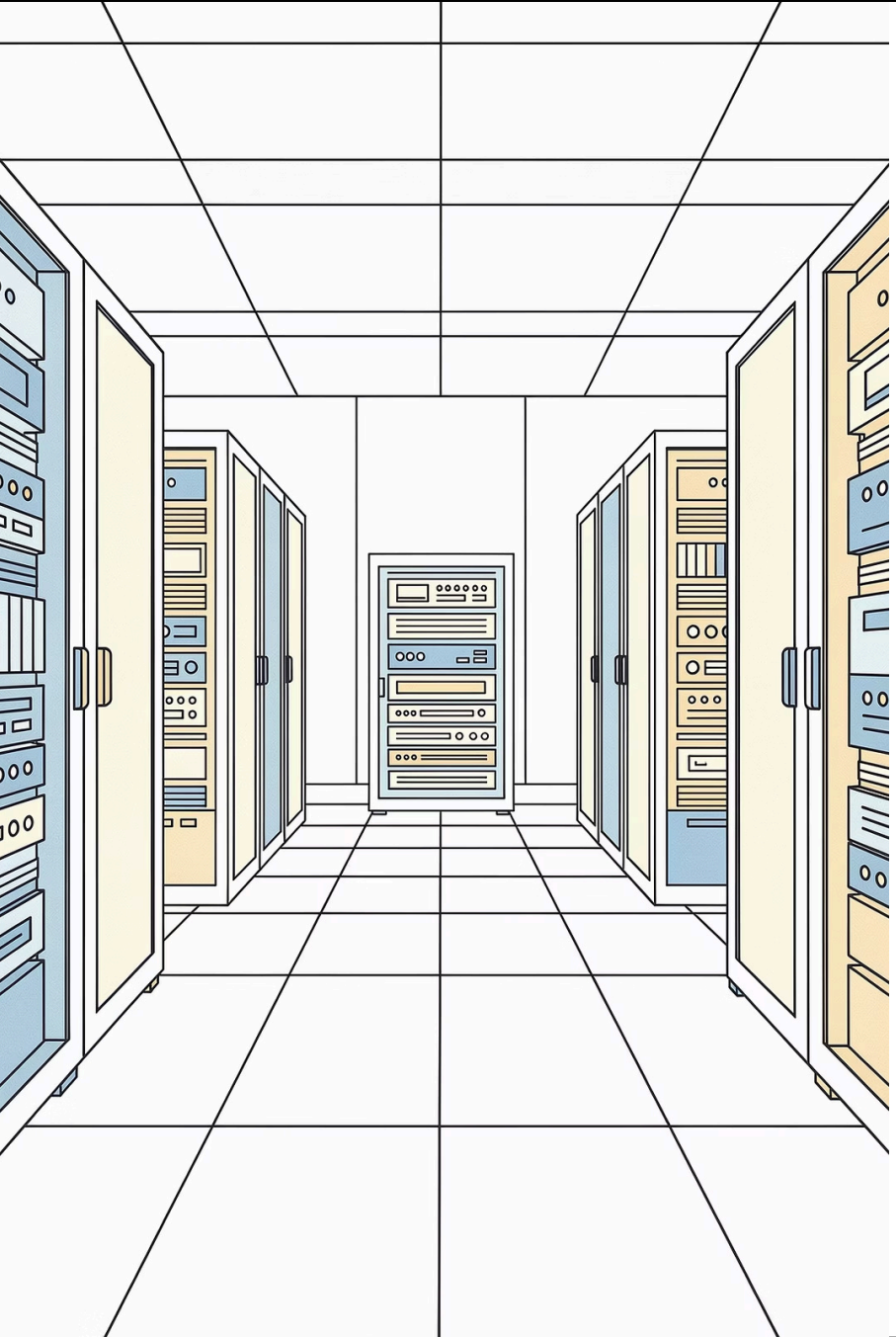


CEFET/RJ

Estruturas de Armazenamento de Dados

Exploraremos os fundamentos de como os dados são fisicamente organizados e armazenados em sistemas de banco de dados modernos.

Eduardo Ogasawara

eduardo.ogasawara@cefet-rj.br
https://eic.cefet-rj.br/~eogasawara

CONCEITOS FUNDAMENTAIS

Organização de Arquivos em Bancos de Dados

Um banco de dados é armazenado como uma coleção de **arquivos**. Cada arquivo é uma sequência de **registros**, e cada registro é composto por uma sequência de **campos**.

Abordagem Básica

A implementação mais simples assume que o tamanho do registro é fixo. Neste modelo, cada arquivo contém registros de apenas um tipo específico, e diferentes arquivos são usados para diferentes relações.

Este é o caso mais fácil de implementar, embora estruturas de comprimento variável sejam consideradas posteriormente. Uma suposição importante é que os registros são menores que um bloco de disco.

ESTRUTURA FIXA

Registros de Comprimento Fixo

Abordagem Simples

Armazena o registro *i* começando no byte *n* × *(i – 1)*, onde *n* é o tamanho de cada registro.

Acesso Direto

O acesso aos registros é simples e direto, permitindo localização rápida através de cálculos matemáticos.

record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

Desafio

Registros podem cruzar limites de blocos, causando ineficiências de I/O.

Modificação Importante: Para melhorar a eficiência, não permitimos que registros cruzem limites de blocos.

ESTRUTURA VARIÁVEL

Registros de Comprimento Variável

Registros de comprimento variável surgem em sistemas de banco de dados por várias razões importantes que refletem necessidades do mundo real.

1

Múltiplos Tipos de Registro

Armazenamento de diferentes tipos de registros em um único arquivo para otimizar o espaço.

2

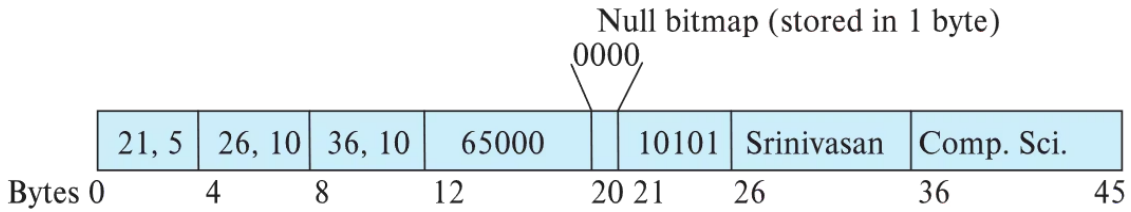
Campos de Tamanho Variável

Tipos de registro que permitem comprimentos variáveis para um ou mais campos, como strings (**varchar**).

3

Campos Repetidos

Tipos de registro que permitem campos repetidos, usados em alguns modelos de dados mais antigos.



Estratégia de Armazenamento

- Atributos armazenados em ordem sequencial
- Atributos de comprimento variável representados por (offset, comprimento) de tamanho fixo
- Dados reais armazenados após todos os atributos de comprimento fixo
- Valores nulos representados por bitmap de valores nulos

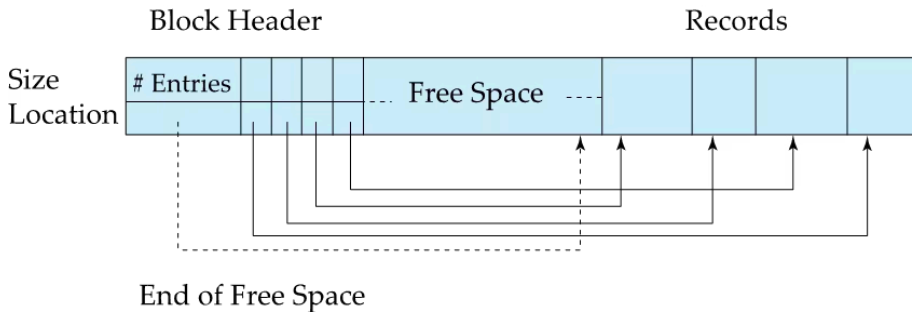
Estrutura de Página com Slots

A **página com slots** é uma estrutura elegante que permite gerenciamento eficiente de registros de comprimento variável dentro de um bloco.

Componentes do Cabeçalho

- Número de entradas de registro no bloco
- Posição do fim do espaço livre no bloco
- Localização e tamanho de cada registro individual

Os registros podem ser movidos dentro de uma página para mantê-los contíguos, sem espaço vazio entre eles. Quando isso ocorre, a entrada no cabeçalho deve ser atualizada.



Importante: Ponteiros não devem apontar diretamente para o registro. Em vez disso, devem apontar para a entrada do registro no cabeçalho, permitindo reorganização interna.

OBJETOS GRANDES

Armazenamento de Objetos Grandes

Tipos como **BLOB** (Binary Large Object) e **CLOB** (Character Large Object) apresentam desafios especiais, pois registros devem ser menores que páginas.

Sistema de Arquivos

Armazenar como arquivos no sistema de arquivos do sistema operacional

Gerenciamento pelo BD

Armazenar como arquivos gerenciados diretamente pelo banco de dados

Fragmentação

Dividir em pedaços e armazenar em múltiplas tuplas em uma relação separada

Exemplo: PostgreSQL utiliza TOAST (The Oversized-Attribute Storage Technique) para gerenciar objetos grandes de forma eficiente.

Organização de Registros em Arquivos

Existem diversas estratégias para organizar registros dentro de arquivos, cada uma com vantagens específicas dependendo dos padrões de acesso aos dados.

Heap (Monte)

Registros podem ser colocados em qualquer lugar do arquivo onde houver espaço disponível. Simples mas sem ordem específica.

Sequential (Sequencial)

Registros armazenados em ordem sequencial, baseados no valor da chave de busca de cada registro.

Clustering Multitabela

Registros de várias relações diferentes podem ser armazenados no mesmo arquivo. Motivação: armazenar registros relacionados no mesmo bloco para minimizar I/O.

B+-tree

Organização de arquivo que mantém armazenamento ordenado mesmo com inserções e exclusões. Mais detalhes no Capítulo 14.

Hashing

Uma função hash é calculada na chave de busca; o resultado especifica em qual bloco do arquivo o registro deve ser colocado. Mais no Capítulo 14.

HEAP

Organização de Arquivo Heap

Na organização heap, registros podem ser colocados em qualquer lugar do arquivo onde haja espaço livre disponível. Uma vez alocados, os registros geralmente não se movem.

Desafio Principal

É crucial ser capaz de encontrar eficientemente espaço livre dentro do arquivo para novas inserções.

Free-Space Map (Mapa de Espaço Livre)

Um array com uma entrada por bloco. Cada entrada usa alguns bits (até um byte) e registra a fração do bloco que está livre.

01

Nível Primário

3 bits por bloco, valor dividido por 8 indica fração livre

02

Nível Secundário

Cada entrada armazena o máximo de 4 entradas do primeiro nível

03

Persistência

Mapa gravado periodicamente; valores antigos são detectados e corrigidos

Organização de Arquivo Sequencial

Quando Usar

Adequado para aplicações que exigem processamento sequencial de todo o arquivo, como relatórios completos ou operações em lote.

Características

Os registros no arquivo são ordenados por uma **chave de busca**, permitindo acesso eficiente a intervalos de dados.

Ordem Garantida

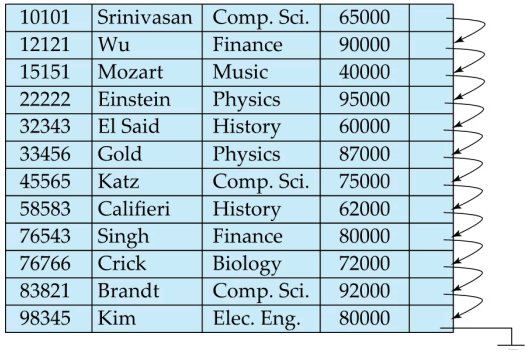
Registros mantidos em sequência lógica

Leitura Sequencial

Acesso rápido para varreduras completas

Busca por Intervalo

Eficiente para consultas de faixa de valores



10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	

Organização de Arquivo com Clustering Multitabela

Armazene várias relações em um único arquivo usando uma organização de arquivo de **clustering multitabela**. Esta técnica é particularmente útil quando há relacionamentos frequentes entre tabelas.

Relação Department

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Physics	Watson	70000

Informações sobre departamentos

Relação Instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000

Informações sobre instrutores

Clustering Combinado

Comp. Sci.	Taylor	100000	
10101	Srinivasan	Comp. Sci.	65000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000
Physics	Watson	70000	
33456	Gold	Physics	87000

Department e Instructor juntos

Benefício Principal: Reduz operações de I/O ao manter registros relacionados fisicamente próximos, melhorando significativamente o desempenho de joins.

OTIMIZAÇÃO AVANÇADA

Particionamento de Tabelas

Particionamento de tabelas permite que registros em uma relação sejam divididos em relações menores que são armazenadas separadamente, oferecendo benefícios significativos de desempenho e gerenciamento.

Exemplo Prático

A relação *transaction* pode ser particionada em *transaction_2018*, *transaction_2019*, *transaction_2020*, etc.

Consultas Inteligentes

Consultas escritas em *transaction* devem acessar registros em todas as partições, a menos que a consulta tenha uma seleção como *year=2019*, caso em que apenas uma partição é necessária.

Redução de Custos

Reduz custos de algumas operações, como gerenciamento de espaço livre, tornando o sistema mais eficiente.

Armazenamento Flexível

Permite que diferentes partições sejam armazenadas em diferentes dispositivos de armazenamento para otimizar desempenho.

Otimização por Idade

Por exemplo, partição de transações do ano atual em SSD, para anos anteriores em disco magnético mais lento e econômico.

Armazenamento do Dicionário de Dados

O **Dicionário de Dados** (também chamado de **catálogo do sistema**) armazena **metadados**, ou seja, dados sobre dados. Este componente crucial mantém informações essenciais sobre toda a estrutura do banco de dados.

Informações sobre Relações

- Nomes das relações
- Nomes, tipos e comprimentos dos atributos de cada relação
- Nomes e definições de views
- Restrições de integridade

Informações de Usuário

- Dados de usuários e contabilidade
- Senhas e permissões
- Controles de acesso

Dados Estatísticos

- Número de tuplas em cada relação
- Dados descritivos sobre distribuição
- Informações para otimização de consultas

Organização Física

- Como a relação é armazenada (sequencial/hash/etc.)
- Localização física da relação
- Informações sobre índices (Capítulo 14)

Representação Relacional de Metadados do Sistema

Armazenamento em Disco

Os metadados são armazenados usando representação relacional no disco, aproveitando as mesmas estruturas que os dados regulares do banco de dados.

Estruturas em Memória

Estruturas de dados especializadas são projetadas para acesso eficiente na memória, otimizando consultas frequentes ao catálogo.

01

Persistência

Metadados armazenados como tabelas relacionais em disco

02

Cache

Estruturas especializadas mantidas em memória principal

03

Otimização

Acesso rápido através de índices e cache inteligente

GERENCIAMENTO DE MEMÓRIA

Acesso ao Armazenamento

Os blocos são unidades tanto de alocação de armazenamento quanto de transferência de dados. O sistema de banco de dados busca minimizar o número de transferências de blocos entre o disco e a memória.

Estratégia de Otimização

Podemos reduzir o número de acessos ao disco mantendo o maior número possível de blocos na memória principal.

Buffer

Porção da memória principal disponível para armazenar cópias de blocos de disco, funcionando como cache de alta velocidade.

Gerenciador de Buffer

Subsistema responsável por alocar espaço de buffer na memória principal e implementar políticas de substituição de blocos.

Operações do Gerenciador de Buffer

Os programas chamam o gerenciador de buffer quando precisam de um bloco do disco. O gerenciador implementa um algoritmo sofisticado para maximizar a eficiência do cache.

Verificação do Buffer

Se o bloco já estiver no buffer, o gerenciador retorna o endereço do bloco na memória principal imediatamente.

Alocação de Espaço

Se o bloco não estiver no buffer, o gerenciador aloca espaço no buffer para o bloco.

Substituição de Bloco

Substitui (descarta) algum outro bloco, se necessário, para abrir espaço. O bloco substituído é gravado de volta no disco apenas se foi modificado.

Leitura e Retorno

Lê o bloco do disco para o buffer e retorna o endereço do bloco na memória principal ao solicitante.

Otimização Importante: Blocos substituídos são gravados de volta ao disco apenas se foram modificados desde a última vez que foram gravados ou buscados do disco, reduzindo operações de I/O desnecessárias.

Referências

Elmasri & Navathe

Fundamentals of Database Systems

Pearson, 2016

Referência abrangente sobre fundamentos de sistemas de bancos de dados, cobrindo aspectos teóricos e práticos.

Korth, Sudarshan & Silberschatz

Database System Concepts

McGraw-Hill, 2019

Texto fundamental que serviu como base para a maioria dos exemplos apresentados nesta apresentação.

Özsu & Valduriez

Principles of Distributed Database Systems

Springer Nature, 2019

Obra especializada em sistemas de bancos de dados distribuídos, essencial para compreensão avançada.

Nota: Os conceitos e exemplos apresentados baseiam-se principalmente na literatura clássica de sistemas de bancos de dados, em especial *Database System Concepts* e *Fundamentals of Database Systems*.