Unidade VII

Bancos de Dados Distribuídos

- Armazenamento Distribuído dos Dados
- Consultas Distribuídas
- Transações Distribuídas
- Concorrência em BDs Distribuídos
- Arquitetura de BDs Distribuídos

Bancos de Dados Distribuídos

- BDs Distribuídos
  - BDs distribuídos não compartilham hardware
  - Se comunicam via rede (são fracamente acoplados)
  - Os dados estão distribuídos por várias localidades
- Diferenças em relação a BDs paralelos
  - BDs paralelos são fortemente acoplados
  - BDs paralelos mantém os dados em um mesmo local
  - BDs distribuídos são mais adequados para aumentar a robustez e a disponibilidade dos dados
  - BDs distribuídos são mais sujeitos a apresentar falhas de funcionamento e de segurança

Bancos de Dados Distribuídos

- BDs distribuídos podem ser classificados como:
  - Homogêneos: todos os sites usam o mesmo software
  - Heterogêneos
    - Usam software diferente
    - Podem usar esquemas de dados diferentes
    - Linguagem de consulta pode ser diferente
- Dificuldades a serem superadas
  - Controlar a consistência dos dados armazenados
  - Processar consultas e transações de modo distribuído
  - Controlar a concorrência e resolver conflitos
  - Conciliar as diferenças em sistemas heterogêneos

Armazenamento Distribuído dos Dados

- Em BDs distribuídos os dados podem ser:
  - Replicados
  - Fragmentados
  - Replicados e Fragmentados
- Replicação de Dados
  - Cada site armazena uma cópia completa dos dados
  - Vantagens: aumento de disponibilidade e paralelismo
  - Desvantagem: atualizações devem ser feitas em todos os sites para manter consistência entre réplicas
  - Apresenta bom desempenho nas operações de leitura, mas causa overhead nas operações de escrita

Armazenamento Distribuído dos Dados

- Fragmentação de Dados
  - Os dados são fragmentados, e cada site armazena um fragmento dos dados
  - Métodos são usados para fragmentação
    - Fragmentação horizontal, vertical ou mista
  - Fragmentação Horizontal
    - Cada fragmento possui um conjunto de tuplas de cada relação
    - Cada tupla de uma relação deve estar em pelo menos um site
    - A relação completa pode ser obtida fazendo a união dos fragmentos

Armazenamento Distribuído dos Dados

- Fragmentação de Dados (cont.)
  - Fragmentação Vertical
    - Relações são decompostas → cada fragmento é uma projeção da relação completa
    - A relação completa pode ser obtida fazendo a junção de todos os fragmentos
  - Fragmentação Mista
    - Combina fragmentação horizontal e vertical
- Fragmentação e Replicação de Dados
  - Dados são fragmentados horizontal ou verticalmente
  - Cada fragmento é mantido em mais de um site

Consultas Distribuídas

- Ao definir a escala de execução distribuída de uma consulta devemos considerar:
  - A replicação dos dados
  - A fragmentação dos dados
  - O custo da transmissão dos dados
  - A capacidade de processamento de cada site
- Influência da replicação
  - Devemos usar as réplicas que podem ser acessadas mais rapidamente
  - A criação de uma réplica de uma relação em um site que a acessa com freqüência pode ser vantajosa

Consultas Distribuídas

- Influência da fragmentação
  - Se a relação foi fragmentada horizontalmente usando o valor de um atributo, podemos considerar esta informação para processar a consulta
  - Podemos desconsiderar os fragmentos que sabemos que não possuem tuplas que irão satisfazer um determinado predicado
  - Podemos ignorar os fragmentos verticais que possuem atributos que não nos interessam
- Influência da transmissão dos dados
  - Devemos evitar trasmitir uma grande quantidade de dados pela rede devido ao retardo de transmissão

Consultas Distribuídas

- Influência da capacidade de processamento
  - Devemos procurar executar as operações em sites com maior poder de processamento
  - É necessário evitar sobrecarregar sites
- Processamento de consultas em BDs distribuídos deve garantir a transparência para o usuário
  - Localização física dos dados é transparente → endereços são manipulados internamente pelo SBD
  - Fragmentação dos dados é transparente → relações são recompostas pelo SBD a partir dos fragmentos
  - Replicação dos dados é transparente → SBD mantém atualizadas as réplicas das relações

Consultas Distribuídas

- Processamento de consultas distribuídas
  - Importante otimizar para evitar a transmissão de muitas tuplas pela rede
  - Exemplo: junção da relação r_1 no site S_1 com r_2 em S_2
    - Podemos transmitir r_1 até S_2 e calcular a junção, ou transmitir r_2 até S_1 e fazer o cálculo
    - Para definir a escala, devemos considerar:
      - O volume de dados a ser transportado
      - O tempo de transmissão de um site a outro
      - A capacidade de processamento de cada site
    - O custo total é dado pelo custo de transmissão mais o custo de processamento em cada site

Consultas Distribuídas

- Processamento de consultas em paralelo
  - Paralelismo intra-operação é inviável devido ao elevado custo de comunicação
  - Paralelismo inter-operação – independente ou por pipeline – pode trazer ganho de desempenho
- Exemplo: junção das relações r_1, r_2, r_3 e r_4 localizadas nos sites S_1, S_2, S_3 e S_4 respectivamente
  - Podemos calcular independentemente a junção de r_1 com r_2 em S_1 ou S_2 e a junção de r_3 com r_4 em S_3 ou S_4 (escolhendo o site com menor custo)
  - As tuplas podem ser passadas em pipeline para um dos sites, que então calcula o resultado final

Consultas Distribuídas

- Processamento de junções
  - Algoritmo de junção parcial
    - Calcula a junção em etapas
    - Minimiza o tráfego na rede
  - Ex.: Junção da relação r_1 no site S_1 com r_2 no site S_2
    - Calcular em S_1 a projeção de r_1 dos atributos em comum entre r_1 e r_2, e enviar para S_2
    - Calcular em S_2 a junção de r_2 com o resultado do passo anterior e enviar para S_1
    - Calcular em S_1 a junção de r_1 com o resultado do passo anterior, obtendo o resultado da junção

Transações Distribuídas

- Transações que afetam os dados em apenas um site são processadas como em BDs centralizados
- Transações que envolvem mais de um site precisam ser coordenadas
- As propriedades ACID precisam ser mantidas em todas as máquinas envolvidas na transação
  - Mudança de estado atômica nas várias máquinas
  - Todos os sites devem ter dados consistentes
  - Transações isoladas apesar do paralelismo
  - Alterações duráveis em todas as réplicas dos dados

Transações Distribuídas

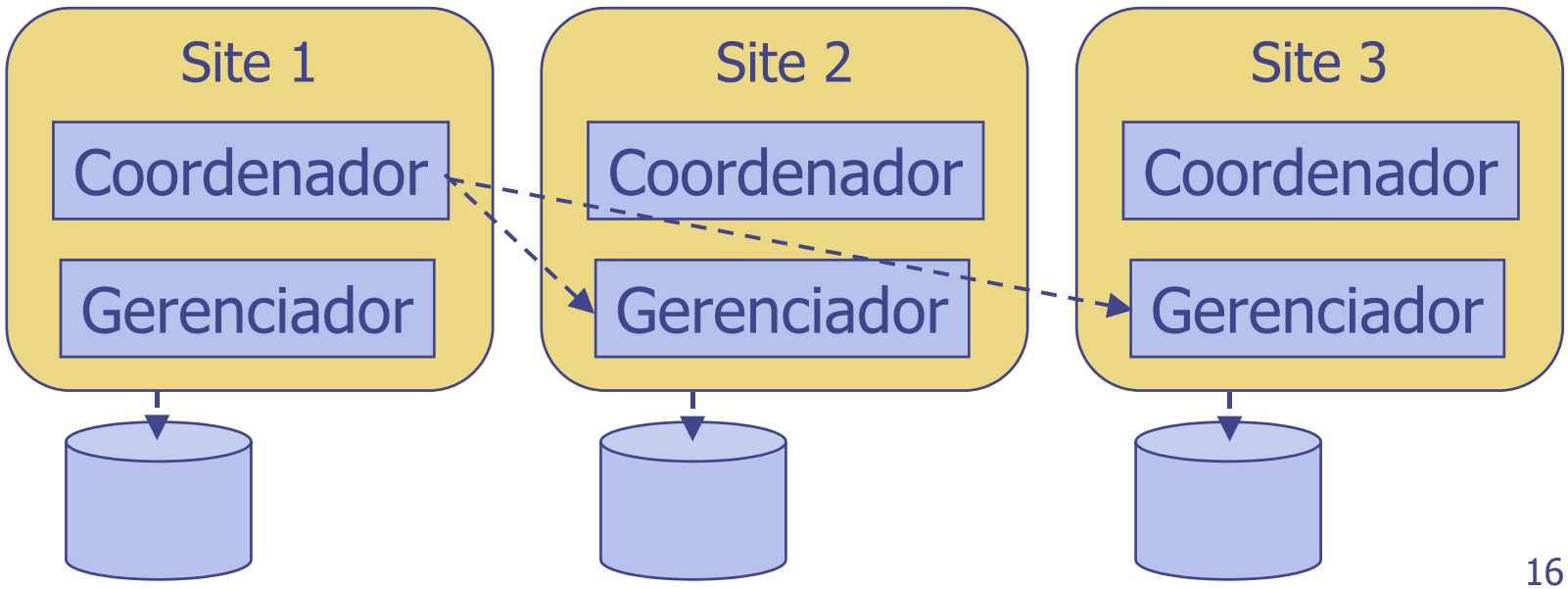
- Falhas podem afetar a execução de transações
- Tipos de falha
  - Falha de uma máquina: pode nos obrigar a abortar a transação caso a máquina não volte logo a funcionar
  - Falha na comunicação
    - Mensagem perdida ou corrompida: o protocolo de rede pode corrigir reenviando a mensagem
    - Falha de uma rota: o protocolo de roteamento pode corrigir escolhendo uma nova rota
    - Particionamento da rede: pode obrigar a abortar a transação caso não seja restabelecida a ligação

Transações Distribuídas

- Quando uma máquina não responde, não é possível saber se ocorreu uma falha na máquina ou se a rede foi particionada
- Tratamento de Falhas
  - Réplicas dos dados no site com falha deixam de ser atualizadas e não serão usadas em futuras consultas
  - Transações ativas no site com falha são abortadas
  - Se algum servidor central falhar, deve ser eleito um novo site para esta função (ex.: servidor de nomes, coordenador de concorrência, detector de deadlock)
  - Um site deve ter seu estado atualizado ao reintegrar-se ao sistema; qualquer conflito deve ser resolvido

Transações Distribuídas

- Coordenação de transações
  - Um coordenador de transações coordena a execução das transações distribuídas iniciadas por um site
  - Um gerenciador de transações administra em cada site as transações que acessam os dados locais



Transações Distribuídas

- Papel do coordenador de transações
  - Iniciar a execução da transação
  - Dividir a transação em sub-transações
  - Distribuir as sub-transações pelos sites apropriados para execução de cada uma delas
  - Fazer a efetivação ou o rollback em todos os sites
- Papel do gerenciador de transações
  - Controlar o acesso concorrente aos dados locais de modo a suportar transações distribuídas
  - Mater um log de operações para permitir recuperação de transações distribuídas

Transações Distribuídas

- Protocolos de efetivação garantem que todos os sites efetivarão a transação ou nenhum o fará
- Protocolo de efetivação em duas fases (2PC)
  - Inicia quando o coordenador recebe mensagens de todos os sites avisando que terminaram a execução
  - Fase 1: coordenador envia prepare T ao gerenciador de cada site participante; cada gerenciador responde se pode fazer o commit enviando ready T ou abort T
  - Fase 2: coordenador envia commit T se todas as respostas forem positivas, ou abort T caso receba uma resposta negativa ou se o timeout esgotar; cada gerenciador executa a ação correspondente

Transações Distribuídas

- Protocolo de efetivação em três fases (3PC)
  - Tolera até N falhas em 2N+1 sites
  - Fase 1: idêntica à fase 1 do protocolo de duas fases
  - Fase 2: o coordenador responde a todos os sites com a mensagem precommit T se todos indicaram dentro do tempo limite que podem efetivar a transação, ou com abort T em caso contrário; os sites devem responder com a mensagem ack T (reconhecimento)
  - Fase 3: ao receber K mensagens de reconhecimento, o coordenador manda a mensagem commit T a todos os sites, que ao recebê-la efetivam a transação

Transações Distribuídas

- Uso de logs nos protocolos de efetivação
  - Mensagens enviadas/recebidas são gravadas em logs
  - Caso o site reinicie após uma falha, ele deve verificar o que ocorreu com as transações registradas no seu log que ainda não foram efetivadas ou abortadas
- Comparação dos protocolos de efetivação
  - Custo de 3PC é mais alto devido ao maior número de mensagens trocadas pela rede
  - 2PC pode causar obstrução se o coordenador falhar
- Soluções possíveis para falha do coordenador
  - Um coordenador de backup pode assumir o seu lugar
  - Um novo coordenador pode ser eleito pelos sites

Transações Distribuídas

- Coordenador de backup
  - Reside em um site diferente do coordenador
  - Recebe as mesmas mensagens que o coordenador
  - Assume o lugar do coordenador ao detectar sua falha
- Algoritmo de eleição de coordenador
  - Sites podem pedir para assumir o lugar do coordenador caso detectem que este falhou
  - Os sites decidem em quem votar em função de seus endereços ou de um identificador do site
  - O novo coordenador deve requisitar as mensagens registradas nos logs de todos os sites para poder dar continuidade às transações em andamento

Concorrência em BDs Distribuídos

- Controle de concorrência é primordial em BDDs
  - É preciso garantir a consistência e o isolamento, apesar da fragmentação e da replicação dos dados
  - Devemos evitar deadlocks distribuídos, que são ainda mais difíceis de detectar e recuperar
- Os protocolos de controle de concorrência são modificados para trabalhar em BDs distribuídos
- Protocolo de bloqueio único (centralizado)
  - Um site funciona como gerenciador de locks, que administra todos os pedidos de bloqueio de dados
  - É simples, mas sujeito a falhas, e pouco escalável

Concorrência em BDs Distribuídos

- Protocolo de bloqueio múltiplo (descentralizado)
  - São usados vários gerenciadores em diferentes sites
  - Cada gerenciador administra os bloqueios de um conjunto de dados
  - Evita o 'gargalo' do protocolo centralizado
  - Impede o acesso ao dado se o gerenciador falhar
- Protocolo com cópia primária
  - Cada dado possui uma cópia primária em um site
  - O bloqueio é solicitado ao site com a cópia primária daquele dado, que administra todos os bloqueios
  - É tão simples e escalável quanto o protocolo anterior
  - Impede o acesso às réplicas se o primário falhar

Concorrência em BDs Distribuídos

- Protocolo da maioria
  - Cada site possui um gerenciador que administra o bloqueio dos dados locais
  - O bloqueio de dados com réplicas é concedido se a maioria dos sites permitir
  - Mais complexo para implementar que os anteriores
- Protocolo parcial
  - Assim como no protocolo da maioria, cada site possui um gerenciador de bloqueio dos dados locais
  - Bloqueios compartilhados são obtidos contactando o gerenciador de somente uma réplica do dado
  - Bloqueios exclusivos devem ser solicitados em todos os gerenciadores de bloqueio de todas as réplicas

Concorrência em BDs Distribuídos

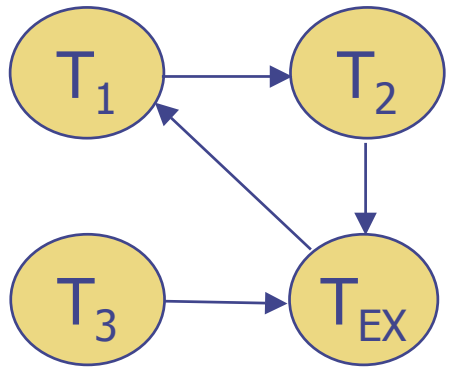
- Deadlocks e replicação
  - É preciso evitar deadlocks ao acessar réplicas
  - Se duas transações forem acessar um dado duplicado e cada uma delas bloquear uma réplica do dado, nenhuma das duas conseguirá prosseguir
  - Solução: obrigar que as transações bloqueiem as réplicas na mesma ordem
- Deadlocks e fragmentação
  - Situação semelhante à anterior pode ocorrer quando duas transações bloqueiam fragmentos de dados
  - Solução: bloquear fragmentos seguindo uma ordem

Concorrência em BDs Distribuídos

- Detecção de deadlock
  - O algoritmo de detecção de deadlock visto na Unidade V pode ser usado em BDs distribuídos se usarmos o protocolo de bloqueio centralizado
  - Para os demais protocolos, precisamos modificar o algoritmo de detecção de deadlock
    - Para montar o gráfico, um gerente central precisa obter informações sobre os locks mantidos por todos os gerenciadores de bloqueio
    - Podem aparecer ciclos falsos no gráfico devido ao atraso na comunicação → abortos desnecessários
  - Outra opção é fazer a detecção de modo distribuído

Concorrência em BDs Distribuídos

- Detecção de deadlock (cont.)
  - Na detecção distribuída, cada site monta um gráfico de espera com os bloqueios mantidos localmente
  - Um nó T_{EX} é adicionado ao gráfico para representar as esperas externas (dados bloqueados por outros sites)
  - Um ciclo envolvendo apenas nós locais indica deadlock
  - Um ciclo passando por T_{EX} indica um possível deadlock



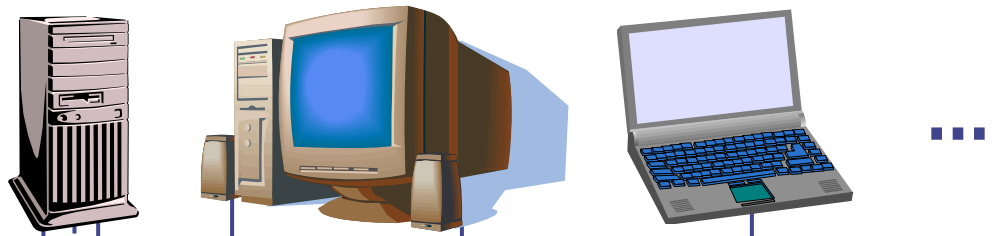
  - Temos que confirmar a possibilidade de deadlock contactando os gerenciadores dos sites envolvidos

Arquitetura de BDs Distribuídos

- Arquitetura Genérica de BDs

Usuários
- Adminstradores
- Programadores
- Usuários Finais

Esquemas Externos

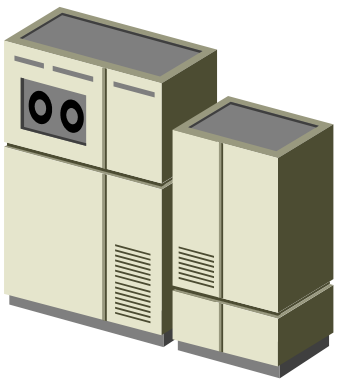


...

EE_1 EE_2 EE_3 ...

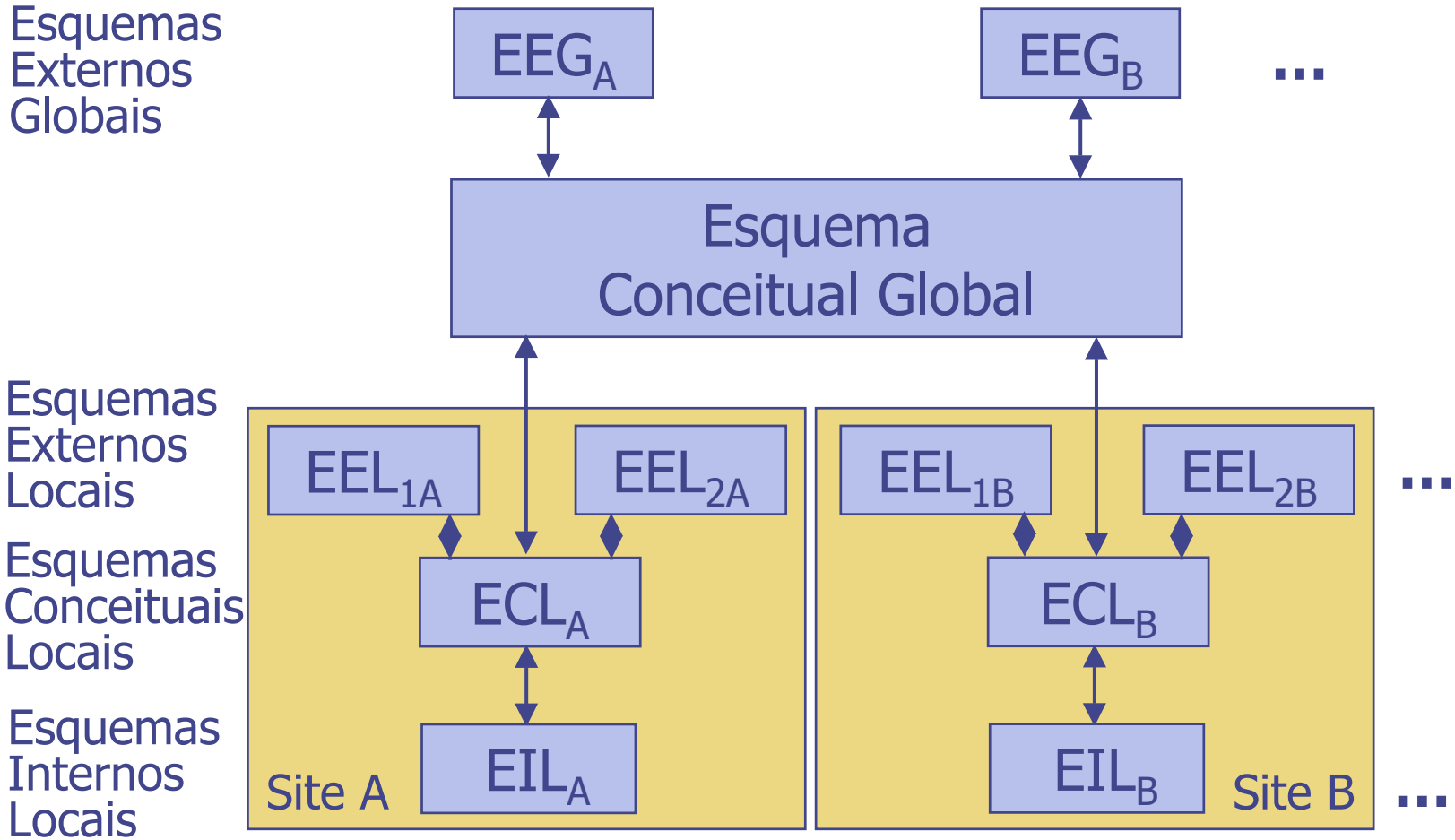
Esquema Conceitual

Esquema Interno



Arquitetura de BDs Distribuídos

- Arquitetura Genérica de BDs Distribuídos



Arquitetura de BDs Distribuídos

- Usuários de BDs Distribuídos
  - Tipos de Usuários
    - Administradores locais ou globais
    - Programadores locais ou globais
    - Usuários finais locais ou globais
  - Usuários Globais
    - Possuem visão global do sistema
    - Visualizam um esquema externo global
  - Usuários Locais
    - Acessam diretamente o servidor local
    - Visualizam um esquema externo local

Arquitetura de BDs Distribuídos

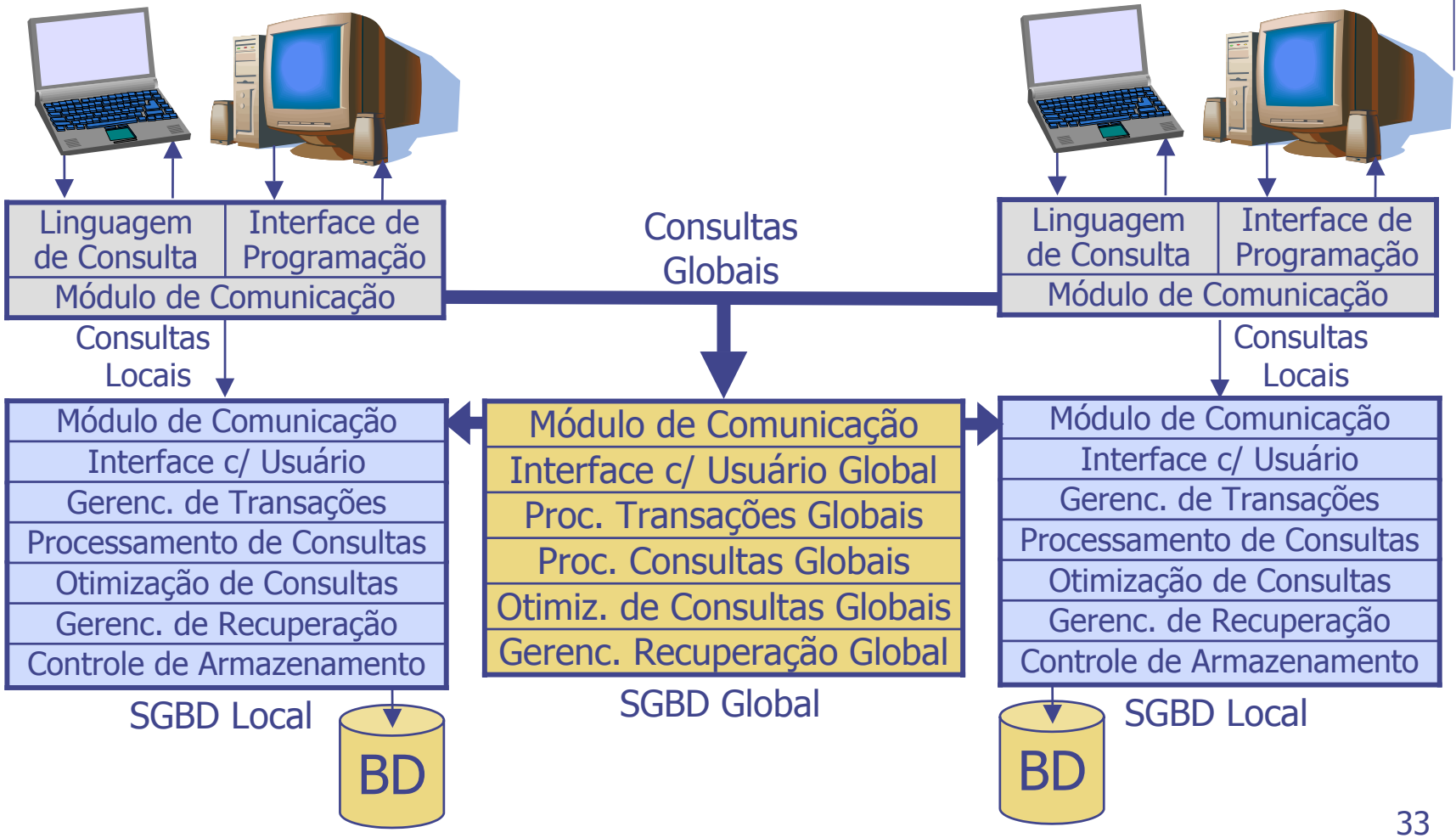
- Gerência de BDs

Arquitetura de BDs Distribuídos

- Problemas em BDs distribuídos
  - Otimização e processamento de consultas/transações distribuídas requer algoritmos/protocolos adequados
  - Mecanismos de controle e gerenciamento devem trabalhar de maneira integrada
- Problemas em BDDs heterogêneos
  - Precisamos conciliar as diferenças entre:
    - Modelos lógicos
    - Linguagens de definição e manipulação de dados
    - Formatos de dados: língua, ordenação dos bits, tamanho dos tipos de dados e representação na memória, tabelas de caracteres, etc.

Arquitetura de BDs Distribuídos

- Sistema de Gerência de BDs Distribuídos



Arquitetura de BDs Distribuídos

- Projeto de BDs Distribuídos
  - Abordagens de projeto
    - Top-Down: usada em sistemas construídos do zero, geralmente homogêneos
    - Bottom-Up: usada quando os sistemas locais já estão instalados e precisam ser integrados
  - Questões a serem respondidas no projeto de BDDs
    - Onde colocar os sites?
    - Como fragmentar dados e distribuí-los pelos sites?
    - Replicação de dados é necessária?
    - Como integrar diferentes sistemas?