Conceitos Gerais

Introdução

Este roteiro explora conceitos sobre SGBDs necessários para o entendimento, uso e projeto de Bancos de dados. Também explora a instalação de algumas ferramentas que serão utilizadas pela unidade curricular ao longo do semestre.

Arquitetura de Bancos de dados

A arquitetura dos SGBDs tem evoluído desde os primeiros sistemas monolíticos, nos quais todo o software GBD era um sistema altamente integrado, até os mais modernos, que têm um projeto modular, com arquitetura cliente/servidor. Geralmente em arquiteturas básicas Cliente/Servidor a funcionalidade do sistema é distribuída entre dois tipos de módulos:

***O módulo cliente:** que normalmente é projetado para executar em uma estação de trabalho ou computador pessoal oferecendo interação ao usuário através de interfaces amigáveis baseadas em menus e indicações visuais.*

***O módulo servidor:** responsável pelo armazenamento dos dados, acesso, pesquisa e outras funções.*

Modelos de dados, esquemas e instâncias

*Em bancos de dados é comum o uso da abordagem oferecendo algum nível de abstração de dados. Isto significa que estamos suprimindo detalhes da organização e armazenamento dos dados, destacando recursos essenciais para um melhor conhecimento desses dados. Um **modelo de dados** significa uma coleção de conceitos que podem ser utilizados para descrever a estrutura de um banco de dados e oferece meios para alcançar a abstração desejada. Existem os modelos de dados de **alto nível ou conceitual** e modelos de dados de **baixo nível ou físicos**.*

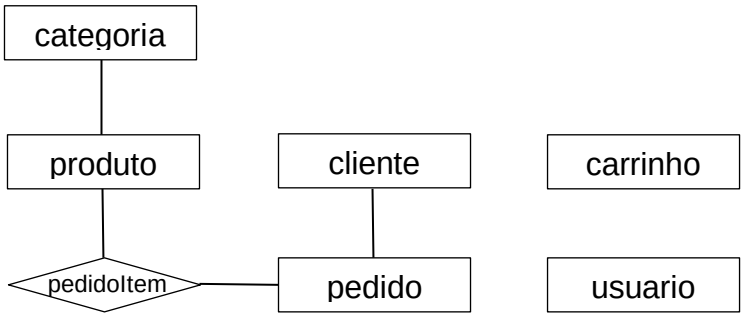
***Modelos de dados de alto nível ou conceitual:** oferecem conceitos que são próximos ao modelo como muitos usuários percebem os dados. Geralmente estes modelos utilizam conceitos como entidades, atributos e relacionamentos. Destes conceitos básicos podemos destacar:*

- ***Entidade:** representa um objeto ou conceito do mundo real, como um funcionário ou um projeto do minimundo (situação ampla para um projeto de banco de dados) que é descrito no banco de dados. Em modelagens simples, pode-se comparar a nomenclatura entidade à **tabela**.*
- ***Atributo:** representa alguma propriedade, como o nome ou salário do funcionário.*
- ***Relacionamento:** representa uma associação entre entidades, por exemplo, um relacionamento **trabalha-em** entre um funcionário e um projeto.*

***Modelos de dados de baixo nível ou físicos:** oferecem conceitos que descrevem os detalhes de como os dados são armazenados no computador, costumam ser voltados para especialistas de computadores. Descrevem o armazenamento dos dados como arquivos no computador, com informações no formato de registro, ordenações de registro e caminhos de acesso.*

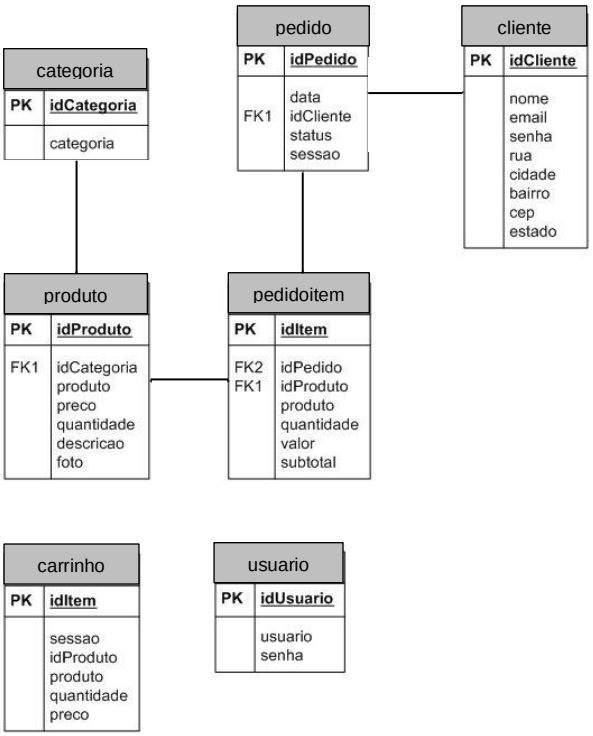
***Modelo Conceitual**:*

O modelo conceitual é um diagrama em blocos que demonstra todas as relações entre as entidades, suas especializações, seus atributos e auto-relações.



***Modelo Lógico**:*

O modelo lógico mostra as ligações entre as tabelas de banco de dados, as chaves primárias, os componentes de cada uma, etc.



Modelo Físico:

Inclui a análise das características e recursos necessários para armazenamento e manipulação das estruturas de dados (estrutura de armazenamento, endereçamento, acesso e alocação física), sendo uma sequência de comandos executados em SQL a fim de criar as tabelas, estruturas e ligações projetadas até então e finalmente criar o banco de dados.

Exemplo de Modelo Físico:

```
CREATE TABLE IF NOT EXISTS `carrinho` (
  `idItem` int(11) NOT NULL AUTO_INCREMENT,
  `sessao` int(11) DEFAULT NULL,
  `idProduto` int(11) DEFAULT NULL,
  `produto` varchar(80) DEFAULT NULL,
  `quantidade` int(11) DEFAULT NULL,
  `preco` decimal(12,2) DEFAULT NULL,
  PRIMARY KEY (`idItem`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=40 ;

CREATE TABLE IF NOT EXISTS `categoria` (
  `idCategoria` int(11) NOT NULL,
```

```
  `categoria` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`idCategoria`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `cliente` (
  `idCliente` int(11) NOT NULL AUTO_INCREMENT,
  `nome` varchar(80) DEFAULT NULL,
  `email` varchar(80) DEFAULT NULL,
  `senha` varchar(50) DEFAULT NULL,
  `rua` varchar(80) DEFAULT NULL,
  `cidade` varchar(80) DEFAULT NULL,
  `bairro` varchar(40) DEFAULT NULL,
  `cep` varchar(10) DEFAULT NULL,
  `estado` varchar(2) DEFAULT NULL,
  PRIMARY KEY (`idCliente`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=4 ;

CREATE TABLE IF NOT EXISTS `pedido` (
  `idPedido` int(11) NOT NULL AUTO_INCREMENT,
  `data` date DEFAULT NULL,
  `idCliente` int(11) DEFAULT NULL,
  `status` varchar(80) DEFAULT NULL,
  `sessao` int(11) DEFAULT NULL,
  PRIMARY KEY (`idPedido`),
  KEY `fkcli` (`idCliente`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=9 ;

CREATE TABLE IF NOT EXISTS `pedidoItem` (
  `idItem` int(11) NOT NULL AUTO_INCREMENT,
  `idPedido` int(11) DEFAULT NULL,
  `idProduto` int(11) DEFAULT NULL,
  `produto` varchar(100) DEFAULT NULL,
  `quantidade` int(11) DEFAULT NULL,
  `valor` decimal(12,2) DEFAULT NULL,
  `subtotal` decimal(12,2) DEFAULT NULL,
  PRIMARY KEY (`idItem`),
  KEY `fkp` (`idPedido`),
  KEY `fkpr` (`idProduto`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=22 ;

CREATE TABLE IF NOT EXISTS `produto` (
  `idProduto` int(11) NOT NULL AUTO_INCREMENT,
  `idCategoria` int(11) DEFAULT NULL,
  `produto` varchar(80) DEFAULT NULL,
  `preco` decimal(12,2) DEFAULT NULL,
  `quantidade` int(11) DEFAULT NULL,
  `descricao` varchar(80) DEFAULT NULL,
  `foto` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`idProduto`),
  KEY `fkcat2` (`idCategoria`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=10 ;

CREATE TABLE IF NOT EXISTS `usuario` (
  `idUsuario` int(11) NOT NULL AUTO_INCREMENT,
  `usuario` varchar(100) NOT NULL,
  `senha` varchar(100) NOT NULL,
  `status` varchar(100) NOT NULL,
  PRIMARY KEY (`idUsuario`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=21 ;

ALTER TABLE `pedido`
  ADD CONSTRAINT `fkcli` FOREIGN KEY (`idCliente`) REFERENCES `cliente` (`idCliente`);

ALTER TABLE `pedidoItem`
  ADD CONSTRAINT `fkp` FOREIGN KEY (`idPedido`) REFERENCES `pedido` (`idPedido`),
  ADD CONSTRAINT `fkpr` FOREIGN KEY (`idProduto`) REFERENCES `produto` (`idProduto`);

ALTER TABLE `produto`
  ADD CONSTRAINT `fkcat2` FOREIGN KEY (`idCategoria`) REFERENCES `categoria` (`idCategoria`);
```

Esquemas

Em bancos de dados é importante distinguir entre a descrição do banco de dados e o próprio banco de dados. A descrição é o chamado esquema de banco de dados (squema database), que é especificado durante o projeto de banco de dados e não se espera que mude com frequência. Em alguns softwares utilizados na modelagem de bancos de dados costuma utilizar a nomenclatura squema para definir a descrição de todos os elementos que compõem o banco de dados.

*Os dados em um banco de dados armazenados em um determinado momento no tempo são chamados de **estado** ou **instante do banco de dados** ou ainda são chamados de conjunto atual de **instâncias** ou **ocorrências** no banco de dados. O estado do banco de dados é alterado pelo simples fato de alteração das informações constantes, como a inserção de um novo registro ou alteração de uma informação de um registro. O SGBD é parcialmente responsável para garantir o estado de um banco de dados como válido, ou seja, que satisfaça as **descrições e restrições** especificadas no esquema, também chamados de **Metadados**. O esquema de um banco de dados às vezes é chamado de **intenção** e o estado de **extensão** do esquema.*

Linguagens e interfaces do banco de dados

*Um SGBD necessita oferecer linguagens e interfaces apropriadas para cada categoria de usuário. Quando um projeto de banco de dados é finalizado e um SGBD é escolhido para implementá-lo, o primeiro passo é especificar esquemas conceituais e internos para o banco de dados e quaisquer mapeamentos entre os dois. Em muitos casos não há uma separação estrita entre níveis, então é utilizada uma linguagem chamada de **Linguagem de Definição de Dados** (**Data Definition Language – DDL**), e é utilizada pelo DBA e projetistas de banco de dados para definir os dois esquemas.*

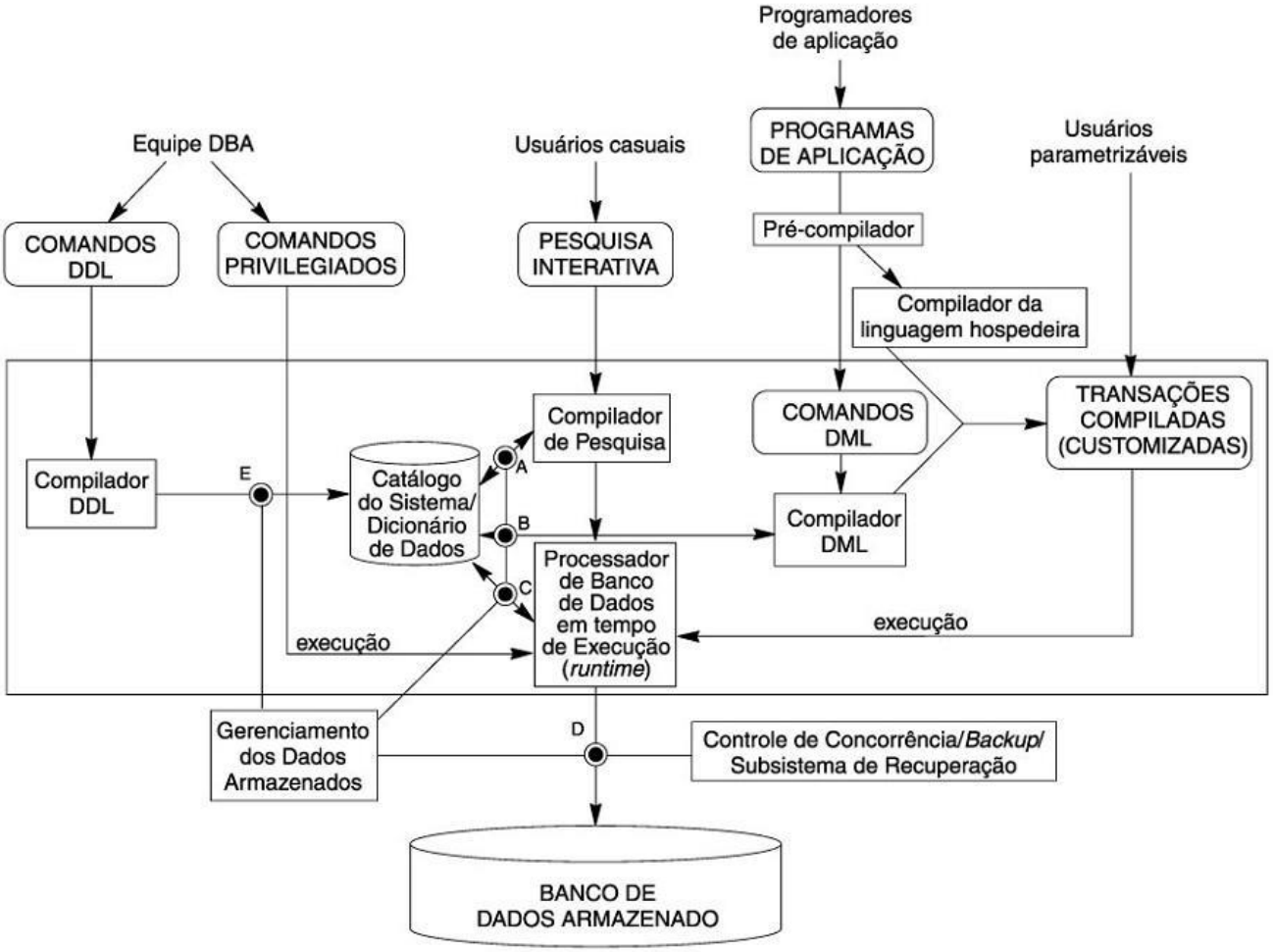
*Em SGBDs que mantêm a separação entre o nível conceitual e interno, a DDL é utilizada apenas para especificar o esquema conceitual, neste caso, o esquema interno do SGBD é definido com uso da **Linguagem de Definição de Armazenamento (Storage Definition Language – SDL)**. Na maioria dos bancos de dados relacionais **não existe** linguagem que utilize SDL. Na realidade os SGBDs relacionais acabam utilizando totalmente a DDL para definir quaisquer tipos de esquemas, e fazem uso da linguagem **SQL** para a realização destas tarefas.*

*Após a definição de bancos de dados é necessária a utilização de algo para poder manipular os seus dados, neste caso entra em ação a **Linguagem de Manipulação de dados** (**Data Manipulation Language – DML**) para estas finalidades, a manipulação inclui tarefas de recuperação, inserção, exclusão e modificação dos dados.*

*O maior exemplo de linguagem de banco de dados é a linguagem de banco de dados **SQL**, que representa uma combinação de **DDL**, **DML** e **VDL**, bem como as instruções para especificação de restrições e demais funcionalidades inerentes de bancos de dados.*

O ambiente do sistema de banco de dados

Um SGBD consiste em um software complexo, e para poder explorar seus componentes é necessário representa-lo em um diagrama estendido capaz de separar as suas funcionalidades e detalhes de sua implementação.



O banco de dados e o catálogo do SGBD geralmente são armazenados em disco. O seu acesso é controlado pelo ***sistema operacional (SO)*** *que escalona a leitura/escrita em disco.*

Um ***gerenciador de dados armazenados*** *controla o acesso às informações do SGBD, tanto para o catálogo quanto para o banco de dados.*

O ***compilador DDL*** *processa as definições de esquema especificadas e armazena as descrições dos esquemas no catálogo do SGBD.*

Uma ***interface de consulta interativa*** *é utilizada por usuários casuais do banco de dados, esta interação é realizada através de consultas analisadas e validadas por um* ***compilador de consulta,*** *e um* ***otimizador de consulta*** *é responsável, como o próprio nome diz, por otimizar a consulta realizada através de procedimentos internos e direcionados ao* ***processador de dados em tempo de execução****, não visíveis ao usuário do banco de dados.*

O ***pré-compilador*** *é responsável por extrair comandos DML de um programa de aplicação escrito em linguagem de programação hospedeira (java, C++, PHP), estes comandos são enviados ao compilador DML para serem compilados em código objeto para o acesso ao banco de dados.*

O restante do programa é enviado ao compilador da linguagem hospedeira, os códigos objeto para comandos DML são ligados formando uma transação programada. Cada execução no banco é considerada uma transação programada.

*Na parte inferior é encontrado o **processador de banco de dados em tempo de execução**, responsável por executar comandos privilegiados, planos de consultas executáveis e transações programadas com parâmetros em tempo e execução, trabalhando em conjunto com o catálogo do sistema.*

Ferramentas, ambientes de aplicação e facilidades de comunicações

*Normalmente, para um projeto de banco de dados é necessário o uso de ferramentas que facilitem o seu planejamento, manutenção e implementação posterior. São utilizadas normalmente diversas ferramentas **CASE** (utilizadas na engenharia de software) na fase de projeto dos sistemas de bancos de dados. O SGBD também precisa realizar a interface com o software de comunicações, cuja função é permitir que os usuários em locais remotos do sistema de banco de dados acessem o banco de dados por meio de terminais de computador, estações de trabalho ou computadores pessoais.*

*Alguns SGBD's fornecem ferramentas nativas para realizar o seu controle e/ou manutenção de configurações e informações de seus bancos de dados, mas existem também algumas ferramentas que podem auxiliar no projeto e manutenção de modelos de bancos de dados como por exemplo a ferramenta **Workbench** (http://www.MySQL.com/products/workbench).*

Classificação de Sistemas gerenciadores de bancos de dados

*Vários critérios são normalmente utilizados para classificar os SGBDs; O primeiro é o modelo de dados no qual o SGBD é baseado. O principal modelo de dados utilizado comercialmente é o **modelo de dados relacional**. O **modelo de dados de objeto** foi implementado em alguns sistemas comerciais, mas não tem seu uso generalizado.*

*Os SGBDs relacionais estão evoluindo continuamente e incorporando muitos dos conceitos que foram desenvolvidos nos bancos de dados de objeto, criando uma nova classe chamada de **SGBDs objeto-relacional**. Desta forma, o **primeiro critério** é de classificar bancos de dados pelo seu modelo de dados: relacionais, objeto, objeto-relacional, entre outros.*

*O **segundo critério** utilizado é o número de usuários do banco: monousuário e multiusuário.*

*O **terceiro critério** é o número de locais sobre os quais o banco de dados está distribuído:*

- *Centralizado: se os dados estiverem armazenados em um único computador;*
- *Distribuído: onde pode ter o banco de dados real e o software de SGBD distribuídos por vários locais, conectados por uma rede de computadores. Ainda assim, divididos em:*
    - *Homogêneos: utilizando o mesmo software SGBD em todos os locais;*
    - *Heterogêneos: utilizado um software SGBD diferente em cada local.*

*O **quarto critério** é o custo: visto que existem SGBDs de código aberto e proprietários, porém sempre que se desejar funções avançadas para administração ou manutenção de grandes quantidades de dados, processamento paralelo, replicação, distribuição, capacidade móvel, etc., são parâmetros que sempre levam em consideração a adoção de bancos de dados profissionais à um custo pelas funcionalidades.*

Referências

- ***Capítulo 2**: Elmasri, Navathe – Sistemas de banco de dados 6ª Edição;*
- *Dia, disponível em http://live.gnome.org/Dia*
- *brModelo, disponível em http://www.sis4.com/brModelo*
- *Draw.io https://app.diagrams.net/*
- *Material disponibilizado em aula*