



Banco de Dados

Sérgio de Sá Leitão Paiva Júnior
Ellen Polliana Ramos Souza

Curso Técnico em Informática

Educação a Distância

2016



EXPEDIENTE

Professor Autor

Sérgio de Sá Leitão Paiva Júnior
Ellen Polliana Ramos Souza

Design Instrucional

Deyvid Souza Nascimento
Maria de Fátima Duarte Angeiras
Renata Marques de Otero
Terezinha Mônica Sinício Beltrão

Revisão de Língua Portuguesa

Letícia Garcia

Diagramação

Izabela Cavalcanti

Coordenação

João Ferreira

Coordenação Executiva

George Bento Catunda

Coordenação Geral

Paulo Fernando de Vasconcelos Dutra

Conteúdo produzido para os Cursos Técnicos da Secretaria Executiva de Educação
Profissional de Pernambuco, em convênio com o Ministério da Educação
(Rede e-Tec Brasil).

Abril, 2016

P142b

Paiva Júnior, Sérgio de Sá Leitão.

Banco de Dados: Curso Técnico em Informática: Educação a distância / Sérgio de Sá Leitão Paiva Júnior ; Ellen Polliana Ramos Souza. — Recife: Secretaria Executiva de Educação Profissional de Pernambuco, 2016.

55 p.: il. tab.

Inclui referências bibliográficas.

1. Banco de dados — Gerenciamento — Programas de computador. 2. Projeto de banco de dados. 3. Software de aplicativos - Desenvolvimento. I. Paiva Júnior, Sérgio de Sá Leitão. II. Souza, Ellen Polliana Ramos . III. Título. IV. Rede e-Tec Brasil.

CDU – 004.658



Sumário

Introdução	5
1.Competência 01 Conhecer os Princípios de Banco de Dados	6
1.1 Abstração dos Dados.....	6
1.2 Visão dos Dados	9
1.3 Nível Lógico dos Dados	9
1.4 SGBD.....	10
1.5 Mysql e Phpmyadmin.....	12
1.6 SQL - Structured Query Language	13
2.Competência 02 Elaborar um Modelo Entidade-Relacionamento	20
2.1 Projeto de Banco de Dados.....	20
2.2 Modelagem de Dados Utilizando o Modelo Entidade-Relacionamento	22
2.3 Entidade	22
2.4 Atributo	23
2.5 Atributos Chave.....	25
2.6 Relacionamentos.....	26
2.7 Cardinalidade de Relacionamento	29
2.8 Criando um Modelo ER	31
2.9 MySQL Workbench	34
3.Competência 03 Construir Tabelas e Dicionários de Dados de um Banco De Dados	36
3.1 Minimundo Supermercado	36
3.2 Identificando as Entidades	36
3.3 Identificando os Atributos.....	37
3.4 Mapeando Relacionamentos	38
3.5 Atributos Chaves em um Relacionamento.....	39
3.6 Integridade de Dados	39
3.7 Representação Tabular	40



3.8 Dicionário de Dados	42
3.9 Criando Tabelas com a Linguagem SQL	44
3.10 Análise do Fluxo de Dados no Banco de Dados	46
3.11 Inserindo, Alterando e Removendo Registros com a Linguagem SQL	48
3.12 Melhorando as Consultas com a Linguagem SQL	51
Referências	54
Minicurrículo do Professor	55



Introdução

Caro (a) aluno (a),

Toda a teoria e prática que você aprenderá neste curso servirão de base para você projetar sistemas computacionais, dos simples aos mais complexos, capazes de manipular dados em grande escala com segurança e eficiência.

Banco de Dados é uma matéria que caminha de mãos dadas com outras disciplinas da Informática, em especial a disciplina de Programação. Alguns conceitos da disciplina de Lógica serão necessários, mas não se preocupe, pois serão devidamente lembrados quando estivermos estudando o assunto.

Nesta disciplina, você dará um passo importante, diria até fundamental, na sua formação. A palavra “informática” significa, num sentido restrito, processamento automático da informação. Pergunto a você: o que é a informação? Podemos dizer que informação é uma interpretação de dados. O banco de dados assume um papel importante quando é capaz de armazenar com segurança, eficácia e rapidez os dados que servirão de apoio à tomada de decisões em empresas e governos.

Este curso está estruturado em três competências: na primeira competência, você aprenderá os conceitos que fundamentam a área de banco de dados, além de já colocar alguns destes conceitos em prática. Na segunda competência, você estudará o modelo que permite a criação de bancos de dados a partir da análise do negócio em que você está trabalhando ou da descrição de um minimundo. Na terceira competência, iremos criar um banco de dados com todos os componentes estudados durante o curso.

Esperamos que esta experiência seja muito enriquecedora para você. Dedique-se com afinco a esta disciplina, pois ela poderá lhe proporcionar conhecimento essencial para um bom profissional da área de Informática.

1.Competência 01 | Conhecer os Princípios de Banco de Dados

Caro (a) aluno (a), vamos começar nossa primeira competência fornecendo uma visão do problema de estruturar, armazenar e recuperar dados em banco de dados. Nos próximos tópicos, abordaremos os conteúdos que construirão a base de que você necessitará para projetar um banco de dados para suas aplicações. Então, preste bastante atenção nos conceitos que seguem neste capítulo, pois um bom programa de computador começa por um bom projeto de banco de dados. Vamos iniciar apresentando os conceitos de abstração dos dados.

1.1 Abstração dos Dados

Antes de apresentarmos o conceito de abstração de dados, precisamos saber o que é um banco de dados. A Figura 1 apresenta um exemplo de banco de dados. Mas é claro que não é este o tipo de banco de dados que vamos estudar aqui.

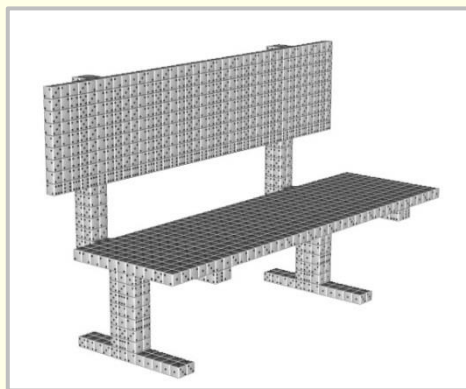


Figura 1 - Banco de dados.

Fonte: os autores (2014)

Descrição: Com objetivo cômico a figura mostra um “banco de praça” (assento) construído usando dados usados em jogos de cartas. Fonte: os autores (2014)

Um banco de dados (BD) é uma coleção de dados persistentes, estruturados e compartilhados por diversos usuários. Os bancos de dados são utilizados pelos sistemas de aplicação e são armazenados em disco. A Figura 2 apresenta um pequeno banco de dados que contém uma única tabela chamada ADEGA. Essa tabela, por sua vez, contém dados relativos ao conteúdo de uma adega de vinhos. A coluna ID# representa um identificador único para cada uma das linhas. A partir deste pequeno BD, o dono da adega tem conhecimento, por exemplo, que possui apenas uma garrafa do vinho Chardonay do produtor Buena Vista.



Banco de dados pode ser definido como um conjunto de dados integrados que tem por objetivo atender a uma comunidade de usuários.



ID#	VINHO	PRODUTOR	ANO	GARRAFAS
1	Chardonay	Buena Vista	1997	1
2	Chardonay	Geyser Peak	1997	5
3	Merlot	Simi	1996	4
4	Mertlot	Clini	1994	9
5	Blanc	Rafanelli	1995	2

Figura 2 - Banco de dados para adega de vinhos (tabela ADEGA).

Fonte: adaptado de Date (2000)

Descrição: Na tabela apresentada na imagem são dispostas as seguintes colunas: Id, vinho, produtor, ano e garrafas. A tabela possui cinco linhas, onde cada célula é preenchida com algum dado. Na coluna vinho, por exemplo, temos os tipos de vinho Chardonay, Merlot e Blanc. São cinco linhas, porém Chardonay e Merlot são repetidos em duas linhas cada. A repetição de dados não é um problema desde que tenha um identificador que diferencie as linhas, neste caso sendo o de nome Id.

Dentre as operações que os usuários podem realizar em um banco de dados estão: inserir, alterar, remover e buscar dados de tabelas existentes. Para realizar tais operações de forma mais simples, foram criados os Sistemas Gerenciadores de Banco de Dados (SGBD). Os SGBD fornecem meios de armazenamento, manipulação e recuperação dos dados armazenados em um banco de dados.



SGBD é um conjunto de aplicativos que incorpora funções de definição, alteração e recuperação de dados em um BD.

A manipulação dos dados é garantida pelo SGDB que “esconde” do usuário a forma como ele executa estas manipulações. Isto é chamado de **ABSTRAÇÃO DOS DADOS**, ou seja, a capacidade do SGBD de esconder detalhes da implantação. Na **Error! Reference source not found.**, as aplicações (em verde) são programas externos que necessitam acessar os dados armazenados em um BD. Os SGBD são responsáveis por controlar e facilitar este acesso.



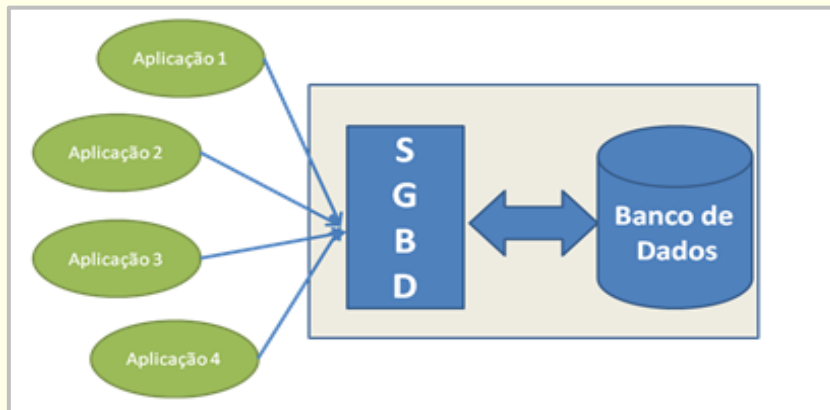


Figura 3 - Relacionamento entre o SGBD, aplicações e banco de dados.

Fonte: os autores (2014)

Descrição: Na figura podemos observar três tipos de componentes: Aplicações, o SGBD e o banco de dados em si. As aplicações fazem requisições ao SGBD que por sua vez controla o acesso aos dados e se encarrega de manipular toda a complexidade de realizar as transações.

O SGBD possui basicamente três níveis de abstração, conforme apresentado na Figura 1:

- **Nível físico:** descreve como os dados ficarão armazenados fisicamente no computador. As estruturas que armazenam e recuperam os dados do computador devem ser eficientes na busca e alteração de dados.
- **Nível lógico:** descreve quais os dados estão armazenados no computador e como eles se relacionam. Representa a realidade do problema, contendo a visão geral dos principais dados e relacionamentos.
- **Nível de visão:** o mais alto nível que descreve apenas partes dos dados. É o nível mais próximo do usuário. Este nível descreve quais porções do BD determinado grupo terá acesso. Existem muitas visões externas e uma visão descreve apenas parte do banco de dados.

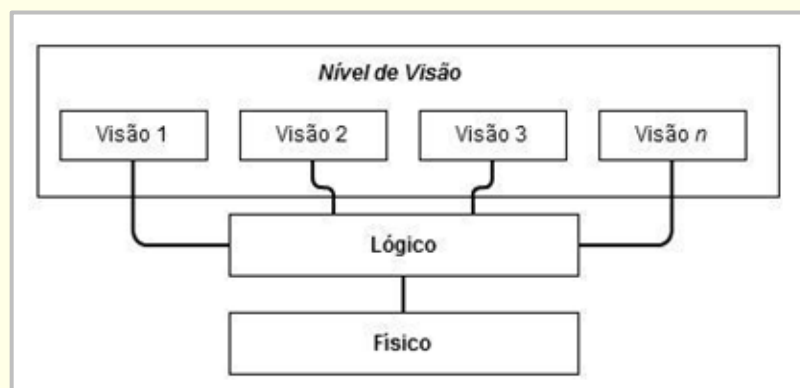


Figura 1 - Níveis de visão de um SGBD.

Fonte: extraído de (Certificação BD, 2013)

Descrição: A figura representa os três níveis de um SGBD. Olhando de baixo para cima temos primeiramente a parte física que descreve como os dados ficarão armazenados fisicamente no computador. Um nível acima temos a parte lógica que descreve quais os dados estão armazenados no computador e como eles se relacionam. Por último temos, conectado à parte lógica, várias visões que descrevem apenas partes dos dados dentro do que chamamos de Nível de Visão.



O nível de visão é responsável por apresentar os dados armazenados no BD, de acordo com a necessidade do usuário ou sistema. Na sequência, vamos estudar as diferentes visões dos dados.

1.2 Visão dos Dados

A primeira e maior missão de um SGBD é facilitar a vida dos usuários, fazendo com que eles (os usuários) vejam apenas o que necessitam, escondendo detalhes da organização interna dos dados. Esta é a chamada VISÃO DOS DADOS. Por exemplo, se você é um aluno do EAD e deseja verificar quais suas notas na disciplina de BD não faz sentido o sistema enviar para você o cadastro de todos os alunos do curso. Esta VISÃO que você deseja dos dados restringe-se apenas às suas notas.

Existem diversos níveis de visão de dados, como por exemplo: a visão do coordenador do curso precisa saber o nome de todos os alunos matriculados, os professores alocados por disciplina, as notas de todos os alunos do curso. Já um professor do curso necessita apenas visualizar os nomes dos alunos das suas turmas e acessar as notas dos alunos matriculados na sua disciplina.

Mas antes de visualizarmos os dados de que precisamos, estes precisam estar disponíveis em um BD. Para isso, criamos modelos que nos auxiliam na definição, relacionamentos, semântica e restrições desses dados. Vamos iniciar estudando o modelo do nível lógico.

1.3 Nível Lógico dos Dados

É o nível de abstração onde o usuário definirá todas as relações existentes entre os dados, primeiramente através de um projeto escrito no papel mesmo, depois através da implantação no SGBD. Para que você entenda melhor este conceito, imagine novamente o exemplo do nosso curso de EAD. Nele, alunos e professores estão relacionados uns aos outros, mas de que forma? O elo entre aluno e professor é uma disciplina. Veja esse exemplo na forma de um diagrama, apresentado na Figura 2. Assim, fica fácil perceber a relação que existe entre um aluno e um professor.

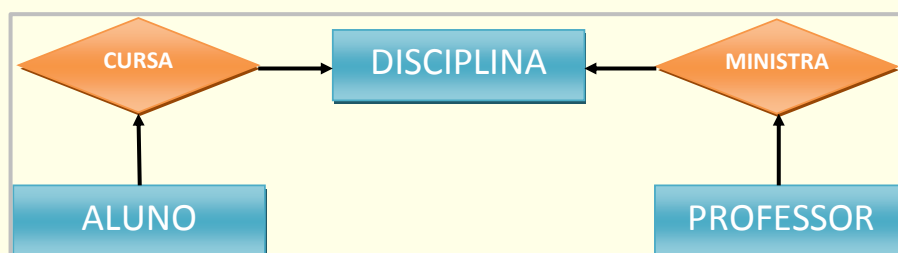


Figura 2 – Relação entre dados.

Fonte: os autores (2014)

Descrição: Nesta figura é apresentado um exemplo de modelo no nível lógico de banco de dados. Temos então três entidades (em retângulos) e dois relacionamentos (em losango). A entidade aluno está relacionada à entidade disciplina através do relacionamento cursa. Professor, por sua vez relaciona-se a disciplina pelo relacionamento ministra.



Modelo de dados é uma descrição formal da estrutura de um banco de dados (Heuser, 2004).

Observe que, a partir deste esquema lógico, podemos relacionar um determinado aluno a um professor, desde que o aluno esteja cursando uma disciplina ministrada por um professor. Os detalhes de implantação serão vistos adiante. No momento, saiba que é no nível de abstração lógico onde resolvemos todos os problemas de ligação ou relacionamento entre os dados.

Para definirmos o nível lógico a partir de como os dados estão interligados, precisamos utilizar um modelo de dados que possa juntar todos os dados de forma coerente. Um modelo é uma representação de alguma realidade. Por exemplo: João da Silva é aluno do curso de EAD, ele possui, além do seu nome, um endereço e um e-mail. Para representar João da Silva e todos os alunos do curso, poderíamos criar uma representação de conjunto no banco de dados onde cada elemento do conjunto é um aluno e cada um tem nome, endereço e e-mail como atributos. Essa representação seria o nosso modelo.

Existem vários modelos de dados e, neste curso, vamos utilizar o modelo Entidade-Relacionamento, também conhecido como MER. Na próxima competência, aprenderemos como projetar um banco de dados de forma coerente, sem redundância e sem inconsistências através do uso de modelos. O conjunto de regras ou conceitos que são aplicados sistematicamente aos dados até que o resultado seja um conjunto de dados é chamado de abordagem de modelagem.

Você está pronto para criar um banco de dados? Vamos agora conhecer o SGBD que utilizaremos para guardar dos dados do nosso BD.

1.4 SGBD

Como apresentado anteriormente, o SGBD é um conjunto de aplicativos e arquivos que tem como principal objetivo retirar da aplicação-cliente (um programa que requisita dados) a complexidade do acesso aos dados. Agora, você pode estar pensando: como o SGDB faz para disponibilizar essas informações? Um SGDB poderá rodar tanto dentro de um computador pessoal como numa grande rede de computadores. De uma forma ou de outra, o SGBD utiliza-se de uma interface que interpreta as informações do aplicativo para o banco de dados utilizando API's (Interface de Programação de Aplicativos), que são funções chamadas dentro de uma linguagem como Java, PHP, etc.; ou através de um DRIVER, que é um programa instalado no computador do cliente. A Figura 3 mostra um esquema desta arquitetura.

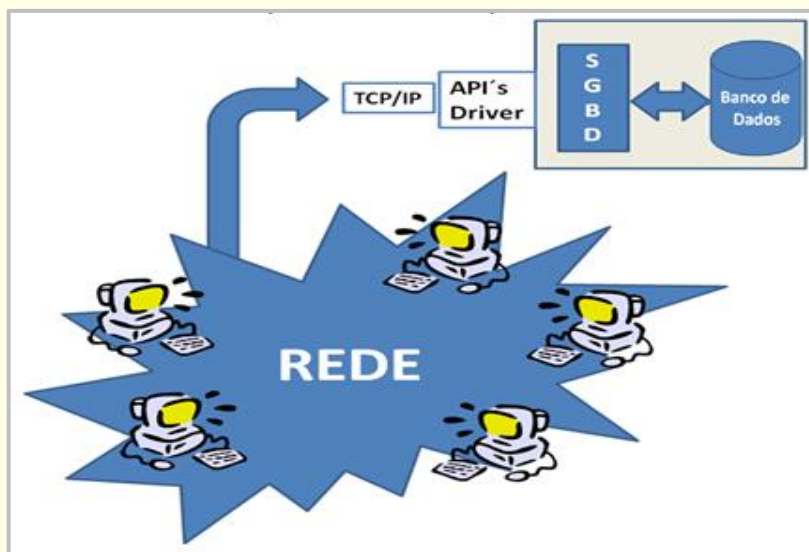


Figura 3 - Esquema de acesso a um SGBD.

Fonte: os autores (2014)

Descrição: O esquema representado nesta figura mostra vários computadores conectados por uma rede os quais podem acessar através de uma conexão TCP/IP um banco de dados. Vale ressaltar que é necessário haver o SGBD controlando tal acesso.

Através da API em PHP, por exemplo, você poderá acessar os dados do seu banco, para isto você deverá informar na API:

- Servidor através de um IP (endereço da rede);
- O *login* e senha de acesso;
- O nome da base que você deseja utilizar.

Veja um pequeno código em PHP que faz a conexão com um SGBD MySQL na Figura 4. Não se preocupe agora, você terá aulas sobre esse assunto numa disciplina posterior. Aqui, é importante você perceber que, para acessar o SGBD, você precisará de uma API que realizará todo o trabalho para você.

```
9
10 // nós conectamos com localhost na porta 3307
11 $link = mysql_connect('127.0.0.1:3307', 'mysql_user', 'mysql_password');
12 if (!$link) {
13     die('Não foi possível conectar: ' . mysql_error());
14 }
15 echo 'Conexão bem sucedida';
16 mysql_close($link);
```

Figura 4 - Código em PHP para acessar um SGBD.

Fonte: os autores (2014)

Descrição: Nesta imagem temos um pequeno código fonte em PHP usado para configurar dados específicos necessários ao acesso a um banco de dados. Neste caso o usuário configurou acesso a uma máquina local (ip 127.0.0.1) e o banco de dados mysql.



No mercado, há vários tipos de SGBD. Neste curso, vamos utilizar um tipo de SGBD, o RELACIONAL, que domina o mercado atualmente. Contudo, muitos dos conceitos que vamos estudar na segunda competência sobre modelagem de dados aplicam-se também a outros tipos, como o SGBD orientado a objeto ou objeto-relacionais. Vamos conhecer agora o SGBD reacional, chamado MySQL, que utilizaremos para criar o nosso BD.



No SGBD relacional, os dados são modelados de uma forma que eles sejam percebidos pelo usuário como tabelas e relações.

1.5 Mysql e Phpmyadmin

Neste curso, trabalharemos com o MySQL, que é um SGBD atualmente mantido pela ORACLE (que também tem um SGBD com nome Oracle). O MySQL possui uma licença GPL (software Livre), mas também tem uma licença para uso comercial. Após a instalação, você verá que o MySQL vem com alguns programas, dentre eles: o MySQL Query Browser, MySQL Administrator e MySQL Command Line.



Existem muitos SGBD. Os mais conhecidos são: PostgreSQL, Firebird, MySQL, Oracle e SQL-Server. Pesquise um pouco sobre eles.

O MySQL Query Browser é uma ferramenta gráfica para criar e executar solicitações SQL, utilizando um ambiente gráfico. Este aplicativo ajuda a analisar dados armazenados num Banco de dados MySQL. O MySQL Administrator é um aplicativo gráfico para realizar operações administrativas no banco de dados. Estas tarefas incluem gerenciar usuários, gerenciar conexões, fazer backup, etc. O MySQL Command Line é uma ferramenta para acessar o banco de dados por meio de uma interface de linha de comando.

Apesar de os programas cobrirem todas as necessidades de administração e acesso aos dados, eles exigem que o usuário tenha conhecimento em SQL para realizar muitas tarefas. Outras interfaces para o MySQL foram desenvolvidas com o intuito de facilitar o acesso. Vamos apresentar com detalhes uma GUI chamada *phpMyAdmin*, que é muito utilizada em diversos sites de administração de páginas que dão suporte ao MySQL.



Assista a nossa primeira videoaula e veja uma introdução ao phpMyAdmin



O phpMyAdmin é uma ferramenta de fácil utilização que serve para controlar o banco de dados MySQL. Essa ferramenta requer a instalação do servidor WEB Apache, do interpretador PHP, o SGBD MySQL, além do phpMyAdmin. Você poderá fazer as instalações utilizando o pacote pré-configurado no endereço <http://sourceforge.net/projects/phptriad/>. Outra boa opção é o Xampp, pacote contendo os mesmos programas do phptriad, porém com a vantagem de poder descompactar em um pendrive e levar para onde você quiser. Você poderá baixar e instalar o Xampp no link www.apachefriends.org/pt_br/xampp-windows.html



Servidor Web: são programas que podem responder a requisições de páginas no protocolo HTTP, geralmente solicitados por navegadores como o Google Chrome ou Internet Explorer. Como exemplo, temos o servidor WEB Apache



No Apêndice A deste caderno, você encontrará um guia rápido de instalação do Xampp

Por fim, para esconder os detalhes de implantação, os SGBD fazem uso de uma linguagem de programação própria para banco de dados. A ideia por trás disto é a seguinte: imagine que o nível físico seja um chinês e o nível lógico, um brasileiro. Acontece que nem o chinês fala português e nem o brasileiro fala chinês, porém os dois falam inglês, então, para haver comunicação, basta que falem na língua inglesa. Assim acontece com o SGBD. Como ele precisa integrar níveis tão diferentes, utiliza uma linguagem padrão. Se você entender esta linguagem, então você poderá manipular bem um banco de dados independente de SGBD. Na próxima seção, vamos estudar a linguagem utilizada para manipular dados em um BD.

1.6 SQL - Structured Query Language

SQL (Structured Query Language) é uma linguagem padrão de consulta para sistemas de banco de dados. Utilizada também para definição de estrutura de dados, modificação de dados no banco de dados e especificação de restrições de segurança. É importante salientar que a SQL não é a única linguagem de acesso a banco de dados, porém é a mais utilizada.

O que podemos fazer com a linguagem SQL? Podemos criar, alterar e remover esquemas lógicos. Podemos manipular de diversas formas os dados, inserindo dados novos ou removendo e alterando dados antigos do BD. Podemos restringir acesso ao banco de dados e ainda podemos controlar as transações dos usuários com o banco de dados. Em outras palavras, a SQL é poderosa o bastante para termos um completo domínio sobre os dados.



Uma linguagem SQL tem características semelhantes a uma linguagem de programação como Java ou C, porém, como é uma linguagem para manipular banco de dados, suas estruturas são específicas para este tipo de problema. Vamos ver algumas características que o SQL possui?

A SQL pode ser dividida em cinco subconjuntos, conforme apresentado na Figura 8. Neste curso, vamos estudar os subconjuntos DML, DQL e DDL.

DML ou Linguagem de Manipulação de Dados é o subconjunto da linguagem SQL que abrange comando para selecionar, inserir, atualizar e apagar dados. DQL ou Linguagem de Consulta de Dados contém o comando SELECT, composto de várias cláusulas e opções que possibilitam a criação de consultas mais elaboradas. DDL ou Linguagem de Definição de Dados abrange comandos para criação e alteração de tabelas, índices e visões. DCL ou Linguagem de Controle de Dados contém comandos de autorização de dados e licenças de usuários com o propósito de controlar quem tem acesso para ver ou manipular dados dentro do banco de dados. DTL ou Linguagem de Transação de Dados define comandos para especificar a iniciação e finalização de transações.

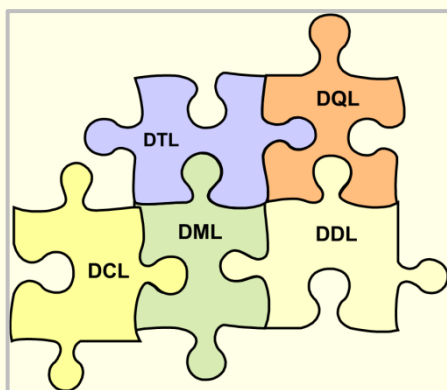


Figura 5 – subconjunto da linguagem SQL.

Fonte: os autores (2014)

Descrição: Nesta imagem temos as cinco subcategorias de comandos da linguagem SQL

(DTL, DQL, DCL, DML e DDL) representados por um quebra cabeça.

Como qualquer outra linguagem de programação, a SQL tem tipos de dados próprios. Veja na Figura 9 os principais tipos de dados utilizados pelo MySQL, um SGBD que utiliza SQL como linguagem de manipulação de dados.





	TIPO DE CAMPO	TAMANHO DE ARMAZENAMENTO
Numéricos	SMALLINT	2 bytes
	INT	4 bytes
	BIGINT	8 bytes
	FLOAT	4 bytes
	DOUBLE	8 bytes
	REAL	8 bytes
Data	DECIMAL(M,D)	M+2 bytes se D > 0, M+1 bytes se D = 0
	DATE	3 bytes
	DATETIME	8 bytes
	TIMESTAMP	4 bytes
Caractere	CHAR(n)	n bytes
	VARCHAR(n)	n+1 bytes

Figura 6 - Tipos de Dados usados no MySQL.

Fonte: os autores (2014)

Descrição: Nesta imagem temos duas colunas, a primeira indicando o tipo de campo utilizado (real, inteiro, palavra) e a segunda coluna informando o tamanho ocupado por cada um destes tipos na memória do computador. Como exemplo, podemos citar o tipo inteiro (ou int sendo mais específico) o qual possui o tamanho 4 bytes.

Operadores aritméticos, lógicos e relacionais são usados em SQL para realizar muitas tarefas, como somar e subtrair (operadores aritméticos), e comparar valores com valores fixos ou variáveis no banco de dados (operadores relacionais). Estes operadores ajudam a construir pesquisas mais exatas, estabelecendo diferenças entre os resultados. A Figura 10 mostra estes operadores.

OPERADOR	TIPO
+(soma)	Operador Aritmético
-(Subtração)	Operador Aritmético
*(Produto)	Operador Aritmético
/(Divisão)	Operador Aritmético
= (Igual)	Operador Relacional
> (Maior que)	Operador Relacional
< (Menor que)	Operador Relacional
>= (Maior igual)	Operador Relacional
<= (Menor igual)	Operador Relacional
<> (Diferente)	Operador Relacional
AND (e)	Operador Lógico
OR (ou)	Operador Lógico
NOT (não)	Operador Lógico

Figura 7 - Operadores usados no MySQL.

Fonte: os autores (2014)

Descrição: Na figura temos na primeira coluna os tipos de operadores, como soma, multiplicação, etc. Na segunda coluna temos os tipos correspondentes. Para o soma por exemplo o tipo é aritmético mas temos também tipos relacionais como o menor que e os tipos lógicos como o E e o OU.

Uma das principais atividades relacionadas a um banco de dados é a consulta aos dados. O SQL



possui um mecanismo muito eficiente de busca que permite ao usuário obter qualquer conjunto de dados das tabelas. A cláusula SELECT é utilizada para pesquisar dados com eficiência. Vamos agora fazer uma introdução ao uso do SELECT.

O comando SELECT é utilizado para retornar registros selecionados de uma ou mais tabelas. Cada expressão SELECT indica as colunas que você deseja recuperar. Sua sintaxe simplificada é dada abaixo:

```
SELECT "coluna" [, "coluna", etc]  
FROM "tabela"  
[WHERE "condicao"];  
  
[] = opcional
```

Onde “coluna” é uma lista com o nome dos campos da(s) tabela(s) que estamos acessando; “tabela” é uma lista com o nome das tabelas que estamos acessando e “condicao” são restrições para acessar os dados. Vamos estudar em breve a cláusula WHERE.

O comando SELECT retorna para o usuário uma lista de valores especificados no comando. Vamos ver um exemplo simples, para que você entenda melhor como funciona o SELECT. Imagine que você tenha os seguintes campos ou colunas em uma tabela chamada ALUNOS, apresentada na Figura 11.

TABELA: ALUNOS		
NOME DO CAMPO	TIPO DO CAMPO	DESCRIÇÃO
matricula	Integer	Identificador único da tabela aluno
nome	Varchar (45)	Nome completo do aluno
e-mail	Varchar (20)	Email do aluno

Figura 11 - Tabela Alunos.

Fonte: os autores (2014)

Vamos supor que a tabela contém os seguintes dados, como mostrado abaixo:

TABELA: ALUNOS		
MATRÍCULA	NOME	EMAIL
1	Jose da Silva	joa@email.com
2	Maria das Dores	maria@email.com
3	Mario Ramos	ramos@email.com
4	Luiz Gustavo	luiz@email.com

Vamos imaginar, então, algumas situações:

- Para listar todos os nomes dos alunos que estão na tabela, o comando: **SELECT nome FROM**



alunos resultaria:

NOME
Jose da Silva
Maria das Dores
Mario Ramos
Luiz Gustavo

- Para listar a matrícula e nome dos alunos que estão na tabela, o comando **SELECT matricula, nome FROM alunos** resultaria:

MATRICULA	NOME
1	Jose da Silva
2	Maria das Dores
3	Mario Ramos
4	Luiz Gustavo

- Para listar todos os campos ou colunas que estão na tabela, o comando **SELECT * FROM alunos** resultaria:

MATRICULA	NOME	EMAIL
1	Jose da Silva	joa@email.com
2	Maria das Dores	maria@email.com
3	Mario Ramos	ramos@email.com
4	Luiz Gustavo	luiz@email.com

Observe que, com o comando SELECT, acessamos todas as linhas da tabela que criamos. Para refinar a nossa consulta, trazendo apenas as linhas que nos interessam, podemos utilizar a cláusula WHERE. Veja alguns exemplos:

- Para listar o nome do aluno cuja matrícula é igual a 1 (um), o comando **SELECT nome FROM alunos WHERE matricula = 1**, resultaria:

NOME
Jose da Silva

- Para listar o nome dos alunos cuja matrícula é maior que 1 (um), o comando **SELECT nome FROM alunos WHERE matricula > 1**, resultaria:

NOME
Maria das Dores
Mario Ramos
Luiz Gustavo



Observe que, neste exemplo, utilizamos o operador relacional $>$ (maior que) para restringir uma pesquisa. Neste sentido, poderíamos utilizar qualquer outro operador, relacional como $<$ (menor que), $<>$ (diferente), dentre outros apresentados na Figura 10.



Veja na nossa primeira videoaula mais alguns exemplos de consulta utilizando a linguagem SQL

Caro (a) aluno (a), nesta competência, apresentamos uma introdução ao grande mundo que é o de banco de dados. Na próxima competência, aprenderemos como projetar um banco de dados e estudaremos mais alguns comandos da linguagem SQL, mas é importante que você faça os exercícios propostos.

ATIVIDADES COMPLEMENTARES

- 1) O que é um SGDB?
- 2) Quais os níveis de abstração de um SGBD?
- 3) O que é um modelo?
- 4) Fale sobre o modelo Entidade-Relacionamento ou MER.
- 5) Explique com suas palavras a diferença entre DML, DQL e DDL.
- 6) Para caracterizar dados, uma linguagem como SQL disponibiliza um conjunto de tipos de dados. Utilizando os tipos descritos neste caderno, descreva quais tipos seriam colocados nos seguintes dados:
 - a) Nome de alunos do curso de EAD
 - b) Data de nascimento dos alunos
 - c) Valor do salário dos professores do curso de EAD
 - d) Valor “P” ou “F” para indicar falta ou presença do aluno
- 7) O que são operadores? Defina operadores lógicos, aritméticos e relacionais.
- 8) Construa comandos SELECT para os seguintes problemas baseados na tabela Alunos:



TABELA: ALUNOS		
MATRICULA	NOME	EMAIL
1	Jose da Silva	joa@email.com
2	Maria das Dores	maria@email.com
3	Mario Ramos	ramos@email.com
4	Luiz Gustavo	luiz@email.com

- a) Consultar os nome e emails dos alunos.
- b) Consultar os nomes de todos os alunos.
- c) Consultar o email do aluno cuja matricula é igual a 3 (três).
- d) Consultar matrícula e e-mail do aluno que tem nome “Jose da Silva”
- e) Consultar todos os alunos cuja matrícula seja menor ou igual a 2 (dois).



2.Competência 02 | Elaborar um Modelo Entidade-Relacionamento

Caro (a) aluno (a), chegamos à segunda competência do curso de Introdução a Banco de Dados. Acredito que você já está familiarizado com o SGBD e programas utilizados na última competência. Neles, construímos tabelas e realizamos diversas consultas com a cláusula SELECT.

Nesta segunda competência, estudaremos os conceitos relacionados à modelagem Entidade-Relacionamento (MER) e como transformar os dados deste modelo para o modelo relacional que, de fato, será implantado no nosso SGBD MySQL. Preste atenção para os conceitos mais importantes desta competência, pois serão úteis sempre que for necessário desenvolver um software. Nesta primeira seção, estudaremos as etapas necessárias para projetar um BD.

2.1 Projeto de Banco de Dados

Modelar, em computação, significa interpretar adequadamente uma realidade do mundo. Para uma modelagem adequada, precisamos de um modelo adequado. Imagine a seguinte situação: você, aluno de informática, se já não recebeu, deve receber em breve alguma proposta, nem que seja de um parente ou amigo, para desenvolver um software para ele. Como você faria isso? Inicialmente, você deveria ter uma boa conversa com ele sobre como ele deseja o software, o que o programa vai fazer e como é o negócio do seu amigo. Veja que estes conceitos são bastante abstratos e apenas com estes conceitos você não conseguirá desenvolver o programa, então o que você precisa? A resposta é construir um modelo.

Um modelo irá possibilitar a implantação do programa, pois ele tem todos os atributos necessários para se construir um software. E onde entra o banco de dados nessa história? O banco de dados sai naturalmente desse modelo, quando você observa o comportamento dos dados envolvidos no mesmo.

Segundo Elmasri & Navathe (2011), um banco de dados representa algum aspecto do mundo real, às vezes chamado de minimundo ou de universo de discurso. Nesse sentido, iniciamos o nosso projeto de BD com a representação de um minimundo, conforme apresentado na Figura 12.



Muitos destes conceitos de modelagem, como minimundo, são estudados por uma área da computação chamada Engenharia de software. Procure aprender sobre esta área da computação.

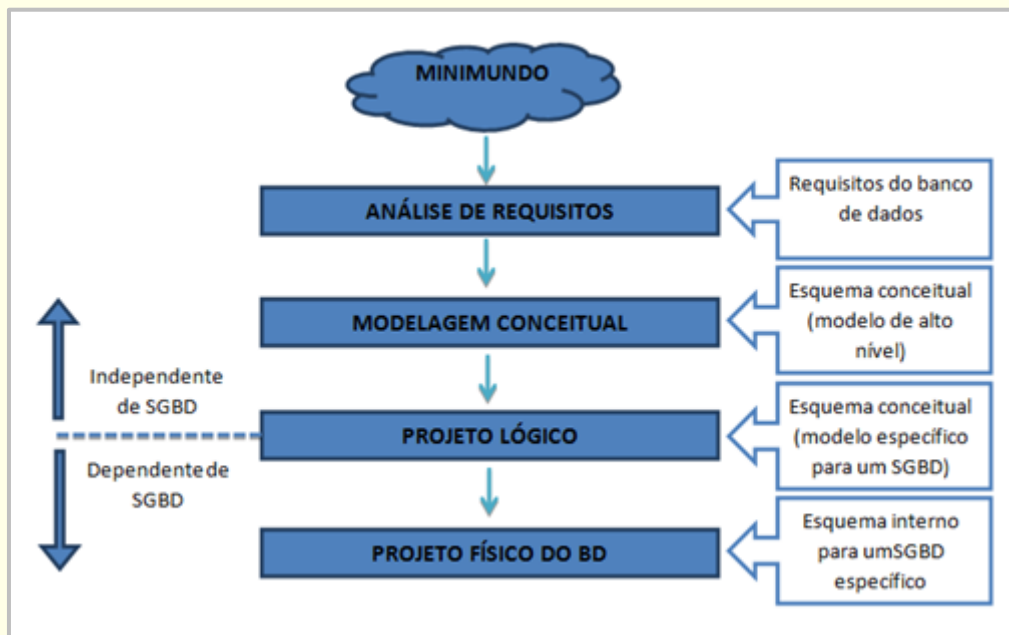


Figura 12 - Etapas do projeto de BD.

Fonte: adaptado de (Figueira 2005)

Descrição: Nesta imagem temos as etapas de um projeto de banco de dados dividido em 4 passos. De cima para baixo é apresentada a análise de requisitos, a modelagem conceitual, o projeto lógico e o projeto físico do BD.

- O minimundo representa uma porção da realidade capturada pelo analista. O mundo real é muito complexo para ser totalmente modelado, sendo assim podemos nos ater apenas a alguns detalhes. Por exemplo, em um supermercado poderíamos imaginar o problema de registrar as vendas para os clientes como um minimundo de toda a complexidade que envolve o dia a dia de um supermercado. Podemos dividir o minimundo em visões. No caso, na hora da venda ao cliente podemos ter a visão da operadora de caixa, que possui operações a serem realizadas diferentes das operações do cliente.
- A modelagem conceitual consiste em mapear toda a realidade a ser modelada utilizando uma linguagem de alto nível. Quando eu falo alto nível, quero dizer que, ao escrever o modelo conceitual, qualquer pessoa que olhar com cuidado, entenderá. Geralmente, utilizamos diagramas para apresentar o modelo conceitual junto com documentos que explicam melhor. O resultado do modelo conceitual é um esquema e não há preocupação com a manipulação ou operação dos dados. Para essa situação, utilizamos os modelos Entidade-Relacionamento (MER), orientado a objetos, hierárquico e outros para representar a nossa realidade. Ainda não há uma preocupação com o SGBD, mas já estamos bem próximo dele.



Modelo conceitual é um modelo abstrato de dados que descreve a estrutura de um BD de forma independente de um SGBD particular (Heuser, 2004)



- O projeto lógico consiste na transformação do modelo conceitual obtido na fase anterior em um modelo lógico. O modelo lógico define como o banco de dados será implantado em um SGBD específico. Como vamos utilizar um SGBD relacional, vamos modelar os nossos dados utilizando o modelo relacional.



Modelo lógico é um modelo de dados que representa a estrutura de dados de um banco de dados, conforme vista pelo usuário do SGBD (Heuser, 2004).

- No projeto físico descrevemos os dados a serem armazenados fisicamente no computador. Como vimos na primeira competência, foi necessário atribuir tipos de dados aos campos criados nas tabelas e definir seus tamanhos (ex: nome: varchar(45)). Esse modelo é um refinamento do modelo lógico, com a perspectiva do nosso SGBD.

Agora que sabemos os passos necessários para a construção de um BD, vamos estudar os principais elementos envolvidos na modelagem Entidade-Relacionamento.

2.2 Modelagem de Dados Utilizando o Modelo Entidade-Relacionamento

A abordagem Entidade-Relacionamento (ER) é a técnica de modelagem de dados mais difundida e utilizada (Heuser, 2004). Nela, o modelo de dados é representado através de um Modelo Entidade-Relacionamento (MER). O MER, por sua vez, pode ser representado graficamente por um Diagrama Entidade-Relacionamento.

A abordagem ER foi criada em 1976 por Peter Chen. Ela pode ser considerada como um padrão, de fato, para a modelagem conceitual. Mesmo as técnicas de modelagem orientadas a objetos, como a UML, baseiam-se nos conceitos da abordagem ER.

Nesta seção, vamos estudar os conceitos envolvidos na abordagem ER: entidade, relacionamento, cardinalidade, atributos e outros. Está pronto?

2.3 Entidade

Uma entidade é uma “coisa” ou um “objeto” do mundo real, mas por que coisa ou objeto? Quando falo coisa, estou me referindo a qualquer entidade do mundo real que possa ser modelada, como por exemplo: um aluno do curso de EAD pode ser uma entidade; um produto de um supermercado pode ser uma entidade, etc. A grande característica desta “coisa” é que ela pode ser identificada unicamente no meio de todas as outras, entende? Então seria a coisa “salário” uma entidade? Bom, depende do caso, mas parece que “salário” não é uma entidade e sim alguma característica de uma entidade, não é? Se professor é uma entidade, então salário poder ser uma característica que o professor possui e não uma entidade.



As entidades podem representar tanto objetos concretos da realidade (pessoa, livro e produto), quanto objetos abstratos (empréstimo do livro, uma viagem de uma pessoa, um departamento). Observe que a característica que define uma entidade é a possibilidade de poder identificar de forma única aquela entidade.



Entidade é o conjunto de objetos da realidade modelada sobre os quais se deseja manter informações no BD.

Um conjunto de entidades (ou do tipo-entidade) é o conjunto que abrange entidades de mesmo tipo e que compartilham as mesmas propriedades ou atributos. Os conjuntos de entidades não são, necessariamente, conjuntos separados ou sempre disjuntos. Por exemplo, o conjunto de todos os clientes de um banco constitui o conjunto entidade-cliente; o conjunto de todos os empregados do banco constitui o conjunto entidade-empregado; a entidade pessoa pode pertencer ao conjunto Cliente ou ao conjunto Empregado ou a ambos ou a nenhum deles.

Em um DER, uma entidade é representada através de um retângulo que contém o nome da entidade, como apresentado na Figura 13.

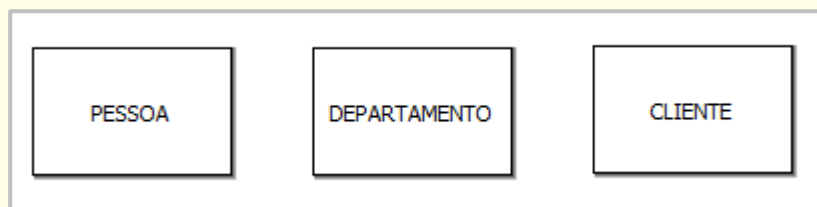


Figura 13 - Representação gráfica de Entidade no MER.
Nesta figura temos três entidades: pessoa, departamento e cliente.
Fonte: os autores (2014)

Neste momento, é muito importante que você saiba distinguir bem uma entidade do minimundo que você está modelando, pois as entidades cumprem um papel fundamental no modelo ER. Você pode agora estar se perguntando: como é que eu posso definir corretamente uma entidade? Um bom começo é buscar os atributos desta entidade. Vamos ver o que é um atributo?

2.4 Atributo

Uma entidade é representada por um conjunto de atributos. Os atributos são propriedades que descrevem cada membro de um conjunto de entidades. Cada entidade tem seus próprios valores nos atributos. Para cada atributo, existe um conjunto de valores possíveis, chamado de domínio. Por exemplo, temos uma entidade chamada Pessoa, esta entidade tem o atributo nome, que descreve o primeiro nome desta pessoa. O domínio é o conjunto de todos os possíveis nomes que uma entidade Pessoa pode ter, por exemplo: {João, Maria, Pedro ou Carlos}. Esse domínio não pode ser, por exemplo, igual a {1, 2,3, 2} pois este conjunto não representa nomes e sim valores



numéricos. Cada entidade pode ser descrita pelo conjunto formado pelos pares (atributo-valor) referentes a cada atributo do conjunto em questão.



Atributo é um dado que é associado a cada ocorrência de uma entidade ou de um relacionamento.

Os atributos podem pertencer a uma das seguintes classes:

- **Simple ou compostos:** atributos compostos, diferentemente dos simples (atômicos), são divididos em partes (em outros atributos). O nome do cliente, por exemplo, pode ser representado por dois atributos: nome e sobrenome.
- **Monovalorados ou multivalorados:** atributos monovalorados assumem apenas um único valor para uma entidade específica, já os multivalorados podem assumir conjunto de valores para uma única entidade. Por exemplo: a entidade produto pode ter um atributo (código), este monovalorado; a entidade empregado pode ter um atributo (nome-dependentes) multivalorado, pois o empregado pode ter vários dependentes; a entidade cliente pode ter um atributo (telefone) também multivalorado.
- **Nulos:** um valor nulo é usado quando uma entidade não possui valor para determinado atributo. Por exemplo, se o empregado não possui número da carteira de reservista, o valor nulo é atribuído a este atributo para esta entidade, significando que o atributo não é aplicável.
- **Armazenados X Derivados:** o valor de um atributo pode ser derivado de outro. Por exemplo, o atributo (idade), do tipo derivado, é calculado a partir do atributo data de nascimento, armazenado no banco de dados; o valor do tempo de serviço de um funcionário em uma empresa pode ser calculado a partir da sua data de contratação.

Veja na Figura 14 a representação gráfica dos atributos em um DER. O atributo telefone aparece como um atributo multivalorado, visto que uma pessoa pode ter mais de um número de telefone.

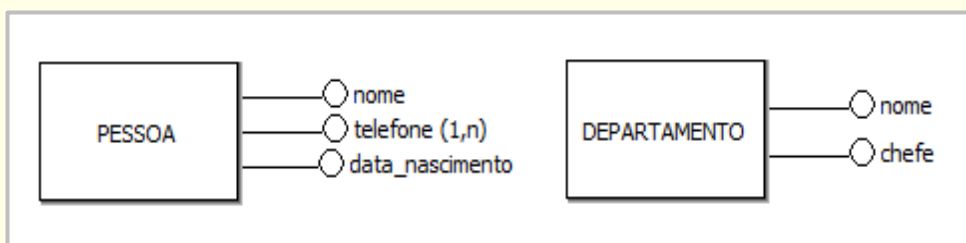


Figura 14 - Atributos de uma entidade.

Fonte: os autores (2014)

Descrição: Nesta figura temos duas entidades (pessoa e departamento) com seus respectivos atributos. Pessoa possui nome, telefone e data de nascimento. Departamento possui nome e o chefe.

Vejamos um exemplo do mundo real: imagine um sistema que controla dados dos clientes em um



banco. De forma muito simplificada, teríamos as seguintes entidades e seus respectivos atributos:

a) Entidades: {Cliente, Agência, Conta, Empréstimo}

b) Atributos:

- Cliente = (**nome_cliente**: string; **cpf**: string).
- Conta = (**número_conta**: string; **saldo**: real).
- Empréstimo = (**número_empréstimo**: inteiro; **total**: real).
- Agência = (**nome_agência**: string; **cidade_agência**: string).

2.5 Atributos Chave

Como distinguir um elemento de uma entidade de outro elemento da mesma entidade? Na base de dados do nosso curso EAD, deve existir uma entidade chamada Aluno que tem um atributo chamado Nome. Para esse atributo, existem milhares de possíveis nomes de pessoas, alguns até iguais. Imagine quantas “Maria da Silva” existem cadastradas no banco de dados? E quantos “José da Silva”?



Um atributo chave ou identificador de entidade é um conjunto de um ou mais atributos cujos valores servem para distinguir uma ocorrência de uma entidade das demais ocorrências da mesma entidade.

Uma chave é um atributo que possui valor único no conjunto de entidades e relacionamentos. O atributo chave, ou chave primária, permite identificar de maneira única uma ocorrência no BD. Para a base de dados do nosso curso EAD, na entidade Aluno, o atributo chave pode ser o número do seu CPF ou mesmo um número de matrícula, pois, como regra, duas pessoas diferentes não podem ter o mesmo CPF ou duas pessoas não podem ter um mesmo número de matrícula. A Figura 15 apresenta o atributo chave CPF da entidade Pessoa.

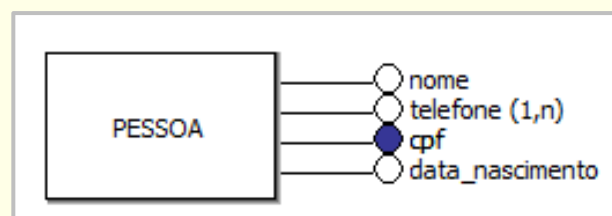


Figura 15 - Atributo chave de uma entidade.

Fonte: os autores (2014)

Descrição: Nesta imagem temos a entidade pessoa que possui como atributos nome, telefone (no caso um ou vários telefones), cpf e data de nascimento.

Nem todas as entidades têm um atributo chave. Nesse caso, chamamos essas entidades de fracas. Por exemplo, imagine uma entidade dependente ligada às pessoas cadastradas no imposto de



renda da Receita Federal. Esta entidade teria os seguintes atributos: **nome_ dependente** e **data_nascimento**. Será que não temos dependentes de Contribuintes diferentes que tenham o mesmo nome e que tenham nascido na mesma data? Observe que, para que cada dependente seja único, é necessário chamar um atributo que seja chave, neste caso poderia ser o CPF do titular do dependente, que é uma chave da entidade Contribuinte. A tabela apresentada na Figura 16 mostra a entidade hipotética Contribuinte, com alguns valores.



O termo “**fraca**” deriva do fato de a entidade somente existir quando relacionada a outra entidade e de usar como parte de seu identificador atributos de entidades relacionadas.

CONTRIBUINTE		
CPF	NOME	DATA_NASC IMENTO
121234321	Jose da Silva	13/01/1974
234122345	Maria da Penha Silva	23/01/1980
122123345	Roberta Miranda	02/01/1970
213454876	Jose da Silva	13/01/1974

Figura 16 - Entidade Contribuinte

Fonte: os autores (2014)

DEPENDENTE			
CPF_CONTRIBUINTE	CODIGO	NOME_DEPENDENTE	DATA_NASC IMENTO
121234321	01	Maria Lúcia da Silva	02/05/1974
121234321	02	Mariana da Silva	22/07/2002
122123345	01	Pedro Miranda	05/06/1968

Figura 17 – Entidade Dependente

Fonte: os autores (2014)

Observe que existem dois campos com o mesmo nome e a mesma data de nascimento (linhas 1 e 4) na entidade Contribuinte. Porém, os números de CPF são diferentes, neste caso devemos supor de que se trata de pessoas diferentes. Observe também que, na entidade Dependente, uma entidade é identificada unicamente através do CPF do contribuinte e do código do dependente.

2.6 Relacionamentos

Um relacionamento é uma operação que relaciona duas ou mais entidades distintas com um significado específico. No exemplo anterior, do banco, o cliente está relacionado com uma conta, ou seja, um cliente possui uma conta. O cliente também está relacionado com uma agência e pode ou não ter um empréstimo no banco. A Figura 18 apresenta o DER que expressa estas relações.

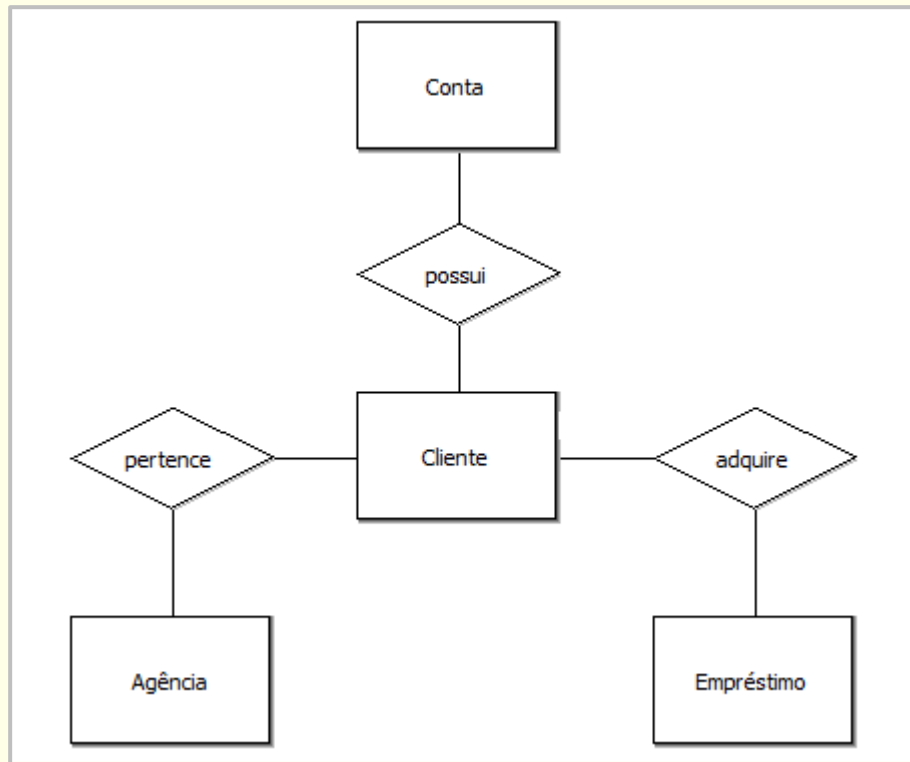


Figura 18 – DER para sistema bancário.

Fonte: os autores (2014)

Descrição: Neste DER temos uma agência que pertence a um cliente. Um cliente por sua vez pode adquirir um empréstimo. O cliente também pode possuir uma conta.



Sintaxe é a forma como as instruções de uma linguagem são escritas. A **semântica** é complementar à sintaxe. Ela corresponde à descrição do significado das instruções válidas de uma linguagem.

Observe no diagrama acima que uma entidade é descrita como um retângulo e o relacionamento entre estas entidades como um losango. Vários outros símbolos são utilizados em um DER. Infelizmente, na prática e na literatura, aparecem muitas versões, que distinguem uma das outras não somente na representação gráfica, mas também na sintaxe e também na semântica. A Figura 19 apresenta os símbolos gráficos que fazem parte do DER.



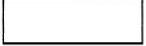
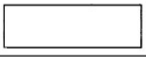




Símbolo	Significado
 ENTIDADE	Retângulos: conjuntos de entidades; <ul style="list-style-type: none"> Linhas duplas: conjuntos de entidades fracas
 ENTIDADE FRACA	
 RELACIONAMENTO	Losangos: conjuntos de relacionamentos; <ul style="list-style-type: none"> Linhas duplas: conjuntos de relacionamentos envolvidos com entidades fracas.
 ATRIBUTO	Elipses: atributos; <ul style="list-style-type: none"> Atributos da chave primárias são sublinhados; Linhas duplas: atributo multivalorado; Linhas pontilhadas: atributo derivado; Estrutura em árvore: atributos composto;
 ATRIBUTO-CHAVE	
 ATRIBUTO MULTIVALORADO	
	Linhas: unem atributos aos conjuntos de entidades e estes aos conjuntos de relacionamentos; <ul style="list-style-type: none">

Figura 19 – Representação gráfica dos conceitos que compõem o modelo ER.

Fonte: os autores (2014)

Descrição: Nesta imagem temos os símbolos em modelagem ER e seus respectivos significados. Os retângulos por exemplo, são conjuntos de entidades. Quando se tem retângulos com duas linhas ao redor estes são denominados de entidades fracas. As linhas unem atributos aos conjuntos de entidades.

Quando do seu surgimento na literatura, a abordagem ER não era ainda apoiada por ferramentas CASE. Com isso, cada autor de livros adotava a representação gráfica que desejasse. A consequência é que, hoje, observa-se uma grande variedade de abordagens que levam o título de ER.



Ferramentas CASE (do inglês Computer-Aided Software Engineering) é uma classificação que abrange todas as ferramentas baseadas em computadores que auxiliam atividades de engenharia de software, desde análise de requisitos e modelagem até programação e testes.



Pesquise na internet e em livros outras notações para o modelo ER. Uma notação bem conhecida é da Engenharia de Informações.



Pesquise por ferramentas CASE que auxiliam na construção de modelos ER.



2.7 Cardinalidade de Relacionamento

Uma propriedade importante de um relacionamento é a quantidade de ocorrências de uma entidade que pode estar associada a uma ocorrência de outra entidade em um relacionamento. Esta propriedade é chamada de cardinalidade de uma entidade em um relacionamento. Há duas cardinalidades a se considerar: a máxima e a mínima.

Cardinalidade (mínima, máxima) é o número mínimo e o número máximo de ocorrências de uma entidade relacionada à outra. Todos os relacionamentos possuem uma cardinalidade associada. A cardinalidade é definida através do mapeamento de restrições.

No exemplo do DER representado na Figura 18, a cardinalidade do relacionamento “pertence” que associa as entidades Cliente e Agência é **1:1** (um para um), que significa que cada cliente só pode pertencer a uma agência. Se um cliente pudesse pertencer a duas agências, como ficaria a cardinalidade? Ficaria assim **1:2** (um para dois), ou seja, um cliente pode pertencer a duas agências. Extrapolando, poderíamos ter a cardinalidade **1:N** que indica que o cliente pode pertencer a “N” (muitas) agências. Os relacionamentos podem ser:

- **Relacionamento um para um (1:1):** uma entidade em A está associada, no **máximo**, a uma entidade em B, e uma entidade em B está associada a, no máximo, uma entidade em A. Veja o exemplo da Figura 20.

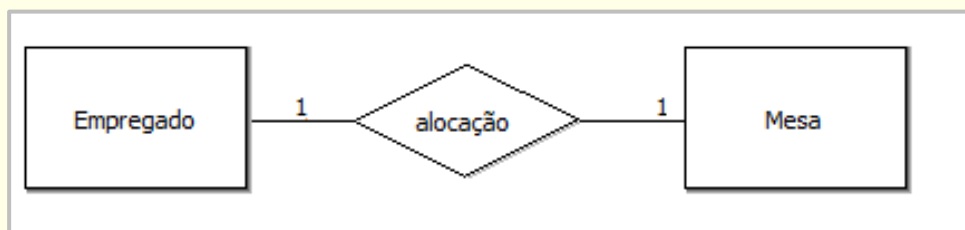


Figura 20 – Cardinalidade máxima 1:1.

Fonte: os autores (2014)

Nesta imagem temos duas entidades (empregados e mesa) ligados por um relacionamento alocação. A cardinalidade neste caso é de um para um pois um empregado apenas pode haver uma mesa e uma mesa pode ser alocada para apenas um empregado.

- **Relacionamento um para muitos (1:N):** uma entidade em A está associada a várias entidades em B. Uma entidade em B deve estar associada no máximo a uma entidade em A. Veja o exemplo da Figura 21. Um Empregado possui N dependentes, mas um Dependente só pode estar relacionado a um Empregado.

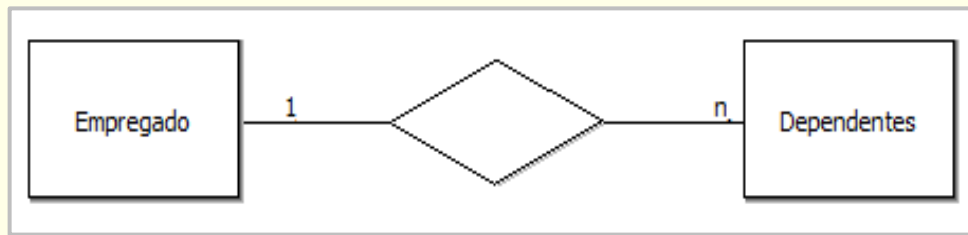


Figura 21 - Cardinalidade máxima 1:N.

Fonte: os autores (2014)

Descrição: Nesta imagem temos uma entidade empregado relacionando-se a dependentes. A cardinalidade desta relação é de um para vários pois um empregado pode possuir vários dependentes e um dependente é pertencido por apenas um empregado.

- **Relacionamento muitos para um (N:1):** uma entidade em A está associada a, no máximo, uma entidade em B. Uma entidade em B pode estar associada a um número qualquer de entidades em A. Este relacionamento é o inverso do apresentado na Figura 21.
- **Relacionamento muitos para muitos (N:N):** Uma entidade em A está associada a qualquer número de entidades em B e uma entidade em B está associada a um número qualquer de entidades em A.

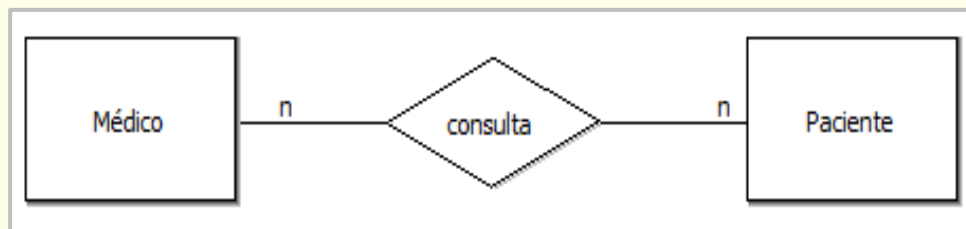


Figura 22 - Cardinalidade máxima N:N.

Fonte: os autores (2014)

Descrição: Neste modelo temos um médico que possui vários pacientes enquanto que um paciente ao invés de apenas um, ele pode ser atendido por vários médicos. A cardinalidade aqui é de muitos para muitos.

Podemos utilizar a notação do tipo **(mim,max)** onde **mim** representa o menor valor e **max** o maior valor para a cardinalidade do relacionamento entre entidades. Veja o exemplo da Figura 23, que mostra alguns tipos de cardinalidades possíveis entre duas entidades.

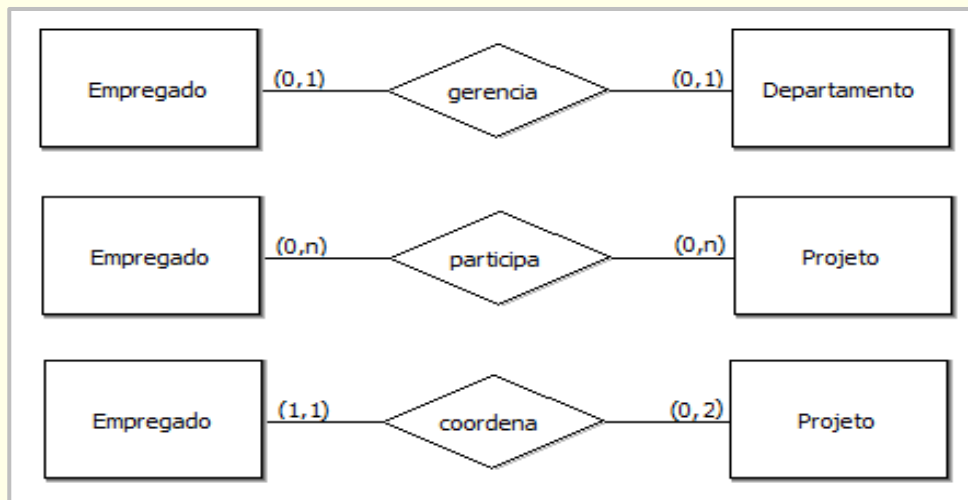


Figura 23 – Cardinalidade mínima e máxima no modelo ER.

Fonte: os autores (2014)

Descrição: Podemos definir limites para a cardinalidade. Um limite mínimo e um máximo.

Nesta imagem podemos observar quatro exemplos de limitação de cardinalidade:

0 ou 1, 0 ou vários, 1 e apenas 1, 0 ou dois.

Em (A), um Empregado pode gerenciar um ou nenhum Departamento. A entidade Departamento pode ser gerenciada por um ou nenhum empregado.

Em (B), um Empregado pode participar de zero ou muitos Projetos dentro de uma empresa. Um Projeto pode ter zero ou muitos Empregados.

Em (C), um Empregado pode coordenar nenhum ou até dois Projetos. Um Projeto, por sua vez, pode ter apenas um empregado em sua coordenação.

Agora que vimos os principais conceitos envolvidos na modelagem ER, vamos juntar tudo e construir um modelo único, através da descrição de um minimundo.

2.8 Criando um Modelo ER

Para entender os passos envolvidos na modelagem ER, vamos exemplificar um caso. Vou descrever um minimundo e depois vamos juntos fazer a modelagem. Vamos lá?

Imagine que você recebeu a tarefa de desenvolver um sistema para um colégio. O dono do colégio gostaria de ter o cadastro dos alunos e dos professores. Ele também quer que os professores sejam alocados nas disciplinas e que os alunos também possam estar relacionados com as disciplinas, nas quais estejam matriculados, é claro.

O primeiro passo envolve identificar as entidades envolvidas no problema. Lembre-se de que as entidades representam conjuntos de objetos que desejamos armazenar em nosso BD. Na sequência, identificamos os relacionamentos existentes entre elas. Por fim, verificamos quais os atributos das entidades, bem como os possíveis atributos para os relacionamentos.



- **Entidades:** com base na descrição do minimundo, podemos definir facilmente duas entidades por serem concretas, são elas: Aluno e Professor. Uma terceira entidade, não concreta, é a entidade Disciplina.
- **Relacionamentos:** agora devemos perguntar às entidades com quem elas poderiam se relacionar. Poderíamos dizer que os alunos devem se matricular nas disciplinas, logo surge um relacionamento “assiste” que relaciona os alunos com as disciplinas. Outro relacionamento é o dos professores com as disciplinas, surge o relacionamento “ministra”, indicando que cada professor deve ministrar uma ou mais disciplinas. Você poderia dizer se há relação entre as entidades Professor e Aluno? Esta relação pode ser deduzida através da disciplina, logo não é necessário criar um relacionamento direto entre Aluno e Professor.
- **Cardinalidade dos Relacionamentos:** agora devemos perguntar aos relacionamentos como eles devem unir cada membro das entidades envolvidas. Para o relacionamento “assiste” entre as entidades Aluno e Disciplina, podemos definir que cada aluno pode assistir várias disciplinas (n disciplinas) e que cada disciplina pode ser assistida por vários alunos (m alunos) logo a cardinalidade máxima desta relação é muitos para muitos (n x m). Já a relação “ministra” que envolve professores e disciplina é diferente, pois uma disciplina só poderá ser ministrada por um professor e um professor poderá ministrar várias disciplinas (n disciplinas), logo temos a cardinalidade máxima 1 para muitos (1 x n).
- **Atributos:** vamos definir todos os atributos envolvidos nas entidades e nos relacionamentos:

Aluno (matrícula, nome, classe, ano)

Professor (RG, nome, escolaridade)

Disciplina (código, nome)

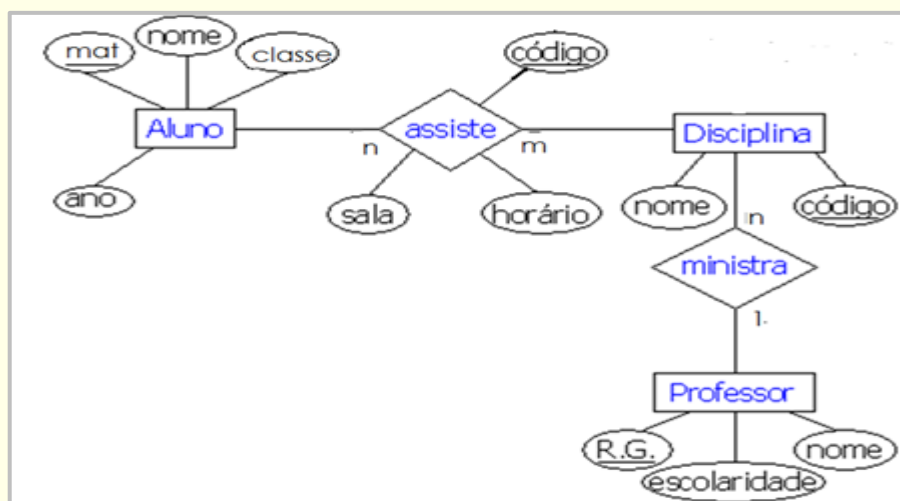


Figura 24 – DER para um colégio.

Fonte: os autores (2014)

Descrição: Neste DER temos um aluno que pode assistir várias disciplinas e uma disciplina pode ser assistida por vários alunos. Um professor que pode ministrar várias disciplinas, mas uma disciplina pode ser ministrada por apenas um professor.



Observe que o relacionamento “assiste” possui três atributos descritivos. Este relacionamento que tem cardinalidade $m \times n$ vai nos gerar uma nova tabela para comportar os atributos descritivos. Veja na Figura 25 as tabelas do modelo relacional que serão geradas a partir deste modelo ER. * indica os atributos chaves. ** indica que este atributo é uma chave estrangeira.



Chave estrangeira é uma coluna ou combinação de colunas cujos **valores aparecem** necessariamente na chave primária de uma tabela. A chave estrangeira é o mecanismo que permite a implantação de relacionamentos em um banco de dados relacional.

ENTIDADE: ALUNO (TABELA ALUNO)	
NOME DO ATRIBUTO	TIPO DO ATRIBUTO
Matricula *	Int
Nome	varchar(45)
Classe	varchar(20)
Ano	Date

ENTIDADE: DISCIPLINA (TABELA DISCIPLINA)	
NOME DO ATRIBUTO	TIPO DO ATRIBUTO
Codigo *	Int
Nome	varchar(45)
Professor_rg **	varchar(12)

ENTIDADE: PROFESSOR (TABELA PROFESSOR)	
NOME DO ATRIBUTO	TIPO DO ATRIBUTO
RG *	varchar(12)
Nome	varchar(45)
Escolaridade	varchar(20)

RELACIONAMENTO: Assiste (Tabela Aluno_Disciplina)	
Nome do atributo	Tipo do atributo
Codigo *	Int
Mat **	Int
Disciplina **	Int
Sala	varchar(12)
Horario	Int

Figura 25– Modelo Relacional para um colégio

Fonte: os autores (2014)



Engenharia reversa é o processo que permite obter um modelo lógico relacional a partir de um modelo físico de um SGBD.

Você deve ter percebido a inclusão das chaves Mat e Disciplina na tabela aluno_professor, que são chaves estrangeiras que vêm da cardinalidade do relacionamento.

Para construir tanto o DER quanto as tabelas relacionais, fazemos uso de ferramentas CASE. Na próxima seção, vamos apresentar uma ferramenta CASE bastante popular para o SGBD MySQL.

2.9 MySQL Workbench

O MySQL Workbench é uma ferramenta CASE para modelagem de dados. Com ela, é possível criar o banco de dados de forma visual. Os comandos SQL são gerados automaticamente pela ferramenta e é possível executá-los no servidor. Toda a criação dos relacionamentos entre as tabelas podem ser baseadas em chaves estrangeiras. Outro recurso que a ferramenta possibilita é realizar a engenharia reversa de esquemas do banco de dados, bem como gerar todos os scripts em SQL.



Veja na nossa segunda videoaula a elaboração de um DER, utilizando uma ferramenta CASE

A diferença básica entre o MySQL Workbench e o phpMyAdmin é que o Workbench possui uma ferramenta visual. No mais, as mesmas operações podem ser realizadas.

Caro (a) aluno (a), nesta competência estudamos um dos modelos conceituais mais conhecidos e utilizados para projetar um Banco de Dados. A partir de um modelo ER, podemos construir um BD para um SGBD em específico. Na terceira e última competência, vamos criar um BD utilizando os conceitos do modelo lógico relacional, juntamente com os comandos da linguagem SQL. Mas, antes disso, vamos rever alguns conceitos através de atividades de aprendizagem.



Nesta competência, estudamos os principais conceitos envolvidos na modelagem ER, contudo existem outros conceitos também úteis para o projeto de um BD. Aprofunde o seu conhecimento pesquisando sobre o assunto na internet e em livros. (Heuser, 2004) contém diversos exemplos de minimundo com seus respectivos modelos ER.



ATIVIDADES COMPLEMENTARES

1) Defina:

- a) Entidade;
- b) Atributo;
- c) Atributo chave;
- d) Chave estrangeira;
- e) Chave primária;
- f) Relacionamento;
- g) Cardinalidade.

2) Utilizando a ferramenta Workbench você deverá criar todas as tabelas e relacionamentos da modelagem apresentada na Figura 25.

3) Faça a mesma operação do exercício anterior, utilizando a ferramenta phpMyAdmin, e comente as principais diferenças que você pode perceber.

4) Usando phpMyAdmin:

- a) Insira quatro alunos na terceira classe e cinco na primeira classe na tabela alunos;
- b) Insira três professores (João da Silva, Maria Penha e Pedro Henrique) na tabela professor;
- c) Insira as disciplinas Matemática, Português e Inglês na tabela disciplina, para cada disciplina escolha um professor;
- d) Matricule três alunos em português, três em inglês e cinco em matemática (cadastrar na tabela aluno_disciplina).

5) Faça as seguintes pesquisas usando o comando SELECT que você aprendeu na primeira competência:

- a) Todos os alunos da terceira classe;
- b) Todos os alunos que fazem matemática;
- c) Todos os alunos que fazem matemática e são da terceira classe;
- d) Todas as disciplinas do professor João da Silva.



3. Competência 03 | Construir Tabelas e Dicionários de Dados de um Banco De Dados

Caro (a) aluno (a), chegamos a terceira e última competência deste curso de banco de dados. Na primeira e na segunda competência você aprendeu sobre os conceitos de banco de dados: você já sabe o que é um SGDB, sabe modelar um banco utilizando a modelagem ER, já sabe criar tabelas com o phpMyAdmin e o Workbench e realizar consultas com o comando SELECT.

Mas falta ainda “botar a mão na massa” e produzir uma modelagem completa, com vários níveis e complexidade de pesquisas, além de conhecer novos comandos SQL. É o que vamos fazer nesta competência. Ao final, você terá um banco de dados de controle de estoque de um mercado completo. Será uma boa experiência para você que brevemente estará programando em empresas ou quem sabe trabalhando por conta própria. Vamos lá?

3.1 Minimundo Supermercado

Antes de começar esta competência, apresento o cenário que desejamos modelar. Para isso, passo a descrever uma situação que poderia ser normal numa entrevista a um cliente que deseja informatizar um supermercado.

Um dono de supermercado lhe convidou para que você fizesse um sistema de controle de estoque dos produtos vendidos no supermercado. Você vai ao supermercado e começa a perguntar ao dono o que ele gostaria de ter no sistema e ele lhe diz:

Gostaria de controlar a quantidade de produtos que entra e sai do meu estoque diariamente e que o sistema permitisse um valor mínimo e máximo de produtos no estoque para que eu não comprasse produtos de mais ou faltassem produtos. Gostaria de ter controle sobre os fornecedores dos produtos (nome ou razão social, CNPJ, telefone) e quais mercadorias os fornecedores vendem. Outra informação importante é sobre os meus clientes: eu gostaria de saber o CPF, nome, telefone e e-mail deles, pois assim poderia melhorar minhas vendas.

Bom, essa entrevista foi muito boa, mas agora você precisará interpretar os dados fornecidos informalmente pelo dono do supermercado. A interpretação irá levá-lo à construção do seu MER (modelo Entidade-Relacionamento) que, então, será a base para o banco de dados. Junto com o MER, você aprenderá a construir um dicionário de dados, que é um documento essencial para o entendimento do sistema como um todo. Vamos começar identificando as entidades envolvidas na nossa modelagem.

3.2 Identificando as Entidades

Uma entidade é uma “coisa” ou um “objeto” concreto ou abstrato do mundo real. Para identificar as entidades envolvidas no minimundo da entrevista que fizemos, vamos tentar localizar estas



“coisas” ou “objetos”. Veja as seguintes frases do cliente destacadas do texto:

Gostaria de controlar a quantidade de produtos que entra e sai do meu estoque diariamente e que o sistema permitisse um valor mínimo e máximo de produtos no estoque para que eu não comprasse produtos de mais ou faltassem produtos. Gostaria de ter controle sobre os fornecedores dos produtos (nome ou razão social, CNPJ, telefone) e quais mercadorias os fornecedores vendem. Outra informação importante é sobre os meus clientes: eu gostaria de saber o CPF, nome, telefone e e-mail deles, pois assim poderia melhorar minhas vendas.

Observe que, nas frases em destaque, podemos encontrar pistas sobre as nossas entidades. Na primeira frase, produtos parece ser uma entidade, mas por quê? Produto é um objeto concreto, real e presente no minimundo. Ele é de interesse, pois é sobre ele que devemos “controlar a quantidade”. Veja que a entidade, além de ser um objeto real, sofre uma ação do sistema. Mais à frente, veremos que ainda podemos definir seus atributos.

Na mesma frase podemos definir outra entidade: já conseguiu identificar qual? A entidade Estoque. Por quê? Observe que também é necessário fazer uma ação sobre esta entidade, que é a de manter o controle da quantidade de produtos, além disto, pense concretamente, pode existir um produto cadastrado no sistema (ex. “sabão em barra vem-ri-ri”) que não tenha estoque dele cadastrado, sendo assim a entidade produto terá o cadastro do produto, mas a entidade estoque não terá uma referência ao produto.

Bem, agora ficou fácil definir as outras entidades envolvidas, são elas: Fornecedores e Clientes. Apesar de serem um pouco diferentes da entidade produto, são fáceis de serem percebidas pelo mesmo motivo de produtos. Será que existem outras entidades? O que você acha?

Agora que já temos nossas entidades, devemos pensar nos atributos e relacionamentos. Observe que a modelagem só ficará completa quando todas as peças se encaixarem, é um verdadeiro quebra-cabeça.

3.3 Identificando os Atributos

Aqui devemos seguir o que o cliente deseja para cada entidade. Poderemos também colocar outros atributos que o cliente não informou, como por exemplo, atributos que servirão como controle. Vamos olhar o que é falado no texto do diálogo sobre cada uma das entidades que definimos antes.

Gostaria de **controlar a quantidade de produtos que entra e sai do meu estoque** diariamente e que o sistema permitisse um **valor mínimo e máximo de produtos** no estoque para que eu não comprasse produtos de mais ou faltassem produtos. Gostaria de ter controle sobre **os fornecedores dos produtos (nome ou razão social, CNPJ, telefone) e quais mercadorias os fornecedores vendem**. Outra informação importante é sobre os meus **clientes**: eu gostaria de saber **o CPF, nome, telefone e e-mail** deles, pois assim poderia melhorar minhas vendas.



- **Entidade produtos:** podemos então definir um atributo estoque mínimo e estoque máximo para controlar esta característica.
- **Entidade estoque:** precisamos pensar em qual atributo será necessário para fazer este controle. Veremos isso quando falarmos dos relacionamentos, ok?
- **Entidade fornecedores:** fica fácil, são eles: razão social (nome da empresa), CNPJ e telefone. Nesse caso, observe que é feito uma referência aos produtos que o fornecedor vende. Típico caso de chave estrangeira (lembre-se da definição).
- **Entidade clientes:** também fácil, os atributos são: CPF, nome, telefone e e-mail.

Observe que ainda não começamos a construir nosso diagrama ER e nem sabemos ainda como fazer um dicionário de dados, mas as informações estão chegando. Vamos agora aos relacionamentos. Lembre-se de que a modelagem chama-se Entidade-Relacionamento. Já vimos as entidades, agora iremos aos relacionamentos.

3.4 Mapeando Relacionamentos

Um bom começo para definir os relacionamentos do nosso banco de dados é olhar para as entidades e tentar relacioná-las uma a uma. Por exemplo, será que a entidade produto se relaciona com a entidade estoque? A resposta é SIM! Se existe uma relação, qual seria a cardinalidade desta relação? Cada produto pode ter um registro no estoque que irá informar a quantidade de produtos.

Pode haver mais de um registro no estoque para o mesmo produto? NÃO, pois se houver mais de um registro como saberemos qual a quantidade real? Então a cardinalidade é 1:1 (um para um).

Será que a entidade produto possui relação com a entidade cliente? SIM, a entidade cliente realiza compra de produtos, precisamos então modelar este relacionamento. A cardinalidade é N:N (N para N), ou seja muitos para muitos, pois um cliente pode comprar vários produtos (itens) e um produto (ou um item) pode ser comprado por mais de um cliente. ATENÇÃO: não confundir o termo produto que se refere à entidade com o termo produto que se refere ao item. Então “Sabão em pó vem-ri-ri” é um produto ou item da entidade produto, certo?

A entidade produto possui uma relação com a entidade fornecedor, qual a cardinalidade desta relação? N:N (N para N), responda o porquê desta cardinalidade.

Agora, podemos montar o nosso DER. A Figura 26 mostra todos os elementos que modelamos, com exceção dos atributos que serão colocados depois, quando criarmos o dicionário de dados.

Este diagrama ainda não está completo, pois faltam os atributos que serão definidos posteriormente. Após esta etapa, será necessário fazer a representação tabular que resultará nas tabelas relacionais propriamente ditas, pois os relacionamentos geram chaves que dependem da cardinalidade da relação. Vamos identificar como criar chaves a partir de um determinado tipo de relacionamento.

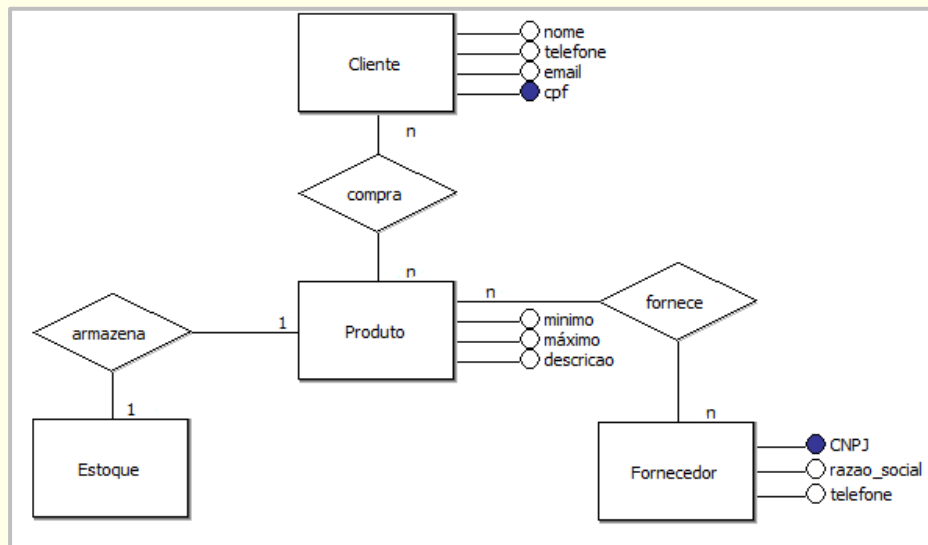


Figura 26 – DER para um supermercado.

Fonte: os autores (2014)

Descrição: Neste DER temos um estoque que pode armazenar um tipo de produto e este produto pertence a um estoque específico. Um cliente que pode comprar vários produtos e um produto pode ser comprado por vários clientes. Por fim, um fornecedor que fornece vários produtos e um produto pode ser fornecido por vários fornecedores.

3.5 Atributos Chaves em um Relacionamento

Quando definimos um relacionamento entre duas entidades devemos também definir qual a chave primária da relação. Por exemplo, quando definimos anteriormente a entidade cliente e agência, temos que ter uma chave que descreva unicamente cada cliente e sua agência pelo relacionamento chamado “pertence”. Para que seja definida uma chave para um relacionamento, devemos olhar para a cardinalidade da relação. Dependendo do tipo do relacionamento, teremos uma forma de definir a chave do relacionamento:

- Relacionamento muitos para muitos (N:N): fazemos a união das chaves da entidade + atributos descritivos. Os atributos descritivos são opcionais, mas às vezes é importante acrescentar os novos atributos para melhor descrever o relacionamento. Deve-se criar uma tabela para comportar os atributos.
- Relacionamento muitos para um ou um para muitos (N:1 ou 1:N): utilizamos a chave da entidade do lado do muitos + atributos descritivos.
- Um para um (1:1): qualquer uma das chaves primárias pode ser usada.

Quando trazemos uma chave de uma entidade para outra chamamos esta chave de “chave estrangeira”. As chaves estrangeiras impõem restrições que devem ser garantidas ao executar diversas operações no BD. A próxima seção apresenta algumas destas restrições.

3.6 Integridade de Dados

Um banco de dados bem projetado tem que garantir a integridade dos dados armazenados. É que,



durante a vida útil do banco de dados, muitas operações serão realizadas, tais como: remoção de um registro, modificação de valores de registros, inserção de novos registros, etc.

Para que o banco de dados possua integridade é necessário que o projetista garanta isso durante a etapa de criação do diagrama ER e do dicionário de dados que veremos em breve. São três os níveis de integridade que devem ser observados:

- **Integridade de domínio:** garante que tipos de dados devem estar corretamente associados aos atributos. Por exemplo, se você definir um campo sexo que pode assumir “F” ou “M” você deve garantir o domínio adequado que é relacionado ao tipo CHAR (Caractere) e não ao tipo INT (Inteiro).
- **Integridade de Entidade:** são as chaves primárias que definimos anteriormente. Temos que garantir que dois registros não tenham a mesma chave primária. Para isso, escolhemos um identificador único para cada registro ou linha da tabela.
- **Integridade referencial:** são as chaves estrangeiras. Devemos garantir que o valor associado a um campo que seja uma chave estrangeira esteja cadastrado na tabela de referência.

Você verá estes tipos de integridade acima mencionados sendo colocadas no nosso projeto. Na próxima seção, vamos mostrar como fazer a representação tabular do diagrama ER, dando origem ao modelo lógico relacional utilizado pelos SGBD relacionais, tais como o MySQL, MS SQLServer e outros.

3.7 Representação Tabular

A representação tabular é a forma de transformar o diagrama ER em tabelas que serão construídas no SGBD Relacional. Para criar a tabela, devemos seguir duas regras básicas:

- Cada entidade vira uma tabela. No caso do minimundo do Supermercado, apresentado na Figura 26, teríamos as seguintes tabelas: **clientes**, **produtos**, **fornecedor** e **estoque**.
- Se uma entidade tiver um relacionamento associado, então devemos organizar as chaves estrangeiras nas tabelas da relação.

Para organizar as chaves nos relacionamentos, devemos olhar para a cardinalidade das relações. Conforme explicado na seção 0, dependendo da cardinalidade, será ou não necessária a criação de uma nova tabela. Vamos analisar cada uma das relações do nosso diagrama ER da Figura 26.

- **Relacionamento compra:** as entidades envolvidas são clientes e produtos, com cardinalidade N:N (muitos para muitos). Nesse caso, devemos fazer a união dos atributos chaves das duas entidades. Onde vamos colocar a união das chaves? Vamos criar duas tabelas: uma **registro_venda** que se relacionará com os clientes (vamos colocar uma chave estrangeira do cliente nesta tabela) e outra tabela chamada **produtos_venda**, que registrará os produtos vendidos a cada venda para o cliente (vamos colocar uma chave estrangeira de produtos e outra de **registro_venda**).



- Relacionamento **fornece**: as entidades envolvidas são fornecedores e produtos com cardinalidade N:N (muitos para muitos). Então, devemos fazer a união dos atributos chaves das duas entidades, como no caso anterior. Vamos criar a tabela **registro_entrada**, que terá atributo chave relacionada ao fornecedor, e a tabela **produto_entrada**, que listará os produtos entregues pelo fornecedor com chaves estrangeiras das tabelas **registro_entrada** e produtos.
- Relacionamento **armazena**: as entidades envolvidas são estoque e produtos com cardinalidade 1:1 (um para um). Sendo assim, não é necessário criar uma nova tabela, devemos apenas colocar o atributo chave estrangeira em cada uma das tabelas Estoque e Produtos.

Resumindo, após análise do diagrama, surgiram oito tabelas que deverão obedecer à distribuição das chaves definidas nesta etapa, são elas: clientes; produtos; estoque; fornecedores que vieram das entidades; e as tabelas registro_venda, produtos_venda, registro_entrada e produtos_entrada, que foram definidas a partir das relações. A Figura 27 apresenta o resultado final da transformação do modelo entidade-relacionamento para o relacional.

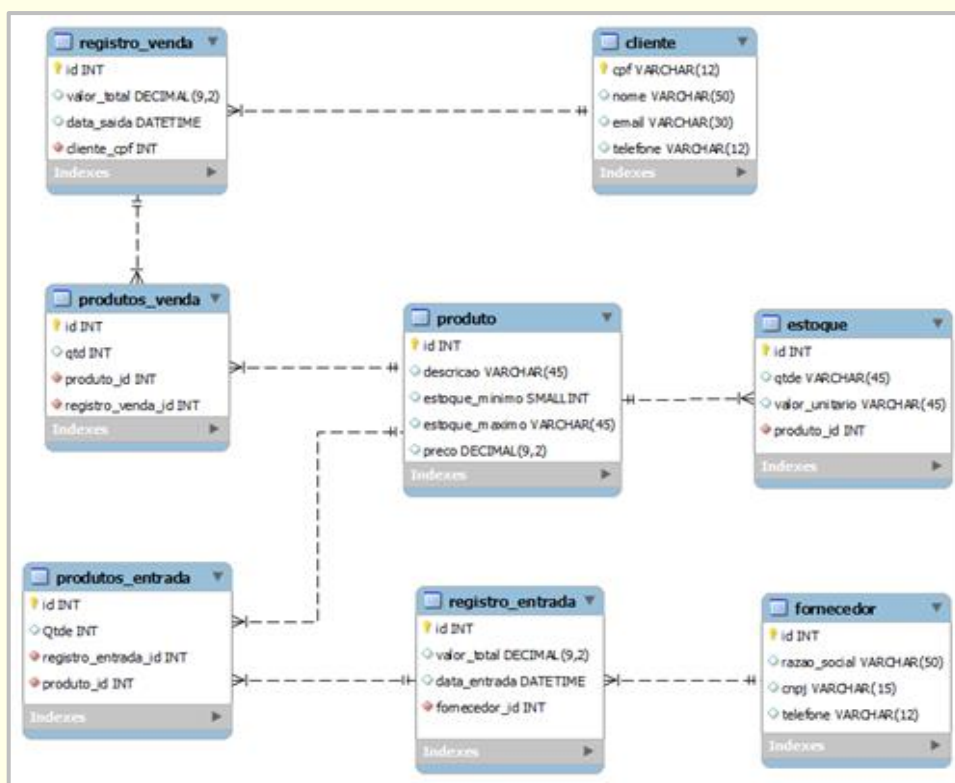


Figura 27 - Modelo relacional para um Supermercado

Fonte: os autores (2014)

Descrição: Nesta imagem temos a consolidação do modelo contendo todas as entidades e relacionamentos transformadas em uma nova representação desenhada na ferramenta de modelagem. As entidades que virarão tabelas são: registro_venda, cliente, produtos_venda, produto, estoque, produtos_entrada, registro_entrada e fornecedor

Para finalizar nossa modelagem, devemos documentar de uma forma mais clara tudo o que ficou definido. Para isso, criaremos o dicionário de dados.



Na modelagem Relacional, implantada pelos SGBD relacionais, existe um passo muito importante. Procure por “Normalização” junto com SGDB, para saber mais.

3.8 Dicionário de Dados

Junto com o MER, é necessário que se mantenha um documento com a explicação de todas as entidades nele criadas. Este documento, que pode ser chamado de **dicionário de dados**, permite que os analistas obtenham informações sobre todos os objetos do modelo de forma textual e direta. Ele vai conter explicações difíceis de incluir no diagrama. É válido lembrar que o objetivo do documento é ser claro e consistente. Para apresentação do dicionário de dados, devemos utilizar uma tabela para cada entidade. Este modelo que irei apresentar é um dos possíveis. Observe que outras colunas podem ser adicionadas, de acordo com a necessidade de explicação do modelo.

Antes de apresentar o dicionário de dados para cada uma das entidades, vamos explicar cada uma das colunas que compõem o dicionário.

- Entidade: é o nome da entidade que foi definida no MER.
- Atributo: deve-se colocar o nome do atributo.
- Classe: pode ser simples, composta, multivalorada, nula, derivada e chave. Consulte a seção da segunda competência onde falamos destes tipos de atributos.
- Domínio: pode ser numérico, texto, data e booleano. Aqui, usamos português mesmo para definir os tipos dos atributos. No banco iremos transformar para o tipo específico do banco.
- Tamanho: define a quantidade de caracteres que serão necessários para armazenar o seu conteúdo. Geralmente, o tamanho é definido apenas para atributos de domínio texto.
- Descrição: é opcional e pode ser usado para descrever o que é aquele atributo ou oferecer informações adicionais que possam ser usadas futuramente pelo analista ou programador do sistema.

Veja agora o dicionário de dados das oito tabelas identificadas no modelo ER do supermercado.

ENTIDADE: PRODUTO				
ATRIBUTO	CLASSE	DOMÍNIO	TAMANHO	DESCRIÇÃO
Id	Chave	Numérico		Atributo chave
Descricao	Simple	Texto	50	Descrição do produto, marca.
Estoque_minimo	Simple	Numérico		Quantidade mínima no estoque
Estoque_maximo	Simple	Numérico		Quantidade máxima no estoque
Preco	Simple	Numérico		Preço de venda do produto



ENTIDADE: ESTOQUE				
ATRIBUTO	CLASSE	DOMÍNIO	TAMANHO	DESCRIÇÃO
Id	Chave	Numérico		Atributo chave
qtde	Simples	Numérico		Quantidade de produtos no estoque
Valor_unitario	Simples	Numérico		O preço do Produto
Produto_id	Estrangeira	Numérico		Chave do relacionamento com a entidade produto

ENTIDADE: CLIENTE				
ATRIBUTO	CLASSE	DOMÍNIO	TAMANHO	DESCRIÇÃO
CPF	Chave	Texto	12	Atributo chave
Nome	Simples	Texto	50	Nome do Cliente
E-mail	Simples	Texto	30	E-mail do cliente
Telefone	Simples	Texto	12	Telefone do cliente

ENTIDADE: FORNECEDOR				
ATRIBUTO	CLASSE	DOMÍNIO	TAMANHO	DESCRIÇÃO
Id	Chave	Numérico		Atributo chave
Razao_social	Simples	Texto	50	Nome do fornecedor
CNPJ	Simples	Texto	15	CNPJ do fornecedor
Telefone	Simples	Texto	12	Telefone do fornecedor

RELACIONAMENTO: REGISTRO_VENDA				
ATRIBUTO	CLASSE	DOMÍNIO	TAMANHO	DESCRIÇÃO
Id	Chave	Numérico		Atributo chave
Data_saida	Simples	Data		Data da Venda
Valor_total	Calculado	Numérico		Valor total da venda
Cliente_CPF	Estrangeira	Texto	12	Chave estrangeira

RELACIONAMENTO: REGISTRO_ENTRADA				
ATRIBUTO	CLASSE	DOMÍNIO	TAMANHO	DESCRIÇÃO
Id	Chave	Numérico		Atributo chave
Data_entrada	Simples	Data		Data da Venda
Valor_total	Calculado	Numérico		Valor total da venda
Fornecedor_id	Estrangeira	Númerico		Chave estrangeira da tabela fornecedor



RELACIONAMENTO: PRODUTOS_VENDA				
ATRIBUTO	CLASSE	DOMÍNIO	TAMANHO	DESCRIÇÃO
Id	Chave	Numérico		Atributo chave
Qtde	Simples	Numérico		Quantidade do produto vendido.
Produto_id	Estrangeira	Numérico		Chave estrangeira da tabela produto
Saida_produto	Estrangeira	Numérico		Chave estrangeira da tabela registro venda

RELACIONAMENTO: PRODUTOS_ENTRADA				
ATRIBUTO	CLASSE	DOMÍNIO	TAMANHO	DESCRIÇÃO
Id	Chave	Numérico		Atributo chave
Qtde	Simples	Numérico		Quantidade do produto recebido.
Produto_id	Estrangeira	Numérico		Chave estrangeira da tabela produto
Registro_ven da_id	Estrangeira	Numérico		Chave estrangeira da tabela registro_venda

Até agora, você criou tabelas utilizando o phpMyAdmin e o MySQL Workbench. Outra forma de criar tabelas é através da linguagem SQL. Veja na próxima seção como criar tabelas utilizando comandos do subconjunto da DDL.

3.9 Criando Tabelas com a Linguagem SQL

A DDL ou Linguagem de Definição de Dados abrange comandos para criação e alteração de tabelas, índices e visões, são eles:

- CREATE TABLE: cria tabela;
- ALTER TABLE: alterar uma tabela existente;
- CREATE INDEX: cria índice para uma tabela;
- ALTER INDEX: altera um índice existente;
- DROP INDEX: remove um índice;
- CREATE VIEW: cria visão de uma tabela;
- DROP VIEW: remove visão existente.

Veja no quadro abaixo o comando SQL para a criação da tabela **produto**.

```
CREATE TABLE produto (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `descricao` VARCHAR(50),
  `estoque_minimo` INT(11),
  `estoque_maximo` INT(11),
  `preco` DECIMAL(9,2),
  PRIMARY KEY (`id`)
)
```



Cada linha do comando significa algo em específico:

- **CREATE TABLE produto:** o comando seguido do nome da tabela.
- **NOT NULL AUTO_INCREMENT:** NOT NULL indica que este campo não pode assumir o valor NULL, ou seja, não nulo. AUTO_INCREMENT indica que este campo será autoincrementado, aumentará em uma unidade cada vez que um novo registro for inserido.
- **PRIMARY KEY ('id'):** indica o campo que será a chave primária.

```
CREATE TABLE registro_entrada (  
  `id` INT(11) NOT NULL AUTO_INCREMENT,  
  `valor_total` DECIMAL(9,2) NULL DEFAULT '0.00',  
  `data_entrada` DATE NULL DEFAULT NULL,  
  `fornecedor_id` INT NOT NULL,  
  PRIMARY KEY (`id`),  
  FOREIGN KEY (`fornecedor_id` )  
    REFERENCES `supermercado`.`fornecedor` (`id` )  
    ON DELETE CASCADE  
    ON UPDATE CASCADE  
)
```

Observe agora a criação da tabela **registro_entrada**, que tem uma chave estrangeira.

Observe que nesta tabela foi necessário criar uma chave estrangeira.

- **FOREIGN KEY ('fornecedor_id'):** indica que fornecedor_id é um campo que será chave estrangeira
- **REFERENCES `fornecedor` ('id'):** indica que a tabela, a qual será feita a referência, através da chave estrangeira, é a tabela fornecedor e o campo desta tabela é id.
- **ON DELETE CASCADE:** significa que se um registro for deletado na tabela fornecedor, os registros correspondentes nesta tabela também serão deletados. Você verá os detalhes mais a frente.
- **ON UPDATE CASCADE:** significa que, se um registro for atualizado na tabela fornecedor, os registros correspondentes nesta tabela também serão atualizados.



A partir destes exemplos, tente construir o comando SQL para criar as demais tabelas do BD

Vamos agora, analisar o fluxo dos dados dentro do banco de dados.



3.10 Análise do Fluxo de Dados no Banco de Dados

Esta etapa é apenas para ficar mais claro como iremos trabalhar com o nosso banco de dados. A Figura 27 mostra o modelo relacional do banco de dados do supermercado.

Para começar, devemos inserir elementos nas tabelas. Mas por quais tabelas devemos começar a inserção?

Iniciamos pelas tabelas que não possuem chaves estrangeiras, pois a inserção de um elemento não dependerá inicialmente de nenhum valor cadastrado em outra tabela. Vamos lá? Insira os seguintes dados em suas respectivas tabelas.

- Tabela cliente:

CPF	NOME	EMAIL	TELEFONE
90878712354	Luiz Gonzaga	lula@gmail.com	87-98787676
98767678790	Marina da Silva	M_silva@uol.com.br	81-89760989
98787965454	Marcos Morais	mm@hotmail.com	81-89767654

- Tabela produto:

ID	DESCRICAO	ESTOQUE_MINIMO	ESTOQUE_MAXIMO	PRECO
123	Sabão em pó	12	100	1,29
134	Leite longa vida	15	100	3,56
235	Feijão preto	20	200	4,89

- Tabela fornecedor:

ID	CNPJ	RAZAO_SOCIAL	TELEFONE
1	29182738473625	Joao costa AS	81-90989876
2	23438372637489	Trafweu	88-98787657
3	98768764645346	Granada Ltda.	81-98789765

Mas, atenção! Para vender os produtos, o dono precisa inicialmente comprá-los dos fornecedores, para tê-los em estoque. Esta operação de inserção de mercadorias no estoque vai envolver o cadastro do fornecedor, o cadastro de produtos e ainda vai mexer com três diferentes tabelas que são as tabelas estoque, registro_entrada e produtos_entrada. Veja a Figura 28, mostrando a sequência de alterações.

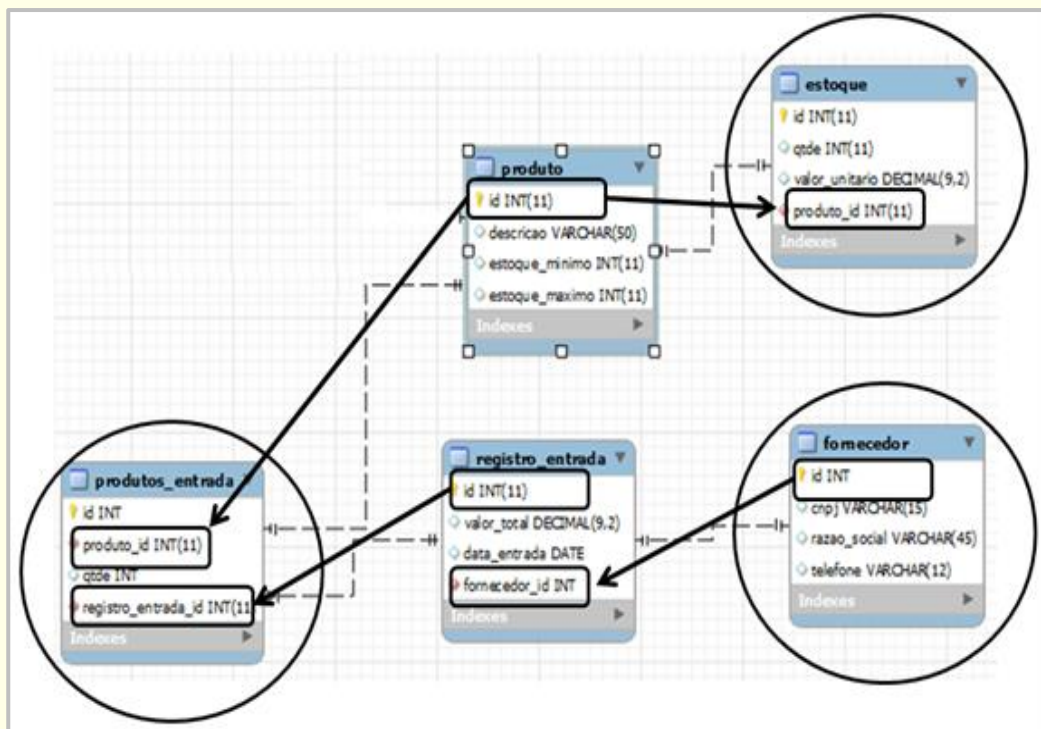


Figura 28 – Fluxo para inserção de um novo produto em estoque.

Fonte: os autores (2014)

Descrição: Nesta imagem temos o destaque de quais tabelas são acessadas para inserir um novo produto em estoque. Como exemplo temos chave estrangeiras de fornecedor presente em registro_entrada. Enquanto que a chave de produto está presente como chave estrangeira em produto_entrada.

Vamos imaginar que o fornecedor “Granada Ltda” cujo ID é 3, vai vender 100 caixas de “Sabão em pó” (ID=123) e 30 kg de “Feijão Preto” (ID=235). Vamos atualizar as tabelas corretamente. Iniciamos pela tabela registro_entrada, depois pela produtos_entrada e depois pela tabela de estoque.

- Tabela registro_entrada

ID	VALOR_TOTAL	DATA_ENTRADA	FORNECEDOR_ID
1	275,70	25/03/2013	3

O valor total é calculado assim: $\text{valor_total} = 100 \times 1,29 + 30 \times 4,89 = 275,70$. A próxima tabela irá registrar os produtos vendidos nesta venda, que foram o sabão em pó e o feijão preto.

- Tabela produtos_entrada

ID	PRODUTO_ID	QTDE	REGISTRO_ENTRADA
1	123	100	1
2	235	30	1



Observe que o campo `produto_id` é uma chave estrangeira que aponta para a tabela `produto` e o campo `registro_entrada` aponta para a tabela `registro_entrada`. Por fim, para concluir o registro da venda completamente, devemos registrar a entrada na tabela `estoque`.

- Tabela `estoque`

ID	QTDE	VALOR_UNITARIO	PRODUTO_ID
1	100	1,29	123
2	30	4,89	235

Pronto. Todas as tabelas estão devidamente atualizadas. Agora é possível realizar as vendas, pois a tabela `estoque` acusa um estoque de 100 caixas de sabão e 30 de feijão.

Caros (as) alunos (as), sei que até agora fizemos um grande exercício para entender todo este processo de inserção. Na prática, a inserção no banco de dados não acontece manualmente, devemos usar a linguagem SQL para isso. Vamos, agora, aprender a inserir registros em tabela utilizando a linguagem SQL.

3.11 Inserindo, Alterando e Removendo Registros com a Linguagem SQL

DML ou Linguagem de Manipulação de Dados é o subconjunto da linguagem SQL que abrange comandos para selecionar, inserir, atualizar e apagar dados, são eles:

- **INSERT:** insere um registro em uma tabela;
- **UPDATE:** muda os valores de dados em uma ou mais linhas de uma tabela;
- **DELETE:** remove linhas de uma tabela.

O comando para inserção nas tabelas do banco de dados com SQL é o comando **INSERT**. Veja como escrever este comando:

```
INSERT INTO cliente (`cpf`, `nome`, `email`, `telefone`) VALUES ('90878712354', 'Luiz Gozaga', 'lula@gmail.com', '87-98787676');
```

O comando **INSERT** é composto das seguintes partes, veja:

- **INSERT INTO:** o início do comando identifica o próprio comando.
- **cliente (`cpf`, `nome`, `email`, `telefone`):** é o nome da tabela e os registros que serão inseridos. Deve-se obedecer à ordem de inserção dos valores com a ordem dos campos colocados nesta parte do comando.
- **VALUES ('90878712354', 'Luiz Gozaga', 'lula@gmail.com', '87-98787676');** os valores que serão inseridos na tabela. Devem seguir a ordem dos nomes dos campos colocados na parte anterior do comando.



Viu como é fácil? Tente você mesmo inserir dados na tabela cliente, preenchendo os valores abaixo, com dados válidos (faça isto em um papel à parte):

```
INSERT INTO _____  
(_____, _____, _____, _____)  
VALUES  
(_____, _____, _____, _____);
```

O mesmo vale para as outras tabelas. Nas atividades de fixação, no fim desta competência, você poderá exercitar mais.

A remoção de registros das tabelas é feita com o comando **DELETE**. Vamos ver como este comando é executado. O comando abaixo irá remover o registro cujo CPF é igual a 90878712354.

```
DELETE FROM `cliente` WHERE cpf='90878712354'
```

Você poderá remover um registro por qualquer campo ou coluna da tabela. Veja este exemplo: será deletado o cliente com nome Luiz Gonzaga.

```
DELETE FROM `cliente` WHERE nome='Luiz Gonzaga'
```

Mas atenção. A remoção de registros de tabelas que tenham chave estrangeira em outras tabelas deve ser seguida da remoção de todos os registros que existirem nas tabelas relacionadas. Isto se deve à necessidade de se manter a integridade referencial no banco de dados. Esta operação é garantida pelo comando **ON DELETE CASCADE** que foi inserido durante a criação da tabela. O MySQL realiza esta operação automaticamente, veja a sequência de remoção da Figura 29.



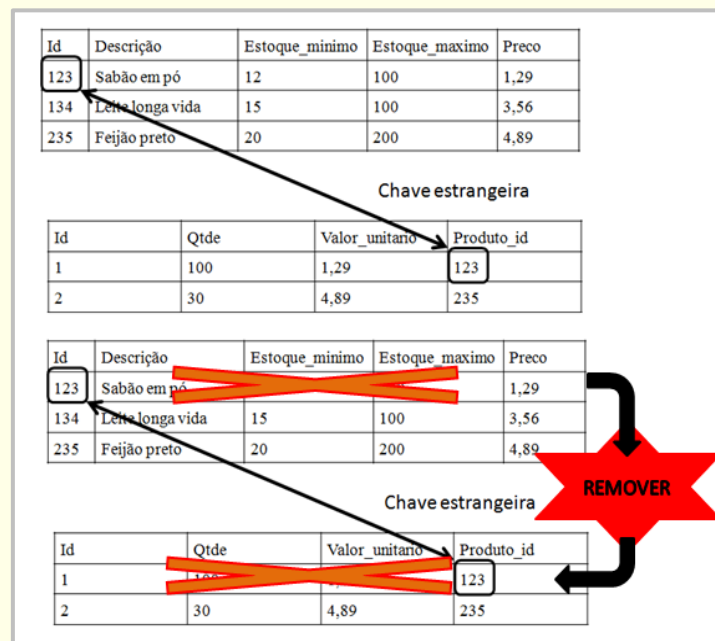


Figura 29 – Esquema de remoção automática pelo MySQL.

Fonte: os autores (2014)

Descrição: Nesta imagem podemos verificar que ao remover um registro de uma tabela que é dependente de outro o SGBD trata de remover ambos registros de ambas as tabelas.

Os comandos para remover e modificar registro usando SQL são muito semelhantes. A questão da integridade referencial também é garantida pelo comando **ON UPDATE CASCADE** que diz ao MySQL que uma chave ao ser alterada deverá “cascatear” esta modificação nos registros que fazem referência a esta chave nas tabelas.

Para modificar um registro com SQL devemos usar o comando **UPDATE**. Veja o exemplo de alteração da descrição de um produto.

```
UPDATE produto SET `descricao`='sabao em po bam-bam' WHERE id =908;
```

Explicando o comando:

- **UPDATE produto:** comando SQL para alterar um registro da tabela produto.
- **SET descricao='sabao em po bam-bam':** SET é a parte do comando que define qual ou quais campos deverão ser atualizados, no nosso caso o campo descrição passará a ter o valor 'sabao em pó bam-bam'.
- **WHERE id=908:** esta parte do comando indica qual o registro que será atualizado. Neste caso, o que tem o campo id = 908.

Agora que o nosso BD está com diversos dados, podemos realizar consultas mais elaboradas, extraindo o máximo de informação dos nossos dados. Para isso, vamos estudar como realizar consultas complexas utilizando a SQL.



3.12 Melhorando as Consultas com a Linguagem SQL

DQL ou Linguagem de Consulta de Dados é o subconjunto da linguagem SQL composto pelo comando SELECT. O comando SELECT é formado de várias cláusulas e opções, que possibilitam a criação de consultas mais elaboradas, são elas:

- Cláusulas FROM, WHERE, GROUP BY, HAVING, ORDER BY, DISTINCT
- Operadores Lógicos AND, OR, NOT
- Operadores Relacionais >, <, <>, <=, >=, =, BETWEEN, LIKE
- Funções de agregação AVG, COUNT, SUM, MAX, MIN

Nas primeiras competências, fizemos várias consultas para exemplificar o uso do comando SELECT. Agora, vamos aprender como fazer diversos tipos de consultas usando o comando SELECT e suas cláusulas.

Imagine que você gostaria de saber quais as quantidades de cada produto no estoque. A consulta é simples. Basta você dar um comando SELECT na tabela estoque:

```
SELECT produto_id, qtde FROM estoque;
```

Agora, imagine que você gostaria que a sua lista incluísse a descrição de cada produto. Esta pesquisa envolve duas tabelas: estoque e produtos. Preste bem atenção neste comando:

```
SELECT produto.descricao, estoque.qtde FROM produto, estoque WHERE  
produto.id=estoque.produto_id;
```

Serão listados os campos descrição da tabela produto que indicamos no comando produto.descricao e o campo qtde da tabela estoque que indicamos com o comando estoque.qtde. Depois da cláusula FROM, seguem os nomes das duas tabelas envolvidas na pesquisa. As duas tabelas serão unidas usando o critério da cláusula WHERE, que associa o campo id da tabela produto com o campo produto_id da tabela estoque, através do comando produto.id = estoque.produto_id.

Imagine que, na pesquisa anterior, você gostaria de ter mostrado os produtos em ordem alfabética crescente. Para isso, devemos usar o comando ORDER BY. Veja o exemplo:

```
SELECT produto.descricao, estoque.qtde FROM produto, estoque WHERE  
produto.id=estoque.produto_id ORDER BY produto.descricao
```

Este comando retornará uma lista de produtos com suas quantidades em estoque e ordenados pela descrição do produto.



Podemos usar uma série de comandos para realizar cálculos. Uma possibilidade é multiplicar valores de colunas com outras colunas. Por exemplo, qual o valor total de cada produto no estoque, ou seja, se existem 30 caixas de “sabão bam-bam” e cada uma custa 1,30, então o valor do estoque do “sabão bam-bam” é de $30 \times 1,30 = 39,00$. O comando abaixo calcula o valor de cada produto no estoque:

```
SELECT produto.preco*estoque.qtde FROM produto, estoque WHERE  
produto.id=estoque.produto_id
```

Também gostaríamos de saber quantos produtos temos cadastrados em nossa base de dados. Para isso, usamos o comando COUNT(). Veja o exemplo:

```
SELECT COUNT(descricao) FROM produtos
```

Este comando retorna apenas uma linha com a quantidade de produtos da tabela. Existem outras funções para realizar cálculos: MAX() retorna o maior valor, MIN() retorna o menor valor, SUM() retorna a soma dos valores.

Podemos realizar consultas e agrupar o resultado por um determinado campo. Por exemplo, gostaríamos de saber quanto cada cliente comprou no supermercado até hoje. Para isso, devemos usar o comando GROUP BY, unindo a uma função de cálculo, no caso, SUM(). O resultado é o código abaixo:

```
SELECT SUM(valor_total), cliente_cpf FROM registro_venda GROUP BY cliente_cpf
```

O resultado terá duas colunas: uma coluna contendo o CPF do cliente e outra a soma das suas compras.

Caro (a) aluno (a), chegamos ao fim do nosso curso de Banco de Dados. Lembro de que este é apenas um curso introdutório. É importante buscar outras fontes de estudo e se aprofundar no tema. Boa Sorte!



ATIVIDADES COMPLEMENTARES

1) Usando o esquema proposto para o nosso banco de dados e a linguagem SQL, tente realizar as seguintes inserções no banco de dados que criamos nesta competência:

a) Tabela Cliente:

CPF	NOME	EMAIL	TELEFONE
90878712987	Maria Aparecida	ma@gmail.com	87-98787676
98789876577	Tereza Cristina	t_silva@uol.com.br	81-89760989
98078676565	Jota Ramos Silva	jj@hotmail.com	81-89767654

b) Tabela produto:

ID	DESCRIÇÃO	ESTOQUE_MINIMO	ESTOQUE_MAXIMO	PRECO
423	Sabao barra JT	200	700	1,78
129	Pasta de Dente kk	15	100	3,56
987	Arroz tio Braz	50	200	4,89

c) Tabela Fornecedor:

ID	CNPJ	RAZÃO_SOCIAL	TELEFONE
4	29182738987678	Moinho Cristao	81-9098876
5	67654545677656	Brasil comida	88-98787865
6	89765546567898	Limpa Brasil	81-98788987

2) Realize as seguintes inserções nas tabelas apropriadas. Utilize a função CURDATE() para informar a data atual.

- a) O fornecedor “Limpa Brasil” vendeu 300 caixas de pasta de dente “KK” pelo preço de R\$ 1,43, e mais 50 caixas de Sabão barra “JT” pelo valor de R\$ 0,90.
- b) O fornecedor “Moinho Cristão” vendeu 200 quilos de Arroz “tio Braz” pelo preço de R\$ 3,45 o quilo.
- c) A cliente “Maria Aparecida” comprou 3 quilos de Arroz “tio Braz”, 2 pasta de dente “KK” e 1 barra de sabão “JT”.
- d) O cliente “Jota Ramos” comprou 5 pastas de dente “KK” e 1 kg de arroz “tio Braz”.

3) Realize as seguintes pesquisas no banco de dados.

- a) As compras realizadas pelo cliente Jota Ramos;
- b) Uma lista de todos os produtos da loja;
- c) A descrição do produto cujo id é 423.



Referências

Date, C. J. Introdução a Sistemas de Bancos de Dados. Editora Campus, 2000.

Elmasri, Ramez; Navathe, Shamkant B. Sistemas de Bancos de Dados. Addison-Wesley, 4ª edição, 2002.

Heuser, Carlos Alberto. Projeto de Banco de Dados. Vol. 4. Editora Bookman, 2004.

CERTIFICACAODB. Disponível em <http://certificacaobd.com.br/2012/05/09/visao-dos-dados/>. Acessado em março de 2013.

MIRELLA, Profa. Disponível em <http://homepages.dcc.ufmg.br/~mirella/DCC011/aula19.pdf>. Acessado em março de 2013.



Minicurrículo do Professor

- **Sérgio de Sá Leitão Paiva Júnior**

Graduado em Licenciatura em computação na UFRPE. Mestrado em Biometria pela UFPE. Atualmente faz doutorado em Bioinformática na UFPE. É professor pela UFRPE, Unidade Acadêmica de Serra Talhada-PE.

Possui experiência com ensino de computação nos níveis técnicos, superior e pós-graduação há mais de nove anos, nas disciplinas Banco de dados, Programação, Lógica de Programação e Estatística Computacional.

Já trabalhou em grandes empresas com desenvolvimento de software em Pernambuco, além de desenvolver vários softwares para empresas do estado.

- **Ellen Polliana Ramos Souza**

Doutoranda do curso de pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco (UFPE) e mestre em Engenharia da Computação pela Universidade de Pernambuco (2008).

Foi bolsista do curso Sequencial de Formação Complementar em Teste de Software oferecido pelo Centro de Informática da Universidade Federal de Pernambuco (UFPE), em parceria com a Motorola (2004).

Foi também Analista de Desenvolvimento do Serviço Federal de Processamento de Dados (SERPRO) e, desde 2009, é Professora Assistente da Universidade Federal Rural de Pernambuco (UFRPE), Unidade Acadêmica de Serra Talhada, lecionando disciplinas de Engenharia de Software, Teste de Software, Gerência de Projetos de Software, Programação Orientada a Objetos e Sistemas de Apoio à Decisão.

- **Francisco Airton Pereira da Silva**

Graduado em Sistemas de Informação pelo IFPI, com Especialização em Engenharia de Software pelo CEUT-PI. Mestre em Ciência da Computação pela Universidade Federal de Pernambuco e atualmente realizando doutorado pela mesma instituição desde 2013.

Possui experiência com ensino de computação em algumas faculdades de ensino superior. Em 2013 e 2014 exerceu a função de professor substituto na Universidade Federal de Pernambuco.

Foi analista de sistemas durante 4 anos em uma fábrica de software onde executou atividades de análise, desenvolvimento e testes de sistemas para gestão de planos de saúde.

