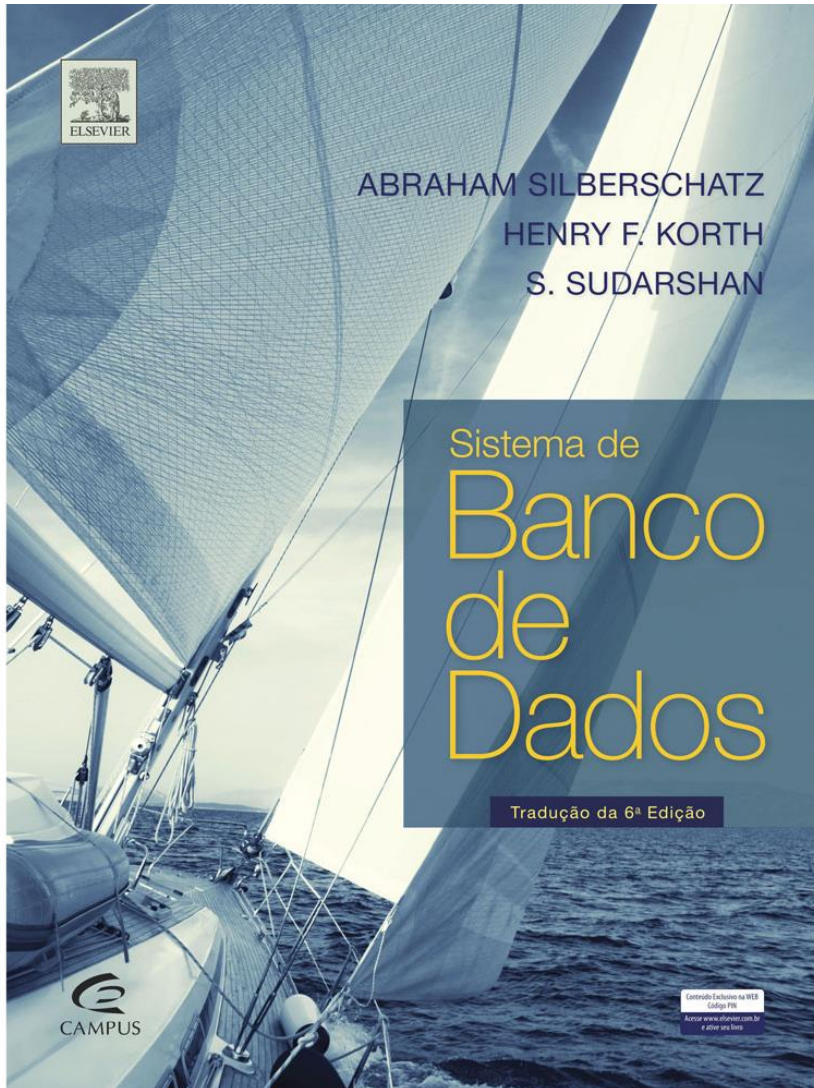


Banco de Dados I

Prof. Diego Buchinger
diego.buchinger@outlook.com
diego.buchinger@udesc.br

Profa. Rebeca Schroeder Freitas
Prof. Fabiano Baldo

Aula Inaugural



- plano de ensino
- Bibliografia (próximo slide)
- buchinger.github.io
- background dos alunos:
 - cidade natal?
 - quantos semestres na UDESC?
 - trabalha? onde? com o quê?
 - já usou Banco de Dados?
 - linguagens que conhece

Aula Inaugural

Bibliografia Básica

- CHEN, P. Gerenciamento de Banco de Dados. São Paulo: McGraw-Hill, 1990.
- DATE, C. J. Introdução a Sistemas de Banco de Dados. 7ª. Edição. São Paulo: Campus, 2000.
- ELMASRI, R.. NAVATHE, S. B., Sistemas de Banco de Dados – Fundamentos e Aplicações. 3ª. Edição. Rio de Janeiro: LTC, 2000.
- HEUSER, C. A. Projeto de Banco de Dados, 2001.
- SILBERSCHATZ, A: KORTH, H. F.; SUDARSHAN, S. Sistema de Banco de Dados, 2005.

Bibliografia Complementar

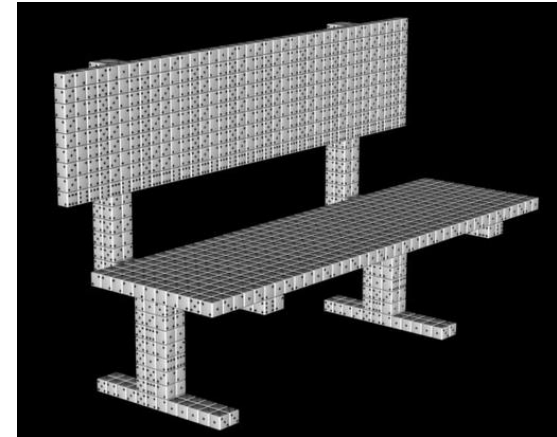
- Batini, C., Ceri. S., Navathe S.B. Conceptual Database Design: An Entity-relationship Approach. 1992
- Ramakrishnan, R e Gehrke, J. Sistemas de Gerenciamento de Banco de Dados. 3ª Ed., 2008

Introdução - BDs

persistência

Evolução: memória → arquivos → banco de dados

- um Banco de Dados (BD) é um conjunto de dados que representa aspectos do mundo real (universo de discurso) de forma organizada e que permite extrair informações [dado vs. informação]
 - composto por dados e metadados!
 - **Arquivo de dados:** conjunto de registros relacionados [entidade, tabela, relação]
 - **Registro:** conjunto de dados relacionados [tupla, linha]
 - **Campo / dado:** valor armazenado [atributo, coluna]

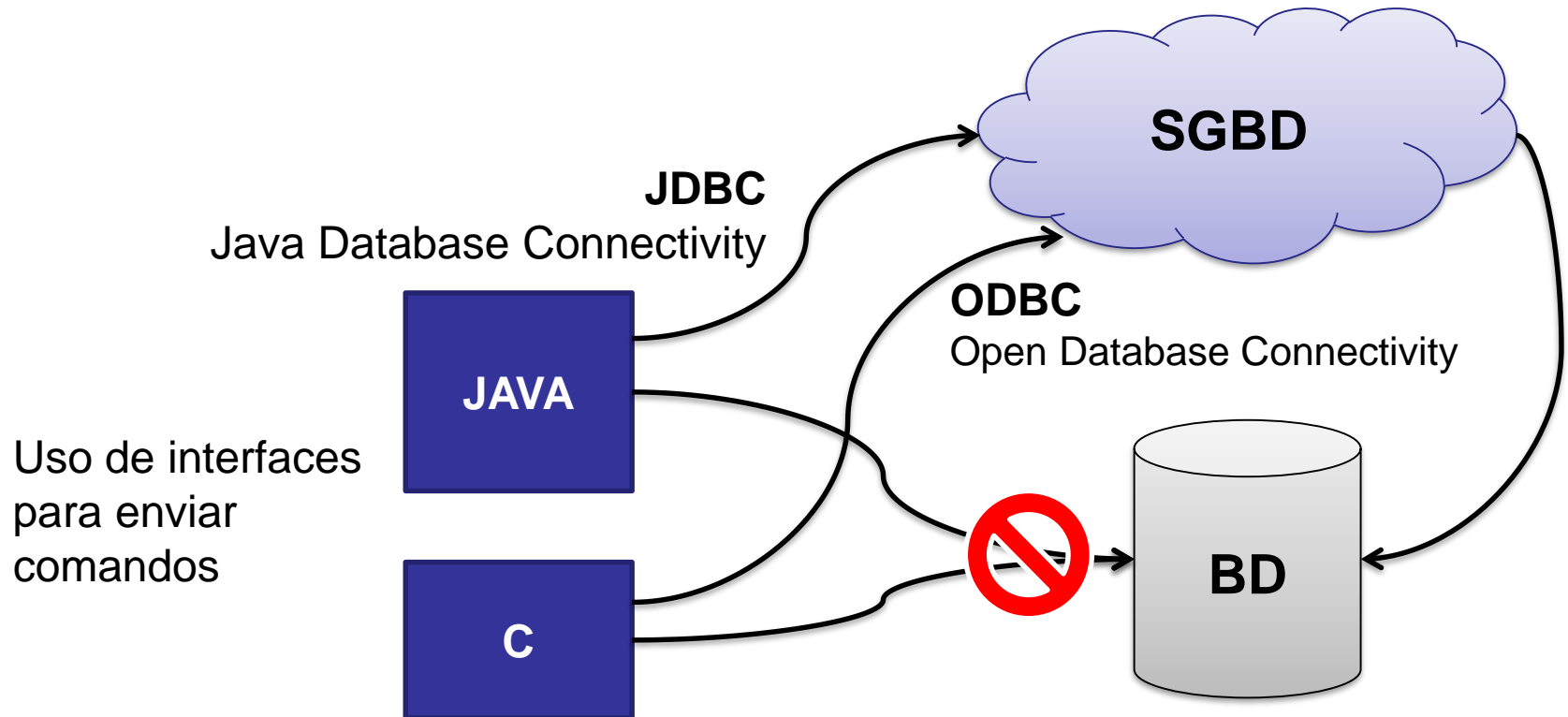


Introdução - SGBDs

Sistema Gerenciador de Banco de Dados (SGBD)

- É um software que gerencia o banco de dados:
 - criar e manter Banco de Dados
 - realizar consultas eficientes no Banco de Dados
 - fazer interface com aplicação
 - garantir desempenho, segurança e confiabilidade
- Aplicações:
 - Informação empresarial
 - Bancos e finanças
 - Universidades, companhias aéreas, telecomunicações
 - (...)

Introdução - SGBDs



Introdução - SGBDs

Vantagens

- Fornece processamento eficiente de consulta (índices, *caching*, otimização de consulta)
- Oferece mecanismos de backup e recuperação
- Pode assegurar restrições de integridade (valor único, integridade referencial, verificação de restrições)
- Criação de gatilhos e procedimentos

Desvantagens

- Requer algum investimento inicial (\$, tempo, hardware)
- Pode não ser necessário caso aplicações sejam simples e bem definidas, sem mudanças

Modelos de Dados descrevem a semântica, os relacionamentos, as restrições e as operações com dados

- PRIMEIRA GERAÇÃO
 - Época: década de 1960 e 1970
 - Modelo: sistema de arquivos
 - Características:
 - ❖ gerenciamento de registros sem relacionamentos
 - ❖ utilizado principalmente em sistemas de mainframes da IBM
 - Exemplos: MVS / VSAM (sist. de arq. orientados a registro)

Histórico e Modelo de Dados

- SEGUNDA GERAÇÃO
 - Época: final da década de 1960 e década de 1970
 - Modelo: hierárquico e em rede (grafo)
 - Características:
 - ❖ primeiros sistemas de bancos de dados
 - ❖ acesso navegacional
 - ❖ modelo hierárquico organizado como estrutura de árvore (registro pode ter múltiplos filhos vinculados por *links*, mas um filho só pode ter um registro pai)
 - ❖ deficiência: falta de uma linguagem de consulta de alto nível
 - Exemplos: IMS, ADABAS, IDS-II

- TERCEIRA GERAÇÃO
 - Época: Meados da década de 1970 até hoje
 - Modelo: relacional
 - Características:
 - ❖ baseado em: entidades, atributos e relacionamentos / tabelas
 - ❖ simplicidade conceitual e de modelagem
 - ❖ teve grande aceitação e se transformou em um padrão
 - Exemplos: DB2, Oracle, MS SQL Server, MySQL

- QUARTA GERAÇÃO
 - Época: Meados da década de 1980 até hoje
 - Modelo: orientado a objetos e relacional estendido
 - Características:
 - ❖ suporte a conceitos e visão de orientação a objetos
 - ❖ suporte a dados complexos (dados multimídia, geográficos)
 - ❖ ajudam no problema da divergência de impedância (diferença entre a representação na aplicação e no BD)
 - ❖ deficiência: estrutura continua rígida
 - Exemplos: Versant, Objectivity/DB, DB/2 UDB, Oracle 10g

- QUINTA GERAÇÃO
 - Época: Meados da década de 1980 até hoje
 - Modelo: não estruturado ou semiestruturado (NoSQL)
 - Características:
 - ❖ representação dos dados (estrutura) flexível [ex XML]
 - ❖ não garante algumas propriedades (ex atomicidade)
 - ❖ desempenho melhor
 - ❖ voltado para aplicações de BigData
 - Exemplos: Google BigTable, CouchDB, MongoDB

Por que usar um SGBD?

Problemas da primeira geração de BDs (sem SGBDs):

- Redundância e Inconsistência: informação replicada com possíveis dados divergentes [ex: aluno de dois cursos]
- Dificuldade de consulta: dificuldade em filtrar dados de maneiras diferentes [ex: filtrar alunos que moram em uma determinada área e filtrar alunos em exame neste semestre]
- Isolamento dos dados: dificuldade em escrever programas para filtrar dados que estão em arquivos e possivelmente formatos/estruturas diferentes [ex: arquivos XML e CSV]
- Problemas de integridade: dificuldade em alterar todos os programas existentes a fim de garantir certas restrições de consistência [ex: $\text{saldo} \geq 0$].

Por que usar um SGBD?

Problemas da primeira geração de BDs (sem SGBDs):

- Problema de atomicidade: dificuldade em garantir que toda uma transação é feita por completo ou nada é feito [ex: apagão]
- Problemas com acesso concorrente: o uso inadequado de múltiplos processos ou *threads* pode gerar dados inconsistentes ao trabalhar simultaneamente com um mesmo dado [ex: transferências bancárias]
- Problemas de segurança: garantir que apenas os usuários corretos possam ver determinados dados [ex: secretária não deveria poder ver os dados das folhas de pagamento; dados nos arquivos devem estar criptografados]

Por que usar um SGBD?

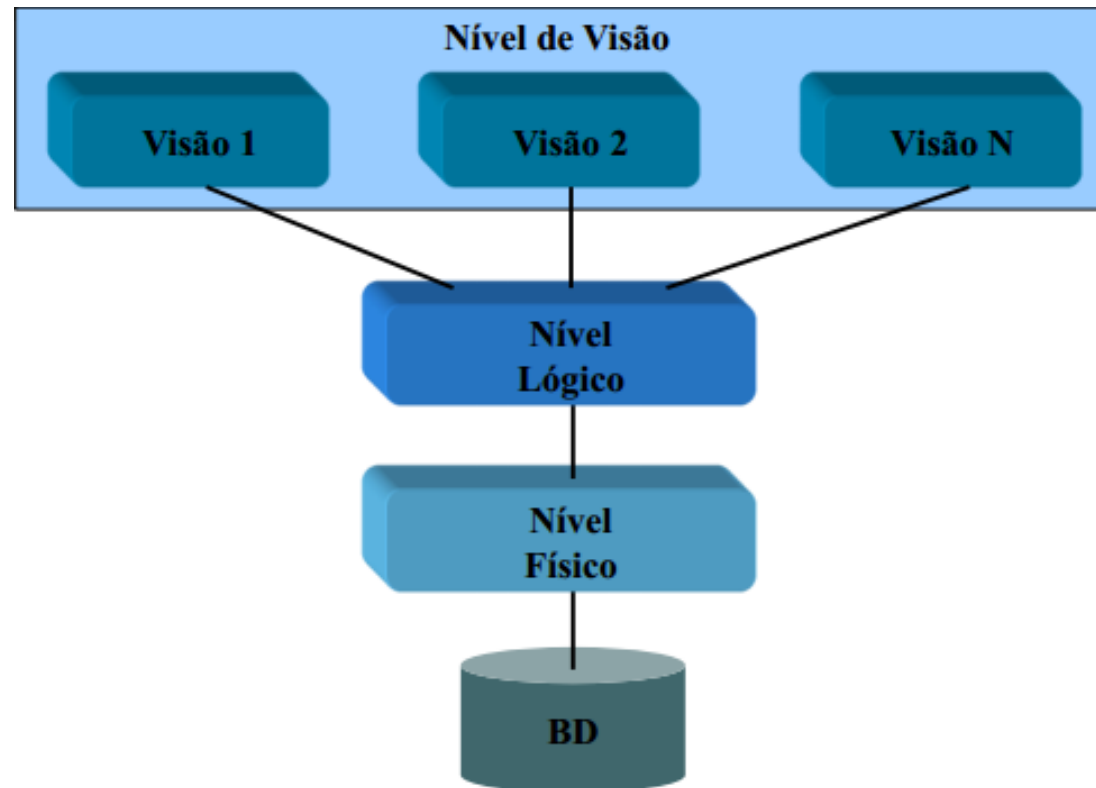
Contrapartida com o uso de um SGBD:

- Redundância controlada (requer bom uso do administrador)
- Eficiência na manipulação e consulta de dados
- Independência de dados (permite modificações estruturais)
- Pode garantir integridade dos dados (basta especificar)
- Pode garantir atomicidade nas operações
- Pode garantir segurança no acesso concorrente
- Pode garantir controle de acesso aos dados
- Proporciona diferentes níveis de abstração

Níveis de Abstração

Os SGBDs fornecem aos usuários uma visão abstrata dos dados e não exatamente como estão armazenados.

É comum o uso de uma abstração de 3 níveis:



Níveis de Abstração

Independência de Dados: os níveis de abstração devem manter um grau de independência uns dos outros:

- Independência Lógica: capacidade de modificar o esquema lógico sem necessidade de reescrever as aplicações/visões
- Independência Física: capacidade de modificar o esquema físico sem necessidade de reescrever o modelo lógico.

Esquema: é a descrição dos dados do BD; seu projeto geral; costuma ser pouco alterado.

Instância ou Estado: coleção de dados armazenados no BD em um determinado momento (fotografia dos dados)

Propriedade ACID

BDs relacionais foram introduzidos com a proposta de garantir certas características. Estas se tornaram uma referência para os SGBDs, que podem implementá-las ou não. São elas:

- **Atomicidade:** realizar operações não divisíveis; **transação** é executada por completa ou não é executada

Transação é um conjunto de operações que realiza uma única função lógica

Propriedade ACID

- **Consistência:** o BD deve respeitar certas regras impostas (restrições de consistência) de tal forma que o BD esteja consistente antes e após qualquer transação
 - **Restrição de chave:** restringe duplicidade de chaves
 - **Restrição de domínio:** restrições de tipos
 - **Integridade de vazio:** restrição de obrigatoriedade de valor
 - **Integridade referencial:** garante o vínculo entre dois ou mais registros (chave estrangeira) em alterações ou exclusões
 - **Assertivas:** outras condições referentes aos dados que precisam ser satisfeitas

Propriedade ACID

- **Isolamento:** duas transações não podem se interferir gerando um resultado inconsistente (ex: transferência bancária). O SGBD pode, contudo, paralelizar transações que não atuam sobre um mesmo item
- **Durabilidade:** valores criados ou alterados em uma transação devem persistir, mesmo havendo falha no sistema [ex: transação terminou, alteração no buffer, mas não alterou arquivo de dados e ... caiu a energia / ocorreu um erro no sistema]

Características BDs NoSQL

- Foram desenvolvidos para ter alta escalabilidade, gerenciar grandes quantidades de dados e garantir disponibilidade
- Abrem mão de algumas propriedades ACID
- **Escalabilidade horizontal:**
aumento no # de máquinas utilizadas para processamento e armazenamento
 - controlar concorrência seria mais dispendioso e por isso opta-se por ausência de bloqueios
 - é comum o uso de *Sharding* – divisão das tabelas pelos nós da rede (complica a lógica relacional)

NOTA: escalabilidade vertical consiste em aumentar o poder de processamento e armazenamento das máquinas

Características BDs NoSQL

- **Ausência de esquema ou esquema flexível:**
 - [+] facilita escalabilidade e aumento de disponibilidade
 - [-] dificulta ou impossibilita a garantia de integridade
- **Suporte nativo a replicação:**
 - reduz o tempo para recuperar informação
 - Master-Slave: escreve-se no nó mestre e replica-se nos nós escravos (melhora a leitura, mas escrita se torna gargalo)
 - Multi-Master: melhora a velocidade de escrita mas pode causar conflito de dados.

Modelos de Dados NoSQL

Os principais tipos de modelos de dados NoSQL são:

- **Chave-valor:** grande tabela *hash* onde os valores estão associados a uma chave (um índice)
 - fácil implementação (basicamente `get()` e `set()`)
 - não permite recuperação de dados com consultas complexas
 - Exemplos: Dynamo, Redis, Riak e GenieDB
- **Orientado a colunas:** dados são indexados por uma tripla (linha, coluna e timestamp)
 - operações de leitura e escrita de uma linha são atômicas
 - permite consistência e fácil particionamento
 - Exemplos: BigTable [Google], Cassandra [Facebook]

Modelos de Dados NoSQL

Os principais tipos de modelos de dados NoSQL são:

- **Orientado a Documentos:** armazena coleções de documentos que possuem identificadores (chaves) e valores
 - não possui um esquema rígido (maior flexibilidade)
 - Exemplos: CouchDB e MongoDB
- **Orientado a Grafos:** composto por nós, arestas (relacionamentos) e propriedades dos nós
 - maior desempenho para consultas complexas (junções etc)
 - Exemplos: Neo4j, AllegroGraph e Virtuoso

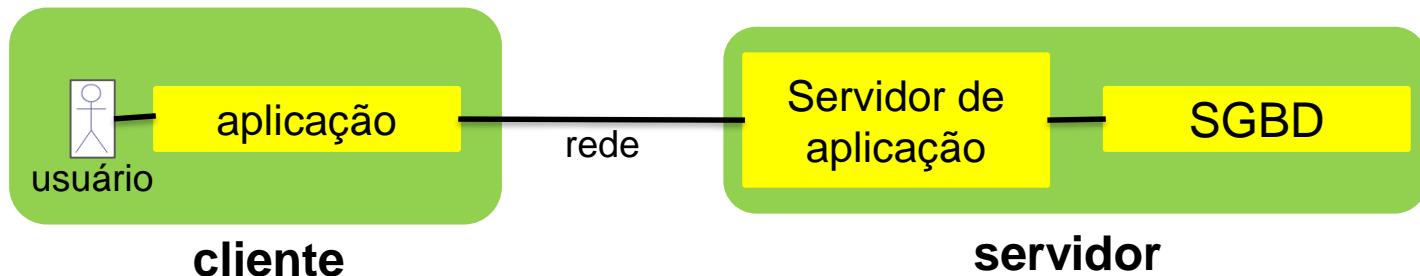
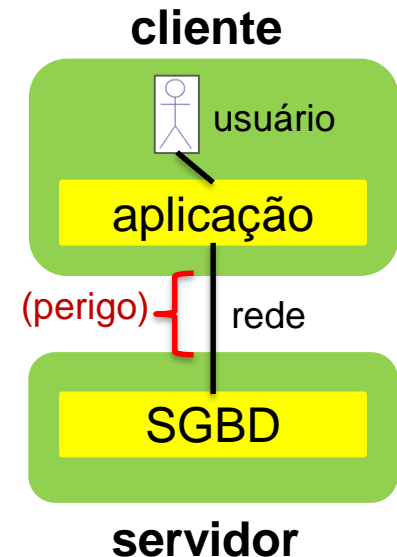
- **Modelos de acesso:**

- Centralizado: utiliza uma máquina mainframe que realiza o processamento e mantem o(s) BD(s).
- Cliente - Servidor: os BDs ficam divididos em vários terminais onde o acesso é específico.
- Distribuído: os BDs estão distribuídos e um SGBD faz a ponte de acesso aos dados [ex: matriz e filial].

Arquitetura

- **Particionamento de aplicações:**

- Duas camadas: o sistema é dividido em um componente para o usuário (cliente) e o SGBD reside em um servidor
- Três camadas: existe uma aplicação no servidor que se comunica com o SGBD

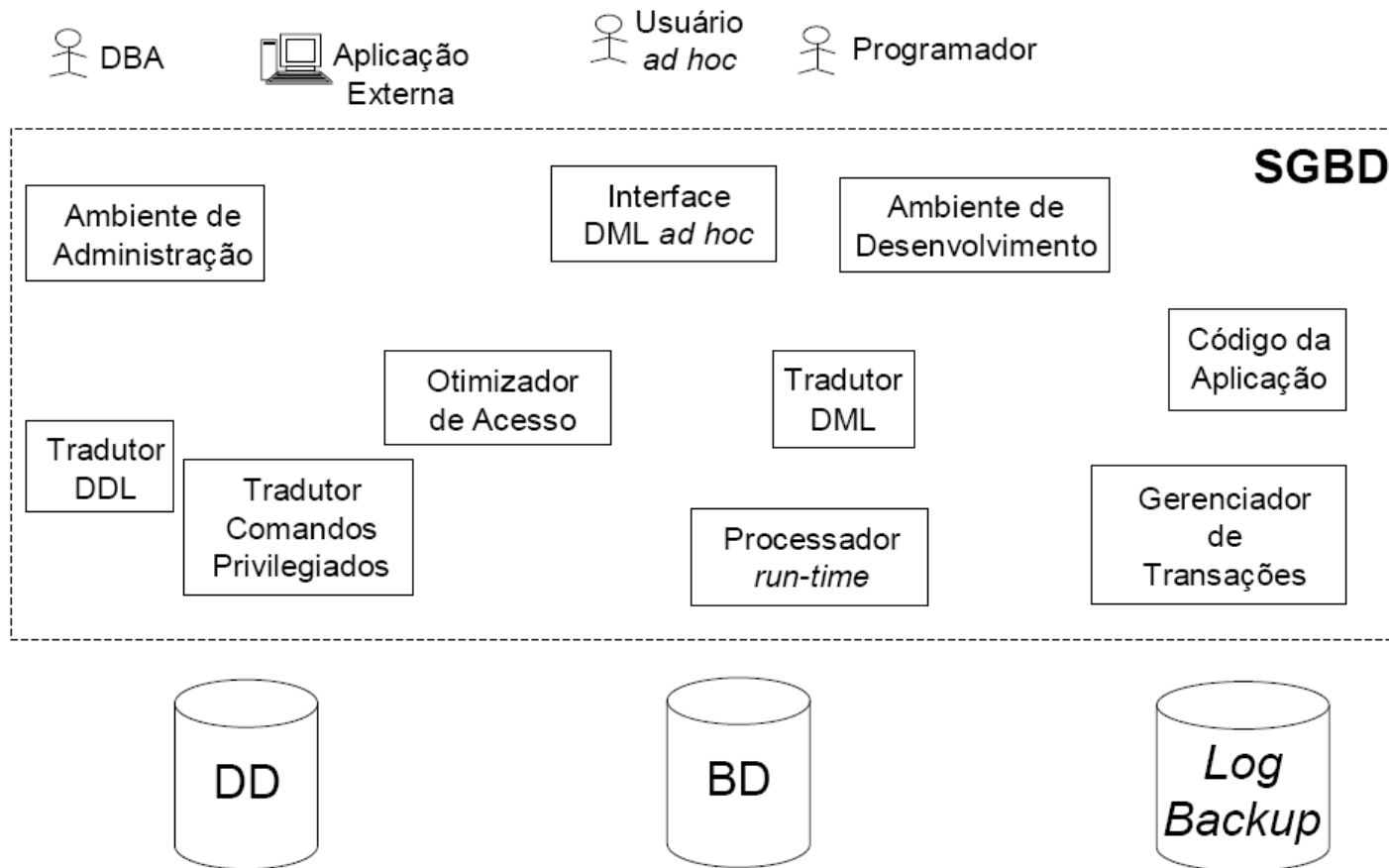


- **Linguagens:**

- DDL – Data Definition Language: permite especificar esquemas, propriedades, estruturas de armazenamento, restrições de consistência etc.
- DML – Data Manipulation Language: permite acessar e manipular dados (recuperar, inserir, excluir e modificar).

Arquitetura

- **Estrutura:** um SGBD é dividido em módulos que tratam de responsabilidades específicas



Arquitetura - Estrutura

Catálogo – metadados:

- especificação do esquema
- restrições de integridade
- autorizações de acesso
- localização dos arquivos
- configurações e estimativas
 - tempo de timeout
 - nro. máximo de usuários
 - tamanho do buffer
 - intervalo de backup auto

Repositório de:

- cópias do BD
- histórico de transações

Repositório de:

- arquivos de dados
- arquivos de índices

DD

BD

Log
Backup

Usuário
ad hoc

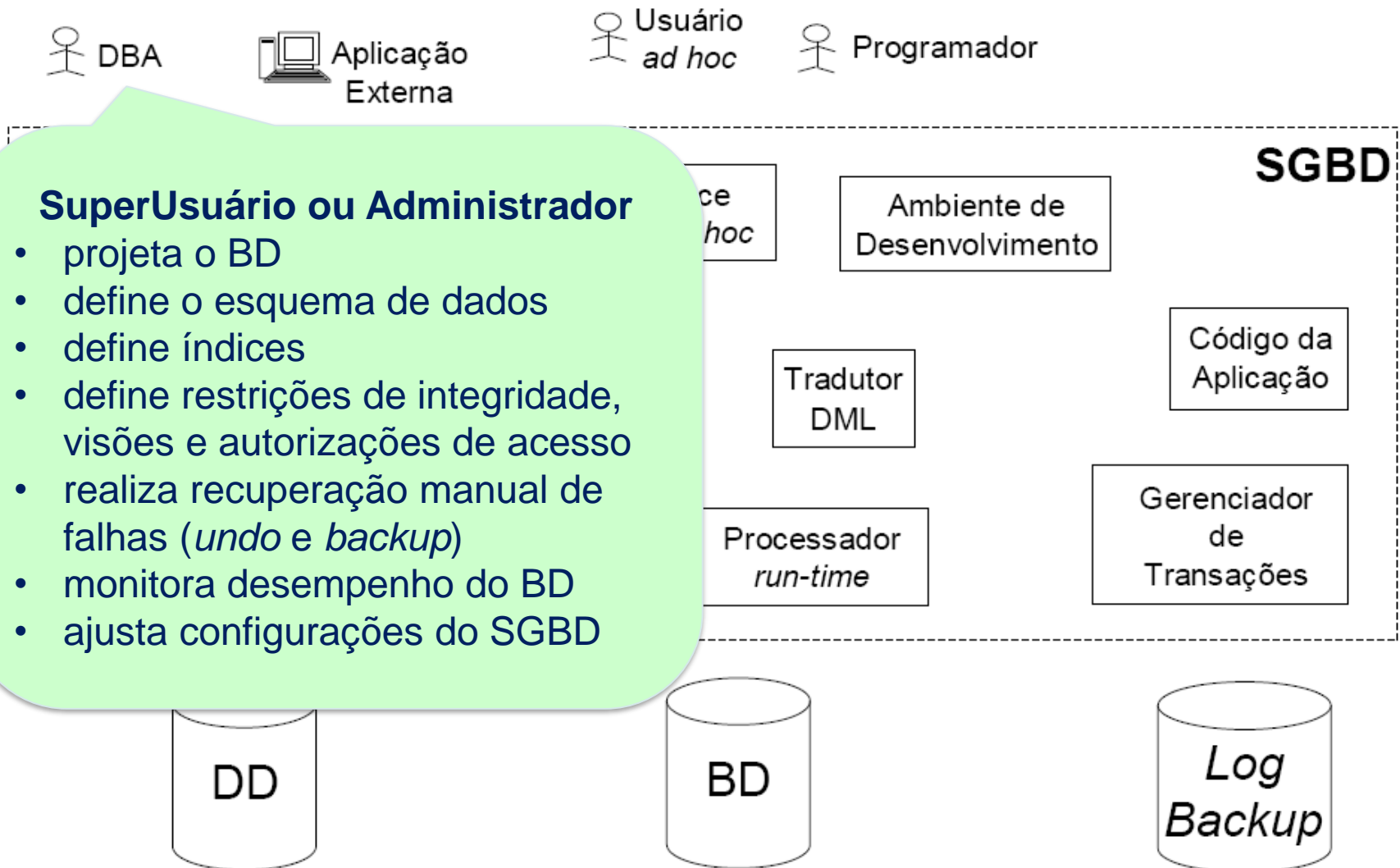
Interface
VL ad hoc

da
ção

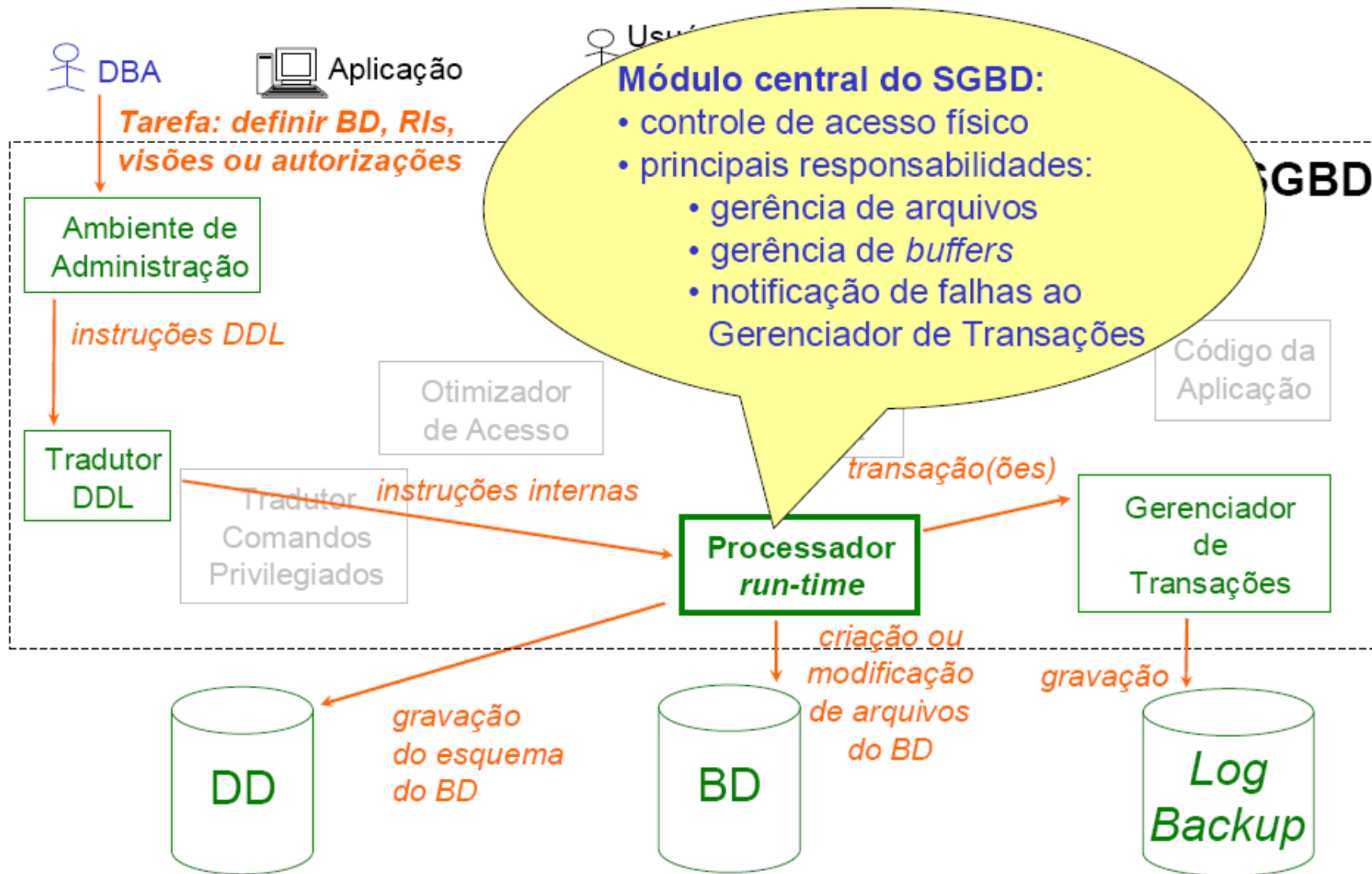
Gerador

Transações

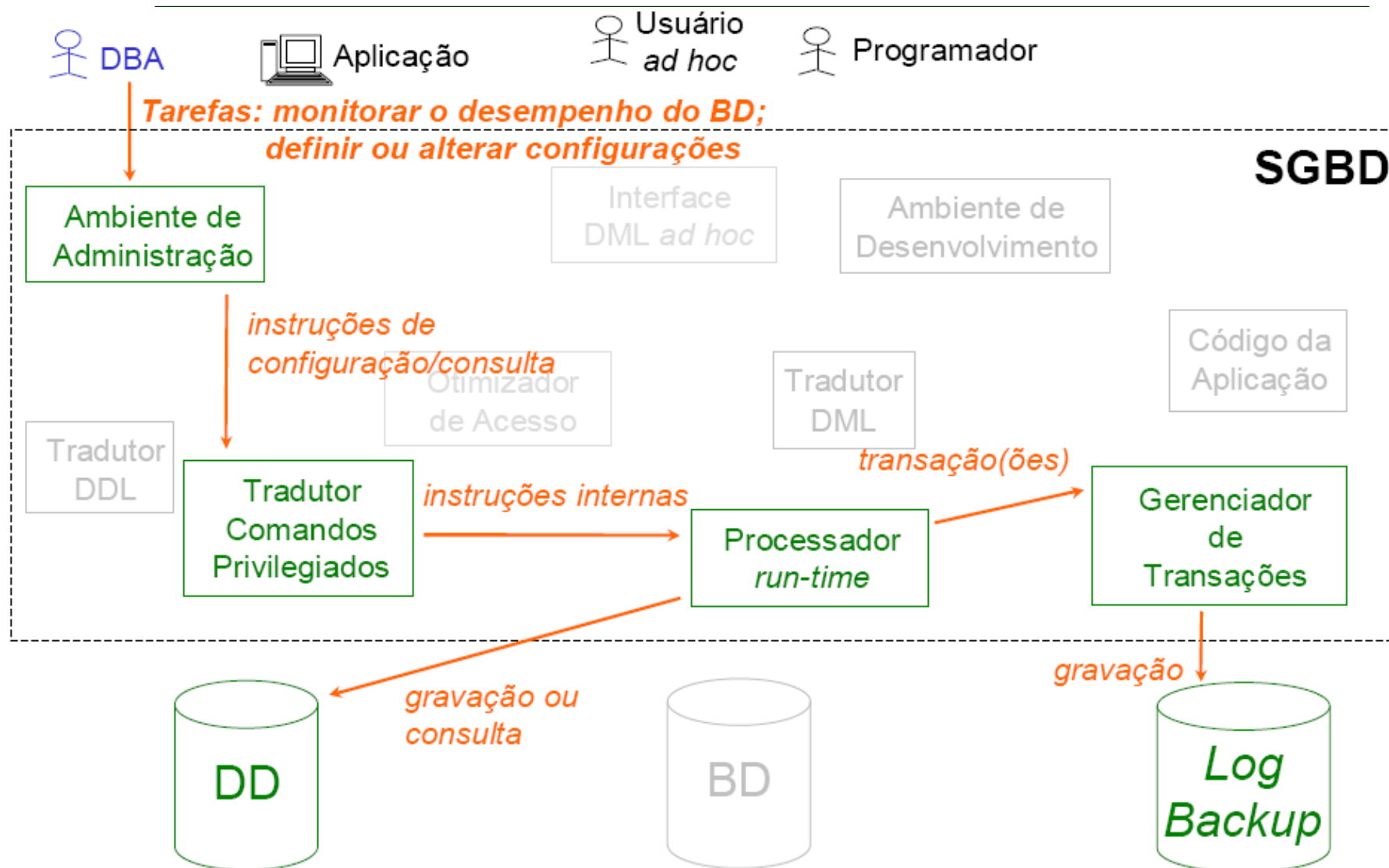
Arquitetura - Estrutura



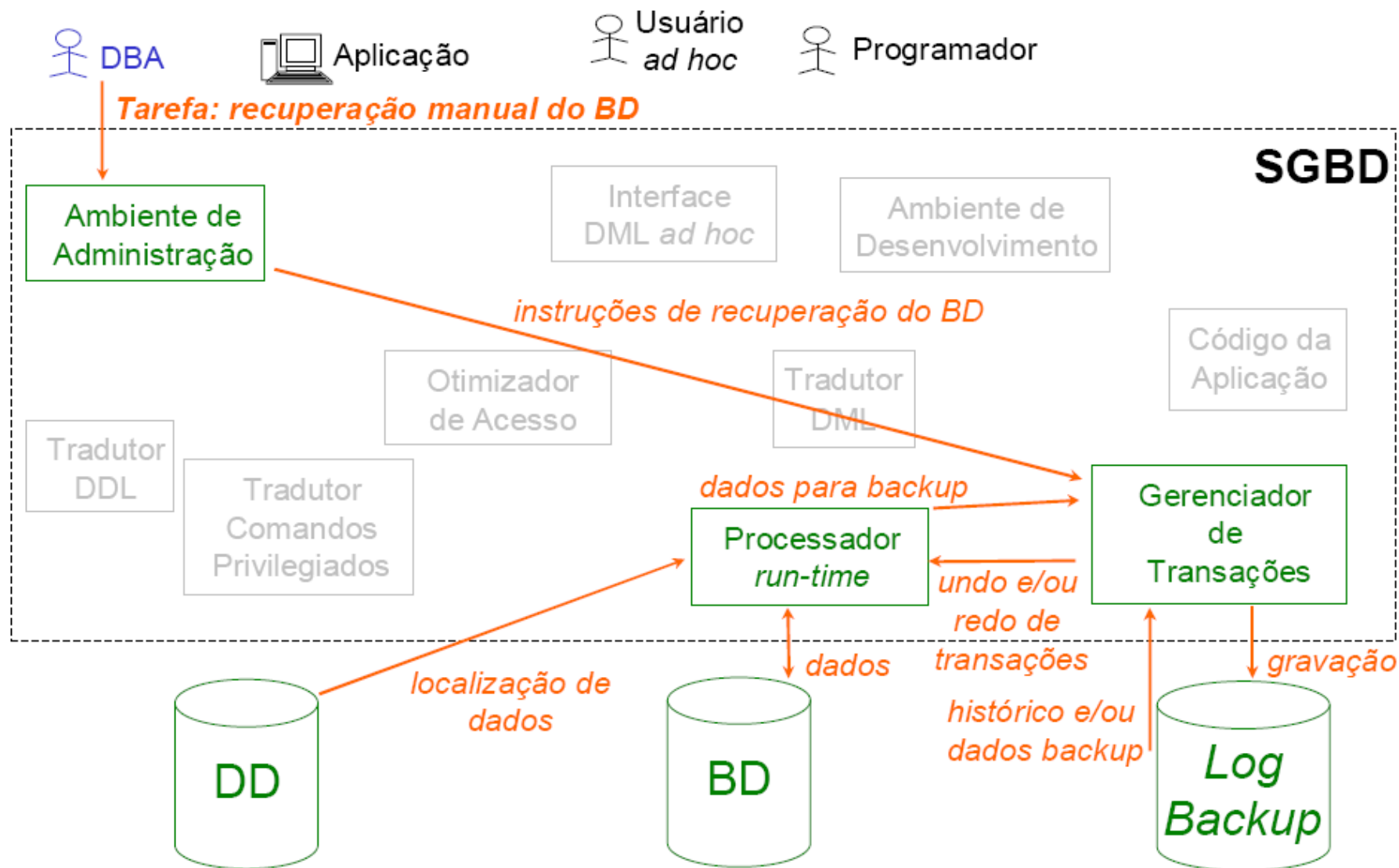
Arquitetura - Estrutura



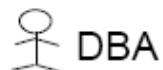
Arquitetura - Estrutura



Arquitetura - Estrutura



Arquitetura - Estrutura



DBA



Aplicação



Usuário
ad hoc



Programador

SGBD

Ambiente de
Administração

- acesso ao BD através de comandos DML pré-compilados e embutidos no seu código
- SGBDs suportam *bindings* com várias LPs (LHs)
- exemplo: SQL Server (SQL embutido em C (ferramenta ESQL/C)):

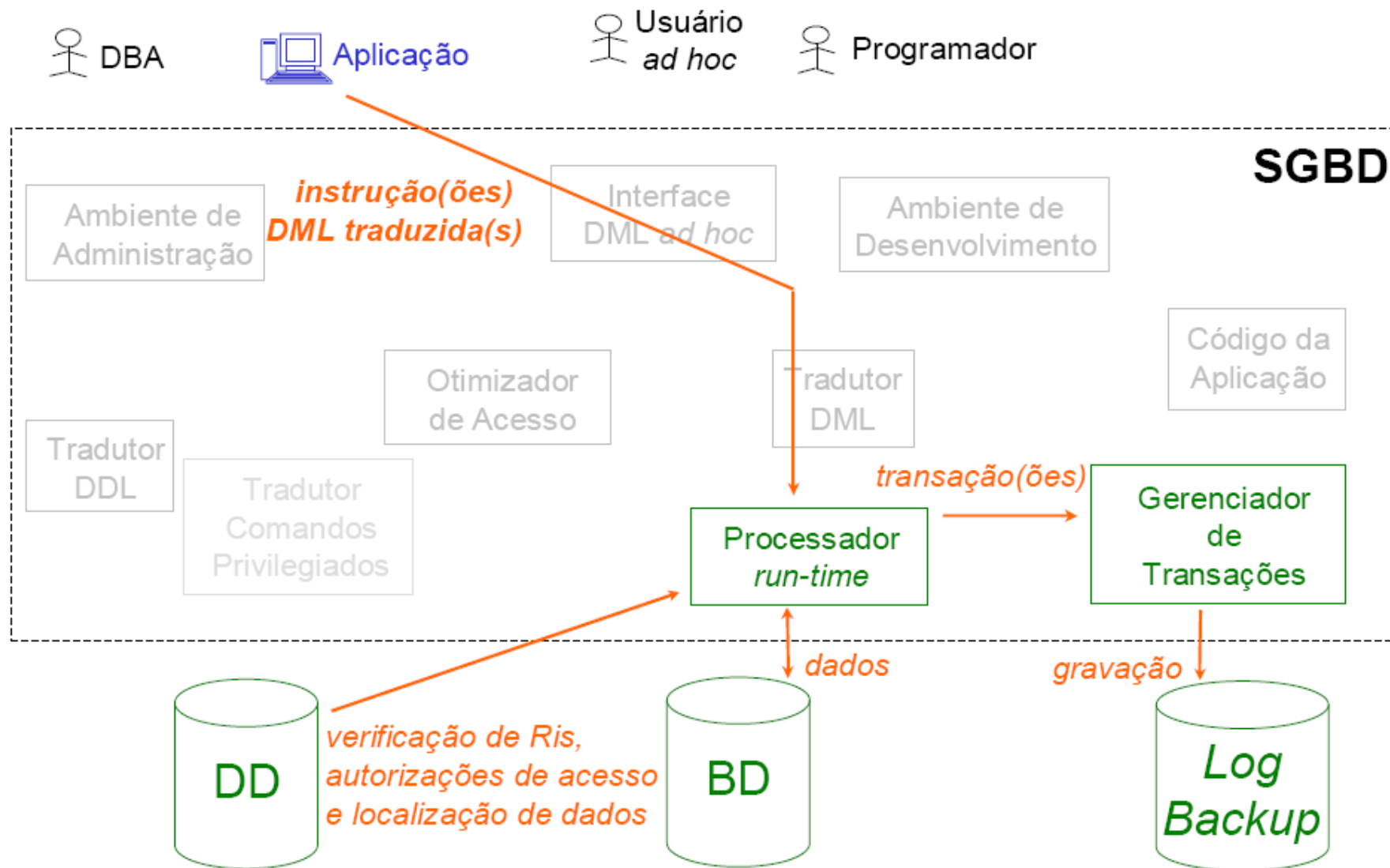
```
...  
EXEC SQL BEGIN DECLARE SECTION;  
    integer mat;  
    char nomeProf[30];  
EXEC SQL END DECLARE SECTION;  
...  
printf("Informe matrícula: ");  
scanf("%i", &mat);  
EXEC SQL SELECT nome INTO :nomeProf  
           FROM Professores  
           WHERE matrícula = :mat;
```

o da
ção

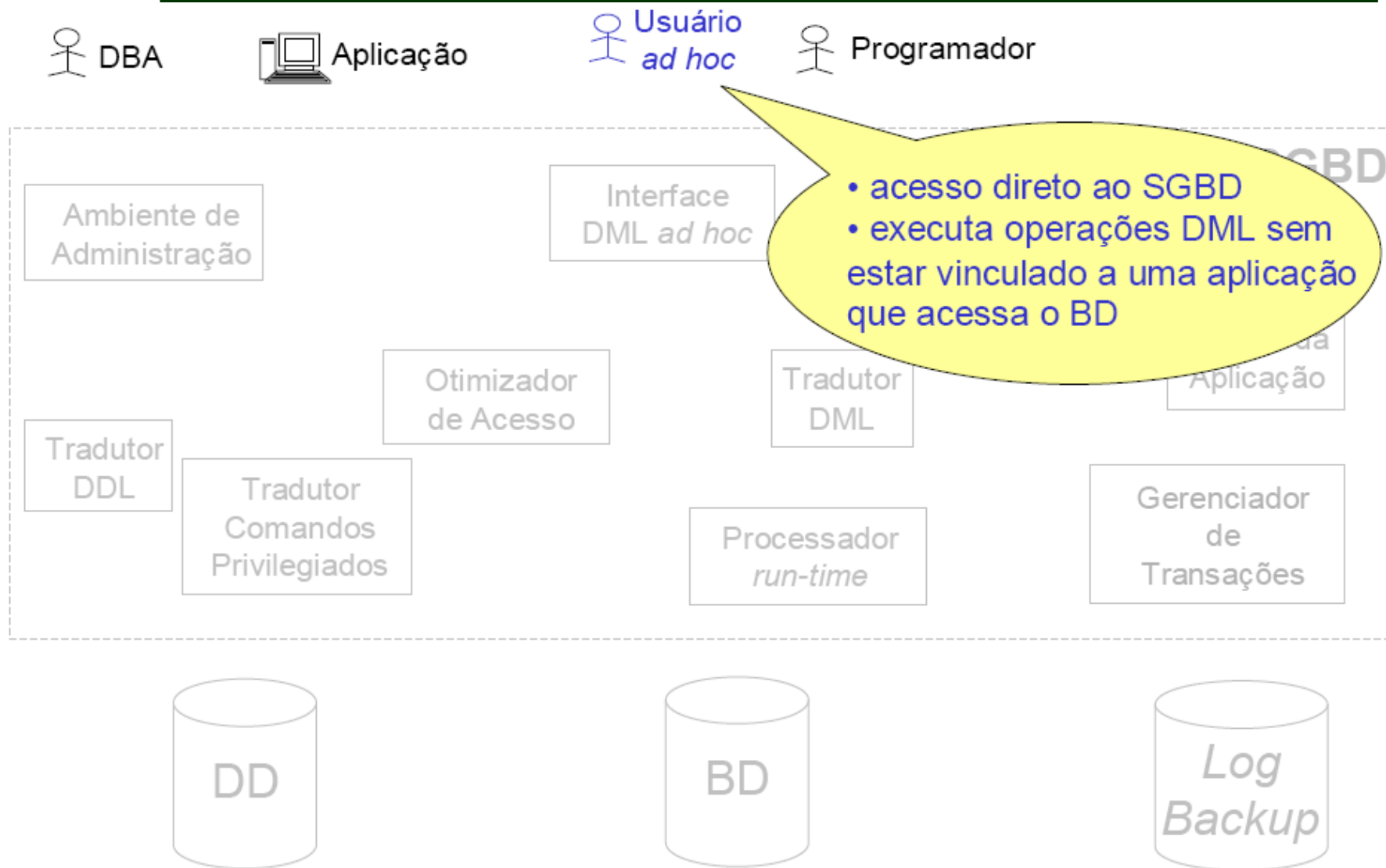
DD

Log
Backup

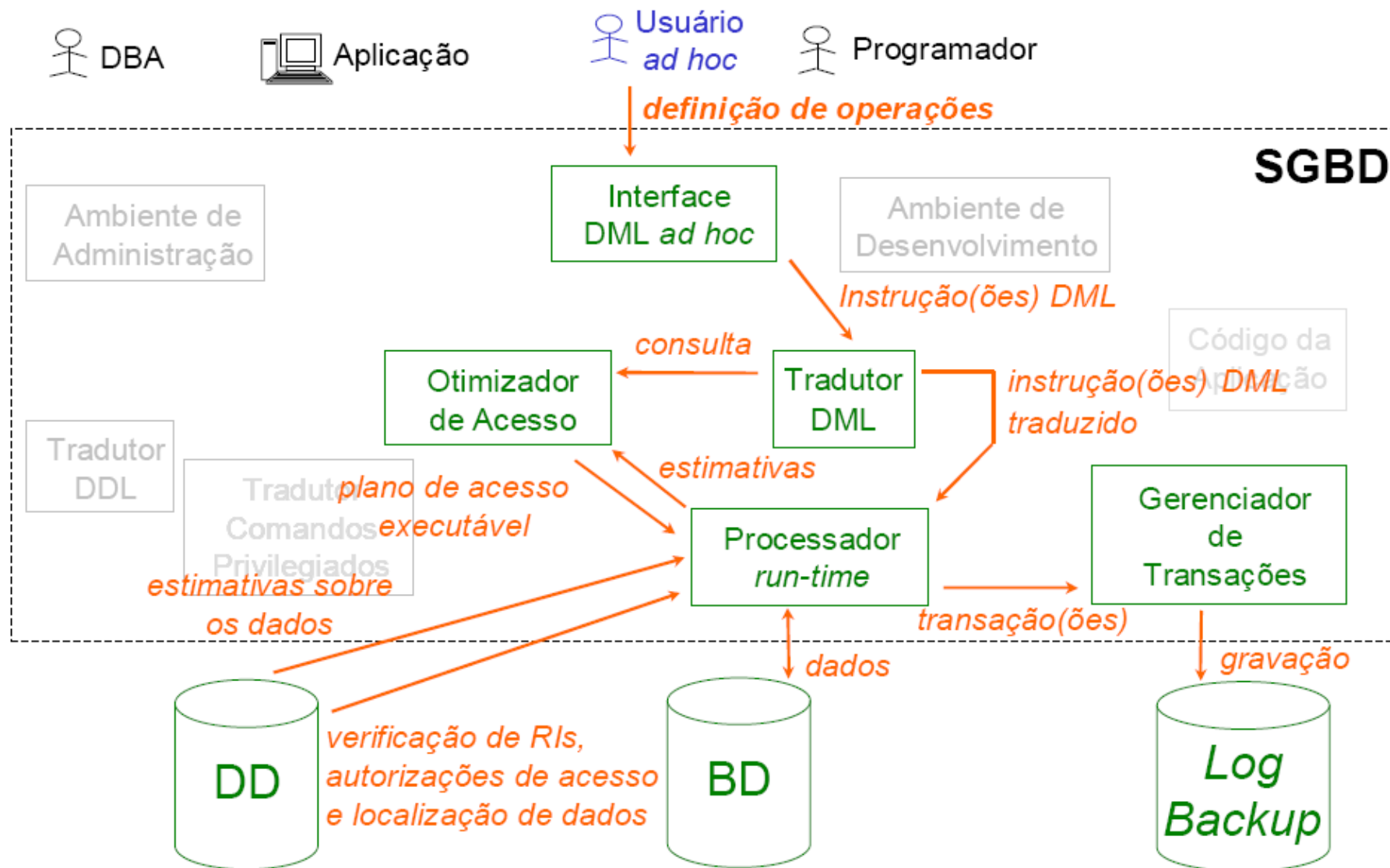
Arquitetura - Estrutura



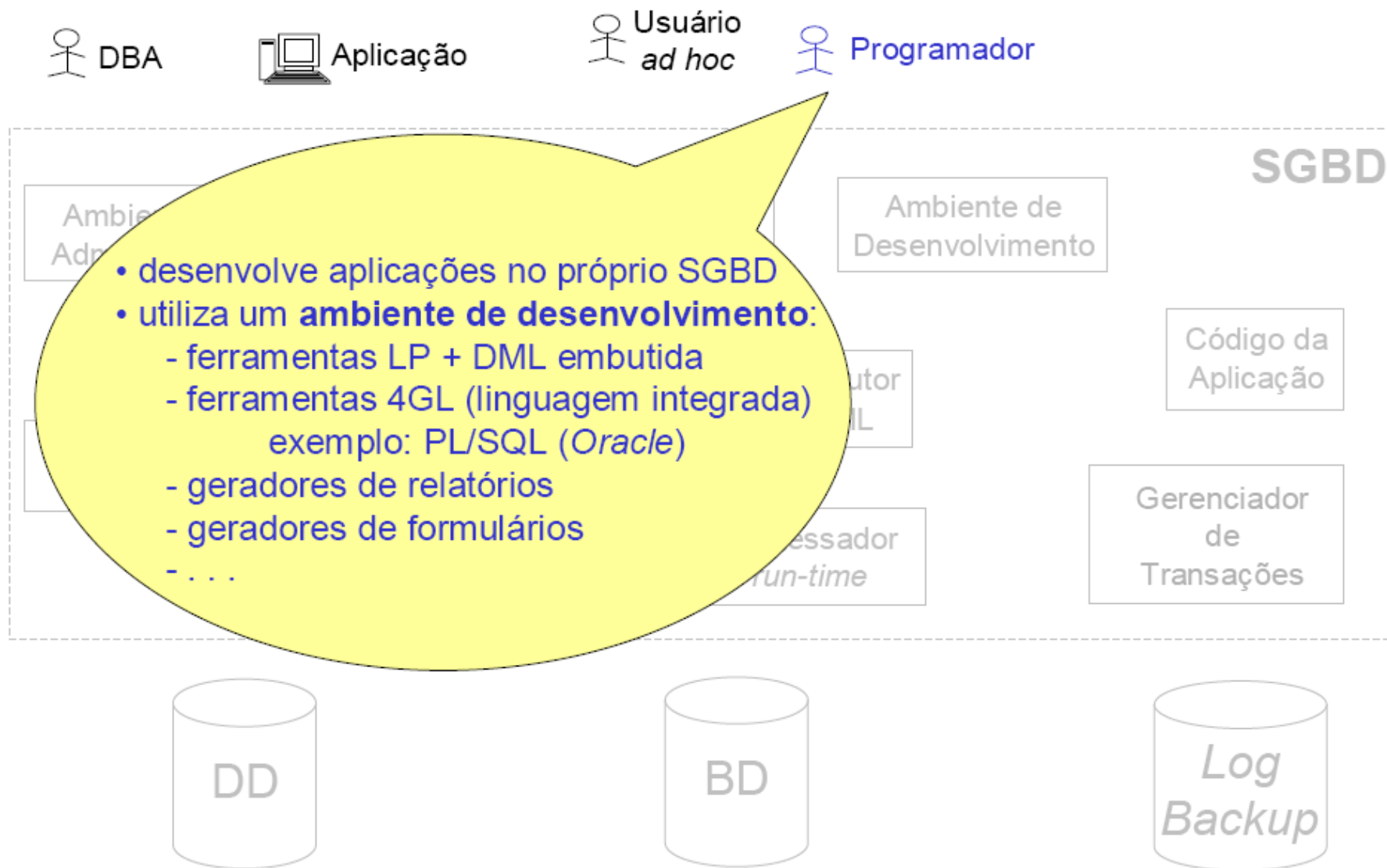
Arquitetura - Estrutura



Arquitetura - Estrutura



Arquitetura - Estrutura



Arquitetura - Estrutura

DBA



Aplicação

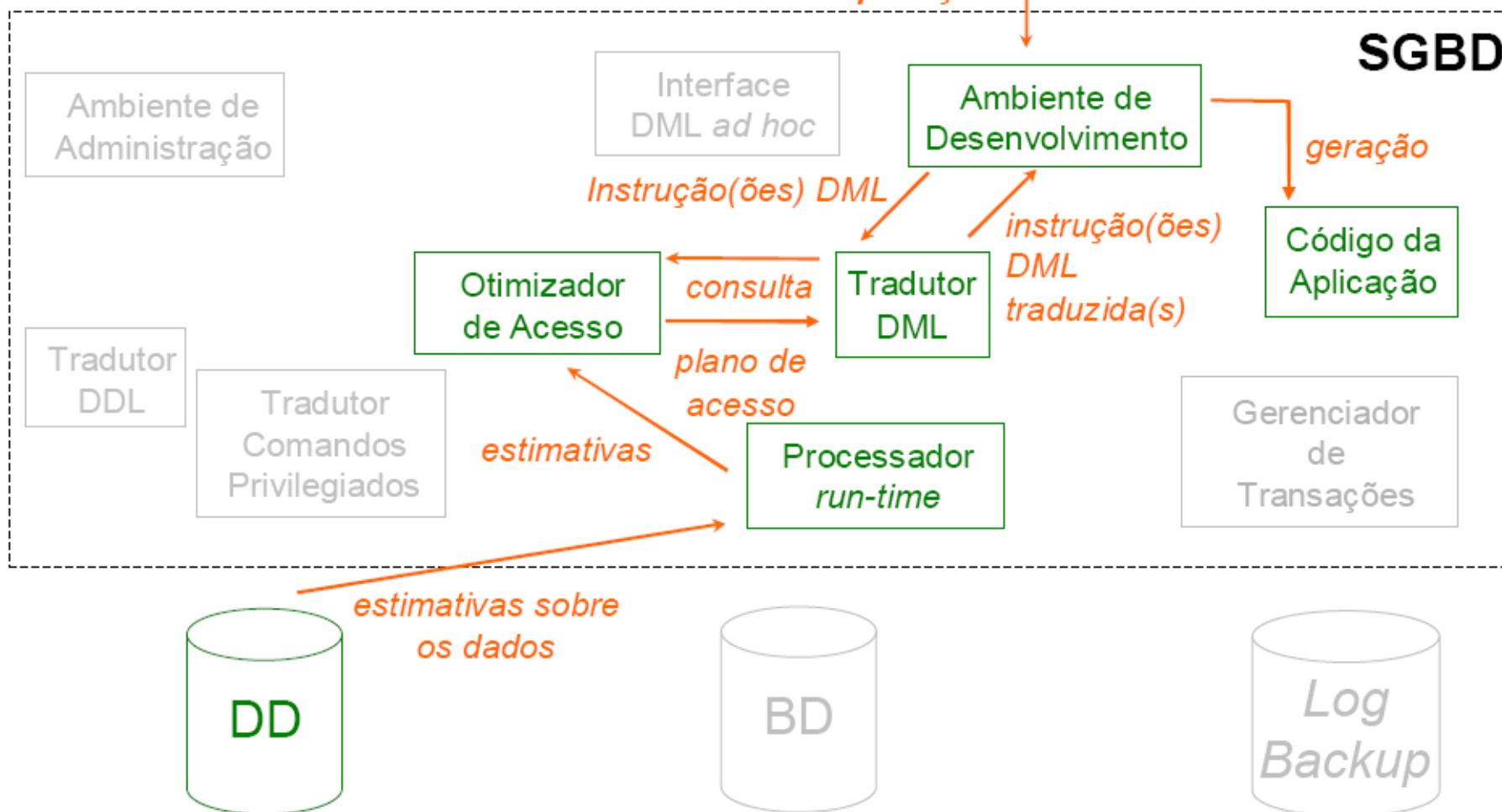


Usuário
ad hoc



Programador

Tarefa: desenvolvimento de aplicações



Plano de Consulta

Procura realizar consultas de forma mais eficiente considerando relacionamentos, predicados, volume de dados, índices etc.

PROFESSORES

(10 registros)

TURMAS

(20 registros)

SALAS

(10 registros)

3 salas no 3º andar

Ex: buscar dados dos professores que lecionam em turmas nas salas do 3º andar:

Alternativa 1: Professores → Turmas → Salas → filtrar 3º andar

Alternativa 2: filtrar 3º andar → salas → Turmas → Professores