# Arquitetura do Sistema de Bancos de Dados

# Arquitetura cliente/servidor

- A arquitetura mais comum em sistemas de bancos de dados é a arquitetura cliente/servidor de 2 ou 3 camadas.
- O módulo cliente corresponde às aplicações e usuários que acessam o banco.
- O SGBD atua como um servidor de consultas ou transações, sendo responsável pelo armazenamento, acesso e busca dos dados.

# Arquitetura cliente/servidor

- O servidor fornece uma API (*Application Programming Interface*) que permite que os clientes se autentiquem no banco e façam requisições.
- O padrão ODBC (*Open Database Conectivity*) define quais serviços serão ofertados e como deve ser feita a conexão entre cliente e servidor.
- Deste modo, basta trocar o *driver* do SGBD para trocar de um SGBD para outro.

# Arquitetura cliente/servidor

- Com o crescimento das aplicações *web*, foi inserida uma terceira camada intermediária.
- Esta camada é o servidor de aplicação, que recebe a requisição HTTP do cliente e estabelece ele mesmo a conexão com o SGBD (servidor de consultas).
- *Frameworks* modernos como Spring Boot (Java), Laravel (PHP), Django (Python) e Rail (Ruby) usam o princípio de *convention over configuration* para tornar mais abstrata ainda a comunicação com o banco de dados.
- Estes *frameworks* possuem mapeadores objeto-relacionais que permitem que o desenvolvedor use pouca SQL (*Structured Query Language*)[1].

[1] Linguagem padrão dos bancos de dados relacionais.

# Arquitetura cliente/servidor

- Salienta-se que a arquitetura cliente/servidor é a arquitetura vista de alto nível.
- Ela define as responsabilidades dos dois ou três agentes (camadas), mas não define como cada um funcionará.

# Modelos de dados

- O banco de dados[2] pode funcionar com diferentes camadas de abstração.
- Cada camada é conhecida como modelo de dados.

[2] Quando falamos banco de dados nos referimos a estrutura e organização dos dados e seus valores.

# Modelos de dados

São três os tipos de modelos de dados:

- conceituais;
- representativos;
- físicos.

# Modelos de dados conceituais

- Os modelos de dados conceituais são os de mais alto nível.
- O modelo Entidade-Relacionamento (ER) é o modelo conceitual mais utilizado ainda para definir um banco de dados neste nível.
- Entretanto, modelos baseados em classes, utilizando ou não linguagens de modelagem como a UML (*Unified Modeling Language*) possuem bastante espaço devido aos mapeadores objeto-relacionais.

# Modelos de dados representativos

- Os modelos representativos são modelos intermediários, pois oferecem conceitos que estão mais próximos da máquina, porém ainda são bastante inteligíveis aos usuários finais.
- O modelo representativo ainda dominante é o modelo relacional.

# Modelos de dados físicos

- Os modelos de dados físicos descrevem como os dados serão armazenados.
- Eles também descrevem como serão realizadas as indexações dos dados.

# Modelos de dados

- É importante lembrar que os modelos de dados definem o esquema do banco de dados, ou seja, a organização estrutural dos dados.
- Eles não definem o estado do banco, que corresponde ao conjunto de valores do banco de dados.
- Os modelos de dados são bastante importantes no projeto de bancos de dados, pois o modelo conceitual permite a discussão em alto nível com os *stakeholders*.
- Já o modelo representativo pode ser obtido através da derivação do modelo conceitual e permitem que os programadores implementem o esquema definido.

# Arquitetura de três esquemas

- Um comitê denominado ANSI/SPARC definiu um padrão arquitetural que até hoje serve de guia para os fabricantes de SGBDs.
- O padrão define 3 níveis:
  - nível externo;
  - nível conceitual;
  - nível interno.

# Arquitetura de três esquemas

- O nível externo é composto pelas visões do banco de dados que serão ofertadas aos usuários/aplicações.
- O nível conceitual é de nível intermediário, pois define o esquema conceitual, ou seja, é uma abstração sobre o nível interno, porém com alguns aspectos vinculados ao mesmo.
- O nível interno descreve a estrutura de armazenamento interno do banco de dados.

# Arquitetura de três esquemas

- Apesar de ser padrão, a implementação da arquitetura ANSI/SPARC varia de acordo com o SGBD.
- Em alguns SGBDs é possível até mesmo o uso de modelos de dados diferentes para os níveis externo e conceitual.
- Pode-se utilizar o modelo orientado a objetos para o nível externo e relacional para o nível conceitual, como faz o UDB da IBM.

# Arquitetura de três esquemas

- A segregação entre as camadas permite a independência lógica e física dos dados.
- Entretanto, a necessidade de mapeamento entre os níveis diminui a performance do banco de dados.

# Arquitetura de bancos de dados

- Apesar de parecidos, os conceitos de arquitetura cliente/servidor, modelos de dados e arquitetura ANSI/SPARC trabalham em níveis de abstração diferentes.
- A arquitetura cliente/servidor oferece uma visão macro, pois organiza a aplicação, os usuários e o SGBD, mas sem oferecer detalhes internos.
- Já os modelos de dados conceitual e representativo servem para definir especialmente o esquema do banco de dados (nível conceitual da arquitetura ANSI/SPARC).
- Eles também podem ser utilizados para definir o nível externo da arquitetura ANSI/SPARC.
- O modelo de dados físico e o nível interno da ANSI/SPARC se sobrepõem e geralmente são tratados pelo próprio SGBD.

# Linguagens dos sistemas de bancos de dados

- Em teoria, os sistemas de bancos de dados possuem diversas linguagens.
- Cada linguagem possui um propósito e atual em uma camada.

# Linguagens dos sistemas de bancos de dados

- A *Data Definition Language* (DDL) serve para definir o esquema do banco de dados.
- A *Storage Definition Language* (SDL) define como será o armazenamento físico dos dados.
- Já a *View Definition Language* (VDL) é usada para descrever as visões que serão oferecidas aos usuários/aplicações.
- Há também a *Data Manipulation Language* (DML), que não trabalha sobre o esquema do banco de dados, mas sobre os valores (estado).
- Os SGBDs relacionais atuais utilizam a *Structured Query Language* (SQL) para todos estes fins.

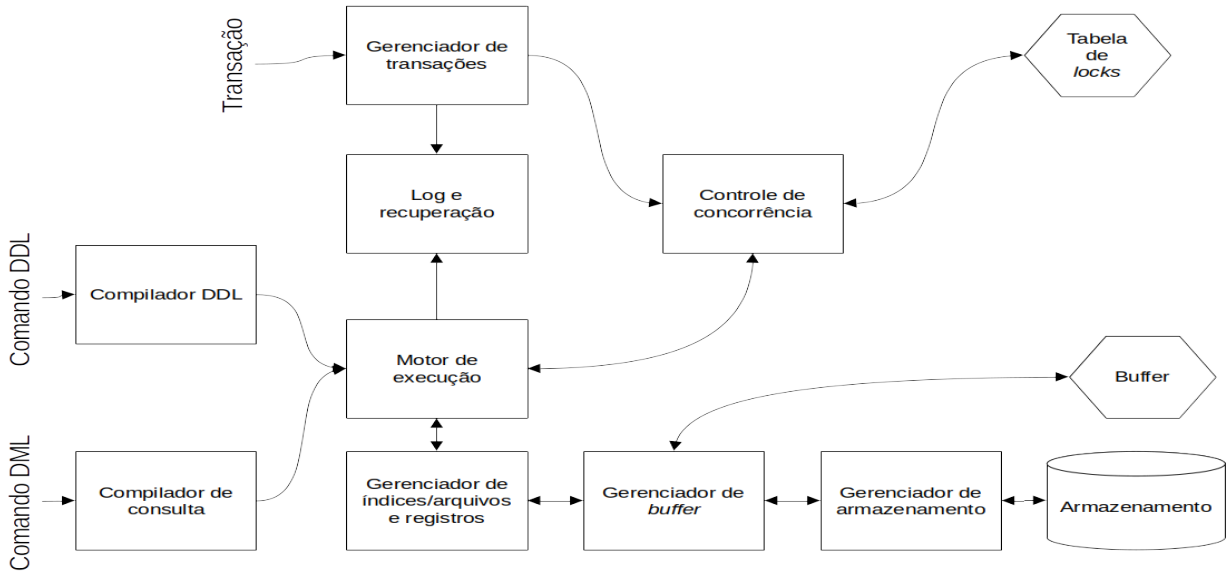
# Funcionamento do SGBD

- É impossível estabelecer um modo único de funcionamento para todos os SGBDs.
- Cada fabricante estabelece um arquiteturae pode usar diferentes paradigmas.
- Um SGBD relacional, como Oracle e Postgres, funciona de maneira diferente de um SGBD NoSQL como Redis ou Mongo[3].

[3] Que funcionam de maneira bastante diferente entre si.

# Funcionamento do SGBD

- A figura abaixo mostra ilustra o funcionamento de um SGBD genérico.
- As linhas sólidas representam fluxo de dados e de controle.
- Os retângulos representam componentes do sistema, enquanto os hexagonos estruturas de dados em memória.

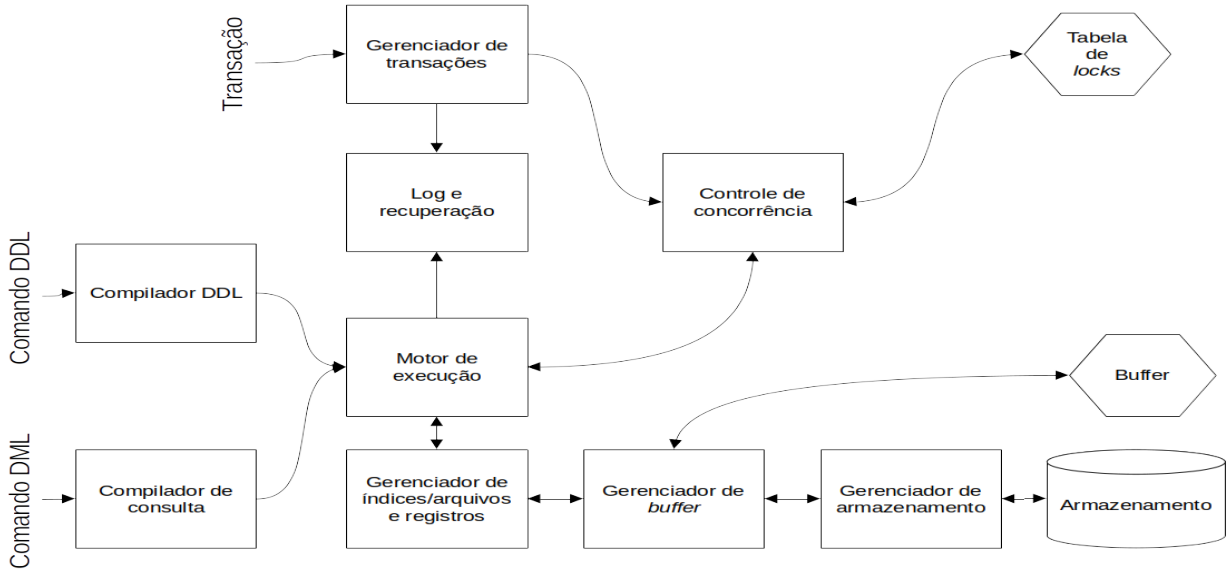



[4] Adaptado de Garcia-Molina (2009).

# Funcionamento do SGBD

O modelo supõe dois tipos de comando:

- comandos de modificação do esquema (comandos DDL);
- comandos sobre o estado do banco (comando DML).



# Funcionamento do SGBD

- O fluxo de execução é bastante similar .
- A diferença inicial é que os comando DDL poassam por um compilador DDL e os comandos DML por um compilador de consultas.
- Além disso, como os comandos DML modificam o estado da base de dados e comumente são executados em sequência com outros comandos DML, são agrupados em transações.

# Funcionamento do SGBD

- Uma transação é uma sequênciade comandos DML válidos com objetivo específico.
- É desejável que um SGBD garanta que as transações tenham propriedades ACID.

# Funcionamento do SGBD

- O comando DML (consulta) é otimizado pelo compilador de consulta com base na *metadata* e em estatísticas.
- O resultado da compilação é uma sequência de ações que devem ser realizadas pelo SGBD para executar a consulta.

# Funcionamento do SGBD

- O motor de execução requisita ao gerenciador de recursos (índices, arquivos e registros) os dados, registros ou tuplas de uma relação.
- As requisições por dados são passadas para o gerenciador de *buffer*, que verifica se os dados estão na memória.
- Caso negativo, o gerenciador de *buffer* trás os blocos de dados necessários do disco par a memória através do gerenciador de armazenamento.

# Funcionamento do SGBD

Para o processamento das transações são executados os seguintes passos:

- *logging*: para garantir as propriedades ACID, o SGBD armazena cada operação sobre o banco em um arquivo de *log*. Em caso de falha, o SGBD pode desfazer operações, garantindo que nenhuma transação será executada em parcialmente ou que o sistema não fique inconsistente;
- controle de concorrência: as transações devem ser isoladas. Por isso, usam-se *locks* (armazenados em um tabela na memória principal) para garantir que uma transação não sobrescreva um dado enquanto outras tentam lê-lo;
- resolução de impasses: o uso de *locks* pode gerar impasses. Deste modo, o gerenciador de transações pode abortar transações para que outras possam ser executadas.

# Referências


- Elmasri, Ramez e Navathe, Shamkant. Sistemas de banco de dados. 2011. 6ª ed. Pearson.
- Garcia-Molina, Hector; Ullman, Jeffrey D.; e Widom, Jennifer. Database Systems: The Complete Book. 2009. 2ed. Pearson.