

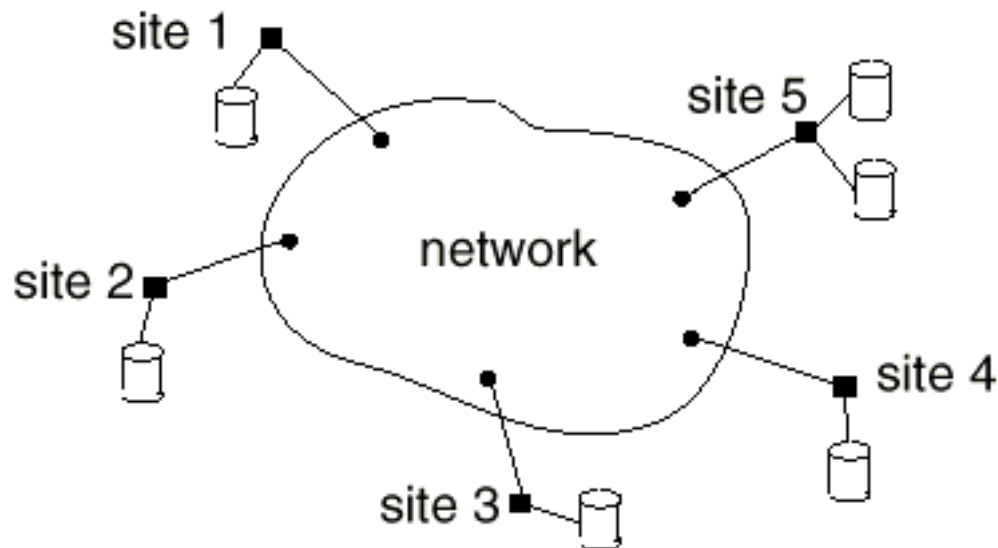


Banco de Datos II

Banco de Datos Distribuído

Banco de Dados Distribuído

SGBD manipula dados armazenados em diversos computadores (geralmente geograficamente distribuídos) através de vários meios de comunicação (geralmente redes).

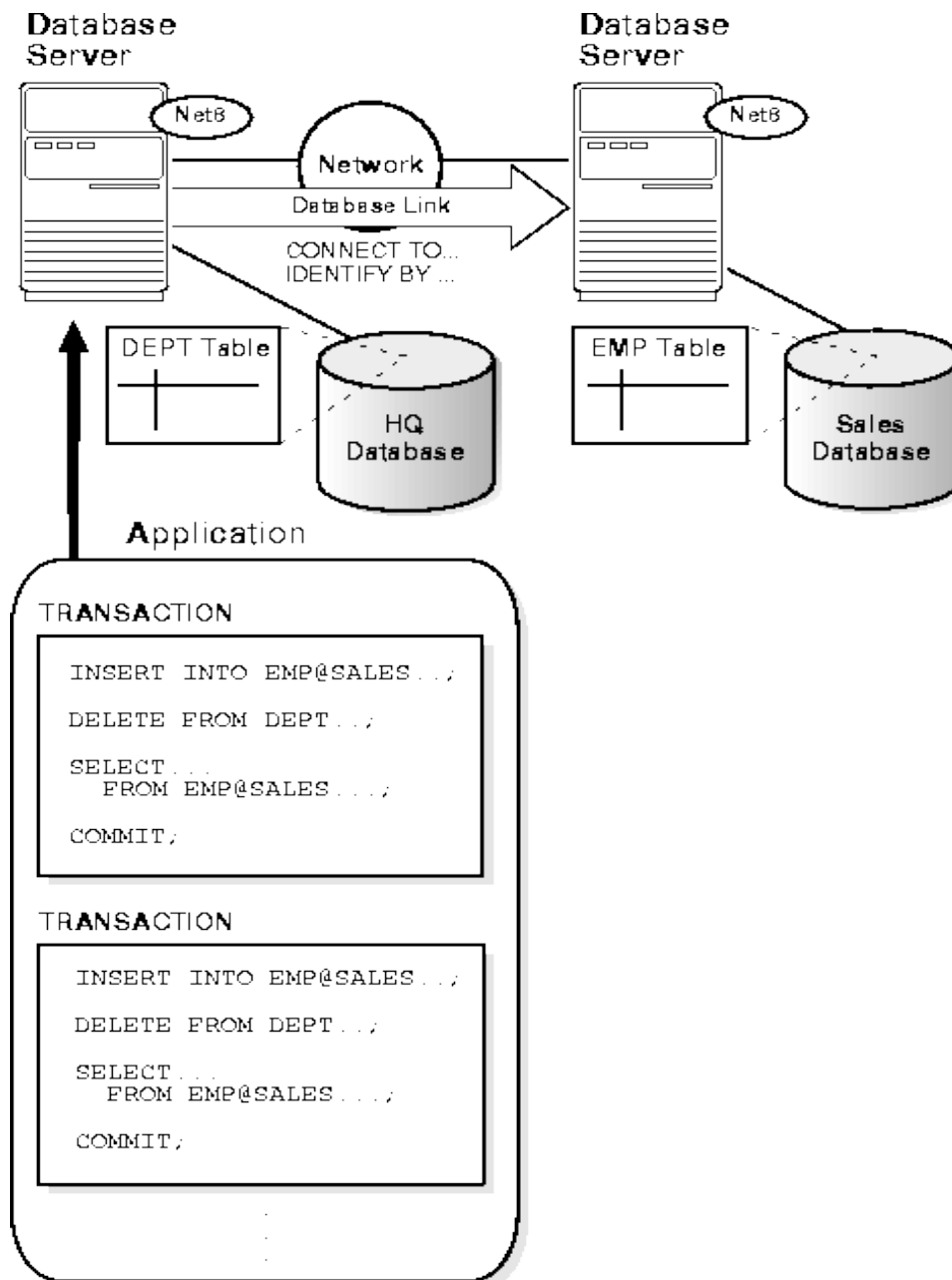


Banco de Dados Distribuído

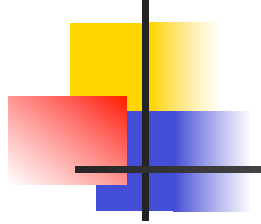


Transações Locais: **acessa dados somente no site onde foi submetida.**

Transações Globais: **acessa dados em diversos sites.**

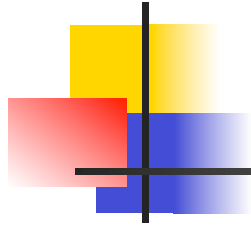


Banco de Dados Distribuído



Por que distribuir dados??

Banco de Dados Distribuído



Por que distribuir dados??

Compartilhamento:

Usuários de um site podem acessar dados de outros sites.

Maior disponibilidade:

Falhas em um site não torna o BD totalmente indisponível.

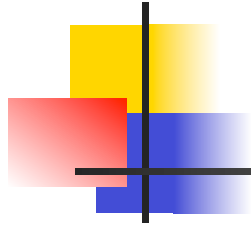
Autonomia:

Operação e controle local dos dados facilita gerenciamento e acesso aos dados;

Aumenta poder de processamento:

Vários computadores operando sobre a mesma base de dados

Banco de Dados Distribuído



Por que distribuir dados??

Facilidade de Reconfiguração e Extensão:

Novos BDs e processadores podem ser adicionados sem alterar os sistemas existentes;

Custos Baixos:

Computadores menores (ainda que em maior quantidade) são extremamente mais baratos do que mainframes.

Diminui Overhead de Comunicação:

Maioria dos dados são locais, portanto mais fáceis de serem acessados (melhor performance)

Banco de Dados Distribuído



Por que não distribuir dados??

Complexidade:

O gerenciamento global do sistema fica extremamente mais complexo. Cada site requer um DBA local. Desenvolvimento de software é mais caro.

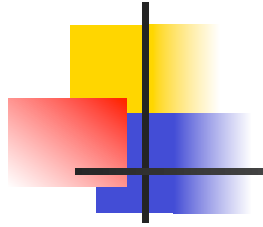
Maior possibilidade de bugs:

É mais difícil assegurar a precisão dos algoritmos (operam em paralelo).

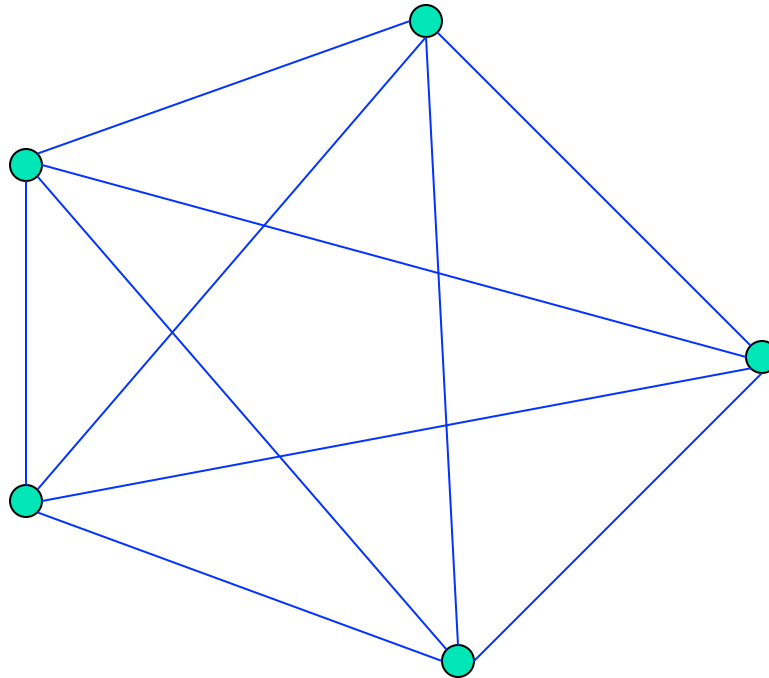
Aumento de Processamento e Overhead:

Decorrentes da necessidade de troca de mensagens.

Banco de Dados Distribuído

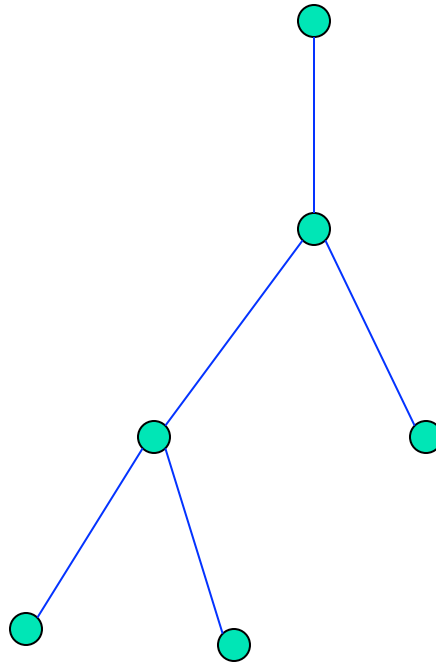


Arquitetura de Rede: Completamente conectada



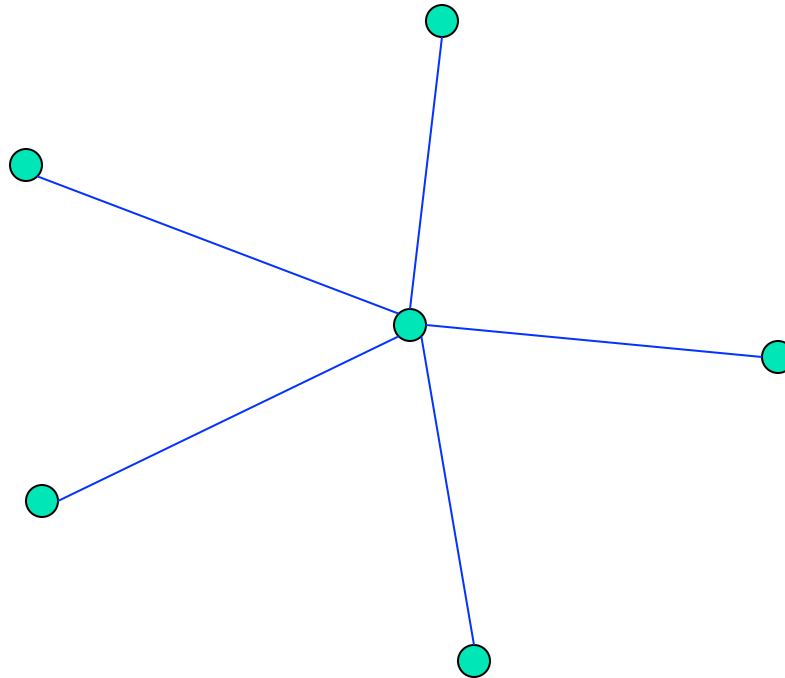
Banco de Dados Distribuído

Arquitetura de Rede: Árvore

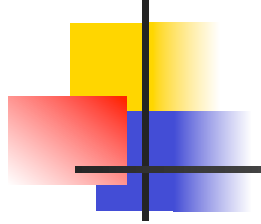


Banco de Dados Distribuído

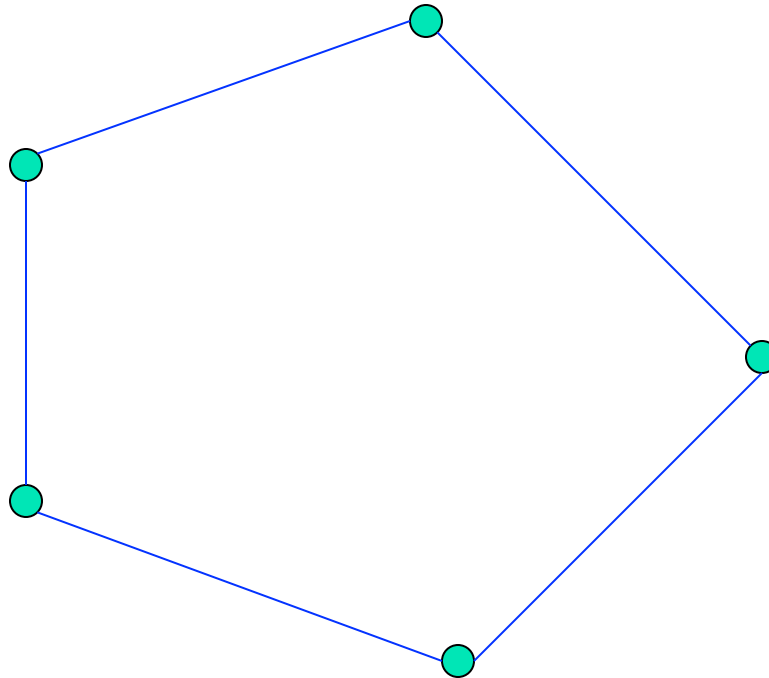
Arquitetura de Rede: Estrela



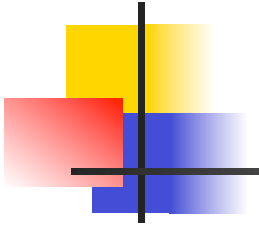
Banco de Dados Distribuído



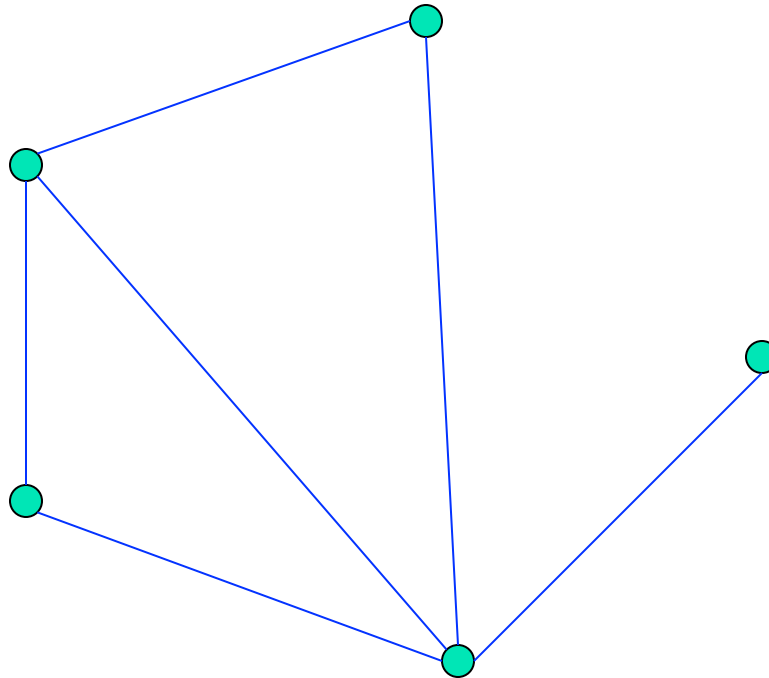
Arquitetura de Rede: Anel



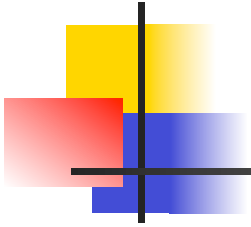
Banco de Dados Distribuído



Arquitetura de Rede: Parcialmente Conectada



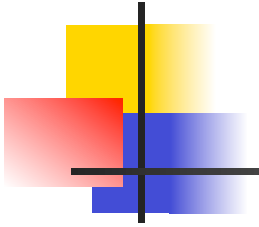
Banco de Dados Distribuído



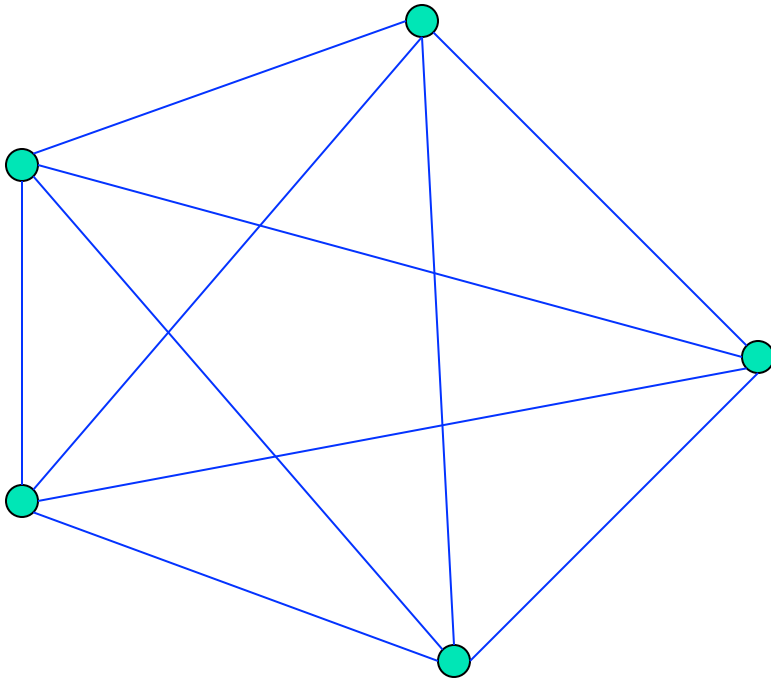
Arquitetura de Rede: CrITÉrios de Comparação

- **Custo de Instalação**
- **Custo de Comunicação**
- **Disponibilidade dos Dados**

Banco de Dados Distribuído



Arquitetura de Rede: Critérios de Comparação

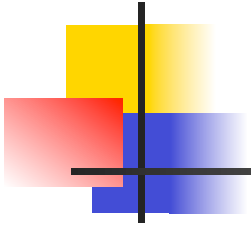


Custo de Instalação: Altíssimo

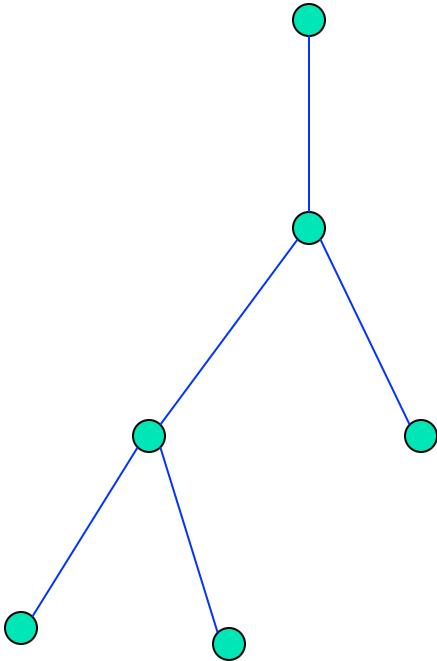
Custo de Comunicação: Baixíssimo

Disponibilidade dos Dados: Altíssima

Banco de Dados Distribuído



Arquitetura de Rede: CrITÉrios de Comparação

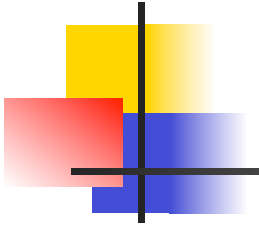


Custo de Instalação: Baixo

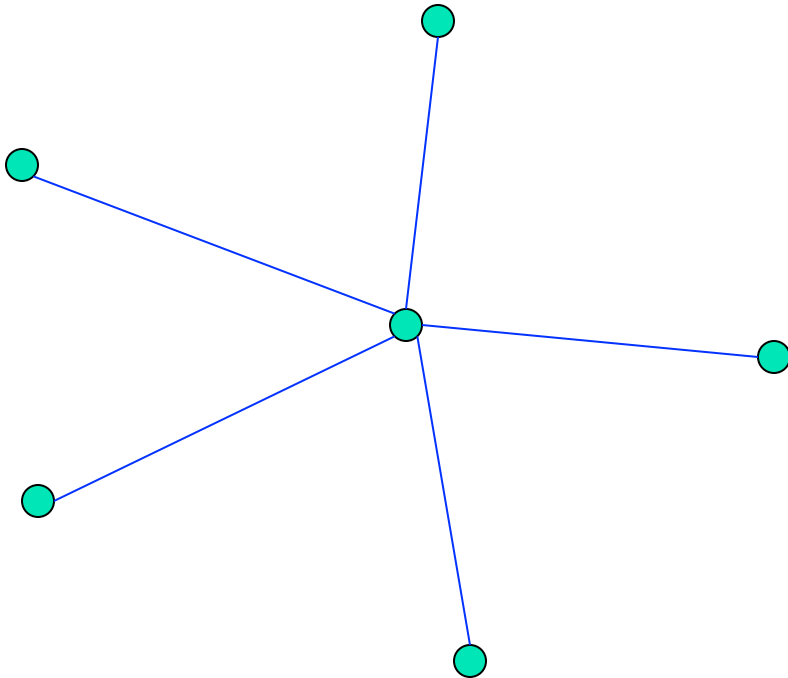
Custo de Comunicação: Alto

Disponibilidade dos Dados: Baixa, se um nó não-folha falhar.

Banco de Dados Distribuído



Arquitetura de Rede: CrITÉrios de Comparação

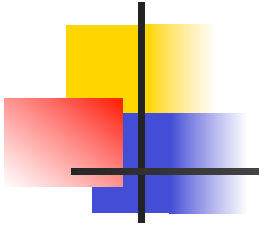


Custo de Instalação: Baixo

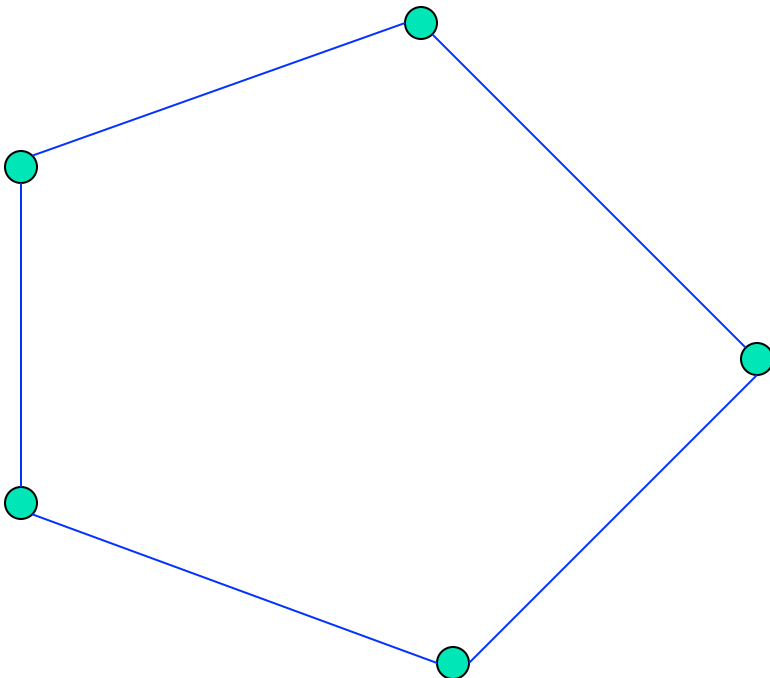
Custo de Comunicação: Baixo

Disponibilidade dos Dados: Baixíssima, se o nó central falhar

Banco de Dados Distribuído



Arquitetura de Rede: Critérios de Comparação

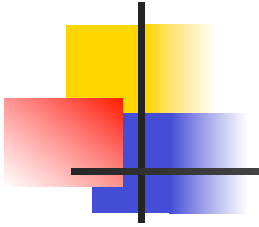


Custo de Instalação: Baixo

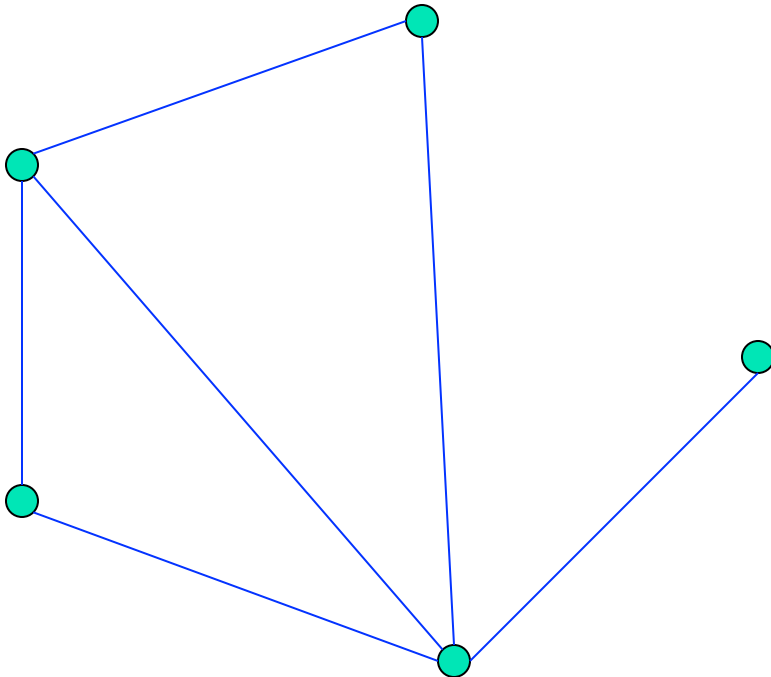
Custo de Comunicação: Médio

Disponibilidade dos Dados: Média

Banco de Dados Distribuído



Arquitetura de Rede: Critérios de Comparação



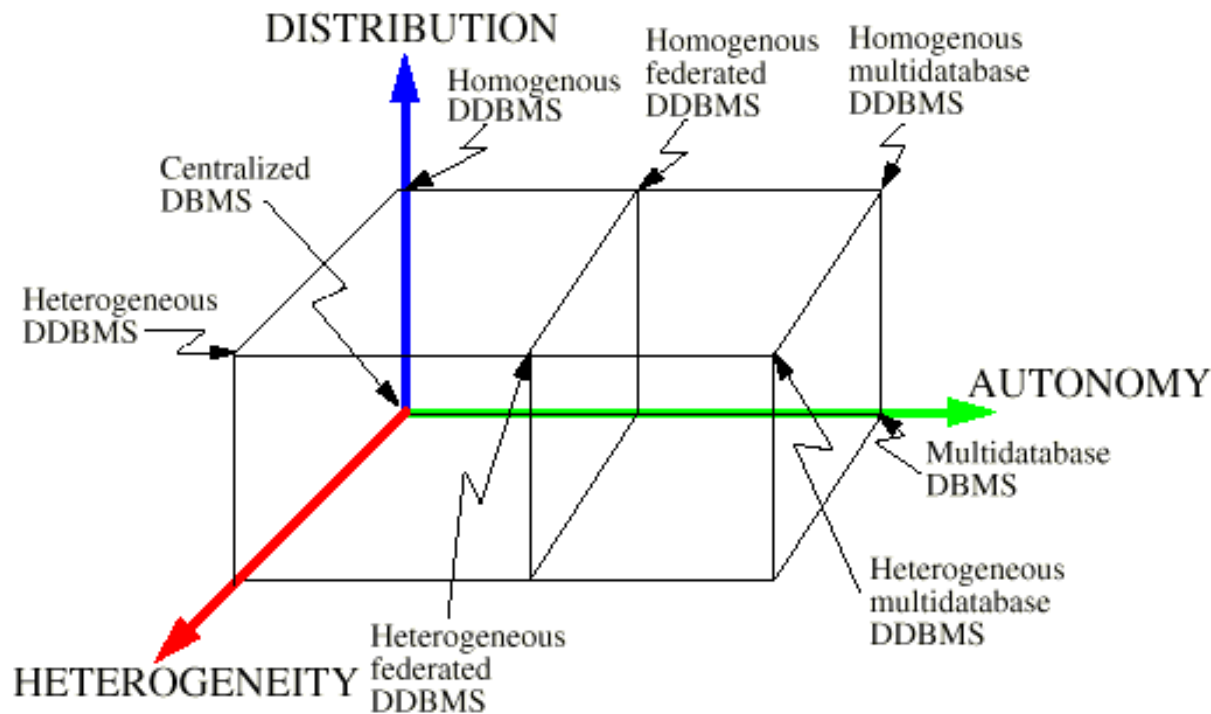
Custo de Instalação: Médio

Custo de Comunicação: Baixo

Disponibilidade dos Dados: Alta

Banco de Dados Distribuído

Arquitetura de SGBD Distribuído



Distribuição: grau de distribuição do Banco de Dados

Heterogeneidade: grau de diferença dos SGBDs locais

Autonomia: grau de controle dos SGBDs locais sobre as operações locais

Banco de Dados Distribuído



Federated Database System

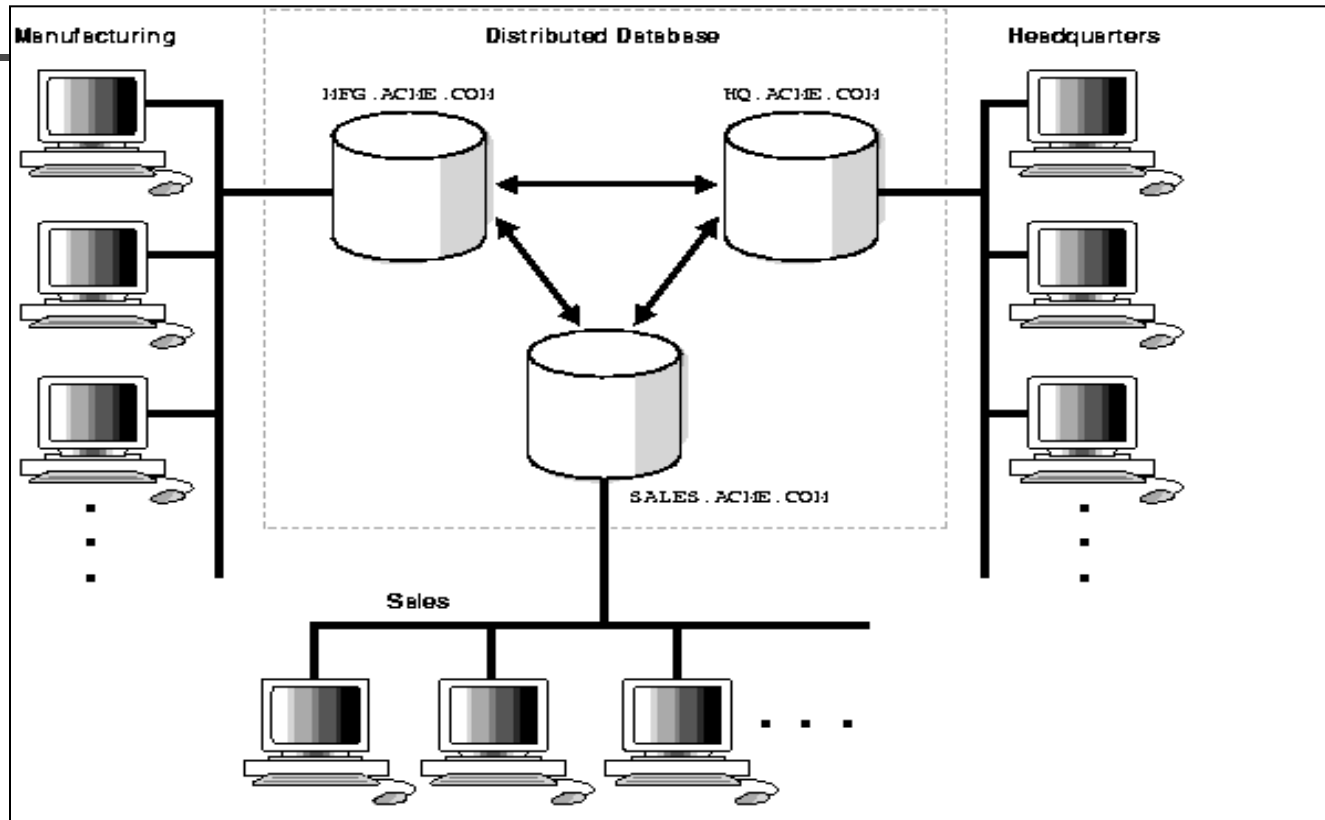
A **federated database system** is a type of meta-database management system (DBMS) which transparently integrates multiple autonomous database systems into a single **federated database**.



Homogenous Distributed Database Systems

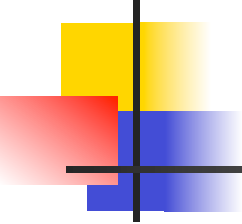
- ❑ All sites have identical software
- ❑ They are aware of each other and agree to cooperate in processing user requests
- ❑ It appears to user as a single system

An Homogenous Distributed Database Systems example



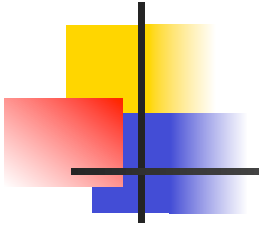
- A distributed system connects three databases: hq, mfg, and sales
- An application can simultaneously access or modify the data in several databases in a single distributed environment.

Heterogeneous Distributed Database System

- 
- In a heterogeneous distributed database system, at least one of the databases uses different schemas and software.
 - A database system having different schema may cause a major problem for query processing.
 - A database system having different software may cause a major problem for transaction processing.

The schema of a database system is its structure described in a formal language supported by the database management system (DBMS). In a relational database, the schema defines the tables, the fields, relationships, views, indexes, packages, procedures, functions, queues, triggers, types, sequences, materialized views, synonyms, database links, directories, Java, XML schemas, and other elements.

Banco de Dados Distribuído



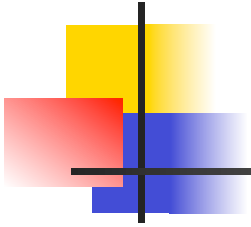
Armazenamento Distribuído dos Dados

Replicação

Fragmentação

Replicação e Fragmentação

Banco de Dados Distribuído



1 - Replicação

“Armazenar uma relação em dois ou mais sites”

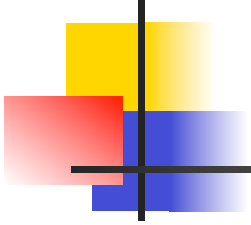
Vantagens:

- **disponibilidade dos dados**
- **aumento do paralelismo e velocidade**

Desvantagem:

- **aumento de overhead na atualização**

Banco de Dados Distribuído



2 - Fragmentação

“Tabelas são fragmentadas horizontal ou verticalmente. Os fragmentos são armazenados em sites distintos”

Banco de Dados Distribuído

2.a - Fragmentação Horizontal

data fragmentation—horizontal $\rightarrow r_i = \sigma_{P_i}(r)$

Staff

employee#	name	address	hkid	duty	shift	salary	ward#
1009	Holmes D.	86 Queen	A450361	Nurse	M	45,000	6
3754	Chan B.	21 Minto	C461378	Orderly	A	30,000	2
8422	Hui J.	16 Peak	F562916	Intern	M	55,000	1
9901	Bell G.	53 Water	A417394	Nurse	M	48,000	2
3106	Wong R.	58 Aster	C538294	Nurse	E	51,000	6
6357	Kwok W.	80 Dinn	K721893	Intern	E	56,000	1
7379	Chui J.	25 Peak	J381924	Orderly	A	33,000	1
1280	Poon R.	16 Cliff	N328401	Intern	A	60,000	2

Banco de Dados Distribuído

2.a - Fragmentação Horizontal

$$\text{Staff}_1 = \sigma_{\text{shift}='M'}(\text{Staff})$$

employee#	name	address	hkid	duty	shift	salary	ward#
1009	Holmes D.	86 Queen	A450361	Nurse	M	45,000	6
8422	Hui J.	16 Peak	F562916	Intern	M	55,000	1
9901	Bell G.	53 Water	A417394	Nurse	M	48,000	2

$$\text{Staff}_2 = \sigma_{\text{shift}='A'}(\text{Staff})$$

employee#	name	address	hkid	duty	shift	salary	ward#
3754	Chan B.	21 Minto	C461378	Orderly	A	30,000	2
7379	Chui J.	25 Peak	J381924	Orderly	A	33,000	1
1280	Poon R.	16 Cliff	N328401	Intern	A	60,000	2

$$\text{Staff}_3 = \sigma_{\text{shift}='E'}(\text{Staff})$$

employee#	name	address	hkid	duty	shift	salary	ward#
3106	Wong R.	58 Aster	C538294	Nurse	E	51,000	6
6357	Kwok W.	80 Dinn	K721893	Intern	E	56,000	1

$$r = r_1 \cup r_2 \cup \dots \cup r_n$$

Banco de Dados Distribuído

2.b - Fragmentação Vertical

data fragmentation—vertical $\rightarrow r_i = \pi_{R_i}(r)$
Staff

employee#	name	address	hkid	duty	shift	salary	ward#
1009	Holmes D.	86 Queen	A450361	Nurse	M	45,000	6
3754	Chan B.	21 Minto	C461378	Orderly	A	30,000	2
8422	Hui J.	16 Peak	F562916	Intern	M	55,000	1
9901	Bell G.	53 Water	A417394	Nurse	M	48,000	2
3106	Wong R.	58 Aster	C538294	Nurse	E	51,000	6
6357	Kwok W.	80 Dinn	K721893	Intern	E	56,000	1
7379	Chui J.	25 Peak	J381924	Orderly	A	33,000	1
1280	Poon R.	16 Cliff	N328401	Intern	A	60,000	2

Banco de Dados Distribuído

2.b - Fragmentação Vertical

$Staff_4 = \pi_{employee\#, name, address, hkid, salary}(Staff)$

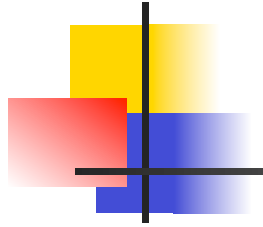
employee#	name	address	hkid	salary
1009	Holmes D.	86 Queen	A450361	45,000
3754	Chan B.	21 Minto	C461378	30,000
8422	Hui J.	16 Peak	F562916	55,000
9901	Bell G.	53 Water	A417394	48,000
3106	Wong R.	58 Aster	C538294	51,000
6357	Kwok W.	80 Dinn	K721893	56,000
7379	Chui J.	25 Peak	J381924	33,000
1280	Poon R.	16 Cliff	N328401	60,000

$Staff_5 = \pi_{employee\#, name, duty, shift, ward\#}(Staff)$

employee#	name	duty	shift	ward#
1009	Holmes D.	Nurse	M	6
3754	Chan B.	Orderly	A	2
8422	Hui J.	Intern	M	1
9901	Bell G.	Nurse	M	2
3106	Wong R.	Nurse	E	6
6357	Kwok W.	Intern	E	1
7379	Chui J.	Orderly	A	1
1280	Poon R.	Intern	A	2

$$r = r_1 \bowtie r_2 \bowtie \dots \bowtie r_n$$

Banco de Dados Distribuído



2.c - Fragmentação Mista

Aplica-se fragmentações horizontais e verticais simultaneamente.

Banco de Dados Distribuído



3 - Replicação e Fragmentação

Fragmenta-se as réplicas.

Replica-se os fragmentos.



DISTRIBUTED DATABASES

WHY FRAGMENT DATA?

⇒ Usage

Applications are usually interested in 'views' not whole relations.

⇒ Efficiency

It's more efficient if data is close to where it is frequently used.

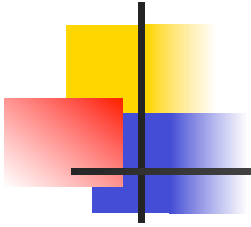
⇒ Parallelism

It is possible to run several 'sub-queries' in tandem.

⇒ Security

Data not required by local applications is not stored at the local site.

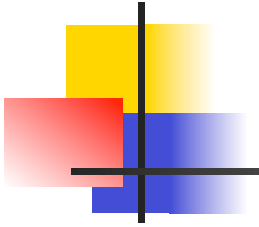
Banco de Dados Distribuído



Transparência de Rede

“Grau de desconhecimento do usuário sobre detalhes relativos à distribuição dos dados na rede.”

Banco de Dados Distribuído

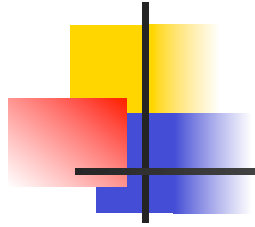


Transparência de Rede

Objetivo

“Maximizar a transparência de tal modo que os usuários percebam o BD distribuído como sendo centralizado.”

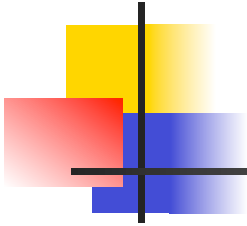
Banco de Dados Distribuído



Transparência de Rede: Denominação dos Itens de Dados
(relações, réplicas, fragmentos, etc)

- **servidor de nomes**
- **utilizar identificação dos sites como prefixos dos nomes**

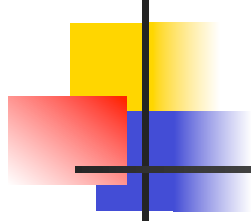
Banco de Dados Distribuído



Servidor de Nomes

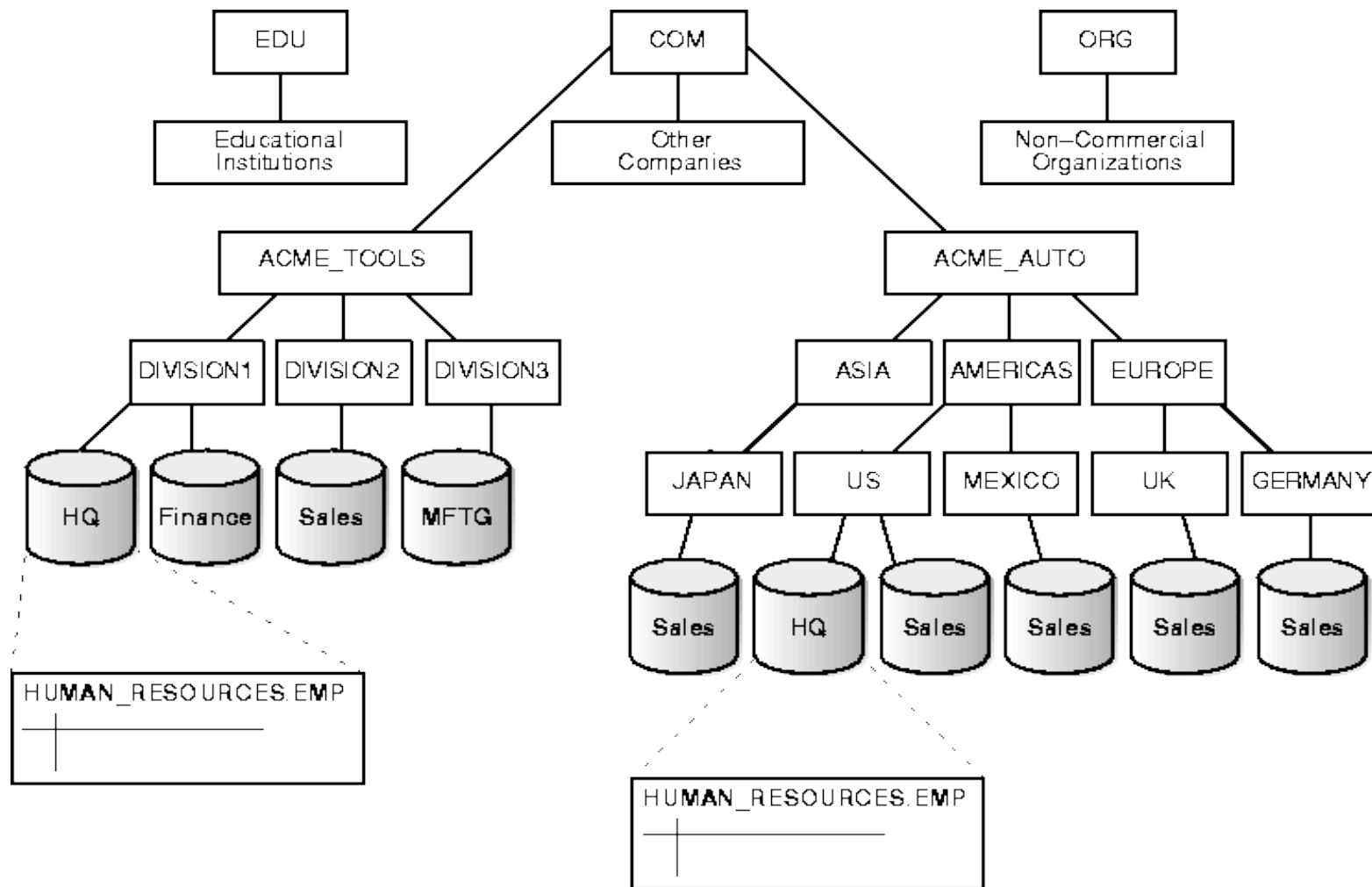
- **Todos os nomes devem ser registrados no servidor de nomes para garantir a não duplicidade de nomes para itens de dados diferentes.**
- **Os itens de dados são localizados pelo nome**
- **Gargalo do sistema → redução do desempenho**
- **Ponto de falha → se o servidor falhar os demais sites não poderão funcionar adequadamente**

Banco de Dados Distribuído



Nomes com Prefixos

- **Todos os nomes gerados por um site local são identificados globalmente por um identificador composto por este nome local (que é único localmente) e por um prefixo que identifica o site.**

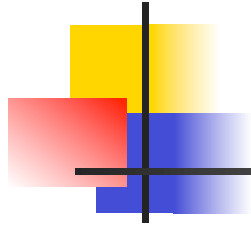


The network domains **US.AMERICAS.ACME_AUTO.COM** and **UK.EUROPE.ACME_AUTO.COM** each contain a **SALES** database:

SALES.US.AMERICAS.ACME_AUTO.COM

SALES.UK.EUROPE.ACME_AUTO.COM

Banco de Dados Distribuído



Nomes com Prefixos

- **Esta estratégia diminui a transparência.**

Solução: utilização de aliases que são armazenados no catálogo do sistema. Este catálogo é distribuído para todos os sites e é utilizado pelo sistema para mapear os aliases aos nomes dos itens de dados correspondente.

Uso de sinônimos para estabelecer transparência de localização das tabelas

Os comandos abaixo criam sinônimos em uma base de dados para tabelas armazenadas em outra base de dados remota:

```
CREATE PUBLIC SYNONYM emp FOR scott.emp@sales.us.americas.acme_auto.com  
CREATE PUBLIC SYNONYM dept FOR scott.dept@sales.us.americas.acme_auto.com
```

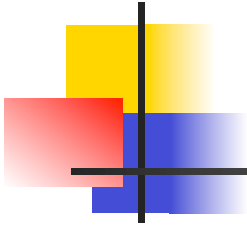
Com isso, ao invés de acessar as tabelas remotas como exemplificado abaixo:

```
SELECT ename, dname  
FROM scott.emp@sales.us.americas.acme_auto.com e,  
      scott.dept@sales.us.americas.acme_auto.com d  
WHERE e.deptno = d.deptno;
```

Uma aplicação pode executar uma consulta de forma mais simples, sem se preocupar com a localização da tabela (ou seja, de forma mais transparente):

```
SELECT ename, dname  
FROM emp e, dept d WHERE e.deptno = d.deptno;
```

Banco de Dados Distribuído



Processamento de Consultas Distribuídas

- uma consulta pode requerer dados de sites distintos;**
- fatores a serem considerados**
 - custo de transmissão de dados pela rede**
 - potencial para processamento paralelo**

Banco de Dados Distribuído

Processamento de Consultas Distribuídas

$$\text{Staff}_1 = \sigma_{\text{shift}='M'}(\text{Staff})$$

employee#	name	address	hkid	duty	shift	salary	ward#
1009	Holmes D.	86 Queen	A450361	Nurse	M	45,000	6
8422	Hui J.	16 Peak	F562916	Intern	M	55,000	1
9901	Bell G.	53 Water	A417394	Nurse	M	48,000	2

$$\text{Staff}_2 = \sigma_{\text{shift}='A'}(\text{Staff})$$

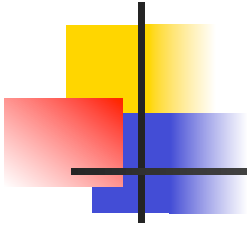
employee#	name	address	hkid	duty	shift	salary	ward#
3754	Chan B.	21 Minto	C461378	Orderly	A	30,000	2
7379	Chui J.	25 Peak	J381924	Orderly	A	33,000	1
1280	Poon R.	16 Cliff	N328401	Intern	A	60,000	2

$$\text{Staff}_3 = \sigma_{\text{shift}='E'}(\text{Staff})$$

employee#	name	address	hkid	duty	shift	salary	ward#
3106	Wong R.	58 Aster	C538294	Nurse	E	51,000	6
6357	Kwok W.	80 Dinn	K721893	Intern	E	56,000	1

$$\text{Staff} = \text{Staff}_1 \cup \text{Staff}_2 \cup \text{Staff}_3$$

Banco de Dados Distribuído



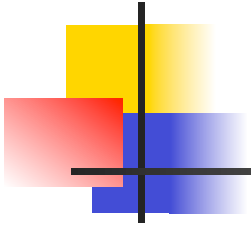
Processamento de Consultas Distribuídas

consider the query: $\sigma_{\text{shift}='E'}(\text{Staff})$

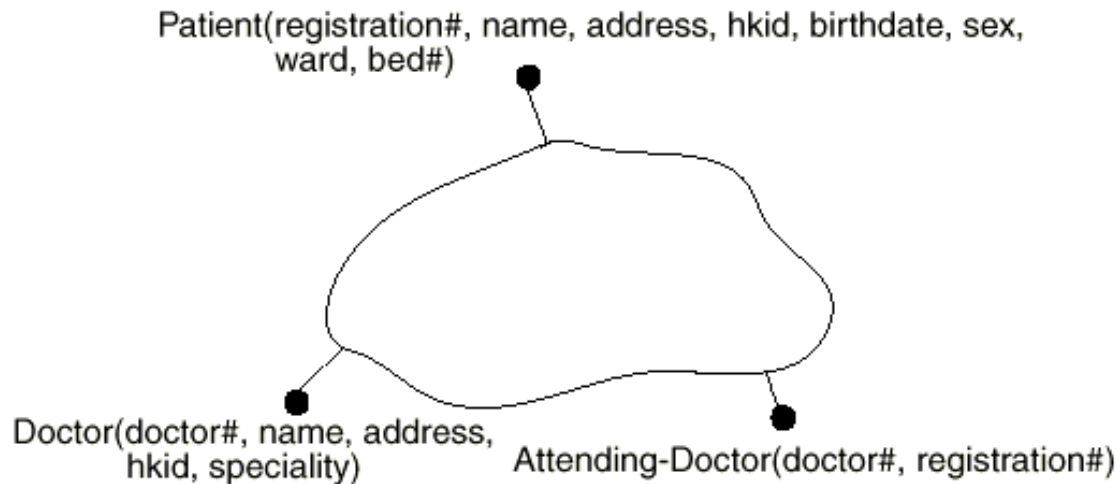
$$\Rightarrow \sigma_{\text{shift}='E'}(\text{Staff}_1 \cup \text{Staff}_2 \cup \text{Staff}_3)$$

$$\Rightarrow \sigma_{\text{shift}='E'}(\text{Staff}_1) \cup \sigma_{\text{shift}='E'}(\text{Staff}_2) \cup \sigma_{\text{shift}='E'}(\text{Staff}_3)$$

Banco de Dados Distribuído

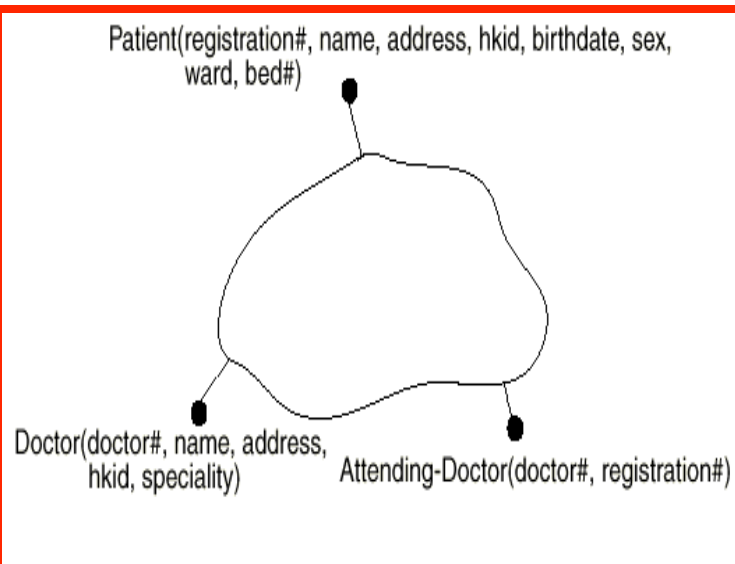


Processamento de Consultas Distribuídas



Banco de Dados Distribuído

Processamento de Consultas Distribuídas



consider the join:

Patient \bowtie Doctor \bowtie Attending-Doctor

Possible strategies (want results at S_I)

1. ship all three relations to S_I
process entire query at S_I
2. ship a copy of Attending-Doctor to Patient and compute
 $TEMP_1 = \text{Attending-Doctor} \bowtie \text{Patient}$
ship $TEMP_1$ to Doctor and compute
 $TEMP_2 = TEMP_1 \bowtie \text{Doctor}$
ship $TEMP_2$ to S_I
3. variants of 2

Banco de Dados Distribuído

Processamento de Consultas Distribuídas

suppose we want to join R and S at site S

semijoin strategy


1. compute $\text{temp}_1 = \pi_{R \cap S}(s)$ at site S
2. ship temp_1 from site S to site R
3. compute $\text{temp}_2 = r \bowtie \text{temp}_1$ at site R
4. ship temp_2 from site R to site S
5. compute $S \bowtie \text{temp}_2$ at site S

result is the same as $r \bowtie s$

Banco de Dados Distribuído

Processamento de Consultas Distribuídas

semijoin: $r \bowtie s = \pi_R(r \bowtie s)$

 selects only those tuples of r that contribute to $r \bowtie s$

Banco de Dados Distribuído



Comandos SQL Remotos e Distribuídos (Oracle)

A **remote query** is a query that selects information from one or more remote tables, all of which reside at the **same remote node**.

For example:

```
CREATE DATABASE LINK sales.us.americas.acme_auto.com .... ;  
SELECT * FROM scott.dept@sales.us.americas.acme_auto.com;
```

Banco de Dados Distribuído



Comandos SQL Remotos e Distribuídos (Oracle)

A *remote update* is an update that modifies data in one or more tables, all of which are located at the *same remote node*.

For example:

```
UPDATE scott.dept@sales.us.americas.acme_auto.com  
SET loc = 'NEW YORK'  
WHERE deptno = 10;
```

Banco de Dados Distribuído



Comandos SQL Remotos e Distribuídos (Oracle)

A *distributed query* retrieves information from **two or more nodes**.

For example:

```
SELECT ename, dname  
FROM scott.emp e, scott.dept@sales.us.americas.acme_auto.com d  
WHERE e.deptno = d.deptno;
```

Banco de Dados Distribuído



Comandos SQL Remotos e Distribuídos (Oracle)

A ***distributed update*** modifies data on two or more nodes. A distributed update is possible using a PL/SQL subprogram unit, such as a procedure or trigger, that includes two or more remote updates that **access data on different nodes**.

For example:

```
BEGIN
UPDATE scott.dept@sales.us.americas.acme_auto.com
SET loc = 'NEW YORK' WHERE deptno = 10;
UPDATE scott.emp
SET deptno = 11 WHERE deptno = 10;
END;
```

Banco de Dados Distribuído



Transações Remotas e Distribuídas (Oracle)

A **remote transaction** contains one or more remote statements, all of which reference the **same remote node**.

For example:

```
UPDATE scott.dept@sales.us.americas.acme_auto.com  
SET loc = 'NEW YORK' WHERE deptno = 10;
```

```
UPDATE scott.emp@sales.us.americas.acme_auto.com  
SET deptno = 11 WHERE deptno = 10;
```

```
COMMIT;
```

Banco de Dados Distribuído



Transações Remotas e Distribuídas (Oracle)

A ***distributed transaction*** contains one or more statements that, individually or as a group, update data on **two or more distinct nodes** of a distributed database.

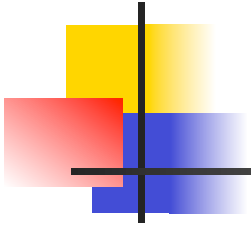
For example:

```
UPDATE scott.dept@sales.us.americas.acme_auto.com  
SET loc = 'NEW YORK' WHERE deptno = 10;
```

```
UPDATE scott.emp  
SET deptno = 11 WHERE deptno = 10;
```

```
COMMIT;
```

Banco de Dados Distribuído



Transações Distribuídas

O SGBD Distribuído precisa garantir as propriedades ACID das transações distribuídas, o que é mais complicado devido à possibilidade de ocorrências de outros tipos de falhas, além daquelas nos próprios sites.

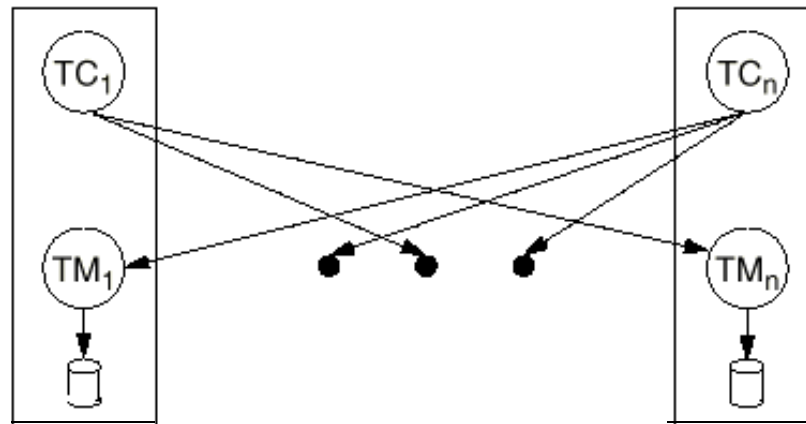
Possíveis Falhas

**Falha em um site
Perda de mensagem
Falha de comunicação
Particionamento da Rede**

Banco de Dados Distribuído

Transações Distribuídas

- at each site we have
 - transaction manager (TM) - manages execution of transactions that access local data only
 - transaction coordinator (TC) - coordinates the execution of the transactions initiated at a site (may span several sites)
 - manages a log for recovery
 - starts transaction
 - decomposes transaction into subtransactions
 - coordinates termination of a transaction: commit or abort



Banco de Dados Distribuído

Transações Distribuídas

Protocolo de Efetivação em Duas Fases (2PC)

```
UPDATE  
scott.dept@sales.us.americas.acme_a  
uto.com  
SET loc = 'NEW YORK' WHERE  
deptno = 10;
```

```
UPDATE scott.emp  
SET deptno = 11 WHERE deptno =  
10;
```

```
COMMIT; ← 2PC é executado neste instante.
```

Banco de Dados Distribuído

Transações Distribuídas

Protocolo de Efetivação em Duas Fases (2PC)

Phase 1 - prepare to commit

the coordinator:

- adds <prepare T> to its log and forces the log
- queries all sites to determine if they are willing to commit or not

each participating site:

- if no: add <abort T> to log and force log
send abort T message to coordinator
- if yes: add <ready T> to log and force log
send ready T message to coordinator

Banco de Dados Distribuído

Transações Distribuídas

Protocolo de Efetivação em Duas Fases (2PC)

Phase 2 - commit or abort decision

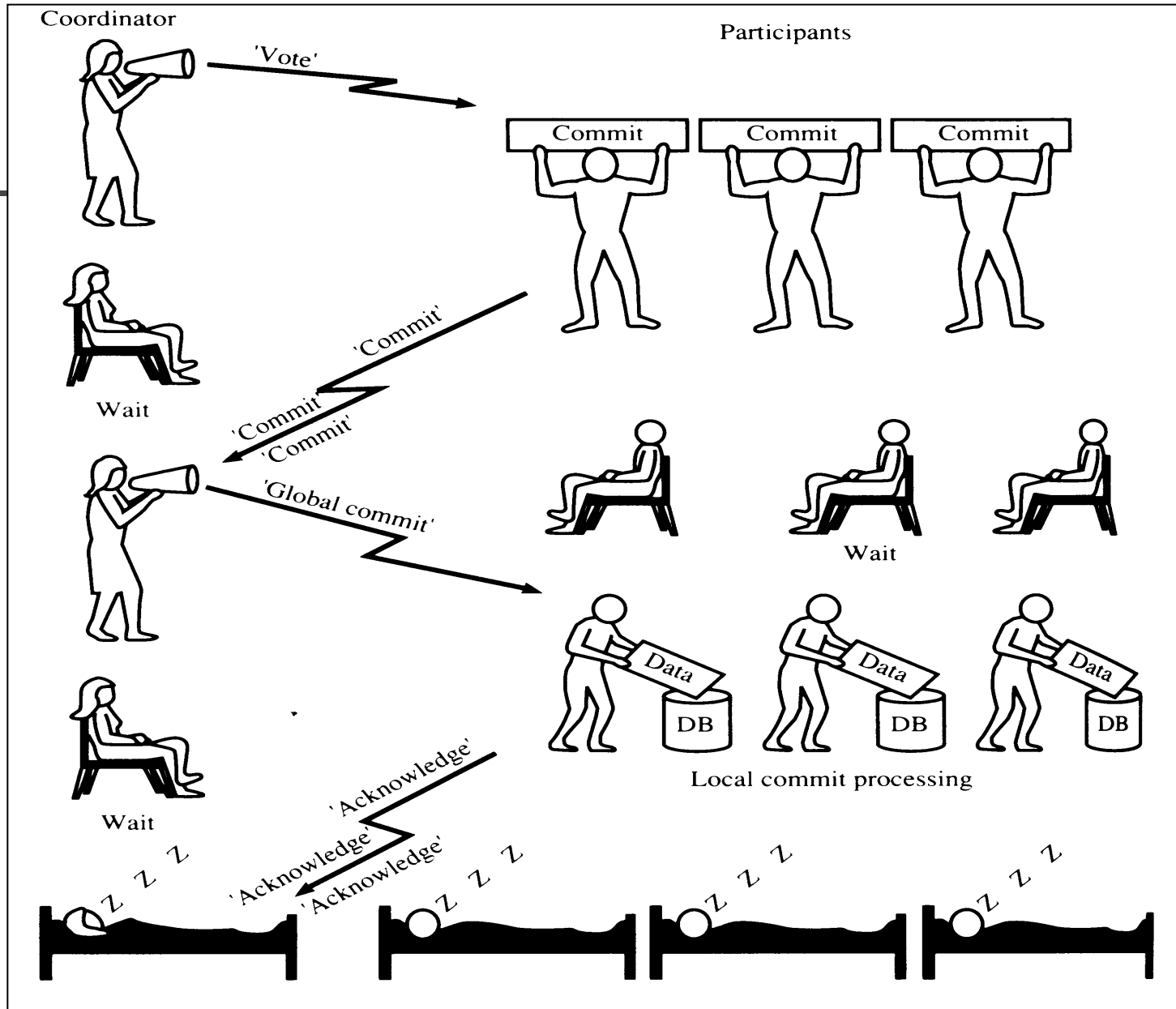
the coordinator:

- if an **abort** T message is received or time-out:
add <**abort** T> to log and force log
send **abort** T message to all participating sites
- if all participants reply ready, coordinator decides whether to commit or abort:
if commit: add <**commit** T> to log and force log
send **commit** T message to all participating sites
if abort: add <**abort** T> to log and force log
send **abort** T message to all participating sites

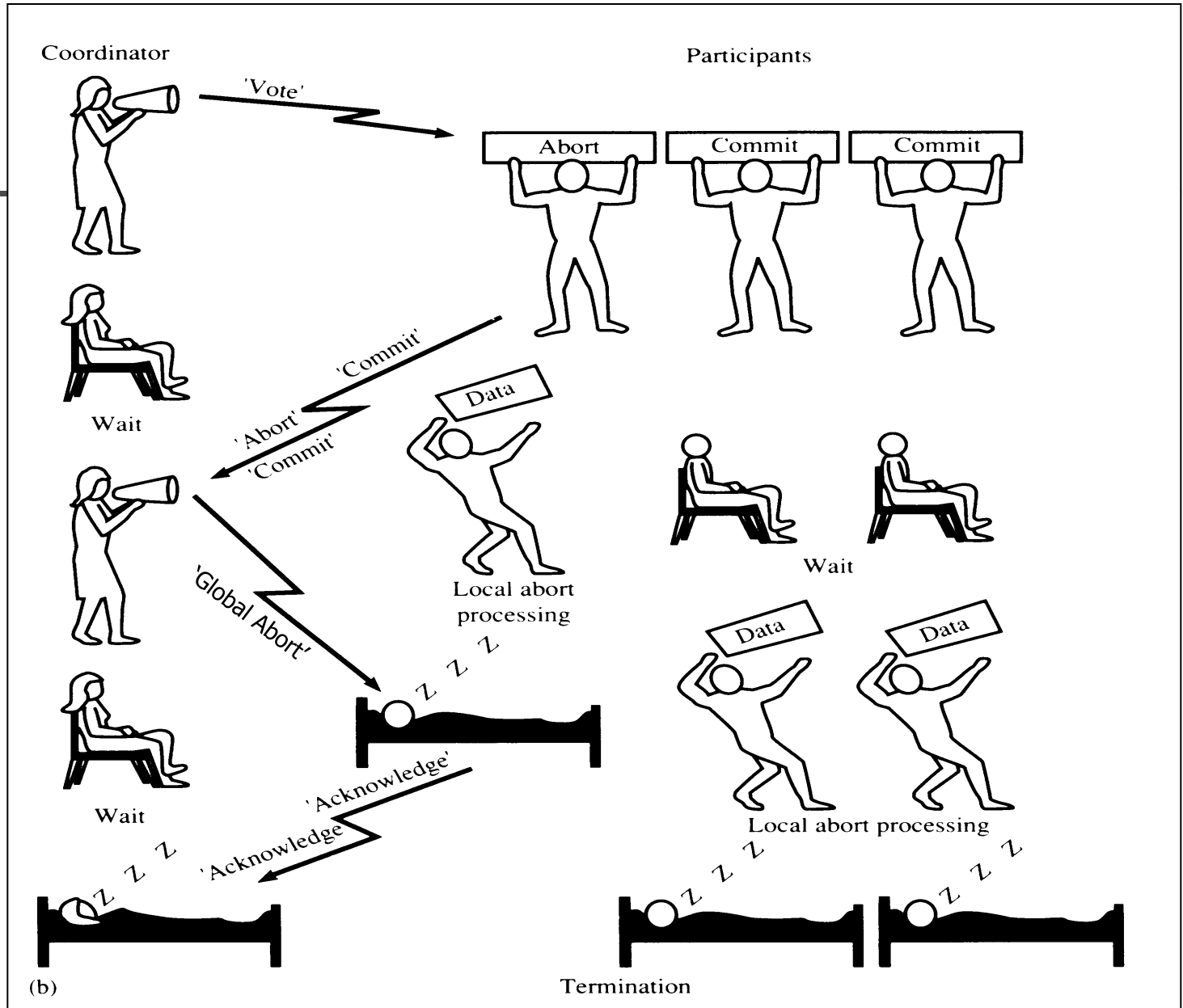
each participating site:

- records decision of coordinator in log and forces log

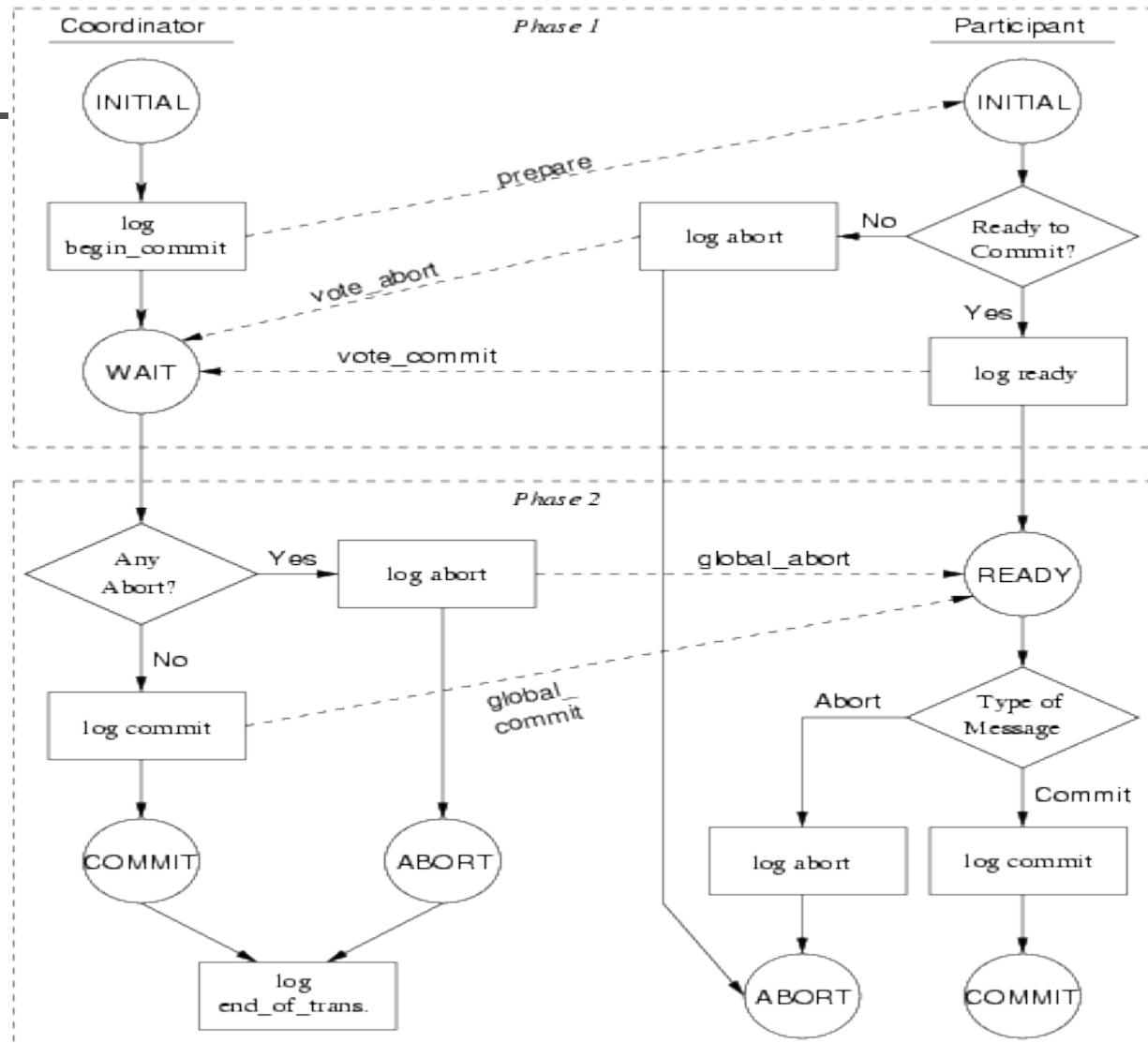
TWO-PHASE COMMIT (2PC) - OK



TWO-PHASE COMMIT (2PC) - ABORT



Two-Phase Commit Diagram



Banco de Dados Distribuído

Manuseio de Falhas no 2PC

site failure

- if before site sends **ready** T message, then assume **abort** T sent
else normal 2PC processing
- when a site recovers, if its log contains:
 - <**commit** T> → **redo**(T)
 - <**abort** T> → **undo**(T)
 - <**ready** T> → consult coordinator
 - if the coordinator is down, send **query-status** T message to
other sites to determine the fate of T
 - log contains no control record for T → **undo**(T)

Banco de Dados Distribuído

Manuseio de Falhas no 2PC

coordinator failure

- if an active site contains:
 - <commit T> → must commit T
 - <abort T> → must abort T
- if some active site does not contain <ready T> → abort T
- if none of the above, then all sites have a <ready T> record
→ cannot decide fate of T

☞ transactions may block since they hold locks which they cannot release until the coordinator comes up

Banco de Dados Distribuído



Manuseio de Falhas no 2PC

network partition

- if coordinator + all participants in same partition we can continue
- else execute 2PC taking into account failure actions

Banco de Dados Distribuído



Recuperação de Falhas no 2PC

A recuperação de um site após uma falha em SGBDs distribuídos pode utilizar a mesma estratégia de recuperação de SGBDs centralizados, entretanto é preciso tratar as transações em dúvida.

Banco de Dados Distribuído



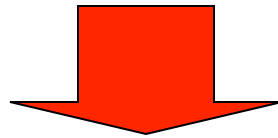
Recuperação de Falhas no 2PC

**transações em dúvida T → existe o registro <ready T>
no arquivo de log, mas não existe o registro <commit T>
nem o registro <abort T>.**

Banco de Dados Distribuído

Recuperação de Falhas no 2PC

transações em dúvida T → existe o registro <ready T>
no arquivo de log, mas não existe o registro <commit T>
nem o registro <abort T>.

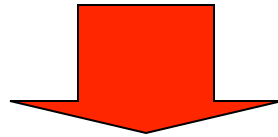


É preciso consultar outros sites para determinar o status da
transação em dúvida T.

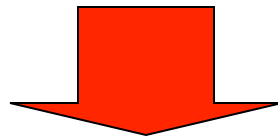
Banco de Dados Distribuído

Recuperação de Falhas no 2PC

É preciso consultar outros sites para determinar o status da transação em dúvida T.



Pode ser preciso esperar por muito tempo. Durante a espera não pode processar outras transações.



Solução: Utilizar **<ready T, L>**, onde L é uma lista de todos os itens de dados locais bloqueados pela transação T.

Banco de Dados Distribuído



Controle de Concorrência

Protocolos Baseados em Bloqueios

➤ **Único Gerenciador de Bloqueios (site Si)**

- **simplicidade de implementação**
- **facilidade de tratamento de impasses (deadlocks)**

- **Si é um gargalo do sistema**
- **Vulnerabilidade (caso Si falhe)**

Banco de Dados Distribuído



Controle de Concorrência

Protocolos Baseados em Bloqueios

- **Múltiplos Gerenciadores de Bloqueios** (gerenciadores residem em sites distintos e gerenciam as solicitações de bloqueios sobre um conjunto de dados)

- **complica tratamento de impasses (deadlocks)**

- **diminui o engarrafamento do sistema**
- **diminui a vulnerabilidade**

Banco de Dados Distribuído



Controle de Concorrência

Protocolos Baseados em Bloqueios

- **Protocolo da maioria** (cada site mantém um gerenciador de bloqueio local para os itens de dados armazenados naquele site)

Quando há réplicas, um bloqueio solicitado é atendido somente quando a maioria dos gerenciadores das réplicas concedê-lo.

Banco de Dados Distribuído



Controle de Concorrência

Protocolos Baseados em Bloqueios

➤ **Protocolo da maioria**

➤ **tratamento de impasses (deadlocks) complexo**

- **diminui o engarrafamento do sistema**
- **diminui a vulnerabilidade**

Banco de Dados Distribuído



Controle de Concorrência

Protocolos Baseados em Bloqueios

➤ Protocolo Parcial

Similar ao protocolo da maioria, entretanto trata de forma distinta os modos compartilhado e exclusivo de bloqueios.

Compartilhado: solicitação encaminhada para apenas **um** site que possui uma réplica do item de dados;

Exclusivo: solicitação encaminhada a **todos** os sites que possuem réplicas do item de dados:

Banco de Dados Distribuído



Controle de Concorrência

Protocolos Baseados em Bloqueios

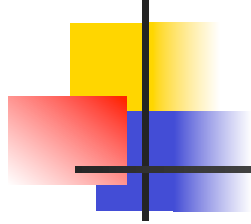
➤ Cópia Primária

No caso de existirem réplicas de Q, uma delas é considerada como a **cópia primária**. A cópia primária de reside em um único site: o **site primário** de Q.

Solicitações de bloqueio sobre Q são encaminhados ao site primário de Q.

Desvantagem: se o site primário de falhar, Q torna-se inacessível, mesmo que possua réplicas em sites ativos.

Banco de Dados Distribuído



Controle de Concorrência

Protocolos Baseados em Timestamp

O protocolo T0 para controle de concorrência em Banco de Dados Distribuídos pode ser aplicado diretamente em Bancos de Dados Distribuídos **sem replicações**, desde que seja possível a geração de **timestamps únicos** entre os sites.

Banco de Dados Distribuído



Tratamento de Impasses (Deadlocks)

- each local site maintains its own wait-for graph
 - if local graph has a cycle \rightarrow deadlock
 - if union of local graphs has a cycle \rightarrow deadlock

Banco de Dados Distribuído

Tratamento de Impasses (Deadlocks)

1. centralized approach

- single site constructs and maintains global wait-for graph
- ⚠ delays in receiving messages or receiving messages in the wrong order can result in detecting false cycles resulting in unnecessary rollback

Banco de Dados Distribuído

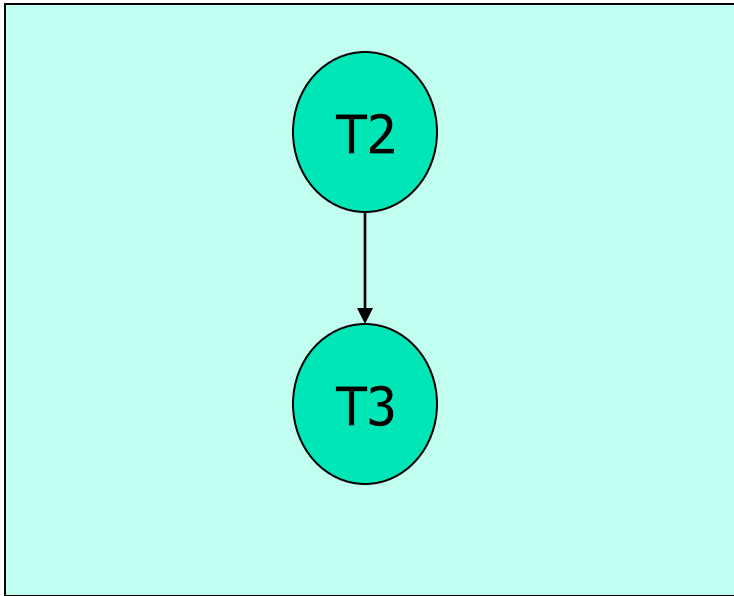
Tratamento de Impasses (Deadlocks)

2. distributed approach

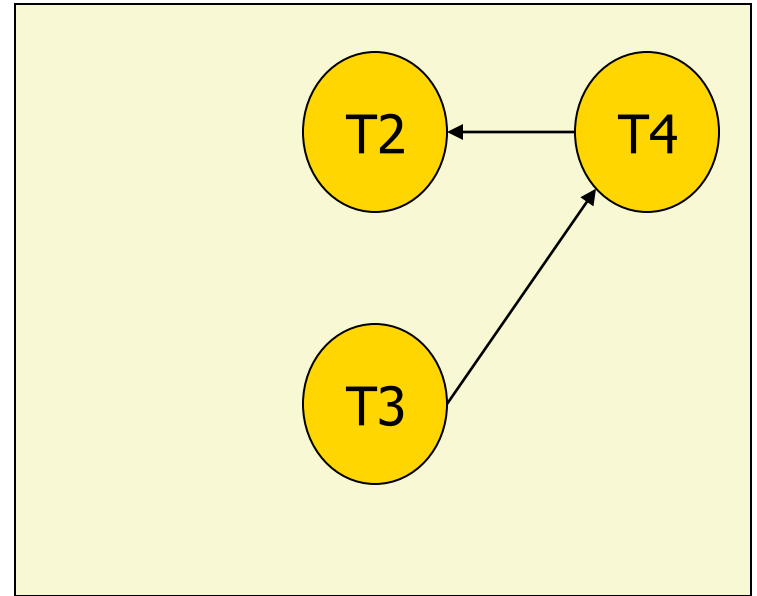
- each site maintains local wait-for graph with one additional node T_{ex} representing external transactions (to the site)
 $T_i \rightarrow T_{ex}$ if T_i is waiting for a data item at another site that is being held by any transaction
 $T_{ex} \rightarrow T_j$ if a transaction at another site is waiting for a data item held by T_j
- if the local graph has a cycle involving T_{ex} , then there is the possibility of a deadlock
 - ☞ a distributed deadlock detection algorithm is then invoked which will discover the cycle if one exists

Banco de Dados Distribuído

Tratamento de Impasses (Deadlocks)



Site S1

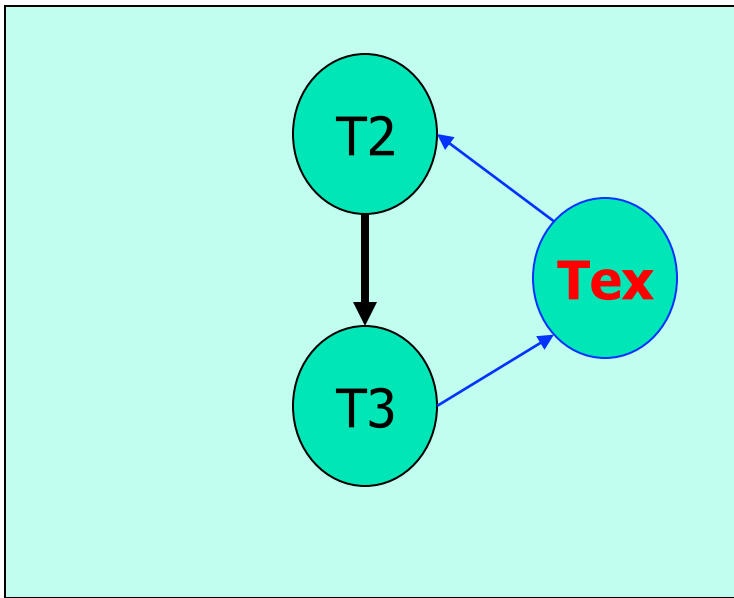


Site S2

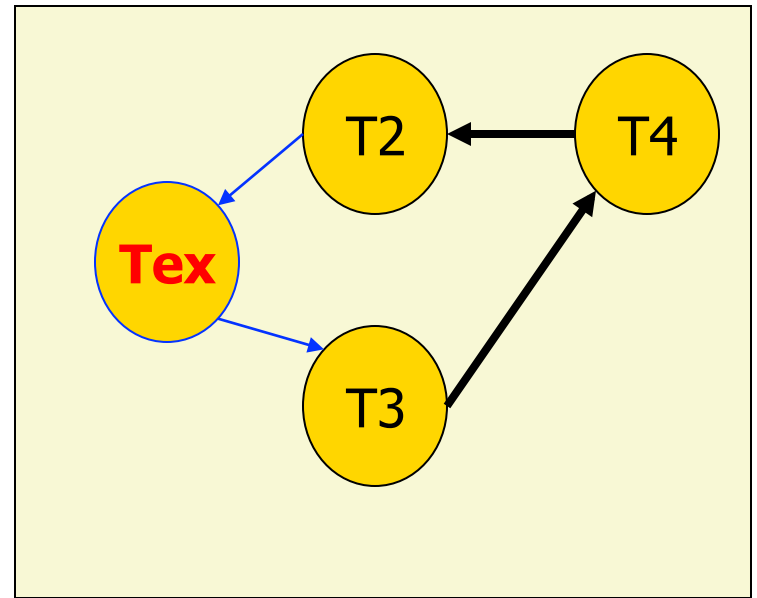
Não há impasse local, mas há impasse global!

Banco de Dados Distribuído

Tratamento de Impasses (Deadlocks)



Site S1



Site S2