



**e-Tec Brasil**  
*Escola Técnica Aberta do Brasil*

# Técnico em Informática para internet

Sistemas de Gerenciamento de Banco de Dados

Prof. Matheus Franco



São João da Boa Vista -SP

2010

Presidência da República Federativa do Brasil  
Ministério da Educação  
Secretaria de Educação a Distância

Este Caderno foi elaborado em parceria entre o Instituto Federal de São Paulo e o Sistema Escola Técnica Aberta do Brasil – e-Tec Brasil.

Equipe de Elaboração  
Instituto Federal de São Paulo

Projeto Gráfico  
Eduardo Meneses e Fábio Brumana

Coordenação  
Instituto Federal de São Paulo

Diagramação  
Juliana Ayres

Professor-autor  
Matheus Franco

Revisão  
Prof. Gustavo Aurelio Prieto e  
Elizabeth Vanni

#### Ficha catalográfica

Projeto gráfico / [equipe de elaboração: Instituto Federal de São Paulo . –São Paulo : IFSP, 2010.

88 p. : il., tabs.

Inclui referências

Desenvolvido para o Sistema Escola Técnica Aberta do Brasil.

ISBN: XXXXXXXXXXXX-XX

I. Instituto Federal de São Paulo.

II. Escola Técnica Aberta do Brasil.

# Apresentação e-Tec Brasil

Amigo(a) estudante!

O Ministério da Educação vem desenvolvendo Políticas e Programas para expansão da Educação Básica e do Ensino Superior no País. Um dos caminhos encontrados para que essa expansão se efetive com maior rapidez e eficiência é a modalidade a distância. No mundo inteiro são milhões os estudantes que frequentam cursos a distância. Aqui no Brasil, são mais de 300 mil os matriculados em cursos regulares de Ensino Médio e Superior a distância, oferecidos por instituições públicas e privadas de ensino.

Em 2005, o MEC implantou o Sistema Universidade Aberta do Brasil (UAB), hoje, consolidado como o maior programa nacional de formação de professores, em nível superior.

Para expansão e melhoria da educação profissional e fortalecimento do Ensino Médio, o MEC está implementando o Programa Escola Técnica Aberta do Brasil (e-TecBrasil). Espera, assim, oferecer aos jovens das periferias dos grandes centros urbanos dos municípios do interior do País oportunidades para maior escolaridade, melhores condições de inserção no mundo do trabalho e, dessa forma, com elevado potencial para o desenvolvimento produtivo regional.

O e-Tec é resultado de uma parceria entre a Secretaria de Educação Profissional e Tecnológica (SETEC), a Secretaria de Educação a Distância (SED) do Ministério da Educação, as universidades e escolas técnicas estaduais e federais.

O Programa apóia a oferta de cursos técnicos de nível médio por parte das escolas públicas de educação profissional federais, estaduais, municipais e, por outro lado, a adequação da infra-estrutura de escolas públicas estaduais e municipais.

Do primeiro Edital do e-Tec Brasil participaram 430 proponentes de adequação de escolas e 74 instituições de ensino técnico, as quais propuseram 147 cursos técnicos de nível médio, abrangendo 14 áreas profissionais. O resultado desse Edital contemplou 193 escolas em 20 unidades federa-

tivas. A perspectiva do Programa é que sejam ofertadas 10.000 vagas, em 250 polos, até 2010.

Assim, a modalidade de Educação a Distância oferece nova interface para a mais expressiva expansão da rede federal de educação tecnológica dos últimos anos: a construção dos novos centros federais (CEFETs), a organização dos Institutos Federais de Educação Tecnológica (IFETs) e de seus campi.

O Programa e-Tec Brasil vai sendo desenhado na construção coletiva e participação ativa nas ações de democratização e expansão da educação profissional no País, valendo-se dos pilares da educação a distância, sustentados pela formação continuada de professores e pela utilização dos recursos tecnológicos disponíveis.

A equipe que coordena o Programa e-Tec Brasil lhe deseja sucesso na sua formação profissional e na sua caminhada no curso a distância em que está matriculado(a).

Brasília, Ministério da Educação – setembro de 2008.

# Sumário

Palavra do Professor Autor	7
Indicação de Ícone	9
Apresentação da Disciplina	11
Aula 1 - Conceitos Fundamentais	13
Aula 2 - Modelos e Modelagem de dados	27
Aula 3 - O modelo de Dados Relacional	45
Aula 4 - Implementação do Banco de Dados	59
Aula 5 - Linguagem de Manipulação e Consulta de Dados	73
Aula 6 - Projeto Final da Dsiciplina	85
Referências	87



# Palavra do professor-autor

Caro aluno!

Antes de tudo vou me apresentar, sou o professor Matheus Franco, mestre em Sistemas de Produção, especialista em Redes e Aplicações Distribuídas pela PUC Minas, bacharel em Ciência da Computação pela Universidade de Alfenas. Atualmente sou Professor do Ensino Técnico e Tecnológico do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo no Campus de São João da Boa Vista. Possuo experiência em Ciência da Computação, atuando principalmente nas áreas de: redes de computadores, redes neurais, banco de dados, programação SQL, sistemas distribuídos e de alta disponibilidade (Cluster's).

Desejo que tenha um ótimo aprendizado!

Um grande abraço!

Prof. Matheus Franco



# Indicação de Ícone

Os ícones são elementos gráficos utilizados para ampliar as formas de linguagem e facilitar a organização e a leitura hipertextual



Atenção: indica pontos de maior relevância no texto.



Saiba mais: oferece novas informações que enriquecem o assunto ou “curiosidades” e notícias recentes relacionadas ao tema estudado.



Glossário: indica a definição de um termo, palavra ou expressão utilizada no texto.



Mídias Integradas: remete o tema para outras fontes: livros, filmes, músicas, sites, programas de TV.



Atividades de aprendizagem: apresenta atividades em diferentes níveis de aprendizagem para que o estudante possa realizá-las e conferir o seu domínio do tema estudado.



# Apresentação da Disciplina

Esta disciplina apresenta uma visão introdutória sobre banco de dados. A ênfase será no modelo relacional, atualmente o modelo mais

utilizado na indústria. Abordaremos aspectos teóricos, tais como:

processamento de consultas e cardinalidade.

Durante a disciplina, será desenvolvido um trabalho que utilizará os tópicos estudados proporcionando experiência na utilização dos conceitos estudados. Analisaremos também os principais Sistemas de Gerenciamento de Banco de Dados existentes no Mercado.

Ao final desta aula, você aluno deverá ter assimilado os conceitos fundamentais sobre Banco de Dados e de Sistemas de Gerenciamento de Banco de Dados, assim como os aspectos de acesso, integridade, modelagem e implementação de um banco de dados relacional.

## Projeto instrucional

Disciplina: Sistemas de Gerenciamento do Banco de Dados

(carga horária: 60h).

Ementa: Idem plano de ensino.

Tabela 1

Aula	Objetivos	Materiais	Carga Horária
1. Conceitos Fundamentais	Apresentar os conceitos básicos sobre banco de dados; Apresentar as principais características de um Sistema de Gerenciamento de Banco de Dados.	Caderno	10 horas
2. Modelos e modelagem de dados	O Objetivo desta aula é apresentar os modelos de dados existentes, bem como o processo de modelagem de dados utilizando o Modelo Entidade Relacionamento.	Caderno, Computador, Software Livre DIA.	10 horas
3. Modelo Relacional	Esta aula tem como objetivo apresentar os principais conceitos do modelo de dados mais utilizado atualmente para implementação de um banco de dados, o Modelo Relacional que representa um banco de dados em um conjunto de relações.	Caderno, Computador, Software Livre DIA.	10 horas
4. Implementação do Banco de Dados	Apresentar os conceitos necessários para implementação de um banco de dados em um Sistema de Gerenciamento de Banco de Dados que implemente o modelo relacional.	Caderno, Computador, Software Livre DIA, Software livre MySQL 5.	10 horas
5. Linguagem de Manipulação e consulta de dados	O Objetivo desta aula é apresentar as instruções necessárias para a manipulação de dados utilizando o ambiente do SGBD MySQL.	Caderno, Computador, Software Livre DIA, Software livre MySQL 5.	10 horas
6. Projeto Final da Disciplina	O Objetivo desta aula é realizar a definição do projeto final da disciplina, bem como orientar a elaboração do projeto.	Caderno, Computador, Software Livre DIA, Software livre MySQL 5.	10 horas

# Aula I - Conceitos Fundamentais

## Objetivos da aula

- Apresentar os conceitos básicos sobre banco de dados;
- Apresentar as principais características de um Sistema de Gerenciamento de Banco de Dados.

### 1.1 Introdução

Todos nós, em nosso dia a dia, temos a necessidade de armazenar e recuperar dados. Cadernos de endereços, listas de telefones, dados

financeiros, receitas, enfim, estamos sempre lançando mão de memórias auxiliares. O ideal seria que estas memórias pudessem ser seguras, confiáveis e estivessem disponíveis quando precisássemos dela. Um dado guardado, mas que não sabemos como recuperar é praticamente tão inútil quanto se não existisse. Assim, temos uma grande necessidade de armazenar informações que não se encontram isoladas, como por exemplo, as fichas de matrícula de um aluno que contém informações diversas sobre o mesmo, conforme a figura abaixo.

 INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO	FICHA INDIVIDUAL DO ALUNO	
NOME:		
ENDEREÇO:		
RG:		
PRONTUÁRIO:		
CURSO:		
ANO INGRESSO:		

Figura 1.1 : Ficha de um aluno

Além de uma forma adequada para definir o armazenamento destas informações, os usuários desejam realizar operações sobre esta coleção de dados, tais como: adicionar novos dados, consultar um determinado subconjunto de dados, atualizar ou modificar a estrutura dos dados e eliminar informações não mais necessárias.

Uma solução para este problema foi apresentada com o advento da tecnologia em Bancos de Dados (BD, Database em Inglês). Um banco de dados é uma coleção de dados relacionados. Os dados são fatos que podem ser gravados e possuem um significado implícito. Por exemplo, considere o exemplo anterior da ficha com dados de um aluno ou uma lista telefônica. Esses dados podem ser armazenados em uma ficha, agenda ou no computador por meio de aplicativos como o Microsoft® Excel ou Access. Essas informações têm um significado implícito, portanto formam um banco de dados.

Algumas definições mais precisas de Banco de Dados podem ser (ELMASRI & NAVATHE, 2005):

- a) Um Banco de Dados é uma coleção logicamente coerente de dados com um determinado significado inherente. Isto significa que um conjunto aleatório de dados não pode ser considerada um Banco de Dados.
- b) Um Banco de Dados é projetado, construído e composto por um conjunto de dados para um propósito específico (como no nosso exemplo acima para armazenar um conjunto de informações de alunos). Existe um grupo de usuários ou algumas aplicações pré-concebidas onde estes dados serão utilizados.
- c) Um Banco de Dados representa aspectos de uma parte restrita do mundo real, denominado de mini-mundo. Alterações que ocorra no mini-mundo são refletidas em todo o Banco de Dados.

Além do conceito de BD, existe o conceito de Sistema de Gerenciamento de Bancos de Dados (SGBD), que é uma coleção de programas que permite ao usuário definir, construir e manipular um Banco de Dados para as mais diversas aplicações. O objetivo principal de um sistema de banco de dados é possibilitar um ambiente que seja adequado e eficiente para uso na recuperação e armazenamento de informações.

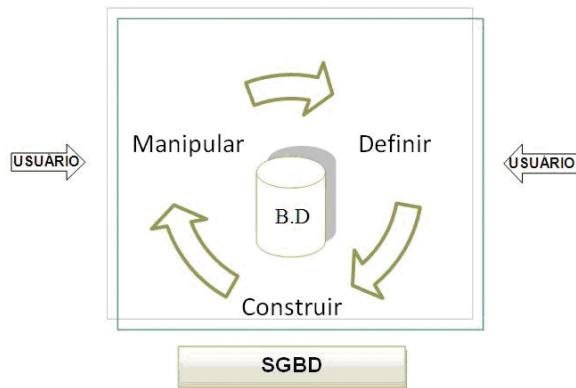


Figura 1.2 – Sistema de Gerenciamento de Banco de Dados

Banco de Dados e seu software são juntos denominados de Sistema de Bancos de Dados (SBD).

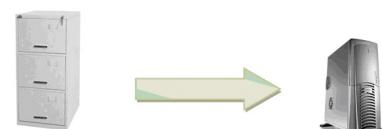
Imagine um armário de aço, com várias gavetas, em cada gaveta contém alguma informação (como a ficha do aluno) que estão agrupadas de acordo com seu tipo. O armário no caso é forma de gerenciamento dos dados ali contidos, lá podemos: inserir, excluir, selecionar ou alterar algum documento que ali contenha. Neste primeiro momento podemos pensar que um banco de dados computacional consiste em “levar” os dados deste armário de aço para o computador, porém seguiremos algumas regras para que o armazenamento seja mais eficiente.



## 1.2. HISTÓRIA

Os primeiros SBD's foram lançados no mercado no final da década de 60 e eram conceitualmente muito simples, de acordo com as necessidades das aplicações da época. No inicio as empresas que

impulsionaram este segmento foram a IBM, ORACLE e SYBASE. Assim como outras tecnologias na computação, os fundamentos de bancos de dados relacionais nasceram na empresa IBM, nas décadas de 1960 e 1970, por meio de pesquisas de funções de automação de escritório. Foi durante um período da história nas quais as empresas verificaram que era necessário雇用 many pessoas, além do alto custo para fazer trabalhos, tais como: armazenar e organizar seus arquivos. Por este motivo, eram importantes os esforços e investimentos em pesquisa para obter-se um meio mais barato e eficiente de armazenamento de dados.





Um banco de dados relacional organiza seus dados em relações. Cada relação pode ser vista como uma tabela, onde cada coluna corresponde a atributos da relação e as linhas correspondem às tuplas ou elementos da relação. Em uma nomenclatura mais próxima àquela de sistemas de arquivos, muitas vezes as tuplas são denominadas registros e os atributos, campos.

Em 1970 a IBM publicou o primeiro trabalho sobre bancos de dados relacionais. Este trabalho tratava sobre o uso de cálculo e álgebra relacional para que usuários não técnicos pudessem manipular grande quantidade de informações.

Devido à natureza técnica e a relativa complicaçāo matemática, o significado e proposição do trabalho não foram prontamente realizados. Assim, a IBM montou um grupo de pesquisa conhecido como System R.

O projeto do Sistema R tinha como proposta a criação de um sistema de banco de dados relacional o qual eventualmente se tornaria um produto. Os primeiros protótipos foram utilizados por várias organizações, como MIT Sloan School of Management. Novas versões foram testadas com empresas de aviação.

O Sistema R evoluiu para SQL/DS tornando-se o DB2. O grupo do Sistema R criou a Structured Query Language (SQL) - Linguagem de Consulta Estruturada. Esta linguagem tornou-se um padrão na indústria para bancos de dados relacionais e hoje em dia é um padrão ISO (International Organization for Standardization).

Mesmo a IBM sendo a empresa que inventou o conceito original e o padrão SQL, ela não produziu o primeiro sistema comercial de banco de dados. Isto foi realizado pela Honeywell Information Systems Inc., tendo seu sistema lançado em junho de 1976. O sistema foi desenvolvido com base em muitos dos princípios que a IBM concebeu, mas foi modelado e implementado fora da IBM.



### 1.3. Usuários (Atores De um Banco de Dados)

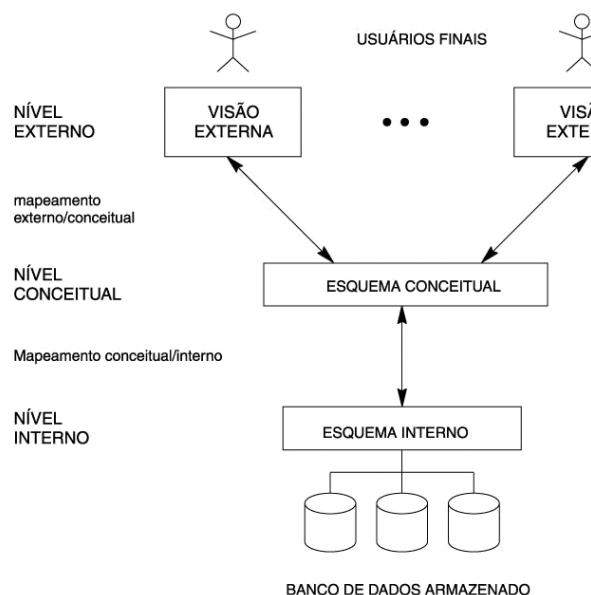
Um Banco de Dados pode apresentar diversos usuários cada qual com uma necessidade em especial, e com um envolvimento diferente com os dados. Os usuários podem ser classificados em categorias:

- Administradores de Bancos de Dados (DBA): em qualquer organização onde muitas pessoas compartilham muitos recursos, existe a necessidade de um administrador chefe para supervisionar e gerenciar estes recursos. Em um ambiente de base de dados, o recurso primário é a própria base de dados e os recursos secundários são o próprio SGBD e software relacionados. A administração desses recursos é de responsabilidade do DBA ("Database Administrator"). O DBA é responsável por autorizar acesso à base de dados e coordenar e monitorar seu uso, como também é responsável por problemas, tais como: quebra de segurança ou baixo desempenho. Em grandes organizações, existe uma equipe de DBA's;
- Analistas de Bancos de Dados (Projetistas): possuem a responsabilidade de identificar os dados a serem armazenados no BD e pela escolha da estrutura apropriada utilizada para armazená-los. Eles devem se comunicar com os possíveis usuários do BD, obter a visão dos dados que cada um possui, integrando-as de forma a se obter uma representação adequada de todos os dados. Estas visões são então analisadas e, posteriormente, integradas para que, ao final, o projeto da base de dados possa ser capaz de dar suporte aos requisitos de todos os grupos de usuários;
- Usuários Finais: existem profissionais que precisam ter acesso à base de dados para consultar, modificar e gerar relatórios. A base de dados existe para estes usuários. Podem ser usuários comuns do sistema, analistas de negócios ou usuários avançados que necessitam de uma maior interação com o BD.
- Analistas de Sistemas e Programadores de Aplicações: Os analistas determinam os requisitos dos usuários finais e desenvolvem especificações para transações que atendam estes requisitos, e os programadores implementam estas especificações, como: programas, testando, depurando, documentando e dando manutenção no mesmo. É importante que, tanto analistas quanto programadores, estejam a par dos recursos oferecidos.

## 1.4. NÍVEIS DE ABSTRAÇÃO

A arquitetura ANSI/SPARC prevê múltiplas visões de dados, um esquema conceitual (lógico) e um esquema interno (físico). Um SBD se divide em geral em três níveis:

- Nível Externo: possui as diversas descrições do BD de acordo com os grupos de usuários.
- Nível Conceitual: descreve a estrutura de todo o BD para uma determinada comunidade de usuários, ocultando detalhes sobre a organização física dos dados e apresentando a descrição lógica dos dados e das ligações existentes entre eles.
- Nível Interno: descreve a estrutura de armazenamento físico dos dados do BD, descreve o modelo físico dos dados que inclui detalhes sobre os caminhos de acesso aos dados internamente



Fonte: ELMASRI & NAVATHE, 2005

Figura 1.4 – Visões de um banco de dados



Sobre os níveis de abstração de um banco de dados podemos pensar assim: Nível externo é o que o usuário pensa e quer e deseja visualizar, nível conceitual é como o projetista irá implementar o banco de dados, e nível interno é como estes dados serão armazenados, formas de acesso físico por exemplo

Observe que os três níveis apresentam apenas descrições dos dados. Como os três níveis apresentam descrições diferentes para os mesmos dados, torna-se necessário converter uma representação em outra, ou seja, definir mapeamentos de dados entre os níveis.

## 1.5. SGBD

Um Sistema de Gerenciamento de Bancos de Dados tem como principais propriedades:

- Um Sistema de Gerenciamento de Bancos de Dados tem como principais propriedades.
- Controle de Redundância: Em um sistema tradicional de controle de arquivos cada usuário normalmente apresenta seus próprios arquivos armazenando o conjunto de dados que é de seu interesse, e nestes casos é comum ocorrer redundância de dados. Esta redundância consiste no armazenamento de uma mesma informação em locais diferentes, o que pode provocar sérios problemas. Alguns destes problemas consistem inicialmente no aumento de esforço computacional para realizar a atualização destes dados; aumento do espaço necessário para o armazenamento dos dados. O problema mais sério é que a representação dos dados desta forma pode tornar-se inconsistente, pois duas informações podem aparecer em locais distintos, mas apresentando valores diferentes. Em um sistema de BD as informações só se encontram armazenadas em um único local ou estão existindo duplicação controlada dos dados.
- Compartilhamento dos Dados: Um SGBD deve incluir um software para o controle de concorrência ao acesso dos dados em um ambiente multiusuário, de forma que possibilite o compartilhamento dos dados, garantindo que se vários usuários tentem realizar operações de atualização sobre um mesmo conjunto de dados, o resultado destas operações possa ser correto e consistente.
- Controle de Acesso: Quando vários usuários compartilham os dados, é comum que alguns não apresentem autorização para acesso a todo o BD. Por exemplo, os dados do faturamento de uma empresa podem ser considerados confidenciais e, desse modo, somente pessoas autorizadas devem ter acesso. Além disso, pode ser permitido, a alguns usuários, apenas a recuperação dos dados. Já, para outros, são permitidas a recuperação e

a modificação. Assim, o tipo de operação de acesso - recuperação ou modificação - pode também ser controlado. Tipicamente, usuários e grupos de usuários recebem uma conta protegida por senhas, que é usada para se obter acesso à base de dados, o que significa dizer que contas diferentes possuem restrições de acesso diferentes. Um SGBD deve fornecer um subsistema de autorização e segurança, que é usado pelo DBA para criar contas e especificar restrições nas contas. O SGBD deve então obrigar estas restrições automaticamente.

- Possibilidade de Múltiplas Interfaces: Diversos usuários com níveis de conhecimento técnico diferentes representam necessidades diversas no que se refere aos tipos de interfaces fornecidas pelo SGBD. Interfaces para consultas de dados, programação, e interfaces baseadas em menus ou em linguagem natural, são exemplos de alguns tipos que podem estar disponíveis



Um usuário com conhecimento técnico pode utilizar uma interface em que recupera dados mediante uma linguagem de consulta de dados, como SQL, já usuários com menor conhecimento técnico devem utilizar uma interface gráfica onde visualizam os dados e os selecionam para obter a consulta que necessitam ao invés de utilizarem uma linguagem para isto

- Representação de Relacionamento Complexo entre Dados: uma base de dados pode possuir uma variedade de dados que estão inter-relacionados de muitas formas. Um SGBD deve ter a capacidade de representar uma variedade de relacionamentos complexos entre dados, bem como recuperar e modificar dados relacionados de maneira fácil e eficiente.
- Forçar Restrições de Integridade: A maioria das aplicações de um banco de dados apresenta serviços que possibilitam garantir a integridade dos dados no BD. A restrição de integridade mais simples consiste na especificação do padrão de formato para os dados ou valores assumidos como um padrão.

Como exemplo de restrição de integridade, podemos pensar quando queremos armazenar o nome e a idade de uma pessoa. O nome deve ser uma cadeia de caracteres (string) menor que 50 caracteres alfabéticos, já a idade deve ser um dado numérico inteiro menor que 150. Estes são dois exemplos de restrições que podemos aplicar ao armazenar dados com intuito de garantir sua integridade



- Garantir Backup e Restauração de Dados: Um SGBD deve prover recursos para realização de cópias de segurança e restauração caso ocorra falhas de hardware ou software. O subsistema de backup e restauração do SGBD é o responsável pela restauração. Por exemplo, se o sistema de computador falhar no meio da execução de um programa que esteja realizando uma alteração complexa na base de dados, o subsistema de restauração é responsável em assegurar que a base de dados seja restaurada no estado anterior ao início da execução do programa. Alternativamente, o subsistema de restauração poderia assegurar que o programa seja re-executado a partir do ponto em que havia sido interrompido.



### 1.5.1. VANTAGENS DE UM SGBD

A escolha da tecnologia adequada de Banco de Dados e sua correta utilização trazem benefícios à maioria das organizações, tais como:

- Potencial para garantir Padrões: a abordagem de base de dados permite que o Administrador do Banco (DBA) defina e force a padronização entre os usuários da base de dados em grandes organizações. Isso facilita a comunicação e a cooperação entre vários departamentos, projetos e usuários. Padrões podem ser definidos para: formatos de nomes, elementos de dados, telas, relatórios, terminologias, etc. O DBA pode obrigar a padronização em um ambiente de base de dados centralizado, muito mais facilmente que em um ambiente onde cada usuário ou grupo tem o controle de seus próprios arquivos e software;
- Redução do Tempo de Desenvolvimento de Aplicações: um dos principais argumentos de venda para o uso da abordagem de um banco de dados é o tempo reduzido para o desenvolvimento de novas aplicações, tal como a

Alguns exemplos de SGBD's modernos que apresentam as vantagens citadas: MySQL, Oracle, MS SQL Server PostgreSQL. Para exemplificar nossas aulas utilizaremos o SGBD MySQL, que recomendo a todos já realizarem o download no site: <http://www.mysql.com/downloads/mysql/>, e posteriormente realizarem a instalação.



recuperação de certos dados da base de dados para a impressão de novos relatórios. Projetar e implementar uma nova base de dados pode tomar mais tempo do que escrever uma simples aplicação de arquivos especializada. Porém, uma vez que a base de dados esteja em uso, geralmente o tempo para se criar novas aplicações, usando-se os recursos de um SGBD, é bastante reduzido. O tempo para se desenvolver uma nova aplicação em um SGBD é estimado em 1/4 a 1/6 do que o tempo de desenvolvimento, usando-se apenas o sistema de arquivos tradicional;

- Independência de Dados: As aplicações de banco de dados não devem depender da forma como os dados estão representados e/ou armazenados;
- Flexibilidade: Pode ser necessário alterar a estrutura de uma base de dados devido a mudanças nos requisitos. Por exemplo, um novo grupo de usuários pode surgir com necessidade de informações adicionais, não disponíveis atualmente na base de dados. Um SGBD moderno permite que tais mudanças na estrutura da base de dados sejam realizadas sem afetar a maioria dos programas de aplicações existentes;
- Disponibilidade para atualizar as informações: Um SGBD disponibiliza o banco de dados para todos os usuários. Imediatamente após um usuário modificar uma base de dados, todos os outros usuários “sentem” imediatamente esta modificação. Essa disponibilidade de informações atualizadas é essencial para muitas aplicações, tais como: sistemas de reservas de passagens aéreas ou bases de dados bancárias. Isso somente é possível devido ao subsistema de controle de concorrência e restauração do SGBD;
- Economia de Escala: a abordagem de SGBD's permite a consolidação de dados e de aplicações, reduzindo-se, desse modo, o desperdício em atividades redundantes de processamento em diferentes projetos ou departamentos. Isto possibilita à organização como um todo investir em processadores mais poderosos e periféricos de armazenamento e de comunicação mais eficientes do que cada departamento adquirir seu próprio (menos potente) equipamento, o que reduz o custo total da operação e gerenciamento.
- Apesar de todas as facilidades oferecidas por um banco de dados, um projeto de implantação pode gerar um alto custo inicial para a organização.



## 1.5.2. LINGUAGENS

Uma vez que o projeto do BD tenha se completado e um determinado SGBD tenha sido escolhido para a sua implementação, o primeiro

passo consiste em realizar uma especificação dos esquemas conceituais e internos, e os respectivos mapeamentos entre eles. Para estas etapas o SGBD oferece algumas linguagens apresentadas abaixo:

- Linguagem de Definição de Dados (DDL - Data Definition Language): Utilizada pelos analistas e projetistas do BD para a definição dos esquemas do banco de dados. O SGBD também apresentará um interpretador para a DDL, o qual será responsável pelo processamento dos comandos da DDL, e realiza o armazenamento do esquema definido em estruturas internas do BD. Por exemplo, os comandos para criar, definir índice de uma tabela fazem parte da linguagem de definição de dados. Uma vez definido e preenchido o BD com os seus dados, estes normalmente sofrerão uma série de operações de acesso às informações nele armazenado.
- Linguagem de Manipulação de Dados (DML): O SGBD fornece esta linguagem para a especificação das operações de acesso ao banco. Os comandos da DML podem aparecer embutidos em outra linguagem (geralmente uma linguagem de programação de alto nível), e neste caso esta é denominada de Linguagem hospedeira, e DML é denominada de Sublinguagem de dados. De outra forma, se DML for utilizada isoladamente de uma forma interativa, passa a ser denominada de Linguagem de consulta (ou "query language" como a SQL).
- Linguagem de Controle de Dados (DCL): No controle de acesso e transações dos dados utiliza-se esta linguagem, que inclusive possibilita estabelecer os diversos níveis de segurança de cada usuário.

## RESUMO DA SEMANA

Nesta semana introdutória, definimos um banco de dados como uma coleção de dados relacionados, na qual os dados significam fatos registrados. Um SGBD é um pacote de software para implementação e manutenção de dados computadorizados. Em seguida, discutimos as categorias principais de usuários de um banco de dados. Depois, apresentamos uma lista de capacidades

que devem ser oferecidas por um software SGBD. Finalmente, vimos as linguagens envolvidas com banco de dados.

### Fórum



Realize uma pesquisa sobre a seguinte questão: [SGBD's gratuitos e pagos](#). Exemplos, vantagens e desvantagens.



### Atividades de aprendizagem

Para apoio complementar a esta primeira semana sugiro dois vídeos para maior contextualização:

[http://www.youtube.com/watch?v=qgnuH\\_qsI9o](http://www.youtube.com/watch?v=qgnuH_qsI9o)

<http://www.youtube.com/watch?v=tEAIls6aB2s&feature=related>

1. Um conjunto de programas que apóiam a criação, manutenção e operação de um banco de dados são chamadas: \_\_\_\_\_
2. A linguagem usada para definir tabelas, esquemas e domínios de atributo de dados é chamada:
  - a. ( ) linguagem de definição de armazenamento
  - b. ( ) linguagem de definição de banco de dados
  - c. ( ) linguagem de definição de dados
  - d. ( ) linguagem de definição de visões
  - e. ( ) linguagem de definição de esquema
3. A linguagem de manipulação do banco de dados permite:
  - a. ( ) modificar o esquema conceitual
  - b. ( ) modificar o esquema interno
  - c. ( ) especificar atualizações e recuperações
  - d. ( ) modificar o esquema externo
  - e. ( ) acessar o conteúdo do banco de dados
  - f. ( ) modificar a estrutura do banco de dados

4. Qual das opções abaixo não consiste em uma vantagem de se usar bancos de dados?

- a. ( ) informações atualizadas
- b. ( ) flexibilidade
- c. ( ) redundância controlada
- d. ( ) interfaces múltiplas com o usuário
- e. ( ) Investimentos iniciais baixos em hardware, software e treinamento.

5. Quem ou o que é responsável por manter a consistência do banco de dados?

- a) ( ) aplicações de banco de dados
- b) ( ) vendedor do SGBD
- c) ( ) todas as anteriores
- d) ( ) administrador do sistema de banco de dados
- e) ( ) usuários finais
- f) ( ) projetista do banco de dados

6. DBA significa \_\_\_\_\_

7. Realize uma pesquisa na internet e descreva as funcionalidades de ao menos 3 SGBD's existentes.

---

---

---

8. Na empresa onde trabalha, ou em outro local que possua um sistema de informação que conheça, qual é o SGBD utilizado? Quem interage com o banco de dados? Quais vantagens você enxerga em sua utilização em contraposição se fosse necessário o armazenamento de forma manual como em um armário de aço.

---

---

---

---

---

---

# Aula 2 – Modelos e Modelagem de Dados

## Objetivos da Aula

O Objetivo desta aula é apresentar os modelos de dados existentes, bem como o processo de modelagem de dados utilizando o Modelo Entidade Relacionamento.

### 2.1. Modelos de Dados

Os SGBDs evoluíram de sistemas de arquivos de armazenamento em disco, criando novas estruturas de dados com o objetivo de armazenar informações de forma mais eficiente e segura. Com o tempo, os SGBD's passaram a utilizar diferentes formas de representação, ou modelos de dados, para descrever a estrutura das informações contidas em seus bancos de dados. O modelo de dados é a principal ferramenta no fornecimento de informações sobre a abstração realizada na parte de interesse específico no mundo real. Os seguintes modelos de dados são normalmente utilizados pelos SGBD's: modelo hierárquico, modelo em redes, modelo relacional (amplamente usado) e o modelo orientado a objetos.

#### 2.1.1. Modelo Hierárquico

O modelo hierárquico foi o primeiro a ser reconhecido como um modelo de dados. Nesse modelo de dados, os dados são estruturados em hierarquias ou árvores. Os nós das hierarquias contêm ocorrências de registros, onde cada registro é uma coleção de campos (atributos), cada um contendo apenas uma informação. O registro da hierarquia que precede a outros é o registro-pai, os outros são chamados de registros-filhos.

Uma ligação é uma associação entre dois registros. O relacionamento entre um registro-pai e vários registros-filhos possui cardinalidade 1:N (um para

muitos, ou seja um pai pode ter vários filhos). Os dados organizados segundo este modelo podem ser acessados segundo uma seqüência hierárquica com uma navegação do topo para as folhas e da esquerda para a direita. Um registro pode estar associado a vários registros diferentes, desde que seja replicado. A replicação possui duas grandes desvantagens: pode causar inconsistência de dados quando houver atualização e o desperdício de espaço é inevitável. Características:

- Cada pai pode ter vários filhos.
- Cada filho pode ter apenas um pai.
- Duas entidades podem possuir apenas um relacionamento.
- Qualquer recuperação de dados deve obrigatoriamente percorrer a estrutura da árvore. (pai  $\Rightarrow$  filho  $\Rightarrow$  neto...)

Como exemplos de banco de dados que utilizam o modelo hierárquico podemos citar: IMS e System 2.

Vamos tomar como exemplo a figura 2.1, uma agência bancária como pai de um relacionamento pode ter vários clientes e um cliente pode ter várias contas, porém se um cliente pertencer a mais de uma agência, seus dados terão que ser replicados, o que pode causar inconsistência e desperdício de espaço.



Figura 2.1 – Exemplo modelo hierárquico

## 2.1.2. MODELO EM REDE

O modelo em rede surgiu como uma extensão ao modelo hierárquico, eliminando o conceito de hierarquia e permitindo que um mesmo registro estivesse envolvido em várias associações. No modelo em rede, os registros são organizados em grafos onde aparece um único tipo de associação que define uma relação 1:N (um para muitos) entre 2 tipos de registros: proprietário e membro.

Ao contrário do Modelo Hierárquico, em que qualquer acesso aos dados passa pela raiz, o modelo em rede possibilita acesso a qualquer nó da rede sem passar pela raiz, pois o modelo em rede permite a existência de entidades pais com muitos filhos e de entidades filhos com muitos pais.

Estes dois modelos: Hierárquico e Rede são orientados a registros, isto é, qualquer acesso à base de dados – inserção, consulta, alteração ou remoção – é feito em um registro de cada vez.

O modelo em rede apresenta como principal vantagem a possibilidade de uma modelagem mais próxima da realidade, porém não se firmou no mercado pelo surgimento do modelo relacional que veremos a seguir.

Vamos tomar como exemplo a figura 2.2, um equipamento pode ter várias bombas e motores, e estes dois podem sofrer manutenção mecânica. Assim neste modelo perde-se a restrição hierárquica.

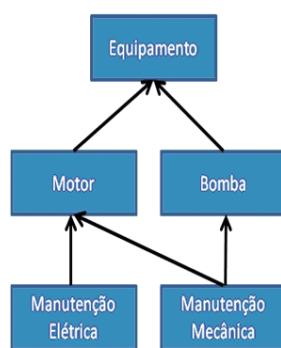


Figura 2.2: Exemplo modelo em rede

São exemplos de SGBD's em rede: DBMS10,IDSII,DMSII,IMAGE.

### 2.1.3. MODELO RELACIONAL

Os SGBD's que apresentaram maior sucesso e confiabilidade no mercado são os que adotaram o Modelo Relacional como base para sua construção. O Modelo Relacional, de modo geral, permite que os dados sejam “vistos” como tabelas. Ora, esta é uma maneira muito natural de armazenar e recuperar dados. Geralmente, quando fazemos uma relação de dados, tendemos a organizá-los em linhas e colunas, no formato de tabelas. Esta é uma importante razão para o sucesso das planilhas eletrônicas no mundo dos negócios.

O modelo relacional surgiu devido às necessidades de se **aumentar a independência de dados** nos sistemas gerenciadores de banco de dados, prover um conjunto de funções apoiadas em álgebra relacional para armazenamento e recuperação de dados além de permitir processamento dedicado. O Modelo relacional revelou-se ser o mais **flexível** e adequado ao solucionar os vários problemas que se colocam no nível da concepção e implementação da base de dados. A estrutura fundamental do modelo relacional é a relação (tabela). Uma relação é constituída por um ou mais atributos (campos) que traduzem o tipo de dados a armazenar. Cada instância do esquema (linha) é chamada de tupla (registro). O modelo relacional não tem caminhos pré-definidos para se fazer acesso aos dados como nos modelos que o precederam. Este modelo implementa estruturas de dados organizadas em relações. Porém, para trabalhar com essas tabelas, algumas restrições precisaram ser impostas para evitar aspectos indesejáveis, como a redundância e perda de dados e a incapacidade de representar parte da informação. Essas restrições são: integridade referencial, chaves e integridade de junções de relações.

Vamos tomar como exemplo a figura 2.3, um aluno pode cursar várias disciplinas e uma disciplina pode ser cursada por vários alunos criando assim uma relação muitos para muitos entre as tabelas Aluno e Disciplina.

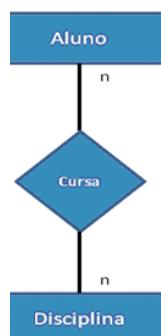


Figura 2.3 – Exemplo modelo relacional

Atualmente, mesmo que informalmente, o Modelo Relacional é visto no mercado quase como um sinônimo de Sistemas de Banco de Dados. Normalmente, quando nos referimos a banco de dados, os usuários já imaginam um conjunto de tabelas relacionadas.

Em nossa disciplina utilizaremos um SGBD que utiliza o Modelo Relacional como base para sua construção. Através das regras e conceitos deste modelo, “enxergaremos” nosso banco de dados implementado computacionalmente como um conjunto de tabelas inter-relacionadas.

#### 2.1.4. MODELO DE DADOS ORIENTADO À OBJETO

Representam os dados como coleções que obedecem a propriedades. São modelos geralmente conceituais dispondendo de poucas aplicações reais. Neste Modelo não seria interessante a existência de uma tabela de funcionários e dentro dela alguma referência para cada registro, de forma a podermos saber onde (em que departamento) o funcionário está alocado. Um conjunto de regras disponibilizaria em separado os funcionários da fábrica, que, no entanto, estariam agrupados aos demais, para o sistema de folha de pagamento.

### 2.2. MODELAGEM DE DADOS

Utilizaremos para nossa modelagem o modelo Entidade Relacionamento, pois este é um modelo de dados conceitual de alto nível, cujos conceitos foram projetados para estar mais próximo possível da visão que o usuário tem dos dados, não se preocupando em representar como estes dados estarão realmente armazenados de maneira física. O modelo ER é utilizado principalmente durante o processo de projeto conceitual de banco de dados.

Quando pensamos no Modelo Entidade Relacionamento temos três conceitos básicos:

- Conjunto de Entidades: Uma entidade pode ser vista como uma pessoa, “coisa” ou “objeto” no mundo real que é distingível de todos os outros objetos, como por exemplo, um cliente de um banco.



Figura 2.4 – Entidade Cliente

Já um conjunto é um grupo de entidades do mesmo tipo que compartilham os mesmos atributos, como por exemplo, um conjunto de clientes bancários, animais ou pessoas.

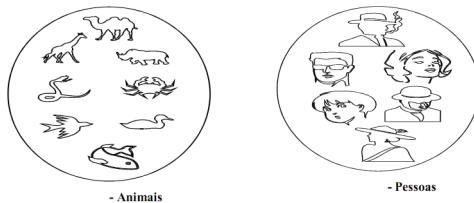


Figura 2.5 – Conjunto de Entidades mundo real

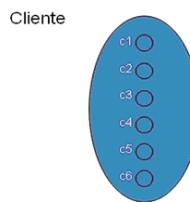


Figura 2.6 – Conjunto de Entidades Banco de Dados

- Conjunto de Atributos: Uma entidade é representada por um conjunto de atributos. Possíveis atributos do conjunto de entidades **cliente**, podem ser: nome-cliente, cpf, rua e cidade-cliente. Possíveis atributos do conjunto de entidade **CONTA** são: número-conta e saldo. Para cada atributo, existe um conjunto de valores permitidos chamado **domínio** daquele atributo. O domínio do atributo nome-cliente pode ser o conjunto de todas as cadeias de texto (strings) de certo tamanho.

Por exemplo, podemos limitar o tamanho do nome de uma pessoa a 50 caracteres, então este é seu domínio. Como também, o domínio do atributo número-conta pode ser o conjunto de todos os inteiros positivos. O atributo idade de uma entidade pessoa poderia ter como domínio os números inteiros entre 0 e 150



Formalmente, um atributo é uma função que mapeia um conjunto de entidades em um domínio.

- Conjunto de Relacionamentos: Um relacionamento é uma associação entre diversas entidades. Por exemplo, podemos definir um relacionamento que associa o cliente José Silva à conta 401. Isto especifica que José Silva é um cliente com conta bancária número 401. Um conjunto de relacionamentos é uma coleção de relacionamentos do mesmo tipo. Esse relacionamento pode ser representado pelo diagrama abaixo:



Figura 2.7 – Exemplo relacionamento entre cliente e conta

Nosso projeto conceitual de Banco de Dados será feito através da modelagem de dados usando o Modelo Entidade-Relacionamento (ME-R). Este modelo fornece as regras e conceitos para a criação de um Diagrama Entidade Relacionamento (DER), que deverá representar o banco de dados em questão, como por exemplo, um banco com vários clientes e contas, uma loja de produtos de beleza, de um consultório odontológico, uma indústria de peças, uma locadora de vídeo, uma escola, enfim, o negócio a que este banco de dados deve servir. Desta maneira, usaremos dois modelos teóricos para a construção de um banco de dados:

- O Modelo Entidade Relacionamento para criação do projeto conceitual do banco de dados;
- O Modelo Relacional para a implementação em um ambiente computacional pré-existente. Este ambiente computacional pré-existente será o nosso Sistema Gerenciador de Banco de Dados Relacional - MySql.

Com intuito de realizar a transferência entre modelos serão aplicadas as regras de [mapeamento](#) do Modelo Entidade Relacionamento ME-R para o Modelo Relacional MRel. Ou seja, para “converter” o diagrama conceitual de nosso banco de dados em um conjunto de tabelas relacionadas que possa ser implementado em um SGBD Relacional, usaremos um conjunto de passos. Este processo é chamado [mapeamento](#) do MER para o MRel.

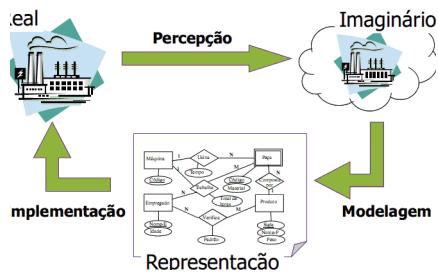


Figura 2.8 – Representação Mundo Real para Modelagem Banco de Dados

Para realizarmos o mapeamento dos dados entre os modelos, primeiramente utilizaremos os requisitos coletados para construção e realizaremos a modelagem utilizando o Modelo Entidade Relacionamento, em seguida “convertere-mos” os diagramas gerados em tabelas sobre o modelo relacional para implementarmos em um SGBD.

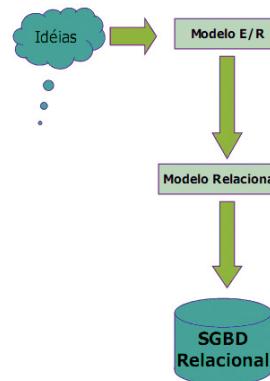


Figura 2.9 – Implementação de um BD

Agora veremos como construir um diagrama Entidade Relacionamento DER para o projeto de um banco de dados. Este será o primeiro passo para a construção de nosso banco de dados. O Modelo Entidade Relacionamento MER é composto por Entidades e Relacionamentos sem nos esquecermos dos atributos.

## 2.2.1. REPRESENTAÇÕES

A estrutura lógica geral de um banco de dados pode ser expressa graficamente por um diagrama ER, que consiste nos seguintes componentes:

1. **Retângulos** que representam conjuntos de entidades;
2. **Elipses** que representam atributos;
3. **Losangos** que representam relacionamentos entre conjuntos de

entidades;

4. Linhas que ligam atributos a conjuntos de entidades e conjuntos de entidades a relacionamentos. Alguns autores chamam as linhas de arestas, em analogia às teorias de grafos e redes.

## 2.2.2. ENTIDADES E ATRIBUTOS

Como descrito acima, a representação para um conjunto de entidades é um retângulo e, para cada atributo, uma elipse, como por exemplo, um conjunto de entidades Aluno pode possuir os atributos nome, prontuário e data de nascimento:

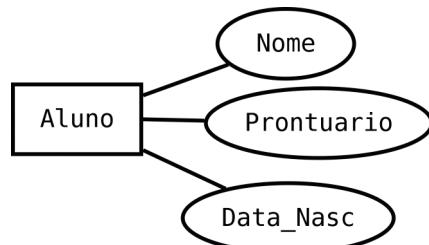
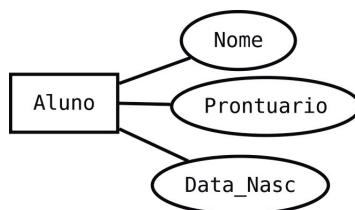


Figura 2.10 – Entidade com seus atributos

Este conjunto de entidades ALUNO possui os conjuntos de atributos Nome, Prontuario e Data\_Nasc. Uma entidade desse conjunto poderia ser {'lago', 10123, '2000-12-16'}.

## 2.2.3. CHAVE DE UM CONJUNTO DE ENTIDADES

É importante poder especificar como entidades e relacionamentos são identificados. Conceitualmente, entidades e relacionamentos individuais são distintos, mas em uma perspectiva de banco de dados a diferença entre eles precisa ser expressa em termos de seus atributos. Uma restrição importante sobre entidades é a aplicação de uma chave única. Um conjunto de entidades sempre possui um atributo cujo valor é diferente e válido para cada entidade. Tal atributo é chamado atributo-chave, e seu valor é usado para identificar cada entidade de modo único, único, como por exemplo, o atributo CPF de uma pessoa é sempre único, ou o prontuário de um aluno é sempre único. A chave de um atributo será sempre sublinhada.



Todo conjunto de entidades deve ter um conjunto de chaves cujo valor **identifique com unicidade** a entidade, pois a chave é o principal meio de acesso a uma entidade.

Figura 2.11 – Atributo chave de um entidade

É possível ocorrer situações onde é preciso mais de um atributo para identificar cada entidade do conjunto. Por exemplo, o RG (registro geral) dos brasileiros não identifica um único cidadão. É possível, que um mesmo número de RG possa ocorrer em duas unidades diferentes da federação, ou seja, um mesmo número de RG poderia ser emitido em São Paulo e Minas Gerais. Assim, para ser identificador, o RG precisa ser composto com o valor do Órgão Emissor, no caso de São Paulo, SSP\_SP. Estes dois atributos (RG e Órgão Emissor) se, juntos, identificam cada brasileiro. Nestes casos, quando é necessário mais de um atributo, a chave do conjunto de entidades será composta.

#### 2.2.4. RELACIONAMENTOS

Nenhuma informação armazenada no Banco de Dados existe isoladamente. Todos os elementos pertencentes ao mundo real (restrito) modelado de alguma forma estão associados a outros elementos. Normalmente estas associações representam ações físicas ou alguma forma de dependência entre os elementos envolvidos. Um **relacionamento** é uma associação entre diversas entidades.

Por exemplo, vamos considerar um conjunto de relacionamentos **TRABALHA** entre os conjuntos de entidades: **EMPREGADO** e **DEPARTAMENTO** que é apresentado na figura 2.18. Este relacionamento associa cada empregado com o departamento em que ele trabalha. Cada instância de relacionamento em **TRABALHA** associa uma entidade empregado a uma entidade departamento. Cada instância de relacionamento conecta uma entidade **EMPREGADO** a uma entidade **DEPARTAMENTO**. Na figura abaixo, os empregados e1, e3 e e6 trabalham para o departamento d1; e2 e e4 trabalham para d2; e e5 e e7 trabalham para d3



A representação dos conjuntos de entidades e de relacionamentos apresentadas acima no Modelo Entidade Relacionamento é representada na figura abaixo:



Figura 2.12 – Modelo ER para relacionamento TRABALHA

O conjunto de relacionamentos é, portanto, representado por um losango. Enquanto que para os conjuntos de entidades os atributos são obrigatórios, para os conjuntos de relacionamentos, eles são optativos.

O conjunto de entidade só faz sentido quando especificamos seus atributos.

Já um conjunto de relacionamentos (CR), nem sempre precisa possuir atributos. Sua existência justifica-se apenas pela função de relacionar uma ou mais entidades. Em geral, os atributos dos conjuntos de relacionamentos, quando existem, especificam dados sobre tempo (data, horário), quantidades, valores, enfim, atributos relativos a transações, ações, ocorrências, que caracterizam os relacionamentos.

## 2.2.5. GRAU DE RELACIONAMENTO

O grau de um conjunto de relacionamentos indica o número de conjuntos de entidades participantes. Um tipo de relacionamento de grau dois é chamado **binário**, de grau três de **ternário**, de grau quatro **quaternário**, acima disso, **n-ário**. A quantidade de Entidades envolvidas em um Relacionamento pode ser determinada por sua semântica. Desta forma, podem-se categorizar os graus de relacionamento em:

- **Unário:** é o grau de Relacionamento que envolve um único Tipo de Entidade.

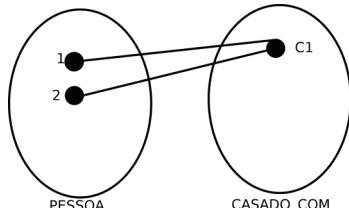


Figura 2.13 – Exemplo de relacionamento Unário

- Binário: é o grau de Relacionamento que envolve dois Tipos de Entidades.

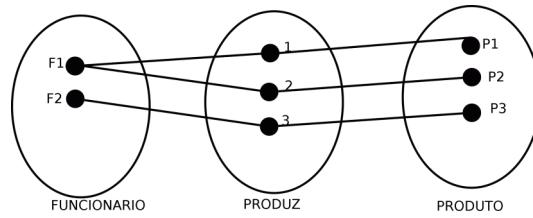


Figura 2.14 – Exemplo de relacionamento binário

- Ternário: é o grau de Relacionamento que envolve três Tipos de Entidades.

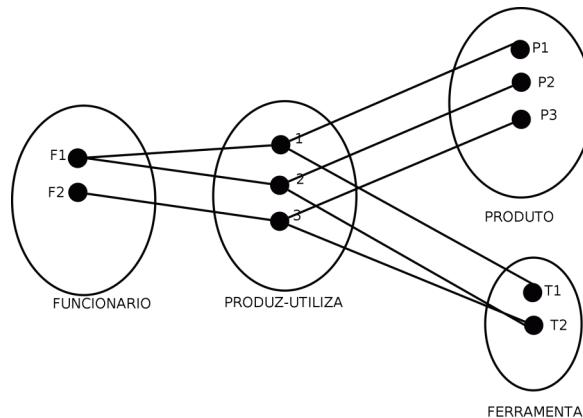


Figura 2.15 – Exemplo de relacionamento ternário

- Quaternário: é o grau de Relacionamento que envolve quatro Tipos de Entidades.

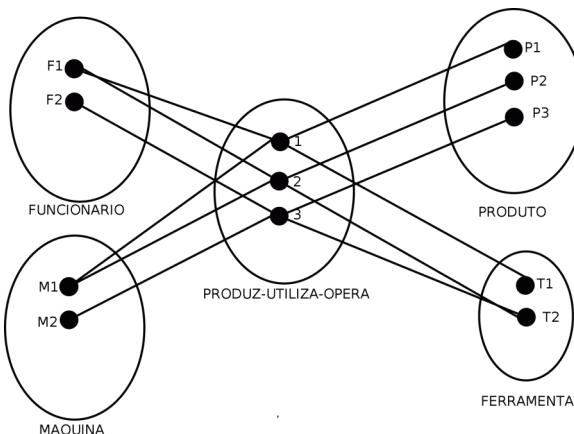


Figura 2.16 – Exemplo de relacionamento quartanário

A quantidade de Entidades envolvidas em cada Relacionamento é determinada pela Cardinalidade do grau de Relacionamento, ou seja, podemos estabelecer a quantidade mínima e máxima de Entidades envolvidas com cada Entidade relacionada.

- A Cardinalidade Mínima determina a quantidade mínima de Entidades relacionadas por um número representativo, ou seja, 0 (zero) 1, 2, ..., N (muitos).
- A Cardinalidade Máxima determina a quantidade máxima de Entidades relacionadas por um número representativo, ou seja, 1, 2, ..., N (muitos)

## 2.2.6. RESTRIÇÕES DE RELACIONAMENTO

Os relacionamentos entre entidades possuem certas restrições que limitam as combinações possíveis das entidades que dele participam. Uma destas restrições é a razão de cardinalidade. A quantidade de Entidades envolvidas em cada Relacionamento é determinada pela Cardinalidade do Tipo de Relacionamento, ou seja, pode-se estabelecer a quantidade mínima e máxima de Entidades envolvidas com cada Entidade relacionada.

Até agora vimos, de modo geral, como construir um diagrama de entidade relacionamento (DER) para projetar um banco de dados. Aprendemos os quatro principais construtores deste modelo: os conjuntos de entidades (CE), os conjuntos de relacionamentos (CR), os atributos de entidades e relacionamento. Agora, veremos um dos mais importantes conceitos do MER: a restrição de razão de cardinalidade do relacionamento. O entendimento deste conceito será imprescindível para o processo de mapeamento do diagrama entidade relacionamento DER para o Modelo Relacional Mrel, ou seja, para transformar o projeto conceitual do banco de dados em um conjunto de tabelas.



Para o grau de Relacionamento Binário (que são mais comuns) podemos citar que as Cardinalidades são as seguintes:

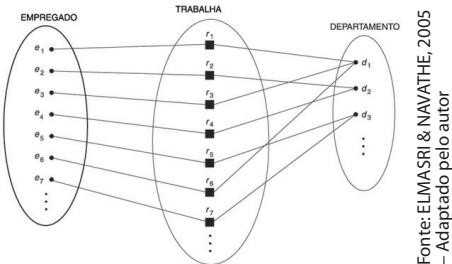
### UM-PARA-MUITOS (1 : N) ou alternativamente (N : 1):

Vamos ver um exemplo. No relacionamento binário TRABALHA, os conjuntos de entidades DEPARTAMENTO : EMPREGADO foram representados a seguir como tendo razão de cardinalidade 1:N (um para muitos) conforme DER abaixo.



Figura 2.17 – Exemplo de Diagrama ER Um-Para-Muitos

Já a figura abaixo apresenta o diagrama de ocorrências para a cardinalidade entre as entidades.



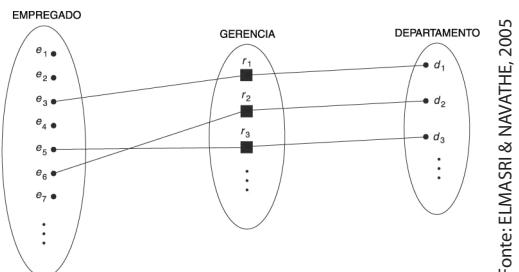
Fonte: ELMASRI & NAVATHE, 2005  
– Adaptado pelo autor

Figura 2.18– Diagrama de ocorrências para o relacionamento um para muitos

Desta maneira, na empresa a qual este diagrama representa, cada entidade EMPREGADO pode estar relacionada a apenas um DEPARTAMENTO (um empregado pode trabalhar apenas para um departamento), mas cada entidade DEPARTAMENTO pode estar relacionada a inúmeras entidades EMPREGADO (um departamento pode conter vários empregados).

#### UM-PARA-UM (1 : 1):

Vejamos um exemplo de cardinalidade 1:1. No caso do relacionamento **GERENCIA**



Fonte: ELMASRI & NAVATHE, 2005

Figura 2.19: Exemplo de diagrama de ocorrências para o relacionamento um para um

Uma entidade DEPARTAMENTO está relacionada a apenas um EMPREGADO, que gerencia esse departamento. Este relacionamento é 1:1, pois cada empregado gerente pode gerenciar apenas um departamento e, dado um departamento, este departamento pode ter apenas um gerente. Podemos ver que, apenas uma seta sai de um empregado ligando-o a apenas um departamento, e vice-versa. A representação deste relacionamento no DER seria:



Figura 2.20: Exemplo diagrama ER para o relacionamento um para um

É importante salientarmos que esta restrição de cardinalidade de UM-PARA-UM é definida pela realidade da empresa que se deseja representar no banco de dados. Se em outra empresa, for possível que um mesmo empregado gerencie diversos departamentos, a razão de cardinalidade 1:1 já não seria adequada a esta representação e sim 1 : N.

### MUITOS-PARA-MUITOS (N : M)

Para exemplificar a razão de cardinalidade N:M (muitos para muitos), tomemos o relacionamento TRABALHA\_EM entre EMPREGADO e PROJETO (ELMASRI & NAVATHE, 2005). Vamos considerar que um empregado pode trabalhar em diversos projetos e que diversos empregados podem trabalhar em um projeto.

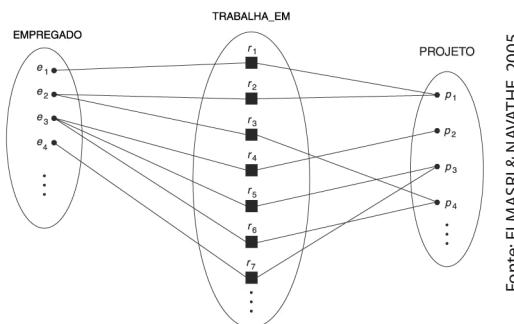


Figura 2.21: Exemplo de diagrama de ocorrência para o relacionamento binário muitos para muitos

A partir da figura 2.21 apresentada acima, podemos ver que o empregado e2 trabalha nos projetos p1 e p4; por sua vez, no projeto p4 trabalha além do empregado e2, também o empregado e3. O empregado e3 trabalha em 3 projetos; o projeto p2 conta com apenas 1 empregado.

Vejamos abaixo como seria a representação deste relacionamento no DER:

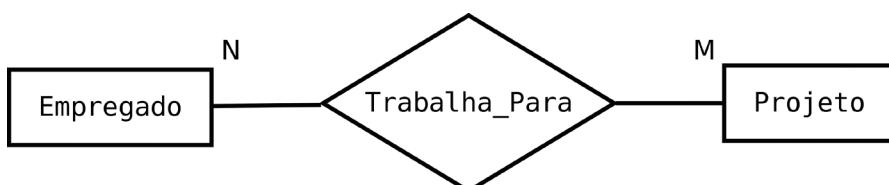


Figura 2.22 : Exemplo de diagrama ER binário muitos para muitos

Figura 2.22 : Exemplo de diagrama ER binário muitos para muitos

Os relacionamentos de grau maior que dois (binários), ou seja, ternários, quaternários e acima, possuem também razão de cardinalidades. Mas, como este conceito não interferirá na criação de um banco de dados simples, não é objeto de estudo de nosso curso. Para saber mais a respeito, veja a bibliografia deste caderno.



## AUTO- RELACIONAMENTO

Para efetuar a modelagem utilizando o modelo entidade relacionamento, faço o download do software DIA no site [http://dia-installer.de/index\\_en.html](http://dia-installer.de/index_en.html) , nele podemos construir nossos diagramas e realizarmos a exportação para imagens

Vimos até agora o conceito de razão de cardinalidade para relacionamentos binários. É bom lembrar que os relacionamentos unários também possuem este conceito. No relacionamento unário os elementos de uma entidade se relacionam a outros elementos dessa mesma entidade. Podemos verificar este tipo de relacionamento no exemplo abaixo:

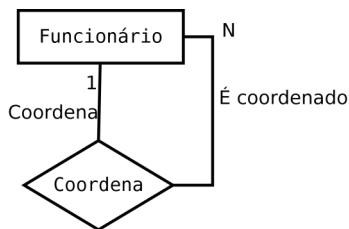


Figura 2.23: Exemplo de relacionamento unário

Com esta cardinalidade, um Coordenador (que é um funcionário) coordena vários funcionários e Funcionário é coordenado por um só Coordenador.

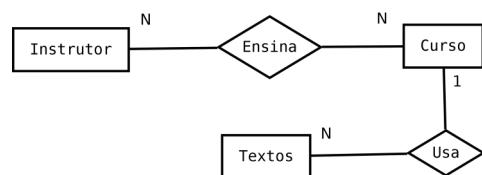
## RESUMO DA AULA

Nesta aula vimos alguns conceitos importantes como os principais modelos de dados possíveis para implementação de um banco de dados. Apresentamos os conceitos de modelagem presentes em um modelo conceitual de alto nível, o modelo Entidade-Relacionamento (ER). Definimos então os conceitos básicos de entidades e atributos no modelo ER. Discutimos também de forma breve, conceitos do modelo ER como o conjunto de entidades e relacionamentos, atributos, atributos-chave, tipos de relacionamentos. Conhecemos as notações para representarmos os esquemas ER na forma de diagramas. Vimos por fim, como aplicar restrições estruturais (cardinalidade) em relacionamentos, especialmente no tipo binário.

## Atividades de aprendizagem:

1. Através de exemplos de Diagramas ER diferentes dos exemplificados neste caderno, ilustre os conceitos a seguir do modelo ER:
  - a. Entidade
  - b. Relacionamento
  - c. Auto-relacionamento
2. Construa diagramas de Entidade Relacionamento com sua cardinalidades para as atividades abaixo propostas:
  - a. Cliente compra carro.
  - b. Médico consulta paciente.
  - c. Professor de informática ministra curso utilizando softwares.
  - d. Empregado trabalha no departamento, sendo que o empregado e o departamento possuem telefone.

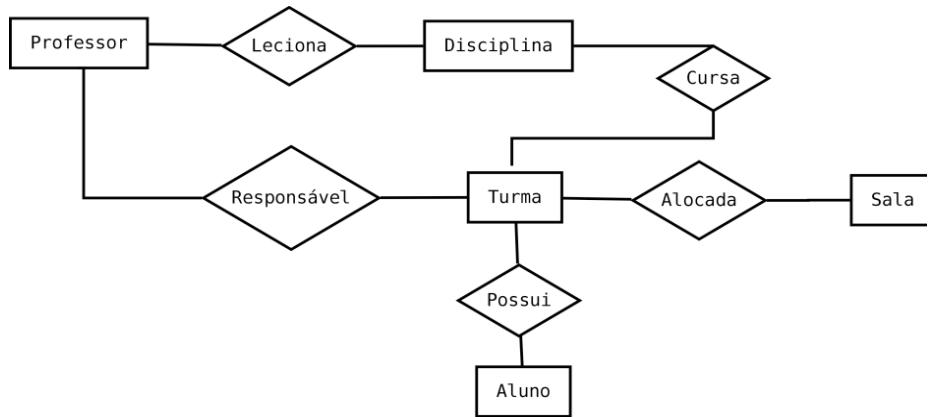
Exemplo do exercício 2: Instrutor ensina curso utilizando vários textos.



3. Tendo em consideração o seguinte texto:

Um professor pode lecionar várias disciplinas que pode ser lecionada por vários professores. Um professor pode ser responsável de diversas turmas e cada turma tem de ter um único professor responsável. Cada turma tem uma só sala onde tem as suas aulas, mas cada sala pode ter mais do que uma turma com aulas ao longo do dia. Cada turma tem vários alunos. Cada aluno pertence à uma e uma só turma. Cada turma pode cursar várias disciplinas, que podem ser cursadas por turmas diferentes.

Dado o diagrama E-R abaixo, marque as cardinalidades à situação descrita.



---

---

---

---

# Aula 3 – O Modelo de Dados Relacional

## Objetivo da aula:

Esta aula tem como objetivo apresentar os principais conceitos do modelo de dados mais utilizado atualmente para implementação de um banco de dados, o Modelo Relacional que representa um banco de dados em um conjunto de relações.

Na aula anterior conhecemos e vimos como construir diagramas utilizando o modelo ER. Nesta aula conheceremos o modelo de dados relacional. O Modelo de Dados Relacional foi introduzido por Ted Codd da IBM em 1970. Entre os modelos de dados de existentes, o modelo relacional é o mais simples e difundido, com uma estrutura de dados uniforme, e também o mais formal.

### 3.1. Conceitos

O Modelo Relacional será a base para implementarmos nosso banco de dados. Ele representa os dados da base de dados como uma coleção de relações. Podemos pensar que, cada relação pode ser entendida como uma tabela ou um simples arquivo de registros (Citação). Por exemplo, a base de dados de arquivos que são mostradas pela Figura 3.1, é similar a representação do modelo relacional. No entanto, possui algumas importantes diferenças entre relações e arquivos como veremos a seguir.

ALUNO	Nome	Numero	Turma	Departamento	
	Smith	17	1	CC	
	Brown	8	2	CC	
CURSO	Nome do Curso		Número do Curso	Creditos	Departamento
	Introdução à Ciência da Computação		CC1310	4	CC
	Estruturas de dados		CC3320	4	CC
	Matemática Discreta		MAT2410	3	MATH
	Banco de dados		CC3380	3	CC
DISCIPLINA	Identificador de Disciplina	Número do Curso	Semestre	Ano	Instrutor
	85	MAT2410	Segundo Semestre	98	King
	92	CC1310	Segundo Semestre	98	Anderson
	102	CC3320	Primeiro Semestre	99	Knuth
	112	MAT2410	Segundo Semestre	99	Chang
	119	CC1310	Segundo Semestre	99	Anderson
	135	CC3380	Segundo Semestre	99	Stone
HISTORICO_ESCOLAR	Número do Aluno	Identificador_Disciplinas	Nota		
	17	112	B		
	17	119	C		
	8	85	A		
	8	92	A		
	8	102	B		
	8	135	A		

Figura 3.1: Exemplo de um arquivo de dados baseado no modelo relacional

Ao pensarmos em uma relação como uma tabela de valores, cada linha representa uma coleção de valores que estão relacionados. Esses valores podem ser interpretados como um fato que descreve uma entidade ou uma instância. Utilizamos o nome da tabela e os nomes das colunas para nos ajudar a interpretar o significado dos valores em cada linha da tabela, que são na verdade, dados a respeito dos dados, chamados **metadados**.

Como exemplo vamos observar a figura 3.1, a primeira tabela é chamada ALUNO porque cada linha representa fatos sobre uma entidade aluno em particular. Os nomes das colunas - Nome, Número, Turma, Departamento - especificam como interpretar os valores em cada linha baseando-se nas colunas que cada valor se encontra



Na linguagem do modelo relacional, cada linha é chamada de **tupla**, a coluna ou cabeçalho é chamado de **atributo** e a **tabela** de relação. Desta maneira, o conjunto de nomes das tabelas e suas colunas são chamados de **esquema da relação**. Assim, o esquema da relação ALUNO é:

$$\text{ALUNO} = \{\text{nome, número, turma, departamento}\}$$

Temos que conhecer também conceito de **grau de uma relação**, este é o número de atributos da relação, no exemplo acima então, o grau de relação do esquema ALUNO é quatro, pois possui quatro colunas.

Uma coluna de dados possui um tipo de dado que descreve os valores que podem aparecer nela, por exemplo, na coluna **NÚMERO** de um aluno esperamos um valor numérico como 17, 18 e não caracteres, este tipo de dado que especifica os possíveis valores de uma coluna é chamada de **domínio**.

No esquema de relação ALUNO podemos especificar alguns domínios para atributos da relação ALUNO, vejamos alguns exemplos:

- Nome: Conjunto de cadeia de caracteres que representa nomes de pessoas;
- Numero: Conjunto de dados numéricos com limite de cinco dígitos;
- Turma: Conjuntos de códigos das turmas da faculdade;
- Departamento: Conjunto de códigos dos departamentos acadêmicos, como CC, EP, etc.

Assim, de acordo com nosso exemplo de domínio para a tabela ALUNO, o que esperamos obter em uma linha (tupla), é o conjunto de valores dos atributos para um determinado estudante. Por exemplo, existe um aluno de nome Smith, seu número é 17, sua turma é 1 e seu departamento CC.



A figura 3.2 apresentada abaixo mostra um exemplo da relação ALUNO. Cada tupla representa uma entidade aluno em particular, cada atributo corresponde a um cabeçalho de coluna, os valores apresentados como nulos (null) representam atributos em que os valores não existem para alguma tupla individual de ALUNO.

ALUNO	Nome	INSS	FoneResidencia	Endereco	FoneEscritorio	Idade	MPG
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	null	19	3.21	
Katherine Ashly	381-62-1245	375-4409	125 Kirby Road	null	18	2.89	
Dick Davidson	422-11-2320	null	3452 Elgin Road	749-1253	25	3.53	
Charles Cooper	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93	
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	null	19	3.25	

Fonte: Adaptado de EL MASRI & NAVATHE, 2005

Figura 3.2: Conjunto de atributos e tuplas da relação ALUNO

## 3.2. ATRIBUTOS-CHAVES

Uma relação é definida como um conjunto de tuplas. Por padrão, todos os elementos de um conjunto devem ser distintos. Assim, todas as tuplas de uma relação também são distintas. Isto significa que nenhuma tupla pode ter a mesma combinação de valores para todos os seus atributos. Desta maneira, temos que ter um valor que chamamos de **atributo-chave** que é utilizado para identificar de modo único uma tupla em uma relação.



Vamos voltar à figura 3.1, nela podemos verificar que o Nr 18 da relação ALUNO serve para identificar o aluno Brown

Geralmente, um esquema de relação pode ter mais que uma chave. Nos casos em que isto ocorra, cada chave é chamada chave-candidata. Por exemplo, o esquema da relação ALUNO poderia ter um atributo adicional Código, para indicar o código interno de alunos na escola. Assim, o esquema teria duas chaves candidatas: Nr e Código.

Após identificarmos as chaves candidatas devemos definir uma delas como a chave-primária da relação. A indicação no modelo de qual chave-candidata é a chave-primária é realizada se sublinhado os atributos que formam a chave-candidata escolhida como podemos ver na figura 3.3.

ALUNO
•Nome
<u>•Nr</u>
•Turma
•Departamento

ALUNO
<u>•Código</u>
•Nome
•Nr
•Turma
•Departamento

Fonte: Elaborado pelo autor

Figura 3.3: Tabela ALUNO com duas chaves candidatas: Nr e Código

Vejamos um exemplo, o atributo Nome da relação ALUNO não deve ser indicado como chave, uma vez que nada garante a que não haja ocorrência de nomes duplicados (homônimos).

Em certas relações pode ser necessário mais de um atributo para identificar cada tupla da relação de forma única. Por exemplo, vejamos a relação PESSOA apresentada na figura 3.4 ao lado::

<b>PESSOA</b>
•Nome
♦RG
♦OrgaoEmissor
•DataNasc

Fonte: Elaborado pelo autor

Figura 3.3: Tabela PESSOA utilizando chave composta

Conforme já vimos na aula dois, um determinado atributo pode não garantir a unicidade de uma tupla, desta maneira a identificação de cada pessoa deve ser feita pelo valor do atributo RG e do atributo OrgaoEmissor. A utilização de mais de um atributo para a composição da chave primária chamamos de chave composta. Alternativamente para não utilizarmos uma **chave composta** na relação PESSOA, poderíamos incluir um campo de identificação única como um código, por exemplo.

### 3.3. CHAVE ESTRANGEIRA

O conceito de chave estrangeira é de grande importância na construção de banco de dados relacional. Vamos começar com exemplo:

<b>FUNCIONARIO</b>
•Nome
♦CPF
•Setor
•DataNasc

<b>SETOR</b>
♦Cod_Setor
•Nome_Setor
◦Localizacao

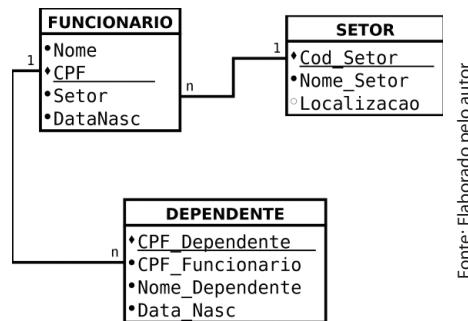
<b>DEPENDENTE</b>
♦CPF_Dependente
•CPF_Funcionario
•Nome_Dependente
•Data_Nasc

Fonte: Elaborado pelo autor

Figura 3.4: Tabelas para um esquema de um banco de dados relacional

No esquema de tabelas apresentada na figura 3.4 temos três relações (tabelas). A relação FUNCIONARIO possui os dados de um funcionário de uma empresa, como o Nome, CPF, Setor onde trabalha e Data de Nascimento. A relação SETOR possui o Nome e Localização de um setor que é identificada por um código. A relação DEPENDENTE possui o CPF deste como chave primária, o CPF do funcionário o qual ele é dependente, o Nome e a Data de Nascimento do dependente.

Podemos verificar que alguns atributos estão presentes em mais de uma tabela. Através das relações apresentadas abaixo podemos verificar que o atributo Setor da tabela FUNCIONARIO representa o código do setor onde o funcionário está lotado, sendo o mesmo atributo Cod\_Setor da tabela SETOR. O mesmo caso ocorre entre os atributos CPF e CPF\_Funcionario das tabelas FUNCIONARIO e DEPENDENTE respectivamente. A figura 3.5 apresentada abaixo ilustra esta relação.



Fonte: Elaborado pelo autor

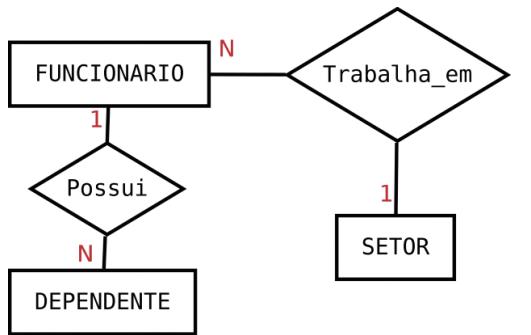
Figura 3.5: Relação entre tabelas

Como podemos verificar na figura 3.5, apenas a chave primária de uma tabela deve ser repetida em outra tabela. É o que acontece no esquema relacional apresentado. A chave primária de FUNCIONARIO está representada na tabela DEPENDENTE e a chave primária da tabela SETOR está representada na tabela FUNCIONARIO, como **chave estrangeira**. Desta maneira, podemos saber, por exemplo, qual é o SETOR de um funcionário (através do atributo Setor) ou quais dependentes possui um FUNCIONARIO através do atributo CPF\_Funcionario.



O valor do atributo Setor na tabela FUNCIONARIO poderia ser null (nulo) se o funcionário não estiver locado em nenhum setor no momento.

Desta forma, a **chave estrangeira** possibilita a implementação do conceito de relacionamento entre entidades. Com isto, podemos no Modelo Relacional, realizar a modelagem representada no Modelo Entidade Relacionamento que mostra um conjunto de entidades relacionado a outro conjunto de entidades, com determinada razão de cardinalidade. Então, o diagrama ER correspondente ao mapeamento realizada na figura 3.5 poderia ser representado conforme a figura 3.6 abaixo.



Fonte: Elaborado pelo autor

Figura 3.6: Diagrama ER entre entidades

### 3.4. INTEGRIDADE REFERENCIAL

A **restrição de integridade referencial** é responsável por manter a consistência entre tuplas (registros) de duas relações. Esta restrição de integridade referencial estabelece que uma tupla de uma relação que se refere à outra relação deve se referir a uma tupla existente naquela relação. Vamos exemplificar para ficar mais claro, vejamos a figura 3.7 abaixo:

FUNCIONARIO			
NOME	CPF	SETOR	DATA_NASC
Jose	33875419294	1	10/05/1980
Carlos	26218188892	2	17/05/1970

Setor		
Cod_Setor	Nome	Localizacao
1	Informatica	Andar 4
2	RH	Terreo

Fonte: Elaborado pelo autor

Figura 3.7: Relações FUNCIONARIO e SETOR

Na figura 3.7 podemos observar que o atributo Setor de FUNCIONARIO indica o número do SETOR que cada funcionário trabalha. Assim, todos os valores do atributo Setor (chave estrangeira) nas tuplas da relação FUNCIONARIO devem pertencer ao conjunto de valores do atributo Cod\_Setor (chave primária) da relação SETOR. Desta maneira podemos definir que o conceito de Integridade Referencial decorre da implementação de chaves estrangeiras em um esquema relacional. Assim temos duas regras para dizer que um conjunto de atributos será chave estrangeira de uma relação:

- Os atributos da chave estrangeira têm o mesmo domínio dos atributos da chave-primária a qual se relaciona. Podemos dizer então que os atributos chave estrangeira fazem referência à chave primária.

- O valor da chave estrangeira será algum valor que ocorre na chave primária a que ela faz referência ou terá o valor null.



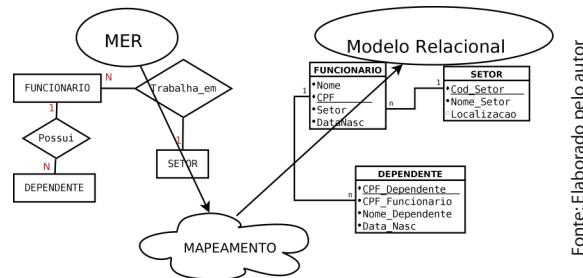
Se as duas regras apresentadas acima não ocorrerem, não há INTEGRIDADE REFERENCIAL no banco de dados implementado

As restrições de integridade devem ser especificadas no esquema da base de dados relacional se o projetista quiser manter essas restrições válidas para toda a base de dados. Desta maneira, em um sistema relacional, a linguagem de definição de dados (DDL) deve fornecer métodos para especificar os vários tipos de restrições tal que o SGDB possa garantir-las automaticamente

### 3.5. MAPEAMENTO

Como já vimos anteriormente, o Modelo Relacional (MRel) implementa as tabelas relacionadas, com muitos recursos para segurança dos dados, controle de acesso, consultas e manutenção dos dados. Porém, este modelo não é o modelo mais adequado para se fazer projetos de banco de dados. Então conhecemos o Modelo Entidade Relacionamento MER, que por meio de seus recursos visuais, se apresenta mais claro, simples e intuitivo.

Desta maneira, em projetos de banco de dados, normalmente a modelagem dos dados é realizada através de um modelo de dados de alto-nível. O modelo de dados de alto-nível geralmente adotado é o Modelo Entidade-Relacionamento (MER) e o esquema das visões e de toda a base de dados é especificado em diagramas entidade-relacionamento (DER).



Fonte: Elaborado pelo autor

Figura 3.8: Mapeamento entre MER e MRel

O Modelo Entidade Relacionamento possui um maior número de abstrações do que as que foram vistas em nossas aulas. Existem ainda definições para: Conjunto de Entidades Fracas, Atributos Compostos, Atributos Multivalueados, por exemplo. Vimos em nossas aulas os principais conceitos do modelo, que

permitem a construção da maior parte dos bancos de dados para as aplicações comerciais mais comuns.

Agora, veremos como traduzir um DER para um esquema MRel. Para isto utilizaremos um conjunto de passos para que possamos implementar nosso banco de dados em um SGBD Relacional.



**Passo 1: Mapeamento dos tipos de entidades:** Consiste na criação de uma relação que inclua todos os atributos de uma entidade. Assim, as entidades do MER são transformadas em tabelas no MRel. O atributo chave da entidade passa a ser a chave primária da relação (tabela). Vejamos um exemplo na figura 3.9.

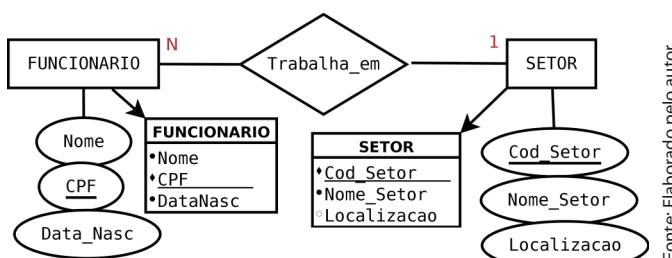


Figura 3.9: Primeiro passo no mapeamento entre MER e MRel

O livro texto deste cADERNO - ELMASRI & NAVATHE, 2005 - descreve todas as extensões possíveis de um banco de dados, é muito útil para o estudante que queira se aprofundar no conteúdo.>

No exemplo da figura 3.9, criamos as relações FUNCIONARIO e SETOR, que correspondem às entidades FUNCIONARIO e SETOR presentes no DER. Neste momento não nos preocupamos ainda com os relacionamentos. O que fizemos foi apenas definir as chaves primárias conforme dito anteriormente.

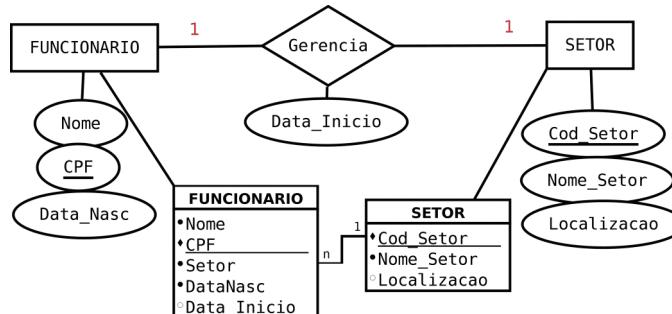
Nos próximos passos iremos realizar o mapeamento dos relacionamentos. Este mapeamento depende do grau do relacionamento (número de conjuntos de entidades nele envolvidos – binário, ternário, etc.) e da razão de cardinalidades (1:1, 1:N e N:M).

**Passo 2: Mapeamento dos Conjuntos de Relacionamentos binários de razão de cardinalidade 1:1:** Para este caso temos 3 possíveis opções de mapeamento: (1) criar chave estrangeira em uma das relações, (2) gerar uma única relação para as entidades e o relacionamento, (3) gerar uma relação exclusiva para o relacionamento, porém, esta opção caracteriza como veremos mais adiante o mapeamento N:M.



As opções mais utilizadas e que devem ser seguidas são a primeira e a terceira.>

Vamos observar na figura 3.10 apresentada abaixo uma opção de mapeamento utilizando a primeira opção.

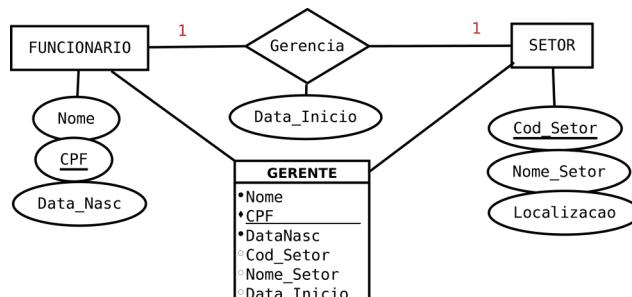


Fonte: Elaborado pelo autor

Figura 3.10: Primeira opção de mapeamento para relação 1:1.

De acordo com a primeira opção de mapeamento apresentada na figura 3.10, foi acrescentada à relação FUNCIONARIO os atributos do relacionamento GERENCIA (Data\_Inicio) e a chave primária da relação SETOR (Cod\_Setor) virou uma chave estrangeira da relação FUNCIONARIO (Setor), assim na prática apenas um percentual de tuplas na relação FUNCIONARIO terá um valor no atributo Setor, o restante dos valores dessa chave estrangeira assumirão o valor null (nulo), tendo em vista que somente alguns funcionários gerenciarão algum setor.

A segunda opção só seria aplicável se pensarmos no relacionamento de forma isolada, pois consistiria em gerar uma única tabela para as entidades e seu relacionamento como podemos observar na figura 3.11.



Fonte: Elaborado pelo autor

Figura 3.12: Terceira opção de mapeamento para relação 1:1

Como podemos observar na figura 3.12, esta opção de mapeamento gerou uma tabela exclusiva para o relacionamento, onde as chaves primárias de FUNCIONARIO (CPF) e SETOR (Cod\_Setor) são chaves estrangeiras em GERENTE (Cod\_Funcionario e Cod\_Setor). A relação GERENTE possui como atributos um código que identifica de forma única suas tuplas e o atributo Data\_Inicio do

relacionamento que originou a tabela.

**Passo 3: Mapeamento dos Conjuntos de Relacionamentos binários de razão de cardinalidade 1:N** - Em primeiro , devemos gerar uma tabela para cada um dos conjuntos de entidades conforme descrito em nosso primeiro passo. Para o mapeamento 1:N a relação que mapeia o Conjunto de entidades do lado N recebe a chave primária do outro Conjunto de Entidades (lado 1) como chave estrangeira e os atributos do relacionamento. Vejamos o exemplo abaixo na figura 3.13:

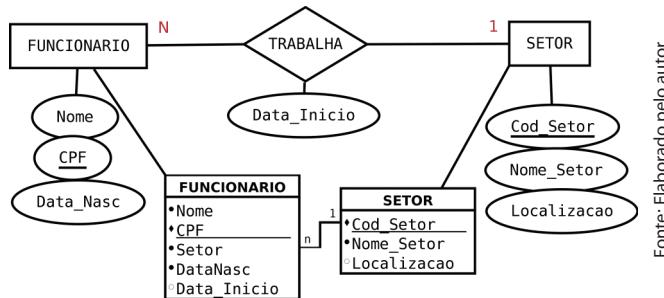


Figura 3.13: Mapeamento para relação 1:N.

Fonte: Elaborado pelo autor

Como podemos observar na figura 3.13, o diagrama ER apresenta as entidades FUNCIONARIO e SETOR com relacionamento 1:N, ou seja, um setor pode possuir vários funcionários. Para o mapeamento deste relacionamento, foi incluído na tabela gerada pela entidade do lado N (FUNCIONARIO) a chave primária da tabela gerada pela entidade do lado 1 (SETOR), desta forma a tabela FUNCIONARIO possui a chave primária de SETOR (Cod\_Setor) como sua chave estrangeira (Setor), além de possuir também os atributos da relação TRABALHA (Data\_Inicio).



Desta maneira, poderíamos ter as seguintes tabelas no banco de dados que foi mapeado a partir do diagrama apresentado na figura 3.14:

FUNCIONARIO				
NOME	CPF	SETOR	DATA_NASC	Data_Inicio
Jose	33875419294	1	10/05/1980	01/01/2000
Carlos	26218188892	2	17/05/1970	01/05/2005

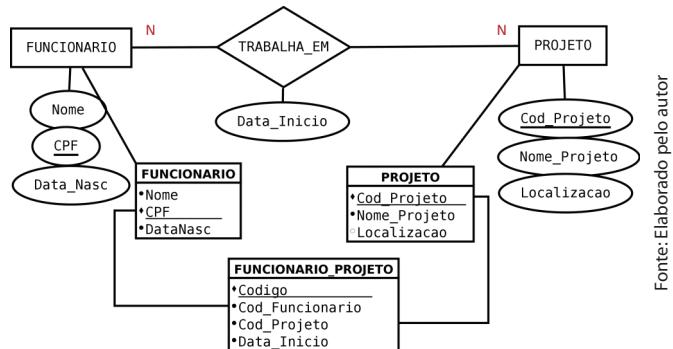
Setor		
Cod_Setor	Nome	Localizacao
1	Informatica	Andar 4
2	RH	Térrio

Fonte: Elaborado pelo autor

Figura 3.14: Tabelas geradas pelo mapeamento da relação 1:N.

As tabelas acima mostram qual funcionário trabalha em cada setor. Sabemos então que José trabalha no setor de Informática desde 01/01/2000.

**Passo 4: Mapeamento dos Conjuntos de Relacionamentos binários de razão de cardinalidade N:M:** Devemos criar uma nova tabela para representar o relacionamento. Nesta nova tabela devemos incluir como chave-estrangeira as chaves-primárias das tabelas que representam os tipos de entidade participantes; sua combinação irá formar a chave-primária desta nova tabela. Vejamos um exemplo na figura 3.15.



Fonte: Elaborado pelo autor

Figura 3.15: Exemplo mapeamento da relação N:M.

Nosso exemplo apresentado na figura 3.15 acima representa uma relação onde um PROJETO pode ter vários funcionários trabalhando e um funcionário pode trabalhar em vários projetos. No mapeamento deste relacionamento foi criada a relação FUNCIONARIO\_PROJETO. Esta relação contém os atributos do relacionamento (no caso Data\_Inicio) e as chaves primárias das relações FUNCIONARIO e PROJETO (Cod\_Funcionario e Cod\_Projeto) como chave estrangeira.

Vejamos um conjunto de tabelas de um banco de dados que foi mapeado a partir do diagrama apresentado na figura 3.16:

FUNCIONARIO		
NOME	CPF	DATA_NASC
José	33875419294	10/05/1980
Carlos	26218188892	17/05/1970

PROJETO		
Cod_Projeto	Nome_Projeto	Localizacao
1	Informatica para todos	Sala 2
2	Melhoria de trabalho	Sala 1

FUNCIONARIO_PROJETO		
Codigo	Cod_Projeto	Cod_Funcionario
1	1	1
2	1	2
3	2	2
4	2	1

Fonte: Elaborado pelo autor

Figura 3.16: Tabelas geradas pelo mapeamento da relação N:M.

Na figura 3.16 podemos observar que na tabela FUNCIONARIO\_PROJETO podemos saber qual funcionário trabalha em qual projeto. Por exemplo, no projeto Informática para todos trabalha os funcionários José e Carlos. Desta maneira podemos ver que um projeto pode ter vários funcionários, assim como um funcionário pode fazer parte de vários projetos. Como chave primária da tabela FUNCIONARIO\_PROJETO criamos um novo campo Código, poderíamos ainda ter identificado esta tabela através de uma chave composta pelas duas chaves estrangeiras da relação (Cod\_Funcionário e Cod\_Projeto).

**Passo 5: Mapeamento de Relacionamentos N-ários:** Para os relacionamentos de grau maior que dois devemos criar uma nova relação (tabela) para representar o relacionamento. Temos que incluir nesta tabela como chave-estrangeira as chaves-primárias das relações que representam os tipos de entidades participantes. Temos que incluir também qualquer atributo simples do tipo de relacionamento n-ário. A chave-primária desta relação é normalmente uma combinação de todas as chaves-estrangeiras fazendo referência às relações que representam os tipos de entidades do relacionamento

## RESUMO DA AULA

Nesta semana nos aprofundamos no Modelo de Dados Relacional, vimos seus conceitos fundamentais e como um projeto de um esquema conceitual do modelo ER pode ser mapeado para um banco de dados relacional. Vimos também conceitos importantes como o de chave estrangeira, domínio e integridade referencial. Foi ilustrado por meio de exemplos os passos necessários para o mapeamento entre MER x MRel.

## ATIVIDADES DE APRENDIZAGEM

1. Defina os conceitos de:
  - a. Chave-primária;
  - b. Chave-composta;
  - c. Domínio de uma relação;
  - d. Integridade referencial.

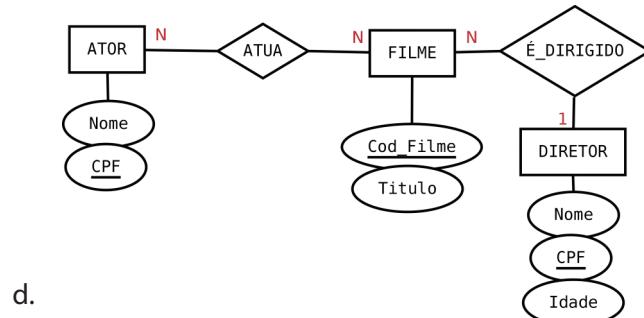
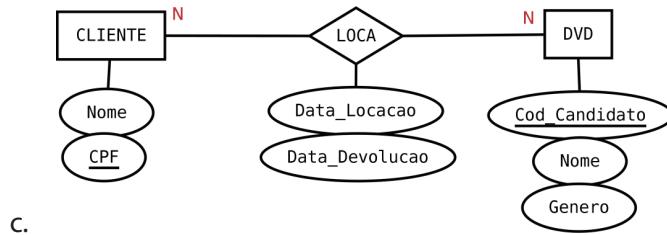
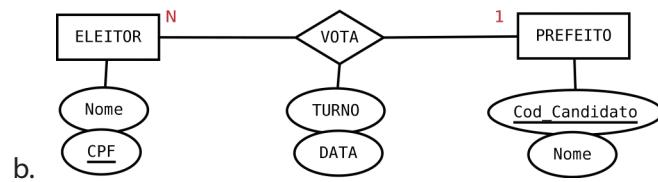
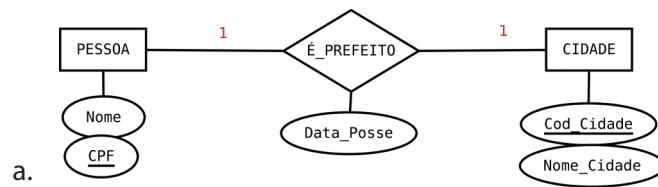
2. O que significa efetuar mapeamento entre MER e MRel?

---

---

---

3. Efetue o mapeamento para os diagramas ER abaixo:



# AULA IV - Implementação do Banco de Dados

## Objetivo:

Apresentar os conceitos necessários para implementação de um banco de dados em um Sistema de Gerenciamento de Banco de Dados que implemente o modelo relacional.

Agora que já vimos como efetuar o mapeamento entre o modelo ER para o MRel iremos realizar a implementação de um banco de dados em um SGBD.

### 4.1. MySQL

Em nossa disciplina utilizaremos o SGBD Relacional MySQL. Este SGBD é gratuito e possui código aberto. É comum a utilização deste SGBD em aplicações para ambiente WEB como é nosso escopo.

Como foi solicitado anteriormente, é necessário o download e instalação do MySQL. Veja a área do desenvolvedor na figura abaixo.

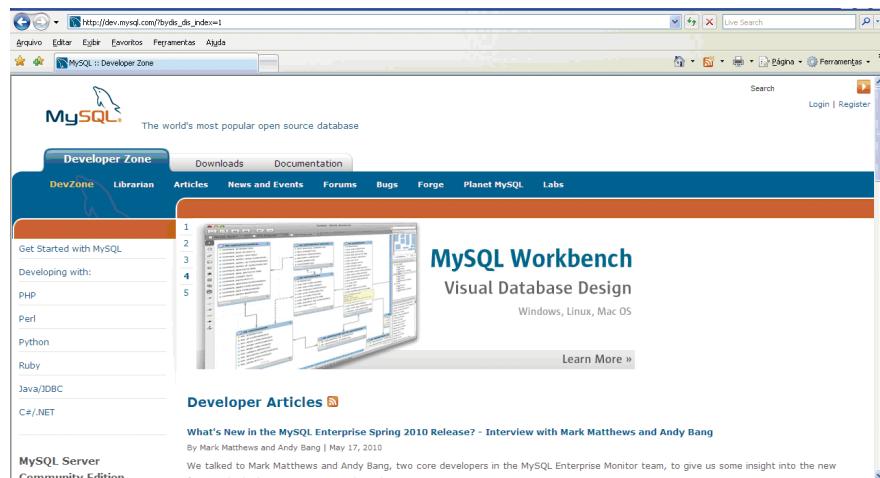


Figura 4.1 – Site MySQL



Veja mais sobre o MySQL em <http://dev.mysql.com/doc/refman/4.1/pt/index.html>, esta documentação é referente a versão 4.1 (português), a documentação das versões mais atuais (5.X) podem ser acessadas em inglês.

Navegue pelo site para conhecer seu conteúdo, documentações, área de download, fóruns, artigos entre outras opções. Para apoio na instalação do SGBD MySQL utilize o exemplo multimídia disponível em nosso ambiente.

No ambiente virtual de aprendizagem está disponível um vídeo que explica passo a passo para a instalação e configuração do MySQL.

Para que possamos criar nosso banco de dados com suas tabelas, utilizaremos um conjunto de comandos que compõe a linguagem de definição de dados DDL, como vimos em nossa primeira aula. Estes comandos possuem uma série de cláusulas. Em nossa disciplina utilizaremos somente as cláusulas para a construção de nosso banco de dados. Utilize nosso livro de referência para conhecer mais sobre as cláusulas de definição de um banco de dados.

## 4.2. CRIANDO UM BANCO DE DADOS

Para que possamos iniciar a criação de um banco de dados vamos utilizar um exemplo.

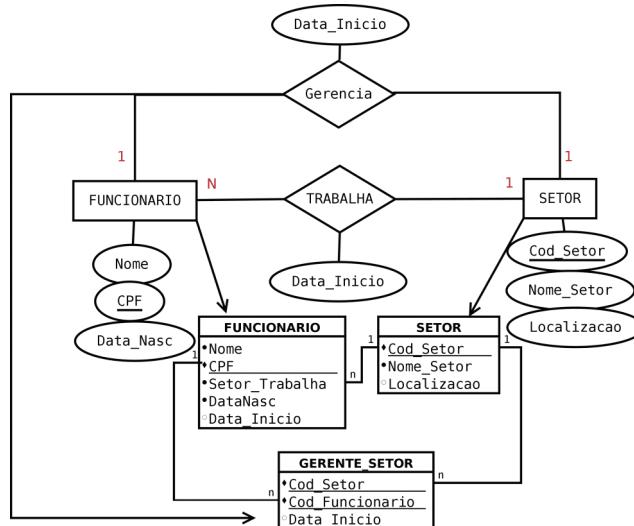


Figura 4.2 – MER e MRel do banco de dados Exemplo

Vamos analisar a figura 4.2, no MER podemos ver que possuímos duas entidades que possuem dois relacionamentos, um relacionamento 1:N e outro 1:1, seguindo nossos passos para mapeamento foram geradas as três tabelas apresentadas.

Para iniciarmos a criação de nosso banco, vamos abrir o MySQL através do MySQL Command Line Client.



Figura 4.3 – Iniciando o cliente de linha de comando do MySQL

Ao abrir o cliente de linha de comando do MySQL, a senha que definiu no momento da instalação será solicitada para acesso.

Todo comando na interface deve ser finalizado com ; (ponto e vírgula).



Após abrirmos o cliente de comando teremos a seguinte tela apresentada na figura 4.4

```
MySQL Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.1.45-community MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Figura 4.4 – Tela MySQL

Nosso primeiro passo será a criação de um banco de dados para abrigar nossas tabelas geradas a partir do mapeamento.

Para criarmos um banco de dados devemos utilizar o seguinte comando:

**CREATE DATABASE <nome>;**

Onde <nome> é a identificação de nosso banco de dados, nosso primeiro banco de dados se chamará exemplo, então o comando utilizado será:

**CREATE DATABASE exemplo;**

Após escrever a instrução acima tecle <Enter> e o MySQL apresentará a mensagem <Query OK, 1 row affected> informando que a criação foi realizada com sucesso. Vejamos a figura 4.5:

```
MySQL Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.1.45-community MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE exemplo;
Query OK, 1 row affected (0.00 sec)

mysql>
```

Figura 4.5 – Criação de um banco de dados

Para visualizarmos os bancos de dados existentes em nosso sistema devemos utilizar o seguinte comando:

SHOW DATABASES;

```
mysql> CREATE DATABASE exemplo;
Query OK, 1 row affected (0.00 sec)

mysql> SHOW DATABASES
-> ;
+-----+
| Database |
+-----+
| information_schema |
| exemplo |
| mysql |
| test |
+-----+
4 rows in set (0.00 sec)

mysql> _
```

Figura 4.6 – Execução do comando SHOW DATABASES

Podemos visualizar na figura 4.6 que nosso banco de dados exemplo foi realmente criado, os outros bancos que aparecem são de estruturas internas ou exemplos do MySQL.

O próximo passo será a criação das tabelas de nosso banco de dados. Antes disso devemos dizer que nosso banco de dados deve ser utilizado para a criação de nossas tabelas. Vamos executar o seguinte comando:

USE exemplo;

Desta maneira, o banco de dados exemplo que criamos anteriormente passa a ser utilizado, assim todas as tabelas que forem criadas farão parte do banco de dados em uso.

### 4.3. TIPOS DE DADOS

A instrução para criação de tabelas é o CREATE TABLE. Vamos ver a sintaxe padrão:

```
CREATE TABLE <nome_tabela> (
<nome_coluna> <tipo_dado> <NULL/NOT NULL> <DEFAULT valor>
<AUTO_INCREMENT>, <PRIMARY KEY (nome(s)_atributo(s))>,
<FOREIGN KEY (nome_atributo) REFERENCES <nome_tabela> > );
```

O tipo de dado apresentado na instrução acima se refere à informação que será armazenada, por exemplo, se formos armazenar o nome de uma pessoa, este tipo de dado será uma cadeia de caracteres. A linguagem SQL possui vários tipos de dados, que podemos classificar em três grupos:

1. Tipos numéricos
2. Tipos de Data
3. Tipos de Cadeia

Vejamos abaixo os principais tipos de dados (CRIAR WEB, 2010):

#### TIPOS NUMÉRICOS:

Existem tipos de dados numéricos, que se podem dividir em dois grandes grupos, os que estão em vírgula flutuante (com decimais) e os que não (inteiros) como já viram anteriormente na disciplina de algoritmos.

**TinyInt:** é um número inteiro com ou sem sinal. Com sinal a margem de valores válidos é desde -128 até 127. Sem signo, a margem de valores é de 0 até 255.

**SmallInt:** número inteiro com ou sem sinal. Com sinal a margem de valores válidos é desde -32768 até 32767. Sem signo, a margem de valores é de 0 até 65535.

**Integer, Int:** número inteiro com ou sem sinal. Com sinal a margem de valores válidos é desde -2147483648 até 2147483647. Sem signo, a margem de valores é de 0 até 429.496.295

**BigInt:** número inteiro com ou sem sinal. Com sinal a margem de valores válidos é desde -9.223.372.036.854.775.808 até 9.223.372.036.854.775.807. Sem signo, a margem de valores é de 0 até 18.446.744.073.709.551.615.

**Float:** número pequeno em vírgula flutuante de precisão simples. Os valores válidos vão desde -3.402823466E+38 até -1.175494351E-38,0 e desde 175494351E-38 até 3.402823466E+38.

**Double:** número em vírgula flutuante de dupla precisão. Os valores permitidos vão desde -1.7976931348623157E+308 até -2.2250738585072014E-308, 0 e desde 2.2250738585072014E-308 até 1.7976931348623157E+308.

#### TIPOS DATA:

Para armazenar datas, o MySQL verifica se o mês está compreendido entre 0 e 12 e que o dia está compreendido entre 0 e 31.

**Date:** tipo data armazena uma data. A margem de valores vai desde o 1 de Janeiro de 1001 ao 31 de dezembro de 9999. O formato de armazenamento é de ano-mes-dia.

**DateTime:** Combinação de data e hora. A margem de valores vai desde o 1 ed Janeiro de 1001 às 0 horas, 0 minutos e 0 segundos ao 31 de Dezembro de 9999 às 23 horas, 59 minutos e 59 segundos. O formato de armazenamento é de ano-mes-dia horas:minutos:segundos

**Time:** armazena uma hora. A margem de horas vai desde -838 horas, 59 minutos e 59 segundos. O formato de armazenamento é 'HH:MM:SS'.

#### TIPOS DE CADEIA:

Para armazenar cadeias de caracteres, os dois principais tipos são:

**Char(n):** armazena uma cadeia de longitude fixa. A cadeia poderá conter desde 0 até 255 caracteres.

**VarChar(n):** armazena uma cadeia de longitude variável. A cadeia poderá conter desde 0 até 255 caracteres. A diferença para o tipo CHAR é que os caracteres em branco a direita são desprezados. Por isso, quando formos armazenar uma cadeia de caracteres utilizaremos este tipo.

## 4.4. CRIANDO TABELAS

A partir dos tipos de dados básicos que acabamos de conhecer, vamos à construção de nossa tabela. Vamos voltar a nossa instrução básica para a criação de tabelas:

```

CREATE TABLE <nome_tabela> (
<nome_coluna> <tipo_dado> <NULL/NOT NULL> <DEFAULT valor>
<AUTO_INCREMENT>, <PRIMARY KEY (nome(s)_atributo(s))>,
<FOREIGN KEY (nome_atributo) REFERENCES <nome_tabela> > );

```

Vejamos o significado das instruções que compõem o comando acima, <nome\_tabela> é nome da tabela que vamos criar; <nome\_coluna> é o campo ou atributo a ser definido; <tipo\_dado> como vimos anteriormente se refere ao tipo de dado a ser armazenado; <NULL/NOT NULL> define se o campo irá aceitar valores nulos; <DEFAULT valor> determina um valor padrão para o campo; <AUTO\_INCREMENT> define se o campo será preenchido automaticamente, sendo muito útil para códigos ou campos de identificação. <PRIMARY KEY> e <FOREIGN KEY> determinam as chaves primárias e estrangeiras da tabela respectivamente. Vamos agora montar o esquema para criação de nossa primeira tabela SETOR.

TABELA: Setor		
CAMPO	TIPO	DESCRIÇÃO
Cod_Setor	Integer	Código do Setor (não nulo)
Nome_Setor	VARCHAR (40)	Nome do Setor (não nulo)
Localizacao	VARCHAR (30)	Localização do Setor
Chave Primária: Cod_Setor		

Figura 4.7: Dicionário de dados para a tabela SETOR



Ao esquema apresentado na figura 4.7 acima, damos o nome de Dicionário de Dados, embora apresentado de maneira simplificada, o dicionário de dados é extremamente importante para a documentação de nosso banco dados a ser criado.

Para criarmos nossa tabela SETOR vamos utilizar então a seguinte instrução:

```
CREATE TABLE setor (
    Cod_Setor      INTEGER NOT NULL,
    Nome_Setor    VARCHAR(40) NOT NULL,
    Localizacao   VARCHAR(30),
    PRIMARY KEY (Cod_Setor
);
```

A figura 4.8 mostra o resultado da instrução de criação da tabela SETOR.

```
mysql> USE exemplo;
Database changed
mysql> CREATE TABLE setor <
->     Cod_Setor      INTEGER NOT NULL,
->     Nome_Setor    VARCHAR(40) NOT NULL,
->     Localizacao   VARCHAR(30),
->     PRIMARY KEY (Cod_Setor)
-> >;
Query OK, 0 rows affected (0.25 sec)
```

Figura 4.8: Criação da tabela SETOR

Para visualizarmos as tabelas que nosso banco de dados possui, basta utilizarmos o comando [SHOW TABLES](#). Para visualizarmos a estrutura da tabela, devemos utilizar o comando [DESCRIBE <nome\\_tabela>](#).

```
mysql> SHOW TABLES;
+-----+
| Tables_in_exemplo |
+-----+
| setor             |
+-----+
1 row in set (0.16 sec)

mysql> DESCRIBE setor;
+-----+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key  | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| Cod_Setor | int(11) | NO   | PRI  | NULL    |       |
| Nome_Setor | varchar(40) | NO   |       | NULL    |       |
| Localizacao | varchar(30) | YES  |       | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.06 sec)

mysql> _
```

Figura 4.9: Resultado da execução das instruções SHOW TABLES e DESCRIBE

Agora vamos dar sequência na criação de nossas tabelas. Temos que criar as tabelas FUNCIONARIO e GERENTE\_SETOR. Vamos criar a tabela FUNCIONARIO. Nesta tabela teremos que implementar uma chave primária (CPF) e uma chave estrangeira (Setor\_Trabalha). Vejamos o dicionário de dados para a tabela FUNCIONARIO.

TABELA: Funcionário		
CAMPO	TIPO	DESCRIÇÃO
CPF	BigInt	CPF do funcionário (não nulo)
Nome	VARCHAR (40)	Nome do Funcionário (não nulo)
Setor_Trabalha	Integer	Código do Setor onde trabalha
DataNasc	Date	Data de Nascimento
DataInicio	Date	Data de Início de Trabalho
Chave Primária CPF		
Chave Estrangeira: Setor_Trabalha referencia Cod_Setor em SETOR		

Figura 4.10: Dicionário de dados para a tabela FUNCIONARIO

Podemos observar na figura 4.10 que o tipo de dado para o campo CPF é **BigInt**, isto se fez necessário pois o tipo inteiro não comportaria o CPF de uma pessoa, pois possui 11 dígitos.

Vejamos agora as instruções necessárias para implementarmos a tabela FUNCIONARIO.

```

CREATE TABLE funcionario (
    CPF      BIGINT NOT NULL,
    Nome     VARCHAR(40) NOT NULL,
    Setor_Trabalha   INT,
    DataNasc    DATE,
    DataInicio   DATE,
    PRIMARY KEY (CPF),
    FOREIGN KEY (Setor_Trabalha) references SETOR (Cod_Setor)
);

```

Observe no código acima que o comando **FOREIGN KEY** é responsável pela definição da **chave estrangeira**. Esta cláusula informa que Setor\_Trabalha referencia o campo Cod\_Setor da tabela SETOR. Assim, os valores permitidos para o atributo chave estrangeira Setor\_Trabalha da tabela FUNCIONARIO devem encontrar correspondência aos valores do atributo Cod\_Setor, chave primária, da tabela SETOR.

Vejamos o resultado da execução da instrução acima na figura 4.11.

```
mysql> CREATE TABLE funcionario (
->     CPF      BIGINT NOT NULL,
->     Nome     VARCHAR<40> NOT NULL,
->     Setor_Trabalha INT,
->     DataNasc   DATE,
->     DataInicio DATE,
->     PRIMARY KEY      (CPF),
->     FOREIGN KEY    (Setor_Trabalha) references SETOR (Cod_Setor)
-> );
Query OK, 0 rows affected (0.19 sec)

mysql>
```

Figura 4.12: Dicionário de dados para a tabela GERENTE\_SETOR

Podemos observar na figura 4.12 que a chave primária desta relação é uma chave composta por dois atributos (Cod\_Setor e Cod\_Funcionario), e esta tabela também possui duas chaves estrangeiras. Observe também que o campo Cod\_Funcionario é do tipo BlgInt pois o campo a qual ele se relaciona é deste tipo (CPF).

Vejamos as instruções para implementação da tabela GERENTE\_SETOR.

```
CREATE TABLE gerente_setor (
    Cod_Setor          INT NOT NULL,
    Cod_Funcionario   BIGINT NOT NULL,
    DataInicio         DATE,
    PRIMARY KEY        (Cod_Setor, Cod_Funcionario),
    FOREIGN KEY        (Cod_Setor) references SETOR (Cod_Setor) ,
    FOREIGN KEY        (Cod_Funcionario) references FUNCIONARIO (CPF)
);
```

Podemos observar no código acima que foram necessárias duas linhas para definição das duas chaves estrangeiras, já a chave primária composta pode ser definida em uma única linha. Vejamos na figura 4.13 o resultado da execução desta seqüência de instruções.

```
mysql> CREATE TABLE gerente_setor <
->     Cod_Setor      INT NOT NULL,
->     Cod_Funcionario BIGINT NOT NULL,
->     DataInicio     DATE,
->     PRIMARY KEY    <Cod_Setor, Cod_Funcionario>,
->     FOREIGN KEY    <Cod_Setor> references SETOR <Cod_Setor>,
->     FOREIGN KEY    <Cod_Funcionario> references FUNCIONARIO <CPF>
-> ;
Query OK, 0 rows affected (0.16 sec)

mysql>
```

Figura 4.13: Resultado da execução das instruções para criação da tabela GERENTE\_SETOR

Agora vamos ver todas nossas tabelas criadas. Qual comando devemos utilizar? E para ver a estrutura das tabelas? Vejamos na figura 4.14

```
mysql> SHOW TABLES;
+-----+
| Tables_in_exemplo |
+-----+
| funcionario
| gerente_setor
| setor
+-----+
3 rows in set (0.00 sec)

mysql> DESCRIBE funcionario;
+-----+-----+-----+-----+-----+-----+
| Field   | Type    | Null | Key  | Default | Extra  |
+-----+-----+-----+-----+-----+-----+
| CPF     | bigint<20> | NO   | PRI   | NULL    |        |
| Nome    | varchar(40) | NO   |        | NULL    |        |
| Setor_Trabalha | int<11> | YES  | MUL   | NULL    |        |
| DataNasc | date    | YES  |        | NULL    |        |
| DataInicio | date   | YES  |        | NULL    |        |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.02 sec)

mysql>
```

Figura 4.14: Tabelas criadas no banco de dados EXEMPLO.

## 4.5. APAGANDO E ALTERANDO TABELAS

Uma tabela pode ser modificada devido a alguma necessidade que surja após sua criação. Podemos adicionar um campo, alterar a chave primária ou até mesmo modificar a estrutura. O comando utilizado é o [ALTER TABLE <tabela> <operação>](#):

Um exemplo seria se precisássemos adicionar um novo campo para armazenar o salário em nossa tabela FUNCIONARIO



A instrução para inserir o campo Salario na tabela FUNCIONARIO será a seguinte:

```
ALTER TABLE funcionario ADD salario FLOAT(10,2);
```

Podemos ver na instrução acima que adicionaremos o campo salário que vai ser do tipo float com tamanho dez e duas casas decimais.

Para vermos a alteração da tabela com o novo campo vamos utilizar o comando **DESCRIBE**.

```
mysql> ALTER TABLE funcionario ADD salario FLOAT(10,2);
Query OK, 0 rows affected (0.28 sec)
Records: 0  Duplicates: 0  Warnings: 0
mysql> DESCRIBE funcionario;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| CPF  | bigint(20) | NO  | PRI | NULL    |       |
| Nome | varchar(40) | NO  |     | NULL    |       |
| Setor_Trabalha | int(11) | YES | MUL | NULL    |       |
| DataNasc | date   | YES |     | NULL    |       |
| DataInicio | date   | YES |     | NULL    |       |
| salario | float(10,2) | YES |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.02 sec)

mysql>
```

Figura 4.15: Alteração da tabela FUNCIONARIO.

Por algum motivo pode ser necessário excluir uma tabela criada, para isto devemos utilizar a instrução **DROP TABLE <nome\_tabela>**. Se fosse necessário removermos a tabela FUNCIONARIO do banco de dados EXEMPLO, usamos o comando DROP:

```
DROP TABLE funcionario;
```

## RESUMO DA AULA

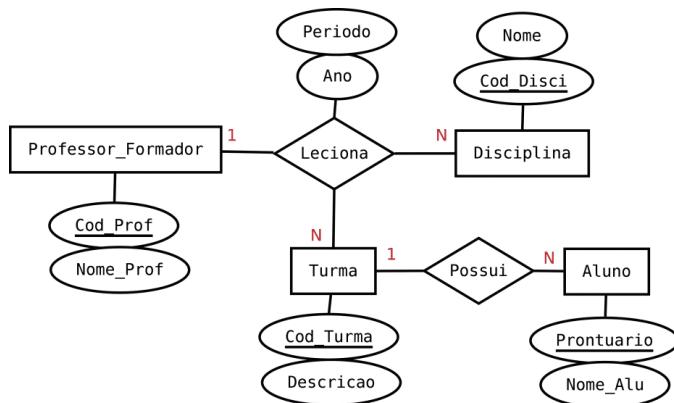
Nesta aula aprendemos a implementar um banco de dados em um SGBD relacional de auto nível. Vimos como criar nosso banco de dados e tabelas, implementamos chaves-primárias e estrangeiras em nossas tabelas. Criamos um banco de dados de exemplo com três tabelas relacionadas. Além disso, vimos como alterar e excluir uma tabela do banco de dados.

### Atividade de aprendizagem

1. Cite e explique as instruções para:
  - a) Criar um banco de dados;
  - b) Visualizar os bancos de dados que seu servidor possui;
  - c) Criar uma tabela;
  - d) Definir chave primária e estrangeira;
  - e) Visualizar os campos de uma tabela.

2. Crie um banco de dados para armazenarmos os dados dos alunos, professores formadores e disciplinas de nosso curso.

- O banco de dados vai se chamar TI (Técnico em Informática);
- Realize o mapeamento e crie as tabelas de acordo com o diagrama ER abaixo:



- Será necessária a criação das tabelas: PROFESSOR, DISCIPLINA, TURMA, LECIONA (para o relacionamento, contendo as chaves de cada uma das entidades para formar sua chave primária), e ALUNO.
- Envie os dicionários de dados de cada uma das tabelas, bem como as instruções necessárias para criação deste banco de dados.



# AULA V – Linguagem de Manipulação e Consulta de Dados

## Objetivos:

O Objetivo desta aula é apresentar as instruções necessárias para a manipulação de dados utilizando o ambiente do SGBD MySQL.

Na ultima aula, aprendemos a criar um banco de dados com suas tabelas e relacionamentos. Agora veremos como manipular os dados em nossas tabelas, inserindo, excluindo, atualizando e consultando.

### 5.1. INSERINDO REGISTROS

A partir do momento em que uma tabela está pronta, ela já pode receber dados. Vejamos a sintaxe da instrução INSERT INTO que é utilizada para inserção de dados em uma tabela.

```
INSERT INTO nome_tabela  
    (nome_coluna1, nome_coluna2,... ,nome_colunaN)  
VALUES  
    (Valor1, Valor2,....,ValorN)
```

Voltemos ao nosso banco de dados EXEMPLO. Vamos começar inserindo dados na tabela SETOR, para isso usaremos o comando **INSERT INTO**.

```
INSERT INTO setor (Cod_Setor, Nome_Setor, Localizacao)  
VALUES (1, 'Informatica', 'Segundo Andar');
```

Os valores textuais (cadeias de caracteres) para os campos Nome\_Setor e Localizacao devem estar entre aspas simples '.'



Na instrução **INSERT INTO**, após o nome da tabela, fazemos referência aos campos, e após, são indicados seus valores. A figura 5.1 mostra o resultado da instrução.

```
mysql> DESCRIBE setor;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| Cod_Setor | int(11) | NO | PRI | NULL |       |
| Nome_Setor | varchar(40) | NO |       | NULL |       |
| Localizacao | varchar(30) | YES |       | NULL |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.06 sec)

mysql> INSERT INTO setor (Cod_Setor,Nome_Setor,Localizacao)
-> VALUES (1,'Informatica','Segundo Andar');
Query OK, 1 row affected (0.09 sec)
```

Figura 5.1: Inserindo dados na tabela SETOR

A partir do exemplo que acabamos de ver acima vamos inserir mais dois registros em nossa tabela setor.

```
INSERT INTO setor (Cod_Setor, Nome_Setor, Localizacao)
VALUES (2, 'Financeiro', 'Térreo - Sala1');
```

```
INSERT INTO setor (Cod_Setor, Nome_Setor, Localizacao)
VALUES (3, 'Recepção', 'Térreo');
```

Caso tentemos incluir um valor duplicado no campo de chave primária da tabela será apresentado um erro de integridade. Vejamos na figura 5.2 apresentado abaixo.

```
mysql> INSERT INTO setor (Cod_Setor, Nome_Setor, Localizacao)
-> VALUES (1,'Informatica','Segundo Andar');
Query OK, 1 row affected (0.09 sec)

mysql> INSERT INTO setor (Cod_Setor, Nome_Setor, Localizacao)
-> VALUES (2, 'Financeiro', 'Térreo - Sala1');
Query OK, 1 row affected (0.05 sec)

mysql> INSERT INTO setor (Cod_Setor, Nome_Setor, Localizacao)
-> VALUES (2, 'Recepção', 'Térreo');
ERROR 1062 (23000): Duplicate entry '2' for key 'PRIMARY'
mysql>
```

Figura 5.2: Erro de integridade na tabela SETOR

Podemos observar na figura 5.2, que tentamos incluir o setor Recepção, com o mesmo Código do setor Financeiro, assim nos foi apresentado o erro: **Duplicata**.

Agora que nossa tabela SETOR possui dados, insira você, dados nas tabelas FUNCIONARIO e GERENTE\_SETOR.

Vejamos um exemplo de inserção de dados na tabela FUNCIONARIO:

```
INSERT INTO funcionario (CPF, Nome, Setor_Trabalha, DataNasc, DataNascimento, salario)
VALUES ('05598128691', 'Matheus Franco', '1983-11-14', '2000-01-01', 1250.00);
```

É importante lembrar que durante a inserção de dados é verificada a integridade referencial, assim ao inserirmos dados na tabela FUNCIONARIO, por exemplo, o campo Setor\_Trabalha deverá possuir como valor um dos valores já existentes no campo Código da tabela SETOR. Repare também, que para inserirmos dados do tipo Data os valores devem estar também entre aspas simples.

## 5.2. CONSULTANDO REGISTROS

A consulta de uma listagem de registros armazenadas em uma tabela é uma das funções mais importantes da linguagem de manipulação de dados, afinal, do que adianta termos dados armazenados se não pudermos consultá-los?

Para realizar a consulta aos dados de uma tabela, a linguagem **SQL** disponibiliza o comando **SELECT**. Este comando é a base para a recuperação de dados nos bancos de dados relacionais. Ele possui vários parâmetros, com diversas combinações possíveis e pode chegar a composições um tanto quanto complexas. Sua forma básica é:

```
SELECT < nome_coluna1, nome_coluna2,... , nome_colunaN>
      FROM <tabela>
      WHERE <condição para escolha dos registros>
```

A cláusula **SELECT** relaciona os campos (colunas) que queremos recuperar. A cláusula **FROM** relaciona as tabelas envolvidas na consulta. A cláusula **WHERE** permite a seleção de registros a recuperar a partir de uma condição.

Quando quisermos apresentar todas as colunas de uma tabela, ao invés de relacioná-las uma a uma na cláusula **SELECT**, podemos utilizar o \* (asterisco). Imagine que queiramos verificar a listagem de todos os setores cadastrados na tabela SETOR. Para isso utilizaremos a seguinte instrução.

```
SELECT * FROM setor;
```



SQL é uma linguagem padronizada para a definição e manipulação de bancos de dados relacionais.

Podemos observar na figura 5.3 que o uso do asterisco faz com que todos os campos da tabela sejam apresentados, como não tivemos cláusula de condição (WHERE), todos os registros da tabela foram apresentados.

```
mysql> SELECT * FROM setor;
+-----+-----+-----+
| Cod_Setor | Nome_Setor | Localizacao |
+-----+-----+-----+
| 1 | Informatica | Segundo Andar |
| 2 | Financeiro | Térreo - Salai |
| 3 | Recepção | Térreo |
+-----+-----+-----+
3 rows in set <0.00 sec>
mysql>
```

Figura 5.3: Consultando a tabela SETOR

Podemos listar também, somente alguns campos conforme a necessidade. Vejamos como selecionar somente os campos Nome e CPF da tabela FUNCIONARIO.

`SELECT nome,CPF FROM funcionario;`

```
mysql> SELECT nome,CPF FROM funcionario;
+-----+-----+
| nome | CPF |
+-----+-----+
| Matheus Franco | 5598128691 |
| Breno Caetano | 6698128691 |
| Cláudio Haruo | 7798128691 |
+-----+-----+
3 rows in set <0.02 sec>
```

Figura 5.4: Consultando campos da tabela FUNCIONARIO

Podemos observar na figura 5.4 que foram apresentados somente os campos definidos na cláusula SELECT.

Agora imaginemos que seja necessário visualizarmos o Nome\_Setor do setor cujo código seja igual a 1. Vejamos a instrução para isto.

`SELECT nome FROM setor WHERE cod_setor=1 ;`

```
mysql> SELECT * FROM setor;
+-----+-----+-----+
| Cod_Setor | Nome_Setor | Localizacao |
+-----+-----+-----+
| 1 | Informatica | Segundo Andar |
| 2 | Financeiro | Térreo - Salai |
| 3 | Recepção | Térreo |
+-----+-----+-----+
3 rows in set <0.00 sec>

mysql> SELECT nome_setor FROM setor WHERE cod_setor=1;
+-----+
| nome_setor |
+-----+
| Informatica |
+-----+
1 row in set <0.00 sec>
```

Figura 5.5: Apresentação do campo Nome\_Setor igual a 1 da tabela SETOR

Agora vamos imaginar que precisamos consultar o nome de um funcionário. Para a consulta de campos de caracteres, nem sempre é desejável a comparação exata do conteúdo de um campo, usando a igualdade. É muito comum a necessidade de se comparar parte desta cadeia. Para testar sequências de caracteres em uma cadeia, podemos usar a cláusula LIKE. Vejamos um exemplo:

```
SELECT CPF, nome, DataNasc FROM funcionario where nome LIKE 'M%';
```

```
mysql> SELECT CPF, nome, DataNasc FROM funcionario;
+-----+-----+-----+
| CPF | nome | DataNasc |
+-----+-----+-----+
| 5598128691 | Matheus Franco | 1983-11-14 |
| 6698128691 | Breno Caetano | 1983-05-05 |
| 7798128691 | Cláudio Haruo | 1980-05-09 |
| 72198128691 | Marcelo Silva | 1973-10-15 |
+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> SELECT CPF, nome, DataNasc FROM funcionario where nome like 'M%';
+-----+-----+-----+
| CPF | nome | DataNasc |
+-----+-----+-----+
| 5598128691 | Matheus Franco | 1983-11-14 |
| 72198128691 | Marcelo Silva | 1973-10-15 |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

Figura 5.6: Consultado dados com a cláusula LIKE

O carácter % dentro das aspas é interpretado pelo SGBD como delimitador da cadeia de caracteres a ser comparada. Assim, no exemplo, serão aceitas todas as cadeias de caracteres que possuírem M na primeira posição, independente dos demais caracteres.

Para testar uma seqüência de caracteres em qualquer posição de uma cadeia, usamos o carácter %.

### 5.2.1. DATAS

Normalmente os SGBD's fornecem um repertório de funções para o tratamento de datas, por exemplo, no MySQL temos as funções YEAR, MONTH e DAY, para consultarmos uma data a partir de um determinado ano, mês ou dia.

Vejamos um exemplo, vamos apresentar os nomes, data de nascimento e salário dos funcionários que nasceram em um determinado ano. Para isso utilizaremos o seguinte comando;

```
SELECT nome,datanasc,salario FROM funcionario WHERE
year(datanasc)=1983;
```

```
mysql> SELECT nome,datanasc,salario FROM funcionario;
+-----+-----+-----+
| nome | datanasc | salario |
+-----+-----+-----+
| Matheus Franco | 1983-11-14 | 1250.00 |
| Breno Caetano | 1983-05-05 | 1050.00 |
| Cláudio Haruo | 1980-05-09 | 1850.00 |
| Marcelo Silva | 1973-10-15 | 1950.00 |
+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> SELECT nome,datanasc,salario FROM funcionario WHERE year(datanasc)=1983;
+-----+-----+-----+
| nome | datanasc | salario |
+-----+-----+-----+
| Matheus Franco | 1983-11-14 | 1250.00 |
| Breno Caetano | 1983-05-05 | 1050.00 |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

Figura 5.7: Consultado datas

Como podemos ver na figura 5.7, com a utilização da função `YEAR` foram apresentados somente os registros cujo ano atendeu a condição. O mesmo pode realizar com as funções `MONTH` e `DAY`.

### 5.2.2. ORDENANDO REGISTROS

Para ordenação dos registros apresentados podemos utilizar a cláusula `ORDER BY`. Podemos ordenar a apresentação dos dados a partir de qualquer registro ou campo da seleção. Vejamos alguns exemplos

```
SELECT nome,salario FROM funcionario ORDER BY nome;
```

```
SELECT nome,salario FROM funcionario ORDER BY salario;
```

```
SELECT nome,salario FROM funcionario ORDER BY nome,salario;
```

Como podemos ver no último exemplo apresentado, podemos ter mais de um critério de ordenação, assim no exemplo, o primeiro critério é o campo Nome, e o segundo o Salário.

```
mysql> SELECT nome,salario FROM funcionario  
-> ORDER BY nome;  
+-----+-----+  
| nome | salario |  
+-----+-----+  
| Bruno Caetano | 1950.00 |  
| Cláudio Haruo | 1950.00 |  
| Marcelo Silva | 1950.00 |  
| Matheus Franco | 1250.00 |  
+-----+-----+  
4 rows in set <0.01 sec>  
  
mysql> SELECT nome,salario FROM funcionario  
-> ORDER BY salario;  
+-----+-----+  
| nome | salario |  
+-----+-----+  
| Bruno Caetano | 1950.00 |  
| Matheus Franco | 1250.00 |  
| Cláudio Haruo | 1950.00 |  
| Marcelo Silva | 1950.00 |  
+-----+-----+  
4 rows in set <0.00 sec>
```

Figura 5.8: Ordenação de registros

### 5.2.3. JUNÇÃO DE TABELAS

Através da junção de tabelas podemos gerar consultas com tabelas relacionadas como se fosse uma única tabela. Vamos pensar no nosso banco de dados exemplo, nele temos as tabelas SETOR e FUNCIONARIO. Imagine que precisamos apresentar todos os funcionários que trabalham no setor de informática, com o que sabemos até o momento seria necessário sabermos o código do setor. Mas se precisarmos apresentar o nome do setor e não o código. Como faremos isto, tendo em vista que os funcionários estão em uma tabela e o nome do setor em outra?

Através da junção das duas tabelas que pode ser realizada pelo relacionamento existente entre elas. Vejamos abaixo:

```
SELECT nome, nome_setor FROM funcionario  
INNER JOIN setor on setor.cod_setor = funcionario.setor_trabalha  
WHERE nome_setor='Informatica';
```

No exemplo apresentado acima é realizada a junção entre as tabelas SETOR e FUNCIONARIO através da utilização da cláusula **INNER JOIN** que realiza a ligação do campo Cod\_Setor com o campo Setor\_Trabalha das tabelas envolvidas. Desta maneira podemos selecionar o campo Nome que pertence a tabela FUNCIONARIO e o campo Nome\_Setor que pertence a tabela SETOR dos registros cujo Nome\_Setor seja igual a Informatica.

```
mysql> SELECT nome,nome_setor FROM funcionario  
-> INNER JOIN setor on setor.cod_setor = funcionario.setor_trabalha  
-> WHERE nome_setor='Informatica';  
+-----+-----+  
| nome | nome_setor |  
+-----+-----+  
| Matheus Franco | Informatica |  
| Breno Caetano | Informatica |  
| Marcelo Silva | Informatica |  
+-----+-----+  
3 rows in set <0.00 sec>
```

Figura 5.9: Junção de tabelas

### 5.3. REMOVENDO REGISTROS

Dentre as operações de manutenção de registros, temos a possibilidade de removermos registros que não sejam mais necessários. Vejamos agora como excluir linhas de uma tabela. Para isto precisaremos utilizar a instrução **DELETE FROM** conforme apresentado abaixo:

```
DELETE FROM <tabela> <condição para remoção>;
```

Fique atento ao utilizar a instrução **DELETE**, muito cuidado! Se não for especificada cláusula **WHERE**, que indica a condição para a seleção da tupla que será excluída, todos os dados da tabela serão eliminados! Por exemplo, o comando:

**DELETE FROM** setor; eliminará todos os registros da tabela!



É importante ressaltar também que no momento da tentativa de uma remoção é verificada a integridade referencial. Vejamos a figura abaixo.

```
mysql> USE exemplo;
Database changed
mysql> SELECT * FROM setor;
+-----+-----+-----+
| Cod_Setor | Nome_Setor | Localizacao |
+-----+-----+-----+
| 1 | Informatica | Segundo Andar |
| 2 | Financeiro | Térreo - Sala1 |
| 3 | Recepção | Térreo |
+-----+-----+-----+
3 rows in set (0.11 sec)

mysql> DELETE FROM setor WHERE cod_setor=1;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails ('exemplo'.funcionario', CONSTRAINT 'funcionario_ibfk_1' FOREIGN KEY ('Setor_Trabalha') REFERENCES 'setor' ('Cod_Setor'))
mysql>
```

Figura 5.10.: Tentativa de remoção – Integridade referencial

Podemos observar na figura 5.10 que o comando

**DELETE FROM setor WHERE cod\_setor=1**

gerou um erro devido a tentativa de remoção do registro 1, que possui chaves estrangeiras vinculadas a este registros, ou seja, temos registros da tabela FUNCIONARIO que possuem o campo Setor\_Trabalha=1;

Como não temos nenhum registro vinculado ao Cod\_Setor =2 vamos removê-lo. Vejamos o comando necessário para isto:

**DELETE FROM setor WHERE cod\_setor=2;**

Podemos observar o resultado desta instrução na figura 5.11 apresentada abaixo.

```
mysql> SELECT nome, setor_trabalha FROM FUNCIONARIO;
+-----+-----+
| nome | setor_trabalha |
+-----+-----+
| Matheus Franco | 1 |
| Breno Caetano | 1 |
| Cláudio Haruo | 3 |
+-----+-----+
3 rows in set (0.00 sec)

mysql> DELETE FROM setor WHERE cod_setor=2;
Query OK, 1 row affected (0.06 sec)

mysql> SELECT * FROM setor;
+-----+-----+-----+
| Cod_Setor | Nome_Setor | Localizacao |
+-----+-----+-----+
| 1 | Informatica | Segundo Andar |
| 3 | Recepção | Térreo |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

Figura 5.11: Remoção e listagem dos registros após remoção

Também podemos observar na figura 5.11 a listagem dos registros após a remoção do setor cujo Cod\_Setor=2.

## 5.4. ALTERANDO REGISTROS

Comumente é necessária a atualização de dados diretamente no banco. A alteração de dados é obtida através do comando **UPDATE** em conjunto com o comando **SET**, que possui a seguinte sintaxe:

```
UPDATE <tabela> SET <campo> = valor_novo,  
<campo2>=valor_novo,.....  
WHERE <condição para alteração>;
```

Por exemplo, vamos modificar a localização do setor 2.

```
mysql> SELECT * FROM setor;  
+----+-----+-----+  
| Cod_Setor | Nome_Setor | Localizacao |  
+----+-----+-----+  
| 1 | Informatica | Segundo Andar |  
| 3 | Recepção | Térreo |  
+----+-----+-----+  
2 rows in set (0.02 sec)  
  
mysql> UPDATE setor SET localizacao='Segundo Andar - Porta 10' WHERE cod_setor=1;  
Query OK, 1 row affected (0.05 sec)  
Rows matched: 1 Changed: 1 Warnings: 0  
  
mysql> SELECT * FROM setor;  
+----+-----+-----+  
| Cod_Setor | Nome_Setor | Localizacao |  

```

Figura 5.12: Apresentação dos registros antes e depois da alteração.

Podemos observar na figura 5.12 a listagem dos registros da tabela SETOR antes e depois da alteração da localização do setor 1.

Como vimos no comando DELETE, a única forma de termos certeza de que apenas uma linha será removida é especificar, na cláusula WHERE, o valor do atributo chave primária. O mesmo cuidado devemos tomar para alteração de valores em um registro.



## RESUMO DA AULA

Nesta aula conhecemos os principais comandos para manipulação de dados: INSERT, DELETE e UPDATE. Estes comandos para a inserção, exclusão e alteração de dados são definidos na linguagem de manipulação de dados ou DML (Data Manipulation Language) que pertence ao padrão de banco de dados relacionais. Além disso, vimos como ordenar dados e realizar junções entre tabelas. A partir de agora já estamos aptos a desenvolver um projeto de um banco de dados para os principais tipos de aplicações comerciais

## Exercícios de aprendizagem

1. Defina a função das seguintes instruções:

a. INSERT INTO:

b. SHOW TABLE:

c. SELECT FROM:

d. UPDADE SET:

e. DELETE FROM:

2. Qual a finalidade da junção de tabelas? Cite exemplos.

---

---

---

---

3. Observando as tabelas ALUNO, PROFESSOR\_FORMADOR e TURMA que já criou na aula passada

ALUNO	PRONTUARIO	NOME_ALU	TURMA
	1234	CARLOS SILVA	2
	2346	ANDRÉ RAMPAZO	1
	3452	LUIS GANACIN	1

PROFESSOR_FORMADOR	COD_PROF	NOME_PROF
	1	MATHEUS FRANCO
	2	CLÁUDIO HARUO
	3	FÁBIO JUSTO

TURMA	COD_TURMA	DESCRICAO
	1	Araraquara - 2009
	2	Franca - 2009

- a. Faça com que as tabelas contenham os dados apresentados.
- b. Tente remover o registro 1 da tabela TURMA. Foi possível, explique a ocorrência.
- c. Altere o nome do ALUNO pronturário 1234 para 'José Silva'.
- d. Apresente somente os nomes e prontuários dos alunos que estudam na turma 1.
- e. Apresente todos os nomes de professores formadores que se iniciam com 'M'.
- f. Faça uma seleção utilizando a junção de tabelas onde deverá ser apresentado o nome do aluno e a descrição de sua turma.

Apresente todas as instruções que utilizou para execução da atividade.





# AULA VI – Projeto Final da Disciplina

## Objetivos

O Objetivo desta aula é realizar a definição do projeto final da disciplina, bem como orientar seu desenvolvimento.

Como já vimos na disciplina de Análise e Projeto de Sistemas, um projeto consiste num esforço temporário com um objetivo pré-estabelecido, seja para criar um novo produto, serviço ou processo. Tem início, meio e fim bem definidos, duração e recursos limitados, com uma sequência de atividades relacionadas. Em nossa disciplina desenvolveremos um projeto de construção de um banco de dados relacional utilizando o SGBD MySQL.

### 6.1. PROJETO DE BANCO DE DADOS

Nosso projeto de banco de dados consiste na criação de uma base de dados para armazenar notas finais bimestrais de alunos de um curso, como por exemplo, o nosso. O Diagrama ER apresentado na figura 6.1 ilustra o banco de dados a ser criado.

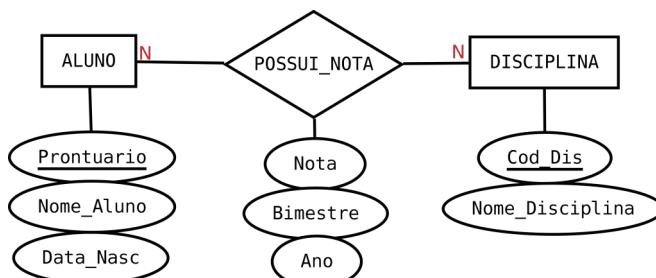


Figura 6.1: Diagrama ER do projeto final

De acordo com o diagrama apresentado na figura 6.1 podemos observar a existência de duas entidades (**ALUNO** E **DISCIPLINA**) e um relacionamento N:N (**POSSUI\_NOTA**). É importante observar que o relacionamento **POSSUI\_NOTA** gerará uma tabela onde sua chave primária deverá ser composta. Assim, de acordo com o diagrama você terá que implementar no banco de dados ESCOLA para armazenamento e recuperação das notas de alunos.

A figura 6.2 apresenta um esquema para desenvolvimento do projeto.

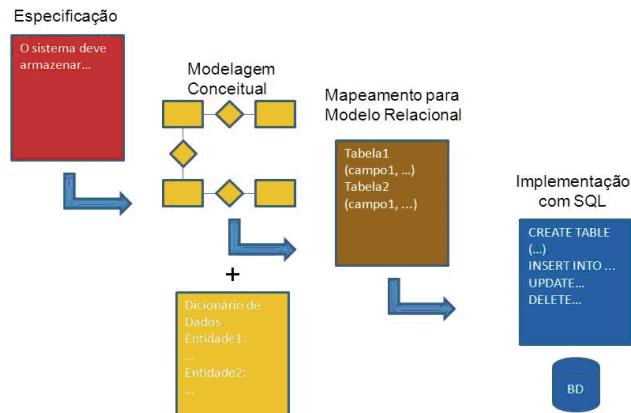


Figura 6.2: Esquema para desenvolvimento do projeto.

Assim, levando-se em consideração que já sabe o que fazer e possui o modelo conceitual (MER), você deverá seguir os seguintes passos:

- Realizar o mapeamento entre o modelo ER para o modelo Relacional;
- Criar os dicionários de dados para cada tabela;
- Criar as tabelas no SGBD MySQL;
- Inserir dados nas tabelas;
- Criar instruções de consulta para o banco de dados.

Como resultado do projeto deverá ser entregue um relatório contendo o mapeamento, dicionário de dados e todas as instruções necessárias para criação das tabelas, além das instruções necessárias para inserção e consulta de dados.

## REFERÊNCIAS

CRIAR WEB. Site Criar Web.

Disponível em <http://www.criarweb.com/artigos/118.php>. Acesso em maio de 2010.

ELMASRI, Ramez; NAVATHE, Shamkant B. Sistemas de banco de Dados. Ed. São Paulo: Addison, 2005.

MANZANO, José Augusto N. G. MySQL 5 Interativo. Guia básico de Orientação e Desenvolvimento. Ed. Érica, 1º Ed. 2007.

