BLOB ou não BLOB, eis a questão:
Armazenar dados em campos BLOB, CHAR, ou VARCHAR?

Texto Original: Ivan Prenosil Tradução: Paulo Vaz

Diferenças entre o tipos CHAR e VARCHAR

Muitas pessoas acreditam que VARCHAR é melhor porque armazena somente dados significativos, enquanto o CHAR armazena em todo o dado em toda sua extensão. Isto **não** é verdade.

De fato, o CHAR e VARCHAR são armazenados no *buffer* de memória com o tamanho declarado. Quando o registro é armazenado no disco, o algorítimo *de compressão RLE* é utilizado para comprimir o registro inteiro, isto é CHARs, VARCHARs, inteiros, datas, etc. todos juntos. Assim se você quiser conservar o espaço, CHARs são ligeiramente melhores do que VARCHARs (a diferença é que VARCHAR armazena o tamanho do campo em 2 bytes).

Há também um erro que faz com que o VARCHAR não limpe corretamente a string se você atribuir um tamanho muito pequeno, assim causando uma compressão muito ruim (este problema já está corrigido a partir do Firebird-0.9.4)

Muitas pessoas acreditam também que VARCHAR transmite apenas os dados significativos através da rede, enquanto o CHAR é transmitido no seu tamanho total. Também **não** é verdade. A comunicação entre o cliente e o servidor é feita através de mensagens de tamanho fixo. Por esta razão o CHAR e VARCHAR são transmitidos em seu *tamanho declarado* (este problema é reparado no Ib-6.5)

Sendo assim, a decisão de usar CHAR ou VARCHAR deve ser baseada unicamente nas exigências da aplicação. Por exemplo, armazene códigos de tamanhos fixos no CHAR, armazene nomes em VARCHAR (para permitir a concatenação correta).

Vantagens/Desvantagens dos BLOBs e VARCHARs

Todos os comentários neste parágrafo de consulta do tipo VARCHAR são válidos para o tipo do CHAR. Cada comentário terminado com **BLOB+**, **VARCHAR+** ou **Implementação**, serve para indicar que tipo de dado é melhor para situação indicada, ou ainda se é uma questão de escolha ou implementação a ser feita.

Você sabe o tamanho máximo de seus dados?
Com VARCHARs você necessita declarar o tamanho máximo dos dados . Com Blobs você não necessita preocupar-se com o tamanho dele.
BLOB +

Você necessita armazenar dados em strings muito longas?
Uma única coluna de VARCHAR é restringida a 32K (isto é, sobre caracteres de 10Kb Unicode).
O tamanho máximo do Blob varia de acordo com o tamanho da página de dados informado na criação do Banco de Dados e são (de acordo com a guia da operação):
Páginas de 1Kb => 64 Mb
Páginas de 2Kb => 512 Mb
Páginas de 4Kb => 4 Gb
Páginas de 8Kb => 32 Gb
Páginas de 16 Kb => 256 Gb
BLOB+

Você necessita armazenar muitas campos de texto longo em uma única tabela?

O tamanho total de um registro (não compactado) é restringido a 64Kb. VARCHARs são armazenados diretamente do registro, portanto você não pode armazenar muitos strings longas em um única registro.

Já os Blobs são representados apenas por sua identificação no registro, já que são gravados fora do registro, e usam entre 8 bytes e 64Kb no máximo.

BLOB +

Você quer minimizar a tráfego entre o cliente e o servidor?

Os dados de VARCHAR são buscados junto com os outros dados de registro em uma unica operação, e geralmente diversos registros são transmitidos pela rede de uma só vez.

Cada Blob necessita de um operação extra de open/fetch.

VARCHAR +

Você quer minimizar a quantidade de dados transferidos entre o cliente e o servidor?

Com Blobs você tem a vantagem que *após* ter recuperado o registro você pode decidir se que ou não buscar dados do tipo Blob ligados à ele.

Com VARCHAR você tem a desvantagem, que estes são transmitidos através da rede no tamanho declarado (VARCHAR's muito longos degradam o desempenho de forma significativa em redes locais, para não mencionar conexões via dial-up.)

Este problema é reparado em Ib-6.5, que transmite somente dados significativos de campos VARCHAR.

BLOB +(**VARCHAR +** para IB 6.5)

Você quer minimizar o espaço usado?

VARCHARs são compactados com RLE (de fato o registro inteiro é comprimido, exceto os Blobs). Na maioria das vezes 128 bytes podem ser comprimidos a 2 bytes. Isto significa que um VARCHAR (32000) vazio, ocupará 500+2 bytes.

Os Blobs não são compactados, mas se vazios (isto é quando estão NULL) ocuparão somente 8 bytes no registro para sua identificação (e isto ainda poderá ser compactado via RLE). Um Blob com conteúdo pode ser armazenado na mesma página que outros dados do registro ou em uma página de dados separada. Um Blob pequeno que cabe na página dos dados acrescenta aproximadamente 40 bytes (ou mais) à página de dados.

Um Blob grande tem os mesmos 40 bytes adicionais na página dos dados, mais 28 bytes em em cada página de Blob adicional ocupada (30 bytes na primeira adicional). Uma página de Blob não pode conter mais de um blob (isto é, as páginas de Blob não são compartilhadas como páginas de dados).

Por exemplo, para o tamanho da página 4Kb, se você armazenar o Blob 5Kb, duas páginas do tipo Blob serão alocadas, o que significa que você perde 3Kb do espaço! Em outras palavras - quanto maior o tamanho da página, maior a probabilidade de que os blobs pequenos caberão na página dos dados, mas mais espaço será desperdiçado se as páginas separadas do blob forem necessárias para Blobs grandes.

VARCHAR + (exceto VARCHARs com tamanho declarado extremamente grande, ou tabelas com muitos Blobs vazios (NULL))

Você necessita de uma tabela com muitos registros?

Cada registro é identificado por uma DB_KEY (chave), que é um valor de 64 bits, onde 32 bits representem a identificação de relação e 32 bits são usados para localizar oregistro.

O número máximo de registros em teoria em uma tabela é 2^32 (mas por diversas razões o máximo real é sempre menor). Os identificadores do Blob (Blob_Id) são atribuídos no mesmo espaço de endereço que as DB_KEYs, e isto significa quanto mais Blobs na tabela, menos DB_KEYs estarão disponíveis para endereçar os registros da tabela.

Por outro lado, quando os registros armazenados são longos (por exemplo, se contiverem VARCHAR longos), então menos registros caberão na página dos dados e muitos DB_KEYs remanescerão sem atribuição de qualquer maneira.

VARCHAR +?

Você quer um bom desempenho?

Porque os Blobs grandes são armazenados fora das páginas dos dados, aumentam a "densidade" dos registros em páginas dos dados e o seu cache dando mais eficiência (reduzindo o número de operações de I/O durante a busca).

BLOB +

Você necessita executar buscas no conteúdo dos campos de texto?

Em campos VARCHAR você pode usar operadores como ' = ', ' > ', BETWEEN, IN(), LIKE (*case sensitive*), STARTING (*case sensitive*) e CONTAINING (*case insensitive*). Na maioria dos casos um índice pode ser usado para acelerar a busca dos dados.

Os Blobs não podem ser indexados, e você está restrito aos operadores LIKE, STARTING, e CONTAINING. Você não pode diretamente comparar Blobs diretamente com os operadores ' = ', ' > ' etc. (a menos que você use uma UDF que suporte isto). Você não pode, por exemplo, unir tabelas através do JOIN com campos do tipo Blob.

VARCHAR +

Você quer procurar o conteúdo destes textos com CONTAINING ?

O operador CONTAINING pode ser usado para se executar a busca *case-insensitive* em campos VARCHAR (não usando índice).

Visto não ser possível definir a ordem de *collation* para campos do Blob, você não pode usar inteiramente a busca *case-insensitive*, como no caso de caracteres nacionais em colunas do Blob (somente uma parte pequena do caracteres será *case-insensitive*). Como alternativa você pode usar uma UDF.

VARCHAR +

Você precisa transformar para UPPERCASE (maiúculas) os campos texto?

Você pode usar a função interna UPPER() em VARCHAR, mas não no Blob. (também o CAST, MIN e MAX, que não podem ser usados em campos Blob)

VARCHAR +

Não é possível classificar a coluna do tipo Blob (nem fazer um GROUP BY, DISTINCT, UNION ou JOIN ON). Não é possível concatenar colunas do tipo Blob.

VARCHAR +

Não há nenhuma função interna de conversão (CAST) para converter Blob em VARCHAR ou VARCHAR em Blob.

(mas é possível escrever UDF para esta finalidade.)

Implementação

Não é possível atribuir um valor à Blob usando comandos SQL diretamente.

Por exemplo não é possivel executar um comando assim:

```
INSERT INTO Tabela(MeuBlob) VALUES('abc ');
```

(mas é possível usar uma UDF convertendo string para Blob).

VARCHAR +

O Firebird-0.9.4 tem já esta funcionalidade

Implementação

Tamanho do arquivo de classificação (SORT FILE)

Às vezes quando as quando há necessidades do interbase classificar o resultado (por exemplo para a ORDER BY, DISTINCT, ou UNION sem ALL, etc..) cria-se um arquivo temporário. Este arquivo contem os valores de todas as registros que você está selecionando, isto é todos os VARCHARs, ou somente os identificadores dos Blobs que são mais curtos. Isto significa que campos Blob pode manter arquivos de classificação menores.

BLOB +

Os blobs podem ter a definição de recommended-segment-size, blob-sub_type, e usam filtros.

VARCHARs não suportam tal funcionalidade.

BLOB +

E*xistem dois tipos de campos Blobs - segmented (padrão) e stream.*

Blobs do tipo *stream* não é estruturado, enquanto os do tipo *segmented* são preservados no formato que foram inseridos no campo Blob, isto é preservam os vínculos do segmento. VARCHARs não suportam tal funcionalidade.

BLOB +

Você necessita de boa segurança nas campos texto?

Para recuperar dados da tabela, você necessita ter concedido o privilégio SELECT. Para recuperar um Blob, você necessita saber somente sua identificação (armazenada na tabela), mas o Firebird não verificará se você tem quaisquer direitos sobre a tabela com o campo Blob. Isto significa que todos que souberem ou acharem a identificação do Blob podem ler o Blob mesmo sem nenhum direito na tabela. (você pode tentar usar o comando BLOBDUMP do ISQL.)

VARCHAR +

Que ferramentas você vai usar.

A decisão final (se prefirar Blob ou Varchar) pode ser determinada pelas ferramentas usadas para acessar a base de dados (componentes, middleware de Delphi...). Algumas ferramentas podem ter problemas com a manipulação Blobs, algumas outras ferramentas podem ter limitações em VARCHARs (por exemplo limite a 255 caracteres ou a inabilidade de distinguir entre um campo vazio ou nulo).

Implementação

<table>
<tr><td>Artigo Original:

http://www.volny.cz/iprenosil/interbase/ip_ib_strings.htm#_strings_blob_varchar

Ivan Prenosil
prenosil@NOSPAM.ms.anet.cz</td><td>  </td></tr>
<tr><td>Tradução e adaptação:

Paulo Vaz
paulo@multi-informatica.com.br</td><td>Comunidade Firebird de Língua Portuguesa

Visite a Comunidade em:

http://www.comunidade-firebird.org</td></tr>
<tr><td colspan="2">A Comunidade Firebird de Língua Portuguesa foi autorizada pelo Autor do Original para elaborar esta tradução.</td></tr>
</table>