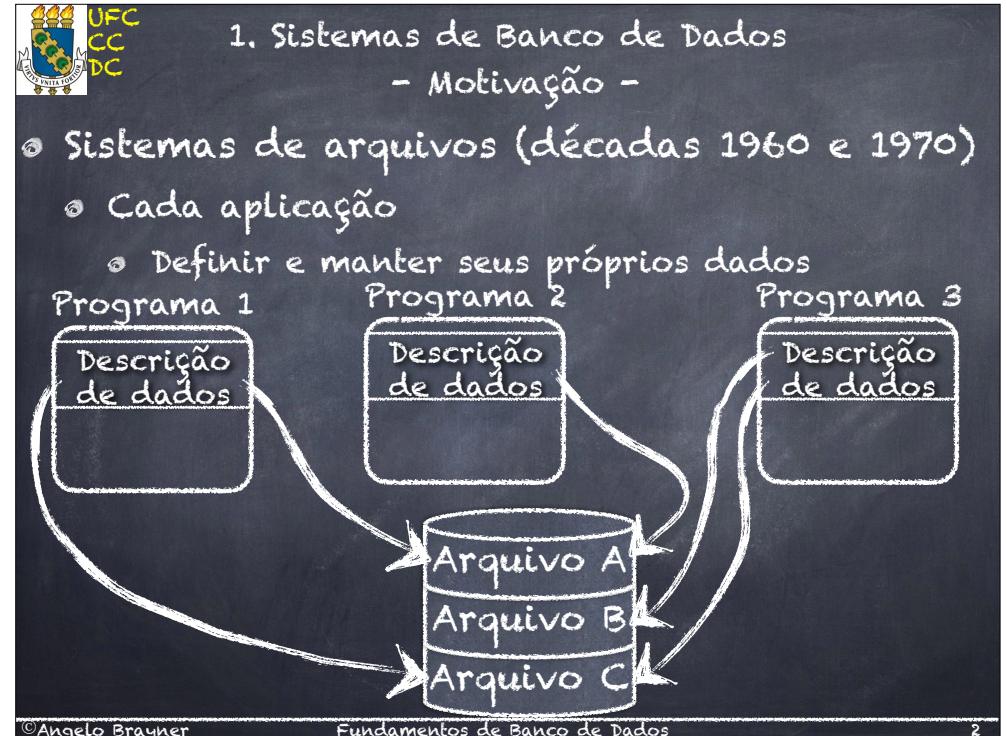


# Administração de Banco de Dados

Angelo Brayner  
Universidade Federal do Ceará  
Departamento de Computação  
brayner@dc.ufc.br

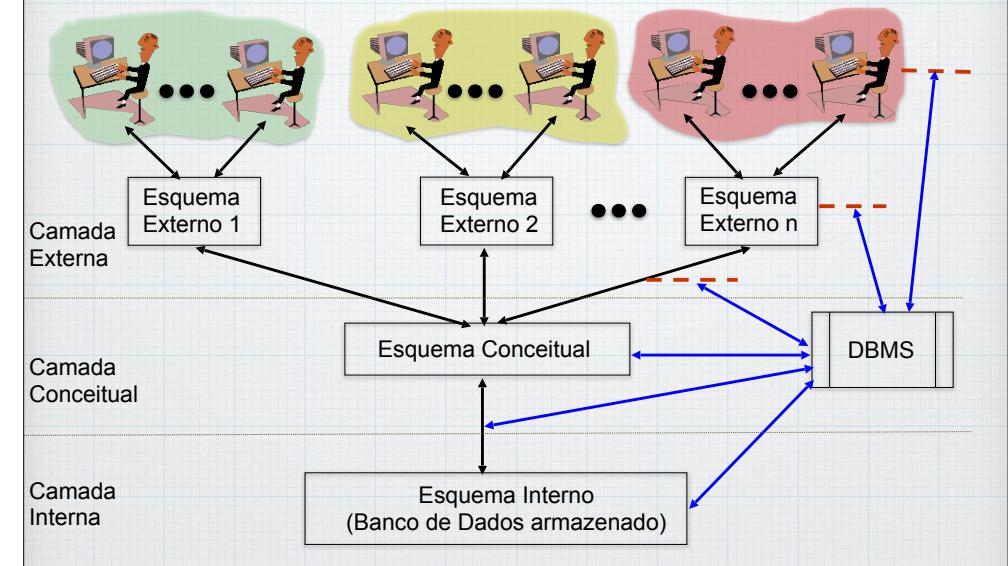


## 1. Sistemas de Banco de Dados - Motivação -

- Sistemas de arquivos (anos 60 e 70)
  - Aplicativo C para alterar nome de um estudante, dada a matrícula e o novo nome
  - Programa Altera\_Struct
- Correspondente na linguagem SQL
 

```
update Estudantes
set nome="Novo Nome"
where matricula=matr
```

## 1. Sistema de Banco de Dados - Arquitetura de Três Camadas -





UNIVERSIDADE  
FEDERAL DO CEARÁ

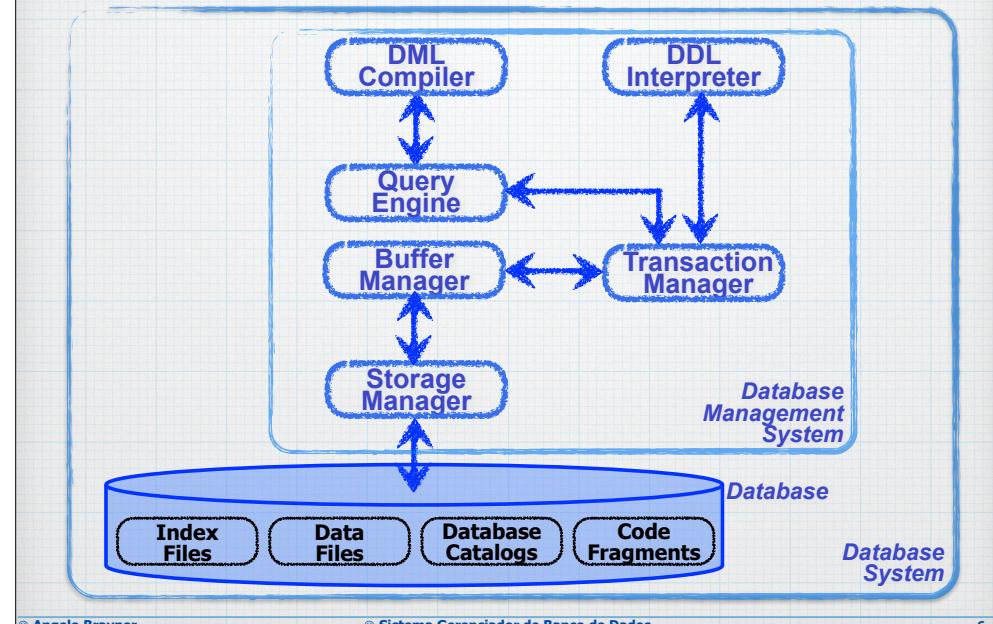
## 1. Database System - Definition -

### \* Main Components

- \* Database
  - \* Set of related data
- \* Database Management System
  - \* Storage
  - \* Access
  - \* Concurrency control
  - \* Recovery (from failures)



## 1. Database System - Architecture -



## 1. Sistema de Banco de Dados

- Componentes da Arquitetura -

### \* SGBD

- \* Compilador DML
  - \* Analisa sintaticamente e semanticamente comandos DML expressos em uma linguagem de consulta (ex. SQL)
  - \* Traduz estes comandos para uma das formas de representação interna de consultas (ex. álgebra relacional)
- \* Interpretador DDL
  - \* Interpreta comandos DDL e os armazena no catálogo
  - \* Tabelas contendo meta-dados
  - \* Descrição do banco de dados
  - \* Esquema

## 1. Sistema de Banco de Dados

- Componentes da Arquitetura -

### \* SGBD

- \* Mecanismo de Consultas
  - \* Responsável pela otimização e geração de planos de execução de consultas
- \* Componente para SQL embutido em LPs
  - \* Pré-Compilador DML
    - \* Traduz comandos DML em chamadas a procedimentos (rotinas) na linguagem hospedeira
    - \* Até SQL Server 2000
      - \* nsqprep precompiler
    - \* SQLJ



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 1. Sistema de Banco de Dados

- Componentes da Arquitetura -

### \* SGBD

- \* Gerenciador de Transações
  - \* Controle de concorrência
  - \* Recuperação do banco de dados após falhas
- \* Gerenciador de Buffer
  - \* Responsável por manter na área de buffer (memória principal) páginas mais acessadas
  - \* Política de alocação de páginas em buffer
    - \* LRU, MRU, FIFO, ARC
- \* Gerenciador de Armazenamento
  - \* Responsável pelo armazenamento físico em memória secundária



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 1. Sistema de Banco de Dados

- Componentes da Arquitetura -

### \* BD

- \* Arquivos de dados
  - \* Armazena os dados
- \* Índices
  - \* Estruturas de índices para os arquivos de dados
  - \* Índices ordenados
    - \* Árvores B+, Arquivos grade, Árvores kd, Árvores quadrantes
  - \* Índices não ordenados
    - \* Indice hash



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 1. Sistema de Banco de Dados

- Componentes da Arquitetura -

### \* BD

#### \* Catálogo

- \* Armazena esquema do banco de dados (meta-dados)
  - \* Nomes das tabelas, Atributos de cada tabela, Definição de índice para uma tabela, etc...
- \* Armazena informações estatísticas
  - \* Cardinalidade de uma tabela, fator de seletividade de colunas, etc
  - \* Utilizadas na otimização de consultas

#### \* Fragmentos de código

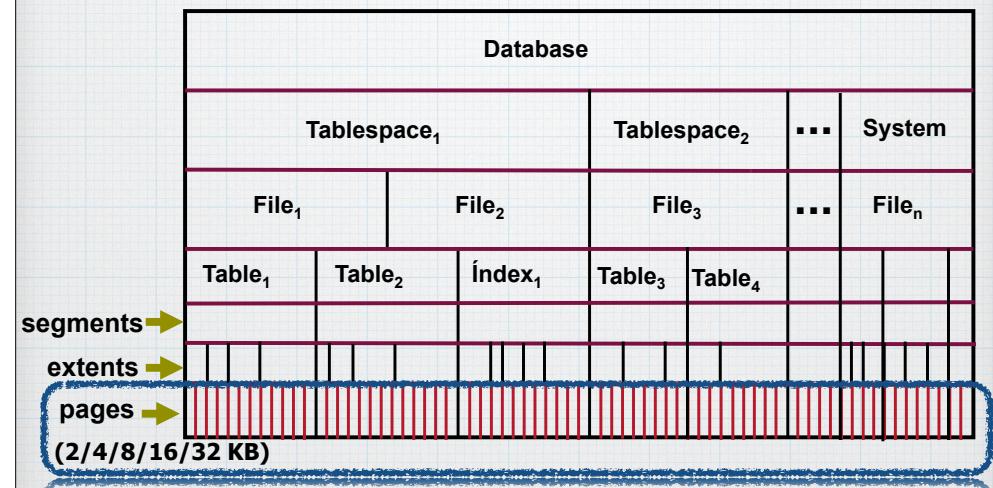
- \* Stored procedures, Triggers, User defined functions



## 1. Database System

- Logical DB Structure -

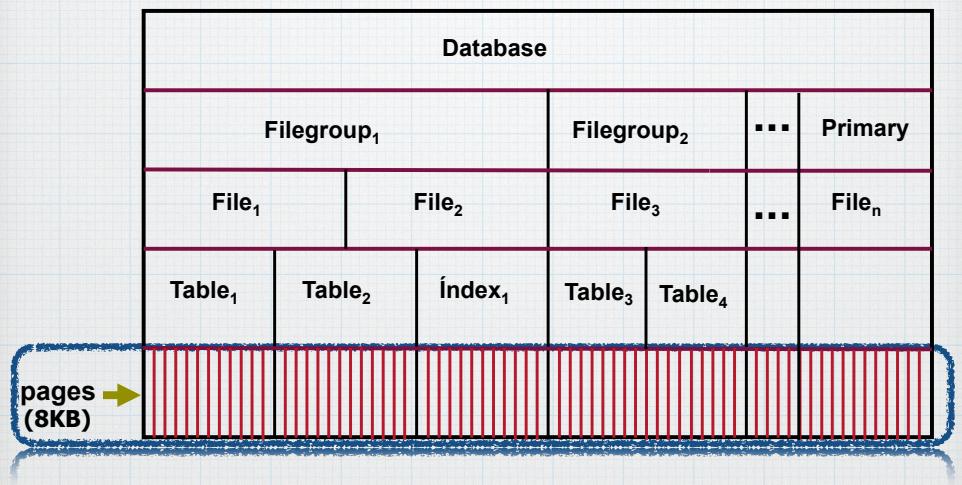
### \* Oracle



## 1. Database System

- Logical DB Structure -

- \* SQL Server



## 1. Database System

- Logical DB Structure -

- \* Database management system accesses pages from the database
- \* Page
  - \* Set of tuples
  - \* A very large tuple may be stored in more than one page
- \* Access unit for DBMS
- \* A table is often stored in several pages



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 1. Sistema de Banco de Dados - Classificação -

### \* Modelo de Dados

- \* Sistema de Banco de Dados Relacional
  - \* Modelo Relacional
  - \* Tipo primitivo de dados
    - \* Relação (tabela)
      - \* Conjunto de tuplas (linhas)
  - \* Conjunto de Operadores
    - \* Álgebra e Cálculo Relacional
  - \* Restrições de integridade
    - \* Integridade de chave primária, integridade referencial, restrição de domínio



## 1. Sistema de Banco de Dados - Classificação -

### \* Modelo de Dados

- \* Sistema de Banco de Dados Orientado a Objeto
  - \* Modelo orientado a objeto
    - \* Objeto, herança, composição, métodos
  - \* Tipos primitivos
    - \* Objeto, conjunto, lista, string, integer, real
- \* Sistema de Banco de Dados Objeto-Relacional
- \* Sistema de Banco de Dados NoSQL
  - \* Grafo
    - \* Neo4J
  - \* Documento
    - \* MongoDB

## 1. Sistema de Banco de Dados

- Classificação -

### \* Modelo de Dados

- \* Sistemas de Banco de Dados vetoriais
  - \* Armazenamento de vetores
  - \* Indexação de vetores
  - \* Busca por similaridade
  - \* Distância entre vetores
    - \* Coseno, euclidiana, euclidiana quadrada, etc ...

## 1. Sistema de Banco de Dados

- Classificação -

### \* Modelo de Dados

- \* Sistemas de Banco de Dados vetoriais
  - \* Armazena vetores
    - \* Representação de qualquer tipo de dados
    - \* Como pontos num espaço vetorial E
    - \* Localização no espaço representa alguma semântica
    - \* Transformar dados em vetores (embedding model)
  - \* Busca por similaridade
    - \* Vetor consulta Q
    - \* Pontos mais próximos de Q em E



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 1. Sistema de Banco de Dados - Classificação -

### \* Modelo de Dados

#### \* Sistemas de Banco de Dados vetoriais

##### \* Oracle 23ai

- \* create table artigos\_sbdb (artigo\_id int, texto\_artigo CLOB, texto\_artigo\_vetor vector)
- \* create table artigos\_sbdb (id int, texto\_artigo CLOB, texto\_artigo\_vetor vector (560,INT8))
- \* SELECT artigo\_id FROM artigos\_sbdb
- \* ORDER BY  
VECTOR\_DISTANCE(texto\_artigo\_vetor, :query\_vector,  
cosine\_distance )
- \* FETCH EXACT FIRST 10 ROWS ONLY;



## 1. Sistema de Banco de Dados - Classificação -

### \* Modelo de Dados

#### \* Sistemas de Banco de Dados vetoriais

##### \* Db2 (watsonx)

##### \* MongoDB Atlas

##### \* Azzure com MongoDB Atlas

##### \* REDIS

##### \* Databricks



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 1. Sistema de Banco de Dados - Classificação -

### \* Arquitetura

#### \* Centralizada

- \* Sistema de Banco de Dados Centralizados
  - \* Os componentes do SBD residem no mesmo host

#### \* Distribuída

- \* Sistema de Banco de Dados Cliente-Servidor
  - \* Distribuição de funções do SGBD entre clientes e servidor
- \* Sistema de Banco de Dados Paralelos
  - \* Distribuição do controle de funções do DBMS entre diversos sistemas computacionais



## 1. Sistema de Banco de Dados - Classificação -

### \* Arquitetura

#### \* Distribuída

- \* Sistema de Banco de Dados Distribuídos
  - \* Banco de dados distribuído em nós de uma rede
    - \* Distribuição de dados através de SBDs homogêneos
  - \* BDs distribuídos
    - \* Distribuição de objetos do banco de dados em nós de uma rede
      - \* Tabela
      - \* Fragmentação de tabelas
        - \* Horizontal
        - \* Vertical
    - \* Replicação
      - \* Cópia dp banco de dados em nós de uma rede



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 1. Sistema de Banco de Dados - Classificação -

- \* Arquitetura
  - \* Distribuída
    - \* Sistema de Banco de Dados Heterogêneos
      - \* Distribuição de dados através de SBDs heterogêneos
        - \* Sistema de banco de dados múltiplos (MDBS)
        - \* Sistema de banco de dados federados (FDBS)
          - \* Wrappers
          - \* Arquitetura de Mediadores
    - \* Sistema de Banco de Dados Móvel
      - \* Distribuição de funções e dados do SGBD entre clientes e servidor em ambiente de computação móvel
    - \* Sistema de Fluxo de Dados
      - \* Distribuição de funções e dados entre componentes que geram e acessam fluxos de dados
      - \* Redes de Sensores Sem Fio



## 1. Sistema de Banco de Dados - Classificação -

- \* Arquitetura
  - \* Distribuída
    - \* Sistema de Banco de Dados Multi-plataforma
      - \* Replicação de dados sob demanda em múltiplos dispositivos

## 2. Ajuste de Performance

- Motivação -

### \* Cenário 1 - Transaction Manager

```
begin transaction T1
set transaction isolation level repeatable read
update lineitem with (rowlock)
set l_quantity=l_quantity
commit
```

### \* Cenário 2 - Query Engine

```
select l_orderkey, s_name
from lineitem inner join partsupp inner join part inner join supplier
on ps_suppkey=s_suppkey on p_partkey=ps_partkey on
l_partkey=ps_partkey
where l_quantity>9
```

## 2. Ajuste de Performance

- Conceitos básicos -

### \* Ajustar performance de um SGBD

- \* Objetivo
  - \* Melhorar a performance do sistema de banco de dados para uma determinada aplicação
- \* Ajustar
  - \* Configuração de hardware
    - \* RAID, memória principal, CPU (processadores multi-core)
  - \* Parâmetros do SGBD
    - \* Granulosidade de bloqueio, área de buffer, intervalo de geração de checkpoint, lock escalation, protocolo de controle de concorrência
  - \* Configuração de projeto
    - \* Lógico
      - \* Denormalização
    - \* Físico
      - \* Índices, áreas de armazenamento, visão materializada

## 2. Ajuste de Performance

- Conceitos básicos -

### \* Gargalo

- \* Recurso do sistema que está sendo utilizado em sua capacidade máxima, limitando a vazão do sistema

### \* Disco

- \* Taxa de I/O atingiu o máximo da largura de banda

### \* Sintonia fina (**tuning**)

- \* Identificação e eliminação de gargalos

## 2. Ajuste de Performance

- Conceitos básicos -

### \* Sintonia fina (**tuning**)

- \* Atividade complexa
- \* Requer conhecimento
  - \* Funcionamento de componentes do SGBD
  - \* Processamento de consultas, controle de concorrência, recuperação, indexação
- \* Outras áreas da computação
  - \* SO, Configuração física de hardware
  - \* Carga de trabalho



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 2. Ajuste de Performance

- Conceitos básicos -

### \* Princípios básicos

- \* Pensar global, otimizar local
- \* Pensar global
  - \* Analisar as estatísticas de hardware
    - \* Identificar a utilização
      - \* Processador
      - \* Vazão de memória secundária (IOPS)
      - \* Paginação
        - \* Taxa de acerto
        - \* Número de páginas retiradas do buffer
          - \* Devido a geração de checkpoints
          - \* Fuzzy checkpoint
    - \* Fazer uma análise global das várias métricas



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 2. Ajuste de Performance

- Conceitos básicos -

### \* Princípios básicos

- \* Pensar global, otimizar local
- \* Erro comum
  - \* Reagir a um valor alto em uma métrica (por exemplo, alta atividade de disco)
  - \* Comprando hardware (comprando mais discos)
- \* Altas taxas de I/O podem ser reduzidas
  - \* Configuração física do banco de dados
    - \* com a criação de índices
      - \* evitar consulta frequente com table scan
    - \* Colocar o arquivo de log em disco dedicado
    - \* Distribuir arquivos de dados em diferentes discos

## 2. Ajuste de Performance

- Conceitos básicos -

- \* Princípios básicos

- \* Distribuição elimina gargalos
  - \* Distribuição de objetos do banco de dados em diferentes nós de uma rede
    - \* Tabelas
    - \* Partições de uma tabela
  - \* Replicação de banco de dados
- \* Servidor de banco de dados isolado da aplicação
  - \* Evitar fragmentos de código
  - \* Definir cursores no cliente

## 2. Ajuste de Performance

- Conceitos básicos -

- \* Princípios básicos

- \* Trade-offs são a regra
  - \* Ganhos com algum ajuste pode implicar perdas em algum outro ponto
  - \* Aumento da área de buffer
    - \* Pode transferir o problema da paginação para o SO
  - \* Criação de índice
    - \* custo alto de manutenção
  - \* Aumento do espaço de busca para o plano de execução de consulta ótimo



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 2. Ajuste de Performance

- Conceitos básicos -

- \* Benchmarks TPC (Transaction Processing Performance Council - <http://www.tpc.org>)
  - \* TPC-A
    - \* Quantificar performance em BDs com atualizações intensivas (Aplicações OLTP)
    - \* Incorpora comunicação com terminais
    - \* Simula aplicação bancária
  - \* TPC-B
    - \* Quantificar performance do núcleo de BDs com atualizações intensivas (Aplicações OLTP)
    - \* TPC-A removendo a comunicação com terminais



## 2. Ajuste de Performance

- Conceitos básicos -

- \* Benchmarks
  - \* TPC-C
    - \* Modela um sistema de vendas de um fornecedor, com produtos distribuídos em vários depósitos
      - \* Entrada de solicitações (entrada de solicitações e entrega de encomendas, armazenamento de faturas e pagamentos)
    - \* Banco de dados
      - \* 9 tabelas
    - \* Carga de trabalho
      - \* 5 transações
        - \* novo pedido, pagamento, status pedido, entrega, status do estoque



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 2. Ajuste de Performance

- Conceitos básicos -

### \* Benchmarks

#### \* TPC-E

- \* Simula um ambiente de transações em bolsas de valores
- \* Banco de dados
  - \* 33 tabelas
  - \* Populated with pseudo-real data
    - \* Distributions based on 2000 U.S. and Canada census data
    - \* Used for generating name, address, gender, etc.
    - \* Introduces natural data skew
  - \* Actual listings on the NYSE and NASDAQ
- \* Carga de trabalho
  - \* 6 R0 transactions
  - \* 4 RW Transactions



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 2. Ajuste de Performance

- Conceitos básicos -

### \* Benchmarks

#### \* TPC-H

- \* Quantificar performance de BDs com consultas intensivas (Aplicações de suporte a decisão)
  - \* Data warehouse application (snowflake)
- \* Banco de dados
  - \* 8 tables
  - \* Several scale factors
    - \* SF 10: ~13GB database
- \* Carga de trabalho
  - \* 22 queries



## 3. Memory System

- Motivation -

- \* Processors

- \* May execute  $128 \times 10^9$  instructions/second (Intel i7 2600k)
  - \*  $\sim 7.8 \times 10^{-12}$ s to execute 1 instruction

- \* Access time - Main Memory

- \*  $7 \times 10^{-9}$ s

- \* In order to process 2 instructions

- \* A processor may wait  $\sim 6.9922 \times 10^{-9}$  s
  - \* The second instruction should be fetched

- \* To reduce CPU wait time

- \* Storage (Memory) Hierarchy

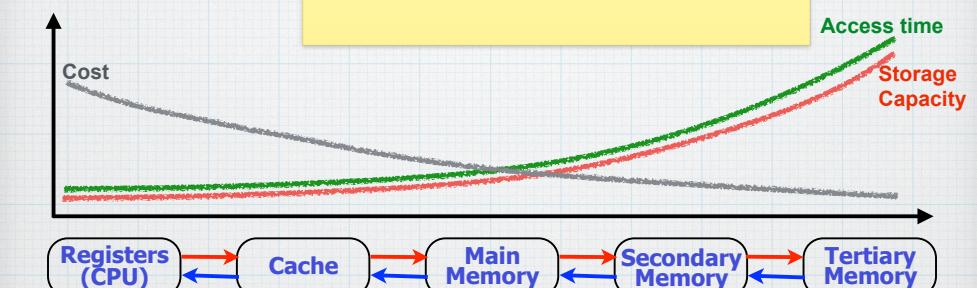


## 3. Memory System

- Motivation -

### \* Storage Hierarchy

The access time varies with more than 8 orders of magnitude between the fastest to the slowest. In turn, the storage capacity varies with at least 7 orders of magnitude. On the other hand, the cost per byte varies more smoothly with about 3 orders of magnitude between the cheapest and the most expensive forms of storage.





UNIVERSIDADE  
FEDERAL DO CEARÁ

### 3. Memory System - Memory Classification -

- \* Data persistence

- \* Volatile

- \* Requires power to persist data

- \* Registers, Cache and Main memory

- \* Non-volatile

- \* Holds stored data regardless energy supply

- \* Hard disks, magnetic tapes, optical disks



UNIVERSIDADE  
FEDERAL DO CEARÁ

### 3. Memory System - Memory Classification -

- \* Access method

- \* Random - RAM

- \* Any memory address is accessed in the same amount of time

- \* Registers, Cache and Main memory

- \* Sequential - SAM

- \* Data stored in different memory addresses may have different access time

- \* Hard disks



### 3. Memory System

- Memory Types [Registers] -

- \* Small amount of storage available as part of a processor (CPU) ➔
- \* Size (word)
  - \* Multiple of 8 bits: [8, 64]
- \* Types
  - \* Memory Buffer Register (MBR)
    - \* Holds data being transferred from cache (L1) to processor and vice versa.



### 3. Memory System

- Memory Types [Registers] -

- \* Types
  - \* Instruction Register (IR)
    - \* Holds the instruction currently being executed
  - \* Memory Address Register (MAR)
    - \* Stores memory address (L1) of data to be transferred to register or the address to which data will be moved into L1
  - \* Access time
    - \*  $\sim 1\text{ns}$  ( $10^{-9}\text{s}$ )
  - \* Physical storage medium
    - \* Metal Oxide Semiconductor (MOS)



### 3. Memory System - Memory Types [Cache] -

- \* Memory between Registers and Main Memory
  - \* To reduce the processor wait time for accessing data/instruction from main memory
- \* Often, divided into two levels
  - \* L1
    - \* Located on the same chip as the processor
    - \* 16k
  - \* L2 →
    - \* Located on a different chip
    - \* 2MB



### 3. Memory System - Memory Types [Cache] -

- \* Physical storage medium
- \* Static RAM (SRAM)
  - \* Semiconductor memory
  - \* Stores a bit
    - \* Bistable Latching circuitry
    - \* Four transistors
  - \* Does not need to be periodically refreshed
  - \* Access time
    - \* ~10ns

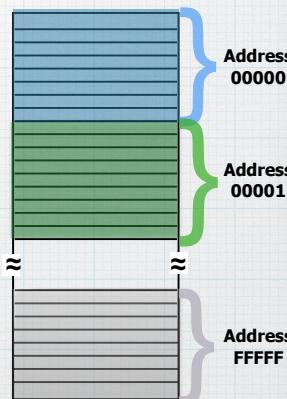
The term static differentiates it from dynamic RAM (DRAM) which must be periodically refreshed. A typical SRAM cell is made up of six MOSFETs. Each bit in an SRAM is stored on four transistors (M1, M2, M3, M4) that form two cross-coupled inverters. This storage cell has two stable states which are used to denote 0 and 1. Two additional access transistors serve to control the access to a storage cell during read and write operations. In addition to such six-transistor (6T) SRAM, other kinds of SRAM chips use 4, 8, 10 (4T, 8T, 10T SRAM), or more transistors per bit.<sup>[4][5][6]</sup> Four-transistor SRAM is quite common in stand-alone SRAM devices (as opposed to SRAM used for CPU caches), implemented in special processes with an extra layer of polysilicon, allowing for very high-resistance pull-up resistors. <sup>[7]</sup>



### 3. Memory System

- Memory Types [Main Memory] -

- \* Word-addressable storage
  - \* Controlled by the CPU
- \* Access unit (r/w)
  - \* Word
    - \* 16, 32, 64 bits
- \* Physical Storage Medium
  - \* Dynamic RAM (DRAM)
    - \* Semiconductor memory
      - \* Composed of cells
        - \* Cells of 8 bits
      - \* Thus, in a memory of 64 bits
        - \* An address for a set of 8 cells
  - \* Access time
    - \* ~60ns



### 3. Memory System

- Memory Types [Secondary Memory] -

- \* Non-volatile memory
  - \* Used to persist application data
- \* Orders of magnitude slower and more capacious than main memory
- \* Access time to get the contents of a given memory address depends on its location on the device



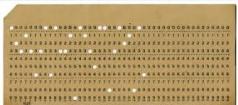
UNIVERSIDADE  
FEDERAL DO CEARÁ

### 3. Memory System

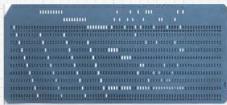
- Memory Types [Secondary Memory] -

\* Used Medias for secondary storage

\* Punched cards



Punch card machine: Storing data on punched cards



IBM 711 card reader on an IBM 704 computer at NASA (1957)



UNIVERSIDADE  
FEDERAL DO CEARÁ

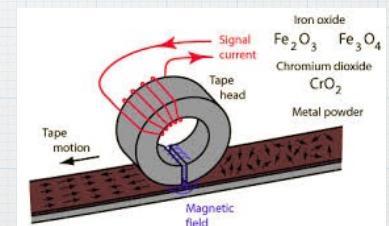
### 3. Memory System

- Memory Types [Secondary Memory] -

\* Used Medias for secondary storage

\* Magnetic tape

\* Long strip of plastic film overlaid by a thin magnetizable coating



### 3. Memory System

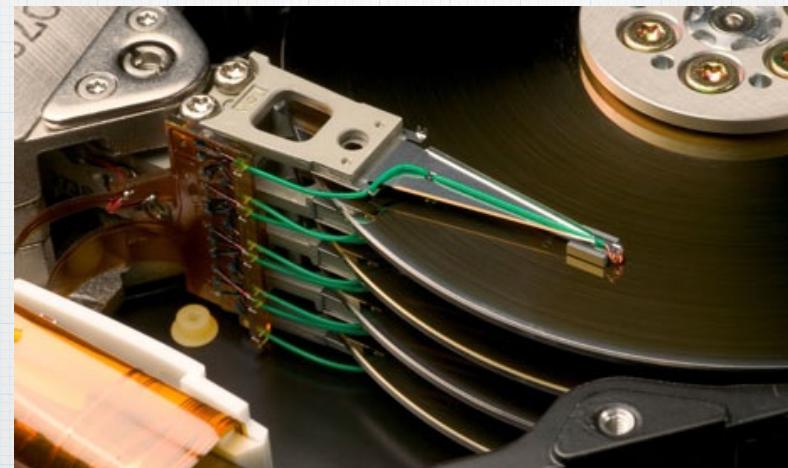
- Memory Types [Secondary Memory] -

- \* Used Medias for secondary storage
  - \* Hard (magnetic) disks
    - \* IBM 305 RAMAC (1956)
      - \* First computer with SMB HDD
        - \* HDD weighted over a ton
      - \* Samsung Spinpoint M9T
        - \* 2.5-Inch HDD
        - \* 2TB
        - \* 5400rpm



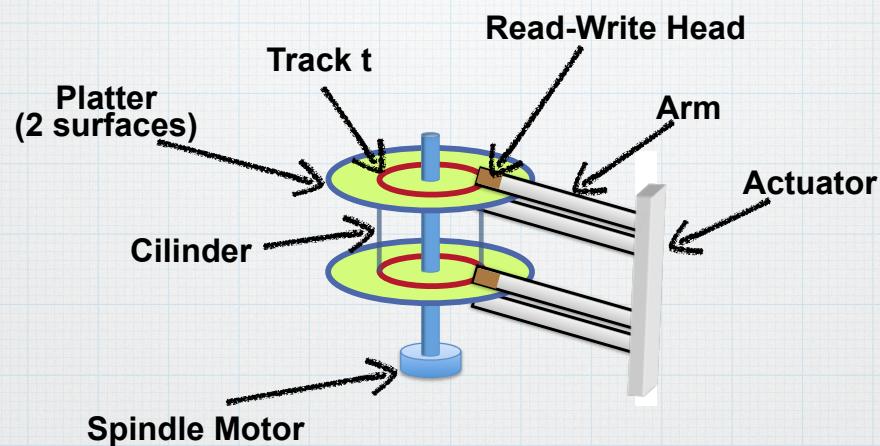
### 3. Memory System

- Anatomy of an HD Drive -





### 3. Memory System - Anatomy of an HD Drive -



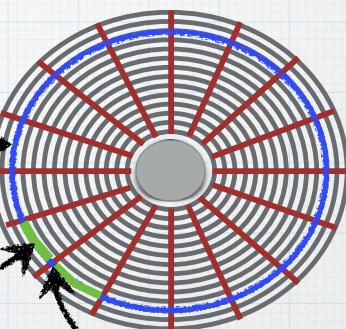
### 3. Memory System - Anatomy of an HD Surface -

#### Surface (2 per platter)

- Covered with a thin layer of magnetic material
- Organized into tracks
- Concentric circles

#### Track

- Circular path on disk surface
- Stores information (bits) →
- Subdivided into Sectors



#### Sector

- Smallest physical storage unit on the disk
- Read/Write unit
- Indivisible unit as far as errors are concerned
  - Should a small region  $R$  of the magnetic coating be corrupted in some way
    - The entire sector containing  $R$  is corrupted as well
      - It can not store information anymore
- Each sector stores the same amount of data
- 4096 Bytes

#### Gap

- non-magnetized area
- Separates sectors
- ~10% of the track



UNIVERSIDADE  
FEDERAL DO CEARÁ

### 3. Memory System - Anatomy of an HD Surface -

- \* Each disk sector is labelled using the factory track-positioning data
  - \* Sector identification data
    - \* Written to the area immediately before the contents of the sector
    - \* Identifies the starting address of the sector
- \* Block/Cluster
  - \* Logical unit
    - \* Used to transfer data between disk and main memory
    - \* Typically, consists of one or more sectors

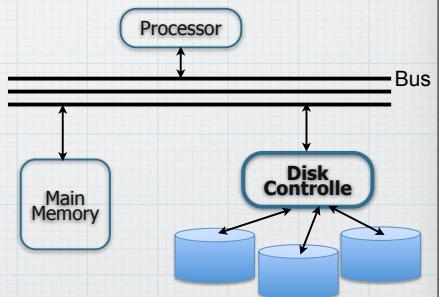


UNIVERSIDADE  
FEDERAL DO CEARÁ

### 3. Memory System - Disk Controller -

#### \* Disk controller

- \* Processor
  - \* Selects plate P and surface S to read/write
  - \* Controls the actuator to position heads at a track t
    - \* Selects sector from t of S
- \* Transfers bits from
  - \* Sector->Main memory (r)
  - \* Main memory->Sector (w)
- \* RAM
- \* Buffer
  - \* To flush burst of writes





UNIVERSIDADE  
FEDERAL DO CEARÁ

### 3. Memory System

- Disk Latency -

- \* Time interval between the moment the command to access the disk is triggered and it is completed
- \* Seek time
  - \* Time to position the r/w head at the proper track
  - \* Speedup time
    - \* Time to accelerate the arm until it reaches a fixed maximum velocity or half of the distance to the proper track
    - \* Acceleration rate may reach 550g
  - \* Travel time (long seeks)
    - \* Time in which the arm moves at constant speed (max speed)

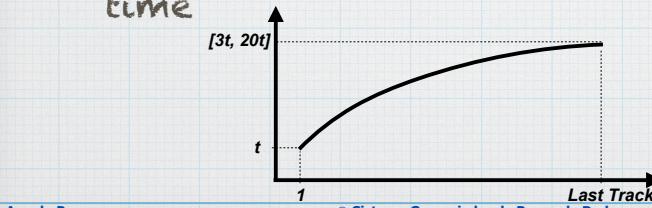


UNIVERSIDADE  
FEDERAL DO CEARÁ

### 3. Memory System

- Disk Latency -

- \* Seek time
  - \* Slowdown time
    - \* Time to bring the arm to rest close to the desired track
  - \* Settle time
    - \* Time for disk controller to adjust the head to position it at the correct track
  - \* In long seeks, travel time dominates seek time





### 3. Memory System - Disk Latency -

#### \* Rotational latency (RL)

- \* Time for the disk to rotate in order to position the proper sector under the r/w head
- \* On the average, time for half rotation
- \* Let  $r$  be the amount of rotations per second
  - \*  $RL = (2r)^{-1}$

#### \* Transfer time (TT)

- \* Time to transfer data
  - \* From the sector to the head ( $r$ )
  - \* From the head to the sector ( $w$ )



### 3. Memory System - Disk Latency -

#### \* Transfer time

- \* Let  $T$  be the track storage capacity and  $S$  the sector storage capacity in bytes
- \*  $TT = S(rT)^{-1}$
- \* Disk latency (DL) to access one sector
  - \*  $DL = S_T + (2r)^{-1} + S(rT)^{-1}$



UNIVERSIDADE  
FEDERAL DO CEARÁ

### 3. Memory System - RAID -

- \* Redundância para aumentar a confiabilidade de HDs
- \* Pode-se aumentar tempo médio de perda de dados através da redundância de dados
  - \* Armazenar dado redundante
    - \* Dado normalmente não necessário
    - \* Utilizado para reconstruir dado perdido
- \* Exemplo
  - \* Espelhamento de discos
    - \* Caso um disco falhe
      - \* dados podem ser acessados no disco espelho
    - \* Perda total de dados
      - \* disco espelho falhar antes que o primeiro seja recuperado



UNIVERSIDADE  
FEDERAL DO CEARÁ

### 3. Memory System - RAID -

- \* Distribuição de dados para aumentar desempenho de HDs
- \* Acesso paralelo
- \* Em um conjunto de N discos
  - \* Acesso paralelo a diversos discos através da distribuição paralela de dados nos múltiplos discos
    - \* Aumentar a taxa de transferência de dados
- \* Estratégias para distribuição de dados
  - \* Distribuição paralela de bits
    - \* Distribuir os bits de um byte em diferentes discos

## 3. Memory System - RAID -

### \* Distribuição de dados

- \* Estratégias para distribuição de dados (cont.)
  - \* Distribuição paralela de blocos
    - \* Distribuir os blocos de um arquivo em diferentes discos
    - \* Política de alocação de blocos de um arquivo
      - \* Bloco  $i$  alocado no disco  $(i \bmod n)+1$
      - \*  $n$  representa o número de discos disponíveis

## 3. Memory System - RAID -

### \* Mecanismo de RAID

- \* Combinar diferentes graus de distribuição e de redundância
- \* Níveis de RAID
- \* Fatiamento (striping)
  - \* Concatenar múltiplas unidades de disco em uma única unidade de armazenamento lógica

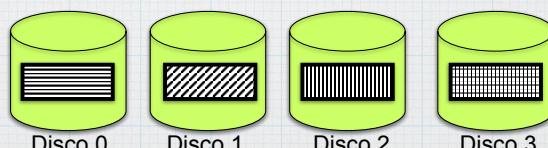
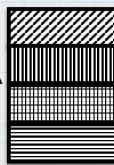


UNIVERSIDADE  
FEDERAL DO CEARÁ

### 3. Memory System - RAID -

#### \* RAID nível 0

- \* Organização de discos com distribuição paralela
  - \* Baseada em blocos
    - \* Blocos de um arquivo são armazenados em diversos discos
- \* Não apresenta redundância
- \* Exemplo



Arquivo A



### 3. Memory System - RAID -

#### \* RAID nível 1 - Proposta original

- \* Distribuição paralela
  - \* Baseada em blocos
- \* Redundância implementada através de replicação
- \* Equivale ao nível 10, existente no mercado
- \* Exemplo
  - \* RAID 1 com quatro discos de dados e quatro discos para replicação





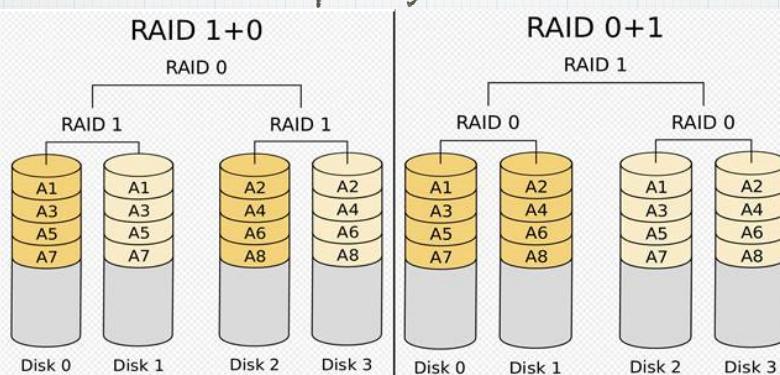
UNIVERSIDADE  
FEDERAL do CEARÁ

### 3. Memory System - RAID -

\* RAID níveis 10 e 01

\* RAID nível 1 - Mercado

\* Só tem e replicação

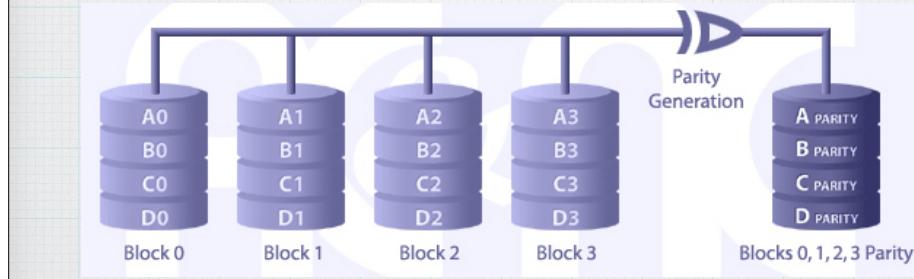


### 3. Memory System - RAID -

\* RAID nível 4

\* Distribuição por blocos

\* Redundância implementada através de blocos de paridade





UNIVERSIDADE  
FEDERAL DO CEARÁ

### 3. Memory System - RAID -

- \* RAID nível 4

- \* Caso um disco falhe

- \* Bloco de paridade mais os blocos dos discos em funcionamento são utilizados para recuperar os blocos do disco que falhou



UNIVERSIDADE  
FEDERAL DO CEARÁ

### 3. Memory System - RAID -

- \* RAID nível 5

- \* Mecanismo de paridade distribuída de blocos entrelaçados

- \* Distribuição

- \* Blocos de dados e de paridade são distribuídos nos n discos
    - \* Lembre-se que no RAID nível 4
    - \* bloco de paridade é armazenado em um disco separado

- \* Redundância implementada através de blocos de paridade

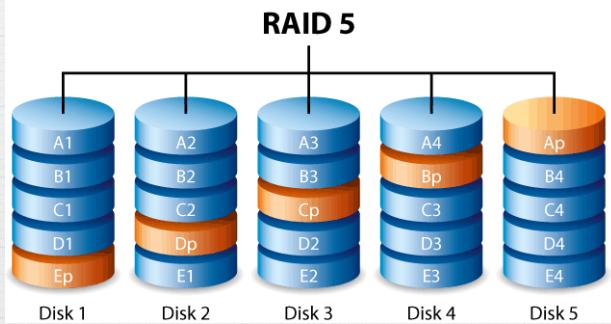
- \* Um bloco de paridade para cada arquivo



### 3. Memory System - RAID -

#### \* RAID nível 5 com 5 discos

- \* Caso um disco falhe
- \* Bloco de paridade de um arquivo mais os blocos de discos operacionais
- \* Utilizados para recuperar um disco

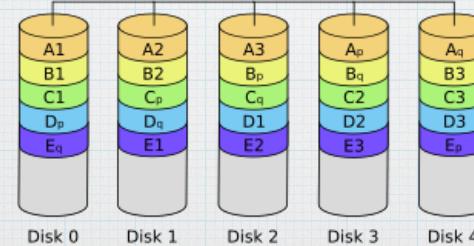


### 3. Memory System - RAID -

#### \* RAID nível 6

- \* Semelhante ao nível 5
- \* Armazena duas paridades por arquivo
- \* Para garantir recuperação em caso de falha de dois discos
- \* Maior confiabilidade que o nível 5

**RAID 6**





UNIVERSIDADE  
FEDERAL DO CEARÁ

### 3. Memory System - RAID -

- \* Escolha do nível de RAID
  - \* Quantidade de disco disponíveis
  - \* Tempo e complexidade da atividade de reconstrução de discos após falhas
  - \* Aplicações que requerem fornecimento contínuo de dados
  - \* RAID 1
    - \* Baixa complexidade e baixo tempo de reconstrução



### 3. Memory System - RAID -

- \* Escolha do nível de RAID
  - \* Tipo de aplicações que acessam discos do mecanismo RAID
    - \* RAID nível 0
      - \* Aplicações de alto desempenho
      - \* Discos bastante confiáveis
    - \* RAID nível 1
      - \* Aplicações que requerem fornecimento contínuo de dados
    - \* RAID nível 5
      - \* Aplicações com fornecimento contínuo de dados e que armazenam grandes volumes de dados



UNIVERSIDADE  
FEDERAL DO CEARÁ

### 3. Memory System - RAID -

- \* Escolha do nível de RAID

- \* Best practices

- \* RAID 1

- \* operating systems, binaries, index and log files

- \* RAID 5

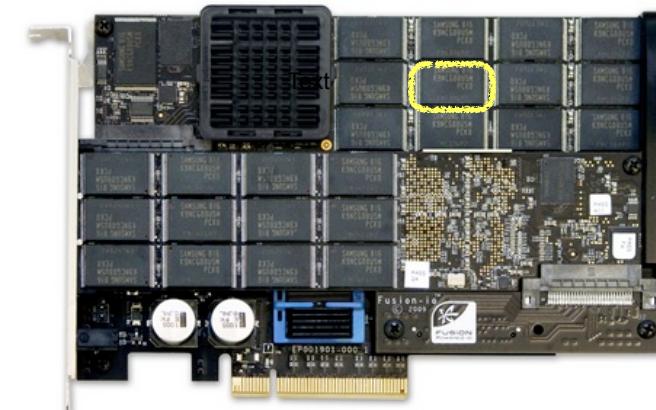
- \* data files with random access and read heavy data volumes, striping is important

- \* Putting log files on a RAID 5 array is not recommended, since RAID 5 does not perform well for write operations.



### 3. Memory System - Solid State Drive -

- \* Storage device made of silicon chips instead of spinning metal platters
- \* No mechanical part



Fusion-io's ioMemory Module of 1.28TB



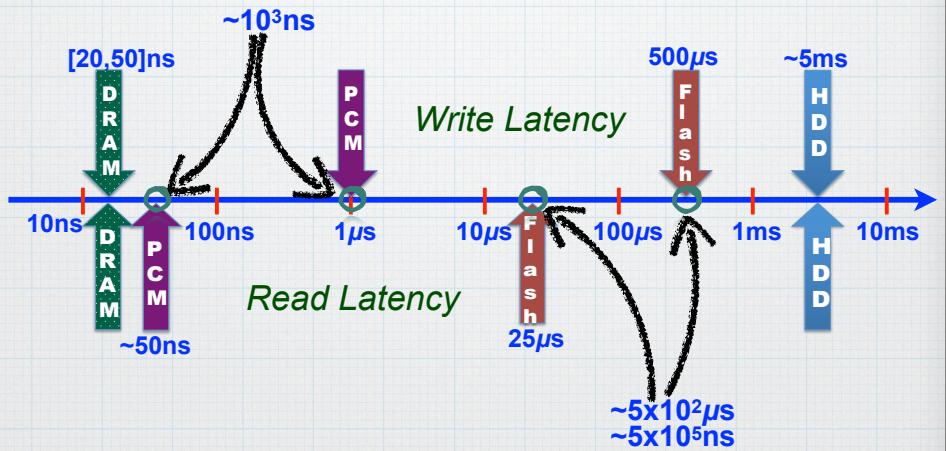
### 3. Memory System - Types of Solid State Drive -

- \* Flash (NAND)
  - \* Computer chip which can be electrically reprogrammed and erased
  - \* Stores data in a floating-gate transistor (cell)
    - \* Array of cells
    - \* Data are represented by the voltage level in a cell
      - \* High voltage (>5v) - 0
      - \* Low voltage - 1
      - \* Default state
  - \* Block/page addressable
    - \* Read on a page
    - \* Write on a block/page

### 3. Memory System - SSD Properties -

#### \* Read/Write Asymmetry

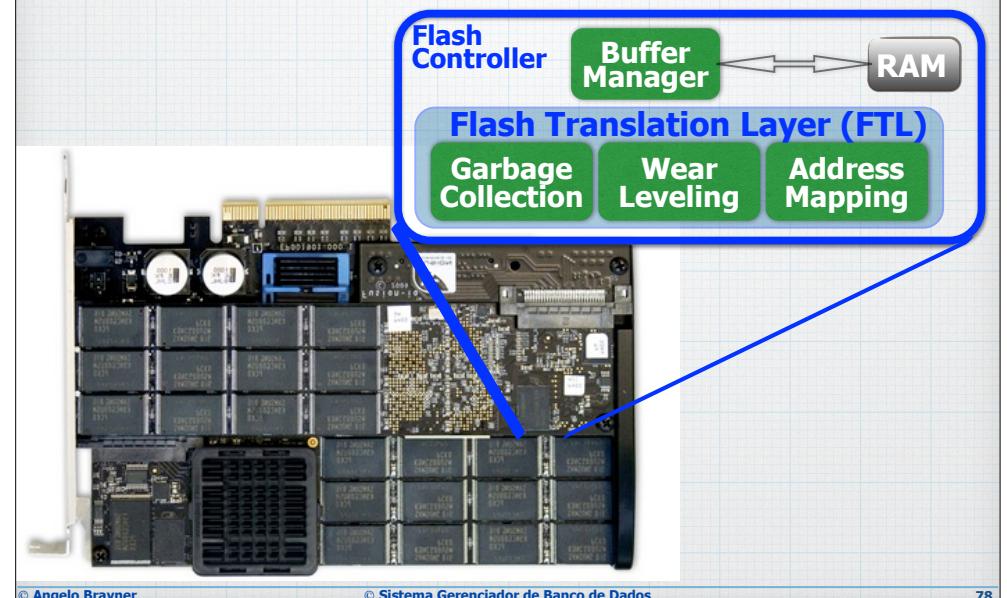
#### \* Execution time



### 3. Memory System - SSD Properties -

- \* Read/Write Asymmetry
  - \* Energy consumption
- \* Write endurance
  - \* Lifetime determined by the number of write operations
- \* 2011
  - \* Flash
    - \*  $10^5$
  - \* PCM
    - \*  $10^7$

### 3. Memory System - Anatomy of a NAND Flash -





### 3. Memory System - Anatomy of a NAND Flash -

**Package:**  
**(4, 6, 8, 16) Chips**



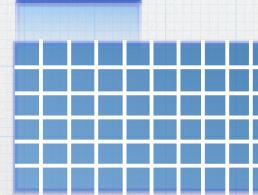
small block of semiconducting material, on which a given functional circuit is fabricated

The die is the smallest unit that can independently execute commands or report status.

**Chip (Die):  
Planes  
(several)**

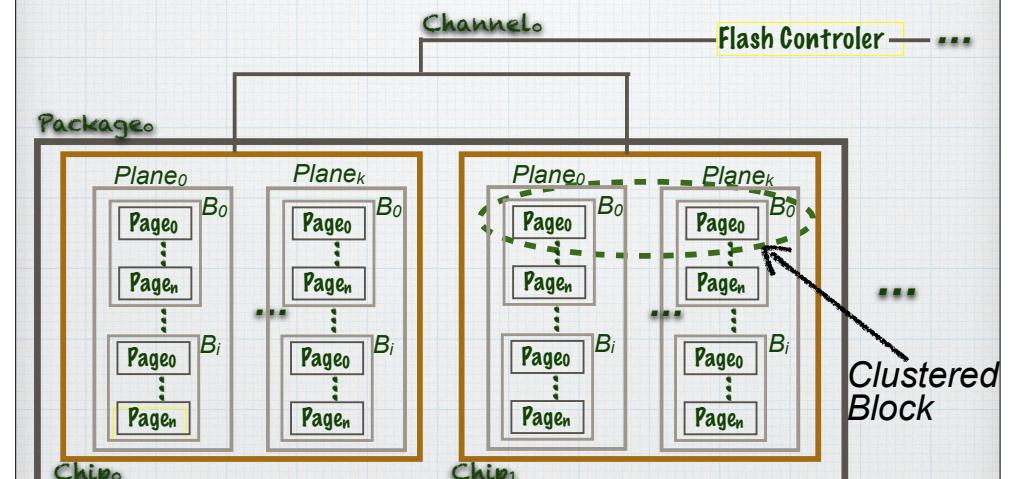


**Plane:**  
**Thousands of blocks  
(2048)**



**Block:**  
**several pages  
(64 2k-pages)**

### 3. Memory System - Anatomy of a NAND Flash -



### 3. Memory System - NAND Flash Properties -

- \* Single Layer Cell (SLC)
  - \* One bit per cell
- \* Multi-Layer Cell
  - \* 2 bits per cell
    - \* 00 - High voltage
    - \* 01 - Medium high voltage
    - \* 10 - Medium low voltage
    - \* 11 - Low voltage

### 3. Memory System - NAND Flash Properties -

- \* Single Layer Cell (SLC)
  - \* Faster
  - \* More reliable
- \* Multi-Layer Cell
  - \* Presents higher density



UNIVERSIDADE  
FEDERAL DO CEARÁ

### 3. Memory System - NAND Flash Properties -

- \* Operations

- \* Read
    - \* Page addressable

- \* Erase

- \* Sets all bits to 1
    - \* Block addressable

- \* Program (physical write)

- \* Sets a bit to 0
    - \* It can only be executed on a "clean" block
      - \* A block with all bits set to 1

- \* Page addressable



UNIVERSIDADE  
FEDERAL DO CEARÁ

### 3. Memory System - NAND Flash Properties -

- \* Once all blocks have at least a bit set to 0

- \* A (logical) write operation has to be executed

- \* Erase and program operations

- \* An erase operation (block addressable)
      - \* To set all bits to 1

- \* A program operation
      - \* Page addressable

- \* Erase and Program operations (logical write)

- \* 2 orders of magnitude slower than a read

- \* Logical writes should be avoided



UNIVERSIDADE  
FEDERAL DO CEARÁ

### 3. Memory System - NAND Flash Properties -

#### \* Wear Leveling

- \* Every time a block is written
  - \* Its lifetime is decreased
- \* Wear leveling aims at minimizing this effect by swapping intensely-used blocks with rarely-used ones
- \* This requires rewriting blocks, which is expensive



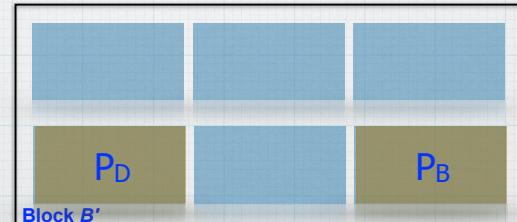
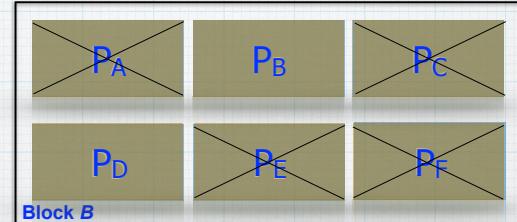
UNIVERSIDADE  
FEDERAL DO CEARÁ

### 3. Memory System - NAND Flash Properties -

#### \* Garbage Collection

- \* Responsible for "cleaning" blocks with stale pages

Erase  
Sets all bits to 1





UNIVERSIDADE  
FEDERAL DO CEARÁ

### 3. Memory System - NAND Flash Properties -

#### \* Write amplification

- \* Both the garbage collection and the wear leveling cause extra writes on SSDs
- \* The amount of actual physical writes on flash disks is thus much larger than the amount of logical writes required by applications



UNIVERSIDADE  
FEDERAL DO CEARÁ

### 3. Memory System - NAND Flash Properties -

#### \* Write endurance

- \* SSDs have lifetime determined by the number of erase/program cycles
  - \* Typically 100k cycles
- \* Low energy footprint
  - \* To perform read/write operations, only logical gates (circuitry) are involved;

### 3. Memory System

- Memory Types [Tertiary Memory] -

- \* Tertiary memory presents smaller cost per bit
  - \* Higher access time
  - \* Larger capacities
- \* Magnetic tapes
  - \* Sony tape
    - \* 185TB per cartridge
      - \* ≈ 3,700 Blu-ray discs
    - \* Dimensions
      - \* 102.0 x 105.4 x 21.5 mm

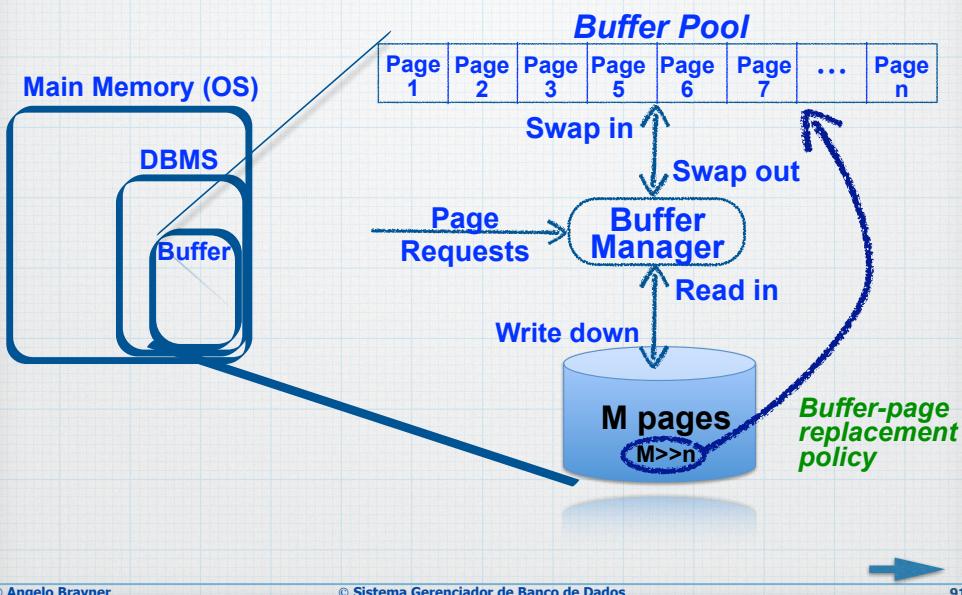


### 3. Memory System

- Memory Types [Tertiary Memory] -

- \* Oracle StorageTek SL8500 Modular Library System
  - \* <http://www.oracle.com/us/products/servers-storage/storage/tape-storage/sl8500-modular-library-system/overview/index.html>
  - \* <https://www.youtube.com/watch?v=cP90pgG--VI>

## 4. Buffer Management



## 4. Buffer Management

- \* Goal

- \* To hold on main-memory (buffer pool) the most referenced pages

- \* Reduces the amount of disk access

- \* Database Buffer Manager

- \* Buffer replacement policy

- \* Efficiency metric

- \* Hit Ratio

- \* Measures the probability of finding the desired page in the buffer pool

## 4. Buffer Management

- \* A taxa de acerto é determinada experimentalmente como descrito a seguir:
  - \* Um conjunto de programas (consultas) é executado
  - \* Os números de referências de endereços satisfeitas por M1 e M2 são registrados
    - \* Números denotados por N1 e N2
  - \* Taxa de acerto
    - \*  $H = N1 / (N1+N2)$
  - \* A taxa de acerto ótima deve tender a 1
    - \* Porque?

## 4. Buffer Management

- \* Buffer-page replacement policy
  - \* LRU (Least Recently Used)
    - \* Replaces the page which has not been referenced for the longest period of time
  - \* Recency
    - \* Buffer pages which have not been referenced for a long time are less likely to be accessed sooner
  - \* Temporal Locality

## 4. Buffer Management

- \* Buffer-page replacement policy
  - \* LRU
  - \* Counter implementation
    - \* The buffer manager has a counter  $C$
    - \* Whenever a page  $P$  is referenced
      - \*  $C$  is incremented
      - \* The new value for  $C$  is stored in  $P$  as well
    - \* The page with the lowest value for  $C$  is selected to be evicted
  - \* Disadvantage
    - \* The frequency and the scale of temporal locality may vary over time

## 4. Buffer Management

- \* Buffer-page replacement policy
  - \* LFU
    - \* Replaces the least frequency referenced buffer page
  - \* Frequency
    - \* Buffer pages which have less frequently been referenced are less likely to be accessed sooner

## 4. Buffer Management

- \* Buffer-page replacement policy
  - \* LFU
    - \* Counter implementation
      - \* A counter  $C$  is associated to each page in the buffer pool
      - \* Whenever a page  $P$  in the buffer pool is referenced
        - \*  $C$  is incremented
      - \* The page with the lowest value for  $C$  is selected to be evicted
    - \* LRU x LFU Efficiency
      - \* Depends on the workload

## 4. Buffer Management

- \* Buffer-page replacement policy
  - \* ARC
    - \* Dynamically adapts its behaviour
      - \* Recency x Frequency
    - \* Depending on workload features

## 4. Buffer Management

- \* Buffer-page replacement policy

- \* ARC

- \* Maintains LRU page lists:  $L_1$  and  $L_2$  (ghost buffers + buffer pages)
  - \*  $|L_1| + |L_2| = 2c$
  - \*  $P_i$  is in  $L_1$  if it has been referenced only once, since the last time  $P_i$  was evicted from  $L_1 \cup L_2$
  - \*  $P_k$  is in  $L_2$  if it has been seen at least twice, since the last time  $P_k$  was evicted from  $L_1 \cup L_2$
  - \*  $L_1$  captures recency and  $L_2$  frequency
  - \*  $L_1$  and  $L_2$  behave as a cache directory holding  $2c$  entries

## 4. Buffer Management

- \* Buffer-page replacement policy - ARC

- \*  $L_1$

- \*  $T_1$
      - \* Recent buffer entries
- \*  $B_1$
    - \* Entries (pages id) recently evicted from the  $T_1$  cache, but are still tracked

- \*  $L_2$

- \*  $T_2$
      - \* Buffer pages referenced at least twice
- \*  $B_2$
    - \* Entries (pages id) recently evicted from the  $T_2$ , but are still tracked



UNIVERSIDADE  
FEDERAL do CEARÁ

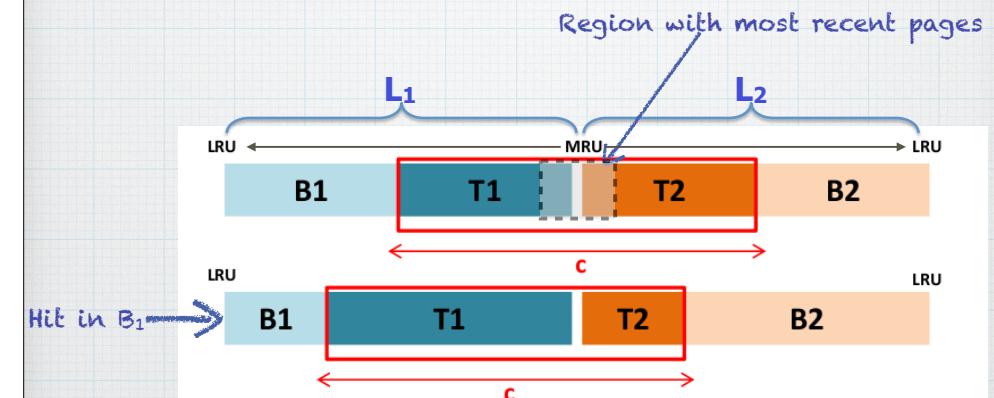
## 4. Buffer Management

- \* Buffer-page replacement policy - ARC
  - \* Pages in T<sub>1</sub> and T<sub>2</sub> reside in the cache directory and in the page cache
  - \* Pages in B<sub>1</sub> and B<sub>2</sub> reside only in the cache directory
  - \* Maintain historical data used to predict hit rates in T<sub>1</sub> and T<sub>2</sub>



## 4. Buffer Management

- \* Buffer-page replacement policy - ARC



## 5. Processamento de Transações

- O Problema de Concorrência em DBS -

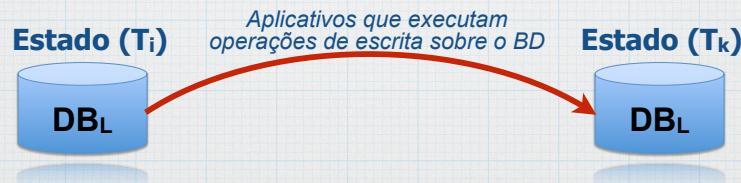
- \* Banco de Dados

- \* Estado

- \* Estado consistente

- \* Restrições de integridade são satisfeitas

- \* Transição de estado



## 5. Processamento de Transações

- O Problema de Concorrência em SBDs -

- \* Concorrência em um ambiente multiusuário

- \* Entrelaçamento de operações

- \* Operações de um programa podem ser executadas entre operações de outro programa

- \* Alterações inconsistentes no banco de dados

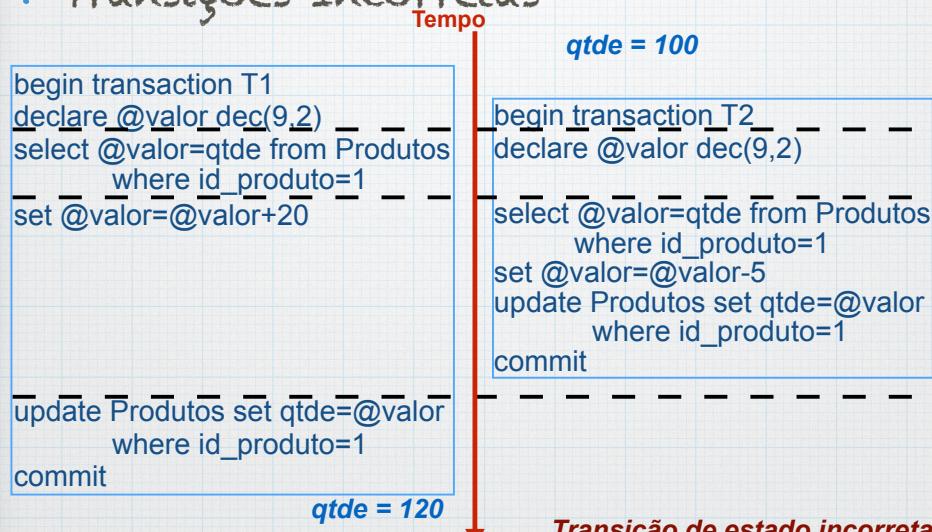
- \* Transições gerando estados inconsistentes do BD

- \* **Transições incorretas**

## 5. Processamento de Transações

- O Problema de Concorrência em SBDs -

### \* Transições Incorretas



## 5. Processamento de Transações

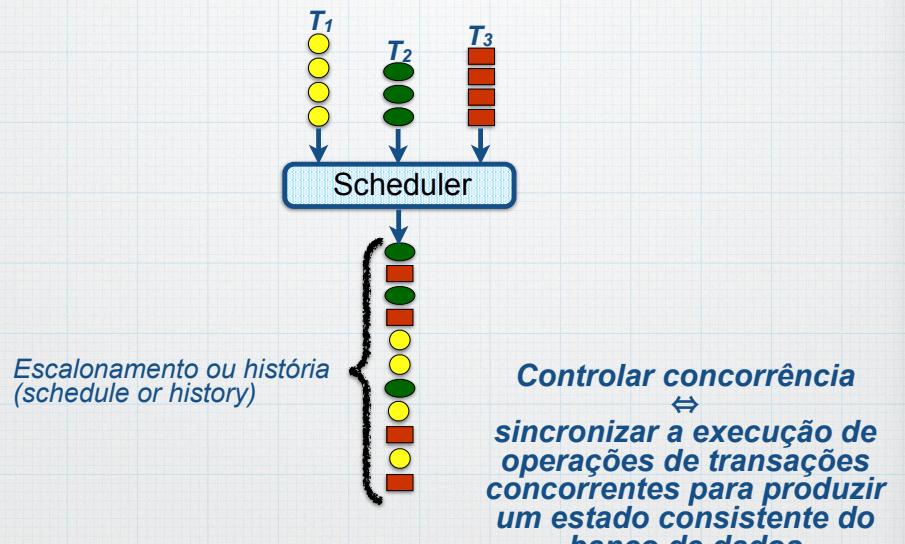
- O Problema de Concorrência em SBDs -

### \* Entrelaçamento de operações de programas distintos

- \* Pode provocar inconsistências no BD
- \* SGBD precisa monitorar e controlar
  - \* Execução concorrente de operações
- \* Controle de Concorrência
  - \* Escalonador (Scheduler)

## 5. Processamento de Transações

- O Problema de Concorrência em SBDs -



## 5. Processamento de Transações

- O Problema de Concorrência em SBDs -

- \* Transação

- \* Abstração que representa a sequência de operações sobre bancos de dados

- \* Resultantes da execução de um programa

- \* Seqüência de reads (r) e writes (w)

```
begin transaction T2
declare @valor dec(9,2)
select @valor=qtde from Produtos
where id_produto=1
set @valor=@valor-5
update Produtos set qtde=@valor
where id_produto=1
commit
```

- \*  $T_2 = r_2(id\_produto=1) w_2(id\_produto=1)$



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 5. Processamento de Transações

- Execução Correta de Transações -

### \* Premissa

- \* A execução de uma transação é correta, se executada isoladamente
- \* Produz sempre um estado consistente

### \* Teorema

- \* Toda execução serial de transações é correta
- \*  $S = T_2 T_5 T_1 T_4 T_3 \dots T_{n-1} T_n$
- \* Padrão para corretude de schedules
- \* Necessidade de um critério que garanta mais paralelismo



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 5. Processamento de Transações

- Execução Correta de Transações -

### \* Conjectura

- \* Seja um schedule  $S$  sobre  $s = \{T_1, T_2, \dots, T_n\}$
- \* Se  $S$  produz um estado do banco de dados igual ao produzido por alguma execução serial do conjunto de transações  $s$ 
  - \* A execução de  $S$  é correta

### \* Serializabilidade

- \* Ilusão que transações são executadas de forma isoladas
- \* Sem interferências de outras transações



## 5. Processamento de Transações

- Execução Correta de Transações

### \* Equivalência de schedules

- \* Dado um conjunto  $S = \{T_1, T_2, \dots, T_n\}$  de transações
  - \* Encontrar escalonamentos, cujas execuções produzam o mesmo estado no BD que a execução de algum schedule serial sobre  $S$

### \* Equivalências

- \* Estado final
- \* Visão
- \* Conflito

## 5. Processamento de Transações

- Execução Correta de Transações

### \* Operações em conflito

- \* Operações de transações distintas
- \* Executadas sobre um mesmo objeto e
- \* Pelo menos uma delas é uma operação de escrita

### \* Exemplo

- \*  $T_1 = r_1(x) w_1(x)$ ;  $T_2 = r_2(x) w_2(y)$
- \*  $S = r_2(x) r_1(x) w_2(y) w_1(x)$



### \* Relação de precedência

- \* Em  $S$ ,  $r_2(x) <_S w_1(x)$
- \*  $r_2(x)$  é escalonada antes de  $w_1(x)$



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 5. Processamento de Transações

- Execução Correta de Transações -

### \* Equivalência de conflito

- \* Dois Schedules  $S$  e  $S'$  sobre  $\Sigma = \{T_1, T_2, \dots, T_n\}$  são equivalentes de conflito ( $S \approx_C S'$ ) se, e somente se,
  - (i) têm as mesmas operações de transações em  $S$  e
  - (ii) Ordenam operações em conflito da mesma forma
    - \*  $\forall p_i \in OP(T_i), q_j \in OP(T_j), i \neq j, p_i$  e  $q_j$  em conflito,
      - \* Se  $p_i <_S q_j$ , então  $p_i <_{S'} q_j$

### \* Exemplo

- \*  $T_1 = r_1(y) r_1(x) w_1(y); \quad T_2 = r_2(x) r_2(y) w_2(x)$ 
  - \*  $S = r_1(y) r_1(x) r_2(x) w_1(y) r_2(y) w_2(x)$
  - \*  $S' = r_1(y) r_1(x) w_1(y) r_2(x) r_2(y) w_2(x)$



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Processamento de Transações

- Execução Correta de Transações -

### \* Serializabilidade por conflito (CSR)

- \* Um schedule  $S$  é serializável por conflito, se  $S$  é equivalente de conflito a algum schedule serial
- \* Identificar se um schedule é CSR pode ser feito em tempo polinomial

## 5. Processamento de Transações

- Execução Correta de Transações →

- \* Grafo de serialização de um schedule  $S$

- \*  $S$  está definido sobre um conjunto

 $S = \{T_1, T_2, \dots, T_n\}$  de transações

- \*  $SG(S) = (V, E)$

- \* Grafo direcionado

- \*  $V = S$

- \*  $T_i \rightarrow T_j \in E \Leftrightarrow$

- \*  $T_i, T_j \in S,$

- \*  $\exists p \in OP(T_i), q \in OP(T_j),$  tal que  $p$  conflita com  $q$  e  $p \ll q$

## 5. Processamento de Transações

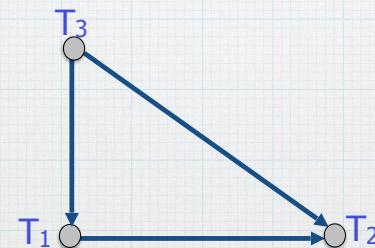
- Execução Correta de Transações →

- \* Exemplo

- \*  $S$  está definido sobre  $S = \{T_1, T_2, T_3\}$

 $S = r_3(y) r_1(y) r_1(x) r_2(x) w_1(y) r_2(y) r_3(x) w_2(x) w_3(z)$ 

- \*  $SG(S)$





## 5. Processamento de Transações

- Execução Correta de Transações -

### \* Teorema

\*  $S \in CSR \Leftrightarrow SG(S)$  não contém ciclos

### \* Verificar existência de ciclos em grafos

\* Resolvido em tempo polinomial

\*  $O(n^2)$ , onde  $n$  representa o número de transações

### \* Exercício

\* Verifique a corretude do seguinte schedule:

\*  $S' = r_3(y) r_1(y) r_1(x) r_2(x) w_1(y) r_2(y) w_2(x) r_3(x) w_3(z)$



## 5. Processamento de Transações

- Confiabilidade de Escalonamentos -

### \* Escalonamento rigoroso S

\* S é um schedule preciso e serializável por conflito

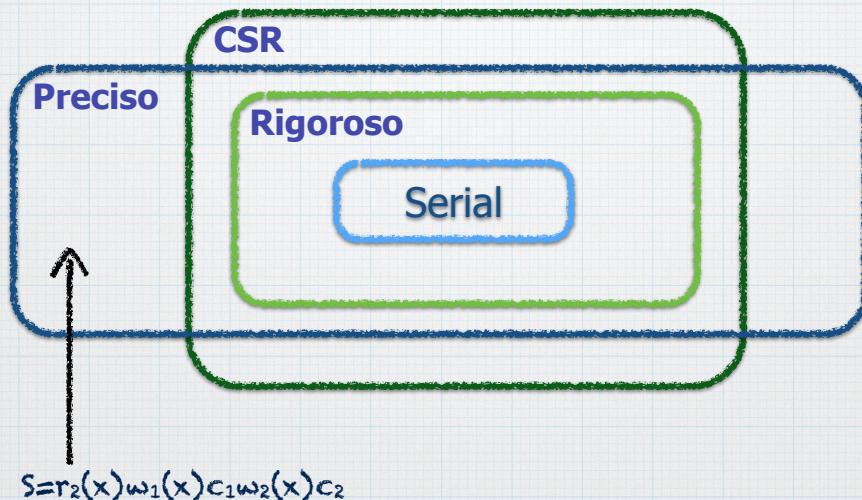
\* Garante corretude e grau de confiabilidade em uma única regra

### \* Regra para escalonamento rigoroso

\* Se  $p_i(x) <_S q_j(x)$ ,  $p_i(x)$  e  $q_j(x)$  estão em conflito  $\Rightarrow c_i <_S q_j(x)$  ou  $a_i <_S q_j(x)$

## 5. Processamento de Transações

- Execução Correta de Transações -



## 6. Protocolos para Controle de Concorrência

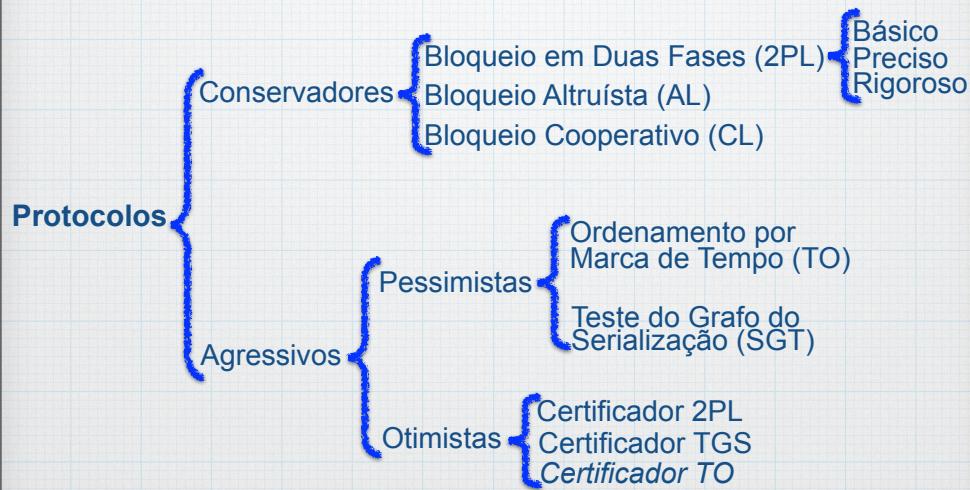
- Conceitos Básicos -

- \* Modelos de processamento de transações são implementados através de protocolos para o controle de concorrência
- \* Responsáveis pela produção de schedules com base em
  - \* Critério de corretude e um grau de confiabilidade
- \* Entrada: operações pertencentes a transações distintas
- \* Saída: Seqüência corretamente sincronizada
- \* Núcleo de um scheduler



## 6. Protocolos para Controle de Concorrência - Conceitos Básicos -

### \* Classificação



## 6. Protocolos para Controle de Concorrência - Conceitos Básicos -

### \* Conservadores

- \* Atrasam a execução de operações para sincronizar corretamente
- \* Baseados em mecanismos de bloqueio

### \* Agressivos

- \* Sincronizam operações imediatamente
- \* Pessimistas
  - \* Decidem se aceitam ou rejeitam a operação
    - \* Se rejeitam, abortam a transação

### \* Otimista

- \* Aceitam a operação imediatamente
- \* Periodicamente verificam a corretude do schedule produzido



## 6. Protocolos para Controle de Concorrência - Protocolos Conservadores -

- \* Protocolo de Bloqueio em Duas Fases – 2PL
  - \* Aquisição de bloqueio
    - \* Um tipo de bloqueio associado a cada objeto do banco de dados acessado por uma transação
    - \* Tipo de bloqueio é definido pela operação executada
      - \* Bloqueio de leitura (read lock - rl)
      - \* Bloqueio de escrita (write lock - wl)
  - \* SQL Server
    - \* syslockinfo
  - \* Liberação de bloqueio
    - \* Bloqueios de leitura e escrita

## 6. Protocolos para Controle de Concorrência - Protocolos Conservadores -

- \* Protocolo de Bloqueio em Duas Fases – 2PL
  - \* Duas Fases
    - \* Aquisição de bloqueios
    - \* Liberação de bloqueios



## 6. Protocolos para Controle de Concorrência

### - Protocolos Conservadores -

- \* Protocolo de Bloqueio em Duas Fases - 2PL
- \* Compatibilidade de bloqueios
  - \* Bloqueios compatíveis
  - \* Podem ser concedidos simultaneamente
  - \* Bloqueios compartilhados
- \* Bloqueios incompatíveis
- \* Bloqueios exclusivos

## 6. Protocolos para Controle de Concorrência

### - Protocolos Conservadores -

- \* Protocolo de Bloqueio em Duas Fases - 2PL
- \* Tabela de compatibilidade de bloqueios

	<i>Transação que solicita bloqueio</i>	<i>rl<sub>i</sub>(x)</i>	<i>wl<sub>i</sub>(x)</i>	<i>Transação que retém bloqueio</i>
<i>rl<sub>j</sub>(x)</i>	+	-		
<i>wl<sub>j</sub>(x)</i>	-	-		



## 6. Protocolos para Controle de Concorrência - Protocolos Conservadores -

### \* Protocolo 2PL Rigoroso

1. Escalonador recebe  $p_i \in \{r, w\}$  da transação  $T_i$

/\* Para escalarar  $p_i$ , precisa conceder bloqueio  $pl_i(x)$  \*/

Se existir bloqueio incompatível com  $pl_i(x)$

atrasa a execução de  $p_i(x)$ , até que  $pl_i(x)$  seja concedido

Senão

concede  $pl_i(x)$  e escalaona  $p_i(x)$

2. Todos os bloqueios só podem ser liberados durante execução da operação de **commit**.

### \* Produz escalonamento Rigoroso

\* Implementado pelos SGBDs existentes

## 6. Protocolos para Controle de Concorrência - Protocolos Conservadores -

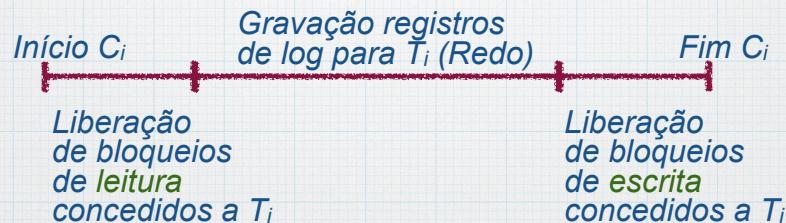




## 6. Protocolos para Controle de Concorrência - Protocolos Conservadores -

\* Execução de commit

\* Não é uma operação atômica



## 6. Protocolos para Controle de Concorrência - Deadlock -

\* Cenário  $\Rightarrow$  Escalonador 2PL Rigoroso

\*  $T_1 = r_1(x) w_1(x) c_1; T_2 = r_2(x) w_2(x) c_2$

$r_1(x) r_2(x) w_2(x) w_1(x)$



**2PL Rigoroso**

$S = r_1(x) r_2(x)$

\*  $T_2$  espera que  $T_1$  libere o bloqueio  $rl_1(x)$

\*  $T_1$  espera que  $T_2$  libere o bloqueio  $rl_2(x)$

\* **Deadlock**

\* Deadlock

\* Detecção e Resolução

\* Prevenção



## 6. Protocolos para Controle de Concorrência - Deadlock -

- \* Detecção de Deadlocks através do Grafo de Espera (Wait-For Graph)
- \* O scheduler deve manter um grafo (WFG)
- \* Nós representam as transações
- \* Arestras são construída da seguinte forma:
  - \* Se  $T_i$  está esperando que  $T_j$  libere algum bloqueio, então
    - \*  $T_i \rightarrow T_j$  deve ser inserida no grafo
  - \* Se a inclusão da aresta  $T_i \rightarrow T_j$  introduzir ciclo em WFG, então
    - \*  $T_i$  e  $T_j$  estão envolvidas em um deadlock



## 6. Protocolos para Controle de Concorrência - Deadlock -

### \* Grafo de Espera

$$* T_1 = r_1(x) w_1(x) c_1 \quad T_2 = r_2(x) w_2(x) c_2$$

$$* S = r_1(x) r_2(x)$$

$w_2(x)$   $w_1(x)$

$T_2$   $T_1$

Deadlock

### \* Resolução

\*  $T_2$  (mais recente) deve ser abortada (vítima)



## 6. Protocolos para Controle de Concorrência - Deadlock -

### \* Prevenção de Deadlocks

#### \* Bloqueio tipo update

- \* Bloqueio para leitura com a intenção de executar operação de escrita no futuro

- \* Ainda há necessidade de converter o bloqueio de update em bloqueio de escrita

- \* Especificado pelo usuário



## 6. Protocolos para Controle de Concorrência - Deadlock -

### \* Prevenção de Deadlocks

#### \* Tabela de Compatibilidade de bloqueios

#### \* Com bloqueio do tipo update

	$rl_i(x)$	$wl_i(x)$	$ul_i(x)$
$rl_j(x)$	+	-	-
$wl_j(x)$	-	-	-
$ul_j(x)$	+	-	-



## 6. Protocolos para Controle de Concorrência - Múltipla Granulosidade de Bloqueio -

- \* Importante parâmetro para performance do SGBD
- \* Granulosidade mais grossa
  - \* Diminui o overhead no gerenciamento de bloqueios
    - \* Menor número de bloqueios
  - \* Reduz o grau de concorrência
- \* Granulosidade mais fina
  - \* Alto overhead no gerenciamento de bloqueios
    - \* Maior número de bloqueios
      - \* Cada tupla na syslockinfo, ocupa 128 bytes (SQL Server)
  - \* Aumenta grau de concorrência

## 6. Protocolos para Controle de Concorrência - Múltipla Granulosidade de Bloqueio -

- \* 2PL com múltipla granulosidade

### \* Grafo de granulosidade

- \* Organização Lógica do BD





UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Protocolos para Controle de Concorrência - Múltipla Granulosidade de Bloqueio -

### \* 2PL com múltipla granulosidade

#### \* Funcionamento

- \* Bloqueio em um nó do grafo de granulosidade

- \* Descendentes são implicitamente bloqueados

- \* Mesmo tipo de bloqueio

- \* Propagar para os ascendentes os efeitos do bloqueio

- \* Bloqueio intencional

- \* Intencional de leitura - irl

- \* Intencional de escrita - iwl

- \* Intencional de update - iul



## 6. Protocolos para Controle de Concorrência - Múltipla Granulosidade de Bloqueio -

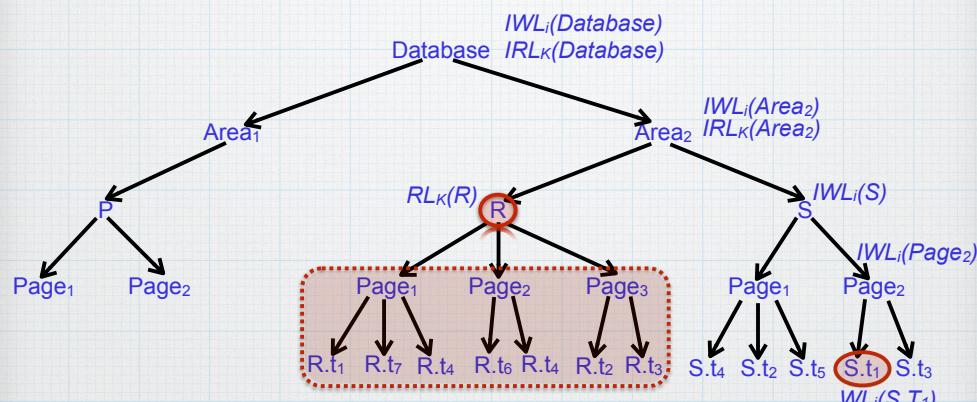
### \* 2PL com múltipla granulosidade

#### \* Tabela de compatibilidade com bloqueios intencionais

	$rl_i(x)$	$wl_i(x)$	$ul_i(x)$	$irl_i(x)$	$iwl_i(x)$	$iul_i(x)$
$rl_j(x)$	+	-	-	+	-	-
$wl_j(x)$	-	-	-	-	-	-
$ul_j(x)$	+	-	-	+	-	-
$irl_j(x)$	+	-	-	+	+	+
$iwl_j(x)$	-	-	-	+	+	+
$iul_j(x)$	+	-	-	+	+	+

## 6. Protocolos para Controle de Concorrência - Múltipla Granulosidade de Bloqueio -

\* 2PL com múltipla granulosidade



## 6. Protocolos para Controle de Concorrência - Múltipla Granulosidade de Bloqueio -

\* Atividade lab

- \* Execute o seguinte, para desligar a função de lock escalation
 

```
alter table nation
set (lock_escalation=disable)
```



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Protocolos para Controle de Concorrência - Múltipla Granulosidade de Bloqueio -

### \* Atividade Lab

#### \* Execute as seguintes transações

```
1 Begin transaction T1_2PL_R
declare @valor int
set transaction isolation level
repeatable read
select @valor=n_regionkey
from nation with (rowlock)
where n_nationkey=1
set @valor = @valor+0
update nation with (rowlock)
set n_regionkey=@valor
where n_nationkey=1
waitfor delay '00:00:40'
commit
```

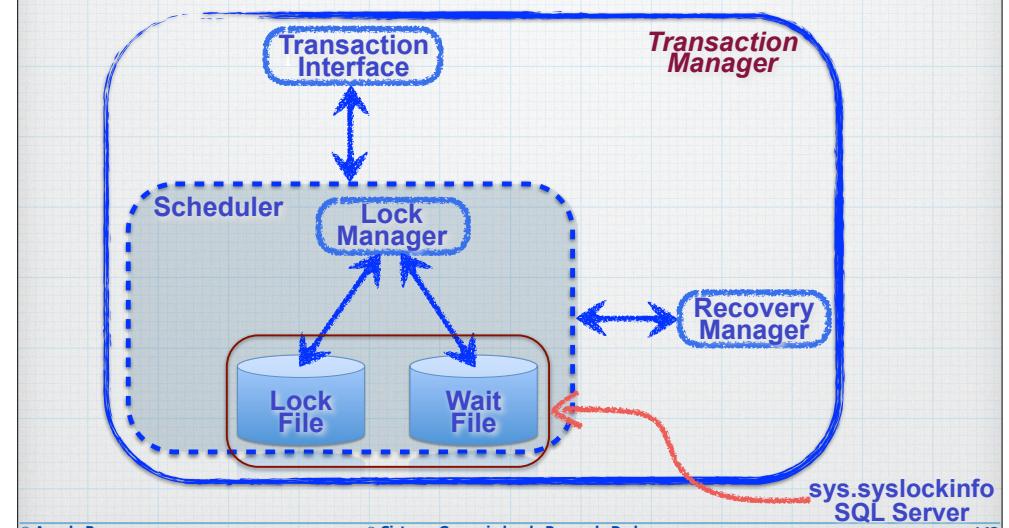
```
2 Begin transaction T2_2PL_R
set transaction isolation
level repeatable read
select * from nation with
(rowlock)
commit
```

```
3 select rsc_dbid, rsc_type,
req_mode, req_status,
req_transactionID
from syslockinfo
```

• Analise os bloqueios concedidos e em que tipos de objetos



## 6. Protocolos para Controle de Concorrência - Arquitetura Gerenciador de Transação 2PL -





UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Protocolos para Controle de Concorrência - Protocolos Agressivos -

- \* Não atrasam a execução de operações
  - \* Não são baseados em bloqueios
- \* Pessimista
  - \* Ordenamento por marca de tempo
  - \* Teste do Grafo de Serialização
- \* Otimistas
  - \* Certificação 2PL
  - \* Certificação TGS
  - \* Certificação TO



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Protocolos para Controle de Concorrência - Protocolos Agressivos [Otimistas] -

### \* Certificadores 2PL

- \* Quando o escalonador recebe o commit de  $T_i$  ( $c_i$ ), verifica

Se  $\exists p \in OP(T_i)$  e  $\exists q \in OP(T_j)$ , onde  $p$  conflita com  $q$ , e  $T_j$  é uma transação ativa

Abortar  $T_i$

Senão

Executar  $c_i$

## 6. Protocolos para Controle de Concorrência - Protocolos Agressivos [Otimistas] -

### \* Certificadores 2PL no SQL Server

- \* Para cada alteração em uma tupla (operação de escrita)
- \* Verifica se outra transação alterou valor lido da tupla a ser alterada
  - \* Compara valores de todas as colunas da tupla
  - \* Compara uma marca de tempo que pode se associada às tuplas de uma tabela

## 6. Protocolos para Controle de Concorrência - Protocolos Agressivos [Otimistas] -

### \* Protocolos não baseados em bloqueios

- \* Vantagens
  - \* Maior grau de concorrência, pois permitem mais entrelaçamentos
  - \* Não sofrem do problema de deadlocks
- \* Desvantagens
  - \* Alta taxa de aborts
  - \* Não garantem grau de confiabilidade
    - \* Deve-se implementar a regra correspondente ao grau a ser garantido



## 6. Protocolos para Controle de Concorrência - Mecanismo de Múltiplas Versões [Duas] -

### \* Princípio de Funcionamento

- \* Uma operação de escrita de T sobre um objeto  $x$  gera  $x_n$ 
  - \* Nova versão para  $x$
  - \* Só T "enxerga"  $x_n$
- \* Para executar a operação de escrita
  - \* T obtém primeiro um bloqueio de escrita sobre  $x$ 
    - \* Após obter  $wL(x)$ ,  $w(x)$  é escalonada (e executada)
      - \* T gera a versão  $x_n$
  - \* Qualquer outra transação pode ler a versão  $x$ 
    - \*  $x$  é a versão realmente armazenada no BD

## 6. Protocolos para Controle de Concorrência - Mecanismo de Múltiplas Versões [Duas] -

### \* Princípio de Funcionamento

- \* Quando T executa seu commit,  $x_n$  torna-se a versão  $x$

### \* Existente no banco de dados

### \* Redução da taxa de bloqueios

- \* Leituras sobre  $x$  não são atrasadas devido a uma escrita sobre  $x$
- \* Escritas sobre  $x$  não são atrasadas devido a uma leitura sobre  $x$

## 7. Protocolos para Controle de Concorrência - Mecanismo de Múltiplas Versões [Duas] -

### \* Protocolo 2V2PL (two version 2PL)

#### \* Tabela de compatibilidade

##### \* Novo tipo de bloqueio

##### \* Certify Lock

	$rl_i(x)$	$wl_i(x)$	$cl_i(x)$
$rl_j(x)$	+	+	-
$wl_j(x)$	+	-	-
$cl_j(x)$	-	-	-

Certify Lock

## 6. Protocolos para Controle de Concorrência - Mecanismo de Múltiplas Versões [Duas] -

### \* Protocolo 2V2PL

#### 1. Scheduler recebe $wj(x)$

Tenta obter  $wlj(x)$

Se existe  $wlk(x)$  ou  $clk(x)$ , então  
 Aguarda a concessão de  $wlj(x)$  e não  
 escalona  $wj(x)$

Senão

Concede o bloqueio  $wlj(x)$

Converte  $wj(x)$  em  $wj(x_n)$

Escalona  $wj(x_n)$



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Protocolos para Controle de Concorrência - Mecanismo de Múltiplas Versões [Duas] -

### \* Protocolo 2V2PL

2. Scheduler recebe  $r_j(x)$

Tenta obter  $rl_j(x)$

Se existe  $cl_k(x)$ , então

Aguarda a concessão de  $rl_j(x)$  e não  
escalona  $r_j(x)$

Senão

Concede o bloqueio  $rl_j(x)$

Se  $T_j$  possuía  $wl_j(x)$  // executou  $w_j(x_n)$

Converte  $r_j(x)$  em  $r_j(x_n)$

Escalona  $r_j(x_n)$

Senão

Escalona  $r_j(x)$



## 7. Protocolos para Controle de Concorrência - Mecanismo de Múltiplas Versões [Duas] -

### \* Protocolo 2V2PL

3. Scheduler recebe a operação  $c_j$

// Tenta Converter todos  $wl_j$  em  $cl_j$

Enquanto houver  $wl_j(x)$ , faça

Se existir  $rl_k(x)$ , com  $0 \leq k \leq n$ ,  $k \neq j$

Aguarda a concessão de  $cl_j(x)$

Senão

Concede o bloqueio  $cl_j(x)$

Fim-enquanto

Escalona  $c_j$



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Protocolos para Controle de Concorrência - Mecanismo de Múltiplas Versões [Duas] -

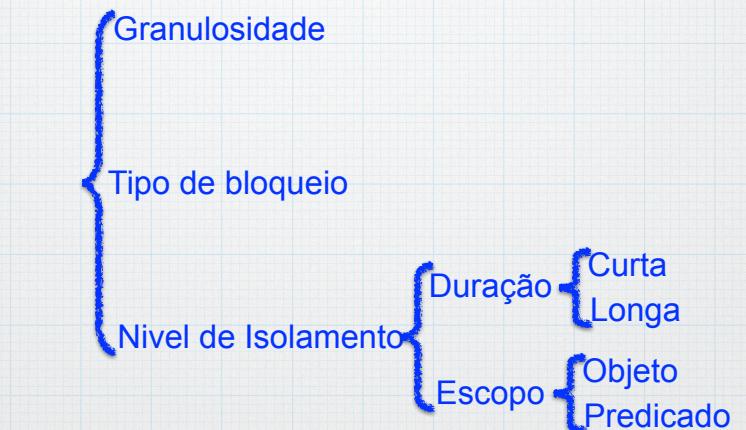
- \* Produz schedules serializáveis por conflito e precisos
- \* Maior grau de paralelismo
- \* Muitos DBMSs existentes já implementam
  - \* Oracle (read consistency isolation)
  - \* SQL Server (snapshot)



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

### \* Bloqueios em SQL





## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

### \* Granulosidade de bloqueios

- \* DB2
- \* Objetos
  - \* Tablespace
  - \* Table
  - \* Page
  - \* Row
- \* Definição
  - \* `create tablespace ... locksize <objeto | any>`
  - \* `any`
  - \* Escalonamento dinâmico pelo DB2
    - \* Tablespace ↔ page



## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

### \* Granulosidade de Bloqueios

- \* SQL Server
- \* Objetos
  - \* Banco de dados
  - \* Arquivo
  - \* Tabela
  - \* Estrutura de índice
  - \* Extensão (extent)
    - \* Conjunto de oito páginas contíguas



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

### \* Granulosidade de Bloqueios

#### \* SQL Server

- \* Objetos

- \* Página

- \* Chave de busca (key)

- \* Utilizado para bloquear tuplas pela estrutura de índice

- \* Intervalo de chave de busca

- \* Tupla (RID)



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

### \* Granulosidade de bloqueios

#### \* SQL Server

- \* Definição

- \* No código SQL da transação

- \* Escalonamento automático de granulosidade de bloqueio

#### \* Oracle

- \* Não implementa múltipla granulosidade de bloqueio

## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

### \* Modo de bloqueio

#### \* Leitura (share - S)

- \* Definido automaticamente pelo tipo de operação

#### \* Escrita (exclusive - X)

- \* Definido automaticamente pelo tipo de operação

#### \* Especificado pelo usuário

#### \* Leitura com intenção de escrita (update - U)

#### \* Especificado pelo usuário

## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

### \* Nível de isolamento

#### \* Escopo de bloqueio

##### \* Objeto

- \* Bloqueio sobre tuplas existentes na tabela

##### \* Predicado

- \* Bloqueio sobre tuplas que satisfazem um predicado

- \* Tuplas existentes no banco de dados

- \* Tuplas ainda não existentes

- \* Satisfariam o predicado se incluídas no banco de dados

- \* Tuplas a serem alteradas e que passariam a satisfazer o predicado após a alteração

## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

### \* Nível de isolamento

#### \* Duração do bloqueio

- \* Longa

- \* O bloqueio é retido até após o commit ou abort

- \* Curta

- \* O bloqueio é liberado imediatamente após a execução da operação associada ao bloqueio

## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

### \* Níveis de Isolamento

#### \* read uncommitted

- \* Não existe bloqueio de leitura

- \* Bloqueios de escrita

- \* Curta duração tanto para objetos como predicados

#### \* read committed Default: Oracle e SQL Server

- \* Bloqueios de leitura

- \* Curta duração tanto para objetos como predicados

- \* Bloqueios de escrita

- \* Longa duração tanto para objetos como predicados



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 7. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

### \* Níveis de Isolamento

#### \* read committed

##### \* Anomalias

###### \* Nonrepeatable read

$$S_1 = r_2(x) w_1(x, 2) c_1 r_2(x) c_2$$

\* Transação T<sub>2</sub> está lendo valores diferentes de x

\* Deve ser evitado !

##### \* Lost update

$$S_2 = r_1(x) r_2(x) w_2(x, x+10) c_2 w_1(x, x+20) r_1(z) c_1$$

\* Operação de escrita w<sub>2</sub>(x) foi sobreposta por w<sub>1</sub>(x)



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

### \* Níveis de Isolamento

#### \* cursor stability

##### \* Bloqueios de leitura

\* Curta duração para objetos e predicados

\* Curta duração Cursor/Resultset

\* Bloqueio no item corrente do cursor é mantido até

\* Posição no Cursor/Resultset seja alterada

\* Cursor/ResultSet seja fechado

##### \* Bloqueios de escrita

\* Longa duração para objetos e predicados



## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

### \* Níveis de Isolamento

#### \* cursor stability

##### \* Evita a anomalia lost update

\*  $S_2 = r_1(x) r_2(x) \underbrace{w_2(x, x+10)}_{C_2} c_2 w_1(x, x+20) r_1(z) c_1$

#### \* SGBDs que implementam

##### \* DB2

##### \* Nível de isolamento

##### \* SQL Server

##### \* Na cláusula DECLARE CURSOR Opção SCROLL\_LOCKS



## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

### \* Níveis de Isolamento

#### \* repeatable read

##### \* Bloqueios de leitura

##### \* Longa duração para objetos

##### \* Curta duração para predicados

##### \* Bloqueios de escrita

##### \* Longa duração para objetos e predicados

#### \* Anomalia

##### \* Fenômeno Fantasma

##### \* T acessa tuplas que satisfazem um predicho P

##### \* Tuplas existentes no BD e que satisfazem P são bloqueados

##### \* Novas tuplas que satisfazem P podem ser inseridas

## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

### \* Níveis de Isolamento

#### \* serializable

##### \* Bloqueios de Leitura

- \* Longa duração para objetos e predicados

##### \* Bloqueios de escrita

- \* Longa duração tanto para objetos como predicados

## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

### \* Sintaxe SQL99

* SET TRANSACTION	READ ISOLATION LEVEL	READ UNCOMMITTED
	READ WRITE	READ COMMITTED
		CURSOR STABILITY
		REPEATABLE READ
		SERIALIZABLE

DB2



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

### \* Sintaxe Oracle

\* SET TRANSACTION ISOLATION LEVEL | READ COMMITTED  
| SERIALIZABLE  
READ  
READ WRITE

READ

Transação não pode alterar o banco de dados  
Assume Transaction-level Read Consistency (default)

READ WRITE

Transação pode alterar o banco de dados  
Assume Transaction-level Read Consistency (default)



## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

### \* Sintaxe SQL Server (>2005)

\* SET TRANSACTION ISOLATION LEVEL | READ UNCOMMITTED  
| READ COMMITTED  
| REPEATABLE READ  
| SNAPSHOT  
| SERIALIZABLE

### \* Snapshot

- \* Implementa o protocolo 2V2PL
- \* Necessário ligar opção  
ALLOW\_SNAPSHOT\_ISOLATION



## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

- \* Sintaxe SQL Server (versão > 2005)
  - \* Protocolo otimista (Cursos - declare cursor)
  - \* OPTIMISTIC WITH VALUES
    - \* Ao executar uma operação de escrita
      - \* Escalonado verifica se valores atuais de todas colunas da tupla são os mesmos de quando a tupla foi lida
      - \* Se não for verdade, a transação é abortada
  - \* OPTIMISTIC WITH ROW VERSIONING
    - \* Compara valores da coluna timestamp
    - \* Criar tabela com esta coluna
  - \* Ligar opção READ\_COMMITTED\_SNAPSHOT
    - \* Nível de isolamento read committed



## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

- \* Escolha do nível de isolamento
  - \* Importante decisão de projeto
  - \* Decidir entre
    - \* Garantir consistência do banco de dados
    - \* Garantir um alto grau de concorrência



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

### \* Atividade lab

- \* Execute o seguinte comando DDL  
`alter table nation  
set (lock_escalation=disable)`

### \* Execute a transação

```
Begin transaction T1  
set transaction isolation level repeatable read  
select * from nation with (rowlock)  
waitfor delay '00:00:20'  
commit
```

### \* Dispare a consulta

```
select rsc_dbid, rsc_type, req_mode,  
req_transactionID  
from syslockinfo
```

### \* Analise os bloqueios concedidos e em que tipos de objetos



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

### \* Atividade lab

- \* Execute a transação

```
Begin transaction T1  
set transaction isolation level repeatable read  
select * from nation with (updlock, rowlock)  
waitfor delay '00:00:40'  
commit
```

### \* Dispare a consulta

```
select rsc_dbid, rsc_type, req_mode,  
req_transactionID  
from syslockinfo
```

### \* Analise os bloqueios concedidos e em que tipos de objetos



## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

### \* Atividade Lab

- \* Execute o seguinte comando DDL  

```
alter table LineItem  
set (lock_escalation=disable)
```

- \* Execute a transação

```
Begin transaction T1  
set transaction isolation level repeatable read  
select * from LineItem with (rowlock)  
waitfor delay '00:00:10'  
commit
```

- \* Dispare a consulta

```
select rsc_dbid, rsc_type, req_mode,  
req_transactionID  
from syslockinfo
```

- \* Analise os bloqueios concedidos e em que tipos de  
objetos



## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

### \* Atividade Lab

- \* Execute a transação

```
Begin transaction T1  
set transaction isolation level repeatable read  
select * from LineItem  
waitfor delay '00:00:10'  
commit
```

- \* Dispare a consulta

```
select rsc_dbid, rsc_type, req_mode,  
req_transactionID  
from syslockinfo
```

- \* Analise os bloqueios concedidos e em que  
tipos de objetos



## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

### \* Atividade Lab

#### \* Execute a transação

```
begin transaction T1
set transaction isolation level repeatable read
select * from lineitem with (rowlock)
waitfor delay '00:00:30'
update lineitem with (rowlock)
set l_quantity=l_quantity
commit
```

#### \* Dispare a transação

```
begin transaction T2
set transaction isolation level repeatable read
select * from lineitem with (tablock)
update lineitem with (tablock)
set l_quantity=l_quantity
commit
```

#### \* Analise o que ocorre durante a execução das duas transações



## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

### \* Atividade Lab A

#### \* Execute a transação

```
begin transaction T1
set transaction isolation level repeatable read
update lineitem with (paglock)
set l_quantity=l_quantity
waitfor delay '00:00:30'
commit
```

#### \* Dispare a transação

```
begin transaction T2
set transaction isolation level repeatable read
select * from lineitem with (paglock)
commit
```

#### \* Analise o que ocorre durante a execução das duas transações



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Protocolos para Controle de Concorrência - Controle de Concorrência em SQL -

### \* Atividade Lab B

#### \* Execute a transação

```
alter database tpc_h set allow_snapshot_isolation on
begin transaction T1
set transaction isolation level snapshot
update lineitem with (paglock)
set l_quantity=l_quantity
waitfor delay '00:00:30'
commit
```

#### \* Dispare a transação

```
alter database tpc_h set allow_snapshot_isolation on
begin transaction T2
set transaction isolation level snapshot
select * from lineitem with (paglock)
commit
```

#### \* Analise o que ocorre durante a execução das duas transações



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 7. Recuperação e Logging - Conceitos Básicos -

### \* Recuperação

- \* Manipulação automática pelo SGBD de todos os tipos de falhas esperados

### \* Condição básica

- \* Armazenamento de informação redundante durante o funcionamento normal (logging)

### \* Paradigma de Transação

- \* Unidade de recuperação (propriedade tudo ou nada)
- \* Persistência após uma execução com sucesso

### \* Estado do BD que deve ser recuperado

- \* Estado consistente mais recente antes da falha



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 7. Recuperação e Logging - Conceitos Básicos -

### \* UNDO

- \* Desfazer por completo os efeitos das operações das transações que não haviam executado commit no momento da falha
  - \* Alterações já tenham sido transferidos para disco
  - \* Até o início da transação
- \* Savepoint
  - \* Desfazer o efeito das operações até um determinado ponto dentro da transação
    - \* Oracle, SQL Server, DB2, Postgres, MySQL



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 7. Recuperação e Logging - Conceitos Básicos -

### \* REDO

- \* Refazer por completo os efeitos de transações que já haviam executado commit no momento da falha
- \* Alterações ainda não haviam sido descarregadas para disco
- \* Banco de Dados

## 7. Recuperação e Logging - Tipos de Falhas -

- \* Falha de uma Transação ativa (abort)
  - \* UNDO
- \* Falha de sistema
  - \* Recuperar o mais recente estado consistente do BD que existia antes da ocorrência da falha
  - \* UNDO
  - \* REDO

## 7. Recuperação e Logging - Tipos de Falhas -

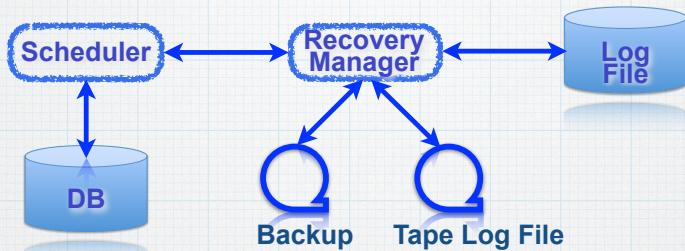
- \* Falha de Media (recuperação de disco)
  - \* Refazer o BD utilizando o último backup
  - \* RAID
- \* Catástrofe
  - \* Uma ambiente computacional espelho



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 7. Recuperação e Logging

### - Arquitetura do Gerenciador de Recuperação -



#### \* Disk Log File

- \* Transaction failure
- \* System failure
  - \*  $DB + Disk\ Log\ File \Rightarrow DB$

#### \* Media failure (long duration)

- \*  $Backup + Tape\ Log\ File + Disk\ Log\ File \Rightarrow DB$



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 7. Recuperação e Logging

### - Técnicas de Logging -

#### \* Logging Físico

- \* Before images e after images de páginas alteradas são armazenados no arquivo de log

#### \* Granulosidade do log

##### \* Página

- \* Se apenas uma tupla da página p é alterada
  - \* Before image e after image de p são armazenados no log

#### \* UNDO

- \* Assumir valores do before image

#### \* REDO

- \* Assumir valores do after image

## 7. Recuperação e Logging - Técnicas de Logging -

### \* Logging Físico

- \* Processo de recovery rápido e fácil de ser implementado
- \* Exige muito espaço para o arquivo de log
- \* Utilizado pela maioria dos DBMSs existentes

## 7. Recuperação e Logging - Técnicas de Logging -

### \* Logging Lógico

- \* Comandos DML (update, insert, delete) com seus parâmetros são protocolados no arquivo de log
- \* UNDO
  - \* Executar operação inversa da armazenada no arquivo de log
    - \* Executar delete para cada insert no log
    - \* Um update equivale
      - \* Delete tupla com valor antigo + Insert tupla com novo valor
- \* REDO
  - \* Executar todos os comandos DML protocolados no log

## 7. Recuperação e Logging - Técnicas de Logging -

### \* Logging Lógico

- \* Utiliza menos espaço no arquivo de log
- \* Processo de recovery complexo e longo

## 7. Recuperação e Logging - Técnicas de Logging -

### \* Logging Físico-Lógico

- \* Granulosidade do log (Físico)
  - \* Página
- \* Para cada página (Lógico)
  - \* Armazenar alterações em tuplas de uma página como comandos DML
- \* UNDO
  - \* Executar operação inversa da armazenada no log
- \* REDO
  - \* Executar todos os comandos DML protocolados no log

## 7. Recuperação e Logging - Técnicas de Logging -

### \* Logging Físico-lógico

- \* Executar operações de UNDO e REDO por página
- \* Menor número de execuções que Logging Lógico
  - \* Checkpoint e teste para verificar necessidade de REDO

## 7. Recuperação e Logging - Técnicas de Logging -

### \* Estratégias para identificar o momento de gravar registros de log

- \* Estratégia WAL (Write-Ahead-Log)
  - \* Antes de transferir uma página P da área de buffer para o BD, gravar no arquivo de log registros referentes às alterações sobre P
- \* Estratégia Force-Log-at-Commit
  - \* Antes de terminar a execução da operação de commit de uma transação T, gravar no arquivo de log registros referentes às alterações executadas por T em páginas ainda residente em buffer

## 7. Recuperação e Logging - Técnicas de Logging -

### \* Estratégia de troca de página no buffer

#### \* Steal

- \* Páginas alteradas podem ser trocadas e descarregadas no BD a qualquer momento
  - \* mesmo antes do EOT
- \* Maior flexibilidade para a política de substituição de páginas
- \* Necessário UNDO

#### \* NoSteal:

- \* Páginas alteradas são mantidas no buffer até o EOT
- \* Não é necessário UNDO
- \* Problemas com transações longas

## 8. Recuperação e Logging - Técnicas de Logging -

### \* Estratégia de processamento de EOT (commit)

#### \* Force

- \* Páginas alteradas são descarregadas no BD durante o processamento de EOT (transação que fez a alteração)
- \* Não é necessário REDO
- \* Muitas escritas em disco são necessárias

#### \* NoForce:

- \* Durante o processamento de EOT, nenhuma propagação de página é disparada
- \* Pouco overhead com escritas em disco durante EOT
- \* É necessário REDO



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 7. Recuperação e Logging - Técnicas de Logging -

### \* Bloqueio

\* Granularidade de log ≤ Granularidade de bloqueio

\* Porque?

\* Slide 229



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 7. Recuperação e Logging

### \* Atividade lab (rollback/savepoint)

\* Execute a seguinte transação

```
begin transaction T1
declare @valor int
select @valor=qtdc from Produtos
where id_produto=1
set @valor=@valor+20
update Produtos set qtdc=@valor where id_produto=1
save transaction SV1 -- Definição savepoint SV1
select @valor=qtdc from Produtos
where id_produto=1
set @valor=@valor-10
update Produtos set qtdc=@valor where id_produto=1
save transaction SV2 -- Definição savepoint SV2
select @valor=qtdc from Produtos
where id_produto=1
print @valor
rollback transaction SV1
commit
```

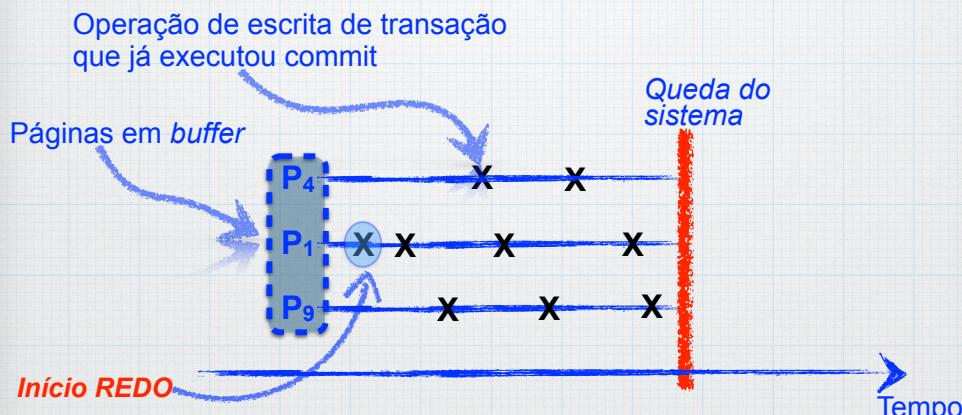
\* Descreva as ações que foram executadas pelo SGBD

\* Verifique o estado do banco de dados gerado



## 7. Recuperação e Logging - Checkpoint -

- \* Mecanismo para limitar a quantidade de REDOs após falha do sistema



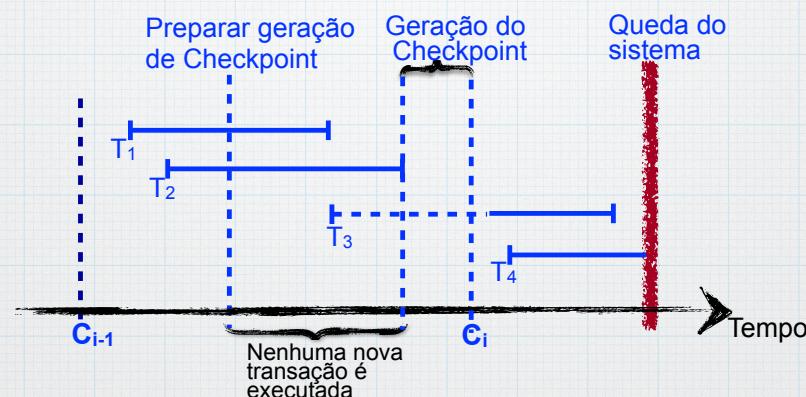
## 7. Recuperação e Logging - Checkpoint -

- \* Recuperação
  - \* REDO precisa retornar muito longe no arquivo de log
- \* Checkpoint
  - \* Armazenamento de informação no arquivo de log
  - \* Reduzir e limitar a quantidade de REDOs após uma falha
- \* Onde armazenar??
  - \* Diferentes estratégias
    - \* Gravar no arquivo de log
    - \* Gravar no arquivo de log e no BD



## 7. Recuperação e Logging - Checkpoint -

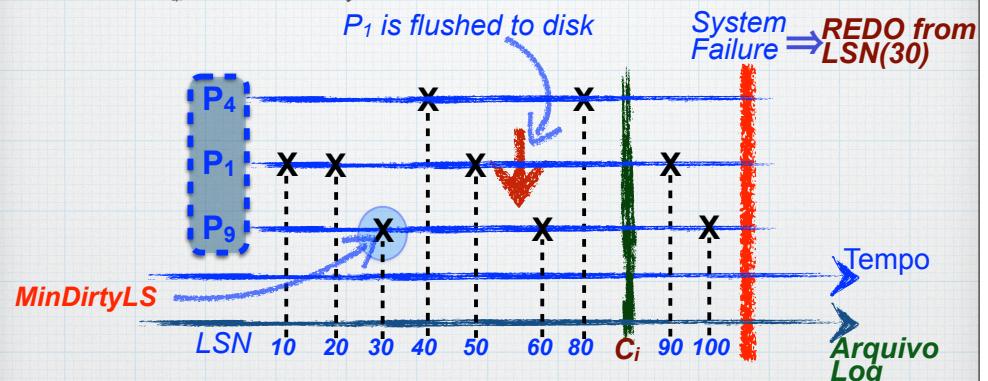
### \* Transaction Consistent Checkpoint (TCC)



\* REDO para  $T_3$

## 7. Recuperação e Logging - Checkpoint -

### \* Fuzzy Checkpoint



- Gravar registro de log do tipo **Checkpoint**
  - Menor LSN de operação de atualização sobre páginas em buffer (MinDirtyPageLSN).
  - Desconsidera-se LSNs de operações sobre páginas já descarregadas

## 7. Recuperação e Logging - Checkpoint -

### \* Fuzzy Checkpoint (cont.)

- \* Identificação da posição de início das operações de REDO
  - \* Encontrar no log o registro do último checkpoint
  - \* Identificar o MinDirtyPageLSN
  - \* Iniciar REDO a partir deste ponto
- \* Descarga de alterações é realizada de forma assíncrona com relação a geração de checkpoint

## 7. Recuperação e Logging - Checkpoint -

### \* Estratégia de Checkpoint SQL Server

- \* Descarga para disco de todas páginas alteradas e residentes na área de buffer
- \* Independentemente da alteração ter sido feita por transação "committed" ou não
- \* Aplica a regra WAL

## 7. Recuperação e Logging - Arquivo de log -

### \* Arquivo sequencial

- \* Cada registro de log é inserido no final do arquivo
- \* Captura ordem temporal de execução das operações de transações concorrentes

### \* Independente do Banco de Dados

- \* Replicado

## 7. Recuperação e Logging - Arquivo de log -

### \* Tipos de registros de log

- \* Begin-of-Transaction (BOT)
- \* Operação de escrita
  - \* Informação de UNDO (before image) e de REDO (after image)
- \* Commit ou abort
- \* Checkpoint
  - \* Begin-Checkpoint
  - \* Dados do Checkpoint
  - \* End-Checkpoint

## 7. Recuperação e Logging - Arquivo de log -

### \* Estrutura de um registro de log

- \* Log Sequence Number (LSN)
  - \* Identifica univocamente um registro de log
  - \* Cresce de forma monotônica
    - \* Capturar a ordem cronológica do registro de log
    - \* Refletir a ordem em que as alterações ocorreram
- \* TRID
  - \* Transaction ID
- \* PageID
  - \* Endereço da página alterado em disco

## 7. Recuperação e Logging - Arquivo de log -

### \* Estrutura de um registro de log

- \* Se a estratégia é logging físico
  - \* Before image
  - \* After image
- \* Se a estratégia é logging físico-lógico ou lógico
  - \* Comandos DML (insert, delete, update)
- \* PrevLSN da mesma transação
  - \* Lista encadeada para trás
- \* Para cada transação
  - \* Vários registros de log
    - \* BOT, um para alteração sobre o BD, savepoint, commit/rollback



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 7. Recuperação e Logging - Arquivo de log [Reg Log SQL Server] -

Column	Description
Current LSN	Current Log Sequence Number
Previous LSN	Previous Log Sequence Number
Operation	Describes the operations performed
Context	Context of the operation
Transaction ID	ID of the transaction in the LOG file
Log Record Length	Size of the row in bytes
AllocUnitName	Object name (table or index) against which the operation was performed
Page ID	Table/Index Page ID
SPID	User Session ID
Xact ID	User Transaction ID – logged only in the LOP_BEGIN_XACT and LOP_COMMIT_XACT rows
Begin Time	Transaction Start time - logged only in the LOP_BEGIN_XACT record
End Time	End Time - logged only in the LOP_COMMIT_XACT record
Transaction Name	Typically refers to the type of transaction: INSERT for example - logged only in the LOP_BEGIN_XACT record
Transaction SID	User Security identifier
Parent Transaction ID	If is a child transaction, will contain the ID of its parent transaction
Transaction Begin	The first LSN of the transaction
Number of Locks	Number of locks
Lock Information	Description of the lock
Description	Transaction LOG row description
Log Record	The hexadecimal content of the transaction, an inserted/deleted row or the content of a page i.e.



## 7. Recuperação e Logging - Arquivo de log [Reg Log SQL Server] -

- \* Tipos de operação
- \* Campo Operation

Operation	Description
LOP_BEGIN_XACT	Begin Transaction
LOP_COMMIT_XACT	End Transaction
LOP_FORMAT_PAGE	Page Modified
LOP_INSERT_ROWS	Row inserted
LOP_DELETE_ROWS	Row deleted
LOP_LOCK_XACT	Lock
LOP_MODIFY_ROW	Row Modified
LOP_MODIFY_COLUMNS	Column Modified
LOP_XACT_CKPT	Checkpoint
LOP_BEGIN_CKPT	Checkpoint start
LOP_END_CKPT	Checkpoint end
LOP_MARK_SAVEPOINT	Save point

## 7. Recuperação e Logging - Arquivo de log -

### \* Atividade de lab

- \* No BD Estoque, execute a seguinte consulta

```
select [Current LSN], Operation, [Page ID], [Transaction ID],[Lock Information], [Begin Time], [End Time], [Log Record], [Previous LSN] from fn_dblog(null,NULL)
```

## 7. Recuperação e Logging - Arquivo de log -

### \* Atividade de lab

- \* Execute a seguinte transação

```
begin transaction Teste_Arq_Log
declare @valor int
select @valor=qtdc from Produtos
where id_produto=1
set @valor=@valor+20
update Produtos set qtdc=@valor where id_produto=1
save transaction SV1 -- Definição savepoint SV1
select @valor=qtdc from Produtos
where id_produto=1
set @valor=@valor-10
update Produtos set qtdc=@valor where id_produto=1
save transaction SV2 -- Definição savepoint SV2
select @valor=qtdc from Produtos
where id_produto=1
print @valor
rollback transaction SV1
commit
```

- \* Execute novamente a consulta para acessar o arquivo de log  
analice os novos registro inseridos no arquivo de log



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 7. Recuperação e Logging - Arquivo de log -

### \* Atividade de lab

- \* Execute a consulta para ler o arquivo de log
- \* Execute a seguinte transação

```
begin transaction Teste_Arq_Log
declare @valor int
select @valor=qtde from Produtos
where id_produto=1
set @valor=@valor+20
update Produtos set qtde=@valor where id_produto=1
save transaction SV1 -- Definição savepoint SV1
select @valor=qtde from Produtos
where id_produto=1
set @valor=@valor-10
update Produtos set qtde=@valor where id_produto=1
save transaction SV2 -- Definição savepoint SV2
if @valor=@valor
    rollback
else
    commit
```

- \* Execute novamente a consulta para acessar o arquivo de log  
analise os novos registro inseridos no arquivo de log



## 7. Recuperação e Logging - Procedimentos de Recuperação -

### \* Quando executar REDO e UNDO ?

- \* Registros de Log de transações que executaram commit antes da falha
  - \* Transações vencedoras
  - \* Podem precisar de REDO
  - \* Quais registros ?
    - \* Protocolam operações de alterações em páginas
- \* Registros de Log de transações que não executaram commit
  - \* Transações perdedoras
  - \* Podem precisar de UNDO
  - \* Quais registros ?
    - \* Protocolam operações de alterações em páginas



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 7. Recuperação e Logging - Procedimentos de Recuperação -

- \* Premissa para identificar quando executar REDO/UNDO
- \* Quando uma página P é propagada para disco
  - \* LSN do último registro de log referente à atualização de p
    - \* Gravado no cabeçalho da página p
    - \* **LSN(P)**



## 7. Recuperação e Logging - Procedimentos de Recuperação -

- \* Quando executar um REDO ?
  - \* LSN da página < LSN do registro de log para REDO
    - \*  $\text{PageLSN}(p) \leftarrow \text{LSN do registro}$
- \* Quando executar um UNDO ?
  - \* LSN da página  $\geq$  LSN do registro de log para UNDO
    - \*  $\text{PageLSN}(p) \leftarrow \text{LSN do registro}$

## 7. Recuperação e Logging - Procedimentos de Recuperação -

### \* Exemplo

- \* Considere duas transações  $T_1$  e  $T_2$  que atualizam objetos de  $P$



#### ■ Ações de Recuperação

- UNDO da alteração em  $P$  protocolada em LSN(90)
  - PageLSN( $P$ )=90
- Não é necessário REDO
  - PageLSN( $P$ )=90

## 7. Recuperação e Logging - Procedimentos de Recuperação -

### \* Procedimento genérico (Update-in-place, Steal, NoForce, TCC)

#### 1. Varredura de Análise

- \* Ler arquivo de log a partir do último checkpoint até fim do arquivo
- \* Identificar as transações vencedoras (committed) e as perdedoras
  - \* Páginas alteradas por estas transações

## 7. Recuperação e Logging - Procedimentos de Recuperação -

### \* Procedimento genérico

#### 2. Varredura de UNDO

- \* Ler arquivo de log de trás para frente, a partir do EOF até BOT da mais antiga transação perdedora
- \* Para cada transação perdedora
  - \* Se LSN da página  $\geq$  LSN do registro
    - \* Desfazer
      - \* Gravar campo Before Image do arquivo de log sobre a página em disco
    - \* Atualizar LSN da página
      - \* PageLSN(p)  $\leftarrow$  LSN do registro

## 7. Recuperação e Logging - Procedimentos de Recuperação -

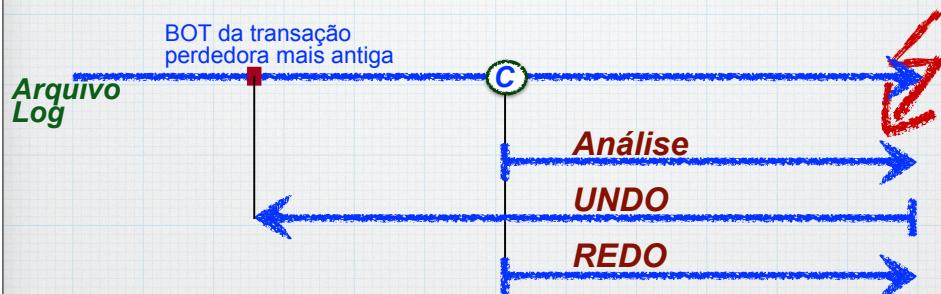
### \* Procedimento genérico

#### 3. Varredura de REDO

- \* A partir do último checkpoint até fim do arquivo de log
- \* Para cada transação vencedora
  - \* Se LSN da página < LSN do registro
    - \* Refazer
      - \* Gravar campo After Image do arquivo de log sobre a página em disco
    - \* Atualizar LSN da página
      - \* PageLSN(p)  $\leftarrow$  LSN do registro

## 7. Recuperação e Logging - Procedimentos de Recuperação -

- \* Procedimento genérico (Update-in-place, Steal, NoForce, TCC)



### Procedimento Idempotente:

*Mesmo em situações de falha durante sua execução, produz o mesmo resultado sobre o Banco de Dados*

## 7. Recuperação e Logging - Procedimentos de Recuperação -

- \* Fase de Undo deve ocorrer antes de Redo
  - \* Exemplo de cenário Redo antes do Undo
    - \*  $T_1$  e  $T_2$  que atualizam objetos de P



Procedimentos de recuperação:

### REDO

After Image de P; PageLSN(P)=90

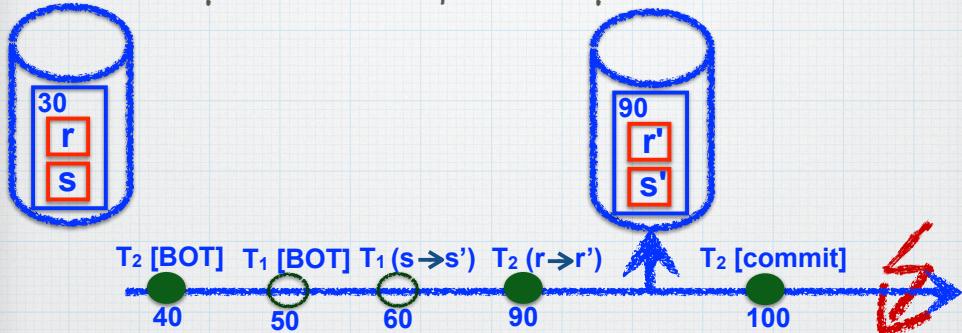
### UNDO

Before image de P; PageLSN(P)=60

*Desfaz a alteração de  $T_2 \Rightarrow$  Estado inconsistente*

## 7. Recuperação e Logging - Procedimentos de Recuperação -

- \* Granulosidade log  $\leq$  Gran. bloqueio
- \* log=página; bloqueio=tupla



### Procedimentos de recuperação:

**UNDO:** Before Image de P (r,s); PageLSN(P)=60

**REDO:** After image de P (r',s'); PageLSN(P)=90

**s' deve ser desfeito  $\Rightarrow$  Estado inconsistente**

## 7. Recuperação e Logging - Procedimentos de Recuperação -

### \* Atividade Lab

- \* Consulte o Log de Estoque
- \* Dispare as seguinte transação sobre o BD Estoque
 

```
begin transaction T1
declare @valor int
select @valor=qtdde from Produtos
where id_produto=1
set @valor=@valor+20
update Produtos set qtdde=@valor where id_produto=1
commit
```
- \* Consulte o Log de Estoque
- \* Descreva as ações que foram executadas pelo SGBD
  - \* Verifique o estado do banco de dados gerado
- \* Consulte o log de Estoque



UNIVERSIDADE  
FEDERAL do CEARÁ

## 8. Indexing

- Basic Concepts -

- \* Each index entry has
  - \* A search key
    - \* One or more attributes
  - \* Pointers to pages which contain tuples with the value of the search key
- \* Reduces the amount of disk access for finding a subset of tuples
- \* Classification
  - \* Ordered
  - \* Non-ordered



UNIVERSIDADE  
FEDERAL do CEARÁ

## 5. Indexing

- Classification -

- \* **Ordered index**
  - \* Search keys are ordered in the index structure
- \* **Primary (Clustered)**
  - \* The table is physically sorted by the search key
- \* **Dense**
  - \* One entry for each tuple
- \* **Sparse**
  - \* One entry for each data page
- \* **Secondary (Non-clustered)**
  - \* The table is not physically sorted by the search key
- \* **Dense**



UNIVERSIDADE  
FEDERAL do CEARÁ

## 5. Indexing

- Classification -

### \* Non-ordered index

- \* The search keys are not ordered in the index structure

### \* Hash index

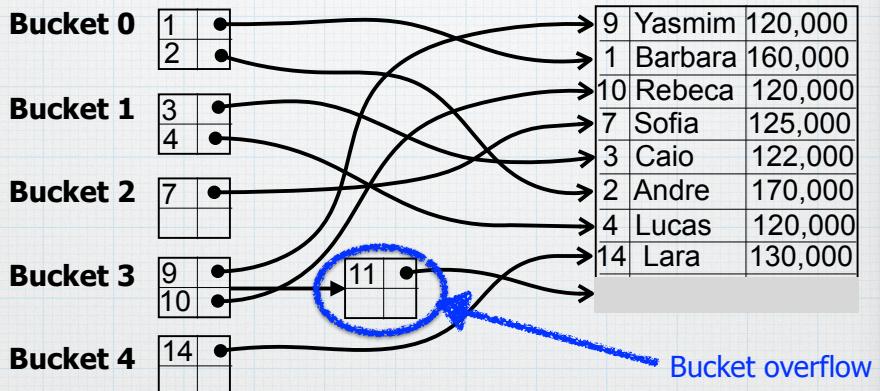


## 5. Indexing

- Hash Index -

### \* Hash Index

$$h = \lfloor ID/3 \rfloor$$



How many disk accesses are necessary to process Q ?  
Q: Select \* from Employee where ID=7



UNIVERSIDADE  
FEDERAL do CEARÁ

## 5. Indexing

### - B<sup>+</sup>-Tree -

- \* B<sup>+</sup>-Tree properties

- \* Search keys are ordered

- Balanced tree

- \* All paths from the root to the leaves have the same amount of nodes

- \* Same length

- n-ary tree

- \* There are at most n children per node

- \* n represents the order of tree (fanout)

- \* Branching factor



UNIVERSIDADE  
FEDERAL do CEARÁ

## 5. Indexing

### - B<sup>+</sup>-Tree -

- \* Rules for a B<sup>+</sup>-Tree of order n (balancing)

- \* The root has at least two pointers

- \* Internal nodes have at most n pointers, which point recursively to a B<sup>+</sup>-Tree

- \* Internal nodes have at most n-1 search keys

- Structure of an internal node

- \*  $\langle PT_1, K_1, PT_2, K_2, \dots, PT_{m-1}, K_{m-1}, PT_m \rangle$ , where  $m \leq n$

- \* Internal nodes have at least  $\lceil n/2 \rceil$  pointers

- \*  $\lfloor (n-1)/2 \rfloor$  search keys



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 5. Indexing

- B<sup>+</sup>-Tree -

- \* Rules for a B<sup>+</sup>-Tree of order n (balancing)
- \* Leaf nodes
  - \* At most  $n-1$  and at least  $\lceil((n-1)/2)\rceil$  pointers
    - \* To data pages
  - \* The last pointer points to the next leaf node



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 5. Indexing

- B<sup>+</sup>-Tree -

- \* Rules for a B<sup>+</sup>-Tree of order n (order)
  - \* At each node
    - \*  $K_1 < K_2 < K_3 < \dots < K_{m-1}$ , where  $m \leq n$
    - \* For every search-key value X belonging to subtree pointed by PT<sub>i</sub>:
      - \* For  $i=1$ ,  $X \leq K_1$
      - \* For  $1 < i < m$ ,  $K_{i-1} < X \leq K_i$
      - \* For  $i=m$ ,  $X > K_{m-1}$



UNIVERSIDADE

FEDERAL DO CEARÁ

## 5. Indexing

### - Inserting an Entry into a B+-Tree -

#### B+-Tree ()

```
Allocate a new node R; /* the root */  
While there are entries  
    do  
        insert_new_entry (L, K); /* L is the node address where to insert k */  
    End-While;  
  
insert_new_entry (L, K)  
If L has n-1 search-keys /* overflow */  
    Allocate new node L' sibling of L; /* node split */  
    If Parent(L) ≠ Ø /* L has a parent */  
        /* the median search-key value M should be inserted into Parent(L) */  
        insert_new_entry (Parent(L), M)  
    Otherwise  
        Allocate a new node R, where R is the parent of L and L'  
        /* the median search-key value M should be inserted into R */  
        insert_new_entry (R, M)  
        /* In R, the pointer PT1 points to L and PT2 points to L' */  
    End-If  
    search-key values X ≤ M remain in node L;  
    search-key values Y > M should be inserted into L';  
Otherwise  
    allocate K in L
```



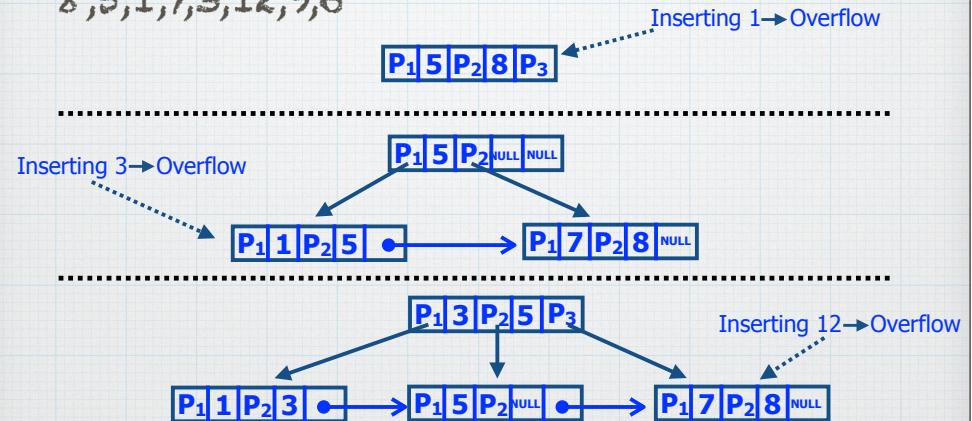
UNIVERSIDADE  
FEDERAL DO CEARÁ

## 5. Indexing

### - B+-Tree -

- \* Build a B+-Tree of order 3 with the following insertion sequence:

8, 5, 1, 7, 3, 12, 9, 6

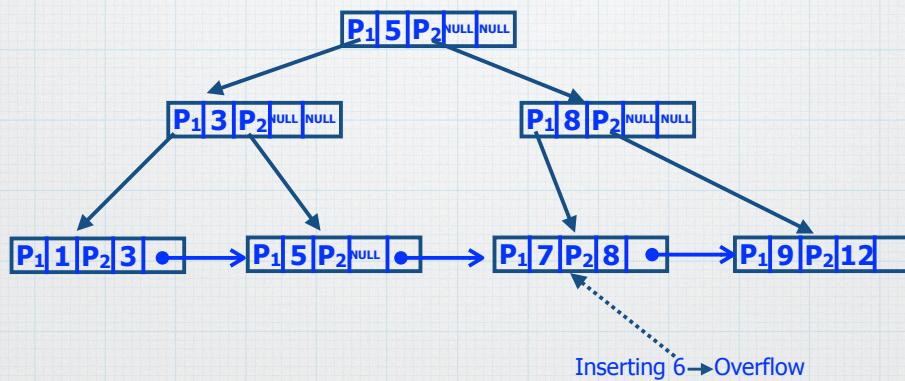




## 5. Indexing

- B<sup>+</sup>-Tree -

- \* Build a B<sup>+</sup>-Tree of order 3 with the following insertion sequence:  
8,5,1,7,3,12,9,6



## 5. Indexing

- B<sup>+</sup>-Tree -

- \* The height should be computed for the worst-case
    - \* Each node (including the root) contains  $\lceil n/2 \rceil$  pointers
  - \* Relation between a level L and the number (M) of nodes in L
    - \*  $L=0; M=1$
    - \*  $L=1; M=n/2$
    - \*  $L=2; M=(n/2)^2$
    - \*  $L=3; M=(n/2)^3$
    - \* ...
    - \*  $L=h; M=(n/2)^h$
    - \*  $\log_{n/2} M = \log_{n/2} (n/2)^h$
- ⇒  $h = \log_{n/2} M$  →



## 5. Indexing

- B<sup>+</sup>-Tree -

- \* Compute the height of a B<sup>+</sup>-Tree for the worst-case, where
  - \* The root has two pointers
  - \* Each node (except the root) contains  $\lceil n/2 \rceil$  pointers



## 5. Indexing

- B<sup>+</sup>-Tree -

- \* Computing the order
  - \* Each node corresponds to a database page
  - \* Let T bytes be the size of a page, PT bytes be the size of a B<sup>+</sup>-Tree pointer and S bytes the size of a search key
  - \* Order of a B<sup>+</sup>-Tree
- \*  $T = n * PT + (n-1) * S$

## 5. Indexing

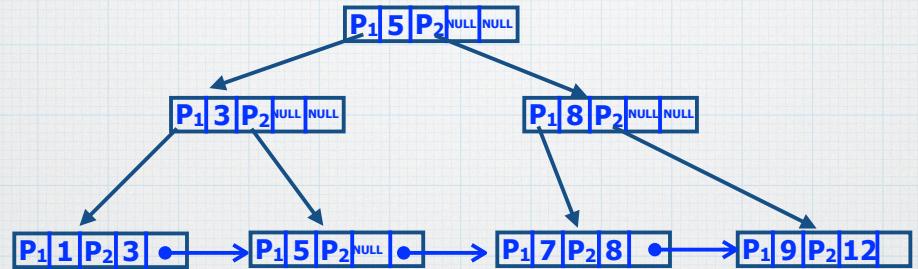
- Read/Write complexity in B<sup>+</sup>-Tree -

- \* Number of read pages to find a tuple with key K<sub>2</sub>
  - \*  $1 + \log_2 M$
- \* Number of written pages to insert a new entry K'
  - \*  $1 + \log_2 M$ 
    - \* In worst-case, all nodes have n pointers
      - \* n corresponds to the B<sup>+</sup>-Tree order

## 5. Indexing

- B<sup>+</sup>-Tree -

- \* Deleting a node entry





UNIVERSIDADE  
FEDERAL DO CEARÁ

## 5. Indexing

- Removing an entry in a B+-Tree -

### Função remoção\_entrada(L, chave\_busca)

Se L é nó raiz

    remoção\_entrada\_raiz(L, chave\_busca)

Senão

    remoção\_entrada\_nó(L, chave\_busca)

Fim;



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 5. Indexing

- Removing an entry in a B+-Tree -

### Função remoção\_entrada\_raiz(L, chave\_busca)

Se L possui apenas uma entrada <chave,PD> /\* L possui 2 filhos \*/

    S=filho(L) /\* retorna filho de L que possui no mínimo (n/2) entradas \*/

    Se S ≠ Ø

        Se S=filho\_esquerda(L)

            /\* Inserir na raiz maior valor de chave de busca em S toda subárvore \*/

            remoção\_entrada(S, K)

            inserção\_nova\_entrada(L, K)

        Senão

            /\* Inserir na raiz menor valor de chave de busca em S \*/

            remoção\_entrada(S, K)

            inserção\_nova\_entrada(L, K)

        Senão /\* fundir os nós filhos de L em um nó R que será a nova raiz da árvore \*/

    Senão

        remover entrada <Ki> de L

        Se Pi ≠ NULL ou Pi+1 ≠ NULL

            atualizar\_chave\_busca(L,i)



## 5. Indexing

- Removing an entry in a B+-Tree -

### Função remoção\_entrada\_nó(L, k)

remover entrada <K> de L;

Se nó L possui ((n- 1)/2) entradas <k> /\* underflow \*/

L'= irmão (L); /\* retorna irmão de L que possui (n/2) entradas <k> \*/

Se L' ≠ Ø

inserção\_nova\_entrada(L,k); /\*inserir V (valor entre L e L' no pai(L)) em L \*/

Se L'=irmão\_esq(L) /\* Substituir V no nó pai por maior valor de chave em L' \*/

remoção\_entrada(L', K); inserção\_nova\_entrada(Pai(L), K)

Senão /\* Substituir V' no nó pai por menor valor de chave de busca em L' \*/

remoção\_entrada(L', F); inserção\_nova\_entrada(Pai(L), F);

Senão /\* Irmãos de L contêm exatamente n/2 - 1 entradas <k> \*/

Se L'=irmão\_esq(L)

inserção\_nova\_entrada(L', V, PD); /\*inserir V em L' \*/

remoção\_entrada(Pai(L), V, PD); /\*remove V de Pai(L) \*/

inserção\_nova\_entrada(Pai(L), K); /\*Substituir V em Pai(L) por maior valor K em L \*/

Transferir entradas de L para L'; remover nó L;

Senão

inserção\_nova\_entrada(L, V, PD) /\*inserir V em L' \*/

remoção\_entrada(Pai(L), V, PD); /\*remove V de Pai(L) \*/

inserção\_nova\_entrada(Pai(L), K) /\* Substituir V no nó pai por maior chave em L' \*/

Transferir entradas de L' para L; remover nó L';

Senão

Se PTi ≠ NULL ou PTi+1 ≠ NULL

atualiza\_chave\_busca(L,PTi);



## 5. Indexing

- Removing an entry in a B+-Tree -

### Função atualiza\_chave\_busca(L, PTi)

S=filho(L, PTi) /\*retorna verdade se filho de L apontado por PTi tem (n/2) entradas <chave,PD> \*/

S'= filho(L, PTi+1)

Se (S) /\* Inserir na raiz maior valor de chave de busca da subárvore apontada PTi\*/

remoção\_entrada(L', K)

inserção\_nova\_entrada(L, K, PD)

Senão

Se (S') /\* Inserir na raiz menor valor de chave da subárvore apontada PTi+1 \*/

remoção\_entrada(L', K)

inserção\_nova\_entrada(L, K, PD)

Senão

/\* fundir os nós filhos de L em um nó S que será a nova raiz da árvore \*/

## 5. Indexing

- B+-Tree -

- \* SQL Sever

- \* Arquitetura

- \* <https://learn.microsoft.com/en-us/sql/relational-databases/sql-server-index-design-guide?view=sql-server-ver16>

- \* Definição

```
CREATE [UNIQUE] [CLUSTERED] |  
NONCLUSTERED] INDEX nome Índice  
ON nome_tab (nome_col [ASC | DESC [,  
{nome_col}])  
INCLUDE (nome_col [, {nome_col}])  
[WITH ([PAD_INDEX= ON | OFF],  
FILLFACTOR = taxa_preenchimento_nós) ]  
[ON filegroup]
```

## 5. Indexing

- B+-Tree -

- \* Cláusula INCLUDE

- \* Permite o armazenamento de um ou mais atributos na estrutura de índice
  - \* Tais atributos não compõem a chave de busca do índice
  - \* Pode ser eficiente em consultas com projeções dos atributos especificados no include
    - \* Evita acesso adicional à tabela
  - \* Cuidado
    - \* Árvore pode ficar mais alta
    - \* Porque ?

## 5. Indexing

### - B+-Tree -

- \* Taxa de preenchimento de nós
  - \* Fillfactor
    - \* Taxa de preenchimento de nós da árvore B+-Tree
      - \* Aplica-se na construção inicial da estrutura
    - \* [1,100] em percentual
  - \* Pad\_Index
    - \* ON
      - \* Fillfactor aplica-se aos nós internos e nós folhas
    - \* OFF
      - \* Fillfactor aplica-se apenas aos nós folhas
  - \* Default
    - \* 0
      - \* pad\_index=off, fillfactor=100

## 5. Indexing

### - B+-Tree -

- \* PostGres
  - \* Definição
    - CREATE [UNIQUE] INDEX [CONCURRENTLY]  
nome Índice ON nome\_tab METHOD tipo Índice  
(nome\_col [ASC | DESC [, {nome\_col}])  
[WITH ([FILLCODE = taxa\_preenchimento\_nós])  
[TABLESPACE nome\_tablespace]]
  - \* Tipo de índice
    - \* B+-Tree, hash, gist, spgist, gin



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 5. Indexing

- B+-Tree -

### \* Atividade Lab I

- \* Execute a seguinte consulta no TPC-H
  - \* (Q1) Select \* from Lineitem where L\_orderkey=5572166
  - \* Marque o tempo de execução
- \* Crie um índice secundário para Lineitem em L\_orderkey, com taxa de preenchimento de 100%, valendo para todos nós
  - \* Execute novamente a consulta e marque o tempo



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 5. Indexing

- B+-Tree -

### \* Atividade Lab II

- \* Remova todos os índices de Lineitem
- \* Crie um índice secundário para Lineitem em L\_orderkey, com taxa de preenchimento de 1%, valendo para todos nós
  - \* Marque o tempo de execução e compare com o da atividade I
- \* Encerre a sessão com a consulta Q1
- \* Inicie uma nova sessão e execute Q1 no TPC-H
  - \* (Q1) Select \* from Lineitem where L\_orderkey=5572166
  - \* Observe o tempo de execução apresentado na Janela de Propriedades



UNIVERSIDADE  
FEDERAL DO CEARÁ

## S. Indexing

- B<sup>+</sup>-Tree -

### \* Atividade Lab III

- \* Especifique a seguinte consulta (índice secundário, com chave de busca L\_orderkey)  
(Q2) Select L\_orderkey, L\_quantity from Lineitem  
where L\_orderkey=5572166
- \* Visualize o plano de execução
- \* Remova o índice (drop index)
- \* Crie um índice secundário para Lineitem em L\_orderkey, com taxa de preenchimento de 100%, valendo para todos nós. Utilize a opção INCLUDE com a coluna L\_quantity
- \* Visualize novamente o plano de execução de Q2
- \* Explique a diferença entre os planos de execução.



UNIVERSIDADE  
FEDERAL DO CEARÁ

## S. Indexing

- B<sup>+</sup>-Tree -

### \* Atividade Lab IV

- \* Especifique a seguinte consulta  
(Q3) Select L\_orderkey, L\_quantity from Lineitem  
where L\_quantity>50
- \* Visualize e explique o plano de execução



## 5. Indexing

- B<sup>+</sup>-Tree -

### \* Atividade Lab V

- \* Remova todos os índices de Lineitem
- \* Crie um índice secundário para Lineitem em L\_suppkey, com taxa de preenchimento de 100%, valendo só para nós folhas
- \* Escreva Q4 e Q5. Verifique o plano de execução de cada uma

(Q4) select \* from Lineitem  
where L\_suppkey>50 and L\_suppkey<100  
order by L\_suppkey

(Q5) select \* from Lineitem  
where L\_suppkey>50 and L\_suppkey<1000  
order by L\_suppkey

- \* Explique porque há diferenças entre os planos de execução



## 5. Indexing

- B<sup>+</sup>-Tree -

### \* Atividade Lab VI

- \* Remova todos os índices de Lineitem
- \* Verifique o plano de execução de Q6  
(Q6) select \* from Lineitem
- \* Crie um índice primário para Lineitem em L\_suppkey, com taxa de preenchimento máxima, valendo para todos nós.
- \* Verifique o plano de execução de Q6
- \* Explique a diferença entre os planos de execução



## 5. Indexing

### - B+-Tree -

#### \* Atividade Lab VII

- \* Remova todos os índices de Lineitem.
- \* Crie um índice secundário para Lineitem em L\_orderkey e L\_suppkey, com taxa de preenchimento máxima, valendo para todos nós.
- \* Execute a seguinte consulta
  - \* (Q7) Select \* from Lineitem where L\_orderkey=5700611 and L\_suppkey=7641
  - \* (Q8) Select \* from Lineitem where L\_orderkey=5700611
  - \* (Q9) Select \* from Lineitem where L\_suppkey=7641
- \* Explique a diferença entre os planos de execução



## 5. Indexação

### - Estruturas de Índices Ordenados Multi-chaves -

#### \* Quad-Trees

- \* Divisão de um espaço k-dimensional em células irregulares
- \* Árvore k-dimensional
  - \* Folhas apontam para um página
  - \* Várias folhas podem apontar para um mesmo bucket
  - \* Um nó em uma quad-tree k-dimensional
    - \* Representação de um ponto
    - \* Possui até  $2^k$  filhos, onde cada filho
      - \* Raiz de uma subárvore correspondente a  $2^k$  quadrantes

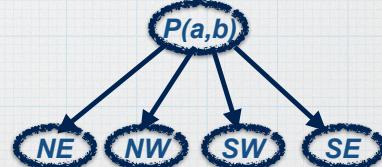
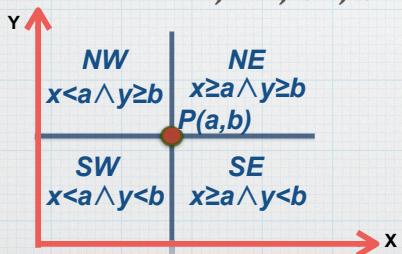


## 5. Indexação

- Estruturas de Índices Ordenados Multi-chaves -

### \* Quad-Trees

- \* Um nó em uma quad-tree bidimensional
  - \* Possui até  $2^2$  filhos, onde cada filho
    - \* Raiz de uma subárvore correspondente a  $2^2$  quadrantes
    - \* Cada quadrante representa uma direção
      - \* NW, NE, SW, SE



## 5. Indexação

- Estruturas de Índices Ordenados Multi-chaves -

### \* Quad-Trees

Salário

$P_2(35,80)$

$P_1(55,45)$

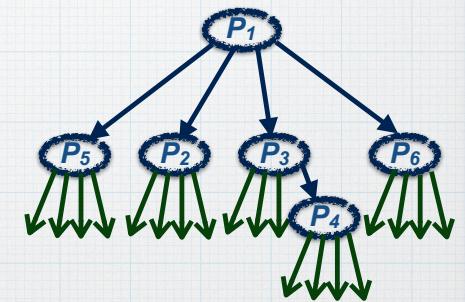
$P_3(10,35)$

$P_4(20,15)$

$P_5(70,60)$

$P_6(60,25)$

Lotação



## 6 Processamento de Consultas - Conceitos Básicos -

### \* Consulta sobre BDs

- \* Expressão de linguagem de consulta que descreve dados a serem acessados

### \* Linguagem de Consulta

- \* Linguagem de alto nível

### \* Além de consultar:

- \* Definir a estrutura de dados (DDL componente)
- \* Modificar dados (DML)
- \* especificar restrições de integridade (e.g., not null)
- \* especificar restrições de segurança

## 5. Processamento de Consultas - Conceitos Básicos -

### \* Consultas em linguagem de alto nível

- \* Mapeadas para outras formas de representação
- \* Suporte para garantir transformações de consultas
  - \* Otimização
- \* Expressar uma grande classe de consultas

### \* Formas de representação para BDs relacionais

- \* Cálculo Relacional
- \* Álgebra Relacional
- \* Grafo de Consulta
- \* Tableaus



## 6. Processamento de Consultas - Conceitos Básicos -

### \* Álgebra Relacional

- \* Coleção de operadores sobre relações
- \* Linguagem procedural
  - \* Descreve operações a serem executadas

### \* Operações básicas

- |                      |              |                    |
|----------------------|--------------|--------------------|
| * Seleção            | ( $\sigma$ ) | Operações unárias  |
| * Projeção           | ( $\Pi$ )    |                    |
| * União              | ( $\cup$ )   | Operações binárias |
| * Diferença          | ( $-$ )      |                    |
| * Produto cartesiano | ( $\times$ ) |                    |



## 6. Processamento de Consultas - Conceitos Básicos -

### \* Operações derivadas

#### \* Junção natural (natural join)

- \*  $r \bowtie_{\theta} s = \Pi_{R \cup S} (\sigma_{r.A_1 \theta s.A_1 \wedge r.A_2 \theta s.A_2 \wedge \dots \wedge r.A_n = s.A_n} (r \times s))$
- \* where  $r \bowtie s = \{A_1, A_2, \dots, A_n\}$  and  $\theta \in \{=, >, <, \leq, \geq, \neq\}$

#### \* Divisão

- \*  $r / s = \Pi_{R-S} (r) - \Pi_{R-S} ((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$

#### \* Interseção

- \*  $r \cap s = r - (r - s)$



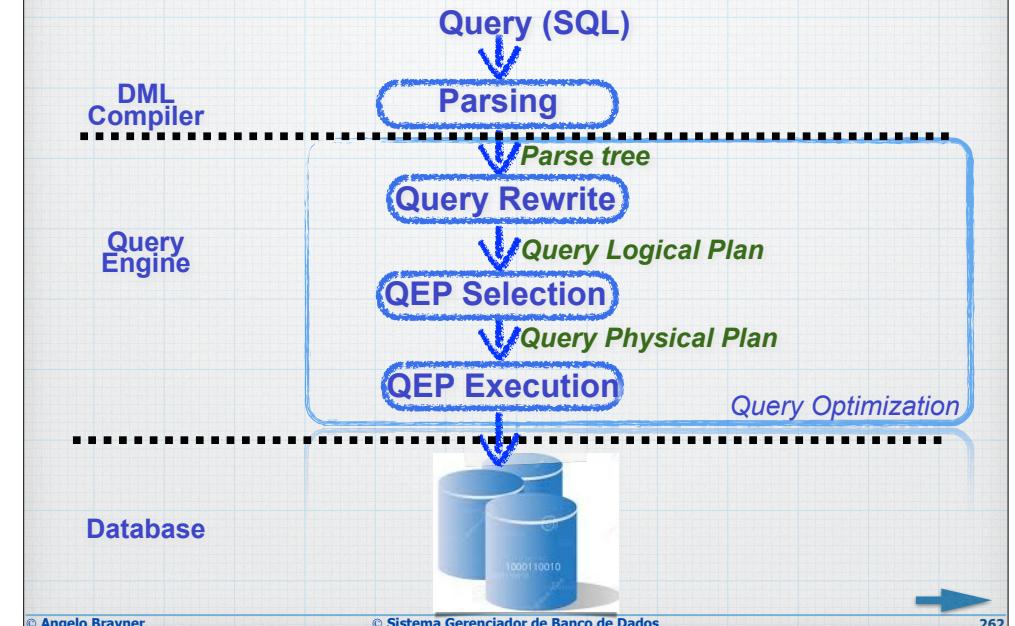
UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Processamento de Consultas - Conceitos Básicos -

- \* Outer joins
  - \* Left outer join 
  - \* Right outer join 
  - \* Full outer join 



## 6. Query Processing - Phases -



## 6. Query Processing - Query Optimization -

### \* Query Execution Plan (QEP)

- \* Logical QEP
  - \* Algebraic operators
    - \* Relational Algebra
- \* Physical QEP
  - \* Physical operators
    - \* Algorithms for implementing algebraic operators
      - \* For each logical operator
      - \* Several physical operators

## 6. Query Processing - Query Optimization -

### \* Query Rewrite

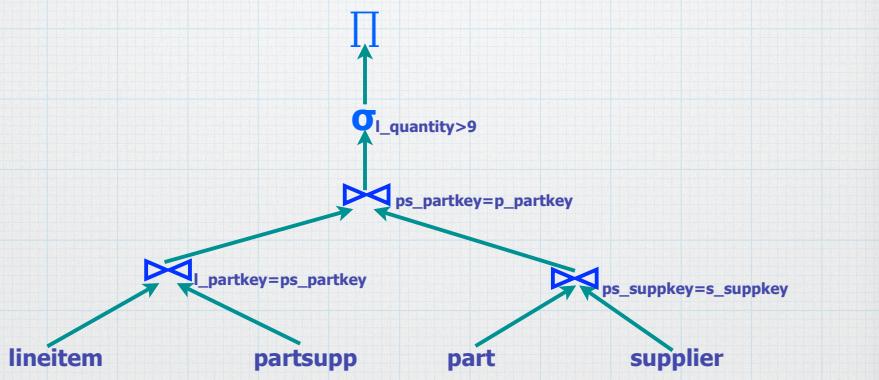
- \* Several relational algebra expressions for a given SQL query
- \* Logical QEP
  - \* Operator graph
    - \* Directed Graph  $OG(Q) = (V, E)$ 
      - \*  $v \in OG(Q)$  iff  $v$  is a relational algebra's operator in  $Q$  or  $v$  is a table in  $Q$
      - \* An edge  $e \in E$  represents the data flow in  $Q$



## 6. Query Processing - Query Optimisation -

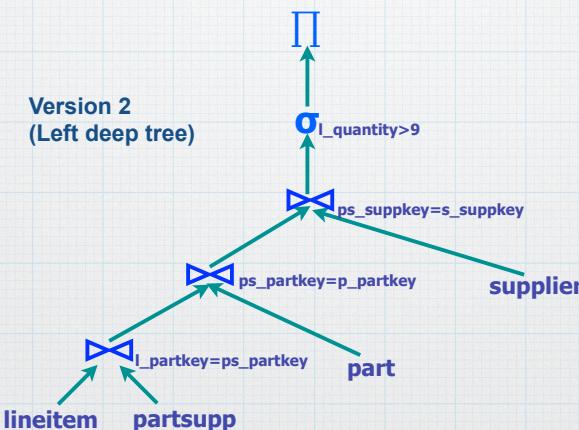
```
select * from lineitem, partsupp, part, supplier  
where l_partkey=ps_partkey and  
p_partkey=ps_partkey and  
ps_suppkey=s_suppkey and l_quantity>9
```

Join condition  
Selection condition



## 6. Query Processing - Query Optimization -

```
select * from lineitem, partsupp, part, supplier  
where l_partkey=ps_partkey and  
p_partkey=ps_partkey and  
ps_suppkey=s_suppkey and l_quantity>9
```





UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Query Processing - Query Optimization -

### \* Query Rewrite

#### \* Transformation rules

- \*  $\sigma_{\theta_1 \wedge \theta_2}(r) \Leftrightarrow \sigma_{\theta_1}(\sigma_{\theta_2}(r)) \Leftrightarrow \sigma_{\theta_2}(\sigma_{\theta_1}(r))$
- \*  $\Pi_{A_1}(\Pi_{A_2}(\Pi_{A_3} \dots (\Pi_{A_n}(r))) \dots) \Leftrightarrow \Pi_{A_1}(r)$
- \*  $\sigma_{\theta_1 \wedge \theta_2}(r \bowtie_{\theta_3} s) \Leftrightarrow \sigma_{\theta_1}(r) \bowtie_{\theta_3} \sigma_{\theta_2}(s)$
- \*  $\Pi_{A_1 \cup A_2}(r \bowtie_{\theta} s) \Leftrightarrow \Pi_{A_1 \cup A_2}((\Pi_{A_1 \cup A_3}(r) \bowtie_{\theta} (\Pi_{A_2 \cup A_4}(s)))$ 
  - \*  $A_1$  attributes of  $r$
  - \*  $A_2$  attributes of  $s$
  - \*  $A_3$  attributes of  $r$  in  $\theta$  (join condition)
  - \*  $A_4$  attributes of  $s$  in  $\theta$  (join condition)



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Query Processing - Query Optimization -

### \* Query Rewrite

#### \* Heuristics

- \* Push down selective operations
  - \* Selection and projection
  - \* Tuple and column filters
- \* Build Logical QEP as a left-deep tree
- \* Output: Efficient Logical Plan



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Query Processing - Query Optimization -

### \* Atividade

\* Construa o plano lógico mais eficiente,  
para a seguinte consulta:

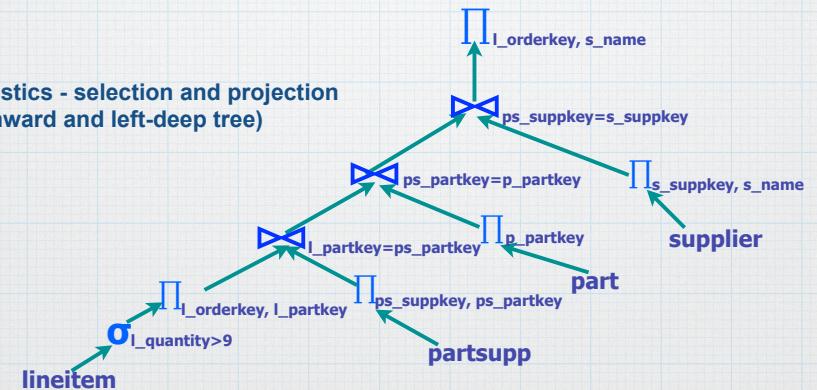
```
Select l_orderkey, s_name  
from lineitem, partsupp, part, supplier  
where l_partkey=ps_partkey and  
p_partkey=ps_partkey and  
ps_suppkey=s_suppkey and l_quantity>9
```



## 6. Query Processing - Query Optimization -

```
select l_orderkey, s_name  
from lineitem, partsupp, part, supplier  
where l_partkey=ps_partkey and  
p_partkey=ps_partkey and  
ps_suppkey=s_suppkey and l_quantity>9
```

(heuristics - selection and projection  
downward and left-deep tree)



## 6. Query Processing - Query Optimization -

### \* Atividade Lab I

#### \* Especifique a consulta

(Q12) Select L\_partkey,ps\_partkey,p\_name,s\_name  
from Lineitem, partsupp, part, supplier  
where L\_partkey=ps\_partkey and  
p\_partkey=ps\_partkey and  
ps\_suppkey=s\_suppkey and L\_quantity>9

#### \* Verifique no plano de execução os fatores para os vértices e arestas

#### \* Identifique e justifique a alteração com relação ao plano da Q11

## 6. Query Processing - Query Optimization -

### \* Query Execution Plan Selection

#### \* Build and selection of QEP

##### \* Physical plan

#### \* Search space

##### \* Set of all possible physical plans

#### \* Criterion for selecting a physical plan

##### \* The lowest query execution cost

##### \* Execution cost

##### \* Well approximated by the number of disk I/O operations

##### \* Adopted metric to estimate query execution cost



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Query Processing - Query Optimization -

- \* Parameters for estimating query execution cost
  - \*  $n_R$ 
    - \* cardinality of R
  - \*  $P_R$ 
    - \* Size of R in pages
      - \* Amount of pages containing tuples of R
  - \*  $V(a,R)$ 
    - \* Cardinality of  $\Pi_a(R)$
    - \* If a is primary key in R
      - \*  $V(a,R)=???$
  - \*  $f_R$ 
    - \* Page factor
    - \* Number of tuples of R in a page



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Query Processing - Query Optimization -

- \* Parameters for estimating query execution cost
  - \*  $SF(a,R)$ 
    - \* Selectivity factor of an attribute a in R
    - \* Number of tuples on average which satisfies a comparison condition on attribute a
  - \* Metric
    - \* Restrictiveness of the selection condition
      - \* Filter efficiency
    - \* High selectivity
      - \* Low number of tuples in the result
    - \* Low selectivity
      - \* High number of tuples in the result



## 6. Query Processing - Query Optimization -

- \* Parameters for estimating query execution cost
  - \*  $SF(a,R)$
  - \* Rules to compute  $SF(a,R)$ 
    - \* Condition:  $a \leq K$ 
      - \*  $SF = 1/V(a,R)$
    - \* Condition:  $a > K$  ( $a < K$ )
      - \*  $SF = (K - \text{Min}(a,R)) / (\text{Max}(a,R) - \text{Min}(a,R)) * \text{Size}(R)$
    - \* Condition:  $a = b$ 
      - \*  $SF = 1/\max\{V(a,R), V(b,S)\}$
  - \* Histogram
    - \* Measures the frequency of occurrence for each distinct value of a given attribute in a table



## 6. Query Processing - Query Optimization -

- \* Histograms SQL Server
  - \* Gerados
    - \* Para atributos que aparecem em consultas, como
      - \* Condição de operações de seleção
      - \* Condição de operações de junção
      - \* Argumentos em cláusulas group by
    - \* Para atributos, utilizados como chave de busca em estruturas de índices



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 5. Query Processing - Query Optimization -

### \* Histograms SQL Server

#### \* Steps

- \* Quantidade de intervalos de valores para o atributo
  - \* Máximo de 200

#### \* Criados pelo SQL Server

#### \* Rows Sampled (RS)

- \* Número de tuplas utilizadas nas estatísticas
- \* Se RS < Cardinalidade
  - \* Resultados apresentados no histograma e densidade são calculados com base no RS

#### \* Range Hi-Key

- \* Limite superior para um intervalo



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Query Processing - Query Optimization -

### \* Histograms SQL Server

#### \* Density

- \* Definido por  $1/DV$ 
  - \* DV representa o número de valores distintos do atributo, excluindo-se os valores de Range-Hi-Key

#### \* Indica a taxa de valores duplicados

- \* Quanto menor DV, maior a densidade de valores repetidos
  - \* Seletividade baixa

#### \* Utilizado até o SQL Server 2008

- \* No caso de um conjunto de atributos um único valor

#### \* Range Rows

- \* Número de tuplas no intervalo, excluindo tuplas com valores do Range-Hi-Key



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Query Processing - Query Optimization -

### \* Histograms SQL Server

#### \* Distinct Range Rows

- \* Número de tuplas com valores distintos para o atributo no intervalo, excluindo-se tuplas com valores do Range-H-Key

#### \* AVG Range Rows

- \* Número de tuplas com valores repetidos para o atributo no intervalo, excluindo tuplas com valores do Range-H-Key

$$\text{AVG} = \text{Range\_Rows} / \text{Distinct\_Range\_Rows}$$

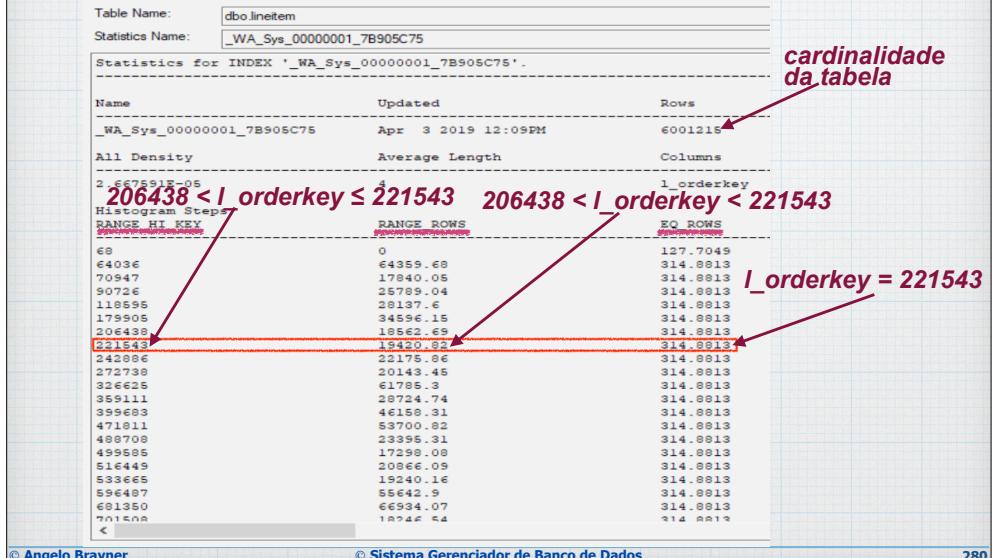
- \* Se  $\text{distinct\_range\_rows} = 0$

$$* \text{AVG\_range\_rows} = 1$$



## 6. Query Processing - Query Optimization -

### \* Histogram for estimating SF(a,R)





UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Query Processing - Query Optimization -

### \* Histogram for estimating SF(a,R)

Quantidade de valores distintos em (206438, 221543)		
DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS	Quantidade média de tuplas com valores repetidos em (206438, 221543) AVG=Range_Rows/Distinct_Range_Rows
0	1	156.9802
410	156.2729	
117	151.1785	
171	154.6968	
182	150.8072	
218	146.9852	
126	146.9773	
122	150.9025	
147	159.3556	
126	154.1785	
401	153.5868	
197	156.5167	
295	156.4706	
343	152.791	
153	141.4446	
122	159.8744	
131	149.7748	
128	154.7183	
360	149.4057	
448	147.04	
128		



## 6. Query Processing - Query Optimization -

### \* Vetor de densidade (SQL Server)

- \* All Density
  - \* Por atributo
  - \* Por conjunto de atributos
    - \* Definidos como chave para um determinado índice
- \* Definido por  $1/DV$ 
  - \* DV representa o número de valores distintos para cada prefixo do conjunto de atributos
  - \* Indica a taxa de valores duplicados
  - \* Quanto menor DV, maior a densidade de valores repetidos



UNIVERSIDADE

FEDERAL do CEARÁ

## 6. Query Processing - Query Optimization -

- \* Vetor de densidade (SQL Server)
  - \* Exemplo
    - \* Remover os índices de LineItem
    - \* Criar índice secundário I1 com chave de busca composta:
      - \* L\_partkey
      - \* L\_suppkey
      - \* L\_orderkey
    - \* Executar o comando
      - \* dbcc show\_statistics (lineitem, I1) with density\_vector



## 6. Query Processing - Query Optimization -

- \* Vetor de densidade (SQL Server)

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, there is a results grid displaying statistics for the 'lineitem' table's secondary index 'I1'. The grid has columns: All density, Average Length, and Columns. It contains three rows of data:

	All density	Average Length	Columns
1	5E-06	4	I_partkey
2	1.250718E-06	8	I_partkey, I_suppkey
3	1.666332E-07	12	I_partkey, I_suppkey, I_orderkey

At the bottom of the results pane, a message says 'Query executed successfully.'



## 6. Query Processing - Query Optimization -

- \* Physical Query Execution Plan

- \* Algebraic operators in logical plans are substituted by physical operators

- \* Selection

- \* Logical operator

- \* Find tuples in a table R which satisfies the selection condition ( $R.a \theta c$ )

- \*  $\sigma_{R.a \theta c}(R)$

- \* Physical operators

- \* Implementation of the algebraic operator

- \* Table Scan

- \* Index Seek

- \* Index Scan

## 6. Query Processing - Selection Physical Operators -

- \* Table Scan on table R ( $\sigma_{R.a \theta c}(R)$ )

- \* All the tuples belonging to table should be read

- \* To check if  $R.a \theta c$  holds

- \* Algorithm

- TableScan ( $\sigma_{R.a \theta c}(R)$ )**

- $Result = \emptyset$

- For each tuple  $t_R \in R$  do

- If  $t_R.a = c$

- $Result = Result \cup \{t_R\}$

- End-For

- End

- \* Estimated cost

- \*  $EC = P_R$



## 6. Query Processing - Selection Physical Operators -

- \* Implementação da seleção utilizando índice
  - \* Dois operadores físicos
    - \* Index Seek
    - \* Index Scan
  - \* Para cada operador físico
    - \* Há variações de implementação em função da configuração física do índice
      - \* Tipo de índice
        - \* Ordenado
          - \* Primário ou secundário
          - \* Hash
        - \* Chave de busca
          - \* Chave primária ou não
      - \* Estrutura da tabela (índice primário)
        - \* Árvore B+ ou arquivo heap

## 6. Query Processing - Selection Physical Operators -

- \* Index Seek
  - \* Primary Index (Clustered) i on R.a
    - \* R.a is primary key
    - \* R as B+-tree
      - \* Leaf nodes are data pages
      - \* Algorithm
  - \* IndexSeek\_PI\_PK\_BS ( $\sigma_{R.a=c}(R)$ )
    - Result =  $\emptyset$
    - K=c /\* K is the search key \*/
    - Search in B+-tree a leaf-node LN for K
    - For each tuple  $t_R \in LN$  do
      - If  $t_R.a = K$ 
        - Result = Result  $\cup \{t_R\}$
        - Break
    - End-For
    - End
  - \* Estimated cost
    - \*  $EC = HT_i$
    - \*  $HT_i$  is the height of i



## 6. Query Processing

### - Selection Physical Operators -

- \* Index Seek

- \* Primary Index (Clustered) i on R.a

- \* R.a is not primary key

- \* R as B+-tree

- \* Leaf nodes are data pages

- \* Algorithm ( $\theta_1, \theta_2 \in \{>, \sum, <, \leq\}$ )

- \* **IndexSeek\_PI\_PK\_BS** ( $\sigma_{R.a \theta_1 C \text{ and } R.a \theta_2 C'}(R)$ )

```
Result = Ø; i=1
```

```
K=a /* K is the search key */
```

```
Search B+-tree first leaf-node LN; with K θ1 C and K θ2 C'
```

```
For 0 < i ≤ X
```

```
For each tuple tR ∈ LN; do
```

```
If tR.a θ1 C and tR.a θ2 C'
```

```
Result = Result ∪ {tR}
```

```
else
```

```
Break
```

```
End-For
```

```
i=i+1
```

```
End-For
```

```
End
```

$$X = \lceil SF(a, R) / f_R \rceil$$

tuplas que satisfazem a condição de seleção estão agrupadas em nós contíguos

primeiro nó folha com tuplas que satisfazem a condição de seleção, já encontrado ao descer a altura da árvore

- \* Estimated cost

- \*  $EC = HT_i + X - 1$

- \*  $HT_i$  is the height of i

© Angelo Brayner

© Sistema Gerenciador de Banco de Dados

289

UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Query Processing

### - Selection Physical Operators -

- \* Index Seek

- \* Primary Index (Clustered) i on R.a

- \* R.a is primary key

- \* R as a heap file

- \* Algorithm

- \* **IndexSeek\_PI\_PK\_HF** ( $\sigma_{R.a \theta c}(R)$ )

```
Result = Ø
```

```
K=c /* K is the search key */
```

```
Search in B+-tree's leaf-node entry with page address PT  
to find a tuple with tr.a=K
```

```
Read Page(PT)
```

```
For each tuple tR ∈ page(PT) do
```

```
If tR.a = K
```

```
Result = Result ∪ {tR}
```

```
Break
```

```
End-For
```

```
End
```

For a search-key value K belonging to a data page pointed by PT; holds:  
For i=1,  $K \leq K_i$   
For  $1 < i < m$ ,  $K_{i-1} < K \leq K_i$

- \* Estimated cost

- \*  $EC = HT_i + 1$

© Angelo Brayner

© Sistema Gerenciador de Banco de Dados

290



## 6. Query Processing - Selection Physical Operators -

### \* Index Seek

- \* Primary Index i on R.a
- \* R.a is not primary key
- \* R as B+-tree

- \* Several leaf nodes might be read
- \* Algorithm

**IndexSeek\_PI\_NPK\_BS ( $\sigma_{R.a=c}(R)$ )**

```
Result = Ø
K=c /* K is the search key */
i=1
Search B+-tree first leaf-nodes LN; for K
For 0 <= i < X
    For each tuple  $t_R \in LN_i$  do
        If  $t_R.a = K$ 
            Result = Result  $\cup \{t_R\}$ 
        else
            Break
        End-For
    i=i+1
End-For
End
```

- \* Estimated cost
- \*  $EC = HT_i + X - 1$

tuplas que satisfazem  
condição de seleção  
estão agrupadas  
em nós contíguos

$$X = \lceil SF(a, R) / f_R \rceil$$

primeiro nó folha com tuplas que  
satisfazem a condição de seleção,  
já encontrado ao descer a altura  
da árvore

## 6. Query Processing - Selection Physical Operators -

### \* Index Seek

- \* Primary Index i on R.a
- \* R.a is not primary key
- \* R as heap file

- \* Several data pages might be read (file)
- \* Algorithm

**IndexSeek\_PI\_NPK\_HF ( $\sigma_{R.a=c}(R)$ )**

```
Result = Ø
K=c /* K is the search key */
Search in B+-tree's leaf-node entry with page address PT to find a
tuple with  $t.a = K$ 
Read Page(PT)
For 0 <= i < X
    For each tuple  $t_R \in Page$  do
        If  $t_R.a = K$ 
            Result = Result  $\cup \{t_R\}$ 
        End-For
    i=i+1
End-For
Read NextPage
End-For
End
```

- \* Estimated cost
- \*  $EC = HT_i + (\lceil SF(a, R) / f_R \rceil)$



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Query Processing - Selection Physical Operators -

### \* Index Seek

- \* Secondary index (non-clustered) i on R.a
- \* R.a is primary key
- \* Algorithm
  - \* Same as IndexSeek\_PI\_PK\_HF ( $O_{R.a\neq c}(R)$ )
  - \* A data page should be accessed
- \* Estimated cost
  - \*  $EC = HT_i + 1$



## 6. Query Processing - Selection Physical Operators -

### \* Index Seek

- \* Secondary index (non-clustered) i on R.a
- \* R.a is not primary key
- \* Algorithm
  - \* Similar to IndexSeek\_PI\_NPK\_HF ( $O_{R.a\neq c}(R)$ )
  - \* Except for the value of X
  - \* Worst case
    - \* Each tuple is located in a different page
    - \*  $X = SF(a, R)$
- \* Estimated cost
  - \*  $EC = HT_i + SF(a, R)$



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Query Processing - Selection Physical Operators -

- \* Index Seek

- \* Hash Index

- \* Hash function on R.a

- \* Estimated cost

- \*  $EC = 2$

- \* Without bucket overflow (best case)

- \*  $EC = 2 + n$

- \* n represents the list size

- \* With bucket overflow (worst case)



## 6. Query Processing - Selection Physical Operators -

- \* Index Scan

- \* All leaf nodes should be visited

- \* Let M be the number of leaf nodes and n the order of i

- \* Primary Index

- \* Estimated cost

- \* R as B<sup>+</sup>-tree

- \*  $EC = HT_i + M - 1$

- \* R as a heap file

- \* For each leaf node LF, for each pointer, the data page should be visited

- \*  $EC = HT_i + (M-1) + (n-1).M$



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Query Processing - Selection Physical Operators -

- \* Index Scan

- \* Secondary Index

- \* Estimated cost

- $$EC = HT_i + (M-1) + (n-1).M$$



## 6. Query Processing - Selection Physical Operators -

- \* Atividade Lab V

- \* Remova todos os índices de Lineitem
  - \* Crie um índice secundário para Lineitem em L\_orderkey, com taxa de preenchimento de 100%, valendo para todos nós. Utilize a opção INCLUDE com a coluna L\_quantity

- \* Especifique a seguinte consulta  
(Q15) Select \* from Lineitem  
where L\_quantity > 50

- \* Visualize e explique o plano de execução



## 6. Query Processing

### - Join Physical Operators -

- \* Join

- \* Logical operator

- \* Associate tuples of two tables through the join condition ( $R \bowtie_{R.a=S.b} S$ )

- \* Physical operators (most implemented by commercial database systems)

- \* Nested-loop Join

- \* Index Nested-loop Join

- \* Block Nested-loop Join

- \* Merge Join

- \* Hash Join

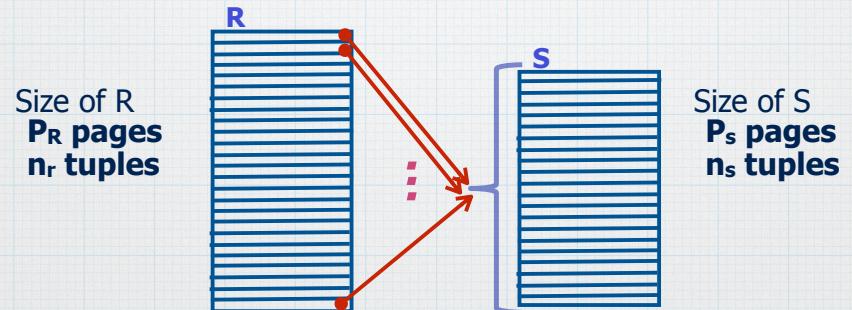
## 6. Query Processing

### - Join Physical Operators -

#### Nested-loop Join ( $R \bowtie_{R.a=S.b} S$ )

```
For each tuple  $t_r \in R$  do
  For each tuple  $t_s \in S$  do
    If  $t_r.a = t_s.b$ 
      Result = Result  $\cup \{(t_r, t_s)\}$ 
  End-For
End-For
```

$$EC = P_R + (n_r * P_S)$$





## 6. Query Processing - Join Physical Operators -

### Block Nested-loop Join ( $R \bowtie_{R.a=S.b} S$ )

```
For each Page Pr of R do
  For each Page Ps of S do
    For each tuple ts ∈ Ps do
      For each tuple tr ∈ Pr
        If tr.a = ts.b
          Result = Result ∪ {(tr, ts)}
```

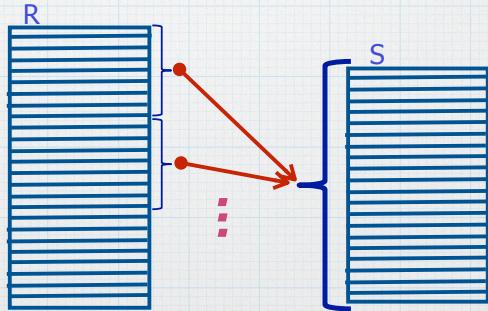
End-For

End-For

End-For

End-For

$$EC = P_R + (P_R * P_S)$$



## 6. Query Processing - Join Physical Operators -

### \* Atividade Lab VIII

\* Remova todos os índices das tabelas Part e Partsupp

\* Defina a consulta

(Q21) Select ps\_partkey, p\_partkey, p\_name  
from part inner join partsupp  
on ps\_partkey=p\_partkey  
where ps\_partkey = 20

\* Visualize e explique o plano de execução gerado



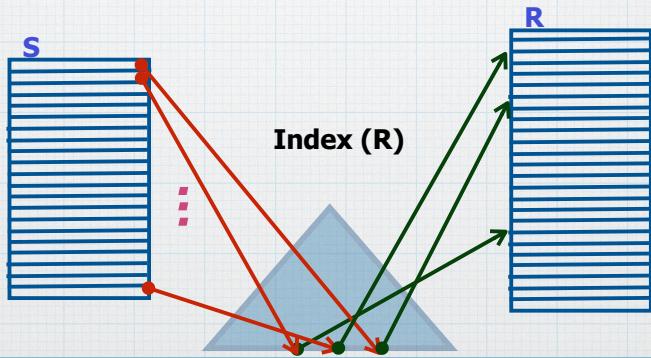
## 6. Query Processing - Join Physical Operators -

### Index Nested-loop Join ( $R \times_{R.a=S.b} S$ )

- An index is defined for R on R.a

```
For each tuple  $t_S \in S$  do
  If (Index_Seek(Ind(R),  $t_S.b$ ) == True)
    Result = Result  $\cup \{(t_R, t_S)\}$ 
End-For
```

$$EC = Ps + (n_s * C), \text{ where } C \text{ represents the cost of an Index Seek}$$



## 6. Query Processing - Join Physical Operators -

### \* Atividade Lab IX

- Crie um índice secundário em Partsupp, tendo como chave de busca ps\_partkey
- Defina a consulta  
(Q22) Select ps\_partkey, p\_partkey, p\_name  
from part inner join partsupp  
on ps\_partkey=p\_partkey  
where ps\_partkey = 20
- Visualize e explique o plano de execução gerado



## 6. Query Processing - Join Physical Operators -

### Merge Join ( $R \bowtie_{R.a=S.b} S$ )

- The tables should be sorted by R.a and S.b
- For some DBS
- There can not exist repetition on the attribute of the join condition in at least one table (external table)

```
Read R
For each tuple  $t_s \in S$  do
  While  $t_s.b \geq t_r.a$ 
    If  $t_r.a = t_s.b$ 
      Result = Result  $\cup \{(t_r, t_s)\}$ 
    End-If
    Read R
  End-While
End-For
```

R.a	S.b
8	2
9	7
	9

$$EC = P_R + P_S$$



## 6. Query Processing - Join Physical Operators -

### \* Atividade Lab X

#### \* Cenário 1

- Remova todos os índices de Lineitem e part
- Defina a consulta Q23 e guarde o plano de execução estimado

(Q23) Select \*

```
from part inner join Lineitem
on p_partkey=l_partkey
```

## 6. Query Processing - Join Physical Operators -

### \* Atividade Lab X

#### \* Cenário 2

- \* Crie um índice primário em Part, tendo como chave de busca p\_partkey, especificando p\_partkey como UNIQUE
- \* Crie um índice primário em Lineitem, tendo como chave de busca l\_partkey
- \* Observe plano de execução gerado para o Cenário 2
- \* Explique e justifique a diferença com o plano de execução do Cenário 1

## 6. Query Processing - Join Physical Operators -

### \* Atividade Lab X

#### \* Cenário 3

- \* Remova o índice de part
- \* Crie um índice secundário em Part, tendo como chave de busca p\_partkey, especificando p\_partkey como UNIQUE
- \* Visualize o novo plano de execução estimado para a consulta Q23
  - \* Justifique a diferença com o plano do cenário 2 (com índice primário em part)



## 6. Query Processing

### - Join Physical Operators -

- \* Grace Hash Join ( $R \bowtie_{R.a=S.b} S$ )

- \* 1<sup>st</sup> Phase (build phase)

- \* R and S are physically partitioned by applying a hash function  $f$  on join attributes (R.a and S.b)

For each tuple  $t_r \in R$  do

$$i = f(t_r.a)$$

$$PR_i = PR_i \cup \{t_r\}$$



For each tuple  $t_s \in S$  do

$$i = f(t_s.b)$$

$$PS_i = PS_i \cup \{t_s\}$$



$$ECB = 2(P_R + P_S)$$

## 6. Query Processing

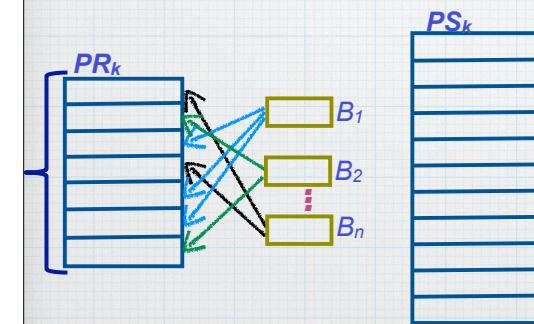
### - Join Physical Operators -

- \* Grace Hash Join ( $R \bowtie_{R.a=S.b} S$ )

- \* 2<sup>nd</sup> Phase (probe phase)

- \* Index nested-loop join on each pair of partition with same hash address

- \* Hash index



For each partition  $PR_k \in R$   
For each tuple  $t_r \in PR_k$  do  
 $i = h(t_r.a)$

$$B_i = B_i \cup \{(t_r.a, \text{Pointer})\}$$

end;

end;

For each tuple  $t_s \in PS_k$  do

$$i = h(t_s.b)$$

case  $t_s.b \in B_i$

Result = Result  $\cup \{(t_r, t_s)\}$

end;

end;

$$EC_P = P_R + P_S$$



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 6. Query Processing - Join Physical Operators -

### \* Grace Hash Join ( $R \bowtie S$ )

#### \* Total estimated cost

$$* EC = 2(P_R + P_S) + (P_R + P_S)$$

$$* EC = 3(P_R + P_S)$$



## 5. Query Processing - Join Physical Operators -

### \* In-memory Hash Join ( $R \bowtie S$ )

#### \* Build table (smallest table)

$$* P_b$$

#### \* Probe table

$$* P_p$$

#### \* In-memory hash table



## 6. Query Processing - Join Physical Operators -

### \* In-memory Hash Join ( $R \bowtie S$ )

```
for each row  $t_b$  in the build table
begin
    calculate hash value on  $t_b$  join key //  $t_b.jk_b$ 
    insert  $t_b$  into the appropriate hash bucket // In-memory bucket
end
for each row  $t_p$  in the probe table
begin
    calculate hash value on  $t_p$  join key //  $t_p.jk_p$ 
    for each row  $t_b$  in the corresponding hash bucket
        if  $t_b$  joins with  $t_p$  //  $t_b.jk_b = t_p.jk_p$ 
            Result = Result  $\cup$   $\{(t_b, t_p)\}$ 
end
```

## 6. Query Processing - Join Physical Operators -

### \* In-memory (classic) Hash Join ( $R \bowtie S$ )

#### \* Estimated Cost

- \* The build table pages may be read several times
  - \* Probe table's cardinality ( $n_p$ )
- \*  $EC = P_b + P_p + n_p$  # Read build table's pages (every tuple of probe table satisfies join condition)
- \* Worst case
- \* There is not enough main memory for holding the hash table

#### \* SQL Server, Oracle, Postgres



## 6. Query Processing - Query Optimization -

- \* Each physical operator has a different cost
  - \* Selection
    - \* Table Scan
    - \* Index Seek
    - \* Index Scan
  - \* Join
    - \* Nested-loop join:  $P_r + n_r \cdot P_s$
    - \* Index nested-loop join:  $P_r + n_r \cdot C$
    - \* Block nested-loop join:  $P_r + P_r \cdot P_s$
    - \* Merge join:  $P_r + P_s$
    - \* Hash join:  $3(P_r + P_s)$



## 5. Query Processing - Query Optimization -

- \* Query Execution Plan Selection
  - \* Join
    - \* Binary operation
      - \* Two operands (tables)  
`select * from lineitem l, partsupp ps, part p, supplier s  
where l_partkey=ps_partkey and p_partkey=ps_partkey  
and ps_suppkey=s_suppkey`
    - \* Possible execution orders
      - \* (((l $\bowtie$ ps) $\bowtie$ p) $\bowtie$ s)
      - \* (((l $\bowtie$ s) $\bowtie$ p) $\bowtie$ ps)
      - \* (((s $\bowtie$ ps) $\bowtie$ l) $\bowtie$ p)
      - \* (l $\bowtie$ ps) $\bowtie$ (p $\bowtie$ s) ...
    - \* 24 different execution orders
      - \* Search space



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 5. Query Processing - Query Optimization -

### \* Query Execution Plan Selection

#### \* Join ordering problem

- \* n-way joins executed as a sequence of 2-way joins
- \* Search space
  - \* All possible execution orders
- \* Search space size
  - \*  $n!$
- \* Complexity to find the most efficient execution order
  - \*  $O(n!)$



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 5. Query Processing - Query Optimization -

### \* Query Execution Plan Selection

#### \* Join ordering problem

- \* Search space inflating
  - \* Each algebraic operator has several physical operators
    - \* There are four different physical operators for join operator
    - \* Indexes
- \* Search space deflating
  - \* Heuristic - left deep tree
  - \* Search space
    - \*  $(2n)!/(n+1)!n!$



UNIVERSIDADE  
FEDERAL DO CEARÁ

## S. Query Processing - Query Optimization -

- \* Query Execution Plan Selection
  - \* Join ordering problem
    - \* Dynamic programming
      - \*  $O(n \cdot 2^{(n-1)})$
    - \* Heuristics
      - \* Left deep tree
      - \* Join selectivity



UNIVERSIDADE  
FEDERAL DO CEARÁ

## S. Query Processing - Join Physical Operators -

- \* Atividade Lab XIV (Join enumeration)
    - \* Visualize o plano de execução estimado para as seguintes consultas. Identifique e justifique as diferenças
- (Q28) Select L\_orderkey, s\_name  
from Lineitem, partsupp, part, supplier  
where L\_partkey=ps\_partkey and  
p\_partkey=ps\_partkey and  
ps\_suppkey=s\_suppkey and L\_quantity>9
- (Q29) Select L\_orderkey, s\_name  
from Lineitem, partsupp, part, supplier  
where L\_partkey=ps\_partkey and  
p\_partkey=ps\_partkey and  
ps\_suppkey=s\_suppkey and L\_quantity>49

## 6. Query Processing - Query Physical Plan [Left Deep Tree] -

SQL

Query1.sql - BR...r-VM\Brayner (54)\*

```
select * from lineitem, partsupp, part, supplier
where l_partkey=ps_partkey and p_partkey=ps_partkey and ps_suppkey=s_suppkey
and l_quantity>9
```

100 %

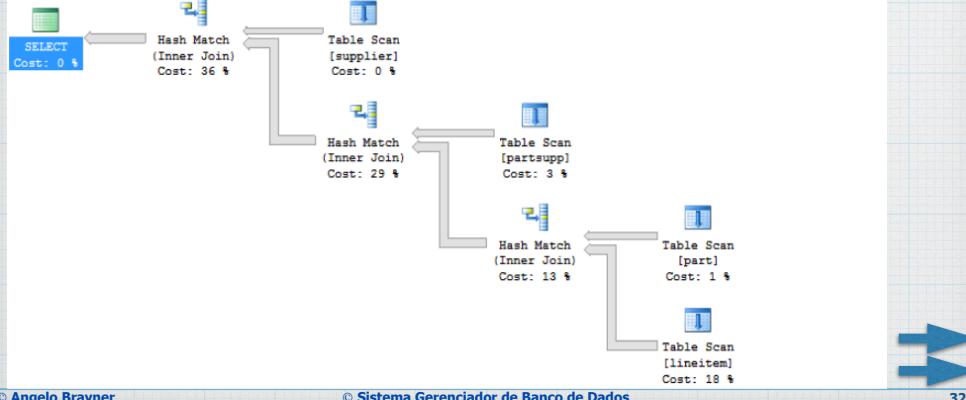
&lt; &gt;

Messages Execution plan

Query 1: Query cost (relative to the batch): 100%

```
select * from lineitem, partsupp, part, supplier where l_partkey=ps_partkey and p_partkey=ps_partkey
```

Missing Index (Impact 12.7387): CREATE NONCLUSTERED INDEX &lt;Name of Missing Index, sysname,&gt; ON



## 6. Query Processing - Query Optimization -

- \* Execution of a Query Execution Plan

- \* Temporary (internal) results

- \* Materialization

- \* Stored

- \* Secondary memory

- \* Pipelining

- \* Results are consumed by the next operator as soon as they are yield

- \* Non Stop-and-go physical operators



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 7. Técnicas para o Ajuste de Performance - Benchmark -

- \* Diferentes DBMSs

- \* Diferentes implementações

- \* Diferentes performance para diferentes cargas de trabalho

- \* Benchmark

- \* Quantificar performance de DBMSs

- \* Número de transações executadas por segundo para uma determinada carga de trabalho

- \* Vazão (throughput)



## 7. Técnicas para o Ajuste de Performance - Benchmark -

- \* Classe de Aplicações de Bancos de Dados

- \* Online transaction processing (OLTP)

- \* Alto grau de concorrência

- \* Técnicas eficientes para otimizar o processamento de commits

- \* Suporte a decisão e Online Analytical Processing (OLAP)

- \* Algoritmos eficientes para a execução de consultas

- \* Otimização de consultas

## 7. Técnicas para o Ajuste de Performance - Benchmark -

\* Benchmarks TPC (Transaction Processing Performance Council - <http://www.tpc.org>)

### \* TPC-A

- \* Quantificar performance em BDs com atualizações intensivas (Aplicações OLTP)
- \* Incorpora comunicação com terminais
- \* Simula aplicação bancária

### \* TPC-B

- \* Quantificar performance do núcleo de BDs com atualizações intensivas (Aplicações OLTP)
- \* TPC-A removendo a comunicação com terminais

## 7. Técnicas para o Ajuste de Performance - Benchmark -

### \* Benchmarks TPC

#### \* TPC-C

- \* Modela um sistema de vendas de um fornecedor, com produtos distribuídos em vários depósitos
  - \* Entrada de solicitações (entrada de solicitações e entrega de encomendas, armazenamento de faturas e pagamentos)
- \* Banco de dados
  - \* 9 tabelas
- \* Carga de trabalho
  - \* 5 transações
    - \* novo pedido, pagamento, status pedido, entrega, status do estoque



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 7. Técnicas para o Ajuste de Performance - Benchmark -

### \* Benchmarks TPC

#### \* TPC-E

- \* Simula um ambiente de transações em bolsas de valores
- \* Banco de dados
  - \* 33 tabelas
  - \* Populated with pseudo-real data
    - \* Distributions based on 2000 U.S. and Canada census data
    - \* Used for generating name, address, gender, etc.
    - \* Introduces natural data skew
  - \* Actual listings on the NYSE and NASDAQ

#### \* Carga de trabalho

- \* 6 R0 transactions
- \* 4 RW Transactions



## 7. Técnicas para o Ajuste de Performance - Benchmark -

### \* Benchmarks TPC

#### \* TPC-H

- \* Quantificar performance de BDs com consultas intensivas (Aplicações de suporte a decisão)
  - \* Data warehouse application (snowflake)

#### \* Banco de dados

- \* 8 tables
- \* Several scale factors
  - \* SF 10: ~13GB database

#### \* Carga de trabalho

- \* 22 queries

## 7. Técnicas para o Ajuste de Performance - Sintonia Fina -

- \* Ajustar performance de um SGBD
  - \* Ajustar diversos parâmetros e decisões de projeto
    - \* Melhorar a performance do sistema para uma determinada aplicação
- \* Gargalo
  - \* Recurso do sistema que está sendo utilizado em sua capacidade máxima, limitando a vazão do sistema
- \* Sintonia fina (**tuning**)
  - \* Identificação e eliminação de gargalos

## 7. Técnicas para o Ajuste de Performance - Sintonia Fina -

- \* Três níveis
  - \* Hardware
  - \* Parâmetros do DBMS
  - \* Projeto



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 7. Técnicas para o Ajuste de Performance - Sintonia Fina -

- \* Tuning a nível de Hardware
  - \* Adicionar mais disco
    - \* Quando o gargalo é o I/O de disco
    - \* RAID
  - \* Adicionar mais memória principal
    - \* Quando o gargalo é área de buffer
  - \* Utilizar CPUs mais rápidas ou dedicadas
    - \* CPU só para processar consultas (DB2)



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 7. Técnicas para o Ajuste de Performance - Sintonia Fina -

- \* Tuning a nível de parâmetros do DBMS
  - \* Aumentar o tamanho da área de buffer (SGBD)
  - \* Reduzir acesso a disco
  - \* Com base na taxa de acerto
  - \* Parâmetro de instalação do SGBD
    - \* Pode ser feito automaticamente por certos DBMS
- \* Componente de Recovery
  - \* Disco de log não deve conter nenhum outro tipo de dados
  - \* Intervalo de geração de checkpoints
    - \* Aumentar ⇒ Reduz gargalo, mas aumenta tempo de Redo

## 7. Técnicas para o Ajuste de Performance - Sintonia Fina -

- \* Tuning a nível de projeto
- \* Particionamento de tabelas
  - \* Horizontal
    - \* Subconjuntos de tuplas freqüentemente acessados
  - \* Exemplo
    - \* Tabela de empregados pode ser particionada horizontalmente por lotação (departamento)

## 9. Performance de SBDs - Sintonia Fina -

- \* Tuning a nível de projeto
- \* Particionamento de tabelas
  - \* Vertical
    - \* Subconjunto de atributos freqüentemente acessados
  - \* Exemplo
    - \* Tabela de clientes de uma companhia telefônica pode ser particionada verticalmente
      - \* Atributos código, nome, telefone e
      - \* Atributos código, leitura-ant, leitura-atual



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 9. Performance de SBDs - Sintonia Fina -

### \* Particionamento de tabela

#### \* Oracle

##### \* Suporte para particionamento horizontal

#### \* Exemplo

##### \* Particionamento de Empregado por Lotação

#### \* Esquemas

##### \* Departamento(cod\_dep, nome, ender)

##### \* Empregado(matr, nome, ender, salário, lotação)



UNIVERSIDADE  
FEDERAL DO CEARÁ

## 9. Performance de SBDs - Sintonia Fina -

### \* Particionamento de tabela

#### \* Exemplo Oracle

```
CREATE TABLE empregado
(matr    NUMBER(9) PRIMARY KEY,
 nome   VARCHAR(50) NOT NULL,
 ender  VARCHAR(30),
 salário DECIMAL(9,2) NOT NULL,
 lotação NUMBER(5)
FOREIGN KEY lotação REFERENCES departamento)
PARTITION BY RANGE (LOTAÇÃO)
(PARTITION p1 VALUES LESS THAN (2) TABLESPACE Arq1
 PARTITION p2 VALUES LESS THAN (3) TABLESPACE Arq2
 PARTITION p3 VALUES LESS THAN (4) TABLESPACE Arq3
 PARTITION p4 VALUES LESS THAN (MAXVALUE) TABLESPACE
Arq4);
```

## 9. Performance de SBDs - Sintonia Fina -

### \* Particionamento de tabela

#### \* Oracle

- \* Inserção de novas tuplas em Empregado
  - \* Alocação automática na partição correta
- \* Consulta sobre Empregado

\* Listar os salários dos empregados do departamento 2

`SELECT salario FROM empregado WHERE lotação=2`

`SELECT salario FROM empregado PARTITION (p2)`

`CREATE VIEW emp-dep02 AS  
SELECT * FROM empregado PARTITION (p2)`

`SELECT salario FROM emp-dep02`

## 9. Performance de SBDs - Sintonia Fina -

### \* Particionamento de tabela

#### \* SQL Server

- \* Suporte para particionamento horizontal

#### \* Passos

\* Criação dos filegroups e respectivos arquivos

\* Criação da função de particionamento

\* Criação do esquema de particionamento

\* Criação da tabela particionada



## 9. Performance de SBDs

### - Sintonia Fina -

#### \* Particionamento de tabela

##### \* SQL Server

###### \* Criação da função de particionamento

\* Create partition function <nome\_função>(<tipo\_dado\_atributo>) as range left | right for values (V<sub>1</sub>, V<sub>2</sub>, V<sub>3</sub>, ..., V<sub>n</sub>)

\* Devem ser definidas n+1 partição

###### \* Opção Left (Default)

###### \* Primeira partição

\* Tuplas com atr\_part ≤ V<sub>1</sub>

\* atr\_part ∈ (min\_valor, V<sub>1</sub>]

###### \* Segunda partição

\* Tuplas com V<sub>1</sub> < atr\_part ≤ V<sub>2</sub>

\* atr\_part ∈ (V<sub>1</sub>, V<sub>2</sub>]

###### \* Última partição (n+1)

\* Tuplas com atr\_part > V<sub>n</sub>

\* atr\_part ∈ (V<sub>n</sub>, max\_val;or]

## 9. Performance de SBDs

### - Sintonia Fina -

#### \* Particionamento de tabela

##### \* SQL Server

###### \* Criação da função de particionamento

\* Create partition function <nome\_função>(<tipo\_dado\_atributo>) as range left | right for values (V<sub>1</sub>, V<sub>2</sub>, V<sub>3</sub>, ..., V<sub>n</sub>)

\* Devem ser definidas n+1 partição

###### \* Opção right

###### \* Primeira partição

\* Tuplas com atr\_part < V<sub>1</sub>

\* atr\_part ∈ [min\_valor, V<sub>1</sub>)

###### \* Segunda partição

\* Tuplas com V<sub>1</sub> ≤ atr\_part < V<sub>2</sub>

\* atr\_part ∈ [V<sub>1</sub>, V<sub>2</sub>)

###### \* Última partição (n+1)

\* Tuplas com atr\_part ≥ V<sub>n</sub>

\* atr\_part ∈ [V<sub>n</sub>, max\_val;or)



## 9. Performance de SBDs

### - Sintonia Fina -

#### \* Particionamento de tabela

- \* SQL Server
  - \* Criação do esquema de particionamento
    - \* Define partições e as associa aos filegroups
    - \* Create partition scheme <nome\_esquema> as partition nome\_Função\_part to ( nome\_fg, ... )
  - \* Criação da tabela particionada
    - \* Create table <nome\_tab> ( <esquema\_tab> ) on nome\_função\_part (atr\_part)

## 9. Performance de SBDs

### - Sintonia Fina -

#### \* Atividade de lab

- \* Particionar a tabela Lineitem\_P (TPC-H) em três partições
  - \* Cada partição deve estar alocada a um arquivo específico
  - \* Alocar tuplas nas partições com
    - \* Distribuição uniforme por número de tuplas por partição
- \* Povoar Lineitem\_P com dados de Lineitem

## 9. Performance de SBDs - Sintonia Fina -

- \* Tuning a nível de projeto
  - \* Reconstrução de tabelas particionadas
    - \* Particionamento horizontal
      - \*  $R=UR_i$
    - \* Particionamento vertical
      - \*  $R=\bowtie R_i$

## 9. Performance de SBDs - Sintonia Fina -

- \* Tuning a nível de projeto
  - \* De-normalização
    - \* Reduz joins
    - \* Simplifica o acesso
    - \* Quando??
      - \* Dados são usados frequentemente de forma conjunta (dados do empregado e nome do departamento onde está lotado)
      - \* Dado redundante é estável
      - \* Dado redundante em pouco volume

## 9. Performance de SBDs

### - Sintonia Fina -

- \* Tuning a nível de projeto
  - \* Índices
    - \* Reduz table scan
    - \* Evita operação de sort
      - \* Índice ordenado
    - \* B+ é mais eficiente que hash para consultas em extensão
    - \* Índice primário
      - \* Esparsos
        - \* Um índice por página de dados
    - \* Candidatos para índices
      - \* Colunas utilizadas em cláusulas
        - \* where, order by, group by e joins
      - \* Colunas com valores únicos

## 9. Performance de SBDs

### - Sintonia Fina -

- \* Tuning a nível de projeto
  - \* Utilização de Índices
    - \* Select ... from Funcionário where matr=123 or matr>540
      - \* Índice não é utilizado
    - \* Alternativa
      - \* Select ... from Funcionário where matr=123 union all
      - \* Select ... from Funcionário where matr > 540
  - \* Cálculo sobre coluna índice implica na não utilização do índice
    - \* ... Where salário /12 > 2000

## 9. Performance de SBDs

### - Sintonia Fina -

#### \* Índices Bitmap no Oracle

- \* Bitmap
- \* String de bits
  - \* Representa uma lista de TID
- \* TID
  - \* Um número inteiro que identifica uma tupla em uma tabela com cardinalidade igual N
  - \*  $TID \in \{0,1,2,3,\dots,N-1\}$

## 9. Performance de SBDs

### - Sintonia Fina -

#### \* Índices Bitmap no Oracle

- \* Construção do bitmap
  - \* Se uma tupla com TID igual a k está no bitmap
    - \* Bit com posição igual a k deve ter valor 1
  - \* Se uma tupla com TID igual a k não está no bitmap
    - \* Bit com posição igual a k deve ter valor 0
- \* Exemplo
  - \* Bitmap
    - \* 10011000010...
  - \* Tuplas com  $TID \in \{0,3,4,9\}$  estão no bitmap



## 9. Performance de SBDs

### - Sintonia Fina -

#### \* Índices Bitmap no Oracle

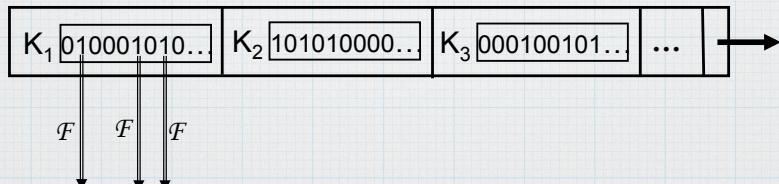
##### \* Estrutura

##### \* Árvore B+

- \* Para cada chave de busca nas folhas
  - \* Um bitmap associado

##### \* Funções

- \*  $F: TID \rightarrow$  Ponteiro
- \*  $F^{-1}: \text{Ponteiro} \rightarrow TID$



## 9. Performance de SBDs

### - Sintonia Fina -

#### \* Tuning a nível de projeto

- \* Controle de Concorrência (Bloqueios)
- \* Seleção da granulosidade
- \* Tipo de bloqueio (read, write, update)
- \* Nível de isolamento
  - \* Cursor stability preferível
  - \* Repeatable read

#### \* Consultas

- \* Tente se antecipar ao otimizador
- \* Tempo de parsing é muito alto
  - \* Armazenar consultas compiladas
- \* Intereração client-server é muito cara
  - \* Recupere conjuntos



## 9. Performance de SBDs - Sintonia Fina -

### \* Tuning a nível de projeto

#### \* Consultas

- \* Evite subselect, utilize sempre que possível joins
- \* Selecione apenas as colunas realmente relevantes
  - \* Colunas adicionais ⇒ CPU adicional
- \* Evite SELECT \*
- \* Consider a possibilidade de utilizar sort externo no lugar de order by
  - \* Para um grande volume de tuplas

## Exemplo de Cursor

```
#include <stdio.h>
exec sql include sqlda;
int main()
{
    EXEC SQL BEGIN DECLARE SECTION;
    char usuario[20], servidor_bd[20]; int cod; float pr_venda_c;
    EXEC SQL END DECLARE SECTION;
    EXEC SQL CONNECT TO :servidor.bd USER :usuário;
    EXEC SQL DECLARE C1 CURSOR SCROLL_LOCKS
        FOR SELECT cod_item,pr_venda FROM Estoque WHERE pr_venda < 500 FOR
        UPDATE pr_venda;
    EXEC SQL OPEN C1;
    EXEC SQL BEGIN TRANSACTION;
    EXEC SQL SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
    EXEC SQL FETCH NEXT C1 INTO :cod,:pr_venda_c;
    while(SQLCODE==0)
    {
        pr_venda_c=pr_venda_c*1.15;
        EXEC SQL UPDATE estoque set pr_venda=:pr_venda_c where cod_item=:cod;
        EXEC SQL FETCH NEXT C1 INTO :cod,:pr_venda_c;
    }
    EXEC SQL CLOSE C1;
    EXEC SQL COMMIT TRANSACTION;
    EXEC SQL DISCONNECT CURRENT; return (0); }
```



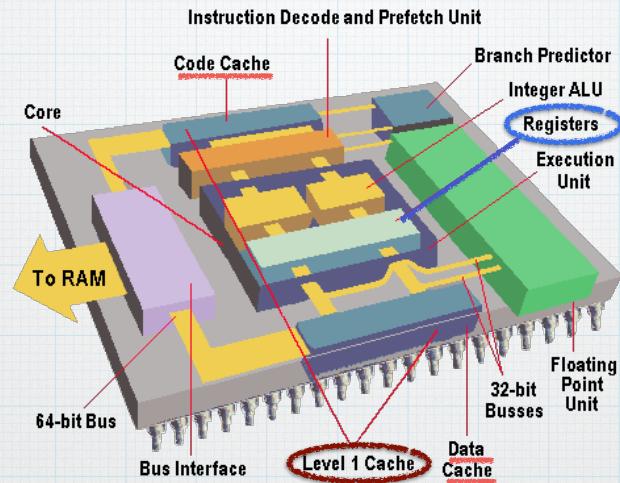
UNIVERSIDADE  
FEDERAL DO CEARÁ

## Exemplo de Cursor (2)

```
declare @cd_mun int, @cd_zona int, @cd_secao int, @cd_cargo int,  
@cd_nr_candidato int, @votos int  
DECLARE c_teste CURSOR scroll_locks for  
select * from ce_voto_secao with (XLOCK)  
SET TRANSACTION ISOLATION LEVEL READ COMMITTED  
begin transaction  
open c_teste  
fetch next from c_teste  
into @cd_mun, @cd_zona, @cd_secao, @cd_cargo, @cd_nr_candidato,  
@votos  
while @@FETCH_STATUS = 0  
begin  
print @cd_mun  
fetch next from c_teste  
into @cd_mun, @cd_zona, @cd_secao, @cd_cargo, @cd_nr_candidato,  
@votos  
end  
close c_teste  
DEALLOCATE c_teste  
commit
```



## Processor Architecture - An Abstract Model -





UNIVERSIDADE  
FEDERAL do CEARÁ

## HD Drive

### - Information Encoding -

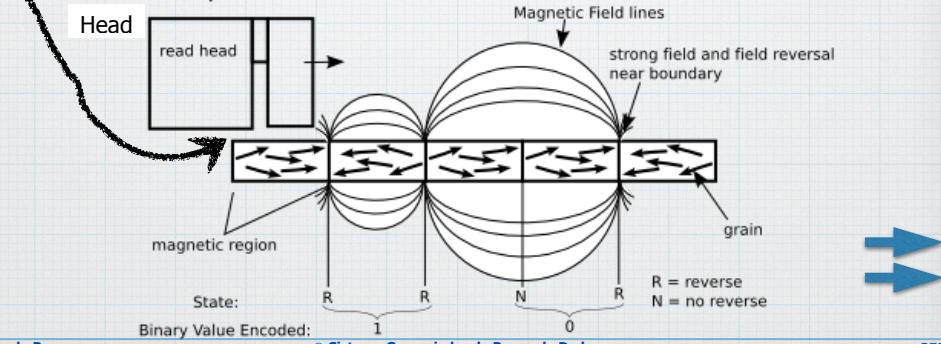
#### \* Head

\* One head for each surface, riding close to the surface

\* Never touches the surface

\* Reads and alters magnetism

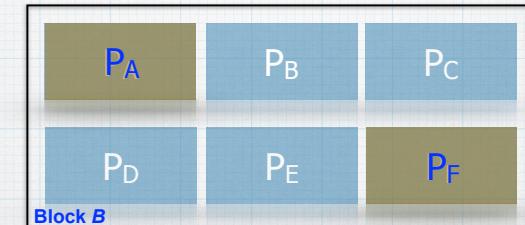
\* R/W operation



UNIVERSIDADE  
FEDERAL do CEARÁ

## Program Operation

### - Physical Write -



S1: Block B with **clean** pages (all bits set to 1)

S2: New data should be inserted into pages

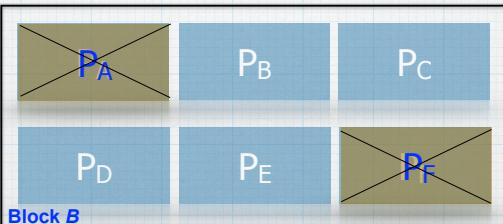
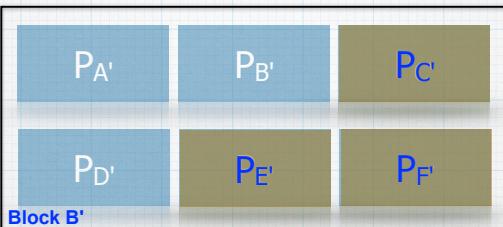
P<sub>A</sub> and P<sub>F</sub>

\* **Program operation**



UNIVERSIDADE  
FEDERAL do CEARÁ

## Logical Write

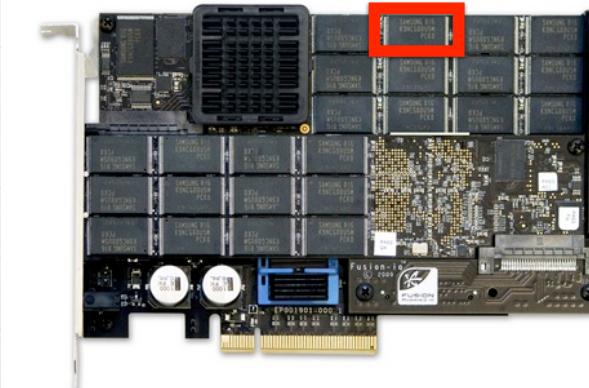


- S3: New data should be inserted into page  $P_E$  of  $B$
- ★ Program operation can not be execute any more
  - ★ Pages of  $B$  are rewritten in a new clean block  $B'$
  - ★ Page  $P_E$  is written on block  $B'$
  - ★ Pages  $P_A$  and  $P_F$  (block  $B$ ) become stale pages

357



UNIVERSIDADE  
FEDERAL do CEARÁ

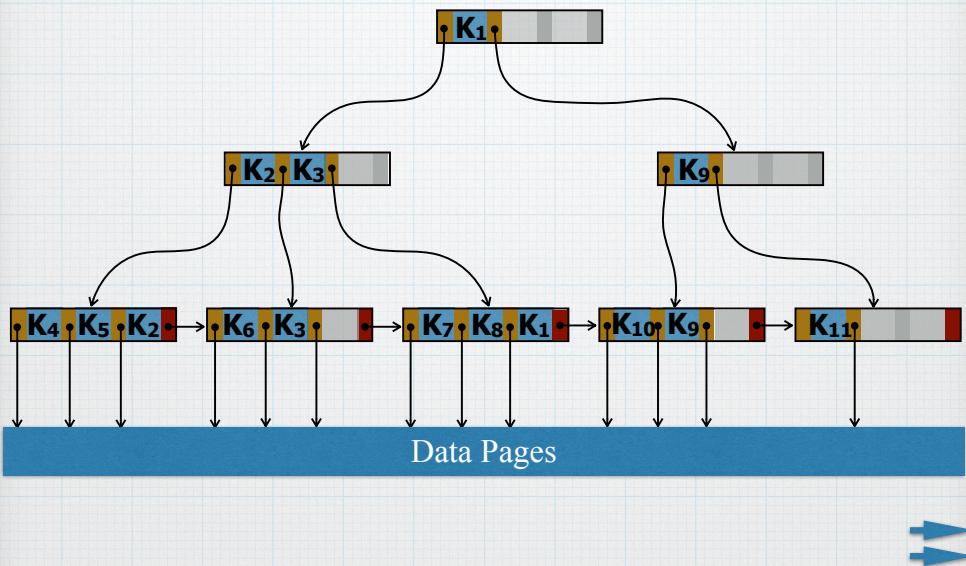


358

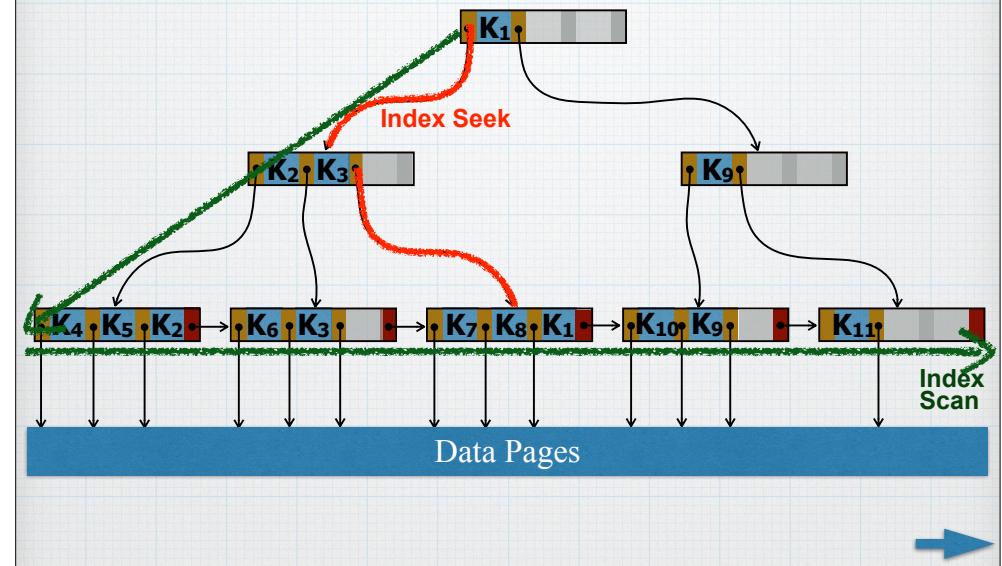


UNIVERSIDADE  
FEDERAL do CEARÁ

## 4. Indexing - B+-Tree Anatomy -

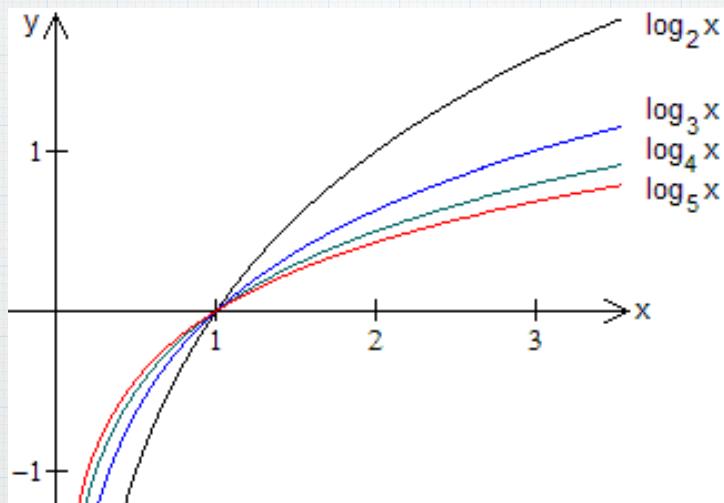


## 4. Indexing - Operations on a B+-Tree -

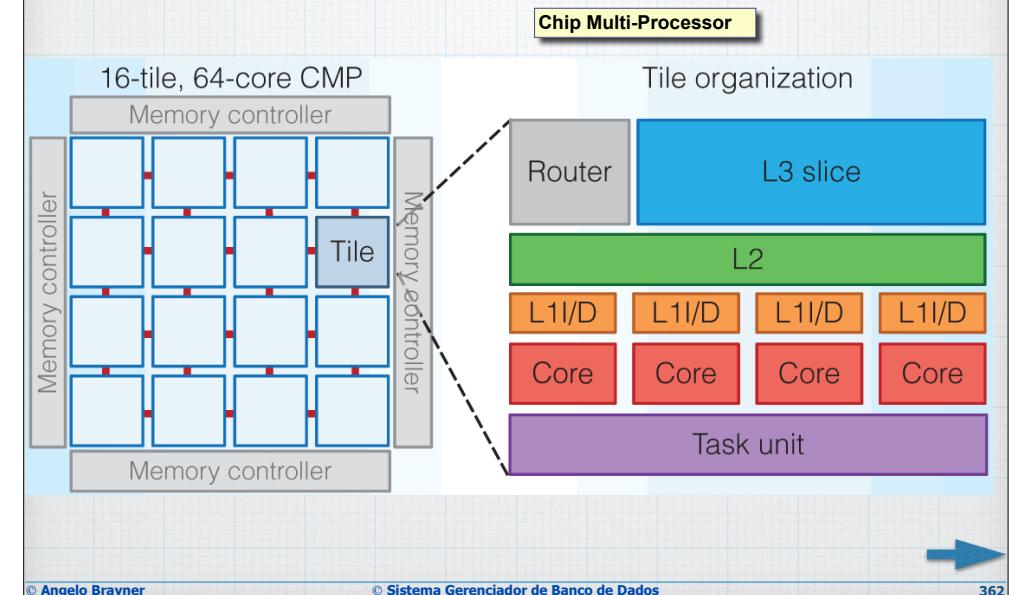


## 4. Indexing

- Operations on a B+-Tree -



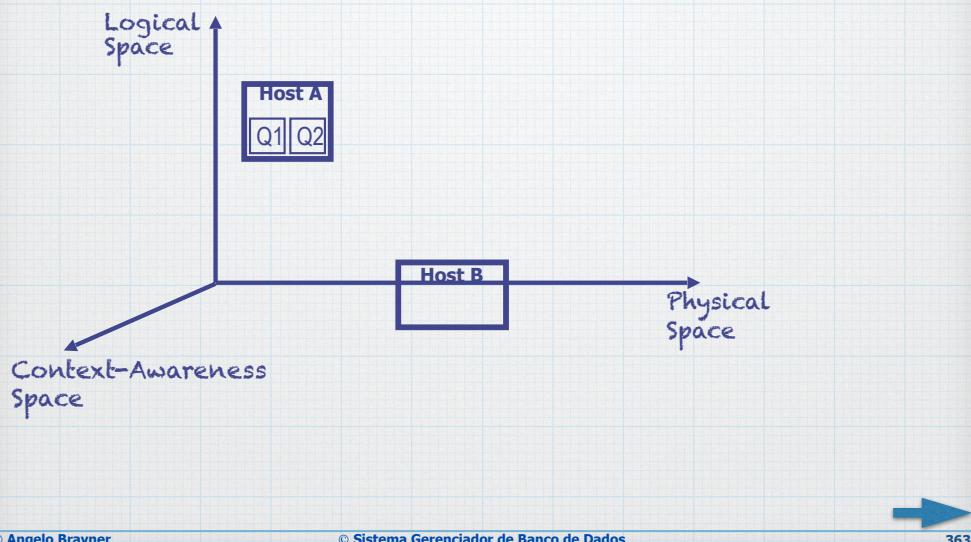
## SBDs Paralelos





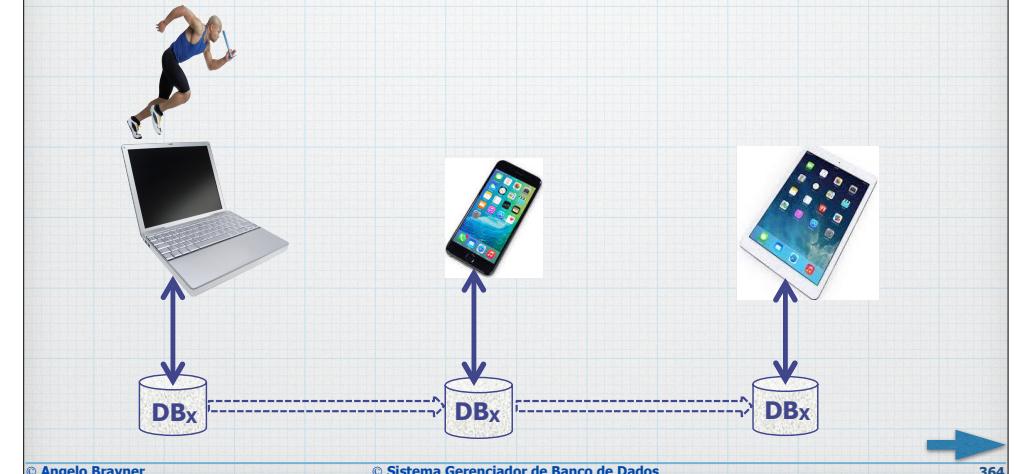
UNIVERSIDADE  
FEDERAL do CEARÁ

## SBDS Móveis



## SBDS Multi-Plataforma

- ⇒ Cada plataforma visitada
- ⇒ Replicação do banco de dados sob demanda



## Sistemas de Fluxos de Dados

SNQL

```
SELECT r.rID, AVG(t.collectedValue), COUNT(t.collectedValue),
FROM Humidity h, Temperature t, SensorRegion r
WHERE r.regionId = t.regionId and r.regionId = t.regionId
AND r.initialLatitude > 034308 AND r.FinalLatitude < 034316
AND r.initialLongitude > 383151 AND r.FinalLongitude < 383114
AND t.collectedValue > 40
AND h.collectedValue < 0.7
Group by r.rID
```

**TIME WINDOW** 3600

**SEND INTERVAL** 120

**SENSE INTERVAL**

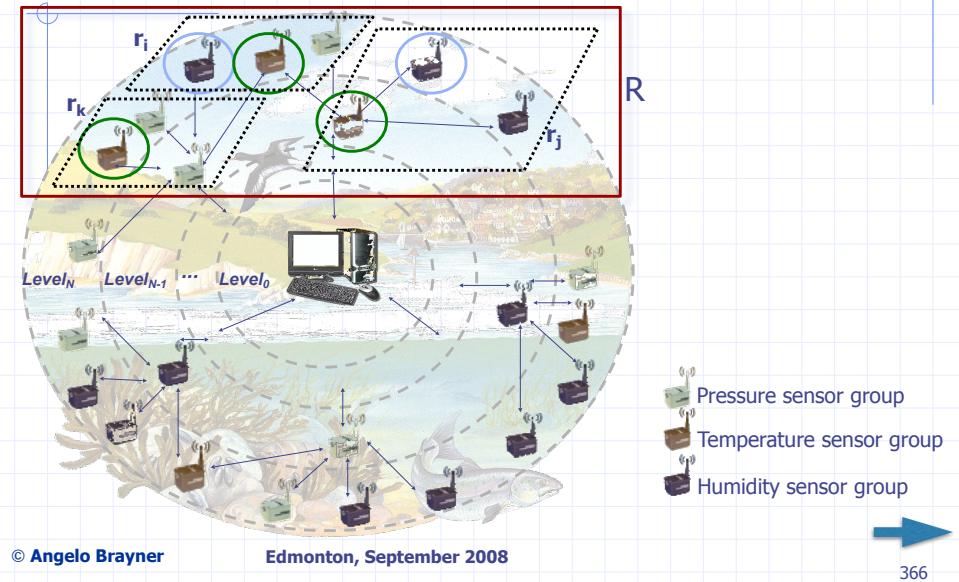
10

**SCHEDULE** 1 'Sept-26-2019 14:00:00'

R

## Sistemas de Fluxos de Dados

In-network Query Processing



© Angelo Brayner

Edmonton, September 2008

366

© Angelo Brayner

Edmonton, September 2008

365

Statistics for INDEX '_WA_Sys_00000001_7B905C75'.		
Name	Updated	Rows
_WA_Sys_00000001_7B905C75	Apr 3 2019 12:09PM	6001215
All Density	Average Length	Columns
2.667591E-05	4	l_orderkey
Histogram Steps		
RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS
68	0	127.7049
64036	64359.68	314.8813
70947	17840.05	314.8813
90726	25789.04	314.8813
118595	28137.6	314.8813
179905	34596.15	314.8813
206438	18562.69	314.8813
221543	19420.82	314.8813
242886	22175.86	314.8813
272738	20143.45	314.8813
326625	61785.3	314.8813
359111	28724.74	314.8813
399683	46158.31	314.8813
471811	53700.82	314.8813
488708	23395.31	314.8813
499585	17298.08	314.8813
516449	20866.09	314.8813
533665	19240.16	314.8813
596487	55642.9	314.8813
681350	66934.07	314.8813
701508	18746.54	314.8813



## 5. Query Processing - Join Physical Operators -

\* In-memory hash join ( $h = \lfloor ID/3 \rfloor$ )

Hash table (in-memory)

Bucket 0	1	●
Bucket 1	3	●
Bucket 2	7	●
Bucket 3	9	●
Bucket 4	14	●

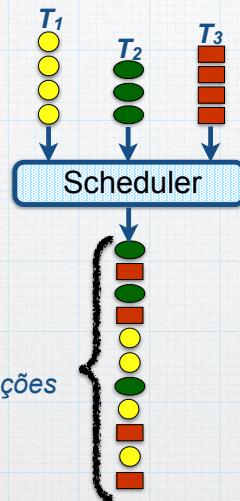
Build table (disk)

9	Yasmim	120,000
1	Barbara	160,000
10	Rebeca	120,000
7	Sofia	125,000
3	Caio	122,000
2	Andre	170,000
4	Lucas	120,000
14	Lara	130,000

## 6. Processamento de Transações

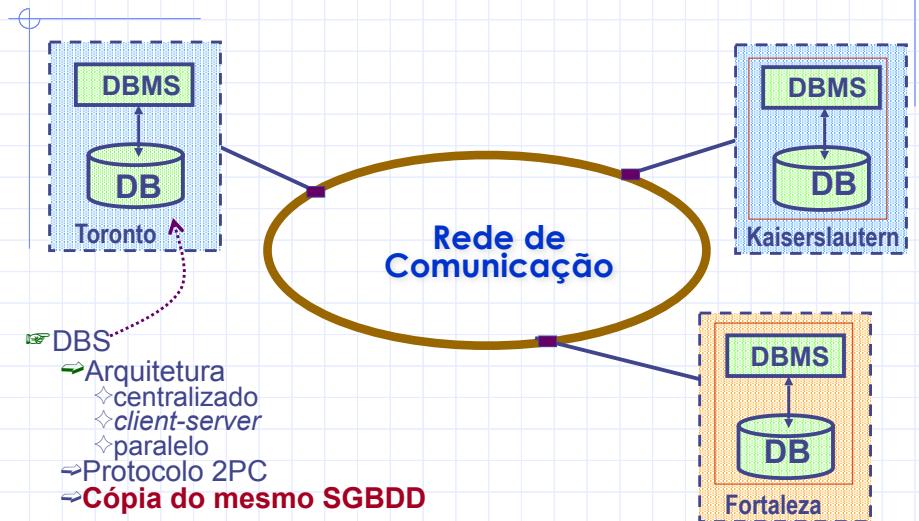
- O Problema de Concorrência em SBDs -

*Entrelacamento de operações de transações*



369

## Sistemas de Bancos de Dados Distribuídos

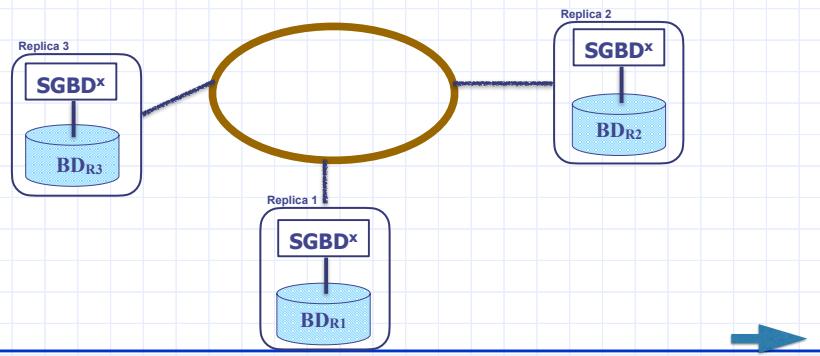


370



## Replicação de Bancos de Dados

- ▶ Definição
- ▶ Mecanismo para manter várias réplica (cópias) dos dados
- ▶ Disponíveis para acesso



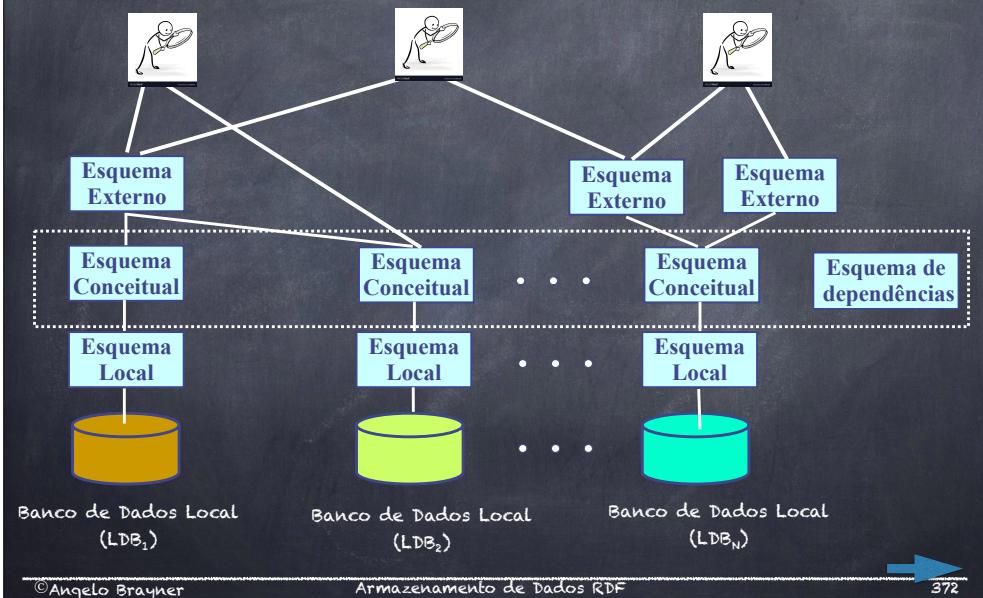
© Angelo Brayner

Sistemas de Bancos de Dados Distribuídos

371



## Integração de Fontes de Dados Heterogêneas - Sistema de Banco de Dados Múltiplos -



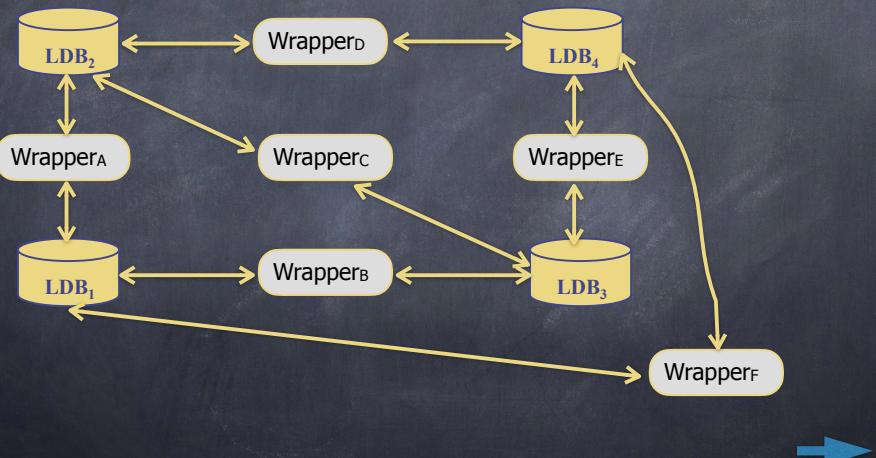
© Angelo Brayner

Armazenamento de Dados RDF

372

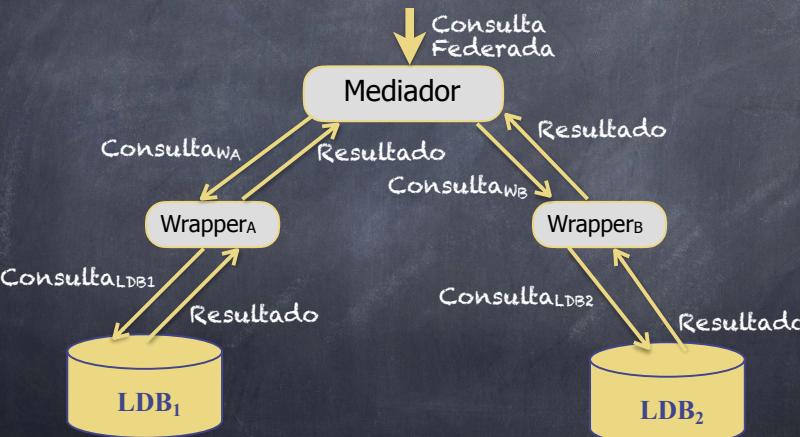
## Integração de Fontes de Dados Heterogêneas - Sistema de Banco de Dados Federados -

### Abordagem wrappers



## Integração de Fontes de Dados Heterogêneas - Sistema de Banco de Dados Federados -

### Mediadores





UFC

CC

DC

## Integração de Fontes de Dados Heterogêneas - Conceitos Básicos -

### Classificação

