



**Entrevista com Nickolay Samofatov  
Entendendo o gStat - parte 1**



**Acessando o MySQL via ODBC**



**Busca textual com o PostgreSQL**

## Editorial

É com grande satisfação que escrevo o editorial da primeira edição da DB FreeMagazine.

A DB FreeMagazine é sua mais nova fonte de informações sobre Banco de Dados; uma publicação eletrônica que trata com carinho especial os Bancos de Dados Open Source.

A “revista” já nasceu diferente, tanto no seu formato (eletrônico) quanto no seu preço (gratuita).

Nessa primeira edição temos artigos sobre os 3 Bancos de Dados Open Source mais importantes da atualidade: **Firebird**, **PostgreSQL** e **MySQL**.

Também fizemos uma entrevista com Nickolay Samofatov, um dos principais desenvolvedores do Firebird.

Espero que vocês gostem do que vão encontrar por aqui. Contamos com sua ajuda para tornar essa iniciativa um sucesso, ajudando desde já, divulgando o site da revista na comunidade, para os seus amigos, conhecidos, etc.

Caso você tenha interesse em **publicar artigos na DB FreeMagazine**, entre em contato conosco via **email**.

Boa leitura, e bom divertimento!

Carlos H. Cantu  
*Editor DB FreeMagazine*

### ANUNCIE NA DB FreeMagazine

Valorize seu produto ou serviço!

[anuncios@dbfreemagazine.com.br](mailto:anuncios@dbfreemagazine.com.br)

#### Informações

DB FreeMagazine nº 001  
Abril/2005  
Contato geral:  
[webmaster@dbfreemagazine.com.br](mailto:webmaster@dbfreemagazine.com.br)

#### Equipe editorial

Carlos H. Cantu  
([cantu@dbfreemagazine.com.br](mailto:cantu@dbfreemagazine.com.br))

Luiz Paulo de Oliveira Santos  
([lpaulo@dbfreemagazine.com.br](mailto:lpaulo@dbfreemagazine.com.br))

#### Contribuíram nessa edição

Nickolay Samofatov  
Diogo Biazus  
Luiz Paulo de Oliveira Santos  
Carlos H. Cantu

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

#### 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.

## Saindo do Forno...

### Continental Airlines migra para MySQL

A empresa aérea Continental Airlines migrou recentemente seu sistema de emissão de tickets para o MySQL. No processo de migração, foi necessário certificar alguns drivers e ferramentas para rodar em modo 32bits nos sistemas operacionais Linux 64bits da HP, enquanto as versões nativas de 64bits não estavam disponíveis.

Fonte: <http://www.computerworld.com/databasetopics/data/software/story/0,10801,100797,00.html?source=x589>

### Lançado o MySQL 5.0.3

A MySQL AB lançou recentemente a versão 5.0.3 do MySQL, o primeiro beta de uma grande atualização nesse BD que roda em plataformas Linux, Solaris, AIX, Windows e Mac OS X. A versão final está estimada para sair no final do segundo trimestre de 2005.

Fonte: <http://www.computerworld.com/databasetopics/data/software/story/0,10801,100737,00.html?source=x589>

### Escolhendo qual BD usar

O site **About** publicou recentemente um artigo que visa ajudar os leitores na escolha de qual banco de dados se adapta melhor

aos seus requerimentos. O artigo faz algumas comparações entre bancos de dados desktop e relacionais, e cita outros fatores que devem ser levados em consideração na escolha.

Fonte: <http://databases.about.com/b/a/155217.htm>

### SQL Server no Linux?

Aparentemente a Microsoft tem um projeto em andamento para portar o SQL Server para ambiente Linux. Provavelmente ela percebeu que não será fácil derrotar o Pinguim, como fez com outros SOs (ex: OS/2), e não perdeu tempo de criar meios para lucrar com o SO concorrente. O prazo estimado de lançamento é para a metade de 2006.

Fonte: <http://www.sqlservercentral.com/columnists/sjones/sqlserveronlinux.asp>

PS: Calma! Vejam bem no final do artigo: Primeiro de Abril :-)

### PostgreSQL 8.0

Foi lançado a versão 8.0 do PostgreSQL. Essa é a primeira versão que traz o servidor rodando nativamente no Windows, além de muitas outras novidades, como Point-in-time recovery, Savepoints, etc.

Fonte: <http://www.postgresql.org/docs/whatsnew>

### Evento brasileiro terá a presença do criador do InterBase

A segunda versão do **Firebird Developers Day (FDD)**, evento promovido pela **FireBase**, contará com a presença de **Jim Starkey** (criador do InterBase e atual desenvolvedor do Firebird Vulcan) e de sua esposa, **Ann Harrison**, presidente da IBPhoenix e conhecida no passado por “mãe do InterBase”. O FDD será realizado no dia 16 de Julho, em Piracicaba - SP.

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

### Firebird 2.0 Alpha1

Saiu a tão aguardada versão Alpha do Firebird 2.0. Essa versão trás inúmeras novidades, como backups incrementais, execute block, melhoria de performance, novo protocolo de acesso local (xnet), novo sistema de indexação, etc. O release notes pode ser baixado em [http://www.ibphoenix.com/download/Firebird\\_v2Alpha01.ReleaseNotes\\_0200\\_02.pdf](http://www.ibphoenix.com/download/Firebird_v2Alpha01.ReleaseNotes_0200_02.pdf)

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

 A Solução para a Replicação de Dados de sua Empresa contato@object.com.br	▪ O Object Multi-Master Replication Server faz replicação multi-direcional de dados, com tolerância a quedas de rede, ótimo desempenho e log completo de todas as transações replicadas.	www.object.com.br 
	▪ Replica dados entre servidores de um mesmo SGBD ou entre SGBD de diferentes fabricantes.	

# Entrevista com Nikolay Samofatov

Entrevista realizada pela FireBase, em Fevereiro de 2005.

## 1) Fale-nos um pouco sobre você...

Meu nome é Nikolay Samofatov, nasci em Moscow, Rússia, no ano de 1982. Trabalho como engenheiro de software desde os 14 anos, tendo estudado física e matemática na Universidade Estadual de Moscow. Sou graduado em Ciência da Computação pela Universidade Técnica Estadual de Moscow. Eu e minha esposa mudamos para Toronto/Canada há um ano atrás, e agora trabalho para a BroadView Software. Gosto dos prazeres da vida, de "projetos impossíveis" no trabalho, e de praticar esportes radicais quando estou de folga (esquiar, downhill, nadar, pescar, etc).

## 2) Como você se envolveu com o Firebird?

Comecei a trabalhar com o InterBase (IB) em 1997, como programador Delphi. A primeira versão do IB que usei foi a 3.2. O que mais me atraiu nele foi a facilidade de uso e o poder do BD.

Em 2001, após o final do meu contrato com a Altey Company, fui trabalhar para a Bank's Soft Systems, empresa líder no desenvolvi-

mento de softwares para bancos na Rússia.

Meu trabalho era desenvolver softwares de última geração para o governo, com extrema escalabilidade, variando desde estações isoladas até centros financeiros que processavam centenas de documentos por dia. O custo com royalties para instalação em terceiros tinha que ser mínimo. Foi então que decidi usar alguns softwares Open Source para implementar as funcionalidades, adotando a seguinte estrutura:

**Linux**  
**Interbase/Firebird**  
**InterClient/Jaybird**  
**Java 2 SE**  
**OpenORB**  
**OmniORB**  
**Delphi Client**

Quando percebemos que a Borland não tinha interesse em atualizar o IB 6.0, migrar para o Firebird foi um caminho natural para nós. Em algum ponto do processo, encontramos vários bugs no IB/FB e no InterClient, e precisamos corrigi-los. Foi então que decidi dar uma olhada no código do InterBase 6.0/FB e fiquei espantado com o que encontrei: Ao compilar, o software produzia cerca de 20.000 warnings, além de ser de difícil compreensão. A base de código Firebird2, gerada por Mike Nordell, me pareceu muito mais clara, então comecei a corrigir os problemas nela, e compilar o servidor nos meus computadores linux. Muito rapidamente, comecei a produzir

versões que eram mais úteis para nós do que o Firebird 1.0, então movemos todo o desenvolvimento para o Firebird 1.5, antes do seu primeiro Alpha.

Logo após, eu percebi que era muito mais fácil adicionar recursos ao Firebird do que criar soluções complexas no lado cliente, o que dificultava ainda mais o compartilhamento de código da nossa aplicação cliente, que deveria rodar tanto com o Firebird como Oracle. Foi por isso que implementei alguns recursos como o posicionamento de nulls (NULLS FIRST/LAST), savepoints e travamento pessimista. Durante o desenvolvimento da nossa aplicação cliente, eu me encontrei trabalhando no código do Firebird cada vez mais, pois a solução de problemas complexos me trazia a sensação de desafio.

***“Quando percebemos que a Borland não tinha interesse em atualizar o IB 6.0 OpenSource, migrar para o Firebird foi um caminho natural.”***

## 3) Quais foram suas maiores contribuições para o projeto até agora?

Desde o início do meu trabalho dentro do projeto, acabei gastando meu

tempo basicamente nas questões de escalabilidade e estabilidade. Eu gosto de trabalhar na solução de bugs difíceis, ou de problemas difíceis de serem reproduzidos.

No Firebird 1.5, implementei os **savepoints**, corrigi diversos gargalos de performance e implementei o novo **gerenciador de memória escalável**. Para o Firebird 2.0, implementei várias classes na biblioteca de templates, adaptei a engine interna para 64bits de forma a eliminar o limite de 30GB nas tabelas,

e fiz algumas outras alterações internas. Uma relação completa pode ser consultada no re-release notes das versões.

#### 4) Você está trabalhando em algum recurso específico para o FB 2.0 ou 3.0?

Para o FB 2.0, implementei a tecnologia de **backups físicos incrementais** (nbackup), possibilitando gerenciar grandes bases de dados eficientemente. Também criei a API de “trace”, que permite que plugins de terceiros possam coletar informações detalhadas sobre a performance de instruções SQL no servidor.

Nos últimos meses, estive trabalhando com Adriano Fernandes no desenho do novo framework de internacionalização, que fará com que o Firebird suporte completamente o UNICODE 4.0, e corrigirá muitas questões nessa área. Esses dois últimos recursos estão disponíveis em uma

árvore de código separada da base de código oficial do Firebird, e ainda não está claro se eles serão incorporados no FB 2.0 ou 3.0.

Para o Firebird 3.0, tenho algumas idéias, como o suporte à gerenciadores de trava externos (OpenDLM) e técnicas de replicação de cache entre nós de um cluster, para aliviar a pressão no subsistema de I/O, quando for possível. Seria interessante permitir ao Firebird executar SQL em data links ODBC, para simplificar as tarefas de integração e a migração, para os usuários do FB.

Também tenho a idéia de inserir alguns con-

ceitos de orientação à objetos em bancos de dados relacionais. Basicamente permitir o uso de herança, polimorfismo, encapsulamento e abstração nas estruturas do banco de dados. Tenho algumas idéias de como implementar isso sem quebrar os princípios de Codd para bancos relacionais. Isso possibilitaria o mapeamento direto das lógicas de negócio orientadas à objeto no esquema de banco de dados, encurtando o caminho do desenho da aplicação até a implementação, sem comprometer o acesso relacional aos dados.

Na maioria dos casos, eu recebo pelo trabalho que estou fazendo. Por exemplo, os recursos do NBACKUP e da API de trace foram financiados pela Broadview. Portanto, se alguma empresa decide que precisa de algum recurso implementado e pretende pagar por ele, eu posso fazer o trabalho. Até então, eu apenas faço alguns experimentos e pesquisas para ver se as idéias são plausíveis.

*“Na maioria dos casos, eu recebo pelo trabalho que estou fazendo. Por exemplo, os recursos do NBACKUP e da API de trace foram financiados pela Broadview.”*

#### 5) Na sua opinião, quais são os principais recursos que o Firebird ainda não possui, e quão difícil seria implementá-los?

O Firebird, da maneira em que está agora, já é um banco de dados muito utilizado, portanto, alguns recursos que você pudesse sentir falta dependem basicamente de qual área específica o BD estaria sendo utilizado. Se você vai desenvolver aplicações web, poderá sentir falta do recurso de busca textual completa, que pode ser compensado pelo uso

de alguma engine externa, como a Lucene.

Se você trabalhar com aplicações corporativas, poderá sentir falta de selects entre bancos de dados. Se você desenvolve aplicações GIS, poderá sentir falta de tipos de dados espaciais e índices R-tree. Se você trabalha com data warehouses, poderá sentir falta de operações de ROLLUP/CUBO, ou de views materializadas, ou re-escrita de queries.

De qualquer forma, geralmente é possível superar a falta desses recursos usando soluções externas, ou mesmo outros recursos do próprio Firebird. Por exemplo, tabelas temporárias globais podem ser facilmente simuladas usando views atualizáveis e variáveis contextuais CURRENT\_TRANSACTION/CURRENT\_CONNECTION para saber quais registros cada conexão ou transação pode enxergar. É fácil escrever procedures para imitar triggers a fim de manter visões agregadas dos dados para propósitos de warehousing, etc.

E se o custo do “quebra-galho” estiver saindo mais caro do que ter o recurso implementado na engine do banco, você mesmo poderá fazê-lo. Não é tão difícil.

#### 6) Como você vê o Firebird comparado com o PostgreSQL?

O PostgreSQL precisa de um DBA para ser utilizado com eficiência. Você precisa se preocupar com o VACUUM, journals e diversas outras coisas. Os algoritmos de garbage collection do Firebird, protocolo de recuperação (“careful write”) e marcação de páginas geracionais estão integradas em caminhos normais de acesso, e não precisam ser manipuladas separadamente.

Um dos pontos fortes do Firebird é a facilidade de uso. Ele tenta gerenciar o máximo possível da lógica internamente, minimizando o número de opções de “tuning”.

Por exemplo, em índices onde as “folhas” são geralmente únicas, então a estrutura do nó utilizada é um simples B-Tree. Mas se o número de folhas duplicadas aumenta, então o índice automaticamente se transforma em uma forma especial de bitmap, compactando amplamente o número de registros para os nós duplicados. Muitas das operações com índices são executadas na recuperação de bitmaps, e múltiplos índices podem ser utilizados para avaliar uma expressão. Não há necessidade do usuário determinar o tipo de índice ou decidir qual índice deve ser utilizado em determinada query. Outro recurso muito útil do Firebird é que você pode mover os arquivos de bancos de dados para onde quiser, e conectá-los quando quiser.

O Firebird é reconhecidamente muito estável quando utilizado com o Windows, enquanto que uma versão nativa do PostgreSQL para Windows só está aparecendo agora.

Ao mesmo tempo, o PostgreSQL é uma engine poderosa e muito flexível, com muitos recursos para usuários corporativos e cientistas.

**7) Sendo um desenvolvedor da engine do Firebird, como você vê a migração entre o Firebird e o Vulcan, no aspecto técnico? O código do FB servirá de base para a integração com o Vulcan ou será o contrário?**

O plano é enumerar e revisar as mudanças

feitas no Vulcan e no Firebird desde a época da divisão entre eles. Depois disso, precisamos revisar as alterações e combiná-las em uma base de código feita de ambas as versões. Provavelmente nem todas as mudanças feitas no Vulcan irão prevalecer na base de código gerada.

O trabalho de revisão está sendo feito por Dmitry Emanov, atualmente.

**8) Até agora as previsões de datas para as versões finais do Firebird parecem ser sempre sub-estimadas. Você acredita que o projeto ainda não tem experiência suficiente para prever datas com mais precisão?**

Você já viu algum software ser lançado dentro do prazo estimado pelos desenvolvedores? :-) Mas sim, o processo de gerenciamento das releases precisa de melhorias. Preciamos nos atrelar mais em versões baseadas em uma programação de tempo pré-determinada.

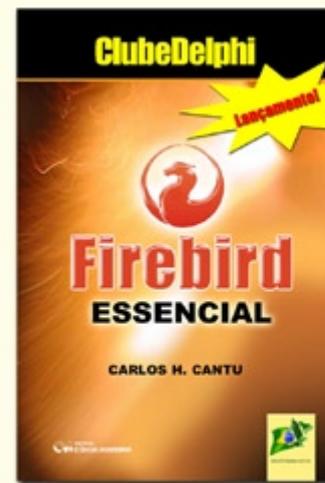
**9) Você foi um dos palestrantes nas duas conferências realizadas em Fulda - Alemanha. Qual é a sua avaliação das conferências?**

Muito boas! Eu gostei de falar lá. É

um bom lugar para encontrar muitos desenvolvedores do Firebird e conversar com vários usuários.

**10) Aparentemente o projeto Firebird não tem um mecanismo oficial de contato direto com os usuários finais, dando a impressão que alguns desejos dos usuários são ignorados pelos desenvolvedores. O que você pensa sobre isso?**

Veja bem, a maioria dos desenvolvedores do Firebird são também usuários. E qualquer usuário do Firebird pode se tornar um desenvolvedor, caso assim deseje. O espírito do Open Source é que se alguém quer que alguma coisa seja feita, ele mesmo pode fazê-la, ou pagar alguém para fazer o serviço. Você também pode criar forks, branches e distribuições com qualquer recurso experimental



## 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/livro/fbessencial](http://www.firebase.com.br/livro/fbessencial)

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

desejado. Tudo depende de você, e se o recurso implementado for bom e estável, provavelmente será incorporado em uma versão oficial do Firebird.

**11) Atualmente o Firebird tem recursos que o colocam à frente do MySQL, mas atrás do Oracle e do MS-SQL. Em quais cenários você acredita que o Firebird se enquadra perfeitamente, e em quais cenários você não o utilizaria?**

Eu não diria que o Firebird fica atrás do MS-SQL e que o MySQL tem menos recursos que o Firebird. O MS-SQL é muito difícil de ser usado. Sua linguagem procedural tem funcionalidade muito limitada. A tentativa da Microsoft de controlar a estrutura da sintaxe da TSQL com opções externas torna quase impossível de se escrever código portátil entre bancos de dados. Tente escrever um trigger para marcar um registro com um valor obtido por um generator do database e você saberá do que eu estou falando.

O MySQL tem vários recursos, mas seu núcleo não é muito poderoso e possui muitos “quirks” e “kludges”

O Oracle é provavelmente o melhor banco de dados corporativo atualmente, mas ele é caro, pesado, e as vezes oferece soluções não muito elegantes para alguns problemas.

O Firebird se enquadra muito bem como um banco amigável para o desenvolvedor, em aplicações corporativas e configuração embarcada. Ele é muito

útil em praticamente qualquer cenário que não envolva uma quantidade monstruosa de dados.

**12) Algum comentário final?**

Eu vejo a comunidade Brasileira de usuários em um papel importante na criação de uma estrutura internacional de suporte e desenvolvimento. Eu gostaria de agradecê-los por usar o Firebird e pelo suporte.

*Nickolay Samofatov*

[Avalie esse artigo](#)



# Entendendo o gStat – parte 1

Carlos Henrique Cantu

Se você alguma vez já teve dúvidas, tais como: “Como saber em qual dialeto está um banco de dados?”, “Qual o tamanho atual das páginas do BD?”, “A escrita síncrona (forced writes) está ativa?”, então é porque desconhece a existência do **gStat!**

O gStat é um utilitário de linha de comando que é instalado junto com o Firebird (na instalação completa) dentro da pasta *bin*, e que retorna diversas informações sobre um determinado banco de dados. As informações retornadas dependem dos parâmetros passados para o utilitário, que podem ser:

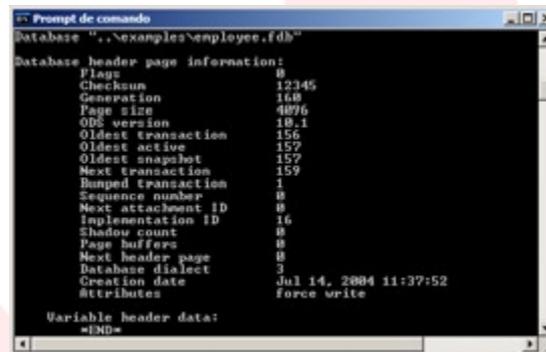
-a	faz a análise dos dados e das páginas de índice
-d	analisa as páginas de dados
<b>-h</b>	<b>analisa a página de cabeçalho do BD (header page)</b>
-l	analisa as “folhas” das páginas de índices
-l	analisa a página de log
-s	analisa os relacionamentos das tabelas de sistema
-u	nome do usuário para conectar no BD
-p	senha para conexão
-r	analisa o tamanho médio dos registros e versões

-t	especifica o nome da tabela a ser analisada
-z	exibe informações da versão do gStat

Analisaremos nesse artigo as opções retornadas pelo parâmetro **-h**, ou seja, as informações do **header** do BD. As outras informações serão analisadas em artigos futuros.

## Cabeçalho (Header)

A figura abaixo mostra o resultado do comando **gstat -h** no BD *employee.fdb*, que acompanha o Firebird 1.5.



Vejamos o que significa cada informação retornada:

Campo	Descrição
<b>Checksum</b>	Esse campo é uma herança do InterBase. No Firebird, ele sempre retornará 12345.

<b>Generation</b>	Esse valor é incrementado cada vez que a página de header é escrita (alterada)
<b>Page size</b>	O tamanho definido para as páginas do banco de dados. Um BD é formado por diversas páginas, de diversos tipos. As páginas têm tamanho fixo, que é determinado no momento da criação do banco de dados ou então quando ele é restaurado de um backup (através do parâmetro <b>-p</b> do <i>gbak</i> )
<b>ODS version</b>	Mostra a versão da estrutura do arquivo do Banco de Dados. Geralmente a adição de novos recursos ao servidor exige alguma alteração na estrutura física do arquivo de banco de dados utilizado por ele. A ODS (On Disk Structure) informa ao servidor qual é a versão da estrutura de um determinado arquivo de banco de dados. A restauração de um backup feito em um servidor de versão mais antiga em um servidor mais recente, automaticamente recria o arquivo de BD usando a versão mais recente da ODS daquele servidor.

<b>Oldest Transaction</b>	Informa qual o número da OIT (Oldest Interesting Transaction), significando a transação mais antiga que não esteja com o status de <i>commit</i> , ou seja, com o status de <i>ativa</i> , <i>limbo</i> ou <i>rolled-back</i> .
<b>Oldest Active</b>	ID da transação mais antiga que ainda está ativa no BD
<b>Oldest Snapshot</b>	ID da transação mais antiga que ainda estava ativa quando a transação identificada pela <i>Oldest Active</i> foi iniciada.
<b>Next Transaction</b>	Mostra o ID da próxima transação a ser criada. Vale uma observação importante: A diferença entre o valor de <i>Next Transaction</i> com o valor de <i>Oldest Transaction</i> determina o momento em que um sweep automático será iniciado. Veja mais informações sobre isso na nota " <i>sweep automático</i> " no final do artigo.
<b>Bumped Transaction</b>	Não é mais utilizado.
<b>Sequence Number</b>	O número seqüencial da página de header, começando por zero (primeira página).
<b>Next attachment ID</b>	ID da próxima conexão com o banco de dados. O ID de uma conexão com o banco pode ser obtido internamente (em <i>procedures</i> , <i>selects</i> , <i>triggers</i> , etc) através da variável <i>current_connection</i> .

<b>Implementation ID</b>	<p>Determina a arquitetura do sistema onde o BD foi criado. Alguns valores:</p> <ul style="list-style-type: none"> <li>• 1 HP Apollo Domain OS</li> <li>• 2 Sun Solaris SPARC, HP9000 s300, Xenix, Motorola IMP UNIX, UnixWare, NCR UNIX, NeXT, Data General DG-UX Intel</li> <li>• 3 Sun Solaris x86</li> <li>• 4 VMS</li> <li>• 5 VAX Ultrix</li> <li>• 6 MIPS Ultrix</li> <li>• 7 HP9000 s700/s800</li> <li>• 8 Novell NetWare</li> <li>• 9 Apple Macintosh 680x0</li> <li>• 10 IBM AIX POWER series, IBM AIX PowerPC</li> <li>• 11 Data General DG-UX 88K</li> <li>• 12 HP MPE/xl</li> <li>• 13 SGI IRIX</li> <li>• 14 Cray</li> <li>• 15 SF/1</li> <li>• 16 Microsoft 32-bit Windows</li> <li>• 17 IBM OS/2</li> <li>• 18 Microsoft Windows 16-bit</li> <li>• 19 Linux Intel</li> <li>• 20 Linux SPARC</li> </ul>
--------------------------	--

<b>Shadow count</b>	Exibe o número de arquivos de <i>shadow</i> desse banco de dados. Arquivos de <i>shadow</i> podem ser criados e refletem a imagem exata do arquivo principal do banco de dados. Geralmente são criados em outro HD, para que no caso de uma falha do HD principal, o <i>shadow</i> possa assumir o lugar do BD principal, evitando que o servidor fique indisponível.
<b>Page buffers</b>	Tamanho do buffer de cache, em número de páginas. Se estiver zero, indica que o valor utilizado é o padrão, definido no arquivo <i>firebird.conf</i> pelo parâmetro <i>DefaultDBCachePages</i> .
<b>Next header page</b>	Caso o BD possua mais de uma página de header, esse valor indica o número da página de header posterior.
<b>Database dialect</b>	Exibe qual é o dialeto atual do Banco de Dados. O Firebird suporta os dialetos 1,2 e 3, sendo que o 1 é para manter a compatibilidade com bancos de dados criados no InterBase 5.x e anteriores; o 2 é utilizado apenas para detectar inconsistências durante a passagem de um BD do dialeto 1 para 3; e o 3 é o dialeto mais rico e mais recente, tendo seu uso recomendado.

<b>Creation date</b>	Data e hora da criação do BD.
<b>Attributes</b>	Exibe alguns atributos do BD: <ul style="list-style-type: none"> <li>•<b>force write</b>: indica que o BD está configurado para o modo síncrono de escrita de dados, ou seja, o Firebird não irá manter os dados gravados no buffer de memória, enviando-os diretamente para o disco.</li> <li>•<b>no_reserve</b>: indica que nenhum espaço está sendo reservado para os registros de versão (ver nota "reservando espaço").</li> <li>•<b>Shutdown</b>: Indica que o BD está em modo shutdown, impossibilitando conexões de usuários que não sejam o SYS-DBA.</li> </ul>

Se a diferença entre Oldest active e Next transaction for muito grande, pode indicar que as transações estão ficando abertas por muito tempo, o que sugere uma falha no controle transacional da aplicação cliente que faz acesso ao BD.

Se a diferença entre Next Transaction e Oldest Transaction for maior que 20.000, pode indicar que o sweep automático está desligado, ou configurado com um valor demasiadamente alto. No primeiro caso, você deverá realizar periodicamente um sweep manual no BD através do utilitário gfix. Para o segundo caso, pode-se usar o gfix para configurar um novo valor para o sweep automático.

Ambas as situações descritas acima podem fazer com que a performance do BD caia bastante em um ambiente de muita concorrência.

### Conclusão

Vimos nesse artigo como recuperar informações sobre um BD, determinando assim suas características gerais como dialeto, tamanho da página, etc.

Aprendemos também que o gstat pode nos ajudar a detectar problemas de concorrência no BD, o que geralmente leva a perda de performance no servidor.

Em um próximo artigo, estaremos anali-

### Detectando possíveis problemas

Um dos usos mais importantes do gStat é para detectar possíveis problemas no gerenciamento transacional, que podem fazer com que o BD fique lento em determinadas situações de grande concorrência, etc.

Para saber se as aplicações clientes estão trabalhando adequadamente com o controle transacional, devemos monitorar, com o BD em uso, os valores:

- Oldest transaction
- Oldest active
- Next transaction

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

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

Cada vez mais empresas desenvolvem sistemas multi-usuários que necessitam que seus dados sejam disponibilizados na Internet, em tempo real, para atender às necessidades de seus clientes. Oferecemos as melhores soluções em hospedagens de bancos de dados MySQL e Firebird (1.0 e 1.5), com acesso direto e sem restrições de conexão.

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/!!	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/!!	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/!!

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

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

sando as outras opções do gstat.  
Um abraço, e até mais!

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.



## Reservando espaço

O Firebird trabalha com o sistema de concorrência chamado **Versioning**, ou MGA (Multi Generational Architecture). Esse modo faz uso de versões temporárias de registros, permitindo que uma transação enxergue os valores de um registro como eles eram em um determinado momento no tempo. Para agilizar o controle de concorrência, o Firebird “**reserva**” automaticamente **20%** do espaço de uma página de dados para trabalhar com essas versões de registro. No entanto, há situações onde o BD será somente para leitura (catálogos em CD, por exemplo) o que torna um desperdício a reserva desse espaço. Podemos indicar durante a restauração de um backup, através do parâmetro `-USE_(ALL_SPACE)` do `gbak`, que não desejamos reservar espaço para os registros temporários, fazendo com que o tamanho do BD fique reduzido.

## Sweep automático

O **sweep** é um processo de limpeza do banco de dados. Através dele, o Firebird libera espaços que não serão mais utilizados para que possam ser reaproveitados no servidor. Diferente do processo automático de **Garbage Collection**, o sweep processa também os registros que foram descartados devido a um rollback de uma transação. O valor padrão para iniciar o sweep automático é 20.000, ou seja, quando a diferença entre o ID da próxima transação e a OIT for igual a 20.000, o sweep será disparado. Obviamente, após o término do sweep, o número da OIT será avançado.

## Acessando MySQL via ODBC

Luiz Paulo de Oliveira Santos

Com a utilização em massa dos bancos de dados, nasceu também a necessidade de acesso as informações armazenadas nessas bases. Nas empresas, é comum a necessidade de gerar arquivos texto para intercâmbio de informações entre aplicações, geração de malas diretas, emissão de etiquetas, etc.

Praticamente todos os setores das empresas necessitam de informações que estão armazenadas em bancos de dados centralizados, sendo função do Departamento de Tecnologia em Informática (ou equivalente) fornecer esses dados, ou o acesso aos mesmos.

A geração de arquivos texto para exportação dos dados é geralmente uma atividade muito custosa e desgastante para o desenvolvedor. Há também o fato de que as informações geradas nos arquivos geralmente ficam desatualizadas em questão de horas após a geração do arquivo.

A necessidade de ter acesso padronizado, fácil e rápido às informações em bases de dados provocou o surgimento de um padrão de drivers, comumente chamados de ODBC (Opened DataBase Connectivity).

Geralmente os principais bancos de dados disponibilizam gratuitamente se os drivers ODBC. Nesse artigo, estaremos citando um passo-a-passo de como instalar e configurar um driver ODBC para acessar uma base de dados MySQL.

O MySQL é gratuito quando utilizado em

aplicações sob a licença GPL. No site do próprio MySQL podemos encontrar um driver ODBC para vários sistemas operacionais (visite o site: <http://dev.mysql.com/downloads/>), sendo que nesse exemplo estaremos utilizando o Windows como cliente.

Acessando o site, clique na opção descrita como "Connector/ODBC 3.51 -- Generally Available (GA) release" e em seguida clique em "Windows Downloads".

Para instalação em ambiente Windows, duas opções de instalação estão disponíveis:

- 1) Através de binary (.EXE) e instalação manual;
- 2) Através de distribuição via MicroSoft Installer (MSI);

Escolha a opção para fazer a instalação via .EXE. Clique na opção **Pick a mirror**.

Teremos a seguinte tela:



Se você é usuário de banda larga, opte pelo protocolo HTTP, pois desta forma o download será mais veloz. Salve o arquivo no HD, para ser instalado depois.

**Obs:** O arquivo que efetuamos download é

o *MyODBC-3.51.11-1-win.exe*. Periodicamente, novos downloads são disponibilizados, e o arquivo pode ter seu nome alterado de acordo com essas novas versões.

Quando iniciarmos a instalação do driver de ODBC, executando o arquivo baixado, teremos a seguinte tela:



Clique em Next, e você verá a tela de licença. Nela, os desenvolvedores do driver ODBC para MySQL citam que o uso desse driver segue as determinações GNU de licenciamento. Se quiser maiores informações a respeito, visite o link [www.gnu.org/philosophy](http://www.gnu.org/philosophy). Em resumo, o software é free (gratuito), desde que mudanças não sejam feitas nele.

Após selecionar o item "I accept the license agreement", clique no botão Next. A tela apresentada agora resume o que a instalação faz, e alerta para que antigas versões de drivers ODBC para MySQL sejam removidos antes de instalar essa nova versão.

Clique novamente em Next. Após clicar no próximo botão Next, a instalação continuará de forma ininterrupta até o fim. Portanto, aten-

ção nesse passo.

Se tudo correu bem, a tela final de instalação será exibida:

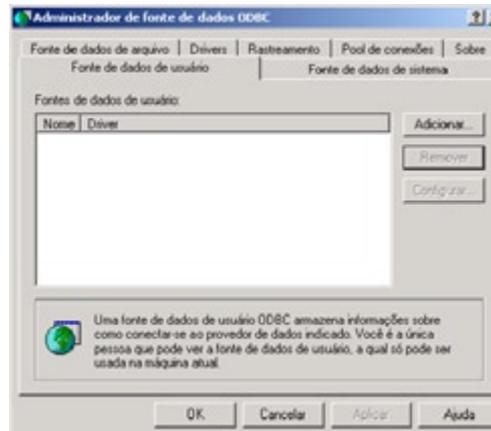


Pronto! A instalação está concluída. A partir de agora, iremos efetuar as configurações do objeto de dados para disponibilizarmos o acesso aos aplicativos.

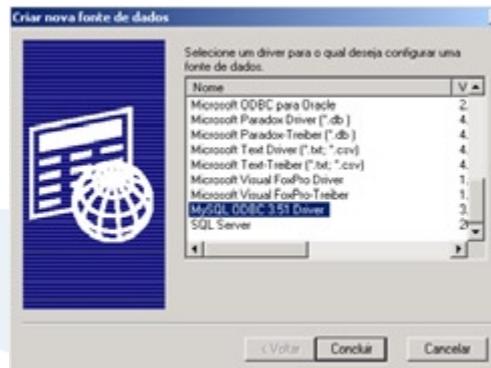
É necessário criar um objeto Fonte de Dados. Isso é feito no Fonte de dados ODBC, que se encontra dentro do ítem "Ferramentas administrativas" no Painel de Controle do MS-Windows XP, indicado pelo ícone abaixo:



Ao acessar o ícone, a seguinte tela será apresentada:



Para criarmos esse objeto, basta clicar em *Adicionar* e teremos a seguinte tela:

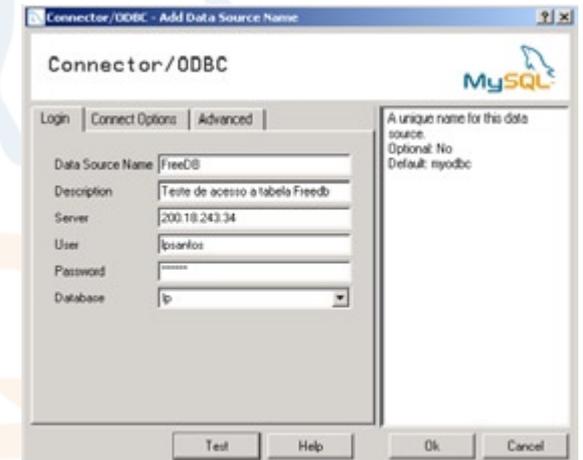


Nessa tela podemos verificar todos os drivers de ODBC disponíveis em seu computador. Usaremos a opção *MySQL ODBC 3.51 Driver*, que é o driver ODBC para o MySQL que instalamos.

**Obs:** Através do ODBC é possível criarmos objetos de dados para acessar Arquivos Texto, bases DBF (X-Base), bases MySQL e

até mesmo bases Oracle. Por exemplo, podemos criar uma aplicação (em Delphi, VB ou outra linguagem qualquer que utilize ODBC) que abra uma tabela que está em uma base MySQL, ligando-a dinamicamente com dados que estão em um Arquivo Texto e salve o resultado em uma tabela ORACLE. Sendo o acesso a todas as bases citadas através do ODBC. Essa facilidade de intercâmbio de informações entre diferentes bancos e bases de dados torna o acesso via ODBC bastante atraente.

Clicando no botão Concluir, veremos a seguinte tela:



Preencha os campos da aba LOGIN da seguinte maneira:

**DATA SOURCE NAME:** Nome do objeto de dados. Esse é o nome pelo qual os aplicativos irão enxergar o banco de dados. No Delphi, esse nome aparecerá em DATABASENAME nos objetos de acesso de

dados, como o TTable por exemplo.

**DESCRIPTION:** É uma descrição do objeto. Geralmente alguns aplicativos exibem essa descrição além do DATA SOURCE NAME, para que possamos escolher o banco correto.

**SERVER:** O número IP ou o nome do equipamento (nome registrado em DNS) que está executando o MySQL Server. Cuidado com Proxies e Firewalls que podem impedir o acesso ao banco de dados. Pode ser um IP ou nome de um servidor fora de sua rede. Vários provedores de hospedagem disponibilizam base MySQL.

**USER:** Nome do usuário do banco (não do equipamento ou rede). O administrador do banco é quem cria os usuários. Em um próximo artigo estaremos descrevendo como administrar o MySQL.

**SENHA:** Senha do USER do banco.

**DATABASE:** É o banco do MySQL que desejamos gerenciar. Não confundir banco com tabela. O MySQL é capaz de gerenciar vários bancos simultaneamente. É nessa opção devemos informar qual banco desejamos acessar. Note que os bancos disponíveis nesse servidor aparecerão em um combo. Obs: Para cada banco de dados devemos criar um DATA SOURCE NAME.

A figura anterior mostra os campos preenchidos com alguns dados de exemplo.

Com os campos já preenchidos, podemos efetuar um teste de conexão. Para isso, basta clicar no botão *Test*, e se tudo der certo, uma janela aparecerá com um aviso de sucesso. Nesse teste, todos os dados de conexão previamente informados são utilizados. Caso alguma mensagem de erro apareça, as

configurações informadas devem ser revistas. Após isso, basta clicar no botão OK e o objeto aparecerá na janela do *Administrador de Fonte Dados*.

Uma vez configurado, podemos usar essa fonte de dados em diversos aplicativos Windows, em nosso exemplo a fonte de dados FreeDB do tipo ODBC, bastando escolher depois qual é a tabela e campos que se deseja utilizar.

Há também uma aba **Connect Options** e outra **Advanced**. Pela aba *Connect Options* pode-se configurar a porta TCP/IP utilizada para acesso ao banco. Em algumas redes, o administrador de banco de dados necessita trocar a porta padrão do MySQL, logo as estações devem ter essa informação, do contrário não conseguirão acessar o servidor MySQL.

Na aba *Advanced*, podemos configurar como os dados recebidos serão exibidos ao usuário e aplicativos, bem como informações de debug como, por exemplo, gravar as queries geradas na estação em arquivos texto.

Discutiremos melhor essas duas abas em um artigo futuro de Tuning em Clientes MySQL.

## Exemplo prático

Um exemplo bastante comum é disponibilizar o acesso a um banco ou tabela, para funcionários(as) que necessitem efetuar buscas ou alterações em dados. Quando implementamos aplicações baseadas em banco de dados cliente/servidor, grande parte dos problemas aparece no lado cliente, pois geralmente tem-se uma aplicação que faz toda

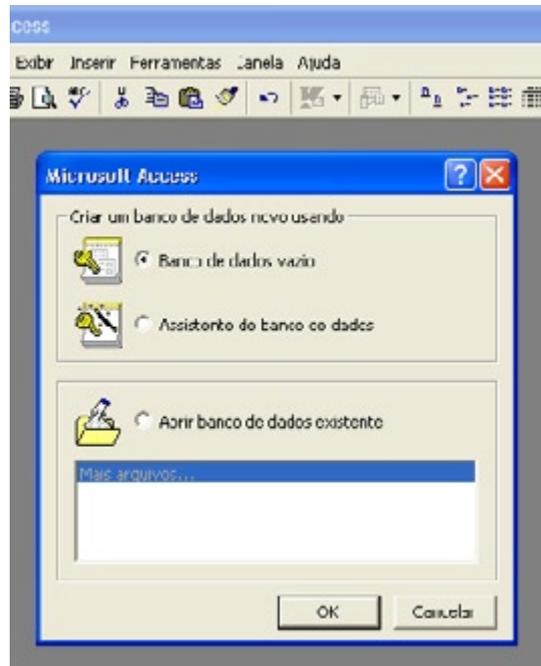
manutenção, buscas/relatórios e acesso da base de dados. Provavelmente o usuário já está acostumado com outra ferramenta para acesso e navegação em bases de dados genéricas. Muitos dos usuários são familiarizados com o MS-Access ou outros sistemas gerenciadores, e por isso são muito produtivos com esses aplicativos. Logo, o ODBC serve como ponte para que esses usuários especializados possam acessar as informações armazenadas no banco através de uma aplicação cliente de sua escolha.

**Nota Importante:** *Os direitos do usuário vão depender unicamente dos direitos atribuídos ao usuário do banco MySQL. No exemplo citado anteriormente, temos um user Ipsantos. Cabe ao administrador da rede ou do banco de dados MySQL atribuir direitos de consulta e atualização, e definir em quais tabelas o usuário pode trabalhar. Em caso de ausência de um user criado, o user padrão é root (geralmente bloqueado por razões de segurança) e, se nenhuma senha foi atribuída pelo administrador, a mesma permanece em branco. Porém, muito cuidado! Abrir a base de dados para GRAVAÇÃO e ALTERAÇÃO dessa maneira pode permitir ao usuário provocar uma bagunça generalizada, ou mesmo danificar a integridade lógica dos dados. Sugestão: Uso de ODBC em Cliente/Servidor apenas para consulta!*

Em nosso exemplo, vamos mostrar um objeto de dados ODBC devidamente configurado e testado de acordo com o exemplo anterior. O nome do objeto de dados é FreeDB. Supondo que desejamos efetuar uma con-

sulta a uma tabela de dados do FreeDB que está no MySQL, utilizando como cliente o MS-Access. A partir dele, qualquer usuário pode gerar desde simples consultas, até poderosos mecanismos de busca e publicação. O MS-Access pode gerar relatórios e listagens, imprimir etiquetas e gerar mala direta.

Ao Carregar o MS-Access, devemos criar um novo banco de dados. Caso o usuário deseje anexar a tabela do MySQL a um banco (.MDB) já existente, é perfeitamente possível, mas sugerimos gerar uma cópia de segurança de seu banco antes de efetuar a operação. A tela que teremos é a seguinte:



Uma vez selecionado o botão OK, teremos

uma tela na qual devemos indicar qual é o arquivo .MDB a ser criado. É nesse arquivo que criaremos o elo entre o usuário e o MySQL. O Nome do banco .MDB será MyAccess.MDB. Logo após informarmos o nome do .MDB, teremos a seguinte tela:



Nesse momento, devemos acessar o Menu e selecionar a Opção Inserir, indicando a opção Tabela, conforme tela abaixo:



Após isso, devemos selecionar a opção de vinculação de tabela ao banco .MDB, ou seja, o MS-Access criará um link lógico com os da-

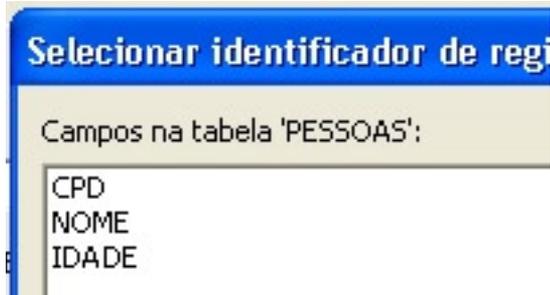
dos no MySQL. No banco .MDB não ficarão armazenados os dados, somente um elo com a tabela armazenada fisicamente no MySQL. Na tela apresentada que foi aberta, selecione "Vinculação de tabela" e clique OK. Agora selecione como na opção "Arquivos do tipo" a opção Banco de dados ODBC, conforme tela abaixo:



Na tela seguinte, mude para a aba Fonte de dados de máquina e selecione FreeDB, que é o objeto de dados que acabamos de criar para acessar o MySQL. Ao clicar no botão OK, as tabelas contidas no banco e disponíveis ao usuário serão listadas, como mostra figura abaixo.



Após selecionar a tabela (no exemplo é a tabela PESSOAS, mas pode ser qualquer tabela que desejarem vincular ao MS-Access) e clicarmos no botão OK, teremos que responder qual é a chave primária da tabela:



No exemplo, selecionaremos o campo NOME e clicaremos no botão. A seguinte tela será exibida:



A partir de agora, podemos utilizar a tabela pessoa como se ela fosse uma tabela do MS-Access. Após efetuarmos um duplo clique na tabela PESSOAS, teremos um browse da tabela, proporcionado pelo MS-Access, em uma tabela MySQL. A tela a seguir demonstra isso:

PESSOAS : Tabela	
CPD	NOME
0001	LUIZ PAULO DE OLIVEIRA S
0002	LARA RITA DE MORAES
0003	JOSÉ DA SILVA
0004	MARIA CAMÉLIA
*	

Nela podemos navegar pelos dados, e de acordo com os direitos do usuário determinados no banco MySQL, alterar, inserir e excluir informações. A partir daí, a tabela está disponível para qualquer atividade no MS-Access.

Pode-se utilizar as informações de acordo com a necessidade de cada um.

Espero ter sido útil nesse artigo e gostaria de receber um retorno de vocês com dicas, sugestões e críticas construtivas. Um abraço e até o próximo artigo.

### Avalie esse artigo

#### 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

Vox on Demand

## JobVox

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

# Indexação textual no PostgreSQL

Diogo de Oliveira Biazus

As vezes é necessário criar aplicações que buscam dados de acordo com palavras contidas em textos extensos. No entanto, a procura de padrões em grandes cadeias de caracteres (strings) é extremamente custosa em termos computacionais/performance. Uma das soluções para esse problema é quebrar os textos em átomos maiores que um caractere, ou seja, utilizar **palavras-chave** para indexar. O uso de palavras-chave reduz consideravelmente o número de elementos distintos à serem indexados, além de possibilitar algoritmos muito mais eficientes para a busca. Torna-se possível utilizar uma estrutura de dados que organize todas as palavras contidas em um texto em um formato que possa ser facilmente percorrido: uma árvore. Outra vantagem na utilização de palavras-chave é que elas possuem um **significado**, o que não acontece com meras cadeias de caracteres. Isso pode ser utilizado para criar ordenações baseadas em relevâncias de palavras em um determinado contexto.

Um ótimo exemplo de pesquisa textual rápida são os sites de busca na web (google, Yahoo, etc), que procuram por palavras-chave presentes em documentos armazenados em imensas bases de dados.

## Tsearch2

O Tsearch2 (<http://www.sai.msu.su/~megera/postgres/gist/tsearch/V2/>) é a segunda geração das ferramentas de indexação de texto para o PostgreSQL. Ele é composto por uma série de funções, operadores e tipos adicionais que podem ser instalados em uma base de dados a partir do módulo presente no diretório *contrib*. Existem várias ferramentas livres para esse mesmo fim disponíveis na internet, mas nem todas são integradas à um SGBD relacional.

O Tsearch2 parte do princípio que podemos dividir um texto qualquer em unidades atômicas (palavras-chave), e dispor essas unidades em vetores que serão indexados em árvores de busca genéricas (GIST - gist.cs.berkeley.edu). O Tsearch2 é desenvolvido por Oleg Bartunov e Teodor Sigaeov, e faz parte do pacote oficial do PostgreSQL desde a versão 7.4, podendo ser instalado também na 7.3.

## Instalação

Para instalar o tsearch2 em uma base de dados, o primeiro passo é obter a biblioteca dinâmica com as funções necessárias. Caso você esteja utilizando pacotes pré-compilados em qualquer distribuição de Linux, provavelmente existe um pacote separado para os itens do *contrib* que necessita ser instalado. No Debian, por exemplo, o nome do pacote é *postgresql-contrib*. Se servidor for o PostgreSQL 8.0 para Windows, o tsearch2 já está disponível no diretório: *INSTALACAO\8.0\lib*. Ainda existe a possibilidade de que o seu ser-

vidor tenha sido instalado à partir dos fontes. Nesse caso, você terá que compilar o pacote do tsearch2: entre no diretório dos fontes e depois em *contrib\tsearch2*, e digite **make install**.

Estando instalada corretamente a biblioteca dinâmica, basta encontrar o arquivo *tsearch2.sql*, também presente no diretório *contrib*, e executar os comandos contidos nele na base de dados destino.

## Configuração

Depois de instalado, o tsearch2 cria quatro tabelas de configuração no banco de dados. Elas são: *pg\_ts\_cfg*, *pg\_ts\_cfgmap*, *pg\_ts\_dict* e *pg\_ts\_parser*. A configuração geralmente só é feita nas três primeiras tabelas, sendo que a última serve para configurar interpretadores, algo que dificilmente faremos, pois o interpretador padrão é suficiente para a maioria dos casos.

Na tabela *pg\_ts\_cfg* encontram-se as configurações do indexador. Normalmente usaremos apenas uma configuração por idioma. No entanto, é comum usarmos mais de uma configuração para um mesmo idioma, gerando tipos de indexação distintas. Poderia ser interessante, por exemplo, criar duas configurações para o português: uma utilizando um algoritmo de redução de palavras ao radical, e outra indexando as palavras exatamente como foram encontradas no texto, e talvez um terceiro método, usando um dicionário *ispell*. Na próxima seção explicarei melhor o sistema de dicionários do tsearch2.

A estrutura da *pg\_ts\_cfg* está descrita abaixo:

```
create table pg_ts_cfg (
  id      int not null primary key,
  ts_name text not null,
  prs_name text not null,
  locale  text );
```

Onde *id* e *ts\_name* são identificadores da configuração, *prs\_name* é o interpretador a ser usado, e *locale* indica qual configuração de localização do sistema define automaticamente essa configuração para a indexação.

A tabela *pg\_ts\_cfgmap* mapeia os tipos de palavras encontradas para os dicionários que serão responsáveis pela indexação. Cada registro na *pg\_ts\_cfgmap* pertence à uma configuração na *pg\_ts\_cfg*. Por exemplo, se você tem uma configuração chamada “português” cadastrada na *pg\_ts\_cfg*, ela deve estar mapeada na *pg\_ts\_cfgmap*, para que o PostgreSQL saiba como indexar as palavras quando ela for utilizada.

A estrutura da tabela *pg\_ts\_cfgmap* é:

```
create table pg_ts_cfgmap (
  ts_name      text not null,
  tok_alias    text not null,
  dict_name    text[],
  primary key (ts_name,tok_alias));
```

Onde: *ts\_name* é o nome da configuração pai, *tok\_alias* é o nome dado ao tipo de palavra, como *lword* para “latin word” ou *email* para endereços de email. *Dict\_name* é a lista de dicionários que podem ser usados para indexar esse tipo de palavra. Note que *dict\_name* é um arranjo, e que os seus elementos devem estar em ordem de prioridade. O *tsearch* vai tentar usar todos os dicionários cadastrados

na ordem de inserção.

Finalmente, a tabela *pg\_ts\_dict* é onde se encontram os dicionários, sendo uma das grandes vantagens do *tsearch2*, pois permite uma flexibilidade na utilização de vários algoritmos para indexar os textos. A estrutura da tabela é a seguinte:

```
CREATE TABLE pg_ts_dict (
  dict_name text not null,
  dict_init oid,
  dict_initoption text,
  dict_lexize oid not null,
  dict_comment text);
```

Onde: *dict\_name* é o nome do dicionário, *dict\_init* é o *oid* de uma função do banco de dados para iniciar o dicionário, *dict\_initoption* são parâmetros passados para essa função, *dict\_lexize* é o *oid* de uma função que vai retornar a unidade a ser indexada à partir das palavras encontradas, e o *dict\_comment* é apenas um comentário sobre o dicionário.

## Dicionários customizados

Podemos usar vários tipos de dicionários para controlar o processo de indexação de palavras-chave. O termo “dicionário” aqui não está sendo usado no seu sentido convencional, pois ele pode ser apenas um algoritmo, ao invés de uma listagem estática de palavras.

Qual o motivo para usar tal recurso? Em primeiro lugar: **performance**. A velocidade de busca nos índices é determinada pela quantidade de palavras-chave distintas presentes em toda a árvore, quanto menor for esse número, maior será a performance das pesquisas. Os dicionários diminuem o número de

elementos diferentes uniformizando as palavras, removendo sufixos, e em alguns casos, eliminando palavras irrelevantes.

Em segundo lugar, para melhorar a usabilidade de um sistema - muitas vezes é interessante para o usuário obter no retorno de suas consultas documentos com palavras semelhantes, mas não exatamente iguais àquelas utilizadas por ele.

Existem basicamente dois tipos de dicionários para cada idioma, um baseado no **ispell** (<http://www.gnu.org/software/ispell/ispell.html>), e outro que usa um algoritmo de **stemm** (redução de palavras ao radical). O dicionário **ispell** usa uma lista de palavras pré-catalogadas em um formato especial para remover os sufixos. Ele funciona de forma extremamente precisa, mas indexa somente palavras encontradas na listagem (o que eventualmente pode ser indesejável). Podemos encontrar arquivos com palavras do português brasileiro no formato **ispell** graças à um projeto hospedado na USP (<http://www.ime.usp.br/~ueda/br.ispell/>).

Já o **stemm** é um procedimento que tenta reduzir as palavras através de regras gerais, o que pode gerar várias imprecisões em termos lingüísticos, mas funciona de forma satisfatória, sendo mais flexível, principalmente em textos cheios de “estrangeirismos” ou neologismos.

Para configurar um dicionário de **stemm** em português, por exemplo, devemos seguir os seguintes passos, no linux:

```
cd PGSQL_SRC/contrib/tsearch2/gendict
```

Depois baixe os arquivos `stem.{c,h}` para o português:

```
wget http://snowball.tartarus.org/portuguese/stem.c
wget http://snowball.tartarus.org/portuguese/stem.h
```

Crie arquivos de modelo para português:

```
./config.sh -n pt -s -p portuguese -v
-C'Snowball stemmer para português'
```

Vários arquivos vão ser criados em *PGSQL\_SRC/contrib/dict\_pt*.

Compile e instale o dicionário:

```
cd PGSQL_SRC/contrib/dict_pt
make install
psql base_de_dados < dict_pt.sql
```

Também existem dicionários específicos para tratamento de números, pois eles são identificados de forma diferente das palavras, e podem ter um tratamento especial. Por exemplo, podemos eliminar números muito grandes usando um dicionário para inteiros. Para mais informações, consulte: <http://www.sai.msu.su/~megera/postgres/gist/tsearch/V2/dicts/README.intdict> (em inglês).

Podemos configurar um arquivo de palavras irrelevantes a serem descartadas durante a indexação. Esse arquivo deve conter uma lista de palavras (uma palavra por linha), e estar em um diretório acessível pelo usuário *postgres*. Então, podemos colocar na tabela de configuração *pg\_ts\_dict* o parâmetro de inicialização de dicionário *dict\_initoption*.

Por exemplo, se temos um arquivo de palavras irrelevantes em */var/lib/postgres/data/portugues.txt* e desejamos usar esse arquivo

em um dicionário de *stemm*, como por exemplo o que foi criado no exemplo acima, usamos um SQL como o seguinte:

```
INSERT INTO pg_ts_dict SELECT
'pt_stem',
(select oid
 from pg_proc
 where proname='snb_pt_init'),
'/var/lib/postgres/data/portugues.txt',
(select oid
 from pg_proc
 where proname='snb_lexize'),
'Stemmer Snowball de Português.';
```

Para usar o dicionário acima, precisamos de registros correspondentes nas tabelas *pg\_ts\_cfg* e *pg\_ts\_cfgmap*. Vou exemplificar a criação de uma configuração que utilize o dicionário que acabamos de inserir na base:

```
INSERT INTO pg_ts_cfg VALUES ('default_portuguese', 'default', 'pt_BR.ISO8859-1');
```

No comando acima, criamos uma configuração chamada *default\_portuguese* que usa o interpretador (parser) padrão do *tsearch*, e é acionada automaticamente quando utilizamos o **locale** *pt\_BR.ISO8859-1*. Agora, vamos mapear os tipos de palavras para os dicionários na nossa configuração. Para isso, usarei o comando *copy* para inserir os registros adequados em *pg\_ts\_cfgmap*:

```
COPY pg_ts_cfgmap FROM stdin;
default_portuguese lword {pt_stem}
default_portuguese nlword {simple}
default_portuguese word {simple}
```

```
default_portuguese email {simple}
default_portuguese url {simple}
default_portuguese host {simple}
default_portuguese sfloat {simple}
default_portuguese version {simple}
default_portuguese part_hword {simple}
default_portuguese nlpart_hword {simple}
default_portuguese lpart_hword {simple}
default_portuguese hword {simple}
default_portuguese lword {pt_stem}
default_portuguese nlhword {simple}
default_portuguese uri {simple}
default_portuguese file {simple}
default_portuguese float {simple}
default_portuguese int {simple}
default_portuguese uint {simple}
\.
```

No exemplo acima, escolhemos o dicionário *pt\_stem* para palavras do tipo *lword* (Latin Word), *lhword* (Latin Hyphenated Word). O “h” antes da palavra *word*, indica que a palavra contém hífen, o “l” é para palavras latinas e o “nl” para palavras não latinas. Existem tipos de termos específicos, como **email** (endereços de email), **url** (endereços URL), **int** (inteiros), **float** (números em ponto flutuante), etc...

## Criando os índices

Primeiro é necessário saber que os índices de palavra-chave não serão criados diretamente sobre o campo com o conteúdo dos textos. É necessário criar um campo com uma estrutura própria para indexação, o tipo desse campo é **tsvector**, podendo ser facilmente gerado a partir de campos texto com a função

**to\_tsvector.** Para adicionar um campo desse tipo basta usar o comando:

```
ALTER tabela ADD campo_indice tsvector;
```

Caso você já tenha linhas na tabela de documentos, use um *update* com a função *ts\_vector* para criar os valores iniciais do *campo\_indice*:

```
UPDATE tabela SET campo_indice = to_tsvector(campo_texto);
```

Em seguida, é necessário manter sempre o *campo\_indice* sincronizado com as suas fontes, no nosso exemplo o *campo\_texto*. Para isso, convém criarmos um trigger que atribua e mantenha atualizado o *campo\_indice* usando *to\_tsvector(campo\_texto)*. Já existe uma função pronta para isso, basta criarmos então o trigger na tabela desejada com o seguinte comando:

```
CREATE TRIGGER tsvector_update  
BEFORE UPDATE OR INSERT  
ON tabela  
FOR EACH ROW EXECUTE PROCEDURE  
tsearch2(campo_indice, campo_texto);
```

Agora você tem um campo do tipo *tsvector* sendo atualizado e pronto para ser indexado. Basta criar o índice *gist* sobre o *campo\_indice*:

```
CREATE INDEX nome ON tabela USING  
gist (campo_indice);
```

O índice *gist* é um dos tipos de índice suportados pelo PostgreSQL. É uma espécie de *B-Tree* mais flexível, projetado para aceitar vários tipos de dados, inclusive tipos compostos. Para mais informações, visitem o site sobre indexação GIST: <http://gist.cs.berkeley.edu/>

Para criar mais de um índice na mesma tabela usando configurações de indexação diferentes, precisaremos criar um campo de índice para cada configuração. Por exemplo, digamos que temos uma base com duas configurações na *pg\_ts\_cfg*: *default\_portuguese* para indexar palavras em português com *stemm*, e *simple* para indexar palavras exatamente como foram encontradas.

Para isso criáramos um segundo campo de indexação:

```
ALTER tabela ADD campo_indice_stemm  
tsvector;
```

E indicáramos nas funções de atualização, qual função estamos usando para cada campo:

```
UPDATE tabela SET campo_indice = to_tsvector('simple', campo_texto);
```

```
UPDATE tabela SET campo_indice_stem = to_tsvector('default_portuguese',
```

```
CREATE FUNCTION tsearch2_estendida() RETURNS TRIGGER AS `  
    NEW.campo_indice := to_tsvector('simple', NEW.campo_texto);  
    NEW.campo_indice_stem := to_tsvector('default_portuguese', NEW.  
campo_texto);  
    RETURN NEW;  
` LANGUAGE 'plpgsql';
```

**Listagem 1.** Versão estendida da função

```
campo_texto);
```

Nosso trigger teria que ser personalizado, para isso basta criar uma função simples em linguagem procedural. Se a **PL/pgSQL** estiver instalada no banco, basta criar uma função como a mostrada na **listagem 1**.

E adaptarmos o nosso trigger com a nova função:

```
CREATE TRIGGER tsvector_update  
BEFORE UPDATE OR INSERT  
ON tabela  
FOR EACH ROW EXECUTE PROCEDURE  
tsearch2_estendida();
```

## Realizando consultas

Tendo o campo *tsvector* e o seu respectivo índice você já pode iniciar as buscas. Porém, para fazer consultas é necessário utilizar mais um tipo especial: o *tsquery*, pois o operador que usaremos para fazer as consultas é o *@@*, que opera um tipo *tsquery* com um *tsvector*. É muito simples gerar um tipo *tsquery* à partir de um texto, bastando usar a função *to\_tsquery*, como em:

```
SELECT titulo, corpo FROM docs WHERE  
campo_indice @@ to_tsquery('brasil');
```

No exemplo acima, usamos apenas uma palavra-chave para a busca (brasil), mas poderíamos ter usado várias palavras concatenadas com operadores lógicos & (para e), | (para ou), ou ! (para negação). Por exemplo:

```
SELECT titulo, corpo FROM docs
WHERE campo_indice @@ to_tsquery('(brasil|brasileiro)&presidente');
```

A consulta acima procura qualquer documento onde existam as palavras (presidente e brasil) ou (presidente e brasileiro).

Notem que os exemplos dados até agora utilizam apenas um parâmetro para as funções de conversão (*to\_tsquery* e *to\_tsvector*). Porém, essa função possui uma versão sobrecarregada (*overloaded*) para usar dois parâmetros, onde o primeiro é a configuração de dicionário e o segundo é o objeto a ser convertido. A configuração são aquelas encontradas na tabela *pg\_ts\_cfg*. Assim, em uma aplicação que usa diferentes tipos de configuração para indexar os textos, poderíamos usar os seguintes comandos:

```
SELECT titulo, corpo
FROM docs
WHERE campo_indice @@ to_tsquery
('default_portuguese', 'brasil');
```

```
SELECT titulo, corpo
FROM docs
WHERE campo_indice @@ to_tsquery
('simple', 'brasil');
```

Se tudo for feito corretamente, a consulta usando o operador @@ vai utilizar o índice gist criado sobre o *campo\_indice*, e a velocidade

de busca vai ser bem superior ao de uma consulta similar usando operadores *LIKE* ou de expressões regulares.

O resultado pode ser observado usando o comando **explain**, que mostra o plano de execução estimado para uma consulta SQL. Na **figura 1**, podemos ver que após a criação do índice gist a pesquisa com o operador @@ se beneficia dele para acelerar a busca.

## Recursos avançados

O *tsearch2* é uma ferramenta bem completa para manipulação de textos, possuindo rotinas para indexar e buscar, bem como algumas rotinas adicionais para facilitar o desenvolvimento de aplicações.

Uma das funções disponíveis é a **rank()**, que utiliza as posições das palavras no texto para gerar um ranking de relevância para ordenar os resultados da consulta. A função *rank* recebe dois argumentos: o primeiro é o *tsvector* alvo, e o segundo é a *tsquery* de busca

utilizada. Pode ser utilizada no ORDER BY da consulta para dar a ordenação adequada ao número de ocorrências dos termos nos textos.

Também podemos definir pesos distintos para as palavras contidas no vetor, dependendo da sua posição. Para isso, podemos executar a função **setweight**, que recebe um *tsvector* e um peso que pode ser D, C, B ou A, sendo D o peso padrão atribuído a todas as posições pela função *to\_tsvector*, e A o maior peso possível. Supondo que a nossa tabela de exemplo agora também vai ter o título indexado, podemos usar uma função mais sofisticada para fazer a indexação, como mostrada na **listagem 2**.

Agora o nosso trigger de indexação inclui o campo título no vetor de índice, mudando o peso das suas palavras para A.

Outra opção para o programador é remover as informações sobre posição e peso das palavras, diminuindo o espaço ocupado pelo campo e pelo índice, e acelerando um pouco a busca. Isso pode ser feito com a operação

```
CREATE FUNCTION tsearch2_estendida() RETURNS TRIGGER AS `
NEW.campo_indice :=
  setweight(to_tsvector('simple', NEW.titulo), 'A') ||
  to_tsvector('simple', NEW.campo_texto);

NEW.campo_indice_stem :=
  setweight(to_tsvector('default_portuguese', NEW.titulo), 'A') ||
  to_tsvector('default_portuguese', NEW.campo_texto);

RETURN NEW;
` LANGUAGE 'plpgsql';
```

**Listagem 2.** Função alterada para incluir a indexação da coluna título

*strip*, que recebe como parâmetro um *tsvector*. Mas note que qualquer tipo de ranking terá que ser feito manualmente depois disso. Um exemplo de como ficaria o trigger de indexação usando a *strip* pode ser visto na **listagem 3**.

Outra funcionalidade interessante é a **headline**, que mostra apenas um extrato do texto original contendo as palavras usadas na busca, já acrescentando uma marcação html para destaque. Para chamar a **headline** é necessário passar como parâmetro o texto, a *tsquery* usada na busca e por último o texto que deve ser usado como marcador de destaque. Caso o terceiro parâmetro não seja definido, o padrão é usar as tags `<b>` e `</b>`.

## Conclusão

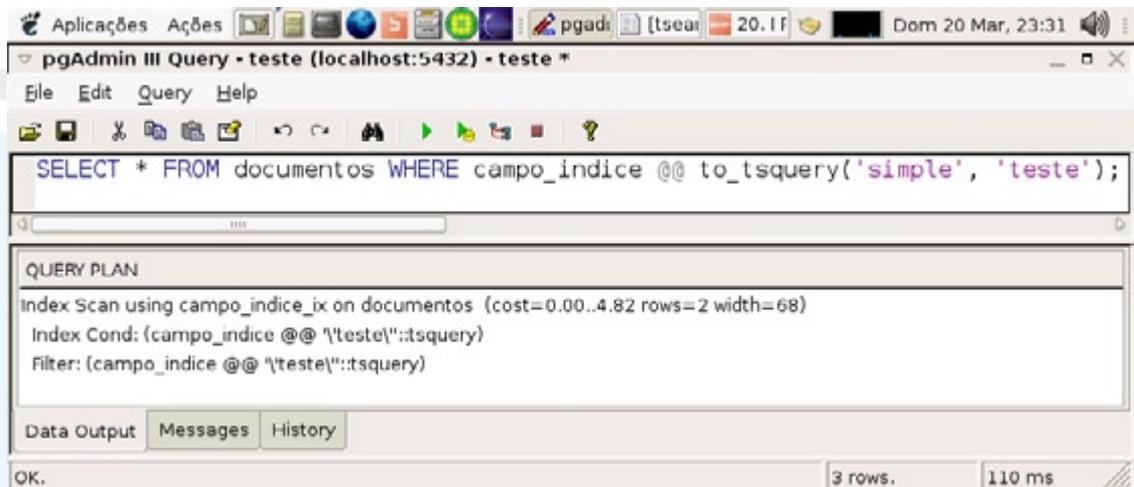
O *tsearch2* é uma poderosa ferramenta para construção de diversos tipos de ferramentas de busca. Podendo ser usada em sites, programas de catalogação eletrônica, datawarehouse e várias outras aplicações. Caso você já tenha uma grande base de documentos dentro de um banco de dados PostgreSQL,

```
CREATE FUNCTION tsearch2_estendida() RETURNS TRIGGER
AS `
NEW.campo_indice := strip(to_tsvector(`simple`, NEW.titulo || ` `
|| NEW.campo_texto));

NEW.campo_indice_stem := strip(to_tsvector(`default_portuguese`,
NEW.titulo || ` ` || NEW.campo_texto));

RETURN NEW;
` LANGUAGE `plpgsql`;
```

**Listagem 3.** Usando a função *strip*



**Figura 1.** Plan de utilização com consulta otimizada

essa é a solução ideal para você. Caso ainda esteja avaliando opções de indexadores independentes de SGBD, considere as vantagens de ter o mecanismo de busca integrado a uma poderosa base de dados relacional.

**Avalie esse artigo**

**Autor:**

Diogo de Oliveira Biazus

**Mini-curriculo**

Diogo de Oliveira Biazus está cursando Ciência da Computação no Instituto de Informática da UFRGS. É membro do Grupo Global de Desenvolvimento do PostgreSQL como contato e tradutor no Brasil e co-mantenedor do projeto PostgreSQL BR (<http://www.postgresql.org.br>). Atualmente trabalha como instrutor e consultor de PostgreSQL na TargetTrust em Porto Alegre. Blog do autor: <http://fumadordetabaco.blogspot.com/>

## Calendário de Eventos

Data	Evento	Site
7 a 9 Abril	Encontro Mineiro de Software Livre 2005 Local: Belo Horizonte - MG - Brasil	<a href="http://psl-mg.softwarelivre.org/tiki-index.php?page=EMSL2005">http://psl-mg.softwarelivre.org/tiki-index.php?page=EMSL2005</a>
8 a 10 Abril	II Festival Software Livre da Bahia Local: Bahia - SA - Brasil	<a href="http://twiki.im.ufba.br/bin/view/Festival">http://twiki.im.ufba.br/bin/view/Festival</a>
18 a 21 Abril	MySQL Users Conference 2005 Local: Califórnia - Santa Clara - USA	<a href="http://www.mysqluc.com/?r=d1">http://www.mysqluc.com/?r=d1</a>
1 a 4 Junho	6º Fórum Internacional de Software Livre Local: Porto Alegre - RS - Brasil	<a href="http://www.softwarelivre.org">http://www.softwarelivre.org</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