Capítulo 3

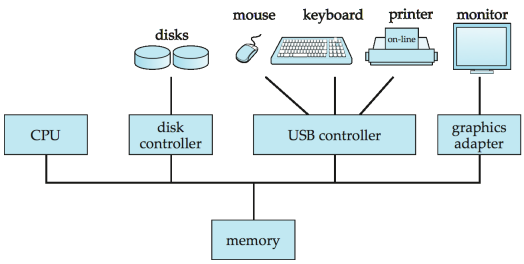
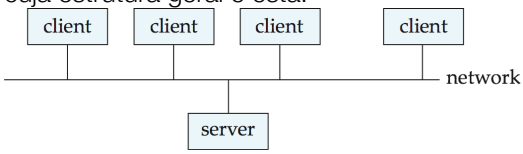
Arquiteturas de Sistemas de BD

Sistema Centralizado

Um sistema centralizado é aquele que corre numa só máquina e não interage com outros computadores.
Existem dois modelos de sistemas centralizados:

1. Single-user System **-** computador de mesa, utilizador único, geralmente tem apenas um CPU e um ou dois discos rígidos; o sistema operativo pode oferecer suporte a apenas um utilizador.

2. Multi-user System **-** mais discos, mais memória, várias CPUs e um sistema operativo de múltiplos utilizadores. Serve assim, um grande número de utilizadores que estão conectados ao sistema através de terminais. Muitas vezes chamados *sistemas de servidor*.

Sistema Centralizado	**Sistema Cliente-Servidor**
	Os sistemas de servidor satisfazem os pedidos geradas em sistemas de *n* clientes, cuja estrutura geral é esta: 

Sistema Cliente-Servidor

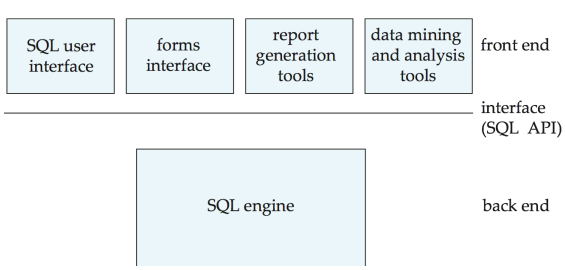
O funcionamento deste sistema pode ser dividido em:

- ***Back-end:***
  Gere estruturas de acesso, avaliação de consultas e otimização, controlo de concorrência e recuperação.
- ***Front-end:***
  Consiste em ferramentas tais como formulários, relatórios e interface gráficas de utilizador.

A interface entre o *front-end* e o *back-end* é através de SQL ou através de uma interface de um programa.

Vantagens de substituição de *mainframes* com redes de estações de trabalho ou computadores pessoais ligados a máquinas de servidores back-end:

- Melhor funcionalidade para o custo
- Flexibilidade na localização de recursos e expansão de instalações
- Melhores interfaces de utilizador
- Manutenção mais fácil

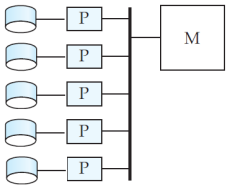
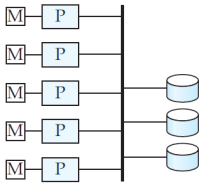
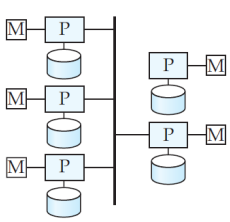
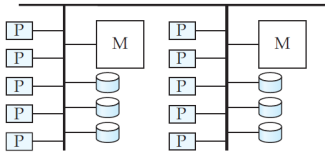


Sistema Paralelo

Sistemas de base de dados paralelos consistem em múltiplos processadores e múltiplos discos ligados numa rede de interconexão rápida.

- A máquina *coarse-grain parallel* consiste de um pequeno número de processadores poderosos
- A máquina *massively parallel* ou *fine gain parallel* utiliza milhares de processadores menores.
- Duas principais medidas de desempenho:
  - rendimento - número de tarefas que podem ser concluídas num determinado intervalo de tempo
  - tempo de resposta - quantidade de tempo que leva para completar uma única tarefa a partir do momento em que são fornecidas

Arquiteturas das Base de Dados Paralelas

Memória Partilhada	Disco Partilhado
- Processadores e discos acedem a uma memória comum, através de uma rede de interconexão - Comunicação extremamente eficiente entre processadores - dados na memória compartilhada podem ser acessados por qualquer processador sem precisar movê-lo, utilizando o software. - *Downside* - Arquitetura não-escalável - Utilizado para níveis mais baixos de paralelismo - entre 4 a 8 	- Todos os processadores podem aceder diretamente a todos os discos através de uma rede de interconexão, mas os processadores têm memórias privadas - Arquitetura tolerante a falhas - se um processador falhar, os outros processadores podem assumir as suas funções, desde que a base de dados seja residente num dos discos que estejam acessíveis aos processadores - *Downside* - *bottleneck* ocorre na interligação ao subsistema de disco - Podem ser dimensionado para um número maior de processadores, mas a comunicação entre processadores é mais lenta. 
Nenhuma Partilha	**Hierárquica**
- Nó é composto por um processador, memória e um ou mais discos. Os processadores num nó comunicam-se com outro processador em outro nó, usando uma rede de interconexão. Um nó funciona como um servidor para os dados no disco ou discos que possui o nó. - Escalável até milhares de processadores sem interferência. - Principal Desvantagem: custo de comunicação e acesso ao disco não-local; o envio de informação envolve a interação de software em ambos os pontos. 	- Combina as características das arquiteturas de memória partilhada e de disco partilhado - Nível superior é atingido numa arquitetura sem partilha, onde os nós estão conectados numa rede de interconexão, e não partilham disco e memória com o outro. - Cada nó do sistema podia ser um sistema de memória partilhada com alguns processadores. - Alternativamente, cada nó pode ser um sistema de disco partilhado, e cada um dos sistemas de partilha de um conjunto de discos pode ser um sistema de memória partilhada. 

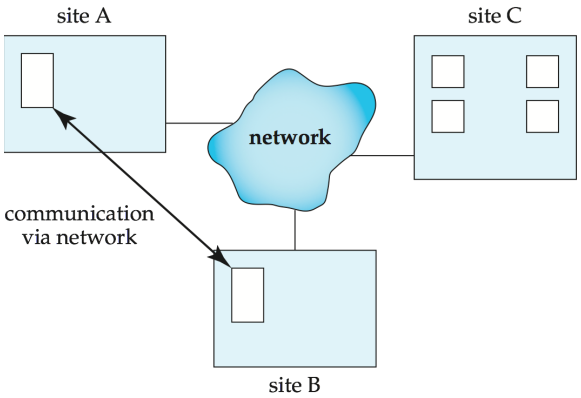
Base de Dados Distribuídas

• ***Base de Dados Distribuídas Homogéneas***

- Mesmo software/esquema em todos as localizações, onde a informação pode ser particionada entre localizações
- Objetivo: Fornecer uma visão de uma única base de dados, escondendo detalhes de distribuição

• ***Base de Dados Distribuídas Heterogéneas***

- Software/esquema diferente em localizações diferentes
- Objetivo: Integrar bases de dados existentes para fornecer funcionalidade útil



Diferença entre transações *locais* e *globais*

• **Transações *locais***

Uma transação local acede a dados na localização em que a transação foi iniciada.

• **Transações *globais***

A transação global acede dados tanto numa localização diferente daquele em que a operação foi iniciada como acede a dados em diferentes localizações

Vantagens

- Partilha de Dados - Utilizadores acedem a dados remotos
- Autonomia - Um nó pode controlar os seus próprios dados
- Disponibilidade - Se uma localização falha, o sistema pode continuar ativo (replicação dos dados)

Desvantagens

- Arquitetura de software mais complexa
- Maiores custos no desenvolvimento de software
- Mais *bugs*
- Aumento da cobrecarga de processamento

Soluções implementadas

- Atomicidade dos dados em dois ou mais nós
- Protocolo de duas fases:
  - Cada transação é executada e só aceite por um coordenador
  - Cada localização segue a decisão de um coordenador

Tipos de Rede

• Local-area networks (LANs)

- Constituída por processadores que estão distribuídos em pequenas áreas geográficas, como um único edifício ou alguns edifícios adjacentes.
- Maior largura de banda e Baixa Latência

• Wide-area networks (WANs)

- Constituída por processadores distribuídos numa grande área geográfica.
- Baixa largura de banda e Maior latência

Indicadores de Desempenho

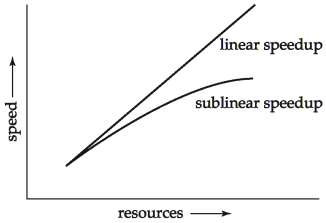
- Capacidade: O número de tarefas que são concluídas num determinado intervalo de tempo
- Tempo de resposta: A quantidade de tempo que leva para completar uma única tarefa a partir do momento em que são fornecidas

Acelerar e Aumentar (Speed-Up e Scale-Up)

• Acelerar - Speed-Up

- Um problema é executado num sistema pequeno e é recolhido por um sistema N vezes maior.

Medido por: speedup = $\frac{\text{tempo decorrido no sistema menor}}{\text{tempo decorrido no sistema maior}}$

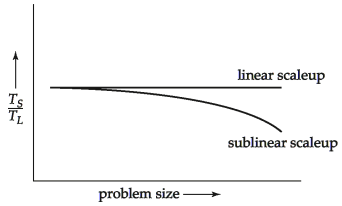


- Speed-Up é linear se a equação for igual a N.

• Aumentar - Scale-Up

- Aumenta tanto o tamanho do problema como o de sistema
- Sistema N vezes maior para realizar operações N vezes maiores

Medido por: scaleup = $\frac{\text{tempo decorrido no problema menor}}{\text{tempo decorrido no sistema maior}}$



- Scale-Up é linear se a equação for igual a 1.

- Batch Scale-Up

- Tarefa grande, tipico na maioria das consultas de apoio à decisão e simulação científica.
- Necessário usar um computador N vezes maior em problemas N vezes maiores

- Transaction Scale-Up

- Consultas submetidas por utilizadores independentes a uma base de dados partilhada
- Adequado para execução paralela

• Fatores Limitantes

Speed-Up e Scale-Up são frequentemente sublinear devido a:

- Custos de iniciação: custo de iniciar vários processos pode dominar o tempo de computação, se o grau de paralelismo for alto.
- Interferência: Processos que acedem a recursos compartilhados
- Inclinação: Aumentar o grau de paralelismo aumenta a variância em tempos de serviço de tarefas em execução paralelamente. Tempo total de execução é determinado pela mais demorada das tarefas paralelas em execução.

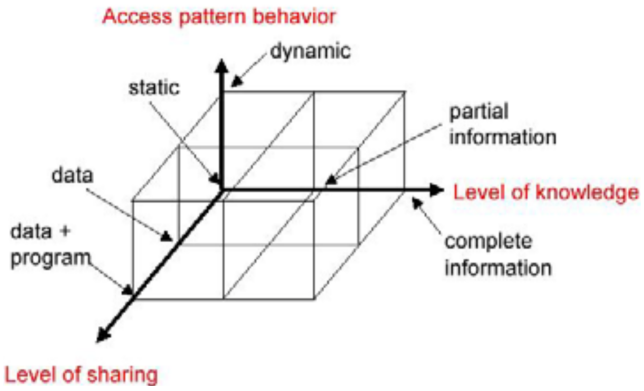
Estrutura de Base de Dados Distribuídas

Problema: Tomar decisões sobre a localização de dados e programas nos nós de uma rede de computadores, assim como possivelmente projetando a rede em si.

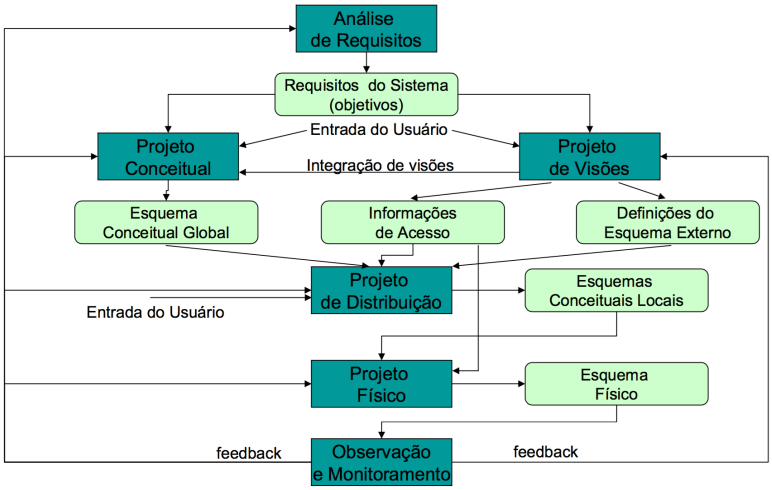
Sistema Gestão Base Dados Distribuídas (SGBDD)

Nos SGBDD, a localização das aplicações engloba:
- Localização do software SGBDD
- Localização das aplicações que executam sobre a base de dados

• ***Dimensões do problema:***
- Nível de partilha:
  - não partilha
  - partilha os dados
  - partilha os dados + programa
- Comportamento de padrões de acesso:
  - estático ou dinâmico
- Nível de conhecimento sobre o comportamento de acesso padrão:
  - nenhuma informação,
  - informação parcial,
  - informações completas



• ***Estratégias alternativas da Estrutura de Distribuição:***
- Abordagem descendente:
  - Na maior parte das vezes no projeto de sistemas a partir do zero
  - Na maior parte das vezes em sistemas homogéneos



-Abordagem ascendente:
  - Quando a base de dados já existe em alguns nós

Fragmentação

Processo de divisão dos tuplos de uma tabela em duas ou mais tabelas (Fragmentos)

- Muitos fatores contribuem para o design ideal
- Informações necessárias:
  - Informações da base de dados
  - Informações da aplicação
  - Informações da rede de comunicação
  - informações do sistema do computador

Vantagens	Desvantagens
- Confiança - Desempenho - Equilibrar a capacidade de armazenamento e custos - Custos de comunicação - Segurança	- Se fragmentos não forem exclusivamente mútuos - Degradação do desempenho em *Joins*

Granularidade

O que é uma unidade de distribuição razoável?

- Relações?

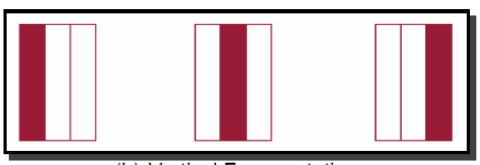
  - Se as queries precisarem de todos os dados da relação e os dados permanecem na localização que usam os dados
    - E se esta não for replicada, então irá haver um alto volume de acessos remotos
    - Se for replicada, irá haver problemas em atualizações

- Fragmentos de relações como unidades de distribuição:
  - As aplicações precisam de um subconjunto de relações
  - As transações podem ocorrer simultaneamente
  - Execução paralela de uma única query
  - Fragmentos menores melhoram desempenho
  - Degradação do desempenho em *Joins*
  - Mas o controlo de dados semântico é mais difícil

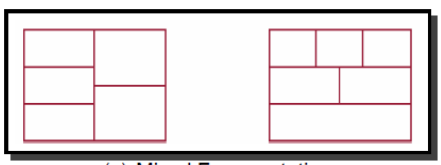
Fragmentos são geralmente a unidade apropriada e não relações inteiras

Tipos

(a) Horizontal Fragmentation



(b) Vertical Fragmentation



(c) Mixed Fragmentation

• *Fragmentação Horizontal*

- Mantém-se as colunas
- Obedece-se à regra imposta

Distribui os tuplos da relação entre os fragmentos

Fragmentação horizontal da relação PROJ

- PROJ1: projetos com orçamentos < 200.000
- PROJ2: projetos com orçamentos >= 200.000

PROJ

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Develop	135000	New York
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris
P5	CAD/CAM	500000	Boston

$PROJ_1$

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Develop	135000	New York

$PROJ_2$

PNO	PNAME	BUDGET	LOC
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris
P5	CAD/CAM	500000	Boston

• **Fragmentação Horizontal Primária**

Operação de selecção sobre uma relação proprietária

$$R_j = \sigma_{Fj}(R), 1 \le j \le w$$

onde F *j* é a fórmula de seleção, que é (preferencialmente) um predicado minterm

Então...

-Um fragmento horizontal R_i da relação R consiste em todos os tuplos de R que satisfazem o predicado minterm m_i

-Dado um conjunto de predicados minterm M, existem tantos fragmentos horizontais da relação R quantos forem os predicados minterm.

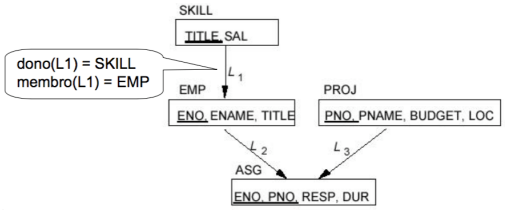
-Conjunto de fragmentos horizontais também é referido como conjunto de fragmentos minterm

• Requisitos de Informação

- Informações da BD

- Esquema conceitual global
  - relações, relacionamentos, cardinalidade

- Informações da Aplicação



Predicados simples (clausulas WHERE)**:**	***Predicados minterm*** (conjunto de predicados simples)**:**
- Dada uma relação R[A_1 , A_2 , ..., A_n], um predicado simples pj é $p_j : A_i \ \theta \ Valor$ onde ∈ {=,<,≤,>,≥,≠}, Valor σ D_i e D_i é o domínio de A_i. Para a relação R definimos $Pr = \{p_1, p_2, ..., p_m\}$ o conjunto de todos os predicados simples definidos sobre R Ex: PNAME = "Maintenance“, BUDGET ≤ 200000	- Dada uma relação *R* e $Pr = \{p_1, p_2, ..., p_m\}$, definimos o conjunto de predicados minterm $M = \{m_1 , m_2 , \ldots, m_r\}$ como $M=\{m_i \mid m_i = \sigma_{p_j^* \in Pr} p_j^*\}, 1 \le j \le m, 1 \le i \le z$ onde $p_j^* = p_j$ ou $p_j^* = \neg(p_j)$

Exemplo

m_1 : PNAME="Maintenance“e BUDGET ≤200000

m_2 : **NOT**(PNAME="Maintenance") e BUDGET ≤200000

m_3 : PNAME= "Maintenance“ e **N**OT(BUDGET ≤200000)

m_4 : **NOT**(PNAME="Maintenance") e **N**OT(BUDGET ≤200000)

Seletividade de minterm sel(m_i)**:**	***Frequências de acesso*** acc(q_i)**:**
- Quantidade de tuplos da relação que seria acessada por uma consulta do utilizador especificada de acordo com um dado predicado minterm	- Frequência com o que uma aplicação q_i do utilizador acede aos dados - Frequência de acesso minterm também pode ser definida

- Algoritmo

Dados: Uma relação R, o conjunto de predicados simples P_r

Saída: Conjunto de fragmentos de R = {R_1, R_2,…,R_w}, que obedecem as regras de correção da fragmentação

Sendo que: P_r deve ser completo

P_r deve ser mínimo

P_r tem que ser completo	**P_r tem que ser mínimo**
Um conjunto de predicados simples P_r é considerado completo sse existe uma probabilidade igual de acesso de cada aplicação a qualquer tuplo pertencente a qualquer fragmento minterm definido de acordo com Pr	- Se um predicado influencia o modo como a fragmentação é efetuada (faz com que um fragmento f seja ainda mais fragmentado em, digamos, f_i e f_j) deve haver pelo menos uma aplicação que acesse f_i e f_j de maneira diferenciada - Cada fragmento simples deve ser "relevante" na determinação de uma fragmentação - Se todos os predicados de um conjunto P_r são relevantes, P_r é mínimo
Exemplo Considere que a relação PROJ[PNO,PNAME,BUDGET,LOC] tem 2 aplicações definidas sobre ela: (1) Encontre os orçamentos dos projetos de cada localidade (2) Encontre os projetos com orçamentos menores que $200000 De acordo com (1), P_r={LOC="Montreal",LOC="New York",LOC="Paris"} ...que não é completo com respeito a (2). Modifique P_r ={LOC="Montreal", LOC="New York", LOC="Paris", BUDGET≤200000, BUDGET>200000} ...que é completo	***Exemplo*** P_r={LOC="Montreal",LOC="New York", LOC ="Paris", BUDGET≤ 200000,BUDGET>200000} é mínimo (e completo). No entanto, se adicionarmos PNAME = "Instrumentation" então Pr não é mínimo.

EXEMPLO

-2 relações : PAY e PROJ

- Fragmentação da relação PAY

-Aplicação: Verifica as informações sobre salário e determina um aumento adequado

-Registros de funcionários são administrados em 2 nós (aplicativo executado em 2 nós)

-Predicados simples

p_1 : SAL ≤ 30000

p_2 : SAL > 30000

P_r = {p_1,p_2}, que é completo e mínimo

-Predicados minterm

m_1 : (SAL ≤ 30000)

m_2 : NOT(SAL ≤ 30000) = (SAL > 30000)

PAY_1

TITLE	SAL
Mech. Eng.	27000
Programmer	24000

PAY_2

TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000

- Fragmentação da relação PROJ

-Aplicação:

-Procura nomes e orçamentos de projetos dados nas suas localizações (executado em 3 nós)

-Acessa informações de projetos de acordo com o orçamento “information according to budget” (um nó acessa ≤ 200000 e outro acessa > 200000

- Predicados simples

-Aplicação 1:

p_1 : LOC = “Montreal”

p_2 : LOC = “New York”

p_3 : LOC = “Paris”

-Aplicação 2:

p_4 : BUDGET ≤ 200000

p_5 : BUDGET > 200000

$P_r = \{p_1, p_2, p_3, p_4, p_5\}$

- Fragmentos minterm restantes após eliminação dos contraditórios

m1 : (LOC = “Montreal”) e (BUDGET ≤ 200000)

m2 : (LOC = “Montreal”) e (BUDGET > 200000)

m3 : (LOC = “New York”) e (BUDGET ≤ 200000)

m4 : (LOC = “New York”) e (BUDGET > 200000)

m5 : (LOC = “Paris”) e (BUDGET ≤ 200000)

m6 : (LOC = “Paris”) e (BUDGET > 200000)

$PROJ_1$

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal

$PROJ_2$

PNO	PNAME	BUDGET	LOC
P2	Database Develop.	135000	New York

$PROJ_4$

PNO	PNAME	BUDGET	LOC
P3	CAD/CAM	250000	New York

$PROJ_6$

PNO	PNAME	BUDGET	LOC
P4	Maintenance	310000	Paris

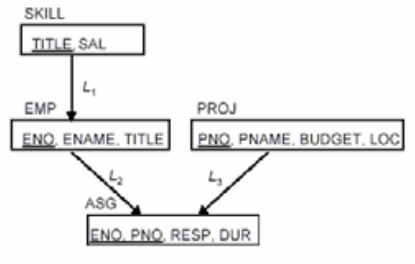
- Correção

Completude	Reconstrução	Disjunção
Já que P_r é completo e mínimo, os predicados de seleção são completos	Se a relação R está fragmentada em $F_R = \{R_1, R_2, \ldots, R_n\}$ $R = \cup_{\text{para todo } R_i \in F_R} R_i$	Predicados minterm que foram a base da fragmentação têm que ser mutuamente exclusivos

• **Fragmentação Horizontal Derivada**

Definida sobre uma relação membro de uma ligação de acordo com uma operação de seleção especificada sobre sua dona

- Cada ligação é uma junção



Dada uma ligação L onde dono(L)=S e membro(L)=R, os fragmentos horizontais derivados de R são definidos como

$$R_i = R \ltimes S_i,\ 1 \le i \le w$$

Onde *w* é o número de fragmentos definidos sobre R e

$$S_i = \sigma_{F_i}(S)$$

Onde F_i é a fórmula segundo a qual o fragmento horizontal primário S_i é definido

EXEMPLO

Dada a ligação L1 onde dono(L1)=SKILL e membro(L1)=EMP

EMP1 = EMP ⋉ SKILL1 **FHD**

EMP2 = EMP ⋉ SKILL2

onde

SKILL1 = σ SAL≤30000 (SKILL)

SKILL2 = σ SAL>30000 (SKILL) **FHP**

EMP_1

ENO	ENAME	TITLE
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E7	R. Davis	Mech. Eng.

EMP_2

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng.
E2	M. Smith	Syst. Anal.
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E8	J. Jones	Syst. Anal.

-Correção

Completude	Reconstrução	Disjunção
Integridade referencial	Mesmo que a FHP	Grafos de junção simples entre fragmentos

• *Fragmentação Vertical*

A fragmentação vertical particiona uma relação R num conjunto menor que Ri, usando a operação de projeção

- Cada Ri contém um subconjunto de atributos de R
- Também utilizada em sistema gestão base de dados centralizados

$$\Pi_{A_1,A_2,\ldots,A_n}(R)$$

· Tipos

Agrupamento	**Divisão**
- Atributos para fragmentos	- Relações para fragmentos
- Fragmentos sobrepostos	- Fragmentos não sobrepostos

Atributos chave replicados não são considerados sobrepostos

- Facilidade para garantir dependência funcionais (verificação de integridade, etc…)

• Requisitos de Informação

- Afinidade de atributos - aff (A_i, A_j):

- Indica a proximidade com que os atributos estão relacionados
- Obtida a partir de dados mais primitivos

A medida de afinidade de atributos entre dois atributos A_i e A_j de uma relação $R[A_1, A_2, ..., A_n]$ com respeito ao conjunto de aplicações $Q = (q_1, q_2, ..., q_q)$ é definida como:

$$\text{aff}(A_i, A_j) = \sum_{\text{todos as aplicações que acessam Ai e Aj}} (\text{acesso do aplicativo})$$

$$\text{acesso da aplicação} = \sum_{\text{todos os nós}} \text{frequência acessa aplicaticação} * (\text{n° acessos/execução})$$

Suponha que cada consulta do exemplo anterior acessa os atributos uma vez durante cada execução.
Suporta as frequências de acesso.

	S_1	S_2	S_3
q_1	15	20	10
q_2	5	0	0
q_3	25	25	25
q_4	3	0	0

Então:

- $aff(A_1, A_3) = 15*1 + 20*1 + 10*1 = 45$
- e a matriz de afinidades de atributos AA:

	A_1	A_2	A_3	A_4
A_1	45	0	45	0
A_2	0	80	5	75
A_3	45	5	53	3
A_4	0	75	3	78

- Valores de uso de atributos:

- Dado um conjunto de aplicações $Q = \{q_1, q_2, ..., q_q\}$ executadas sobre a relação $R[A_1, A_2, ..., A_n]$

$$\text{use}(q_i, A_j) = \begin{cases} 1 \text{ se o atributo } A_j \text{ é referenciado pela consulta } q_i \\ 0 \text{ caso contrário} \end{cases}$$

Considere as 4 consultas (aplicações) sobre a relação PROJ

q_1: **SELECT** BUDGET **FROM** PROJ **WHERE** PNO=Value

q_2: **SELECT** PNAME,BUDGET **FROM** PROJ

q_3: **SELECT** PNAME **FROM** PROJ **WHERE** LOC=Value

q_4: **SELECT** **SUM**(BUDGET) **FROM** PROJ **WHERE** LOC=Value

A_1= PNO, A_2= PNAME, A_3= BUDGET, A_4= LOC

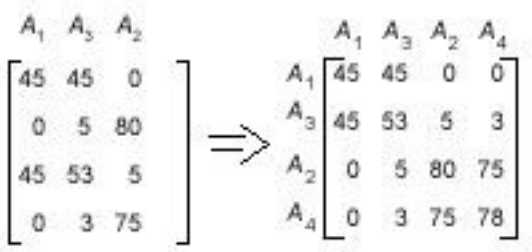
	A_1	A_2	A_3	A_4
q_1	1	0	1	0
q_2	0	1	1	0
q_3	0	1	0	1
q_4	0	0	1	1

- Algoritmo de agrupamento

Considere a matriz de afinidade de atributos AA e reorganize a ordem dos atributos para formar “grupos” onde os atributos em cada grupo têm alta afinidade entre si
“Bond Energy Algorithm” (BEA)

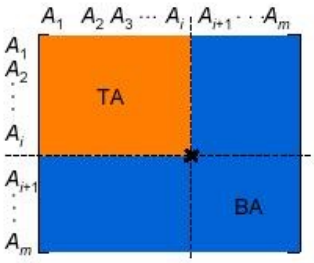
- Maximiza a função de afinidade global

AM = $\sum_i \sum_j$ (afinidade de A_i e A_j com seus vizinhos)



- Algoritmo

Como dividir um conjunto de atributos agrupados $\{A_1, A_2, ..., A_n\}$ em 2 (ou mais) conjuntos $\{A_1, A_2, ..., A_i\}$ e $\{A_i, ..., A_n\}$ tal que nenhuma (ou um número mínimo de) aplicação que acessam dois (ou mais do que um) conjuntos.



Sejam:
TQ = conjunto de aplicações que acessam apenas TA
BQ = conjunto de aplicações que acessam apenas BA
OQ = conjunto de aplicações que acessam tanto TA quanto BA
e
CTQ = num. total de acessos a atributos por aplicações que acessam apenas TA
CBQ = num. total de acessos a atributos por aplicações que acessam apenas BA
COQ = num. total de acessos a atributos por aplicações que acessam tanto TA quanto BA

Encontre o ponto na diagonal que maximize

$$CTQ * CBQ - COQ^2$$

Dois problemas:

Grupo formado no meio da matriz agrupada	Mais de 2 grupos
- Desloque uma linha para cima e uma coluna para a esquerda e aplique o algoritmo para encontrar o “melhor” ponto de visão - Repita para todos os possíveis deslocamentos - Custo = $O(m^2)$	- Divisão m-caminhos - Tente 1,2, …, m-1 pontos de divisão sobre a diagonal e tente encontrar o melhor ponto para cada um deles - Custo = $O(2^m)$

- Correção

Uma relação R, definida sobre o conjunto de atributos A e chave K, fragmentada verticalmente FR = {R1, R2, ..., Rr }

Completude	Reconstrução	Disjunção
$A = \cup A_{Ri}$	$R = \bowtie_K R_i$, $R_i \in FR$	- IDs, quando presentes, não são considerados sobrepostos pois são mantidos pelo sistema (transparente ao utilizador) - Chaves duplicadas não são consideradas sobrepostas

Fragmentação vertical da relação PROJ

- PROJ1: informações sobre orçamentos do projeto
- PROJ2: informações sobre o nome do projeto e localização

• Definimos nova coluna, a partir da regra imposta

PROJ

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Develop.	135000	New York
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris
P5	CAD/CAM	500000	Boston

$PROJ_1$

PNO	BUDGET
P1	150000
P2	135000
P3	250000
P4	310000
P5	500000

$PROJ_2$

PNO	PNAME	LOC
P1	Instrumentation	Montreal
P2	Database Develop.	New York
P3	CAD/CAM	New York
P4	Maintenance	Paris
P5	CAD/CAM	Boston

Correção da Fragmentação

Fragmentação da relação R em R_1, R_2, …, R_n

• Completude

- Decomposição de uma relação R em fragmentos R_1, R_2, ..., R_n é completa sse cada item dos dados em R pode ser encontrado em pelo menos um fragmento R_i

• Reconstrução

- Se uma relação R é decomposta em fragmentos R_1, R_2, ..., R_n , deve ser possível definir um operador relacional (ou de objetos) ∇ tal que:

$$R = \nabla_{1 \le i \le n} R_i$$

• Disjunção

- Se uma relação R é decomposta em fragmentos R_1, R_2, ..., R_n , e o item de dados d_i está em R_j , então d_i não está em qualquer outro fragmento R_k $(k \neq j)$.
- Não existem dados repetidos itens no R_n (exceto: PK na Fragmentação Vertical)
- Itens de dados são: Fragmentação Horizontal (tuplo), Fragmentação Vertical (atributo)

Alocação

Descrição do problema	Otimidalidade
F={F1,F2,…,Fn} fragmentos S={S1,S2,…,Sm} nós da rede Q={q1,q2,…,qq} aplicações *Encontre a distribuição "ótima" de F em S*	**Custo mínimo** - Comunicação + armazenamento + processamento (leitura e escrita) - Custo em termos de tempo (geralmente) **Desempenho** - Tempo de resposta e/ou vazão do sistema **Restrições** - Por nó (armazenamento e processamento)

Requisitos de Informação

Informação da BD	Informações de app	Informações dos nós	Informações da rede
- Seletividade dos fragmentos - Tamanho dos fragmentos	- Tipos e números dos acessos - Localização dos acessos	- Custo unitário de armazenamento de um dado num nó -Custo unitário de armazenamento num nó	- Largura de banda - Latência - Overhead de comunicação

Alocação de arquivos (PAA) vs. Alocação de Base de Dados (PABD)

Fragmentos não são arquivos individuais

-Relacionamentos têm que ser mantidos

Acesso às BD é mais complicado

-Modelo de acesso a arquivos remotos não é aplicável

-Relacionamento entre alocação e processamento de consultas

Custos adicionais que devem ser considerados

-Manutenção da integridade

-Controlo de concorrência

Modelo de alocação

Forma Geral: Min(custoTotal)

Sujeito às restrições

De tempo de resposta

De armazenamento

De processamento

Variável de decisão

X_{ij} = 1 se fragmento F_i é armazenado no nó S_j

0, caso contrário

Modelo de alocação

Métodos de solução	Heurísticas baseadas em	Tentativas de reduzir o espaço de soluções
PAA é NP - completo PABD também NP - completo	- Problema da mochila - Técnica de otimização branch and bound	Ignorar replicação

Algoritmo de Huan e Chen:

2 passos:

-Aloca réplicas em cada nó com consulta de leitura

-Retira réplicas para minimizar custo consultas de atualização