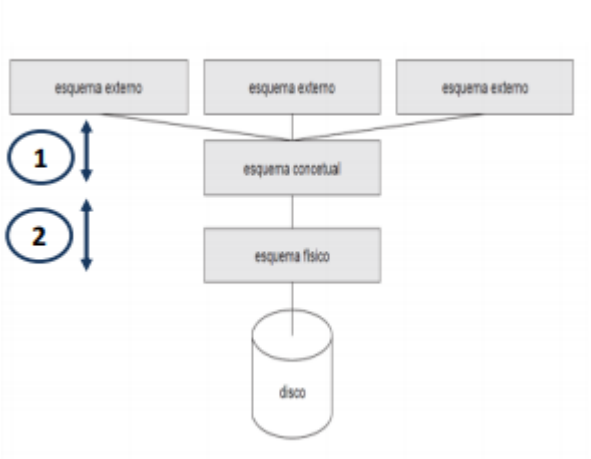
ARMAZENAMENTO



ANSI-SPARC Arquitetura

1 – Independencia lógica

2 – Independencia física

Estrutura dos ficheiros

Uma BD é mapeada num conjunto de ficheiros persistidos em disco sobre o filesystem do SO

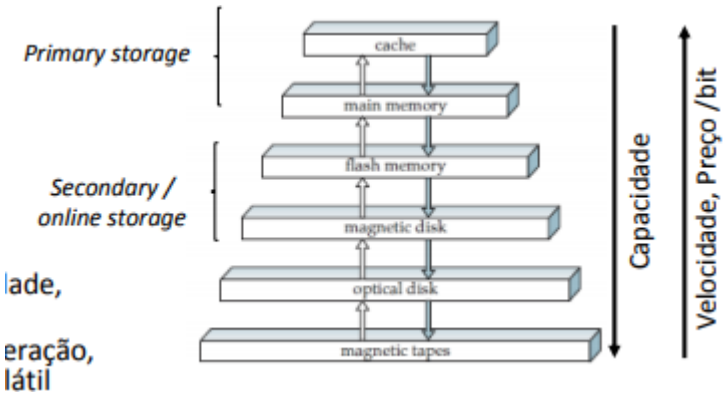
Um ficheiro:

- Constituido por conjunto de blocos;
- Um bloco conterá vários registos.

Caracteristicas:

- Capacidade (e disponibilidade);
- Velocidade de acesso;
- Preço (por unidade de dados);
- Fiabilidade.

Tipicamente temos os níveis:



- **Cache**: Rápida, de reduzida capacidade, gerida pelo SO;
- **Memoria RAM**: para dados em operação, capacidade média/baixa, muito volátil;
- **Flash/SSD**: acesso rápido, capacidade media alta, não voláteis, caros melhora o desempenho num nível adicional de cache;
- **Discos magnéticos**: na grande maioria ainda os mais utilizados (devido ao custo) grande capacidade, performance qb (dependendo de outros fatores de otimização), suporta a persistência permanente das alterações à BD e embora suscetíveis de falha existem precauções possíveis.

Caracteristicas:

Ainda o modo mais comum/disseminado de persistência

Prato: duas faces

Sector: ~512 bytes

Pistas (tracks): ~50,000 to 100,000/prato

Conjunto: de 1 a 5 pratos

Cilindro: considerando o movimento solidário de todas as cabeças/braços!

Só uma cabeça está ativa em cada instante

Controlador/Interfaces (velocidades): SATA, SATA II, SATA 3, SCSI…

Medidas de Desempenho

Velocidade de rotação: numero de rotações por minuto;

Tempo de busca: tempo necessário para deslocar a cabeça de leitura para o cilindro pretendido;

Atraso de Rotação: tempo necessário para o disco na sua rotação, posicionar o sector pretendido, de forma a ser lido pela cabeça de leitura;

Track-to-Track Seeks: tempo necessário para mover a cabeça de uma pista para a adjacente (factor relevante na leitura sequencial);

Average Seek Time: tempo que em media as cabeças levam a pesquisar informação em pistas aleatórias (factor relevante na leitura aleatória);

Acess time: intervalo de tempo entre pedido de leitura/escrita e inicio da transferência de dados;

MTTF: tempo medio entre falhas, medida de fiabilidade dos discos

Tecnicas de recolha dos dados

Persistidos em disco segundo a estrutura de blocos:

- Buffering;

- Read-ahead;

- Scheduling;

- File system frag/defrag techniques

- Non-volatil memory(intermediary) buffers;

- Log disk w/ sequencial read/write operations, pre definitive persistence

RAID (redundant array of independente disks)

Método de combinar vários discos rígidos de forma a aumentar capacidade e performance, bem como, originar redundância de dados, prevenindo assim falhas do disco rígido.

Redudancia = Fiabilidade

- Não será completamente eficiente;

- Pode também não ser completamente eficaz

Paralelismo = melhoria de performance

Raid 0

Usa a técnica de data striping (segmentação de dados)

Várias unidades de disco rígido são combinadas para formar um volume grande;

Pode ler e gravar mais rapidamente do que uma configuração não-RAID, visto que segmenta os dados e acede a todos os discos em paralelo

Não permite redundacia de dados;

Necessita de pelo menos 2 unidades de disco rígido.

Raid 1

Cria um espelho do conteúdo de uma unidade, noutra unidade do mesmo tamanho

A replica fornece integridade de dados otimizada e acesso imediato aos dados se uma das unidades falhar

Requer um mínimo de duas unidades de disco rígido e deve consistir em um numero par de unidades.

Raid 5

Proporciona o equilíbrio entre redundância de dados e capacidade de disco

Segmenta todos os discos disponíveis num grande volume

O espaço equivalente a uma das unidades de disco rígido será utilizado para armazenar bits de paridade

Se uma unidade de disco rígido falhar, os dados serão recriados através dos bits de paridade.

Raid 10 (1+0)

É a combinação de discos espelhados (raid-1) com a segmentação de dados (data stripping) (raid-0).

Oferece as vantagens da transferência de dados rápida da segmentação de dados, e as características de acessibilidade dos arranjos espelhados.

A performance do sistema durante a reconstrução de um disco é também melhor que nos arranjos baseados em paridade.

Fatores de Decisão

Custo de discos extra

Requisitos de performance

Requisitos de performance em caso de falha (operacional e de reconstrução)

Implementação: Software vs Hardware (fiabilidade e desempenho)

Registos nos ficheiros:

Fixed-length records

Variable-length records

Variable-Length Records

Armazenamento de múltiplos registos que contenham campos de tipos com "comprimento variável" ou, oriundos de "entidades tipo" diferentes

Como representar estes registos? (e armazenar nos blocos?) De forma a que os valores dos seus campos sejam acedidos/manipulados com facilidade.

Como armazenar então nos blocos registos de comprimento variável?

Normalmente existe um header no inicio de cada block, com:

1. O numero de registos persistidos,
2. A indicação do final do espaço livre no bloco;
3. Um array com a indicação da localização e tamanho de cada registo.

Organização dos Registos em Ficheiros

Vimos como são representados os registos, Como se podem então organizar os registos pelos ficheiros?

Alternativas:

- Heap File Organization: um registo pode ser colocado em qualquer local do ficheiro em que exista o espaço. Não há nenhuma ordenação. Tipicamente um ficheiro por cada "relação" (MR)

- Sequential File Organization: Os registos são guardados sequencialmente segundo a "chave de pesquisa"

- Hashing File Organization: A localização é função do calculo de uma função hash que mapeia o registo no seu bloco dentro do ficheiro

Multi-table clustering

Bases de dados simples e (expectavelmente) "pouco" populadas, têm tipicamente uma estrutura simplificada:

- Tuplos de uma relação são representados como fixed-length records;

- E as relações podem estar mapeadas numa estrutura de ficheiros simples: 1relação–1ficheiro

Em BDs mais exigentes (em performance e considerando dimensão e volume de dados):

- O SGDB fará a gestão do(s) ficheiro(s);

- dos blocos nos ficheiros;

- dos registos a persistir nos blocos

Nos casos em que múltiplas relações são guardadas no mesmo ficheiro existe por vezes a preocupação de que os blocos contenham registos de uma só relação

Contudo, pode justificar o acrescento de complexidade de juntar tuplos de relações diferentes num mesmo bloco.

Buffer Manager

Um dos objetivos de uma implementação de um SGBD é minimizar o numero de transferências de blocos entre o disco e a memoria.

O buffer é o local onde se mantem copias em memoria dos dados (parcela) persistidos em disco

Buffer Management

- Sempre que há uma chamada query/execução ao SGBD

Se já existir em buffer os blocos respetivos este é (são) disponibilizados

Se o bloco não está no buffer, este terá de alocar espaço para o recolher do disco

Se necessário “livra-se” de algum(s) existente(s)

Copiando-os de volta para o disco se a versão em memoria for mais atualizada

Este processo “interno” é transparente para o programa que solicita o SGBD

Files & Filegroups

As bases de dados são criadas tendo como suporte um conjunto de ficheiros específicos

Os ficheiros podem ser agrupados em filegroups de forma a facilitar a sua gestão, localização/peristencia e performance no seu acesso

Uma base de dados tem que ser criada com pelo menos um ficheiro de dados e um ficheiro de log (exclusivos de cada base de dados)

Tipos de ficheiros

Primario, secundario e log de transações

Ficheiro primário

Contem informação de inicialização da base de dados

Agrupa tabelas (de sistema):

- geradas/atualizadas automaticamente aquando da criação de uma nova BD
- Contem os utilizadores, objetos e permissões de acesso

Contem referencias para os restantes ficheiros da BD

Dados do utilizador podem ser persistidos neste ou, num ficheiro secundario

Existe apenas e obrigatoriamente um ficheiro primário por base de dados

Ficheiro secundario

São opcionais

- A base de dados pode não ter ficheiro secundario, se todos os dados estiverem armazenados num ficheiro primário

São definidos pelo utilizador

Podem armazenar tabelas/objetos que não tenham sido criados no ficheiro primário

Algumas bases de dados beneficiam de múltiplos ficheiros secundários de forma a dispersar os dados por vários discos

Permitem melhorar desempenho e escalabilidade, se/quando previsível que o Primary data file atinja o limite máximo de um ficheiro para o SO

Log de transações

Utilizado para armazenar o "rasto" de transações

Persiste a informação necessária à recuperação da base de dados

Todas as bases de dados têm pelo menos um log file

Embora em sistemas simples a aproximação default é manter os data e transaction files na mesma path;

Em casos mais complexos/exigentes é recomentado que os ficheiros de dados e de log de transações sejam colocados em discos separados.

Filegroups

Permitem aumentar a performance da base de dados ao facilitarem a distribuição dos ficheiros que os constituem, por vários discos ou sistemas RAID.

Tipos de filegroups:

- Primario

- Utilizador

- Por omissão

Cada BD conterá sempre um Primary filegroup (este contem primary e secondary data files)

O utilizador pode criar Filegroups para agregar data files com objetivos.

Fatores a considerar no projeto/dimensionamento da BD

A função do sistema – tipo de aplicações:

- OLTP – muitos utilizadores simultâneos; tempo de resposta limitado;

- "OLAP"/DSS – consultas complexas com tempos de resposta alargado;

- Batch – Processamento intenso sem interface c/ utilizador

Requisitos não funcionais:

- Performance mínima para as transações do sistema;

- Capacidade do sistema;

Periodos de disponibilidade (Uptime)

Uma das fases mais importantes do desenho do sistema é a determinação da localização dos ficheiros da base de dados

Considerar:

1. Uma tabela frequentemente acedida deve ser colocada num filegroup localizado num array com grande numero de discos.
2. Se não for utilizado RAID mas houver múltiplos drives de disco, pode-se ainda utilizar filegroups
3. Pode ser colocado um ficheiro por drive de disco e por filegroup definido pelo utilizador.

Recomendações RAID

RAID 1

- Sistema Operativo;

- Transaction Log.

RAID 5

- Tabelas de dados com acesso essencialmente em modo de leitura;

- BDs temporárias.

RAID 10

- Tabelas de dados com acesso frequente em modo de Escrita;

- Também se usa no Transaction Log;

- BDs temporárias.

METADATA

Uma BD sobre a BD

A BD tem de manter um conjunto de dados sobre os dados!

Esta informação está persistida no que se constuma designar o Catalogo ou Dicionário.

Sys views

1. Uma das formas (preferencial = desempenho) de aceder aos metadados.
2. Por setem views suportam a independencia de eventuais alterações "físicas" às tabelas de sistema.
3. Quer as views quer as suas colunas são auto-descritivas, de forma a aprender a informação relativa aos metadados solicitados.

Information Schema views

1. Seguem as definições standard ISO relativas às vistas sobre o catalogo.
2. Apresentam a informação dos metadados em formato independente de qualquer implementação das tabelas do catalogo.
3. As aplicações quando usam esta coleção de views são tendencialmente mais portáveis entre diferentes SGBDs.

INDICES

Definições:

Índices: mecanismos de indexação para aumentar o desempenho no acesso à informação.

Fornecem um caminho de acesso aos registos.

Um ficheiro de indexação é constituído por registos, no seguinte formato: Chave de pesquisa – Apontador

Chave de pesquisa: atributo ou conjunto de atributos usado para "procurar" os registos num ficheiro.

Os ficheiros de indexação são de menor dimensão que os ficheiros originais.

A existência de índices não modifica as relações nem a semântica das consultas.

Tipos de índices:

1. **Índices ordenados:** as chaves de pesquisa são armazenadas por uma certa ordem.
2. **Índices Hash:** as chaves de pesquisa são distribuídas uniformemente por "buckets" através de uma função de Hash.

Indices ordenados

Indice primário vs secundario

As entradas de indexação são ordenadas pela chave de pesquisa

Indice primário: num ficheiro sequencial ordenado, é o índice que a chave de pesquisa especifica como ordem sequencial do ficheiro

Num clustered índex a chave de pesquisa pode não coincidir com a chave primaria (a chave de pesquisa segue a organização/ordenação do ficheiro, contudo não é campo chave).

Indice secundario (nonclustered índex): índice em que a chave pesquisa especifica uma ordem diferente da ordem sequencial do ficheiro.

Indices "**densos**":

A cada valor da chave de pesquisa corresponde um registo de índice.

Se for um dense nonclustering índex então cada entrada de registo de índice terá de guardar todos os ponteiros de para todas as ocorrências do valor apontado.

Indices "**esparsos**":

Contem registos de índice, apenas para alguns dos valores da chave de pesquisa.

Aplicável quando os registos estão ordenados sequencialmente pela chave de pesquisa.

Para localizar um registo com o valor K da chave de pesquisa:

1. Localizar o registo de índice com o maior valor < K, da chave de pesquisa.
2. Pesquisar sequencialmente o ficheiro a partir do registo apontado pelo registo de índice.

Ocupa menos espaço, menor “overhead” nas operações de insert e delete, normalmente mais lento na procura de registos.

Avaliação das diferentes técnicas de indexação

Nenhuma é absolutamente melhor que outra em todos os aspetos.

Há que considerar:

- Tipos de acessos à informação:
  1. Registos com um atributo, igual a um valor especifico;
  2. Registos com um atributo, com valor num determinado intervalo de valores.
- Tempo de acesso.
- Tempo para insert.
- Tempo para delete.
- Espaço adicional.

Dica: uma entrada num índice esparso por cada bloco:

O custo de processamento está normalmente associado a encontrar e trazer o bloco do ficheiro do disco, uma vez no bloco o tempo de procura dentro do mesmo é negligenciável.

Indices multinível

Se o índice primário não “cabe” em memoria o acesso torna-se dispendioso.

De modo a diminuir o número de acessos a disco, tratar o índice primário como um ficheiro sequencial e criar um índice esparso sobre o índice primário:

- **Indice interno:** O ficheiro sequencial do índice primário;
- **Indice extrior:** Indice esparso do índice primário.

Se o índice extrerior não “couber” em memória, criar um novo nível de índice -> mais um nível.

Todos os níveis dos índices devem ser atualizados nas operações de insert, update e delete.

Indices secundários

Motivação

Frequentemente, é necessário pesquisar registos por certos atributos, que não são o atributo da chave de pesquisa do índice primário ou clustered.

Considerações

Os índices secundários têm de ser densos

Os índices aumentam o desempenho na consulta de registos

Quando a informação é modificada, cada ficheiro de índices também tem de ser atualizado

O varrimento sequencial sobre o índice primário é eficiente.

O varrimento sequencial sobre índices secundários é dispendioso.

Indices B+ -Tree

Balanced Tree como alternativa aos índices sequenciais:

Desvantagem dos índices sequencias:

1. O desempenho diminui à medida que a dimensão do ficheiro aumenta;
2. É necessária reorganização periódica do ficheiro.

Vantagem dos índices B+ -Tree:

1. Reorganização automática em alterações pequenas e locais, resultantes das operações insert e delete;
2. Não é necessária a reorganização total frequente do ficheiro de modo a manter o desempenho.

Desvantagens dos índices B+ -Tree:

1. Acrescimo de processamento em insert e delete
2. Acrescimo de espaço ocupado

As vantagens sobrepõem-se às desvantagens, por isso são comumente utilizados

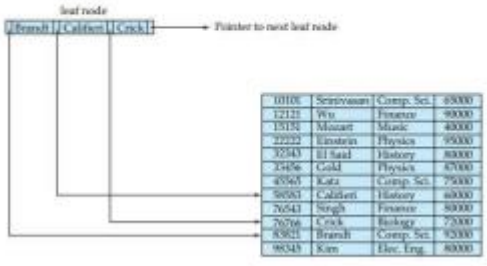
Caracteristicas

Normalmente índices densos

Organização em arvore que têm as seguintes propriedades:

- Todos os caminhos da raiz às folhas têm o mesmo comprimento;
- Cada nó que não é raiz ou folha tem entre [n/2] e n filhos;
- Cada nó folha tem entre [(n-1)/2] e n-1 valores
- Casos especiais:
  1. Se a raíz não é folha tem pelo menos 2 filhos;
  2. Se a raiz é folha, pode ter entre 0 e (n-1) valores.

Estrutura dos nós



Ki são os valores da chave de pesquisa.

Pi são os apontadores para os filhos (para nós que não são folha) ou, apontadores para registos ou buckets de registos (para os nós folha)

Os valores da chave de pesquisa estão ordenados no nó K1<K2<K3<…<Kn-1

O apontador Pi, aponta para um registo no ficheiro com o valor da chave de pesquisa Ki

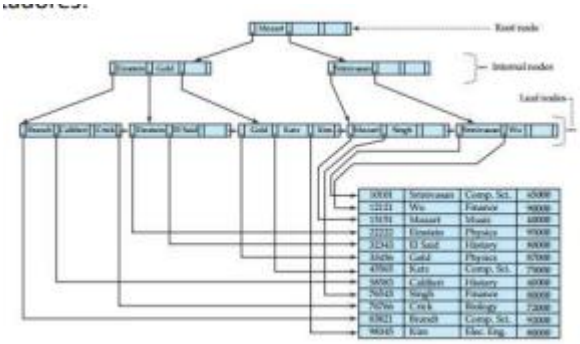
P**n** aponta para a próxima folha ordenada pela chave de pesquisa

Estrutura dos nós não folha

Constituem um índice esparso multinível dos nós folha

Para um nó não folha com **m** (<n) apontadores:

1. A subárvore apontada por P1 conterá chaves de pesquisa de valor <= K1



Para 2 <= i <= n-1,

1. Todos os valores da chave de pesquisa da subárvore para a qual Pi aponta são maiores ou iguais a Ki-1 e menores que Km-1

Contem até n apontadores, e pelo menos [n/2]

O numero de apontadores num nó é denominado de fanout do nó

Altura da árvore e Espaço necessário

Consideremos uma entrada no índice: 12(chave de pesquisa) + 8 (apontador) = 20 Bytes

Com páginas/blocos de 8KB

Com cerca de 400 entradas por pagina

Assumindo páginas semipreenchidas contendo cerca de 250 entradas

Nível	N.º de chaves	Tamanho (*bytes*)
1	250	8 K
2	$250^2 = 62\,500$	2 MB
3	$250^3 = 15\,625\,000$	500 MB

Mesmo contendo 15 milhões de chaves, mantendo somente a raiz em memoria (8K) é possível encontrar qualquer chave com um máximo de 2 acessos ao disco.

Atualmente dada a memoria disponível é possível com 1 ou mesmo nenhum acesso ao disco.

Consultas

Obter os registos com o valor K da chave de pesquisa

1. Inicio no nó raiz: Examinar o nó para o valor mais pequeno da chave de pesquisa > K. Se o valor existe, assumir que é Ki. Seguir para o nó filho apontado por Pi. Senão com K>= Km-1, seguir para o nó apontado por Pm.
2. Se o nó "seguido" pelo apontador do passo anterior não é folha, repetir o procedimento anterior.
3. Quando no nó folha, para a chave i, Ki = K, seguir o apontador Pi para o registo ou bucket. Senão não existem registos com o valor K.

Indices B-Tree

Definição

Semelhantes às B+ -Tree, mas… Ekimina redundância de armazenamento dos valores das chaves de pesquisa, pois estes apenas aparecem uma vez na arvore.

Caracteristicas

Vantagens:

1. Menor numero de nós
2. Por vezes não é necessário percorrer a arvore até às folhas para obter o valor da chave de pesquisa.

Desvantagens:

1. Só uma pequena fração de todos os valores da chave de pesquisa são obtidos mais "cedo"
2. Os nós não folha tem maior dimensão
3. As operações de insert e delete têm processamento mais complicado

Tipicamente as vantagens não se sobrepõem às desvantagens.

Hashing – estático

Definições

Um bucket é uma unidade de armazenamento, que contém um ou mais registos.

Num ficheiro com organização hash, o acesso direto a um bucket, através do valor da chave de pesquisa, é obtido utilizando uma função de hash

Com K o conjunto de valores da chave de busca e B o conjunto de todos os buckets, **H é uma função de hash de K para B**

A função de hash, é utilizada para aceder, inserir e remover registos.

Registos com diferentes valores de pesquisa, podem estar associados ao mesmo bucket;

Funções de Hash

No pior caso, a função hash mapearia todos os valores da chave de pesquisa, para o mesmo bucket;

1. Assim o tempo de acesso é proporcional ao numero de valores da chave de pesquisa existentes;

A função hash deve cumprir os seguintes requisitos:

1. Uma função uniforme, ou seja cada bucket está associado ao mesmo numero de valores da chave de pesquisa (de todo o intervalo de valores possíveis).
2. Uma função aleatória, em cada bucket contem aproximadamente o mesmo numero de valores da chave de busca, considerando a distribuição atual.

Tipicamente as funções calculam e utilizam a representação binária dos valores da chave de busca.

Bucket overflow

Pode ocorrer por:

1. Numero de buckets insuficiente;

2. Alguns buckets têm mais valores que outros, denominado como bucket skew: múltiplos registos com o mesmo valor da chave de pesquisa; a função de hash escolhida, resulta numa distribuição não uniforme dos valores da chave de pesquisa
3. Solução: Cadeias de overflow (os buckets de overflow são associados através de uma lista. É denominado closed Hashing).

Indices Hash

Definições

Um índice hash, organiza os valores da chave de pesquisa, com os respetivos apontadores para os registos, num ficheiro com uma estrutura de hash.

Indices hash, são sempre índices secundários.

Eficientes em consultas de igualdade.

Deficiencias

A função de hash associa os valores na chave a um conjunto fixo de buckets:

1. Se o numero de buckets é reduzido, o aumento da BD, provoca bucket overflow, logo diminuição do desempenho.
2. Se o numero de buckets é elevado (prevendo o aumento da BD), temos desperdício de espaço inicialmente.
3. Se a BD diminui, temos desperdício de espaço (devio ao bucket overflow)

Solução: **Hash dinâmico**

Hash Dinamico

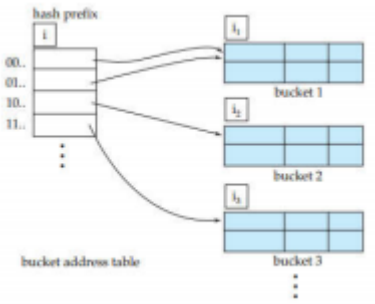
Caracteristicas

Eficientes para bases de dados que aumentam e diminuem de dimensão.

Permite a modificação dinâmica da função de hash

Hash extensível:

1. A função de hash gera valores dentro de um intervalo alargado.
2. Utiliza a cada momento um prefixo para endereçar um conjunto de buckets.
3. Sendo i o comptimento do prefixo e 0 <= i <= 32.
4. Numero máximo de buckets: 2^32.
5. O valor de i varia com a dimensão da base de dados.



Comparação

Hash extensível vs outros métodos

Vantagens:

1. O desempenho não se degrada com o aumento da dimensão do ficheiro;
2. Overhead de espaço mínimo.

Desvantagens:

1. Nivel extra para obter o registo procurado;
2. A tabela de endereços de buckets, pode tornar-se muito grande (não caber em memoria);
3. Alterar a dimensão da tabela de endereços de buckets, é uma operação dispendiosa.

Tipos de índices MS SQL

Clustered

- Ordenado (agrupado)

- "armazena" a informação na estrutura de índice

Non-clustered (apenas "aponta" para a informação.

Uma tabela apenas pode contar um índice clustered e até 999 non-clustered.

Uma tabela sem um índice clustered é denominada de head.

Cover vs Composite Index

Composite: Indexa um conjunto de colunas (interessa a ordem)

Cover: Indica num Indice non-clustered, informação adicional para ser armazenada juntamente no índice. (não interessa)

Filtered índex

Exclui do índice um conjunto de registo de acordo com um critério

- Tem o potencial de melhorar a performance, considerando o numero de entradas e operações de atualização.

SINTESE

1. Para tabelas frequentemente utilizadas utilizar poucas colunas indexadas;
2. Numa tabela com muitos dados, mas taxa de utilização baixa, é recomendada a utilização de vários índices de acordo com o tipo de queries previsto;
3. Colunas indexadas em modo clustered índex devem ter valores de pequena dimensão, não nulos e preferencialmente/tendencialmente únicos;
4. Em regra, quantos mais valores duplicados existirem na coluna indexada pior é a performance da indexação;
5. Em índices compostos, interessa a ordem das colunas, colunas cujo os valores serão testados na clausula WHERE das queries devem ser colocados em primeiro no índex;
6. É muitas vezes desnecessário indexar tabelas de reduzida dimensão

7. Em geral, quando não é a chave de organização dos dados deve ainda assim indexar-se a chave primária;
8. É normalmente pertinente indexar chaves estrangeiras;
9. Evitar indexação de atributos que são frequentemente alvo de atualizações;
10. Evitar indexar atributos cujo o resultado da querie traga mais de 25% dos registos da relação;
11. Evitar indexar atributos do tipo char de grande comprimento;
12. Deve-se fazer ensaio sobre se o índice, traz melhorias desempenho, seestas são significativas ou se estão a degradar desempenho noutras queries ou operações;
13. Pode ser útil em operações de carregamento ou manutenção desativar os índices;
14. Alguns SGBD requerem uma atualização explicita das estatísticas no catalogo para que os planos de execução passem a considerar a presença de um novo índice
15. Documentar as escolhas de índices.

PROCESSAMENTO E OTIMIZAÇÃO, MONITORIZAÇÃO

Parser

Para o inicio do processamento "efetivo" o SGBD tem de traduzir/representar a query para uma forma "acionável" ao nível de "sistema"

- O SQL é para humanos!

- O formato suporta-se na Algebra relacional e suas equivalências

O **parser** realiza duas tarefas principais:

1. A validação sintatica (garantindo que a query está "conforme" com as relações da BD a que se refere – utilizando o catalogo)
   Traduz a query para uma arvore que representa as "sub-operações" que terão de ser realizadas » **Plano de execução**

Planos de execução

Diferentes planos de execução têm associados diferentes "**custos de execução**"

Não é da responsabilidade de quem elabora a query, estrutura-la de forma que reduza o seu "custo de execução"

Será o query optimizer que gerará planos alternativos de execução e avaliará para cada o seu custo de execução afim de selecionar o que efetivamente será executado, considerando: as propriedades dos operadores da álgebra relacional e as heurísticas.

A geração das alternativas

A transformação da consulta inicial em alternativas na forma canónica SPJ(seleção-projeção-junção) dos operadores da álgebra relacional, envolve:

1. Formar uma lista dos produtos das relações;
2. Aplicar a essa lista o predicado da clausula WHERE
3. Aplicar os agrupamentos
4. Aplicar à essa lista o predicado da clausula HAVING
5. Aplicar as projeções

6. Aplicar as ordenações

As regras de equivalência a álgebra relacional podem permitir grandes reduções no numero de tuplos a processar

Tambem as heurísticas permitem ganhos de performance na execução, eg:

1. Reorganizar as seleções para que as mais restritivas sejam executadas primeiro
2. Descer projeções o mais possível

Avaliação-Estimação do custo

A avaliação de alternativas logicas de planos de execução poderá ponderar diversos fatores, incluindo: Acessos ao disco; Tempo de CPU; E no caso de BDs distribuídas o custo da comunicação

Deverão ser considerados os custos (físicos) associados a cada "sub-operação" e combinados num custo total da query.

Tipicamente a grande fatia do custo a ser ponderada (e mais "facilmente" estimada) prende-se com as necessidades das operações relativamente ao acesso ao disco.

Contudo as estimativas são de fiabilidade questionável porque:

1. A resposta dependerá dos conteúdos e ocupação do memory buffer;
2. Com vários discos dependerá de como os dados estão distribuídos.

Na prática, é impossível determinar um custo exato, da mesma forma que não é possível gerar em tempo útil todas as alternativas possíveis de processamento de uma dada consulta.

O custo calculado pelo SGBD não é o custo real da execução, mas um valor que permite comparar planos alternativos entre si.

Outras operações que não a seleção têm cálculos mais complexos!

Ainda, terá de haver lugar a avaliação de expressões para lá de operações individuais, considerando a modalidade:

1. Materialization: persistência de resultados temporários em disco se excedem buffer.
2. Pipelining: modalidade que vai disponibilizando às operações ascendentes resultados à medida que se disponha destes.

Estatisticas sobre os custos

O catalogo da BD guarda informação de cada relação:

1. NR ou [R]: numero de tuplos da relação R;
2. BR: numero de blocos com tuplos de R;
3. SR: tamanho de um tuplo de R em bytes;
4. FR: numero de tuplos de R que cabem num bloco;
5. VD: numero de valores distintos da coluna C. Se C tiver valores únicos, por exemplo se for uma chave candidata, VD(C) = NR. Alguns SGBD armazenam também a densidade da coluna C: 1/VD(C);
6. Percentagem de valores nulos;
7. Min(C): valor mínimo da coluna C;
8. Max(C): valor máximo da coluna C;

9. A informação guardada sobre os índices.

As estatísticas são muitas vezes calculadas com base numa amostra dos dados.

As estatísticas não estão sempre a ser atualizadas

A informação estatística sobre as relações é atualizada com periodicidade controlada pelo DBA

As atualizações podem ser despoletadas “manualmente” ou periodicamente e/ou associadas a eventos.

Plano Fisico

O plano físico contem o acoplamento entre os operadores relacionais e os operadores físicos escolhidos para implementar.

Um plano físico especifica como a consulta vai ser executada, sendo que a sua construção parte do plano lógico anterior, adicionando os operadores físicos mais adequados a cada operação logica, juntamente com os respetivos custos associados.

Geralmente, um plano lógico pode originar vários planos físicos.

Os operadores físicos implementam as operações da álgebra relacional

O mesmo operador relacional pode ser implementado por mais do que um operador físico e o mesmo operador físico pode implementar mais do que um operador relacional

Alguns operadores físicos e estratégias de implementação

Projeção:

1. Com ordenação
2. Com hashing

Junção:

1. Junção de blocos em ciclo;
2. Junção com ordenação e fusão;
3. Junção com hashing.

Seleção:

1. Varrimento sequencial da relação numa coluna não ordenada;
2. Varrimento sequencial da relação numa coluna ordenada;
3. Utilização de índices;
4. Utilização de hashing.

Observações:

Os planos de execução gerados são guardados:

1. Para poderem serem reutilizados;
2. Uma vez que uma recompilação tem um custo associado.

Contudo,

1. A reutilização de um plano nem sempre é a melhor opção;
2. Dependerá da atual distribuição de dados na relação;

3. E eventuais alterações à metadata.

Monitorização

Objetivo

A monitorização continua/reglar permite intervenção pro-ativa:

1. Por oposição a medidas relativas menos detalhadamente pensadas/dimensionadas,

Existem várias ferramentas builted-in and third party que apoiam a monitorização:

1. Performance Monitor (Perform), Activity Monitor, SS Profiler, Task Manager

Será necessário identificar a baseline, e.g. relativa a:

1. CPU
2. Memory
3. Physical Disk IO
4. SQL server activity such as: buffer manager statistics
5. SQL statistics, sucj as: Index Usage Statistics, Missing Indexes, Plan Cache statistics, Query Statistics and Plans,…

Selecionar modo de recolha; persistência e visualização dos dados da baseline.

Avaliar tempos de execução e planos de execução de queries.

Avaliar taxa de crescimento:

1. Dados
2. Utilizadores

Estabelecer plano de monitorização com indicadores de:

1. Atividade em disco
2. Utilização do processador
3. Memoria

Observações

A avaliação das medidas face às métricas deve ser efetuada de forma conjunta sob risco de se tirarem conclusões erradas quanto às possíveis causas de um indicador em baixa performance.

TRANSAÇÕES E CONCORRENCIA

Contexto

Transações

As operações realizadas pelo SGBD são agrupadas segundo uma unidade de divisão de trabalho a **Transação**

Concorrencia

Considerando que:

1. O SGBD é um multi-utilizador

2. Necessita de tempos de resposta reduzidos

A sua disponibilidade e performance é conseguida à custa de uma execução intercalada das transações (multi-tasking)

Operações

De forma abstrata e simplificada, uma transação **Ti** pode ser vista como constituída por um conjunto de operações de leitura e escrita de dados

Ler(x) transfere o dado x da base de dados para a memoria alocada ao SGBD ficando disponível à transação que solicitou a operação de leitura (read).

Escrever(x) transfere o dado x de memoria para os ficheiros da base de dados em disco, cumprindo a solicitação da escrita (write) efetuada pela transação.

Estados

O estado de uma transação é sempre monitorizado pelo SGBD

1. Que operações foram efetuadas?
2. As operações foram concluídas? Devem ser desfeitas?

Estados de uma transação:

1. Ativa;
2. Em efetivação;
3. Efetivada;
4. A desfazer;
5. Concluida.

Propriedades [ACID]

Atomicidade (Tudo ou nada)

Consistencia (uma transação conduz sempre a BD de um estado consistente para outro estado consistente)

Isolamento (num contexto de transações concorrentes, a execução de uma transação Ti deve acontecer como se Ti se executasse isoladamente e não sofrer interferências de outras transações executando concorrentemente

1. A execução simultânea das operações de duas transações distintas (escalonamento) resulta numa situação equivalente ao estado obtido se uma das transações tivesse sido executada após a outra (serialização), independentemente da ordem.

Durabilidade (é garantido que as alterações efetuadas por uma transação, que concluiu com sucesso, persistem na BD).

Delimitação de Transações (T-SQL)

A DML possui instruções para delimitar o conjunto de operações que constituirão uma transação.

Por omissão os “comandos individuais” são transações (com autocommit)

Em T-SQL a definição de transações (explicitas), suporta-se em:

1. BEGIN TRANSACTION (inicia uma transação)
2. COMMIT TRANSACTION (confirma as operações de uma transação)
3. ROLLBACK TRANSACTION (desfaz as operações de uma transação)

Controlo de concorrência

Suportando o SGBD diversos utilizadores com acesso simultâneo aos dados, serão inevitavelmente geradas múltiplas transações concorrentes.

Como é garantido o isolamento das transações?

1. Solução ineficiente: executar uma transação (completa) de cada vez! (escalonamento serializado de transações)
2. Solução melhor: Execução concorrente de transações preservando o isolamento, através de um escalonamento (Schedule) (mas assegurando a integridade dos dados)

A execução concorrente permite um maior aproveitamento de recursos e melhoria dos tempos de espera face a transações (longas) serializadas

Na ausência de escalonamento integral

Efeitos na correção dos dados originados por escalonamentos sem isolamento integral:

1. Atualização perdida (lost-update)
2. Leitura suja (dirty-read)
3. Analise inconsistente (nonrepeatable read)
4. Leitura fantasma (phantom read)

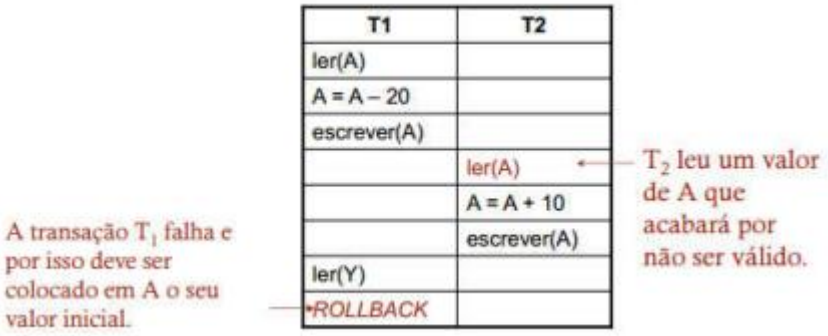
Lost-update -> a transação T2 sobrescreve um registo atualizado anteriormente pela transação T1

T1	T2
ler(A)	
A = A – 50	
	ler(C)
	A = C + 10
escrever(A)	
ler(B)	
	escrever(A) ← a atualização de A por T_1 foi perdida!
B = A + 30	
escrever(B)	

Dirty-read

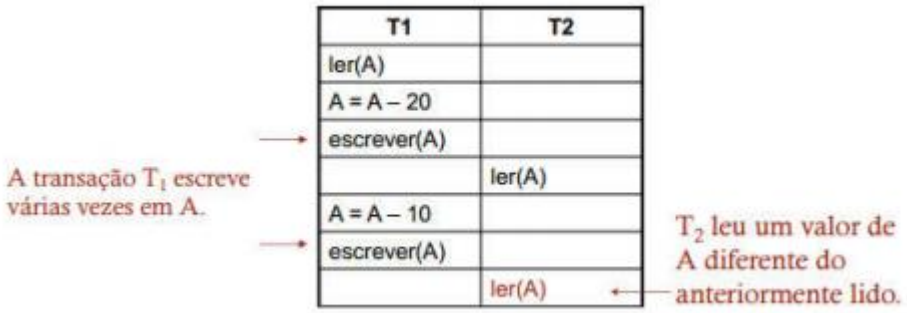
T1 atualiza o registo A, acedido posteriormente por outra transação T2.

T2 leu um valor incorreto de A pois T1 falhou.



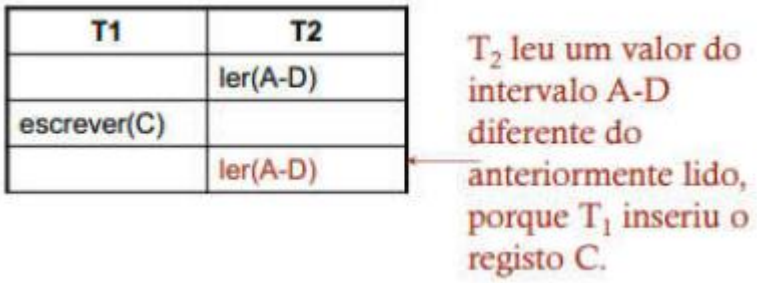
Nonrepeatable-read

T1 atualiza um registo A varias vezes. Após cada alteração o registo é acedido por outra transação T2 que não obtém um valor idêntico de A



Phantom-read

T1 insere/apaga registos num intervalo de A-D. Após a interseção/remoção de registos, o intervalo é acedido por outra transação T2 que não obtem um valor idêntico para o intervalo A-D.



MS SQL

O MS SQLServer suporta os níveis de isolamento do standard SQL através da instrução:
SET TRANSACTION ISOLATION LEVEL nível
nível pode ser:

1. SERIALIZABLE: Ti executa com completo isolamento;
2. REPEATABLE READ (default): Ti só lê dados efetivados mas outras transações podem criar dados no intervalo lido por Ti.
3. READ COMMITTED: Ti só lê dados efetivados, mas outras transações podem escrever em dados lidos por Ti
4. READ UNCOMMITTED: Ti pode ler dados que ainda não sofreram efetivação

Efeitos secundários admitidos por nível de isolamento.

Nível de Isolamento	Dirty read	Non-repeatable read	Phantom
Read uncommitted	√	√	√
Read committed		√	√
Repeatable read			√
Serializable			

Implementação do controlo de concorrência

Mecanismos:

1. Locking – bloqueio temporário aos dados em causa (hierarquia e tipos de locks)
2. Timestamps – read and write transactions timestamps on each data item
3. Multiple versions (snapshot isolation) – possibilidade de permitir acesso a uma copia temporária de dados (overhead) que ainda não tenham sofrido um commit que imponha uma alteração como definitiva (efeito repeatable read mas sem lock/block)

BACKUP & RESTORE

Objetivos

Minimizar a possibilidade de perda de dados devido a falhas no hardware, software aplicacional ou devido a acidentes naturais.

Trata-se de uma operação que copia a informação do sistema para um dispositivo de Backup

No MS SS pode ser realizado em qualquer momento por recurso ao management studio e ao comando backup.

A realização de backups com utilizadores acedendo à base de dados, causa quebra de desempenho

Fatores a considerar

Na seleção da politica de backups a ser seguida dever-se-á ponderar:

1. Quantidade de perda de dados admissível
2. Natureza da base de dados (data warehouse ou OLTP)
3. Numero de alterações que sofre a base de dados
4. Tempo de recuperação aceitável
5. Existe tempo para manutenção da base de dados
6. Dimensão da base de dados.

Tipos de backup

Completo (full):

1. Captura toda a informação da base de dados numa única operação
2. Pode ser realizado com a base de dados online
3. Regista o log sequence number (LSN), quando o backup começa e quando está finalizado

LSN – numero sequencial que identifica a cobertura das alterações na base de dados salvaguardadas, e pode ser utilizado para recuperar a base de dados a partir de um dado momento temporal

Diferencial (differential):

1. Captura apenas as alterações efetuadas na base de dados, desde o ultimo backup completo (diferencial base)
2. As alterações estão identificadas por uma flag e apenas estas são escritas no backup.
3. Otimoza o espaço necessário para backup
4. Otimiza o tempo necessário para backup e recuperação

Necessita do ultimo backup completo para recuperação.

Parcial (partial):

1. Em algumas aplicações existe informação que não sofre alterações frequentes ou são somente para consulta.
2. Est informação deve ser colocada em filegroups read-only
3. Este tipo de backup apenas guarda a informação que não faz parte dos filegroups read-only
4. Reduz o espaço necessário e tempo do backup

Diferencial e parcial (diferential partial):

1. Idêntico ao diferencial, mas apenas para os filegroups que não são read-only

De cópia (copy-only):

1. Identico ao completo ma não altera a informação que indica se os dados estão já salvaguardados, não interfere com politica de backup definida
2. Utilizado na copia de bases de dados

Registo das transações (transaction log):

1. Guarda a informação dos ficheiros utilizados para registo de transações
2. É apenas suportado nos modelos de recuperação, full e bulk

Recuperação

Introdução

Operações a executar quando existe uma falha (hardware ou software) do servidor
Uma base de dados tem associado um modelo de recuperação, que determina como as transações são escritas nos ficheiros de log

Modelos:

Completo (full recovery):

1. Proteção mais elevada contra a perda de informação, todas as alterações são escritas no ficheiro de logs de transações.
2. As alterações incluem os comandos insert, delete e update, assim como os comandos que alteram a informação na base de dados.
3. É o mais dispencioso em es+aço necessário para o ficheiro de logs, e desempenho pois, cada transação é guardada no ficheiro de log

Bulk logged:

1. Identico ao full, difere apenas na forma de como as operações de BULK são capturadas no ficheiro de transações (e reflexo no seu backup): Em full, cada transação é escrita no ficheiro de transações; Este modo apenas guarda as "extensões" das paginas que foram alteradas
2. Diminui o espaço do ficheiro de transações
3. Pode não ser possível a recuperação para um "ponto" temporal recente.

Simples:

1. Modelo de administração mais simplificada
2. Maior possibilidade de perda de informação
3. O registo de transações é "truncado" automaticamente com base no processo de checkpoint da base de dados
4. Não necessária recuperação para determinado ponto de falha
5. Não existe recuperação através dos ficheiros de logs de transações
6. Não é recomendável em bases de dados de produção

Dispositivos de backup (devices)

Dispositivos onde são armazenados os ficheiros de backup da base de dados.
Podem ser físicos ou lógicos ("alias" para o dispositivo físico)

Tipos:

1. Tapes
2. Disco
3. Network: SAN (Storage Area Network); NAS (Network Attached Storage)

Base de dados aplicacional (utilizadores):

1. Após a alteração da sua estrutura
2. Definir um plano de backup, para execução de backups periódicos

Base de dados de sistema:

1. Master: sempre que existem alterações nas bases de dados dos utilizadores;
2. MSDB: sempre que existem alterações nos jobs, backups,…
3. Tambem devem fazer parte do plano de backup.

Evitar durante o processo:

1. Criação e modificação da estrutura da base de dados

2. Criação de índices
3. Redução do tamanho da base de dados

Cenários

Transaction log backup:

1. Guarda as alterações desde a ultima operação de backup log, até às ultimas entradas do log atual
2. Trunca o ficheiro de log
3. Opção NO_TRINCATE: pertinente quando BD está danificada
4. Opção NORECOVERY: as ultimas transações (correntes) que ainda não foram sujeitas a backup – tail do log – são guardadas; deixa a base de dados no estado de RESTORING

Full Database Backup:

1. A utilização deste tipo de backup sem outro adicional, só é usado em sistemas em que a perda de transações não é importante.
2. Exemplos de utilização: Backup completo à noite; Administração facilitada e menos espaço para backup; Problema da possível perda das transações que ocorreram durante o dia

Full Database com Transaction log backup:

1. Aumenta a possibilidade de recuperação de informação
2. Os backups do log de transações são efetuados periodicamente, de modo a capturar a atividade incremental da base de dados
3. Dimensionar a frequência dos backups

Differencial backups:

1. São utilizados para reduzir o tempo necessário para a realização de backup e de recuperação
2. Uteis em sistemas sujeitos a poucas transações
3. São cumulativos, isto é, contem todas as alterações desde o ultimo backup completo
4. Para recuperar a base de dados apenas é necessário o backup completo e o ultimo diferencial.

Processo

Conjunto de ações a executar pelo DBA no SGBD, de modo a assegurar a rápida recuperação da operacionalidade da base de dados.
O comando RESTORE DATABASE, que verifica se:

1. A base de dados existe
2. Os ficheiros associados ao comando são idênticos aos existentes no backup
3. Existem todos os ficheiros necessários à operação de restore

Particularidades do processo:

1. Não é necessário efetuar o drop da base de dados a recuperar
2. Os ficheiros da base de dados são automaticamente recuperados

SEGURANÇA

Motivação

Proteger … Isto é manter operacional e de acesso controlado:

1. Dados
2. Objetos
3. Regras/Logica
4. BD(s)
5. SGBD

Varias dimensões da segurança:

1. Segurança física
2. Segurança da rede
3. Modalidades de autenticação
4. Esquemas de autorização
5. Opções na definição e atribuição de (tipo de) contas
6. Auditoria/monitorização regular e pró-ativa
7. Politica de backups
8. Validação de dados submetidos pelos "utilizadores"
9. Ponderação das alternativas exequíveis de encriptação
10. Desativação de serviços desnecessários
11. Eventual alteração de pontos default

Autenticação

Processo de verificação de que um utilizador é genuíno
Modos de autenticação MSSS (Login):

1. Windows – Modo integrado com o SO, as contas e grupos do SO são confiáveis na autenticação perante o MSSS
2. Misto (mixed) – Modo Windows + Credenciais (utilizador + senha) definidos e armazenados no MSSS

Autorização

Processo que especifica os direitos de acesso de determinado utilizador/grupo relativamente aos objetos da BD
Privileges & roles – São especificados os privilégios sobre os objetos (sistema + específicos) da BD

Logins (autenticação – acesso ao servidor) vs Users (autorização – operações na BD)

1. Faculta o acesso ao MSSS
2. Podem estar associados a contas Windows ou MSSS
3. São armazenados na BD master

Roles

Modelo flexível de administração de segurança

Conjunto agregado de privilégios

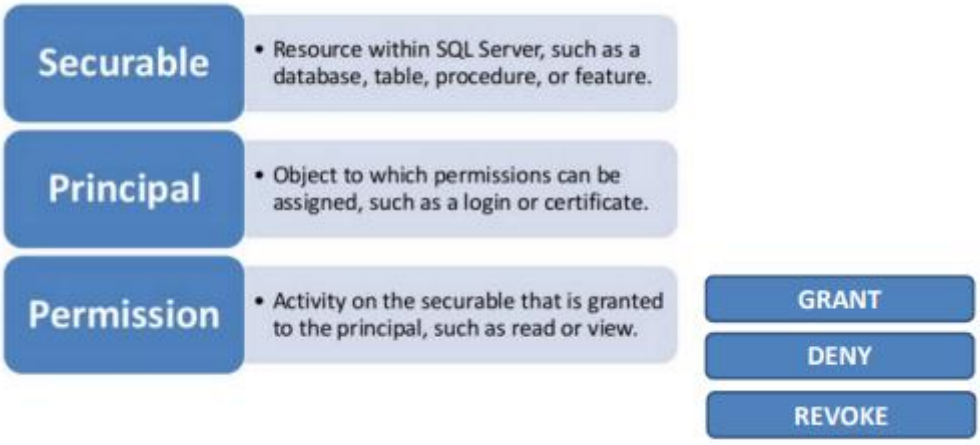
Podem ser geridos/alterados “dinamicamente”

Preferível a atribuição de permissões/priviliegios a Roles do que a users

Disponiveis no MSSS:

1. Server Roles
2. Custom Server Roles
3. Database roles
4. Custom Database roles -> mais granular, comum e recomendado

<u>Vocabulário</u>



GRANT:

1. Confere permissão ao Principal sobre o Securable em determinada ação

DENY:

1. Nega determinada ação ao Principal sobre o Securable
2. Se houver conflito com a existência de GRANT e DENY simultaneamente, o DENY sobrepõe-se sempre

REVOKE:

1. Repõe no estado anterior à ultima permissão concebida, seja esta GRANT ou DENY

Hierarquia de Securables:

1. Instance
2. Database
3. Schema
4. Database Objects (Tables, views, SP’s, functions…)

Encriptação

Uma forma adicional de proteger dados

Adequado para informação particularmente sensível (email, password, nº cartão de credito)

Pode ser considerada uma hierarquia:

1. Nivel SO
2. Nivel SGBD
3. Nivel BD

E adicionalmente considerar também a encriptação na transmissão dos dados

Modos de encriptação

Chaves simétricas, o emissor e recetor partilha uma única e comum chave usada para encriptar e desencriptar os dados, mais simples e performante mas, a chave tem de ser transmitida.

Chaves Assimétricas utilizam-se duas chaves uma para encriptar e outra para desencriptar
O par é gerado conjuntamente e é contituido por: uma chave public-key que será distribuída, uma chave private- key
O emissor utiliza a chave publica para realizar a encriptação o recetor utiliza a sua chave privada correspondente para desencriptar (assim a chave publica não permite a desencriptação

MS SQL

Encriptação ao nível da BD
Possibilita a encriptação da totalidade dos dados (Ler e Escrever precisa do processo de desencriptação/encriptação)
degradação da performance

Encriptação ao nível das colunas
Somente colunas que persistem dados sensíveis são encriptadas

Melhor performance, mas atenção se as colunas são chaves, ou alvo das clausulas WHERE ou JOIN ON de queries; degradará a performance dessas queries

Mais comumente utilizado

REPLICAÇÃO

Motivação

Numa arquitetura completamente centralizada, se

1. Muitos acessos
2. Muitos dados persistidos
3. Muitos dados trocados

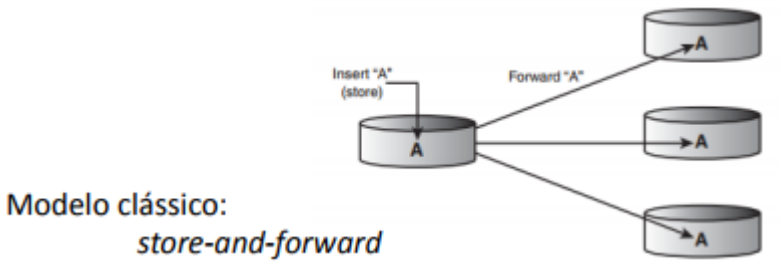
Podem ter problemas de:

1. Performance
2. Disponibilidade
3. Manutenção

Uma das opções a considerar passa pelo recurso à Replicação

Objetivos

A replicação permite manter múltiplas copias de partes de uma BD em diferentes/múltiplos locais.

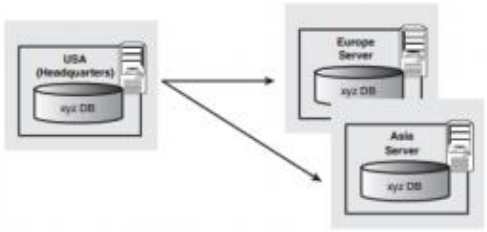


Modelo clássico:
store-and-forward

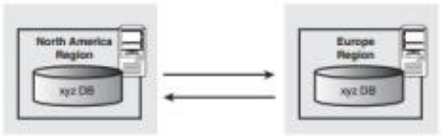
Cenários e.g.

isolate reporting

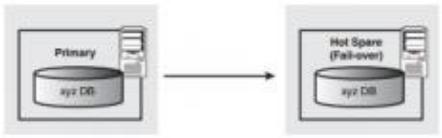
partinioning by region



bidiretional



faillover server



Metáfora Publisher - Subscriber

Distributor: Servidor de BD responsável pela coordenação da operação de replicação

1. Armazena a base de dados distribution (store-and-forward)
   (Contem a metadata, dados históricos e transações).
2. Tem um papel mais relevante nas replicações do tipo snapshot e transacional

Publisher: Servidor de BD responsável pela manutenção da informação na BS de origem e envio da informação alterada, para o **Distributor**.

Subscriber: servidor de BD responsável pela manutenção da copia da informação pela receção da informação oriunda do **Distributor**

Informação a Replicar

Artigo:

1. Dados(toda ou, parte da informação, de uma tabela) ou
2. Objetos da BD

Publicação: conjunto de um ou mais artigos, que contitui uma unidade de subscrição

As réplicas podem ser referentes a 1 ou mais publicações

Tipos de Replicação:

1. Snapshot
2. Transacional
3. Merge

Um tipo de replicação aplica-se a uma publicação, assim podem coexistir carios tipos de replicação na mesma BD

Snapshot

Quando ocorre o momento da sincronização, é gerada e enviada aos subscribers uma fotografia dos dados, Isto é, são replicados os dados exatamente como se encontram em determinado momento, sem monitorização das alterações.

Quando utilizar:

1. Com alterações substanciais mas pouco frequentes dos dados
2. Os subscribers necessitam de acesso a dados apenas em leitura
3. Existe um grande desfasamento nos períodos de atualização dos dados

Transacional

Quando as alterações incrementais na base de dados de origem são replicadas na base de dados de destino através do Distributor, minimizando o período de desfasamento da informação entre bases de dados, Os dados alterados nas bases de dados envolvidas, são atualizados simultaneamente, isto é, no âmbito da mesma transação

Quando utilizar:

1. Os subscribers devem receber as alterações com a maior brevidade possível;
2. Maioritariamente em server-to-server

Disponivel também numa versão de updatable subscriber

Permite atualizações de dados nos subscrivers (contudo pouco frequentes) com envio para o publisher; 2 modos: Immediate (baseado em triggers) Queued (alterações enviadas para uma queue gerida pelo quele reader agente)

Merge

Caso em que é permitido que as bases de dados – subscriber – efetuem alterações autonomamente, procedendo-se posteriormente (periodicamente ou a pedido) à reconciliação das alterações.

Quando utilizar:

1. Cenários Server-to-Client
2. Os subscribers necessitam de efectuar alterações e propaga-las no Publisher e nos outros Subscribers;
3. Os subscribers necessitam de receber dados efectia alterações offline e sincronizá-las no Publisher e nos outros Subscribers;
4. Subscribers trabalham com uma parte da publicação e não com a publicação total.

Formas de subscrição

Push subscription:
A subscrição é definida para um ou vários Subscribers, no Publisher aquando da criação/alteração da Publicação.
As alterações são assim enviadas para os Subscribers logo que ocorrem no Publisher.

Pull subscription:
A subscrição é definida e iniciada no Subscriber, para as Publicações disponibilizadas pelo Publisher (acessíveis aos Subscirbers que se registem para o efeito)

Fatores a considerar

Na seleção do tipo de replicação deve atender-se, a:

1. Grau de desfasamento aceitável
2. Grau de autonomia de Base de Dados
3. Nivel de Consistencia Transacional

Agentes

O MS SS utiliza diversos agentes para conduzir as tarefas associadas ao processo de replicação

Snapshot: prepara o esquema da base de dados, as tabelas publicadas e as stored procedures necessárias à replicação, armazenando essa informação na base de dados distribution; corre no Distributor

Distrivution: utilizado na replicação Snapshot e Transacional, por forma a implementar a definição da replicação estabelecida na base de dados distribution no Distributor; corre no Distributor (push subscription) e nos subscribers (pull subscription)

Log Reader: Utilizado na replicação Transacional

1. Copia todas as transações marcadas para replicação, do log de transações do publisher, para a base de dados distribution no Distributor, para que este as possa distribuir pelos subscribers
2. Corre em cada uma das bases de dados.

Merge: utilizado na replicação do tipo Merge

1. Aplica um snapshot inicial ao subscriber para, com esta base, proceder futuramente à reconciliação das alterações que venham a ocorrer, do subscriber para o publisher e vice-versa.
2. Corre no Distributor (Push Subscription) e nos Subscribers (Pull Subscription);

Queue Reader: utilizado na replicação Transacional

1. Responsavel pelo tratamento das mensagens em fila e sua aplicação às respetivas publicações.
2. Corre no Distributor.

MONGO DB

Base de dados não relacionais

Graph:

1. Alta performance
2. Dados com diferentes níveis de relação

Ex: neo4J, OrientDB, MongoDB

Key-value:

1. Alta performance
2. Base de dados de memória

ex: redis, mongoDB

Base de dados orientada a documentos (BDOD)

NoSQL

1. Not only SQL
2. Na verdade deveria chamar-se NoREL

Alternativa ao modelo relacional

Base de dados adaptada ao utilizador final

Redundancia e dados não “normalizados”

BDOD utilizam o conceito de dados e documentos autocontido e auto descritivos, isso implica que o documento em si já define como este deve ser apresentado e o signidicado dos dados em cuja estrutura estão armazenados.

Ex: mongoDB, CouchDB, RavenDB

Quando utilizar?

1. Acesso a grandes volumes de dados
2. Dados apresentam uma grande variedade na sua estrutura
3. Flexibilidade
4. Escalabilidade
5. Performance
6. Modelo relacional perde desempenho neste tipo de ambiente

JSON

JavaScript Object Notation

Facil leitura e escrita

Facilita a comunicação entre máquinas

Tamanho reduzido quando comparado com outros (XML)

Lista ordenada de valores (key/value)

Baseada no JavaScript

MongoDB

Cirada em 2007

A mais utilizada na comunidade, com versões estáveis

Escrita em C++

Multiplataforma

Adotada por grandes empresas como: BOSCH, AstraZeneca, MetLife, Facebook Urban Outfitters, Sprinklr, theguardian

Analogia ao SQL

MongoDB	SQL
JSON (key => value)	*Colunas*
Coleção	Tabela
Índice	Índice
Agregação (Pipeline e *Map-Reduce*)	Group
Embedding e Linking	Join

Algumas características

UPSERT: Possibilidade de acrescentar um registo por meio de um update

A coleção não obdece a um Schema, originando coleções dinâmicas (mas também podemos forçar um schema, se for esse o objetivo)

Possibilidade de acrescentar uma coleção por processo de um insert (no caso da coleção não existir, ela é criada automaticamente)

Criação de base de dados (use nome_bd)

Escalabilidade

Sharding:

1. Distribui os dados por diferentes maquinas
2. Util para bases de dados grandes
3. E para operações de alto rendimento

Replica set:

1. Grupo de bases de dados que contem os mesmos dados
2. Redundância e alta disponibilidade dos dados
3. Separação entre o servidor primário e N secundários
4. Primários para escrita
5. Secundários para leitura
6. Solução para casos de falha em servidores de produção

Performance

Para 2 servidores com características iguais

Inserir 10.000 registos (mongoDB -> 2.032 segundos; MSSQL -> 204.215 segundos)

Outras aplicações

Video HD

Geolocalização

Mensagens em tempo real

Realidade aumentada

Streaming HD

Vantagens:

1. Flexibilidade
2. Disponibilidade
3. Performance
4. Linguagem nativa (JavaScript)
5. Manutenção quase inexistente (dispensa DBA)
    a. Reutilização de espaço;
    b. Mecanismos de re-index automáticos.

Desvantagens:

1. Perda de consistência e de relações

REDIS

<u>O que é?</u>

Oferecer o maio throughput possível:

1. Efetuar milhões de operações/segundo
2. Na menor latência possível (<1ms)
3. E com o mínimo de recursos

Base de dados Key-value

1. Alta performance
2. Base de dados de memoria

Open source (BSD licensed)

Estrutura em memoria:

1. Escrito em C
2. Otimizado para complexidade O(1)

Estrutura de dados:

1. Strings, hashes…
2. Lists, sets…
3. Bitmaps
4. Geospacial

<u>Quando utilizar?</u>

BD de alta performance:

1. Para escrita;
2. E para leitura

É uma das top 3 BD NOSQL mais utilizadas

Facil manutenção e instalação

Escalável

Drivers disponíveis para todas as linguagens mais utilizadas

Escalabilidade

Diferentes níveis de persistência no disco:

1. RDB
    a. Performs point-in-time snapshots of your database at specified intervals
    b. Backups da BD em ficheiros compactados
    c. Pode existir perda de dados entre o ultimo backup e a possível falha
2. AOF
    a. Logs every write operation received by the server
3. Possibilidade de desligar a persistência
    a. If you want your data to just exist as long as the server is running
4. AOF e RDB combinados

Se podemos viver com a perda de alguns dados, então utilizar apenas RDB, caso contrário RDB + AOF

Replicação:

1. Grupo de bases de dados que contem os mesmos dados
2. Redudancia e alta disponibilidade dos dados
    a. Separação entre servidor Primario e N secundários
    b. Primários para escrita
    c. Secundários para leitura
3. Solução para casos de falha em servidores de produção

Sentinela:

1. Utilizado para garantir que o sistema funciona em caso de falha sem intervenção humana
2. Monitorização
    a. Verifica o funcionamento da MASTER e SLAVES
3. Notificação
    a. Informa o estado do sistema
    b. Bem como possiveis falhas
4. Failover automático
    a. Promove um SLAVE a MASTER
5. Fornece a configuração
    a. Disponibiliza a informação aos drivers

Performance

Inclui uma ferramenta redis-benchamrk:

1. Simula comandos enviados por N clientes em simultâneo
2. Enviado M queries no total
3. É idêntico ao Apache's ab