9. Indexação e otimização

introdução

índices estruturas de dados que oferecem uma forma alternativa de acesso aos dados, permitindo:

- \- tempo de consulta
- \+ volume de dados armazenados
- \+ tempo de inserção

é possível indexar qualquer atributo de relação, criar múltiplos índices sobre atributos distintos e ainda criar índices com vários atributos

index key nome dado aos atributos indexados

implementação com diversas estruturas de dados para diferentes objetivos

organização física dos dados
- os índices têm um valor ordenado (atributo indexado) e um ponteiro para a sua localização
  - modo não denso início de página
  - modo denso offset do tuplo na página
- são guardados em páginas

organização do SGBD
- os tuplos são distribuídos por várias páginas em disco, tendo cada uma suporte para vários tuplos, que estão geralmente distribuídos por várias páginas

tipos genéricos

single-level ordered
- estruturas de um único nível que indexam um atributo da relação
- armaz. valor indexado e localização do tuplo na relação
- os índices são ordenados, o que permite a pesquisa binária → complexidade $\log_2 n$

tipos
- primary indexa atributo chave da relação (sem repetições)
- clustered indexa um atributo que se pode repetir
  - os atributos estão agrupados, por isso ponteiro para o 1.º
- secondary indexa outros atributos (chave candidata ou ñ chave) não usados na ordenação física dos dados originais

obs: na primary e clustered, o atributo indexante é o atributo pelo qual os elementos estão ordenados

secondary com atributos que se repetem
ponteiro do atributo aponta para um bloco de ponteiros com as várias ocorrências desse atributo, que por sua vez têm os ponteiros para os tuplos da relação

multilevel index

estruturas de indexação com vários níveis, reduzindo complexidade da pesquisa para $\log_{fo} n$

fo fan out, fator pelo qual o espaço de procura é reduzido a cada etapa do algoritmo usualmente $fo > 2$

implementação através de árvores balanceadas
os nós de nível mais baixo (primeiro) são os mais próximos dos tuplos (extremidades da árvore)

árvore de ordem p cada nó terá no máximo
p-1 valores
p ponteiros para nós da árvore

árvore balanceada se a distância de qualquer folha à raiz for igual

operações de inserção e remoção são feitas por algoritmo que mantem a árvore equilibrada.

SQL

CREATE INDEX <indexName> ON <table> (<attr1>, <attr2>, ...)
criar índice

DROP INDEX <indexName>
eliminar índice

índices multiatributo
justificam-se se efetuarmos pesquisas contendo todos os atributos associados ao índice!

seleção de índices

como vimos inicialmente os índices têm pros e contras, devendo a sua criação ser um compromisso entre
- perceber se vamos ter necessidade de efetuar muitas pesquisas sobre um determinado atributo
- perceber se a relação vai ter modificações frequentes dos seus dados

fator organização física

como vimos, tal como os tuplos, os índices são guardados em páginas, havendo custos temporais no seu acesso e modificação

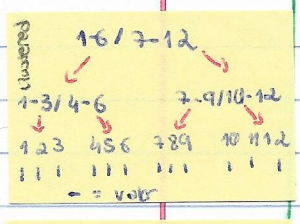
∴ índices que necessitam de carregar poucas páginas oferecem pesquisa + rápida

pesquisa de um único tuplo obriga ao carregamento em memória de toda a página onde ele se encontra!

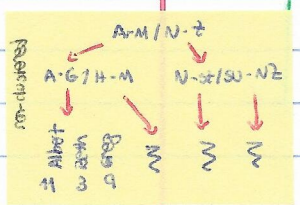
indexação das chaves da relação | pesquisamos frequentemente por atributos chave, que sendo únicos, ou existe tuplo ou não
=> só uma página é carregada para ter o tuplo

indexação de atributos não chave | • o atributo deve ter poucos valores repetidos
• a relação ter o atributo indexado do tipo clustered

SQL server | tem dois tipos de índices, ambos implementados com árvores balanceadas



clustered * ao nível folha têm os próprios dados da relação
tabela ordenada pelo índice (só um por relação)
analogia agenda de contactos telefónicos



non-clustered índices apontam para a tabela base (clustered ou non-clustered)
podem haver vários numa relação
analogia índice de um livro

* CREATE CLUSTERED INDEX <name> on <table> (...);

B-tree page split | os índices devem manter-se ordenados pelo key-index, pelo que operações de alteração dos dados obrigam a uma reorganização

quando a página está cheia o SGBD divide-la em duas
page split cria uma nova página
copia parte dos índices para esta página
reflete nova realidade nos nós superiores
insere o novo índice
muito penalizador no desempenho temporal

opções de especialização

unique | index key seja única (sem duplicados)
obs | ao criarmos uma chave primária, por defeito é criado um unique clustered index
CREATE UNIQUE CLUSTERED INDEX <name> ON <table>(..);

composite | índice com vários atributos (ordem importa)
obs | um índice só é considerado qnd primeira coluna faz parte da query, pelo que podemos ter necessidades de índices com o mesmo atributo em ordens diferentes
CREATE CLUSTERED INDEX <name> ON <table> (<attr1, attr2>);

filtered | utilização da cláusula WHERE no CREATE INDEX
só disponível para non-clustered index
`CREATE INDEX <name> ON <table>(--) WHERE <condition>;`

inclusão de atributos | podemos incluir atributos nas folhas de um índice non-clustered, chamadas query "cobertas"
(todos os dados que a query necessita estão no índice)
`CREATE INDEX <name> ON <table>(--) include(<attr>);`

---

heap vs clustered table | heap insere registos no final da tabela (não ordenado)
clustered registos inseridos na B-Tree segundo ordem do cluster index key

desempenho | depende das características da chave primária;
do facto dos novos tuplos estarem ordenados;
dos page splits

B-Tree tuning | para minimizar os Page Splits, um índice guarda espaço livre em cada página para novas entradas
fill factor % de espaço a ocupar (100 - % de espaço livre)
pad index aplicar fill factor só aos nós folha (ou não) ON/OFF

SQL | `CREATE NONCLUSTERED INDEX <name> ON <table>(-) WITH (FILLFACTOR = 85, PAD-INDEX = ON);` 15% de espaço livre

compromisso | fill de 100% para inserções ordenadas
65-85% para desordenadas

---

desfragmentação de índices | permite eliminação de espaços vazios

Consultar estado de fragmentação `sys.dm_db_index_physical_stats`

reconstrução caso haja muita fragmentação
`ALTER INDEX <name> ON <table> REORGANIZE`
desfragmenta folhas de acordo com fill factor do índice
`ALTER INDEX ALL ON <name> REBUILD WITH (FILLFACTOR = <>)`
reconstrói índice na totalidade (= DROP + CREATE)

desempenho de queries MSSQL	consultar o Execution plan Results | Messages | Execution Plan
desempenho da BD	SQL Server Profiler captura eventos • sujeitar BD a conjunto de consultas durante alguns dias • utilizar resultados na ferramenta Database Engine Tuning Advisor

Antes de fazer qualquer teste de desempenho!

DBCC FREEPROCCACHE;
remove todos os recursos na cache do plano

DBCC DROPCLEANBUFFERS;
remove todos os buffers limpos

TeO
S