer page: 157,
Index root page: 158
Average record length: 60.16,
total records: 31
Average version length: 0.00,
total versions: 0, max versions: 0
Data pages: 3, data page slots: 3,
average fill: 78%
Fill distribution:
 0 - 19% = 0
20 - 39% = 0
40 - 59% = 1
60 - 79% = 0
80 - 99% = 2
```

**Average record length:** Média do tamanho dos registros armazenados (em bytes);

**Total records:** Total de registros na tabela, incluindo os ativos e inativos;

**Average version length:** Média do tamanho dos registros de "back versions" (temporários). Registros de back version são criados quando linhas são removidas (delete) ou atualizadas (update);

**Total versions:** Total de back versions existentes na tabela;

**Max versions:** O número máximo de versões que um registro possui;

## Índices

Os índices no Firebird são estruturas na forma de “árvores BTree”. Cada tabela do banco possui sua própria IPR (*Index Page Root*). A IPR pode apontar para outras páginas de ponteiros (páginas btree), até chegar à última página da árvore, chamada de **Leaf Page**. A *Leaf Page* contém os ponteiros que levam diretamente para os registros da tabela. Quando um índice é muito pequeno, a própria IPR pode ser também a *Leaf Page*.

## Páginas de índices (-i)

Através do parâmetro `-i`, o `gfix` retorna informações sobre os índices existentes no BD. Abaixo listamos o retorno apresentado para um dos índices definidos na tabela JOB

```
JOB (129)
  Index MAXSALX (2)
    Depth: 1, leaf buckets: 1,
    nodes: 31
  Average data length: 10.00,
  total dup: 5, max dup: 1
  Fill distribution:
    0 - 19% = 1
    20 - 39% = 0
    40 - 59% = 0
    60 - 79% = 0
    80 - 99% = 0
```

Entendendo as informações apresentadas:

**Index:** Nome do índice;

**Depth:** Os índices são estruturas na forma de “árvores” (*ver box Índices*). O Depth indica quantos níveis (páginas) existem entre a primeira página de ponteiros do índice e as páginas “leaf”. Quando `depth = 1`, indica que não há nenhuma página intermediária. Se

`depth` for maior que 3, problemas de performance com os índices podem acontecer. Nesse caso, é interessante aumentar o tamanho da `page_size` do BD a fim de diminuir o `depth`;

**Leaf buckets:** Indica o número de páginas “leafs” utilizadas. As páginas “leafs” contêm informações que apontam para as linhas (registros) da tabela;

**Nodes:** Representa o total de páginas de índices que compõem a árvore;

**Average data length:** A média de tamanho das informações que compõem a chave do índice (em bytes). Note que o tamanho considerado para o cálculo da média não está diretamente relacionado com o tamanho da informação digitada, pois o Firebird usa processos de compressão sobre os dados, e a média é calculada sobre esses dados já comprimidos;

**Total dup:** O número total de linhas que apresentam chaves duplicadas/repetidas;

**Max dup:** Apresenta o total de duplicados referente à chave que possui o maior número de repetição;

**Fill distribution:** Histograma representando as páginas de índices e suas distribuições por porcentagens de ocupação. No exemplo apresentado, como há poucos dados na tabela, não há nenhuma página com ocupação maior que 19%. Em situações com grande volume de informação, uma ocupação menor que 80% sugere que devemos reconstruir o índice (desativando e ativando ele novamente), para desfragmentá-lo e ganhar performance;

## Performance e duplicados

Até a versão 1.5 do Firebird, a existência de índices com alto índice de chaves duplicadas pode causar sérios problemas de lentidão, principalmente no processo de *Garbage Collection* (coleta de lixo) e na remoção (delete) de registros. Um novo algoritmo utilizado no Firebird 2.0 acaba com esse problema.

## Conclusão

O `gStat` é uma ferramenta muito importante para avaliarmos as estatísticas internas do banco de dados, e também identificarmos possíveis problemas de lógica, principalmente no que diz respeito ao controle transacional.

Avalie esse artigo

Autor:

Carlos Henrique Cantu

Mini-curriculo

Bacharel em Ciência da Computação, microiro desde os tempos de Apple II, trabalha com desenvolvimento de sistemas há 15 anos. É consultor especializado em bancos de dados Firebird e InterBase 6.0. Mantenedor do site FireBase ([www.firebase.com.br](http://www.firebase.com.br)) e autor do livro Firebird Essencial e presidente do DUG-BR (Delphi Users Group Brasil), tendo ministrado palestras sobre Firebird para milhares de pessoas em diversas cidade do Brasil.

## Review do Livro "Firebird Essencial"

Por Artur Anjos

O aparecimento de um livro sobre Firebird produzido originalmente em língua portuguesa é uma grande alegria. Por um lado, revela a importância deste engine de base de dados na comunidade que fala português, por outro lado prova que esta mesma comunidade não se limita a seguir um projeto, integrando-se claramente no mesmo.

Carlos Henrique Cantu assina a obra. O Carlos acompanha o Projecto Firebird de perto desde a sua criação, e é indiscutivelmente um dos profissionais de língua portuguesa que mais conhece esta base de dados.

Firebird Essencial é um coletânea de artigos publicados nas revistas brasileiras ClubeDelphi e SQLMagazine, tendo sido revistos e ampliados de forma a que se enquadrem numa obra de leitura muito agradável (Infelizmente estas revistas não são publicadas em Portugal, pelo que as conheço apenas por referências).

O Firebird é apresentado logo de início. Carlos Cantu incide sobre as características

principais do Firebird, com especial destaque para a sua arquitetura: pequenos parágrafos resumem as funcionalidades do Versioning, Transações, Domains, Triggers, UDF's... Esta leitura deixa qualquer iniciante com uma introdução clara a tudo o que distingue o Firebird de outras bases de dados. O Firebird 1.5 marca a diferença com o seu antecessor InterBase, e o capítulo seguinte mostra todas as novidades.

Um capítulo incide apenas sobre a instalação



### Firebird Essencial

Primeiro livro brasileiro que trata especificamente dos recursos do SGBD Firebird (versões 1.0 e 1.5). O autor reuniu no livro todo o material produzido por ele para as revistas ClubeDelphi e SQLMagazine. Os artigos foram revisados, atualizados e muitos deles complementados, de forma a proporcionar ao leitor uma fonte de informação rica, atualizada e confiável. Um capítulo inédito sobre a criação de UDFs foi escrito exclusivamente para o livro.

Você aprenderá a instalar o SGBD, criar procedures, catálogos em CDROM, criar backups, gerenciar usuários, utilizar campos BLOB de forma adequada, identificar os tipos de dados disponíveis no Firebird, e muito mais!

Verifique o sumário do livro em [www.firebase.com.br/fb/livro/fbessencial](http://www.firebase.com.br/fb/livro/fbessencial)

[www.firebase.com.br](http://www.firebase.com.br)

em Windows, a diferença entre as versões classic e superserver, múltiplas instancias, etc.

Sendo um livro dirigido especialmente ao programador Delphi, existem vários capítulos dedicados a diferentes componentes de acesso, com especial destaque para o uso

do IBO e do driver dbExpress da UIB. Como grande parte dos utilizadores Delphi usaram no passado o Paradox, as diferenças entre os mesmos são focadas num capítulo, existindo ainda um outro dedicado à migração de Paradox para Firebird. Outros artigos focam problemas comuns para um programador Delphi: a manipulação de datas e de horas, campos BLOB, uso de Stored Procedures seleccionáveis, construção de UDF's, etc.

A programação em PSQL (a linguagem de programação dos triggers e stored procedures) é bem abordada, com diversos exemplos, o que torna a obra mais rica para qualquer programador, independente de que linguagem. Outros pontos focam diretamente o SGBD, como a criação de um instalador mínimo, os backups e restores, dicas para um uso seguro e eficaz da base de dados, a manipulação de usuários, a compreensão dos índices, os collates e os charsets, e ainda a utilização do servidor Embedded (uma novidade do Firebird 1.5).

A obra termina com um apanhado de FAQ's às perguntas mais frequentes que surgem na lista de suporte da Firebase (<http://www.firebase.com.br>).

O livro Firebird Essencial é uma obra altamente recomendável para todos aqueles que se estão a iniciar no mundo do Firebird, e para todos aqueles utilizadores que já utilizam

o Firebird e que querem tirar um maior proveito da base de dados.

O nível técnico do livro é de uma qualidade muito elevada, que espelha claramente a experiência do autor na matéria. A quantidade de exemplos práticos ajuda qualquer leitor a entender todos os temas abordados.

Todos os capítulos possuem uma estrutura clara de introdução, seguida de conteúdo, (alguns ainda referências) e conclusão, que deixam não só o leitor elucidado sobre a matéria como indicam caminhos possíveis para aprofundar os temas.

Por fim, vale a pena mencionar o cuidado com o acabamento gráfico da obra e a utilização de ícones padronizados durante todos os capítulos para indicar dicas, cuidado, recursos do FB 1.5, etc.

Para uma segunda edição, “revista e aumentada”, deixo apenas a pequena sugestão de um capítulo dedicado às formas de acesso nativo por outras linguagens que não o Delphi - como o PHP, Java, .Net - que certamente irão despertar o interesse de mais leitores.

O livro está disponível nas principais livrarias do Brasil, ou pode ser adquirido online no site da FireBase. Pessoas de outros países podem adquiri-lo através de lojas online que despachem para outros países, como o Submarino. Uma relação com diversos sites onde o livro pode ser adquirido está na página oficial do livro: <http://www.firebase.com.br/>

fb/livro/fbessencial. Uma edição especial, autografada pelo autor, é comercializada somente no site da FireBase, a um preço ligeiramente superior, valor esse que reverte para a *Fundação Firebird*.

Artur Trindade Anjos  
Maio de 2005  
Portugal

Avalie esse artigo



Vox on Demand

Sistemas de automação de voz

Automação de discagem e emissão de recados

URA - Unidade Remota de Atendimento

Gravação de ligações telefônicas

Site: <http://www.jobvox.com.br>  
E-Mail: [SAC@JOBVOX.COM.BR](mailto:SAC@JOBVOX.COM.BR)

Piracicaba São Paulo



## Precisando de cursos ou treinamentos de Firebird?

A FireBase oferece cursos/treinamentos de Firebird ministrados dentro da sua empresa. Os cursos são ministrados por **Carlos H. Cantu**, um dos maiores evangelistas do Firebird no Brasil, autor do livro Firebird Essencial. Mais informações pelo email [cursos@firebase.com.br](mailto:cursos@firebase.com.br)

**Envie seu artigo para ser  
publicado na  
DB FreeMagazine!**

**Valorize-se como profissional  
sendo um autor da  
DB FreeMagazine.**

## Entrevista com João Prado Maia

**Fale um pouco sobre você, e de como é estar envolvido em um projeto com as proporções do MySQL?**

Meu nome é João Prado Maia e trabalho na **MySQL, Inc.** há quase 2 anos, agora como um *Senior Software Engineer*. Trabalho no departamento de IT da companhia, mantendo o *Eventum*, que é uma aplicação escrita em PHP utilizado pelo departamento de suporte para lidar com os nossos clientes. Recentemente completei o desenvolvimento do MySQL Network – visando simplificar os produtos que são oferecidos pela empresa num pacote de serviços, com suporte técnico, um knowledge base e um sistema de notificação de alertas.

Nasci em Jacksonville, Florida - USA. Tenho somente um tio e alguns primos morando em Los Angeles. Estava no terceiro ano de Engenharia Química na UFRJ quando percebi que seria muito mais feliz fazendo Informática, tentando então mudar o meu curso. No meio desse processo, consegui um emprego em um setor interno da UFRJ, onde ganhei experiência com aplicações para a Web. Depois, consegui um emprego em Los Angeles para trabalhar como desenvolvedor PHP, o que fez com que eu mudasse para LA. Depois de 3 outros empregos nessa área, estou agora trabalhando para a MySQL do Texas. Nas horas vagas, também mantenho o site **PHPBrasil** :)

**Como conheceu o MySQL e a equipe de desenvolvimento?**

Em 2003, eu já estava pronto para começar a comercializar o *Eventum*, quando fui contatado por uma pessoa da MySQL, pois eles estavam interessados em uma aplicação Web para lidar com os pedidos de suporte da companhia. A empresa então comprou os direitos autorais sobre esse produto, me contratou para continuar trabalhando nele, e abriu o código fonte do *Eventum* pela licença GPL. Se alguém estiver interessado, mais detalhes estão disponíveis na URL: <http://www.mysql.com/eventum/>

**Quais as suas funções na equipe de desenvolvimento?**

Atualmente lido com todo o tipo de desenvolvimento necessário no *Eventum*, e agora no *MySQL Network* também.

**Como nasceu o MySQL? A partir de que idéia, e quem teve o insight de criá-lo?**

O MySQL nasceu no início dos anos 90 quando Monty Widenius e David Axmark trabalhavam para uma consultoria na Suécia, e estavam interessados em adicionar uma camada de SQL em cima do banco de dados que eles já tinham desenvolvido. Nessa mesma época, eles começaram a pensar em licenciar o código em GPL para conseguir mais usuários, e com isso testes e relatórios de bugs. A empresa começou pequena, e foi crescendo devagar até que no final dos anos

90 eles trouxessem investimento de fora, fazendo com que a companhia agora tenha quase 200 funcionários.

**O MySQL é baseado em GPL. Como deve ser tratado o uso do banco em projetos que não estão sob essa licença?**

Sim, o MySQL é licenciado como um projeto GPL. O código fonte é totalmente gratuito, e aberto para que você possa fazer as suas mudanças, caso necessário. Se você criar e vender um produto que necessita do MySQL para funcionar, é necessário licenciar o seu produto pelo GPL, ou comprar uma licença para utilizar o MySQL com o mesmo. Não existe nenhuma versão do servidor que você tenha que pagar para usar, mas quando se fala em distribuição do MySQL junto a um produto proprietário (ou com código fechado), aí as coisas mudam.

**Quantos membros atualmente têm a equipe de desenvolvimento?**

A equipe de desenvolvimento tem atualmente 66 membros. Temos vários grupos dentro dessa equipe, variando desde um grupo só para aplicações GUI como o *MySQL Administrator* e *MySQL Query Browser*, até um grupo específico para otimizar a performance do servidor.

**Quais as diferenças básicas entre os engines MyISAM e INNO?**

A diferença mais importante é que o *InnoDB* tem o suporte para transações.

**O MYSQL possui diversas versões de servidor. Você poderia nos descrever cada uma delas, e para que são indicadas?**

Temos 3 opções atualmente: as versões 4.0, 4.1 e 5.0.

O MySQL 4.0 é o mais estável, e por isso mais apropriado para se usar em um servidor de produção (onde os seus clientes se conectam).

As versões 4.1 e 5.0 ainda estão sofrendo muitas mudanças, mas contém recursos aguardados há muitos anos. A versão 4.1 traz suporte a sub-queries (ou sub-selects), que é muito interessante para aplicações mais complexas.

As opções atualmente na página de download são *MySQL Standard* e *MySQL Max*. A diferença é que na versão Max você tem acesso ao *NDB (MySQL Cluster)* e outras otimizações que não são encontradas na versão *Standard*.

**Você poderia citar algumas vantagens do MySQL sobre outros bancos de dados (OpenSource ou comerciais).**

Acho que a vantagem mais importante é que o MySQL é muito simples de usar e administrar. Adicione o fato de que é totalmente gratuito, e a grande oferta de outros componentes OpenSource com suporte ao MySQL, como PHP, Perl e Python, e você terá uma área de desenvolvimento muito produtiva.

**Quais os recursos mais importantes das próximas versões do MySQL?**

A versão 5.0, que ainda está em desenvolvimento, irá trazer suporte a *stored procedures, triggers* e *views*. Com certeza é uma das versões mais aguardadas da empresa, mas muitas mudanças ainda vão ocorrer.

**A maior parte dos usuários de PHP prefere utilizar o MySQL em seus scripts. Do seu ponto de vista, quais seriam os motivos que justificam essa escolha?**

Acho que isso vem com a história do PHP, onde o MySQL era o banco de dados mais fácil de usar e instalar, e os próprios desenvolvedores do PHP acabaram adicionando vários recursos específicos para o MySQL.

**Você poderia nos explicar o que são as certificações de MySQL e como elas podem ser obtidas?**

O programa de certificação da MySQL é similar aos programas que existem para produtos da Oracle ou Microsoft. Você consegue as credenciais que validam o seu

conhecimento do produto, e isso será de grande ajuda para te diferenciar no mercado de trabalho.

A MySQL fez uma parceria com a **Pearson VUE**, empresa que fornece a infra-estrutura de centros de certificação pelo mundo. Os candidatos à certificação MySQL pagam uma

taxa para fazer a prova, que pode ser realizada na própria cidade onde moram, em um dos centros de certificação da *Pearson VUE*. Mais detalhes na UR: [https://shop.mysql.com/?sub=pg&pg\\_no=15](https://shop.mysql.com/?sub=pg&pg_no=15)

**Quais são as limitações atuais do MySQL, no que diz respeito a quantidade de conexões e performance?**

Basicamente o que limita o MySQL são o hardware e sistema operacional que está sendo usado no servidor. Dependendo do sistema operacional, limites existirão no tamanho do banco de dados, ou dependendo

de como você otimizou a sua instalação, o servidor será mais rápido ou não.



## O que você pode nos falar sobre a clusterização com MySQL?

O novo produto chamado *MySQL Cluster* também é licenciado pelo GPL, ou seja, completamente gratuito. É bom lembrar que esse produto ainda está sendo desenvolvido, e por isso não é tão estável quanto o servidor MySQL normal.

## Se houvesse a possibilidade de unificação do MySQL a outro banco, OpenSource ou não, você acharia isso possível?

Não acredito que isso seja possível. Temos aqui uma empresa totalmente concentrada em continuar a prover esse sistema de qualidade, e uma união com outro banco de dados iria contra esse princípio básico.

## O que é o MaxDB?

MaxDB é a nova versão do **SAP-DB**, que é ainda desenvolvido totalmente pela **SAP** na Alemanha. A *MySQL AB* fez uma parceria com a **SAP** para prover serviços de suporte para esse banco de dados.

O servidor MySQL está sendo expandido para eventualmente ser totalmente compatível com os programas da **SAP**, e poderá ser uma nova opção de banco de dados para esse tipo de cliente.

## O que são sub-queries e onde podem ser aplicadas?

*Sub-queries* (também conhecido como *sub-select*) é um recurso que permite, em certas

ocasiões, executar em uma única query, consultas mais complexas, que normalmente precisariam de 2 ou mais queries para fornecer o resultado.

Por exemplo, se tivesse que achar uma lista dos pedidos de compra baseados em emails de seus clientes, você teria que executar 2 queries (isso num banco de dados bem normalizado):

1- Query para achar o ID dos clientes baseado nos emails:

```
SELECT cliente_id FROM clientes WHERE email IN ('cliente1@exemplo.com', 'cliente2@exemplo.com');
```

2- Guarda os Ids dos clientes num vetor para se usar na query seguinte

3- Query para achar os detalhes dos pedidos de compra para esses clientes:

```
SELECT * FROM pedidos_compra WHERE cliente_id IN (123, 124);
```

Ao invés disso, com sub-queries você pode fazer tudo numa query só:

```
SELECT * FROM pedidos_compra WHERE cliente_id IN (SELECT cliente_id FROM clientes WHERE email IN ('cliente1@exemplo.com', 'cliente2@exemplo.com'));
```

Esse é um exemplo simples, mas é em situações mais complicadas que o recurso se

torna muito interessante.

## Como está o suporte a GIS no MySQL?

O MySQL tem suporte **experimental** a GIS a partir da versão 4.1.

## Como você vê o uso do software livre no Brasil? O que estaria faltando para a disseminação em massa desse tipo de ferramenta?

Essa é uma pergunta difícil. Não sei exatamente o que está faltando para esse tipo de ferramenta se disseminar no Brasil, mas acho que é uma combinação de melhorar os produtos tendo em vista a usabilidade no dia a dia, e de ter uma infra-estrutura para que empresas brasileiras possam se sentir confortáveis em utilizar software livre nos seus serviços.

## Você poderia nos citar nomes de alguns importantes usuários do MySQL?

Google, Yahoo, Sabre (que lida com o backend do *Travelocity.com*), e várias outras grandes empresas.

## Que mensagem você enviaria aos desenvolvedores que ainda não utilizam o MySQL? E qual mensagem deixa aos que já usam o MySQL?

Acho que a mensagem mais importante é que o MySQL é muito simples de usar, funcionando

extremamente bem em plataformas Linux e Windows. Se você ainda não trabalhou com o MySQL, teste-o para o seu próximo projeto.

Existem milhares de sites pela Internet com informação sobre como usar e administrar esse servidor, então será simples começar a usá-lo.

**Avalie esse artigo**

SQL> SELECT \* FROM HOSPEDAGEM WHERE QUALIDADE="INSUPERAVEL";

+-----+  
 | WWW.BAVS.COM.BR |  
 +-----+

MySQL Firebird

+ PLANOS  
 acesse:  
**WWW.BAVS.COM.BR**  
 + INFORMAÇÕES

PLANOS DE HOSPEDAGEM		
PRO I	PRO III	SEMI D. II
Principais Características: ✓ 100 Mb espaço em disco ✓ Firebird 1.0 ✓ MySQL 3.23 ✓ PHP 4 ✓ Perl 5 ✓ CGI ✓ Diretório SSL Gratuito ✓ Configuração Gratuita ✓ Mensalidade: R\$ 29,00	Principais Características: ✓ 300 Mb espaço em disco ✓ Firebird 1.5 ✓ MySQL 4 ✓ PHP 4 / Perl 5 / CGI ✓ JSP (Tomcat) ✓ Servlet ✓ ASP .NET (Mono/C#) ✓ Configuração Gratuita ✓ Mensalidade: R\$ 69,00	Principais Características: ✓ 1 Gb espaço em disco ✓ Firebird 1.5 ✓ MySQL 4 ✓ PHP 4 / Perl 5 / CGI ✓ JSP (Tomcat) ✓ Servlet ✓ ASP .NET (Mono/C#) ✓ Configuração Gratuita ✓ Mensalidade: R\$ 145,00

clientes.com.satisfação  
 Email: info@bavs.com.br  
 Atendimento Eletrônico 24hs:  
 (19) 3421-0251  
 Vendas On-line (ambiente seguro):  
 http://www.bavs.com.br

WWW.BAVS.com.br

# Desvendando o SELECT

Luiz Paulo de Oliveira Santos

O SQL (Structured Query Language) é uma linguagem que visa padronizar e facilitar o gerenciamento de informações em bancos de dados relacionais. No entanto, a padronização dos comandos sofre com as implementações proprietárias, ou seja, cada fabricante de banco de dados, embora em sua maioria implemente ao menos em parte o SQL ANSI, implementa também características próprias que o torna diferente do SQL padrão, criando vícios de programação que acabam por prender o desenvolvedor a esse ou aquele banco.

Isso ocorre com praticamente todos os SGBDs, sejam eles comerciais, "Open Sources" ou "Freewares". Nesse artigo, trataremos instruções que operam de maneira similar na maioria dos SGBDs, entre eles o MySQL, PostgreSQL, Firebird e Oracle.

O SQL é uma linguagem bastante simples, com instruções de alto nível, mas que também permite a escrita de códigos complexos.

Nesse artigo, estaremos abordando exclusivamente o comando **SELECT**. Essa instrução é sem dúvida uma das mais utilizadas do SQL, sendo responsável por realizar consultas na base de dados.

A seguir temos um resumo da sintaxe do SELECT:

```
SELECT [ DISTINCT | ALL ] campos FROM
tabela1 [, tabela n]
[ JOIN condição ]
[ WHERE condição ]
[ GROUP BY expressão ]
[ HAVING condição ]
[ ORDER BY expressão [ASC | DESC] ]
```

A instrução SELECT não se restringe somente a sintaxe apresentada, mas será esta sintaxe o objeto do nosso estudo. Com exceção da cláusula JOIN, toda cláusula deve aparecer somente uma única vez no comando, ou seja, não há como usar um WHERE duas vezes para o mesmo SELECT, ou dois ORDER BY.

Analisando a sintaxe apresentada, temos:

Tudo que aparece entre colchetes é opcional, ou seja, você pode ou não utilizar essas cláusulas, sem que isso gere qualquer tipo de erro.

**Campos:** São as colunas retornadas pela instrução. Pode ser empregado um coringa "\*" quando se deseja recuperar todos os campos. A maioria dos SGBDs exige que a cláusula FROM esteja presente no SELECT.

**DISTINCT | ALL:** Indica se o select deve descartar informações repetidas, ou se deve trazer todas as linhas encontradas. A cláusula ALL é o padrão, podendo ser omitida, e recupera todas as linhas/registros.

**Tabela1 – Tabela n** São exemplos de nomes de tabelas, sendo que quando mais de uma tabela forem especificadas, os nomes devem ser separados por vírgula. Podemos também designar apelidos para as tabelas

(alias), indicando-os logo após o nome de cada tabela. Os apelidos são válidos apenas para o select em questão.

**Condição:** Fator pelo qual a query irá filtrar os registros. Podemos utilizar operadores lógicos nas comparações como, por exemplo, OR (ou), AND (e), etc.

**Expressão:** São as informações pela qual a cláusula irá operar. Pode ser um campo, lista de campos, ou em alguns casos até mesmo uma condição.

Estaremos utilizando como exemplo neste artigo duas tabelas (pessoas e acessos), com a seguinte estrutura:

PESSOAS		ACESSOS	
ID	INTEGER	ID	INTEGER
UNAME	CHAR(15)	DATA	DATETIME
NOME	CHAR(50)	PESSOA	CHAR(15)
IDADE	INTEGER	QTDE	INTEGER

## Cláusula FROM

Basicamente a cláusula FROM é utilizada para indicar de onde será extraída a informação retornada pela query, ou seja, de quais views, tabelas, ou em alguns bancos até mesmo stored procedures. Exemplo de utilização:

```
SELECT * FROM PESSOAS;
```

O \* (asterisco) indica que desejamos receber todos os campos da tabela PESSOAS. Podemos no lugar do \* (asterisco) indicar

o nome do campo desejado, ou nomes dos campos separados por vírgula.

Também é possível designar apelidos para as tabelas (alias), indicando-os logo após o nome da tabela. Os apelidos são válidos apenas para o select em questão, e são associados às colunas recuperadas, determinando portanto à qual fonte de dados a coluna está associada. Exemplo:

```
SELECT UNAME as "USER NAME"  
FROM PESSOAS;
```

*Obs.: No exemplo anterior, a cláusula AS é opcional e poderia ser removida.*

A definição explícita de qual é a fonte de dados de uma determinada coluna torna-se obrigatória quando mais de uma fonte de dados envolvida no select possui campos com o mesmo nome. Abaixo mostramos um exemplo disso com duas tabelas:

```
SELECT pessoas.codigo,  
       vendedores.codigo,  
       vendedores.codpes,  
       vendedores.nome  
FROM pessoas, vendedores  
WHERE pessoas.codigo = vendedores.  
codpes  
ORDER BY vendedores.nome;
```

Nesse exemplo, observe que para toda coluna sendo recuperada, foi especificado o nome da tabela a quem ela pertence. Isso é necessário pois o campo CODIGO existe tanto na tabela PESSOAS como na VENDEDORES.

Recomendo que seja sempre especificado qual é a fonte de dados de cada coluna, pois

torna o select mais legível.

Ainda no mesmo exemplo, podemos também atribuir apelidos para as tabelas, desta forma enxugando o código e tornando-o ainda mais legível.

Abaixo segue o mesmo exemplo anterior, só que usando apelidos (aliases):

```
SELECT PES.CODIGO,  
       VEN.CODIGO,  
       VEN.CODPES,  
       VEN.NOME  
FROM PESSOAS PES, VENDEDORES VES  
WHERE PES.CODIGO = VEN.CODPES  
ORDER BY VEN.NOME;
```

## Comentários

Comentários no SQL podem ser representados por -- (dois sinais de menos, sem espaços) que definem que o restante da linha é um comentário, ou /\* e \*/ para comentar um trecho dentro do código SQL, que pode inclusive se estender por mais de uma linha.

Exemplo:

```
SELECT *  
FROM PESSOAS -- Aqui é comentário  
WHERE 1;
```

OU

```
SELECT *  
FROM /* comentário */ PESSOAS  
WHERE 1;
```

## Dica

Os bancos de dados citados nesse artigo permitem cálculos diretamente pelo SELECT, sem o uso da cláusula FROM, ou seja o uso do SELECT para obter resultado de cálculos aritméticos básicos. Como no exemplo abaixo:

```
SELECT 10 - 2;  
10-2  
8
```

## Cláusula JOIN

A cláusula JOIN é empregada para permitir que um mesmo select recupere informações de mais de uma fonte de dados (tabelas, views, etc.). Em geral, as tabelas referenciadas possuem algum tipo de relacionamento entre elas, através de um ou mais campos que definam a ligação entre uma tabela e a outra (integridade referencial).

Há duas maneiras de implementar um join: a primeira é chamada de **non-ANSI** ou estilo **theta**, que utiliza a cláusula WHERE para efetuar a junção de tabelas; a segunda é chamada de **ANSI Join**, e é baseada no uso da cláusula JOIN propriamente dita.

### Simple ligação

Um exemplo de JOIN em estilo ANSI:

```
SELECT p.uname, p.nome, a.qtde  
from PESSOAS p  
CROSS JOIN ACESSOS a;
```

Um exemplo de JOIN em estilo *theta*:

```
SELECT p.unicode, p.nome, a.qtde
from PESSOAS p, ACESSOS a;
```

Note que na chamada ANSI utilizamos CROSS JOIN, que é a sintaxe utilizada para recuperar todos os registros das tabelas ligadas, formando um produto cartesiano. É basicamente um INNER JOIN (citado adiante) sem condições.

## Tipos de junções

### Inner Joins

Somente as linhas/registros que satisfaçam a ligação determinada pelo JOIN serão recuperados pelo select, sendo assim, os registros que **não** se enquadram no relacionamento definido pelo join **não serão recuperados**.

Um exemplo de INNER JOIN em estilo ANSI:

```
SELECT p.unicode,
       p.nome,
       a.qtde
from PESSOAS p
INNER JOIN ACESSOS a
        on p.unicode=a.pessoa
order by p.unicode;
```

O mesmo JOIN em estilo *theta*:

```
SELECT p.unicode,
       p.nome,
       a.qtde
```

```
from PESSOAS p, ACESSOS a
WHERE p.unicode = a.pessoa
order by p.unicode;
```

### Left Joins

Através do uso do **LEFT**, todos os registros na tabela à esquerda da query serão listados, independente de terem ou não registros relacionados na tabela à direita. Nesse caso, as colunas relacionadas com a tabela da direita voltam nulos (NULL).

Um exemplo de uso LEFT JOIN:

```
SELECT p.unicode,
       p.nome,
       a.pessoa,
       a.qtde
from PESSOAS p
LEFT JOIN ACESSOS a
        on p.unicode=a.pessoa
order by p.unicode;
```

No exemplo acima, todos os registros da tabela PESSOAS serão listados, independente de terem ou não registros associados na tabela ACESSOS. Caso não existam registros associados na tabela ACESSOS, os campos *a.pessoa* e *a.qtde* retornarão NULL.

### Right joins

É o inverso do Left Join, ou seja, todos os registros da tabela à direita serão listados, independente de terem ou não registros relacionados na tabela à esquerda.

Um exemplo de uso RIGHT JOIN:

```
SELECT p.unicode,
       p.nome,
       a.pessoa,
       a.qtde
from PESSOAS p
RIGHT JOIN ACESSOS a
        on p.unicode=a.pessoa
order by p.unicode;
```

Ou seja, todos os registros da tabela ACESSOS serão listados, e caso não haja correspondentes na tabela PESSOAS, a query devolve NULL para os campos p.unicode e p.nome.

## Cláusula WHERE

A cláusula WHERE permite aplicar filtros sobre as informações vasculhadas pelo SELECT. É extremamente abrangente e permite gerar condições complexas de pesquisa.

Os operadores aceitos pela cláusula WHERE são: =, >, <, <>, >=, <= e **BETWEEN**. Há também a possibilidade de se utilizar operadores de proximidade como o LIKE, que permite realizar buscas por apenas uma parte de um string. Pode-se negar uma comparação com o operador **NOT**, bem como utilizar operadores lógicos (**OR** ou **AND**).

Outros operadores suportados são:

- **IN**: verificar se um valor está contido em um grupo de valores;
- **EXISTS**: verifica se um valor existe no resultset retornado por um select;

Vale a pena lembrar que podemos utilizar parênteses, determinando a ordem das condições a serem aplicadas.

### NULO ou NÃO NULO, eis a questão...

NULL determina um estado e não um valor, por isso deve ser tratado de maneira especial. Quando queremos saber se um campo é nulo, a comparação a ser feita é "campo **is null**" e não "campo = null", sendo que essa última sempre retornará FALSO. Do mesmo modo, para determinar se um campo não é nulo, usamos "campo **is not null**".

## Exemplos de utilização

Adiante temos exemplos de aplicação de WHEREs:

```
SELECT *
from PESSOAS
WHERE (IDADE >= 90);
```

*Filtra todos os registros da tabela PESSOAS no qual o campo IDADE é maior ou igual a 90.*

```
SELECT *
FROM PESSOAS
WHERE (IDADE >= 1 AND IDADE <= 17);
```

ou

```
SELECT *
FROM PESSOAS
WHERE IDADE BETWEEN 1 AND 17;
```

*Lista todos os registros da tabela PESSOAS cujo conteúdo do campo IDADE seja maior e igual a 1 e menos e igual a 17.*

```
SELECT *
from PESSOAS
WHERE NOME LIKE 'JO%';
```

ou

```
SELECT *
from PESSOAS
WHERE NOME STARTING WITH 'JO'; -- No
Firebird
```

*Retorna todos os campos e todos os registros da tabela pessoa que possuam as letras "JO" como iniciais do nome.*

```
SELECT *
FROM PESSOAS
WHERE NOME LIKE '%AN%';
```

ou

```
SELECT *
FROM PESSOAS
WHERE NOME CONTAINING 'AN';
```

No caso anterior, retorna todos os registros que possuam no campo nome as letras AN, seja no começo, meio ou fim do campo.

Outro coringa que pode ser utilizado é o "\_" (underline). Podemos utilizá-lo quando desejarmos mascarar uma ou mais letras. Por exemplo:

```
SELECT *
FROM PESSOAS
WHERE NOME LIKE "LUI_";
```

*Nesse caso, obteremos todos os registros cujo campo NOME possui as três primeiras letras LUI, sendo que a quarta letra pode ser qualquer caractere.*

```
SELECT *
FROM ACESSOS
WHERE QTDE IS NULL;
```

*Retorna todos os campos e todos os registros onde o campo QTDE é NULL.*

```
SELECT PE.UNAME,
       PE.NOME
FROM PESSOAS PE
WHERE EXISTS(SELECT * from ACESSOS
             WHERE PESSOA = PE.UNAME);
```

*Retorna os campos UNAME e NOME de todos os registros da tabela PESSOAS que possuem registros na tabela ACESSOS. O uso da cláusula EXISTS é na verdade uma junção com uma subquerie (segundo select). Nesse caso, somente serão exibidos os registros da query que atendam a condição da subquery. Caso um registro na tabela pessoas não possua um registro correspondente na tabela ACESSOS, ele não será recuperado.*

```
SELECT *
FROM PESSOAS
WHERE IDADE IN (32,24);
```

Serão retornados os registros cuja idade é 32 ou 24. Poderia ser implementada por um OR, mas quando a lista de campos na condição é extensa, o OR pode deixar a instrução muito longa e de difícil compreensão.

## Dica

Pode-se utilizar o NOT em conjunto com praticamente todos os operadores de condição. Exemplo: NOT com o BETWEEN (NOT BETWEEN), com o LIKE (NOT LIKE), com o IN (NOT IN) e outras tantas maneiras. Dessa forma, as possibilidades de filtragem aumentam significativamente.

## Cláusula GROUP BY

As cláusulas GROUP BY e HAVING são geralmente utilizadas quando utilizamos funções de agrupamento. As principais funções de agrupamento são:

SUM	Soma todos os valores da coluna informada
MAX	Retorna o maior valor da coluna especificada.
MIN	Retorna o menor valor da coluna informada.
COUNT	Retorna o número de registros da tabela.
AVG	Retorna a média aritmética dos valores da coluna informada.

## Você sabia?

Os registros cujo campo é NULL são desprezados quando se utiliza uma função de agregação.

Exemplos:

```
SELECT SUM(QTDE) FROM ACESSOS;
```

*Query devolve a somatória do campo QTDE da tabela ACESSOS.*

```
SELECT MAX(QTDE) FROM ACESSOS;
```

*Query devolve o maior valor do campo QTDE da tabela ACESSOS.*

```
SELECT NOME,  
       COUNT( AC.QTDE)  
FROM PESSOAS PE, ACESSOS AC  
WHERE PE.UNAME = AC.PESSOA  
GROUP BY(AC.PESSOA);
```

## Você sabia?

As funções SUM e AVG somente podem ser utilizadas em campos com tipos numéricos. As funções MAX e MIN podem ser utilizados em numéricos, data e com caracteres também, e COUNT em campos de qualquer tipo de dado.

Quando um mesmo select recupera tanto campos individuais e valores agrupados através das funções de agrupamento, é necessário utilizar o *group by* listando nele as colunas individuais, para que o select possa ser executado.

## Cláusula HAVING

A cláusula HAVING aplica um filtro sobre o resultado de um GROUP BY e, portanto, é

utilizada em conjunto com o mesmo. Ela não interfere no resultado obtido na Query, pois age após a totalização, ou seja, não altera resultados de totalizações, apenas filtra o que será exibido. Exemplo:

```
SELECT PESSOA, SUM(QTDE) AS TOTAG  
FROM ACESSOS  
GROUP BY PESSOA  
HAVING TOTAG > 5;
```

ou

```
SELECT PESSOA, SUM(QTDE) AS TOTAG  
FROM ACESSOS  
GROUP BY PESSOA  
HAVING SUM(QTDE) > 5;
```

A query acima agrupa os registros pelo campo PESSOA, depois soma os conteúdos do campo QTDE retornando-os com o alias TOTAG (por pessoa), e somente exibe os que possuem TOTAG maior que 5.

## Cláusula ORDER BY

Como o próprio nome sugere, essa cláusula ordena informações obtidas em uma query, de forma [ASC]endente (menor para o maior, e da letra A para Z), que é o padrão e pode ser omitida, ou de maneira [DESC]endente.

O NULL, por ser um estado e não um valor, aparece antes de todos na ordenação. Alguns SGBDs permitem especificar se os nulls devem aparecer no início ou fim dos registros, como por exemplo o Firebird, com o uso das cláusulas **NULLS FIRST** e **NULLS LAST**. Exemplo:

```
SELECT * FROM  
PESSOAS  
ORDER BY PESSOA;
```

A query acima retorna os registros da tabela PESSOAS ordenados por PESSOA de maneira ascendente.

Pode-se ordenar por "n" colunas, ou seja, fazer uma sub-ordenação. Isso é feito indicando os campos separados por vírgula, conforme a query abaixo:

```
SELECT *  
FROM ACESSOS  
ORDER BY PESSOA, QTDE;
```

Desta maneira, a query retornará o resultado ordenado por PESSOA e sub-ordenado pela QTDE.

Pode-se também montar queries onde se ordena de maneira ascendente uma coluna, e de maneira descendente outra. Exemplo:

```
SELECT *  
FROM ACESSOS  
ORDER BY PESSOA DESC, QTDE ASC;
```

A query retorna todos os registros da tabela ACESSOS ordenados pelo campo PESSOA de maneira descendente, e pelo campo QTDE na forma ascendente.

O ORDER BY também permite ordenações referenciando as colunas como números. Assim, a primeira coluna retornada no query é a 1, a segunda é a 2, e assim sucessivamente. Desta forma, podemos escrever códigos conforme o abaixo:

```
SELECT CAMPO1, CAMPO2, CAMPO3, CAMPO4  
FROM ACESSOS  
ORDER BY 3;
```

No exemplo, a ordenação será através da coluna CAMPO3.

## Dicas adicionais

Existem recursos adicionais que podem ser utilizados nos SELECTs. Geralmente são funções e variáveis que podem nos poupar bastante trabalho. Abaixo listo algumas funções interessantes, presentes na maioria dos bancos de dados, sejam de forma nativa, seja na forma de UDFs:

Funções ANSI para tratar Strings:

### Lower( campo )

Devolve o conteúdo do campo em minúsculos

### Upper( campo )

Devolve o conteúdo do campo em maiúsculos

### Character\_Length( campo )

Devolve o tamanho ocupado em bytes do campo.

### Position( string1 IN string2 )

Busca pelo String1 na String2. Se encontrar devolve o offset de posição, do contrário devolve 0 (zero). O LIKE internamente em alguns bancos utiliza essa função.

### Concat( String1, String2, ..., StringN )

Devolve um String que é a concatenação dos strings informados. Exemplo:

```
SELECT CONCAT( UNAME, '-', NOME ) AS  
"NOME" FROM PESSOAS;
```

## Funções para implementar rotinas de segurança no banco:

### System\_user()

Devolve o nome do usuário do sistema operacional para o servidor SQL.

### Session\_user()

Devolve um string com a autorização da sessão SQL atual.

### User()

Devolve o nome do usuário ativo. No Firebird é CURRENT\_USER.

## Constantes de data e hora:

### Current\_Date

Devolve a data atual

### Current\_Time

Devolve a hora atual

Existem otimizações que poderiam ser feitas nas tabelas utilizadas em nossos exemplos como, por exemplo, a implementação de índices. Em um próximo artigo iremos falar sobre otimizações do banco.

## Diferenças entre bancos

Cada banco de dados possui sintaxes próprias, que adicionam recursos e facilidades ao SELECT padrão. Infelizmente não dispomos de recursos para escrever todas essas diferenças em um único artigo, pois ele se tornaria muito extenso.

Porém, se você quiser ter essas informações detalhadas, avalie esse artigo no site da DB FreeMagazine deixando-nos sugestões do que você gostaria de encontrar na revista. Para isso, deixe uma mensagem para **LPAULO** diretamente no site da revista. Basta se logar, selecionar CAIXA DE ENTRADA a partir do MENU PESSOAL e ENVIAR UMA MENSAGEM.

Em nosso próximo artigo estaremos escrevendo sobre outras duas instruções: INSERT e UPDATE.

Um forte abraço a todos.

Luiz Paulo

Autor:

Luiz Paulo de Oliveira Santos

Mini-curriculo

Luiz Paulo de Oliveira Santos é formado em Tecnologia de Processamento de Dados, especialista em Análise de Sistemas e Redes de Computadores. É analista de suporte de redes na Universidade Metodista de Piracicaba e diretor da JobVox Sistemas Informatizados. lpaulo@jobvox.com.br

Avalie esse artigo

# PostgreSQL 8.0 Checklist de Performance

JOSH BERKUS

Esse documento é uma lista de regras para configuração do PostgreSQL 8.0. Muitas das informações citadas abaixo são relatos de evidências ou testes práticos escaláveis; Nós e a OSDL (Open Source Development Labs) ainda estamos trabalhando muito na questão de performance do BD. Todas as informações contidas nesse artigo são válidas em 12 de janeiro de 2005 e provavelmente serão atualizadas no futuro. Discussões sobre as configurações abaixo poderão aparecer posteriormente às recomendações.

## Cinco princípios de hardware para configurar seu servidor PostgreSQL:

- **Discos > RAM > CPU**  
Se você vai investir dinheiro no seu servidor PostgreSQL, gaste-o em array de discos de alta performance, em um processador de média capacidade e memória RAM adequada. Caso você não tenha muita verba disponível, invista apenas em memória RAM. O PostgreSQL, como qualquer outro SGBDR compatível com o padrão ACID (Atomicity, Consistency, Isolation, Durability) utilizam muito mais os dispositivos de I/O

(dispositivos SCSI de armazenamento) do que a CPU. Compre uma CPU com bom custo-benefício se isso permite que você também adquira uma boa controladora RAID e vários discos.

- **Mais discos == melhor**  
Como o uso de múltiplos discos, o PostgreSQL e o sistema operacional irão realizar as leituras e gravações no banco de dados de forma paralela. Isso reflete em uma enorme diferença no processamento de transações do sistema, e uma significativa melhora em qualquer aplicação onde o banco de dados não caiba totalmente na memória RAM. Como os menores discos atualmente possuem capacidade de aprox. 72 GB, você pode usar apenas um disco, ou um par de discos espelhados em um RAID 1. Entretanto, descobrirá que usando 4, 6 ou até 14 discos haverá um bom acréscimo de performance. Usando SCSI, o ganho é ainda mais significativo se comparado com IDE ou mesmo Serial ATA.
- **Separando o log de transação do banco de dados**  
Assumindo que você já tenha separado o dinheiro para comprar um array de discos de tamanho decente, ainda há alternativas mais inteligentes do que simplesmente jogar tudo em um simples RAID. Colocando os logs transacionais do banco de dados (pg\_xlog) num disco dedicado (seja ele um array ou disco isolado), aumenta praticamente em 12% a performance do banco de dados em situações de alta atividade de escrita. Isso é especialmente

indicado para sistemas com SCSI lentos ou com discos IDE. Mesmo em um servidor com dois discos, pode-se deixar o log no disco do sistema operacional e conseguir algum ganho de performance.

- **RAID 1+0/0+1 > RAID 5**  
O sistema de RAID 5 com 3 discos tem se tornado um padrão infeliz entre os fabricantes de servidores econômicos. Essa é possivelmente a configuração de arrays mais lenta para o PostgreSQL. Provavelmente, as queries teriam apenas 50% da performance comparado com um simples disco SCSI. É melhor focar no RAID 1 ou 1+0 ou 0+1 para qualquer conjunto de 2, 4 ou 6 discos. Acima de 6 discos, o RAID 5 começa a ter uma performance aceitável, e a comparação então tende a se voltar para a placa controladora. Uma controladora barata pode ser responsável por uma performance ruim, e não é raro ter casos onde um RAID emulado por software torna-se melhor do que usar uma controladora adaptec "onboard" no seu servidor.
- **Aplicações precisam rodar bem juntas:**  
Outro grande erro que costumo observar é que muitas organizações estão colocando o PostgreSQL em um servidor junto com outras aplicações que competem pelos mesmos recursos. A situação piora ainda mais se colocarmos o PostgreSQL e outros SGBDRs na mesma máquina, pois ambos os bancos de dados irão disputar a banda de disco, cache do sistema operacional e consequentemente ambos terão performances ruins. Servidores de arquivos

e programas de log de segurança são uma analogia a situação citada. O PostgreSQL pode compartilhar a máquina com uma aplicação que utilize mais CPU e memória RAM, como Apache por exemplo, mas certifique-se que há memória suficiente para ambos.

## Doze configurações para você ajustar no arquivo PostgreSQL.conf:

Existe um bom número de novas e assustadoras opções de configuração no arquivo PostgreSQL.conf. Até mesmo opções já familiares nas últimas 5 versões do PostgreSQL tiveram seus nomes e formato de parâmetros alterados. Isso foi feito com a intenção de dar mais controle ao administrador de banco de dados, mas pode levar um tempo até se acostumar com elas.

A seguir, listaremos algumas configurações focando principalmente a performance. Há alguns ajustes específicos que a maioria dos usuários não irão utilizar, mas que outros acharão indispensáveis.

*Lembre-se: As opções do PostgreSQL.conf devem ser descomentadas para terem efeito, mas recomentá-las nem sempre significa que os valores padrões serão restaurados!*

### Connecting

**listen\_addresses:** Substitui as configurações **tcp\_ip** e **virtual\_hosts** da versão 7.4. Padrão para localhost em muitas instalações, habilita

somente conexões a partir do console (no próprio servidor). Muitos administradores desejarão configurá-la para "\*", de maneira a disponibilizar todas as interfaces, após configurar as devidas permissões no arquivo pg\_hba.conf para tornar o PostgreSQL acessível na rede. Como uma melhoria sobre as antigas versões, o "localhost" padrão não permite conexões na interface de "loopback", 127.0.0.1, habilitado por diversos utilitários baseados em browser.

**max\_connections:** exatamente como nas versões anteriores, essa configuração deve ser configurada para o número atual de conexões simultâneas que você espera utilizar. Usar valores altos irá requerer mais memória compartilhada (shared memory). Caso o número de conexões exigidas seja muito alto, o ideal é usar de maneira eficaz um pool de conexões. Por exemplo, 150 conexões ativas em um servidor médio, mono-processado de 32 bits rodando Linux irá consumir de maneira significativa os recursos do sistema, sendo que 600 conexões seria o "limite" do hardware. Claro que o hardware, mesmo que sobrecarregado, irá permitir mais conexões.

### Memory

**shared\_buffers:** Um lembrete: Esta informação não determina o total de memória que o PostgreSQL tem para trabalhar. Indica o bloco de memória dedicada que o PostgreSQL pode usar para operações ativas, e pode ser uma pequena parte do total de memória RAM da máquina, desde que o PostgreSQL use o cache de disco do Sistema Operacional sem

problemas. Infelizmente, o montante exato de **shared\_buffers** requerido é obtido através de um complexo cálculo que deve levar em conta o total de memória RAM, tamanho do banco de dados, número de conexões e complexidade das queries. Assim, é melhor ter algumas regras para sua alocação, e monitorar o servidor (particularmente pg\_stat) para determinar os ajustes.

Em servidores dedicados, valores comumente usados estão entre 8MB e 400MB (entre 1.000 e 50.000 páginas de 8K). Fatores que podem influir no aumento do **shared\_buffers** são porções ativas do bancos de dados, queries grandes e complexas, grande número de queries simultâneas, procedures ou transações longas, aumento da memória RAM disponível e aumento no poder de processamento da CPU (incluindo SMP) e, é claro, ter outras aplicações rodando na mesma máquina. Contrariando algumas expectativas, alocando-se muito **shared\_buffers** pode baixar a performance e aumentar o tempo requerido para buscas. Abaixo segue informações obtidas em comentários e testes TPC em servidores Linux:

**Laptop:** processador Celeron, 384MB RAM, banco de dados de 25MB  
Sugestão: 12MB/1500

**Servidor:** processador Athlon, 1GB RAM, banco de dados de suporte à decisão de 10GB  
Sugestão: 120MB/15000

**Servidor:** quad-processado Pentium III, 4GB RAM, Banco de dados de 40GB de

pesado processamento transacional com 150 conexões  
**Sugestão:** 240MB/30000

**Servidor:** quad-processado Xeon, 8GB RAM, Banco de dados de 200GB de pesado processamento transacional com 300 conexões  
**Sugestão:** 400MB/50000

Note que aumentar o **shared\_buffers**, e alguns outros parâmetros de memória, irá requerer que você modifique os parâmetros do sistema operacional System V (unix). Veja a documentação do PostgreSQL para maiores informações.

**work\_mem:** chamado anteriormente de **sort\_mem**, foi renomeado desde que passou a ser associado à Sort, agregando um pouco de outras operações. O parâmetro trata a memória não compartilhada, que é alocada por operação (entre 1 e *n* vezes por query), essa opção existe para estabelecer um teto no montante de memória RAM que qualquer operação pode gerenciar antes de forçar o flush em disco. Isso pode ser calculado dividindo-se a memória RAM disponível (subtraindo memória das aplicações e **shared\_buffers**) pelo maior número de queries esperadas, multiplicando pela média da quantidade de memória usada por operação por query.

Considerações precisam ser feitas ao valor de **work\_mem** necessário a cada query; processando grandes volumes de dados irá requerer um valor mais alto. Aplicações WEB geralmente exigem valores baixos, pois geralmente o número de conexões é

alto e as queries são simples; 512K à 2048K geralmente são suficientes. Em contrapartida, aplicações de suporte à decisão com queries de 160 linhas e 10 milhões de registros agrupados necessitam de um valor maior, como 500 MB em um servidor. Para bancos de dados em situações diversas (bancos grandes e pequenos), esse parâmetro pode ser configurado por conexão, no momento da query, pedindo mais memória RAM para algumas queries específicas.

**maintenance\_work\_mem:** anteriormente chamado **vacuum\_mem**, essa é a quantidade de memória RAM que o PostgreSQL usa para VACUUM, ANALYZE, CREATE INDEX e também para chaves estrangeiras (foreign keys). Quanto maior o tamanho das suas tabelas e a quantidade de RAM disponível, maior deverá ser o valor desse parâmetro. Uma configuração de 50% a 75% do tamanho ocupado no disco pela sua maior tabela ou índice é uma boa regra, ou então um valor entre 32MB e 256MB quando esse fator não puder ser determinado.

## Discos e WAL

**checkpoint\_segments:** define o tamanho do cache de disco do log de transações para operações de escrita. Você pode ignorá-lo em aplicações onde predominam as leituras, como aplicações WEB, mas para bancos com processamento transacional ou para geração de relatórios envolvendo grande carga de dados, é fundamental o aumento do valor para melhorar a performance. Dependendo do volume de dados, comece com um valor

entre 12 e 256 segmentos, e aumente caso observe mensagens de alerta no log. O espaço requerido no disco é igual a  $(\text{checkpoint\_segments} * 2 + 1) * 16 \text{ MB}$ , então certifique-se há espaço disponível no disco. **Exemplo:**  $32 \Rightarrow (32 * 2 + 1) * 16 \text{ MB} = 1040$ , logo 1 GB pelo menos é necessário.

**max\_fsm\_pages:** configura o tamanho do registro que gerencia as páginas de dados parcialmente preenchidas e que podem receber novos dados; se configurado corretamente, agiliza o processo de VACUUM e descarta a necessidade do VACUUM FULL ou REINDEX. Deve ser um pouco maior que o número total de páginas de dados que serão “tocadas” pelos processos de *updates* e *deletes* entre os *vacuums*. As duas formas de se determinar esse número é rodar o VACUUM VERBOSE ANALYZE, ou se estiver usando o autovacuum (ver abaixo) configure de acordo com o parâmetro -V como sendo uma porcentagem do total de páginas de dados usados pelo seu BD. As *fsm\_pages* requerem muito pouca memória, portanto é melhor ser generoso aqui.

**vacuum\_cost\_delay:** Se você possui tabelas muito grandes e uma quantidade significativa de escrita concorrente no banco, você pode se beneficiar desse novo recurso que diminui o uso de I/O dos VACUUMs, pelo preço de tornar o processo de limpeza mais lento. Como esse é um recurso recente, ele é um complexo de cinco configurações dependentes das quais nós temos apenas alguns testes de performance. Configurar o *vacuum\_cost\_delay* para um valor diferente

de zero ativa esse recurso; use um valor razoável, entre 50ms e 200ms. Para um processo de sintonia fina, aumentar o valor de `vacuum_cost_page_hit` e diminuir o valor de `vacuum_cost_page_limit` irá suavizar o impacto dos vacuums, fazendo com que eles demorem mais para terminar; Nos testes em processos de transação realizados por Jan Wieck, um delay de 200, `page_hit` de 6 e limite de 100, diminuiu o impacto do vacuum em mais de 80%, triplicando o tempo de duração do processo.

## Otimizador de Queries

Esta configuração habilita o otimizador de queries para fazer estimativas exatas de custos da operação, e assim escolher o melhor plano de execução de uma query. Há dois ajustes globais:

**effective\_cache\_size:** informa ao otimizador o maior objeto possível do BD que espera-se entrar no cache. Geralmente deve ser configurado para 2/3 da RAM, no caso de um servidor dedicado. Em um servidor misto, você deverá estimar o quanto de memória será utilizada pelas outras aplicações e subtrair esse valor.

**random\_page\_cost:** estima o custo de se fazer buscas indexadas em páginas de dados já bufferizadas. Em máquinas velozes, com arrays de disco, esse valor deve ser decrementado para 3.0, 2.5 ou mesmo 2.0. No entanto, se a porção ativa do seu BD é muitas vezes superior que a sua RAM, você poderá querer voltar o valor para o padrão

(4.0). Como alternativa, você pode realizar os ajustes com base na performance de algumas queries. Se o otimizador se mostrar favorável a realizar pesquisas seqüenciais ao invés de indexadas, abaixe o valor. Se ele estiver usando índices lentos quando não deve, aumente o valor. **Tenha certeza de testar diversas queries. Não coloque valores inferiores a 2.0;** se isso se mostrar necessário, você deve mexer em outras áreas, como as estatísticas dos planos.

## Logging

**log\_destination:** substitui o **syslog setting** das versões anteriores. Suas opções são: usar o log administrativo do Sistema Operacional (syslog ou eventlog) ou usar um log separado do PostgreSQL (stderr). A primeira opção é a melhor para monitoração do sistema; a última é melhor para detectar problemas no banco de dados e ajustes (tuning).

**redirect\_stderr:** Se você decidir por utilizar um logo separado para o PostgreSQL, essa opção permite que você gere o log diretamente em um arquivo, usando um utilitário nativo do PostgreSQL ao invés de redirecionamento de linha de comando, permitindo uma rotatividade automática dos logs. Configure para True, e depois configure o parâmetro `log_directory` indicando onde os arquivos de logs serão gerados. A configuração padrão dos parâmetros `log_filename`, `log_rotation_size`, e `log_rotation_age` geralmente são adequadas para a maioria das pessoas.

## Autovacuum e você

Na medida em que você utilizar o PostgreSQL 8 em produção, poderá determinar um plano de manutenção que inclui VACUUMs e ANALYZEs. Se seu banco de dados possui um fluxo razoavelmente constante de escrita de dados, mas não requer uma carga massiva de dados e remoções ou freqüentes “restarts”, isso pode indicar o uso do **pg\_autovacuum**. Ele é melhor que o agendamento de vacuums porque:

As tabelas sofrem o vacuum baseadas em sua atividade, evitando o vacuum em tabelas somente-leitura.

A freqüência de vacuums cresce automaticamente com o aumento da atividade no banco de dados.

É fácil calcular o mapa do espaço livre requerido e evitar o inchaço no banco de dados.

Configurar o autovacuum requer a compilação do módulo no diretório `contrib/pg_autovacuum` do código fonte do PostgreSQL (Os usuários do PostgreSQL para Windows encontrarão o AutoVacuum incluído no pacote do PGInstaller). Verifique a configuração de stats descrita no arquivo README. Execute o autovacuum depois que o PostgreSQL foi carregado, como um processo separado, sendo que ele será encerrado automaticamente quando o PostgreSQL for baixado.

A configuração padrão para o autovacuum é bastante conservadora, embora seja apropriada para uma base de dados muito pequena. Geralmente costumo usar

uma configuração mais agressiva, como:

```
-D -v 400 -V 0.4 -a 100 -A 0.3
```

O vacuum ocorrerá nas tabelas depois que 400 linhas + 40% da tabela tiver sido atualizada ou apagada, e a análise será feita depois que 100 linhas + 30% da tabela tiver sofrido inserts, updates ou deletes. A configuração acima também leva `max_fsm_pages` para 50% das páginas de dados do banco, com confiabilidade de que esse número não será excedido, causando o inchaço do banco de dados. Nós estamos testando várias configurações no OSDL (Open Source Development Labs) e teremos mais informações a respeito disso em breve.

Note que você também pode usar as configurações de autovacuum para indicar a opção "vacuum delay", ou seja, tempo para vacuum. Essa configuração encontra-se no `PostgreSQL.conf`. O Vacuum delay pode ser de vital importância para sistemas com tabelas ou índices enormes, de outra forma, a execução fora de hora do vacuum poderá interromper uma operação importante.

Existe infelizmente algumas sérias limitações no PostgreSQL 8.0 autovacuum, que provavelmente serão eliminadas nas versões futuras. São elas:

- Não tem memória permanente: o autovacuum se esquece de toda a atividade já realizada quando você reinicia o servidor. Assim, se você reiniciar regularmente o servidor, terá que fazer um VACUUM ANALYZE

antes ou depois.

- Não analisa como o servidor é utilizado: existem planos de se checar a carga do sistema antes de efetuar um "vacuuming", mas não é um recurso disponível atualmente. Caso você tenha picos de extrema carga no servidor, o autovacuum pode não ser útil para você.

**Avalie esse artigo**

**Autor:**

Josh Berkus

**Mini-curriculo**

Trabalha com banco de dados desde 1994, incluindo Paradox, Rememdy, dBase, Clipper, MSAccess, MSSQL, MySQL e PostgreSQL. Também fui escultor, padeiro, auxiliar de laboratório, captador de recursos, e fui também lider do projeto OpenOffice.org.  
Artigo original em inglês em: <http://www.power-postgresql.com/PerfList>

## Calendário de Eventos

Data	Evento	Site
23, 24 e 25 de Maio de 2005	VI CONINFOR - Convenção de Informática da UNINCOR Univ. Vale do Rio Verde Três Corações/Minas Gerais	<a href="http://www.unincor.br/coninfor">http://www.unincor.br/coninfor</a>
1 a 4 Junho	6º Forum Internacional de Software Livre Local: Porto Alegre - RS - Brasil	<a href="http://www.softwarelivre.org">http://www.softwarelivre.org</a>
21 a 24 de junho de 2005	BLUSOFT BRASIL'2005 FEIRA E CONGRESSO DE TECNOLOGIA Blumenau -SC	<a href="http://www.blusoftbrasil.com.br/port/index.php">http://www.blusoftbrasil.com.br/port/index.php</a>
16 Julho	2º FireBird Developers Day Local: Piracicaba - SP - Brasil	<a href="http://www.FirebirdDevelopersDay.com.br">http://www.FirebirdDevelopersDay.com.br</a>



Se você sabe de algum evento focando em **Banco de Dados** ou em **Software Livre** que não esteja listado aqui, envie-nos um email com os dados do evento para que possamos incluí-lo.

MAGAZINE



fisl6.0