A partir desta edição, começaremos uma série de colunas, nas quais falaremos detalhadamente sobre *índices;*por que criar e como criar, como manter, qual a sua importância, melhores práticas, como funciona a “famosa” árvore binária, índices clusterizados e não-clusterizados... Enfim, iremos abordar tudo o que for importante saber para trabalhar bem essa *feature* básica e muito relevante nos bancos de dados objeto-relacionais.

Vale salientar, que os sources SQL/T-SQL aqui apresentados foram escritos e testados em ambiente Windows XP Professional, rodando SQL Server 2000 com Service Pack 4 instalado, usando o módulo Query Analyser.

**Como o SQL Server armazena e acessa dados**

O uso de índices pode trazer grandes melhorias para o desempenho do banco de dados. Pensando nisso, devemos então, primeiramente, entender como funciona o mecanismo que está trabalhando nos bastidores.

Os registros são armazenados em páginas de dados, páginas estas que compõem o que chamamos de pilha, que por sua vez é uma coleção de páginas de dados que contém os registros de uma tabela. Cada página de dados tem seu tamanho definido em até 8 Kb, apresenta um cabeçalho, também conhecido como header, que contém arquivos de links com outras páginas e identificadores (hash) que ocupam a nona parte do seu tamanho total (8 Kb) e o resto de sua área é destinada aos dados. Quando são formados grupos de oito páginas (64 Kb), chamamos este conjunto de *extensão*.



Os registros de dados não são armazenados em uma ordem específica, e não existe uma ordenação seqüente para as páginas de dados. As páginas de dados não estão vinculadas a uma lista, pois implementam diretamente o conceito de pilhas. Quando são inseridos registros em uma página de dados e ela se encontra quase cheia, as páginas de dados são divididas em um link é estabelecido para marcações e ligações entre elas.

**Como os dados são localizados**

Dentro da Arquitetura de índices do SQL Server, (assunto que detalharemos mais à frente) existem dois métodos para acesso a dados:

* **Exame de tabela**, que examina todas as páginas de dados das tabelas, começando do início da tabela passando por todos os registros, página a página e extraindo aqueles que satisfazem aos critérios da consulta.

* **Usando índices**, percorrendo a estrutura da árvore do índice para localizar os registros, por comparação, extraindo somente aqueles registros necessários para satisfazerem os critérios passados pela consulta.

Antes de tomar qualquer das decisões que foram apresentadas, o otimizador de consultas, componente responsável pela análise do melhor plano de execução da consulta, determina qual método será mais eficiente para recuperar os dados.

Mas a pergunta que surge rapidamente é, “Por que devo criar índices?”*.*

**Por que criar índices**

Os índices aceleram a recuperação dos dados. Por exemplo, imagine que você compre um livro de 800 páginas para suas pesquisas acadêmicas e este não apresente em seu conteúdo um índice reportando o seu conteúdo. Uma pesquisa talvez não fosse tão pavorosa, mas se você precisar de várias pesquisas, seria muito desagradável ficar horas procurando o conteúdo que deseja estudar. Por outro lado, um livro que apresente um índice de suas abordagens, se faz muito mais fácil e torna as pesquisas até prazerosas, pois teremos condição de irmos direto ao ponto que queremos.

Índices são sempre bem vindos em colunas de grande seletividade, como por exemplo, além da chave primária, que muitas vezes pode circular como identificador único da entidade na sua aplicação, você pode ter também um índice para colunas que poderão lhe auxiliar em consultas em que estas contarão com a cláusula WHERE, precisando ou não usar os operadores AND, OR ou \*NOT, que muitas vezes, em casos específicos, alteram a performance da consulta.

**Nota:** *\*O operador NOT sempre deixará sua consulta mais lenta que o normal.*

Um bom exemplo da criação necessária de índices, são aplicações bancárias que atendem à caixas eletrônicos. Sempre que solicitamos uma determinada transação ou mesmo informação, tal solicitação tende a ser cada vez mais rapidamente atendida. E quantos correntistas geralmente têm os grandes bancos? Será que quanto mais correntistas, mais lenta será a consulta?

Se não os índices, uma pesquisa pelo seu saldo demoraria quase o tempo de um almoço para retornar seu saldo ou mesmo, retornar uma resposta a sua solicitação de saque. Uma vez tendo ciência do funcionamento dos índices, respeitando a sua regra de negócios, uma consulta deverá ter resposta em tempo satisfatório.

**Por que não criar índices**

Os índices são muito bons no sentido de performance do banco de dados, otimizam as buscas de dados, mas, por outro lado, consomem muito espaço em disco, o que pode se tornar concorrente do próprio banco se você o detém em um espaço generoso ou pode se tornar caro quando de detém o banco em um storage.

**Considere as seguintes observações antes de criar índices**:

·         Quando colunas indexadas são modificadas, o SGBD desloca recurso internamente para manter esses índices atualizados e associados;

·         A manutenção de índices requer tempo e recursos, portanto, não crie índices que não serão usados efetivamente;

·         Quando se contém grande quantidade de dados duplicados, índices apresentam mais custo que benefícios. Assim como usar índices com atributos de pouca variação, como “sexo” ou atributos do tipo flag.

**Arquitetura de Índice**

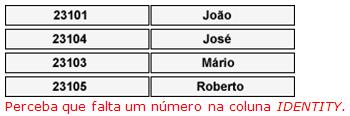
A arquitetura de índice contemplada dentro do SQL Server 2000 compreende-se em torno de tipos de índices e pilha de dados.

Existem três tipos de índices:

* **Índices de agrupamento ou ordenados***:* Os dados são armazenados em uma página de dados, em rodem crescente. A ordem dos valores nas páginas de índice também é crescente.
* **Índice sem agrupamento e de hash, criado sobre uma pilha***:*Quando um índice sem agrupamento é criado sobre a pilha, o SQL Server usa os identificadores de registros das páginas de índice que indicam os registros das páginas de dados.
* **Índices sem agrupamento ou de hash criados sobre um índice agrupado ou ordenado***:* Quando um índice sem agrupamento é criado sobre uma tabela com um índice de agrupamento, o SQL Server usa uma chave de agrupamento nas páginas de índice que indicam o índice de agrupamento. A chave de agrupamento armazena informações sobre a localização dos dados (headers em forma de hash).

Para manipular as pilhas, o SQL Server apresenta um mecanismo chamado “*IAM”* (Index Allocation Map), que contêm informações sobre onde às extensões de uma pilha são armazenadas. São usadas para navegar pela pilha e encontrar espaços disponíveis para os novos registros inseridos e, além disso, são responsáveis por conectar as páginas de dados.

No caso que você tenha um atributo inteiro, definido como chave primária e sendo assim, declarado com *IDENTITY*, a pilha de dados poderá não conter a mesma ordem física, caso seja uma tabela com grande volume de inserções e exclusões. A figura abaixo mostra uma pilha contendo a chave primária ‘*código*’ e um índice qualquer ‘*nome*’. Olhando bem a figura você compreenderá que o mecanismo de arrumação da pilha, rapidamente, após uma exclusão seguida por um novo cadastro, faz a realocação do novo registro e este é inserido onde anteriormente existia um valor. Resumindo, o mecanismo restaura o espaço para novos registros na pilha após exclusões.



**Usando índices de agrupamento ou ordenado (CLUSTERED)**

Os índices agrupados são criados automaticamente na maioria das tabelas que criamos, pois, quando não declaramos *NONCLUSTERED* em uma chave-primaria, este campo automaticamente assume o valor de *CLUSTERED.*Mas, alguns fatos devem ser levados em conta na criação de índices agrupados, tais como:

* Cada tabela poderá ter apenas um e somente um índice agrupado, uma PK por exemplo;
* A ordem dos registros da tabela e a ordem dos registros do índice são iguais. Você deve criar índice de agrupamento antes de criar índices sem agrupamento, pois um índice de agrupamento altera a ordem física dos registros da tabela. Os registros são classificados em uma ordem seqüenciada e mantidos nessa ordem;
* A exclusividade de valores de chave é mantida de modo explícito, com a palavra-chave *UNIQUE*, ou de modo implícito, com um identificador interno único. Esses identificadores são internos do SQL Server e não podem ser acessados pelo usuário;
* 5 % do tamanho da tabela é o limite para o tamanho permitido para o valor do atributo indexado;
* Durante a criação de um índice de agrupamento, será necessário espaço em disco de aproximadamente 1,2 vezes o tamanho atual da tabela na qual pertence à coluna que será indexada. Após esta operação, o espaço em disco é restaurado automaticamente.

**Usando índices de sem agrupamento ou de hash (NONCLUSTERED)**

Os índices sem agrupamento são úteis quando os usuários precisam de várias maneiras para pesquisar dados. Por exemplo, um leitor pode pesquisar frequentemente em um livro sobre jardinagem os nomes comuns e científicos das plantas. Você poderá criar um índice sem agrupamento para recuperar os nomes científicos e um índice de agrupamento para recuperar os nomes comuns.

Você tem toda flexibilidade para combinar declarações SQL para manipular tais índices, como criar um índice sem agrupamento, que mantenha unicidade usando *UNIQUE* em meio à declaração de criação de índice, que veremos mais à frente.

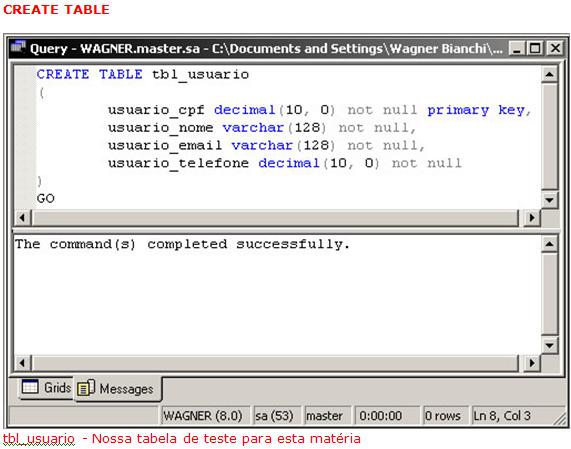
Alguns fatos que devem ser entendidos:

* A ordem das páginas de dados de índices sem agrupamento não apresentam a mesma ordenação que a ordenação física ou das tabelas;
* A exclusividade é mantida no nível folha com chaves de agrupamento ou identificadores de registro;
* Podem existir até 249 índices sem agrupamento por tabela;
* A criação de índices sem agrupamento é mais eficiente em colunas que apresentem seletividade alta e única;
* Os identificadores de registro especificam a ordem lógica dos registros e consistem na identificação do arquivo, no número da página e na identificação do registro.

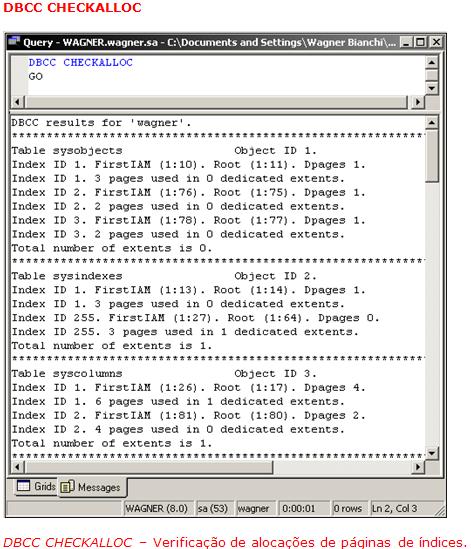
**Um pouco de prática...**

No SQL Server podemos exibir, estando já no Query Analyser e conectando a uma base de dados, os mapas de alocação de índices de forma bem fácil.

Vamos então, criar uma tabela para buscarmos em cima dela, algumas definições de índices básicas e de grande relevância para a conceituação.

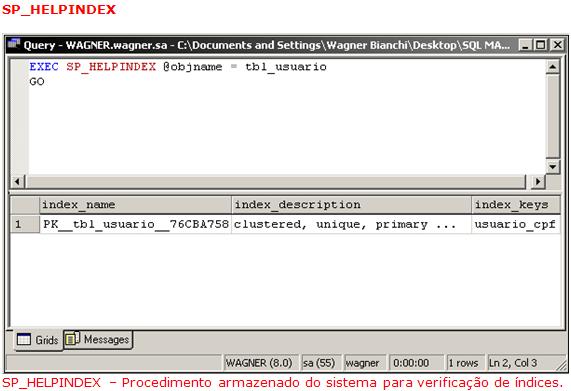


O seguinte comando exibe as informações dos mapas, também conhecidos como “*IAM”*, já mencionado aqui neste artigo.



Executando o comando *DBCC CHECKALLOC*, visualizamos as páginas de alocação de índices, quantidade de extensões daquele determinado índice e a qual objeto do nosso banco de dados ele pertence.

Temos também outro recurso próprio, uma system stored procedure, que nos ajuda a verificar índices próprios de uma tabela específica, dentro de um schema. O seguinte procedimento nos mostrará índices contidos em nossa tabela.



Veja que a coluna *index\_name* exibe o nome do índice que pertence à tabela que passamos na declaração *@objname.*Como só podemos criar um índice clusterizado/agrupado por entidade, podemos criar outros índices não agrupados caso seja pertinente com a regra de negócios a ser aplicada. Na última imagem, podemos perceber também, a qual atributo da entidade está aplicado o índice e também suas descrições.

Lembrando que o nome do índice apresentado no como valor do atributo *index\_name*poderá variar de servidor para servidor.

**Como o SQL Server recupera os dados armazenados**

Todos os índices criados em uma base de dados dentro do SQL Server, tem suas informações armazenadas em uma tabela chamada *SYSINDEXES*, que contém informações estatísticas, como o número de registros e páginas de dados em cada tabela, além de descrever como localizar as informações que são apontadas pelos índices.

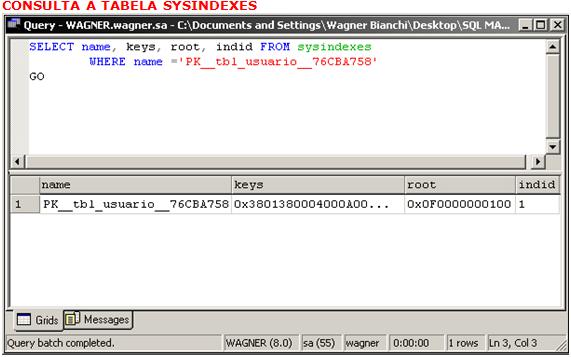
Levando em conta que cada tabela possui uma coleção de páginas de dados, cada tabela e índice são identificados de forma exclusiva pela combinação entre coluna identificadora (PK, por exemplo) e a coluna identificadora de índices (*INDID*).

**Consultando informações na tabela sysindexes**

A tabela do sistema sysindexes é o local central para informações vitais sobre objetos como entidades e índices destas entidades. Contém informações estatísticas, como o número de registros e páginas de dados em cada tabela. Além disso, descreve como localizar as informações armazenadas em uma tabela de dados.

Os ponteiros de páginas da tabela sysindexes ancoram todas as coleções de páginas de tabelas e índices. Cada tabela possui uma coleção de páginas de dados, além de coleções de páginas adicionais para implementar cada índice definido para a tabela.

Um registro na tabela sysindexes de cada tabela e um índice é identificado de forma exclusiva pela combinação entre a coluna identificadora de objetos (**id**) e a coluna identificadora de índices (**indid**).



É fato que bancos de dados trabalham bem mais rápido com campos que armazenam números, já que os processadores, tanto de tecnologia *Cisc* (Complex Instruction Set Computer) quanto *Risc* (Reduced Instruction Set Computer), são ótimos em comparar maiores, menores, múltiplos, divisores, iguais, diferentes, enfim, trabalham bem com números. Essa pequena abordagem se deve ao fato de, tudo dentro de um sistema de computação estará sendo analisado com conversões binária ou hexadecimal. Com bancos de dados e mais precisamente com a tabela sysindexes também não se faz contrário.

**Descrevendo o que vemos na última imagem**:

* **name e indid**: o nome do índice que está sendo usado no campo usuário\_cpf da entidade tbl\_usuario, que é nossa (*PK*), isso quer dizer diretamente que seu *INDID* ou identificador de índice na tabela sysindexes é igual a 1;
* **keys e root**: os campos que guardam dados varbinary e bynari, respectivamente, são uma identificação única dos índices que são criados, localização na árvore e indicadores para seus dados nas páginas de dados.

**Localizando registros sem índices**

Quando não existe nenhum índice em uma determinada tabela, o Otimizador de consultas é acionado e então utiliza a varredura de tabela para recuperar registro, uma das formas que este usa, a qual já vimos anteriormente.

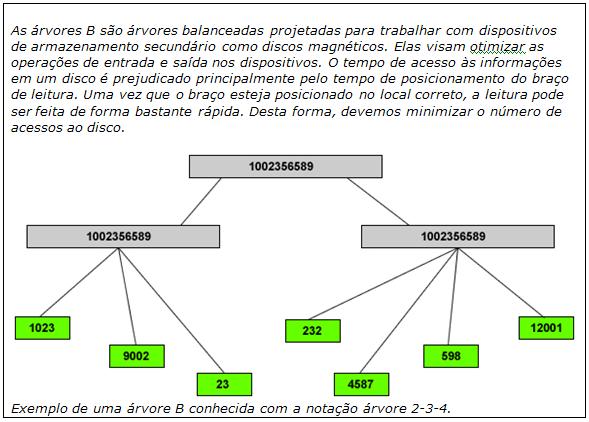
Não é uma boa prática visto que, em uma tabela com muitos registros ou mesmo uma tabela que possa ser considerada com grande, não haverá apontamentos para indicar onde estão os dados que estamos buscando. A performance em buscas desse tipo pode não ser tão satisfatória quando se deseja recuperar poucos dados.

Os registros são retornados fora da ordem. Talvez eles sejam inicialmente retornados na mesma ordem da inserção, mas essa ordem não será mantida, já que após algumas exclusões as novas inserções ocuparão esses espaços, tornando a ordem imprevisível.

**Localizando registros sem agrupamento/não-clusterizados em uma pilha**

Nesse caso, as disposições índices e dados estarão em planos diferentes, sendo que, os índices dispostos como um índice de um livro e os dados como o conteúdo do livro. Aí que entram as idéias de apontamentos. Os ponteiros indicam o local de armazenamento dos itens indexados na tabela subjacente.

Os índices dentro da Arquitetura do SQL Server são organizados, implementado o conceito de \***árvore B**, sendo que cada página de índice contém um cabeçalho de página seguido por registros de índice. Cada registro de índice contém um valor de chave e um ponteiro para outra página ou registro de dados, formando os cabeçalhos já vistos outrora aqui.



O SQL Server utiliza os níveis da árvore B com notações nó de índice, nível raiz e nível folha ou nó folha. Quaisquer níveis entre os nós raiz e folha são chamados de níveis intermediários. Cada página nas camadas intermediárias ou inferiores tem ponteiros ou apontamentos anteriores ou posteriores em uma lista dupla relacionada.

Em uma entidade que só contenha um índice sem agrupamento, os nós folha possuem localizadores de registros com apontamentos para registros de dados que contém os valore de chave. Cada ponteiro (RID ou ROWID – identificador de registro ou de linha) é criado com base na identificação do arquivo, no número da página e no número do registro da página.

**Localizando registros em um índice de agrupamento**

Os índices de agrupamento e sem agrupamento compartilham da mesma estrutura dentro da “*árvore B”*, mas com algumas diferenças:

·         As páginas de dados de um índice de agrupamento são os nós folha da estrutura da “*árvore B*”;

·         Os registros são armazenados em ordem sequencial baseada na chave do agrupamento;

Um índice de agrupamento (clusterizado ou ordenado) é como um índice remissivo de um livro, como já citamos, os assuntos estão agrupados todos por uma ordem ascendente, facilitando a pesquisa de dados e localização dos mesmos dentro da árvore. Lembrando que não importa se esta é muito ramificada devido ao seu tamanho.

Como um índice de agrupamento determina a seqüência em que os dados são armazenados em uma tabela, só pode haver um índice deste tipo por entidade.

**Localizando registros em um índice de agrupamento com índice sem agrupamento**

Quando um índice sem agrupamento é adicionado a uma tabela que já tem um índice de agrupamento, o localizador de registro de cada índice sem agrupamento contém o valor de índice da chave de agrupamento do registro.

Quando forem usados índices de agrupamento e sem agrupamento em uma mesma tabela, as estruturas da árvore B e dos índices devem ser percorridas para que os dados sejam localizados. Isso gera custo alto com I/O.

Como o valor de um índice de agrupamento é maior do que o RID de 8 bytes usado para a pilha, os índices sem agrupamento pode ser substancialmente maiores em tabelas de agrupamento indexadas do que quando criados em pilhas. Se você mantiver baixos os valores de chave do índice de agrupamento, isso lhe ajudará a criar índices menores e mais rápidos.