



Iniciativa da CNI - Confederação  
Nacional da Indústria

SÉRIE TECNOLOGIA DA INFORMAÇÃO - SOFTWARE

# CRIAÇÃO E MANIPULAÇÃO DE BANCO DE DADOS





SÉRIE TECNOLOGIA DA INFORMAÇÃO - SOFTWARE

# **CRIAÇÃO E MANIPULAÇÃO DE BANCO DE DADOS**



## **CONFEDERAÇÃO NACIONAL DA INDÚSTRIA – CNI**

*Robson Braga de Andrade*  
Presidente

## **DIRETORIA DE EDUCAÇÃO E TECNOLOGIA – DIRET**

*Rafael Esmeraldo Lucchesi Ramacciotti*  
Diretor de Educação e Tecnologia

## **SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL – SENAI**

### **Conselho Nacional**

*Robson Braga de Andrade*  
Presidente do Conselho Nacional

### **SENAI – Departamento Nacional**

*Rafael Esmeraldo Lucchesi Ramacciotti*  
Diretor-Geral

*Gustavo Leal Sales Filho*  
Diretor de Operações



*Iniciativa da CNI - Confederação  
Nacional da Indústria*

TECNOLOGIA DA INFORMAÇÃO - SOFTWARE

# **CRIAÇÃO E MANIPULAÇÃO DE BANCO DE DADOS**



© 2019. SENAI – Departamento Nacional

© 2019. SENAI – Departamento Regional de Santa Catarina

A reprodução total ou parcial desta publicação por quaisquer meios, seja eletrônico, mecânico, fotocópia, de gravação ou outros, somente será permitida com prévia autorização, por escrito, do SENAI.

Esta publicação foi elaborada pela equipe de Educação a Distância do SENAI de Santa Catarina, com a coordenação do SENAI Departamento Nacional, para ser utilizada por todos os Departamentos Regionais do SENAI nos cursos presenciais e a distância.

### **SENAI Departamento Nacional**

Unidade de Educação Profissional e Tecnológica - UNIEP

### **SENAI Departamento Regional de Santa Catarina**

Gerência de Educação

## **FICHA CATALOGRÁFICA**

---

S491c

Serviço Nacional de Aprendizagem Industrial. Departamento Nacional.  
Criação e manipulação de banco de dados / Serviço Nacional de  
Aprendizagem Industrial. Departamento Nacional, Serviço Nacional de  
Aprendizagem Industrial. Departamento Regional de Santa Catarina. Brasília :  
SENAI/DN, 2018.  
98 p. il. (Série Tecnologia da Informação – Software).

ISBN 978 - 85 - 505 - 2420 - 7

1. Computação. 2. Banco de dados. 3. Projeto de banco de dados. 4.  
Desenvolvimento organizacional. I. Serviço Nacional de Aprendizagem Industrial.  
Departamento Regional de Santa Catarina. II. Título. III. Série.

CDU: 004.65

---

### **SENAI**

Serviço Nacional de  
Aprendizagem Industrial  
Departamento Nacional

### **Sede**

Setor Bancário Norte • Quadra 1 • Bloco C • Edifício Roberto  
Simonsen • 70040-903 • Brasília – DF • Tel.: (0xx61) 3317-  
9001 Fax: (0xx61) 3317-9190 • <http://www.senai.br>

# Lista de Ilustrações

---

Figura 1 - Diagrama Simplificado de um Ambiente de Banco de Dados.....	17
Figura 2 - Arquitetura de 3 esquemas.....	21
Figura 3 - Representação do processo de <i>checkout</i> de um supermercado.....	23
Figura 4 - Representação dos itens do minimundo de um <i>checkout</i> de um supermercado .....	23
Figura 5 - Representação gráfica de entidades .....	25
Figura 6 - Representação gráfica de uma entidade com um atributo identificador .....	26
Figura 7 - Representação gráfica de uma entidade com um atributo não identificado .....	26
Figura 8 - Representação gráfica de uma entidade com um atributo multivalorado .....	26
Figura 9 - Representação gráfica de uma entidade com atributo composto.....	27
Figura 10 - Representação gráfica de um relacionamento entre duas entidades .....	27
Figura 11 - Representação gráfica de um relacionamento recursivo .....	28
Figura 12 - Representação gráfica de um relacionamento com atributos .....	29
Figura 13 - Representação gráfica de uma entidade com identificador simples .....	29
Figura 14 - Representação gráfica de uma entidade com identificador composto .....	30
Figura 15 - Representação gráfica do modelo lógico .....	31
Figura 16 - Representação gráfica do modelo conceitual de duas entidades .....	32
Figura 17 - Representação do modelo lógico de duas tabelas relacionadas .....	32
Figura 18 - Representação gráfica da inclusão de um novo registro na entidade cidade .....	33
Figura 19 - Representação gráfica da exclusão de um estado .....	34
Figura 20 - Representação gráfica da alteração da sigla do estado de uma cidade .....	34
Figura 21 - Tela Principal do brModelo.....	38
Figura 22 - Criação de entidade no brModelo .....	39
Figura 23 - Opções para criação de atributos no brModelo .....	39
Figura 24 - Alteração de atributo no brModelo.....	40
Figura 25 - Criação de relação no brModelo .....	40
Figura 26 - Ligando objetos no brModelo.....	41
Figura 27 - Ligando objetos no brModelo.....	41
Figura 28 - Cardinalidade do modelo conceitual no brModelo .....	42
Figura 29 - Modelo Conceitual do problema proposto sem atributos .....	43
Figura 30 - Modelo Conceitual do problema proposto com atributos.....	44
Figura 31 - Tela Principal do MySQL Workbench.....	45
Figura 32 - Seção Model Overview do MySQL Workbench.....	45
Figura 33 - Acesso a criação de tabela no MySQL Workbench .....	46
Figura 34 - Detalhamento de colunas em nova tabela no MySQL Workbench.....	46
Figura 35 - Detalhamento de colunas na tabela curso no MySQL Workbench .....	47
Figura 36 - Criação de um relacionamento n:m no MySQL Workbench .....	49
Figura 37 - Criação de um relacionamento recursivo no MySQL Workbench.....	50
Figura 38 - Criação de um relacionamento entre Disciplina e Turma no MySQL Workbench.....	50
Figura 39 - Modelo Conceitual no MySQL Workbench.....	51
Figura 40 - Connection Options de Forward Enginner no MySQL Workbench .....	56

Figura 41 - Informação da senha de conexão no MySQL Workbench.....	57
Figura 42 - Criação e Seleção de um Database no MySQL Workbench.....	58
Figura 43 - Consulta com utilização de uma view no MySQL Workbench .....	65
Figura 44 - Representação Gráfica de um Inner Join.....	70
Figura 45 - Execução de um Insert no MySQL Workbench .....	74
Figura 46 - Execução de Comando Update no MySQL Workbench.....	76
Figura 47 - Execução de uma transação com rollback no MySQL Workbench .....	78
Figura 48 - Execução de uma transação com Commit no MySQL Workbench .....	79
Quadro 1 - Inconsistências dos sistemas de arquivos.....	19
Quadro 2 - Principais tipos de dados no MySQL.....	48
Quadro 3 - Principais comandos de alteração de uma tabela .....	61
Quadro 4 - Os operadores de comparação da linguagem SQL.....	67
Quadro 5 - Os Operadores Lógicos da Linguagem SQL.....	68
Quadro 6 - Os Operadores Aritméticos da Linguagem SQL .....	69





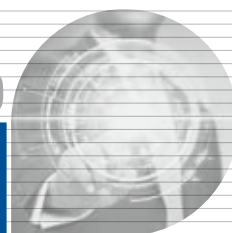


1 Introdução.....	13
2 Introdução aos Sistemas de Banco de Dados.....	15
2.1 Conceitos e características.....	16
2.2 Sistema de arquivos x Sistema de banco de dados .....	17
2.3 Arquitetura (Relacional e Não Relacional) .....	20
2.4 Modelagem de dados.....	20
2.4.1 Definição .....	20
2.4.2 Arquitetura de 3 esquemas.....	21
2.5 Modelo conceitual .....	21
2.5.1 Conceitos .....	21
2.5.2 Abstração de dados .....	22
2.5.3 Modelagem de dados usando o modelo entidade/relacionamento.....	24
2.5.4 Entidades .....	24
2.5.5 Atributos .....	25
2.5.6 Relacionamentos .....	27
2.5.7 Identificadores de entidades .....	29
2.6 Modelo lógico e físico.....	30
2.6.1 Modelo Lógico – Definição.....	30
2.6.2 Design e Restrições .....	33
2.6.3 Integridade referencial.....	33
2.6.4 Modelo Físico .....	35
3 Construção e Manutenção de Sistema de Gerenciamento de Banco de Dados .....	37
3.1 Criação do modelo conceitual no brModelo .....	38
3.1.1 Criação do modelo lógico.....	44
3.1.2 Dependência funcional .....	51
3.1.3 Normalização.....	52
3.2 Sistemas de Gerenciamento de Banco de Dados.....	53
3.2.1 Tipos.....	53
3.2.2 Instalação .....	53
3.3 Manipulação de banco de dados .....	54
3.3.1 Ferramentas.....	54
3.3.2 Linguagem SQL .....	55
3.3.3 DDL – Data Definition Language .....	58
3.3.4 DQL – Data Query Language.....	66
3.3.5 DML – Data Manipulation Language.....	73
3.3.6 DTL – Data Transaction Language .....	77
3.3.7 DCL – Data Control Language.....	79

4 Aspectos Organizacionais.....	83
4.1 Aspectos éticos da informática na sociedade.....	84
4.2 Diretrizes empresariais .....	88
4.2.1 Missão .....	88
4.2.2 Visão.....	88
4.2.3 Valores.....	89
4.2.4 Política de Qualidade.....	89
Referências.....	93
Minicurrículo do Autor .....	95
Índice .....	97







Olá, seja bem-vindo à Unidade Curricular Criação e Manipulação de Banco de Dados.

Vivemos a era da informação. São grandes sistemas empresariais, bancários, redes sociais, aplicativos, ambientes de aprendizagem a distância, jogos e tantas outras ferramentas que utilizamos. Estamos um momento que aprendemos a utilizar a tecnologia para facilitar a nossa vida diária, seja pessoal ou profissional.

Essas ferramentas são desenvolvidas utilizando as mais diversas tecnologias, mas têm sempre algo em comum: precisam utilizar alguma tecnologia de armazenamento de dados.

Quando você chama um táxi por um aplicativo, por exemplo, é possível identificar as avaliações dos condutores, o modelo e a placa do veículo? É possível ao motorista visualizar todas as corridas que ele fez e quanto irá receber? É possível que o motorista identifique seus dados como passageiro? Para todas essas perguntas a resposta é sim, e a tecnologia utilizada é de banco de dados.

Na maioria dos casos, é utilizado o tipo de banco de dados relacional, que foi criado há mais de 40 anos e segue evoluindo e sendo largamente utilizado. Na área de Tecnologia da Informação, a maioria das tecnologias utilizadas tem uma vida útil mais curta. Isso significa que aprender sobre banco de dados lhe traz um conhecimento que irá durar mais tempo.

Nesta unidade curricular, você conhecerá a estrutura de um sistema gerenciador de banco de dados, terá capacidade de modelar bancos de dados, criá-los e mantê-los. Isso lhe tornará apto a desenvolver a estrutura necessária para suportar as necessidades de armazenamento e recuperação de informação das mais diversas soluções em software que você irá desenvolver como um Técnico de Informática para Internet.

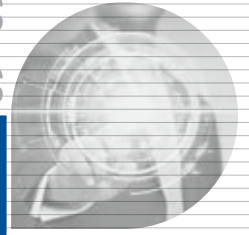
Bons Estudos!





# Introdução aos Sistemas de Banco de Dados

## 2



Aqui você vai conhecer os conceitos básicos da área de banco de dados que são necessários à compreensão de um projeto de banco de dados, tais como: sistema de gerenciamento de banco de dados, modelagem de dados e de banco de dados. Além disso, serão fornecidos os subsídios para permitir a criação modelos conceituais e lógicos de dados.

Sendo assim ao final deste capítulo, você terá subsídios para:

- a) compreender os conceitos básicos de banco de dados e como eles permeiam a vida diária;
- b) diferenciar os tipos de modelos de dados existentes e entender cada uma de suas aplicações;
- c) reconhecer os elementos que compõem o modelo conceitual;
- d) desenvolver abstração de dados e o modelo conceitual de uma situação proposta.

Inicie seus estudos conhecendo os conceitos e características de Banco de Dados.

## 2.1 CONCEITOS E CARACTERÍSTICAS

O desenvolvimento de software vem evoluindo dia após dia. Você consegue imaginar a sua vida sem a utilização de algum tipo de software? Suponha que no início de seu dia, você acorda com o despertador do celular, que já é controlado por um software. Na sequência, você procura o horário do próximo ônibus em um aplicativo. Se for ao supermercado, suas compras serão registradas em um sistema. Você lembra que precisa pagar a fatura de energia elétrica e utiliza para isso o Internet Banking de seu banco favorito.

Para que todos esses exemplos citados funcionem, dados precisam ser armazenados e recuperados. precisa é preciso criar estruturas que permitam esse armazenamento de forma íntegra, garantindo facilidade para consulta e manipulação.



### FIQUE ALERTA

Para Heuser (2009), um banco de dados é um conjunto de dados integrados que tem por objetivo atender a um grupo de usuários.

Já um Sistema de Gerenciamento de Banco de Dados (SGDB) é uma coletânea de programas que permite aos usuários criar e manter um banco de dados. É um software que permite a definição, construção, manipulação e compartilhamento de bancos de dados entre diversos usuários e aplicações. Quando falamos em definição de banco de dados, estamos dizendo a respeito de como os dados serão armazenados: sua forma, tipificação, estrutura e restrições. Já a construção é o artifício de armazenar esses dados em algum meio controlado pelo SGDB (Sistema de Gerenciamento de Banco de Dados). A manipulação envolve funções que consultam dados específicos ou atualizações nos dados para refletir mudanças ocorridas. O compartilhamento ocorre quando o banco de dados permite que vários usuários e/ou aplicações acessem e mantenham simultaneamente esses dados.

Na figura a seguir, é possível visualizar um esquema completo de banco de dados, uma aplicação que acessa um software de SGDB, que é composto por um software para processar consultas e um software para acessar os dados armazenados. Eles acessam duas bases de dados, uma que contém a definição do banco de dados, chamada de metadados, e outra que especificamente contém os dados armazenados.

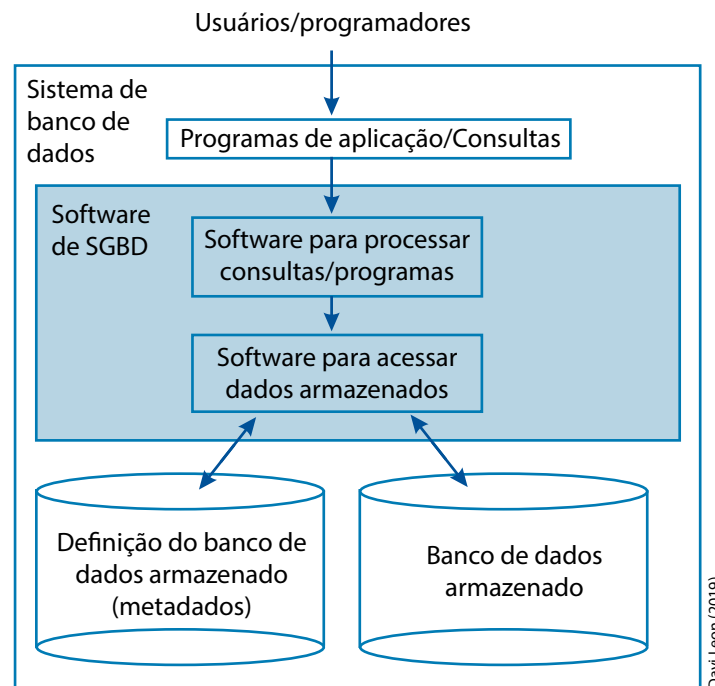


Figura 1 - Diagrama Simplificado de um Ambiente de Banco de Dados  
Fonte: Elmasri; Navathe (2011)

Os bancos de dados passaram por evoluções em suas estruturas com o passar dos anos. A seguir, você irá conhecer um pouco dessa história e os motivos pelo quais esse aperfeiçoamento foi necessário.

## 2.2 SISTEMA DE ARQUIVOS X SISTEMA DE BANCO DE DADOS

Os primeiros programas de computador utilizavam estruturas próprias para armazenar seus dados diretamente no sistema operacional. Essas composições estavam incorporadas diretamente nestes programas. Esse tipo de armazenamento é denominado Sistema de Arquivos. As bases de dados eram formadas por um conjunto destes arquivos espalhados pelo sistema operacional. Atualmente, esse tipo de armazenamento é ultrapassado, e não é mais utilizado para novas aplicações. Apenas sistemas que foram construídos a muito tempo, chamados de sistemas legados, se utilizam deste artifício. Mas um sistema de arquivos não lhe parece simples? Vários arquivos propagados no sistema operacional, sendo controlados diretamente pela sua aplicação, parecem um mundo ideal, não é? Parece, mas não é! A seguir, você verá porque os Sistemas de Gerenciamento de Banco de Dados vieram substituir os sistemas de arquivos na maioria das suas aplicações. Os sistemas de arquivos apresentam uma série de inconsistências, tais como:

**Redundância e inconsistência de dados:** considerando que os dados estão espalhados em arquivos soltos no sistema operacional, é grande a chance de manipulação por vários tipos de programas, mantendo informações repetidas e custo alto de armazenamento, com estas informações espalhadas;

**Dificuldade de acesso a dados:** os dados estão em arquivos comuns. Todo e qualquer acesso precisa ser programado especificamente. Não há meios facilitadores. Para cada necessidade nova é preciso uma implementação nova específica. Isso atrapalha profundamente a produtividade no desenvolvimento de software;

**Problemas de integridade:** não há um concentrador comum de validações de integridade. Por exemplo, num sistema de cadastro de veículos, foi definido que o campo placa tem três letras e 4 números (XXX-9999), nessa sequência. Porém, como a partir de 2018 o Brasil passou a utilizar o padrão do Mercosul, que é uma combinação de 4 letras e 3 números, essa restrição precisa ser alterada. Em um sistema de arquivos, será preciso percorrer todos os programas que manipulam esta informação e adaptar. Se algum local ficar de fora, teremos um problema de integridade;

**Problemas de atomicidade:** isso ocorre quando você tem uma interrupção abrupta no meio de um processamento. Imagine que você acessou o Internet Banking através do seu celular para pagar um boleto bancário. No meio do processamento ocorreu um erro. Nesse momento, a aplicação já tinha realizado o débito do saldo no seu arquivo de extrato. Porém ainda não tinha registrado a baixa do boleto. Nesse caso, o dinheiro saiu da conta, mas não efetivou o pagamento da conta. Quando se fala de atomicidade, isso quer dizer que um processo deve ter início, meio e fim. E que se, no meio o caminho, você tiver algum problema, deve retornar todos os dados ao estado inicial, para que a situação esteja completamente estornada;

**Anomalias de acesso concorrente:** esse tipo de problema ocorre quando dois usuários ou mais e/ou duas aplicações ou mais manipulam os mesmos dados simultaneamente. Imagine, por exemplo, o sistema de uma companhia aérea, que disponibiliza vendas pelo site e por *app* de celular. Dois usuários, simultaneamente, escolhem o mesmo trecho de voo, sendo que só havia uma poltrona disponível. Nesse caso, pode ocorrer de que os dois usuários consigam comprar a mesma poltrona, causando o que é chamado de overbooking. Para que isso não ocorresse, as aplicações precisariam implementar algum tipo de supervisão de controle simultâneo aos dados. Algo que não é simples, nem rápido;

**Problemas de segurança:** é importante lembrar, mais uma vez, que, em sistemas de arquivos, os dados estão armazenados de forma comum no Sistema Operacional, o que não impede que usuários não autorizados tenham acessos a informações estratégicas e confidenciais, como, por exemplo: salários, dívidas, saldos bancários, entre outros tantos dados organizacionais que por muitas vezes precisam ser mantidos em sigilo. Já os Sistemas de Gerenciamento de Banco de Dados têm nativamente características que resolvem naturalmente esses problemas. Você vai estudar sobre estas características e entender o porquê de a tecnologia de banco de dados estar sendo utilizada há tantos anos;

**Aprimoramento e compartilhamento de dados:** os SGDBs fornecem um ambiente que permite o compartilhamento de um grande volume de dados e da gerência facilitada deles;

**Aprimoramento da segurança de dados:** em um SGDB estão contidos mecanismos de controle de acesso em nível de grupos e de usuários. Você pode, por exemplo, determinar qual usuário terá ou não acesso a um ou mais conjuntos de dados. Pode, inclusive, determinar se um usuário pode só consultar, só inserir, só atualizar e se ele pode ou não excluir dados;

**Flexibilidade:** quando há necessidade de alterações estruturais devido, por exemplo, a inclusão de novos requisitos de negócio para um software, o SGDB permite que estas modificações sejam realizadas sem afetar o funcionamento das demais estruturas já existentes e utilizadas;

**Gerenciamento e armazenamento de dados:** o SGDB já tem seu próprio software de controle e armazenamento de dados. Você não precisa gastar tempo desenvolvendo o controlador dessas estruturas para cada aplicação que faz. Pode focar em desenvolver o seu software, se comunicando com o SGDB para que ele faça essa gerência;

**Gerenciamento de transações:** o gerenciamento de transações é uma das características essenciais de um SGDB. Por meio dele, você garante que os problemas de atomicidade e concorrência de usuários não ocorram e os dados permaneçam íntegros em nosso banco. Da mesma forma que o gerenciamento e armazenamento de dados, o SGDB tem seus próprios controles, dos quais você precisa apenas configurar e utilizar, sem a necessidade de implementação específica em cada programa construído.

Quadro 1 - Inconsistências dos sistemas de arquivos  
Fonte: Do autor (2019)



#### CURIOSIDADES

Você sabia que a Oracle é a maior companhia de banco de dados do mundo? Para entender melhor o domínio dessa empresa, veja estatísticas sobre os clientes dela:

- a) 10 das 10 principais companhias aeroespaciais e de defesa;
- b) 20 das 20 principais companhias aéreas;
- c) 20 das 20 maiores empresas automotivas;
- d) 20 dos 20 maiores bancos;
- e) 9 das 10 maiores empresas de bens de consumo;
- f) 9 das 10 principais empresas de engenharia;
- g) 20 dos 20 principais governos;
- h) 20 das 20 maiores empresas de alta tecnologia;
- i) 20 das 20 principais seguradoras;
- j) 20 dos 20 principais fabricantes;
- k) 20 das 20 principais empresas de dispositivos médicos;
- l) 20 das 20 maiores empresas de petróleo e gás;
- m) 20 das 20 maiores empresas farmacêuticas;
- n) 20 das 20 maiores empresas varejistas;
- o) 10 dos 10 principais fornecedores de SaaS (Software como Serviço);
- p) 20 das 20 principais empresas de cadeias de suprimentos;
- q) 20 das 20 principais empresas de telecomunicações;
- r) 20 das 20 melhores universidades.

Visite o site a seguir e saiba mais sobre essa grandiosa companhia: <<http://www.oracle.com/us/corporate/oracle-fact-sheet-079219.pdf>>.

Agora que você conheceu os motivos pelos quais, atualmente, é utilizada a tecnologia de Sistema de Gerenciamento de Banco de Dados em vez do Sistema de Arquivos, é importante conhecer dois padrões arquiteturais de banco de dados: o relacional e o não relacional.

## 2.3 ARQUITETURA (RELACIONAL E NÃO RELACIONAL)

Os bancos de dados relacionais são um padrão de mercado, e vem sendo utilizados em quase 100% de todos os sistemas que precisam utilizar armazenamento de dados. A tecnologia foi altamente difundida, os fabricantes de banco de dados se aprimoraram, definiram padrões em comum, enfim, é um grande sucesso, sem dúvida alguma. Porém, desde a última década, alguns desafios, principalmente vinculados ao volume de dados, fizeram surgir uma nova tecnologia de banco de dados. Para tanto, algumas iniciativas de bancos de dados não relacionais foram construídas. Os bancos de dados não relacionais, também chamados de NoSQL, são voltados para a escalabilidade, tendência forte das aplicações web. Em virtude da flexibilização necessária para garantir essa característica de distribuição, os bancos de dados não relacionais possuem características inversas as dos bancos de dados relacionais. Por exemplo, uma característica forte de um banco de dados relacional é da normalização, um conceito que determina que cada informação fica armazenada na entidade que lhe compete. Já em um banco de dados não relacional, a ideia é que dados que são utilizados juntos sejam mantidos juntos. Sobre a checagem de integridade de dados, ela é considerada de forma eventual, podendo ocasionar falha nos dados armazenados. Portanto, a utilização de bancos de dados não relacionais é uma medida a ser utilizada em determinados contextos dentro de uma aplicação. É muito discutível utilizá-la sem a necessidade de escalabilidade e alta disponibilidade, um cenário em que os bancos de dados relacionais atenderiam as necessidades tranquilamente.

## 2.4 MODELAGEM DE DADOS

Para a criação de um banco de dados de uma forma adequada, é necessário, primeiramente, realizar a modelagem deste banco de dados. A modelagem de banco de dados é tarefa essencial para que o banco de dados esteja alinhado às necessidades dos usuários que utilizarão os sistemas que estarão conectados nesse banco de dados. Outro viés importante é o de manutenção. Você já precisou realizar uma reforma em uma casa sem ter os projetos arquitetônicos e de engenharia? Da mesma forma, um banco de dados de uma determinada aplicação precisará ser modificado em algum momento. A existência de um bom modelo de dados facilitará este processo, maximizando a produtividade e minimizando impactos.

### 2.4.1 DEFINIÇÃO

Um modelo de banco de dados é uma descrição dos tipos de informações que estão armazenadas em um banco de dados. Para construir um modelo de dados, você usa uma linguagem de modelagem de dados. As linguagens de modelagem de dados podem ser classificadas de acordo com a forma de apresentar modelos, em linguagens textuais ou linguagens gráficas. Um mesmo modelo de dados pode ser apresentado de várias formas. Cada apresentação do modelo recebe a denominação esquema de banco de dados (HEUSER, 2009).

## 2.4.2 ARQUITETURA DE 3 ESQUEMAS

A arquitetura em 3 esquemas proporciona uma forma de entender um modelo de negócios que precisa ser modelado de forma clara e gradativa. Para representar como funciona esse esquema, veja a figura a seguir:



Figura 2 - Arquitetura de 3 esquemas  
Fonte: Do autor (2019)

O **modelo de processo de usuário** é como as atividades, tarefas e processos do dia a dia dos usuários para os quais se pretende desenvolver uma solução que envolva banco de dados está inserido.

O **modelo conceitual** já utiliza os conceitos de banco de dados, porém sem apego a qualquer tecnologia. Nessa etapa, estamos apenas modelando o domínio de negócio, delimitando o minimundo, sem considerar aspectos detalhados da modelagem.

Já o **modelo lógico** apresenta um modelo mais ligado à estrutura que o banco de dados terá quando for implementado. Detalhes mais específicos de regras e restrições são detalhados. Deve ser possível visualizar como todas as estruturas do banco de dados funcionarão, permitindo uma série de análises e otimizações.

No **modelo físico**, como seu nome já sugere, estamos falando de detalhes de métodos de armazenamento e de acesso aos dados. Reflete exatamente o banco a ser criado e manipulado. Tem ligação direta com o hardware e software empregado para implementar aquela solução de banco de dados.

## 2.5 MODELO CONCEITUAL

Dentro da divisão da modelagem de dados na arquitetura de três esquemas, o primeiro modelo a ser estudado (e também é o primeiro a ser desenvolvido em um projeto) é o modelo conceitual. A seguir, você estudará os conceitos e a linguagem visual que caracterizam o modelo conceitual.

### 2.5.1 CONCEITOS

A elaboração de um modelo conceitual é uma descrição do banco de dados de forma independente da sua implementação em um SGBD. Um modelo conceitual descreve que dados podem existir no banco de dados, mas não registra como estes dados estão armazenados no nível de SGBD físico. O modelo conceitual serve para que possamos ter uma ideia da estrutura de um banco de dados. O modelo conceitual tem uma linguagem visual simples, o que permite até que usuários leigos consigam compreender o modelo e fazer suas análises e observações.

### 2.5.2 ABSTRAÇÃO DE DADOS

A abstração de dados é o processo em que você deve compreender um problema do mundo real e traduzir para o universo de um modelo de dados, neste primeiro momento, um modelo conceitual. Esse processo tem origem na etapa de levantamento de requisitos, momento em que no papel de desenvolvedores de sistemas, são realizadas entrevistas, questionários, reuniões, observações junto aos usuários que estão demandando o software a ser desenvolvido.

Ao realizar este trabalho, você compreende como funciona o processo que pretende informatizar. Nesse momento, é essencial realizar a delimitação do que é chamado de minimundo. A demarcação do minimundo é essencial para que se possa seguir um foco determinado para o desenho da modelagem de dados de nosso sistema. Deve-se seguir a premissa da simplicidade. Aquilo que não é essencial não deve ser modelado. Quanto mais simples for nosso modelo de dados, melhor. Ser simples, não significa ser incompleto. Quando se menciona a simplicidade, está se levando em consideração que ele atende às necessidades identificadas nesse processo de levantamento de requisitos. Alguns autores chamam o minimundo de domínio do problema.



#### FIQUE ALERTA

O processo de levantamento de requisitos é considerado o mais importante do ciclo de desenvolvimento de software. Todas as etapas posteriores, inclusive a modelagem e a construção do banco de dados, são baseadas nas definições realizadas nessa etapa. Invista tempo e atenção nessa fase, para construir softwares de qualidade.

E na prática, de que estamos falando, quando o assunto é abstração de dados, minimundo, domínio do problema? Imagine uma situação conforme a figura a seguir:



Se você for solicitado a apresentar uma modelagem de dados em relação à situação anterior, o que pode identificar? Perceba que o processo exibido na imagem é de *checkout* das compras em um supermercado. Você já parou para pensar, como funciona esse processo?



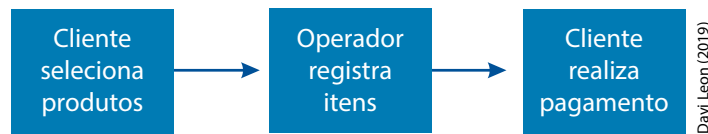


Figura 3 - Representação do processo de *checkout* de um supermercado  
Fonte: Do autor (2019)

Agora, você precisa pensar que tipos de informações do mundo real necessitam ser armazenadas no meio computacional dentro deste contexto. Ao analisar este caso, o primeiro conceito que vem à mente é do produto. Afinal, um conjunto de um ou mais produtos irá compor uma compra. Outro conceito bem importante é do pagamento. Ao concluir o registro dos itens da compra, o operador de caixa precisa informar qual a forma de pagamento. Dentro desse cenário, o cliente e o operador também são relevantes, para fins de registro. Pronto! Rapidamente, você consegue delimitar o minimundo a ser modelado na sequência. Veja a representação a seguir:



Figura 4 - Representação dos itens do minimundo de um *checkout* de um supermercado  
Fonte: Do autor (2019)



## CASOS E RELATOS

### Começando pelo fim

Certa vez, uma empresa de desenvolvimento de software foi contratada por uma grande indústria para desenvolver um novo sistema de controle de estoque. O Diretor orientou seu gerente a entregar um sistema funcionando em três meses. O gerente sem tempo para pensar em como atender uma demanda tão grande em tão pouco tempo, pressionou a sua equipe, afirmando o quanto era importante aquele contrato para a empresa. Os analistas de sistemas passaram imediatamente a desenvolver o modelo lógico para essa solução, visando logo poder construir o banco de dados e sair desenvolvendo este software rapidamente, afinal, já tinham desenvolvido outras soluções do estoque para outras empresas. O gerente ficou satisfeito com a rápida iniciativa e confiante. Ao passarem os três meses, Diretor, Gerente e Analista de Sistemas foram apresentar o software na Indústria. Tomaram um banho de água fria. Em 30 minutos de apresentação ficou claro que aquele software não serviria para aquela indústria. Ao retornar a empresa, todos se perguntaram: o que aconteceu? Com tanta experiência, como poderiam ter errado tanto?

A realidade daquela indústria era totalmente diferente de todas as outras experiências que os analistas de sistema haviam tido. A falha está na ausência de discussão sobre o processo dos usuários, como aquela indústria funcionava, a construção de um modelo conceitual e a exaustiva validação com os usuários-chave, para daí sim a conversão do modelo conceitual para o lógico e a posterior criação definitiva do banco de dados.

---

Muitas vezes, você irá realizar anotações, diagramas de processos entre outras representações para poder entender o domínio do problema especificado. Com esse entendimento em mãos, a próxima fase é traduzi-lo em um modelo conceitual de dados, conforme você verá a seguir.

### 2.5.3 MODELAGEM DE DADOS USANDO O MODELO ENTIDADE/RELACIONAMENTO

O modelo ER (Entidade Relacionamento) visa representar de forma conceitual o banco de dados, de modo a permitir o entendimento pelo usuário final. A ideia é representar objetos do mundo real que conhecemos e que estão envolvidos no contexto de negócio das organizações. Um modelo ER irá demonstrar elementos que permitem esse entendimento do contexto de negócio, como: entidade, atributos e relacionamentos. O esquema conceitual criado utilizando-se os conceitos do Modelo Entidade Relacionamento é denominado de Diagrama Entidade Relacionamento (DER).



#### CURIOSIDADES

Dentro das diversas carreiras na área de Tecnologia da Informação, o papel do DBA (Database Administrator ou Administrador de Banco de Dados) é um dos mais requisitados e bem-remunerados. Esse profissional é responsável pela manutenção, configuração, evolução e segurança dos bancos de dados de uma organização. Segundo o site Love Mondays, que é uma plataforma de pesquisa de satisfação profissional, a média salarial de um profissional desta área é de mais de R\$ 8.000,00 por mês.

### 2.5.4 ENTIDADES

Entidades são representações de objetos do mundo real, dentro de um banco de dados. Elas podem representar objetos físicos, organizações, pessoas, lugares, processos.

As entidades são o conceito fundamental do modelo entidade relacionamento. Para um modelo de dados, o que importa são apenas os objetos que se deseja manter informações. Estes objetos serão as entidades de seu banco de dados.

Dentro do modelo conceitual, as entidades são representadas por meio de um retângulo com a denominação da entidade ao centro.

**FIQUE ALERTA**

É importante que você tenha certo cuidado ao definir os nomes das entidades. O nome de uma entidade deve ser claro o bastante para que o interlocutor que esteja avaliando o modelo de dados consiga interpretar do que a entidade se trata. É importante manter a nomenclatura no singular, pois, sendo um banco de dados, é óbvio que vários registros serão armazenados ali.

Exemplos de entidades: veículo, imóvel, produto, venda, vendedor, pagamento, banco, agência.



Figura 5 - Representação gráfica de entidades  
Fonte: Do autor (2019)

Na figura apresentada, cada retângulo representa um conjunto de objetos sobre os quais se deseja guardar informações. Assim, o primeiro retângulo designa o conjunto de todos os bancos sobre as quais se deseja manter informações no banco de dados, enquanto o segundo retângulo designa o conjunto de todas as agências das quais se deseja manter informações. Caso seja necessário se referir um objeto particular (uma determinada agência ou um determinado banco), chamamos de ocorrência da entidade.

## 2.5.5 ATRIBUTOS

São características que definem uma entidade ou um relacionamento. Na prática, os atributos não são representados graficamente, para não sobrecarregar os diagramas, pois uma entidade geralmente tem um grande número de atributos. Mas, em modelos conceituais, eles aparecem principalmente para entendimento do escopo do modelo.

Os atributos podem ser classificados da seguinte forma, acompanhe!

### ATRIBUTO IDENTIFICADOR

É representado por meio do círculo preenchido na extremidade do atributo. Atributos identificadores identificam ou compõem a identificação única de uma ocorrência em uma entidade.

**FIQUE ALERTA**

É importante frisar que uma entidade e/ou relacionamento pode possuir mais de um atributo identificador, desde que estes, em conjunto, componham a identificação única.



Figura 6 - Representação gráfica de uma entidade com um atributo identificador  
Fonte: Do autor (2019)

## ATRIBUTO NÃO IDENTIFICADO

Representado por meio de um círculo vazio na extremidade do atributo. A grande maioria dos atributos de uma entidade, serão um atributo não identificado. Um ponto importante a se mencionar em relação aos atributos não identificados é que podem ser opcionais, ou seja, em algumas instâncias da entidade estarão nulos (não preenchidos). A informação se um atributo permite ou não nulo é essencial para compreensão do modelo, e devemos refletir sobre essa propriedade de cada um dos atributos de uma entidade.



Figura 7 - Representação gráfica de uma entidade com um atributo não identificado  
Fonte: Do autor (2019)

## ATRIBUTOS MULTIVALORADOS

Representado pela sua cardinalidade demonstrada na extremidade do atributo. O primeiro número identifica o número mínimo de ocorrências, e o segundo, o número máximo. Os atributos multivalorados são utilizados para representar dados que podem ter mais de uma ocorrência para a mesma instância da entidade. O exemplo mais clássico é o do telefone. Geralmente, temos mais de um número de telefone para contato, então para este caso utilizaríamos um atributo multivalorado.

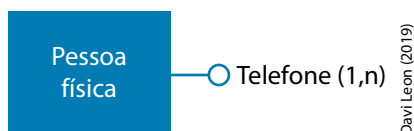


Figura 8 - Representação gráfica de uma entidade com um atributo multivalorado  
Fonte: Do autor (2019)

## ATRIBUTOS COMPOSTOS

Representados por meio de uma oval com vários nós na extremidade do atributo.

Um atributo composto é utilizado quando uma informação contém várias partes. O melhor exemplo é o do endereço. Um endereço é composto por rua, número, bairro, cidade, estado, cep, e pode ser representado de forma conjunta.

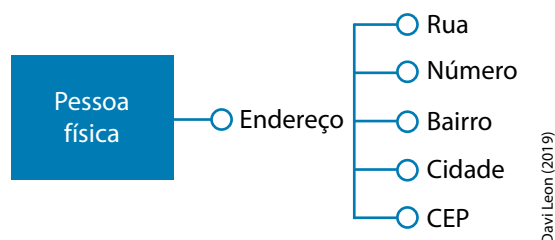


Figura 9 - Representação gráfica de uma entidade com atributo composto  
Fonte: Do autor (2019)

### 2.5.6 RELACIONAMENTOS

Um relacionamento dentro de um banco de dados é uma associação entre entidades. As entidades que estão vinculadas a um relacionamento também são chamadas de entidades participantes.

O relacionamento deve ser sempre identificado por um verbo, que permitirá que a pessoa que estiver lendo o modelo de dados consiga compreender a motivo da existência daquele relacionamento. Outra característica importante é que um relacionamento sempre opera em ambas as direções.

No modelo conceitual, um relacionamento é representado por um losango, com o verbo que o representa contido dentro do losango.

Voltando ao exemplo que foi utilizado para entender o conceito de abstração de dados, do supermercado, segue um exemplo de relacionamento entre duas entidades.



Figura 10 - Representação gráfica de um relacionamento entre duas entidades  
Fonte: : Do autor (2019)

São representadas as duas entidades, Venda e Produto. Entre elas, o relacionamento é representado pelo losango e seu verbo identificador. Próximo de cada entidade, envolvida no relacionamento é apresentado a sua cardinalidade no relacionamento. Esse conceito será detalhado melhor, por enquanto você irá aprender a fazer a leitura deste diagrama. Lembre que foi conceituado que o relacionamento deve operar em ambas as direções? Isso significa que você deve efetuar a leitura também nos dois sentidos.

Uma venda contém um ou muitos produtos e um produto pode estar contido em nenhuma ou em muitas vendas.

Primeiro fato: quando fazer a leitura, a cardinalidade deve ser lida do lado mais próximo da entidade destino. O próximo passo é entender que existe uma cardinalidade mínima e uma cardinalidade máxima nos relacionamentos. O primeiro número dentro dos parênteses é a cardinalidade mínima, e o segundo a cardinalidade máxima.

**FIQUE ALERTA**

Cardinalidade é a quantidade de ocorrências de uma entidade que podem estar associadas a uma determinada ocorrência por meio do relacionamento.

Agora, você tem condições de entender que a cardinalidade mínima é o número mínimo de ocorrências de uma entidade podem estar associadas a outra, e máxima o número máximo. No exemplo anterior, uma venda deve conter no mínimo um produto e no máximo  $n$  produtos, o que significa que não há um limite específico. Geralmente, o símbolo  $n$  é lido como muitos. Já um produto pode não estar em nenhuma venda, pois a sua cardinalidade mínima é zero. Mas o mesmo produto pode estar em várias vendas representado pelo símbolo  $n$ .

Agora pense um pouco sobre outros tipos de relacionamentos.

Será que relacionamentos só acontecem entre entidades diferentes? E se você precisar representar um caso do mundo real em que uma ocorrência da mesma entidade se relaciona com outra? Vamos à prática?

Imagine que você está desenvolvendo um software de Recursos Humanos de uma empresa. Um dos principais conceitos deste sistema é o Funcionário, não é mesmo? Sempre é importante também conhecer a hierarquia da empresa e saber quem é o superior imediato de cada funcionário. Pensando assim, a entidade funcionário precisa se relacionar consigo mesma! Sim isso existe, e se chama relacionamento recursivo. Veja a representação a seguir.

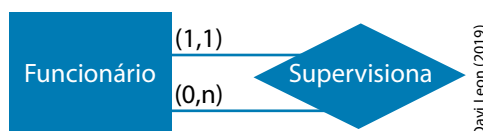


Figura 11 - Representação gráfica de um relacionamento recursivo  
Fonte: Do autor (2019)

Vamos fazer a leitura, já praticando a cardinalidade? Então:

Um funcionário supervisiona zero ou muitos funcionários e um funcionário é supervisionado por um e somente um funcionário.

Continuando a estudar outros tipos de relacionamentos, veja um exemplo em que um ou mais atributos são definidos para um relacionamento. Você pensava que atributos eram ligados apenas a entidades, não é mesmo?

Agora pense em exemplo de um consultório dentário. Um dentista realiza o atendimento de seus pacientes diariamente. É importante registrar a data/horário desses atendimentos e os procedimentos realizados. Dê uma olhada no modelo a seguir:

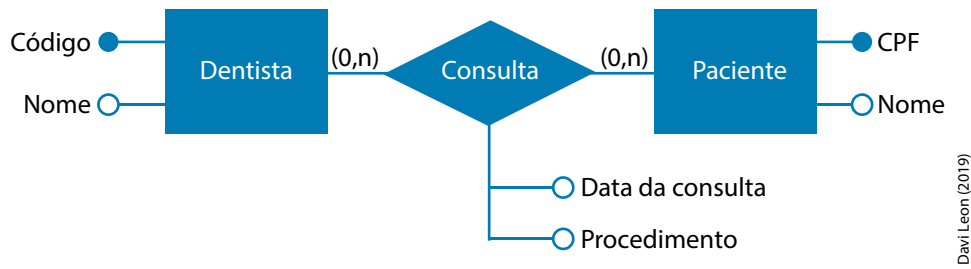


Figura 12 - Representação gráfica de um relacionamento com atributos  
Fonte: Do autor (2019)

Agora você irá fazer a leitura desse relacionamento: um dentista consulta muitos pacientes. Um paciente pode ser consultado por muitos dentistas. Observe que para cada consulta são registrados a data de consulta e qual procedimento foi realizado. Dessa forma, é possível saber quando o paciente esteve na clínica, por quem foi atendido e o que foi realizado. Pensando por outro lado, também é possível saber quantos atendimentos um dentista fez num dia, para quem, e o que foi realizado nesses atendimentos.

### 2.5.7 IDENTIFICADORES DE ENTIDADES

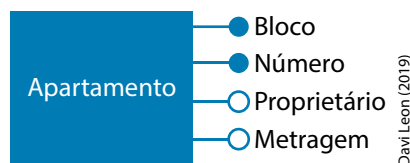
Retornando ao conceito da entidade, é preciso falar sobre a necessidade de toda entidade ter um identificador. O identificador de uma entidade pode ser formado por um único atributo ou por um conjunto deles. A utilidade do identificador de uma entidade é distinguir cada ocorrência de uma entidade as outras. No modelo conceitual, os atributos identificadores são simbolizados pelo círculo preenchido. Quando é composto por apenas um atributo, é chamado de identificador simples, conforme o exemplo a seguir.



Figura 13 - Representação gráfica de uma entidade com identificador simples  
Fonte: Do autor (2019)

Para a entidade Pessoa Física, você define o atributo CPF como identificador. Você pode estar se perguntando: poderia ter definido o nome, não é mesmo? Mas você já parou para pensar quantas pessoas tem exatamente o mesmo nome? Quantos João da Silva existem? E Pedro Souza? E Maria Fernandes? Não se pode identificar uma entidade através de um atributo que pode correr o risco de ser duplicado entre ocorrências diferentes da entidade. O CPF, neste caso, é o mais indicado, pois, considerando as regras da Receita Federal do Brasil, cada número de CPF (Cadastro de Pessoa Física) se refere a uma única pessoa.

Agora, será trabalhado com o exemplo de identificador composto. Veja a seguir a representação de uma entidade de um sistema de condomínio:



Dav Leon (2019)

Figura 14 - Representação gráfica de uma entidade com identificador composto  
Fonte: Do autor (2019)

Para explicar o caso anterior, será falado um pouco do contexto. Em um sistema de condomínio, é necessário registrar quem é o proprietário de cada apartamento e sua metragem. Esse condomínio é dividido em 10 blocos com 12 apartamentos por andar. Nesse caso, a numeração de cada unidade é feita da seguinte forma: Número do Andar (de 1 a 6) + Número do Apartamento (1 a 12). Agora você vai entender o porquê de ter definido também o bloco como identificador. Caso você precise identificar uma unidade, só com o número, você conseguiria? Pense o seguinte: você chega na portaria deste prédio e diz ao porteiro, eu vim entregar uma encomenda no apartamento 408. Com certeza o porteiro te responderá com uma pergunta: 408 de qual bloco? Portanto fica claro que para identificar unicamente um apartamento neste condomínio precisamos da junção de Bloco + Número.

## 2.6 MODELO LÓGICO E FÍSICO

Os modelos lógico e físico de banco de dados são a continuidade da divisão em três camadas da modelagem de dados. O lógico faz uma ponte entre o desprendimento da tecnologia abordada característica do modelo conceitual e a rigidez da vinculação a uma tecnologia presente no modelo físico.

### 2.6.1 MODELO LÓGICO – DEFINIÇÃO

Em um projeto de banco de dados, a segunda etapa é construir um modelo lógico. O modelo lógico é um desenvolvido a partir do modelo conceitual, e diferentemente deste, traz uma visão mais do como este modelo será implementado no banco de dados, do que uma visão abstrata de negócios.

Nessa etapa, já é necessário se preocupar com aspectos de nomenclatura e restrições mais completas de consistência e integridade. Nesse momento é que você irá decidir pela forma com que as estruturas que armazenarão os dados da nossa aplicação serão construídas. O modelo lógico também é chamado de modelo relacional.

Existem algumas conversões de nomenclatura entre o modelo conceitual e modelo lógico relacional, o que era chamado de entidade, agora é tabela ou relação, o que era chamado de atributo, é uma coluna, os relacionamentos são chamados de chaves estrangeiras, entre outras conversões que são descritas a seguir.



## CHAVE PRIMÁRIA, CHAVE CANDIDATA E CHAVE ESTRANGEIRA

Dentro do modelo conceitual, é usado o conceito de atributos identificadores. Já no modelo lógico relacional, esse conceito se transporta para chave primária e chave candidata. Esses dois tipos de chaves são muito parecidos. Será explicado aqui primeiro a chave Candidata. Uma chave candidata é composta por um ou mais colunas que identificam unicamente aquela tabela. Uma tabela pode ter uma ou mais chaves candidatas. Será utilizado um exemplo para ficar mais claro! Pense em sistema do Departamento de Trânsito Oficial que precisa armazenar os dados sobre os veículos. Naturalmente, será necessário modelar uma tabela de veículos. Veja o desenho desta tabela já utilizando a notação do modelo lógico.

VEÍCULO
Placa:
Renavan:
Chassi:
Marca:
Modelo:
Cor:

Davi Leon (2019)

Figura 15 - Representação gráfica do modelo lógico  
Fonte: Do autor (2019)

Visualizando as colunas definidas, qual delas poderia ser utilizada como identificadora da tabela Veículo? Pensando bem, tanto a Placa quanto o Renavan (que é o registro nacional dos veículos) e o número do Chassi poderiam ser utilizadas como identificadores únicos, não é mesmo? Então, estas são as chaves candidatas. São chaves candidatas a chave primária, como se fosse um concurso. A vencedora deste concurso será a chave primária. E como escolher entre as chaves candidatas, qual será a chave primária? Primeiramente, precisamos verificar, entre as chaves candidatas, quais terão uma maior utilização pelos usuários do banco de dados. Se formos pensar nesse caso dos veículos, a placa é mais comum, não é mesmo? Não há prejuízo na escolha de outra coluna, porém quanto mais fácil de identificar aquela tabela, melhor.

A chave primária é utilizada para identificar unicamente cada registro de dados de uma tabela. Também é utilizada para formalizar os relacionamentos entre as tabelas, portanto a definição correta dela é muito importante! Imagine uma tabela que é central em um sistema e se relaciona com outras 20 tabelas. Essa chave primária será transportada para outras 20 tabelas como chave estrangeira, e se precisar alterar isso no futuro será uma grande dor de cabeça. É necessário reforçar algumas características essenciais das chaves primárias:

- a) ser unívoca, ou seja, ter um valor único para cada registro daquela tabela;
- b) ser não nula, ou seja, nenhum dos atributos que compõe a chave primária podem estar vazios;
- c) ser não redundante, no caso de uma chave primária composta, não devem ser incluídos mais atributos do que o mínimo necessário para se identificar aquela entidade.

Falou-se de chave estrangeira mais acima, e agora é preciso explicar detalhadamente isto: primeiramente, você precisa entender o nome da chave, estrangeira não significa que ela veio de outro país, e sim que ela veio de outra tabela. A fronteira neste caso são as tabelas.

As chaves estrangeiras são o meio de materializar as relações entre as tabelas. Será utilizado como exemplo as tabelas de estado e cidade. Veja o modelo conceitual:

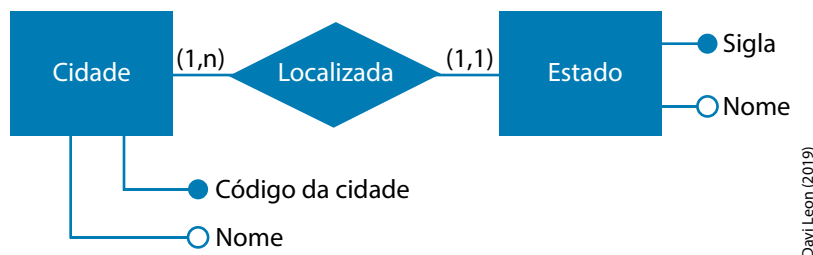


Figura 16 - Representação gráfica do modelo conceitual de duas entidades  
Fonte: Do autor (2019)

Já no modelo lógico, isso vai ser transformar, em duas entidades ligadas por uma chave estrangeira, veja a seguir a representação inclusive com registros:

ESTADO		CIDADE		
SIGLA	NOME	CÓDIGO	NOME	SIGLA_ESTADO
RJ	Rio de Janeiro	1	Búzios	RJ
PR	Paraná	2	Fortaleza	CE
AC	Acre	3	Rio Branco	AC
AM	Amazonas	4	Manaus	AM
CE	Ceará	5	Niterói	RJ
		6	Pato Branco	PR

Figura 17 - Representação do modelo lógico de duas tabelas relacionadas  
Fonte: Do autor (2019)

Considerando que um Estado pode conter mais de uma cidade, e que uma cidade pode estar localizada em apenas um Estado, a tabela Cidade recebe o atributo Sigla do Estado, que é a chave primária da tabela Estado. A tabela Cidade, portanto, fica com três colunas, Código, Nome e Sigla do Estado. Dessa forma, sempre que for cadastrada uma Cidade, deve ser informado seu código, seu nome e qual seu Estado. É importante mencionar que só podem ser informados Siglas de Estados que estejam cadastrados na tabela Estado. Esse conceito se chama integridade referencial, do qual você verá mais detalhadamente a frente.

### 2.6.2 DESIGN E RESTRIÇÕES

Nesta etapa de construção do modelo lógico, você se torna realmente arquiteto de dados. O conceito de design remete a concepção de um determinado produto em relação a sua forma física e funcionalidade. É o que você faz nesta etapa para construir o modelo lógico. Talvez você esteja se perguntando por que se criar primeiro o modelo conceitual, e não diretamente o modelo lógico? Para responder essa pergunta, você precisa reforçar que o modelo conceitual é um modelo de definição, mas alinhado aos negócios do que a como este banco de dados será implementado. O modelo conceitual pode e deve passar por uma série de validações com os usuários chave do sistema a ser desenvolvido, para que depois de bem consolidado seja evoluído para o modelo lógico. Dessa forma, obtemos produtividade, pois só iremos construir o modelo lógico após o modelo conceitual ser considerado aprovado.

O modelo lógico é construído por meio da lógica dos predicados e na teoria dos conjuntos. No modelo lógico, os dados são representados através de relações, que são costumeiramente chamadas de tabelas.

### 2.6.3 INTEGRIDADE REFERENCIAL

O conceito de integridade referencial é um dos que mais justifica a nomenclatura dos bancos de dados que estão utilizando, bancos de dados relacionais. A integridade referencial é a propriedade que garante que os dados que compõe uma chave estrangeira estejam sempre íntegros. Significa que nenhum dado em uma tabela destino de uma chave estrangeira não esteja contido na tabela origem, e que nenhum dado contido na tabela origem seja alterado ou excluído enquanto estiver vinculado a tabela destino. Veja novamente o exemplo das cidades e estados:

ESTADO		CIDADE		
SIGLA	NOME	CÓDIGO	NOME	SIGLA_ESTADO
RJ	Rio de Janeiro	1	Búzios	RJ
PR	Paraná	2	Fortaleza	CE
AC	Acre	3	Rio Branco	AC
AM	Amazonas	4	Manaus	AM
CE	Ceará	5	Niterói	RJ
		6	Pato Branco	PR
		7	Tubarão	SC

Davi Leon (2019)

Figura 18 - Representação gráfica da inclusão de um novo registro na entidade cidade  
Fonte: Do autor (2019)

No exemplo anterior, você está simulando a inclusão de uma nova cidade, com os seguintes valores: para código, 7; para nome, Tubarão; e para sigla do estado, SC. Considerando que há uma chave estrangeira entre Cidade e Estado, representada através da chave primária de estado, criada como atributo sigla\_estado em cidade, podemos concluir que esta inserção causaria uma violação da integridade referencial, pois "SC" não é a Sigla de nenhum dos Estados contidos na tabela Estado. O SGDB, neste caso, não permitiria a operação e emitiria uma mensagem de erro.

ESTADO		CIDADE		
SIGLA	NOME	CÓDIGO	NOME	SIGLA_ESTADO
RJ	Rio de Janeiro	1	Búzios	RJ
PR	Paraná	2	Fortaleza	CE
AC	Acre	3	Rio Branco	AC
AM	Amazonas	4	Manaus	AM
CE	Ceará	5	Niterói	RJ
		6	Pato Branco	PR

Davi Leon (2019)

Figura 19 - Representação gráfica da exclusão de um estado  
Fonte: Do autor (2019)

Já no caso anterior, você irá simular a exclusão do Estado AC – Acre. Ao solicitar esta operação ao SGDB, ele irá verificar que há uma cidade, neste caso “Rio Branco” vinculada ao estado “AC” – Acre. Este também é um caso de violação de integridade referencial, o que fará com que o SGDB retorne uma mensagem de erro.

ESTADO		CIDADE		
SIGLA	NOME	CÓDIGO	NOME	SIGLA_ESTADO
RJ	Rio de Janeiro	1	Búzios	RJ
PR	Paraná	2	Fortaleza	CE
AC	Acre	3	Rio Branco	AC
AM	Amazonas	4	Manaus	AM
CE	Ceará	5	Niterói	RJ
		6	Campo Grande	RJ -> MS

Davi Leon (2019)

Figura 20 - Representação gráfica da alteração da sigla do estado de uma cidade  
Fonte: Do autor (2019)

No exemplo anterior, você está simulando a atualização da sigla do estado da cidade de Campo Grande. Estava registrado como “RJ”, mas o correto é “MS” – Mato Grosso do Sul. Ocorre que “MS” não é a sigla de nenhum estado registrado na tabela de estados. Essa atualização infringe a regra de integridade referencial e o banco também retornará uma mensagem de erro.

Em resumo, a integridade referencial é um guardião da qualidade dos dados de um banco. Essas regras são checadas a todo o momento que um dado vinculado a um relacionamento é modificado, seja na sua tabela de origem (chave primária) ou na sua tabela destino (chave estrangeira).

### 2.6.4 MODELO FÍSICO

É uma descrição de um banco de dados no nível de abstração visto pelo usuário do SGBD. Dessa forma, esse modelo depende do SGBD que está sendo usado. Nesse modelo, são detalhados os componentes da estrutura física do banco, como tabelas, campos, tipos de valores, índices. É o momento em que o banco de dados em si finalmente pode ser criado, por isso se chama modelo físico.



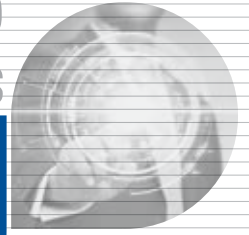
#### RECAPITULANDO

Neste capítulo, você conheceu o mundo fantástico dos bancos de dados. Por meio destes conhecimentos, você pode iniciar a modelagem de sistemas de qualquer área. Você iniciou conhecendo como um banco de dados funciona, como eles surgiram, e por que são tão importantes para nossa sociedade. Após, reconheceu como a modelagem de dados se divide em 3 esquemas, e se aprofundou no modelo conceitual conhecendo seus elementos. Você continuará evoluindo e aprendendo como trabalhar com esta ferramenta incrível que é o banco de dados.



# Construção e Manutenção de Sistema de Gerenciamento de Banco de Dados

## 3



Neste capítulo, você terá oportunidade de conhecer na prática, utilizando ferramentas computacionais adequadas as etapas para a criação de um banco de dados desde o seu modelo conceitual, até a criação física, manipulação e manutenção do banco de dados já materializado. Sendo assim, ao final deste capítulo você terá subsídios para:

- a) compreender a criação do modelo conceitual no brModelo;
- b) conhecer a conversão do modelo conceitual para o lógico;
- c) identificar os sistemas de gerenciamento de banco de dados;
- d) instalar e manipular banco de dados.

Agora inicie este capítulo conhecendo como criar o modelo conceitual no brModelo.

### 3.1 CRIAÇÃO DO MODELO CONCEITUAL NO BRMODELO

Para criação do modelo conceitual, a ferramenta sugerida é o brModelo. Essa ferramenta foi desenvolvida por Carlos Henrique Cândido, sob a orientação do Prof. Dr. Ronaldo dos Santos Mello, da Universidade Federal de Santa Catarina, como trabalho de conclusão do curso de pós-graduação em Banco de Dados. É uma ferramenta *freeware* voltada para ensino de modelagem em banco de dados relacional com base na metodologia defendida pelo Professor Carlos Heuser em sua obra chamada “Projeto de Banco de Dados”.

A ferramenta funciona na plataforma Windows e o download da ferramenta pode ser realizado em vários sites, basta digitar no buscador de sua preferência “download brModelo”. Não há necessidade de realizar instalação, basta executar. Em sua primeira execução, o software solicitará autorização para associar a extensão brM.

A seguir, você pode conhecer a tela principal da ferramenta brModelo.

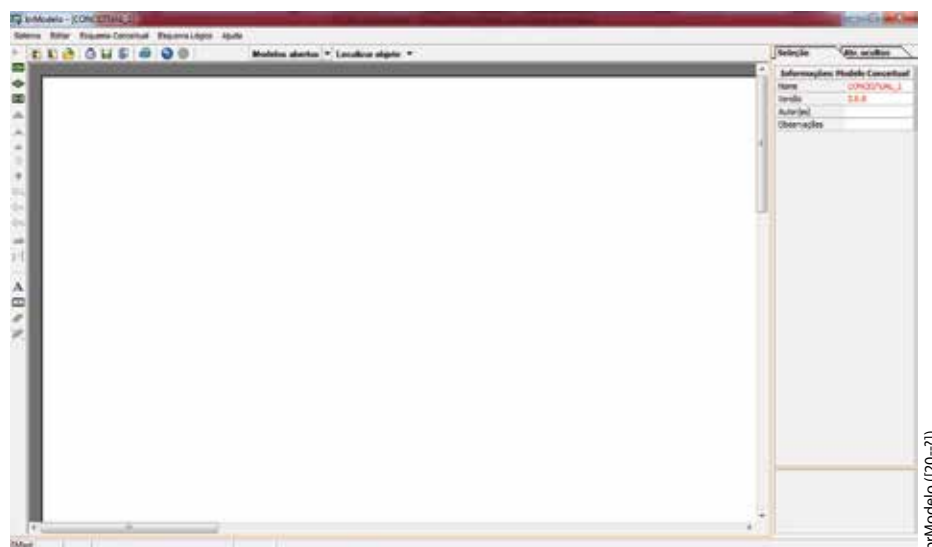


Figura 21 - Tela Principal do brModelo  
Fonte: Do autor (2019)

Para explorar as funcionalidades da ferramenta, você irá trabalhar a criação de um modelo conceitual baseado numa necessidade de desenvolvimento de um sistema. O problema proposto é de um sistema acadêmico de uma Universidade. O sistema precisa controlar os professores, os alunos, os cursos, as disciplinas, as turmas, as avaliações. Enfim, informações que cercam o registro acadêmico do aluno de uma instituição de ensino superior.

Como você estudou anteriormente, neste momento, é importante realizar a abstração de dados e a delimitação do minimundo. Para tanto, nada melhor que desenvolver o modelo conceitual para esse problema.

Na figura a seguir, você pode verificar como incluir uma entidade no seu modelo conceitual por meio da ferramenta brModelo.





Figura 22 - Criação de entidade no brModelo  
Fonte: Do autor (2019)

Ao clicar no retângulo da entidade, é possível alterar seu nome e descrição.

Na figura a seguir, você pode identificar as opções de criação de atributos disponibilizadas pelo brModelo.

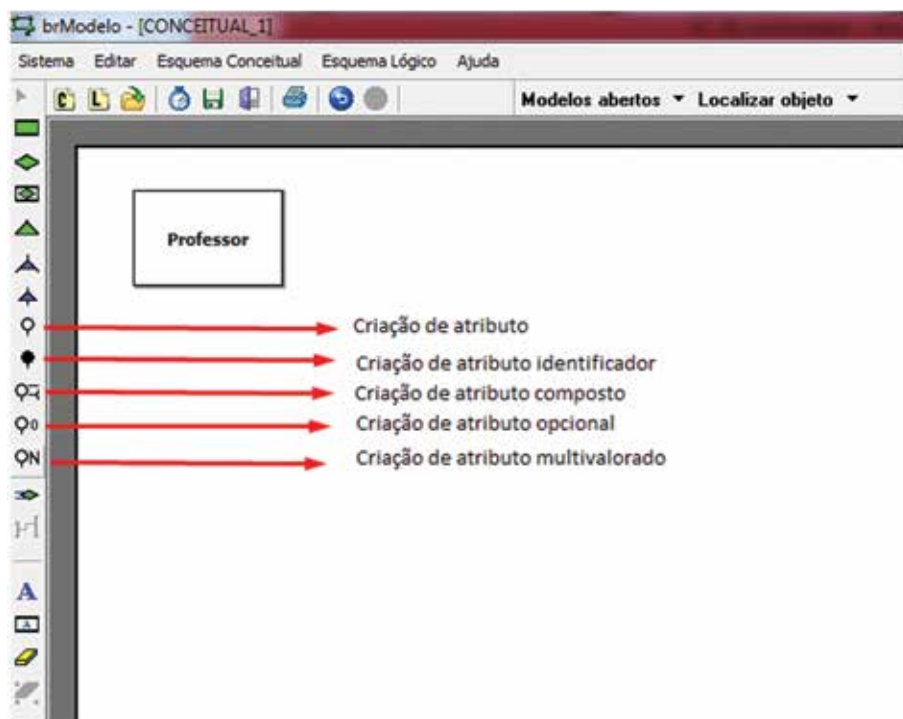


Figura 23 - Opções para criação de atributos no brModelo  
Fonte: Do autor (2019)

Após a criação do atributo, ao clicar sob o nome dele, é possível alterar suas propriedades. A principal propriedade a ser alterada é o Nome. Uma outra informação essencial é em relação a obrigatoriedade do campo, que pode ser configurada através da propriedade “Opcional” que pode ser informada como “Sim” ou “Não”. A figura a seguir demonstra a alteração de propriedades de um atributo no brModelo.

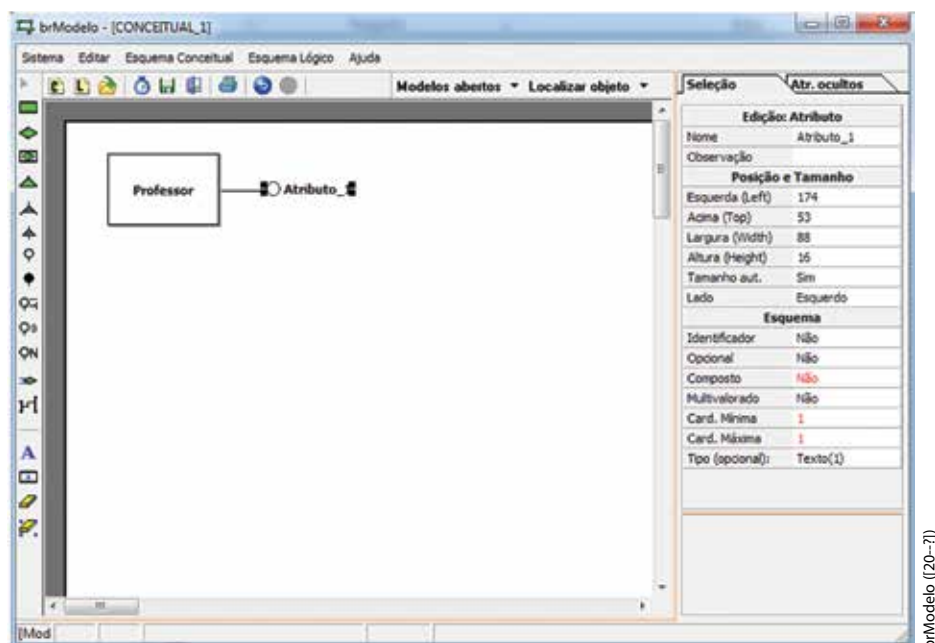


Figura 24 - Alteração de atributo no brModelo  
Fonte: Do autor (2019)

Dentro da modelagem de dados um dos conceitos essenciais é o da relação. A figura a seguir apresenta o ícone para criação da relação.

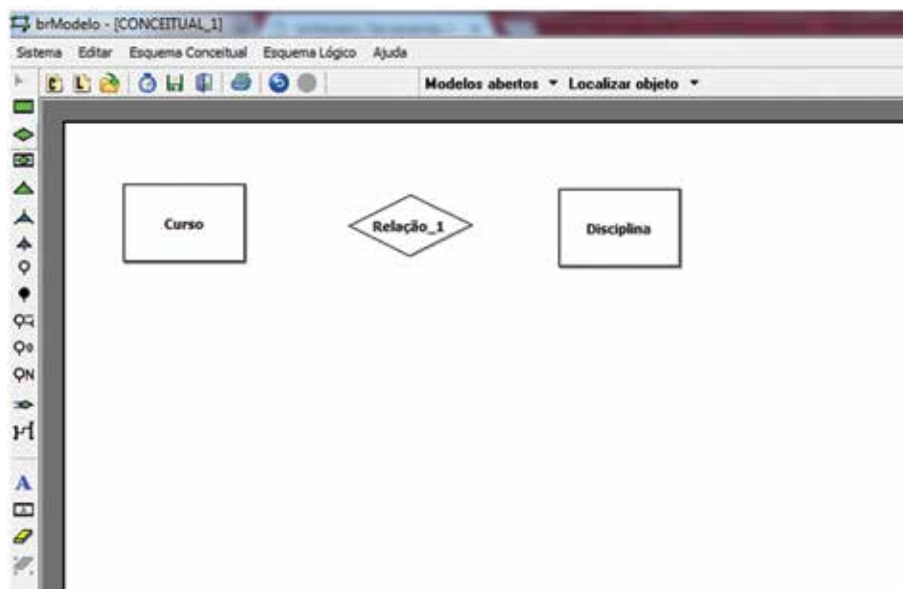


Figura 25 - Criação de relação no brModelo  
Fonte: Do autor (2019)

Em um modelo conceitual, as relações devem ser nomeadas, para isso, no brModelo, basta clicar no losango que representa a relação e editar a propriedade nome. Lembre-se de sempre utilizar verbos que façam sentido. Para ligar uma relação às entidades que a compõem, você deve utilizar a opção Ligar Objetos, conforme destacado na figura a seguir:

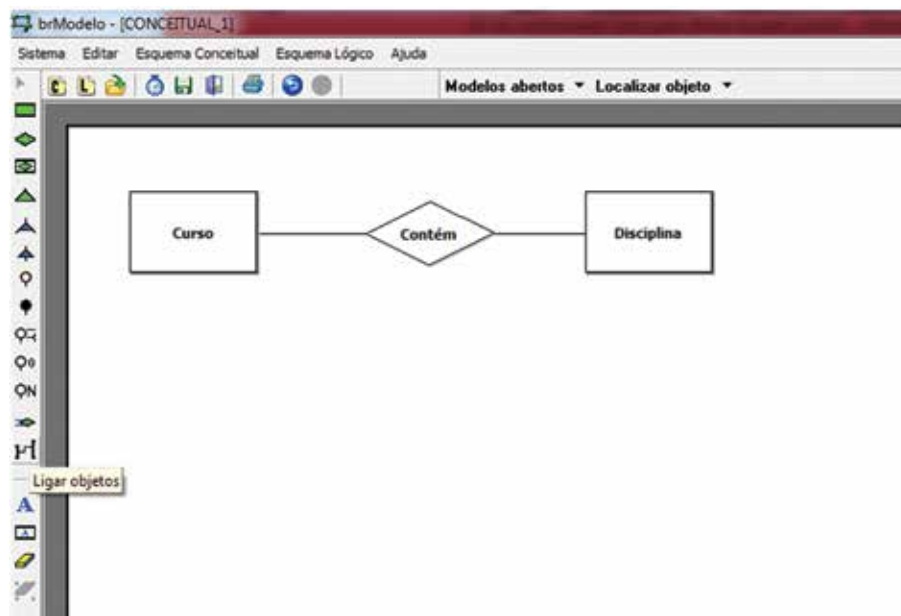


Figura 26 - Ligando objetos no brModelo  
Fonte: Do autor (2019)

brModelo ([20--?])

Outra propriedade essencial a ser definida de um relacionamento é a cardinalidade. Para fazer isso no brModelo, você deve clicar na linha que liga a relação à tabela e definir a propriedade de cardinalidade, conforme demonstrado na figura abaixo.

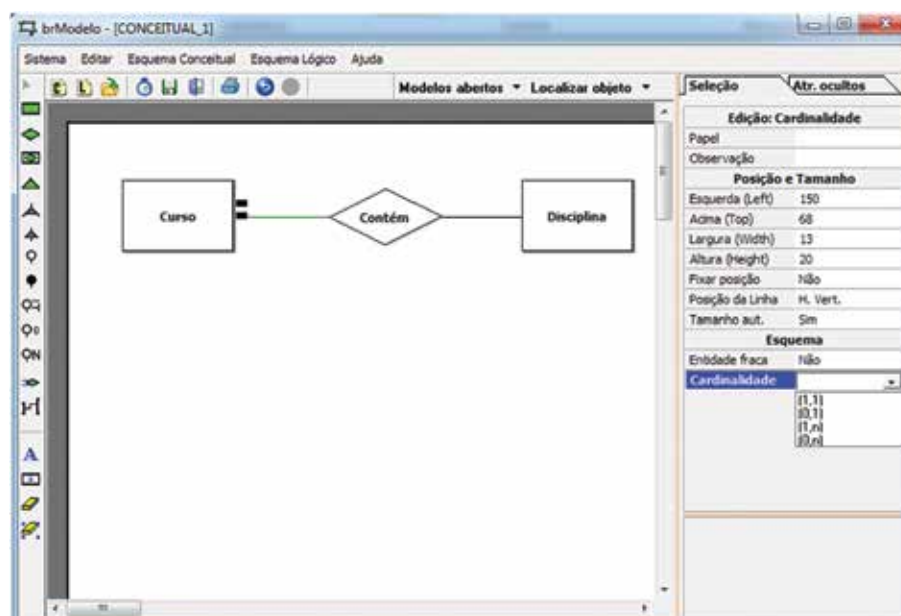


Figura 27 - Ligando objetos no brModelo  
Fonte: Do autor (2019)

brModelo ([20--?])

É importante lembrar que a cardinalidade deve ser definida em ambos os lados da relação, e que a leitura da cardinalidade é feita sempre do lado oposto. Na figura a seguir, a leitura correta é: um curso contém uma ou muitas disciplinas e uma disciplina está contida em um ou muitos cursos.

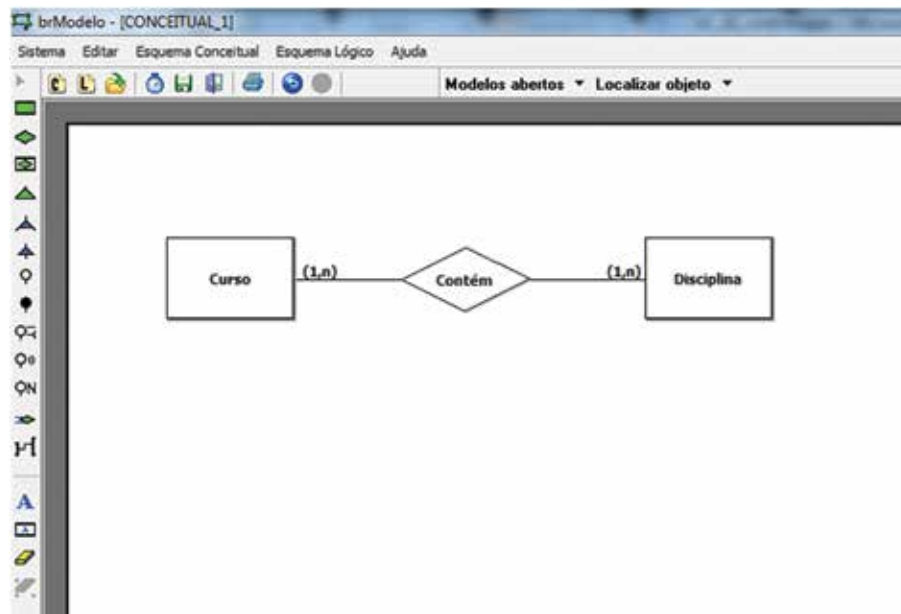


Figura 28 - Cardinalidade do modelo conceitual no brModelo  
Fonte: Do autor (2019)

Agora, munido das principais funcionalidades que o brModelo proporciona para criação do modelo conceitual, mãos à obra.

Crie o seu modelo conceitual do problema proposto. Não se esqueça de primeiramente pensar nas entidades e suas relações.

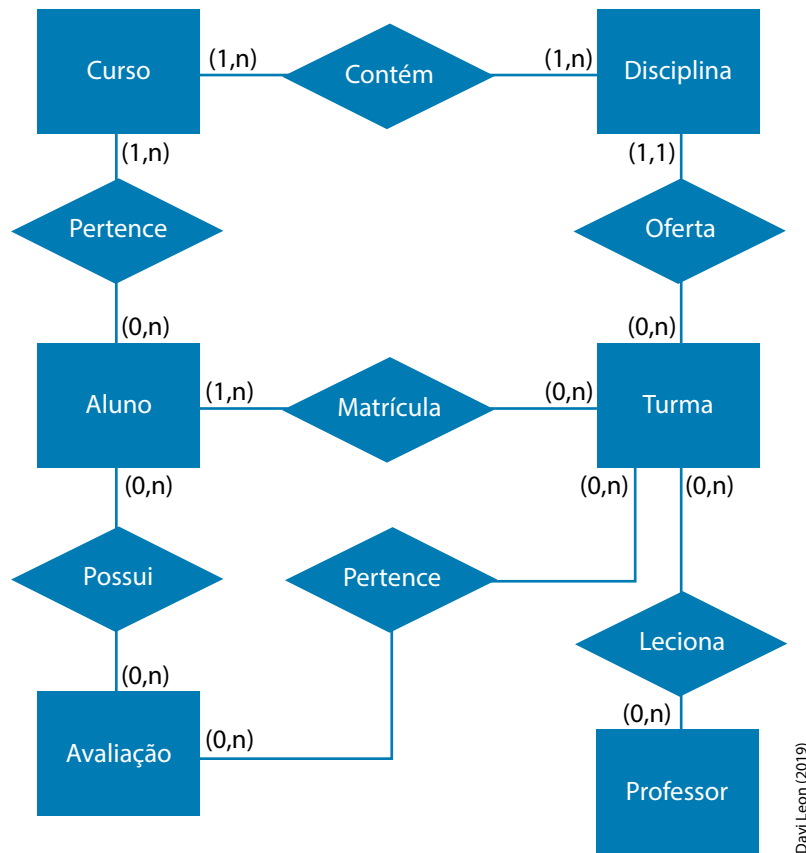


Figura 29 - Modelo Conceitual do problema proposto sem atributos  
Fonte: Do autor (2019)

Davi Leon (2019)

Depois, você precisa detalhar os atributos de cada entidade/relação, sem esquecer-se de definir os atributos identificadores e das cardinalidades dos relacionamentos.

Após a conclusão do seu trabalho, o resultado deverá ser próximo da figura abaixo, lembrando que uma modelagem de dados pode ser diferente de outra e ambas atenderem o problema proposto, o que importa é seguir boas práticas de modelagem de dados e ser uma solução que atenda o problema proposto em todos os seus aspectos. Nessa resolução, foram definidos atributos básicos para cada entidade; num processo real, será necessário identificar junto aos usuários chave do sistema quais são as informações que precisam ser mantidas pelo sistema, tornando-o o mais completo possível.

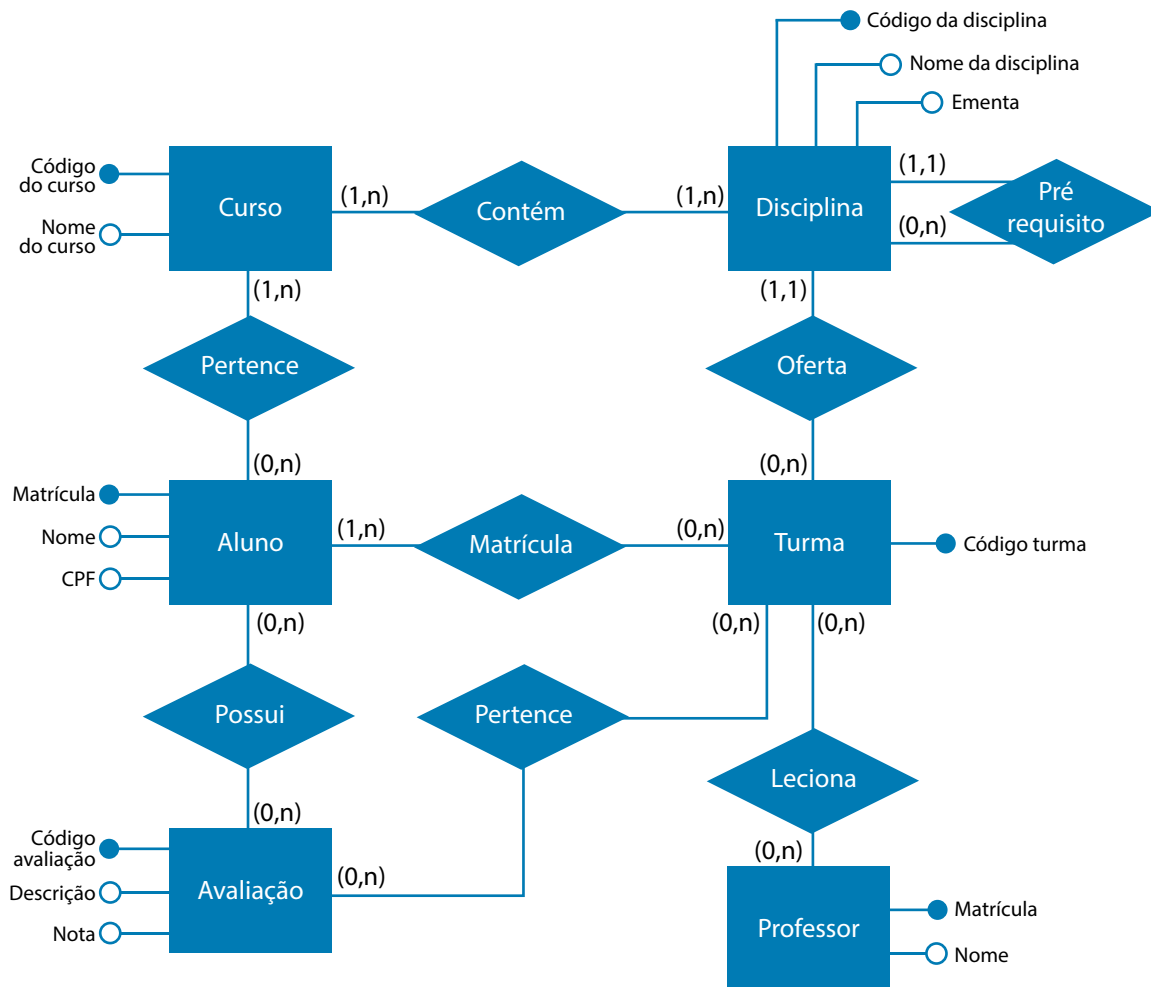


Figura 30 - Modelo Conceitual do problema proposto com atributos  
Fonte: Do autor (2019)

Davi Leon (2019)

Agora que você concluiu o modelo conceitual, você aprenderá a como convertê-lo para o modelo lógico.

### 3.1.1 CRIAÇÃO DO MODELO LÓGICO

Para criar um novo modelo lógico no MySQL Workbench, você deve acessar a opção File >> New Model, conforme representado na figura a seguir:

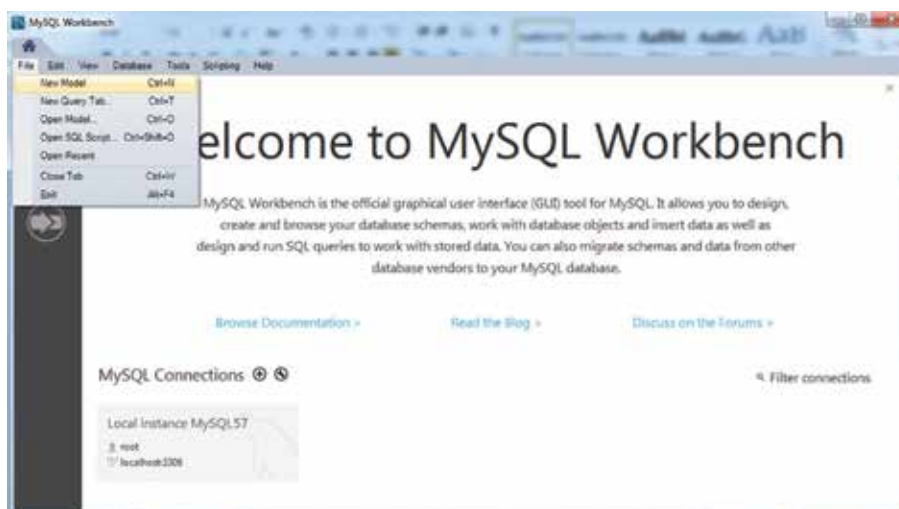


Figura 31 - Tela Principal do MySQL Workbench

MySQL Workbench (2019)

Na tela que foi aberta, na seção Model Overview, acessar a opção Add Diagram, conforme destacado na figura a seguir.

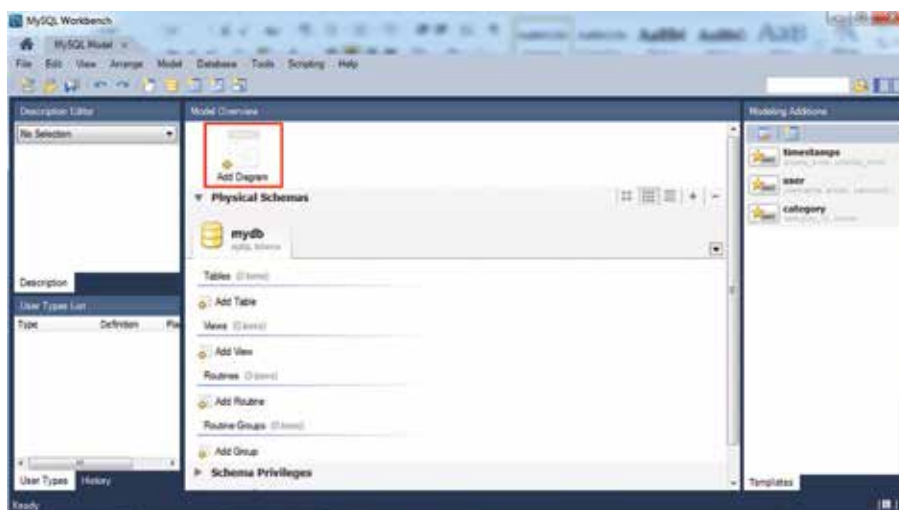


Figura 32 - Seção Model Overview do MySQL Workbench

MySQL Workbench (2019)

Este é o momento para analisar como criar o modelo lógico baseado no modelo conceitual. As entidades virarão tabelas, os atributos: campos, as relações: chaves estrangeiras. Lembrando que nem sempre estas relações serão de 1 para 1. Uma entidade no modelo conceitual pode se desdobrar em várias tabelas no modelo lógico. Até mesmo uma relação no modelo conceitual pode se tornar uma entidade no modelo lógico (cardinalidade muitos para muitos).

É essencial que esta análise seja feita com toda atenção e cuidado. O modelo lógico refletirá exatamente o seu banco de dados físico no futuro.

Para criar uma nova tabela, você deve acessar a opção Place a New Table

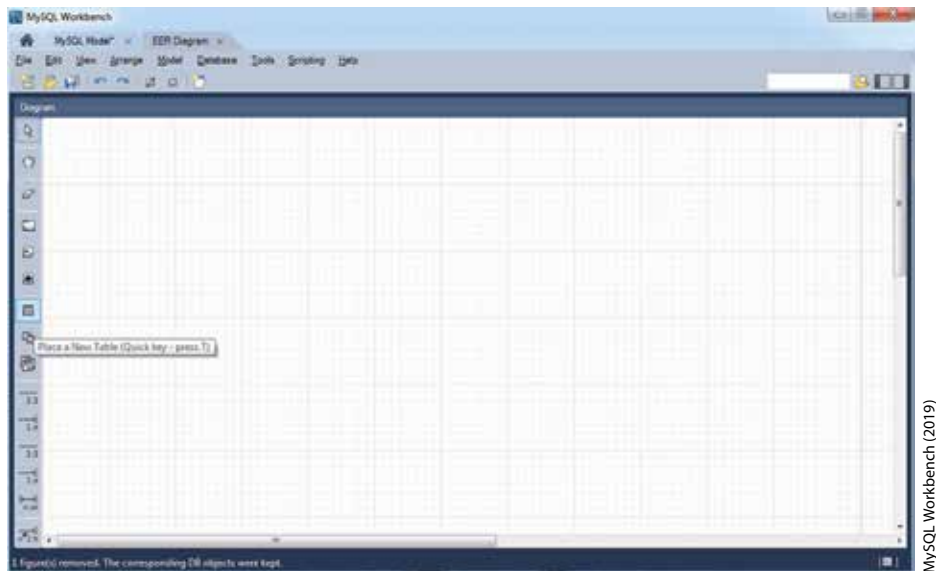


Figura 33 - Acesso a criação de tabela no MySQL Workbench

Para atribuir as propriedades de cada tabela criada, você deve clicar nela e na seção que é aberta informar o nome da tabela e suas colunas. Em cada coluna, você deve informar suas propriedades específicas:

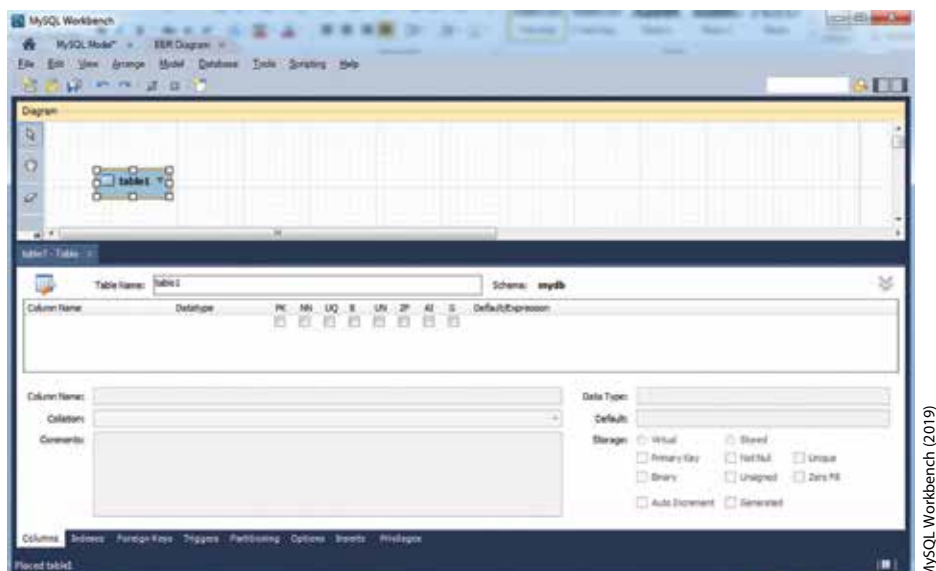


Figura 34 - Detalhamento de colunas em nova tabela no MySQL Workbench



As propriedades específicas de cada coluna a serem definidas são: Nome da Coluna, Tipo de Dado, indicação se a coluna é chave primária ou não, se a coluna é obrigatória ou não, se a coluna é um índice único ou não, se a coluna receberá auto incremento ou não. É essencial realizar estas definições de forma adequada, pois isso vai determinar como o banco de dados que você está criando funcionará na prática.

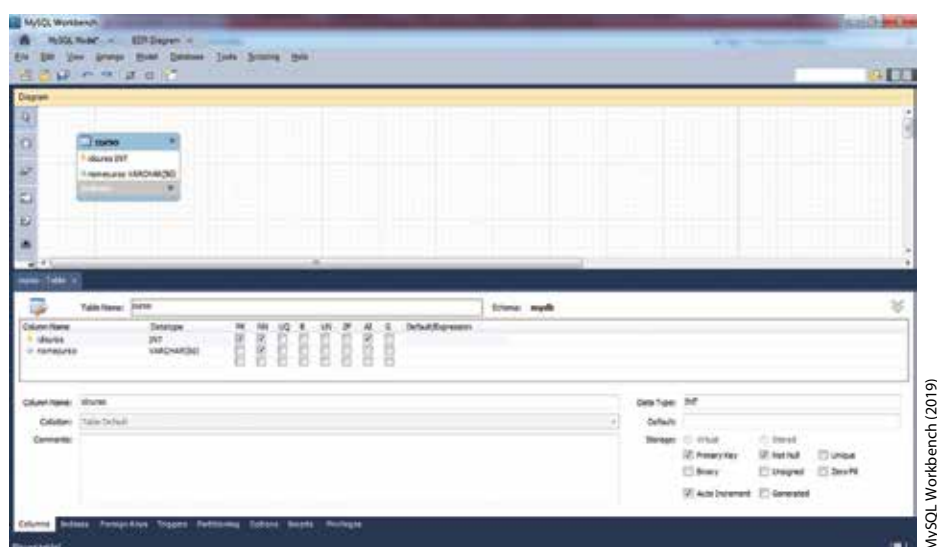


Figura 35 - Detalhamento de colunas na tabela curso no MySQL Workbench

## Tipos de dados

Você precisa conhecer mais sobre tipo de dados para modelar seu banco da melhor forma. Tipo de dado é uma das mais elementares restrições que existem em um banco de dados. O tipo de dado, que é chamado também de domínio do atributo, significa o que será permitido informar naquela coluna. Os tipos de dados mais utilizados são: numéricos, alfanuméricos e de datas. Dentro de cada uma dessas grandes divisões existem subtipos específicos. Quanto melhor selecionado o tipo de dado de uma coluna, melhor este banco será funcionar e será compreendido. Se você verifica que um campo é de um determinado tipo, já consegue compreender que tipo de informação será armazenado nele.

A obrigação de criar restrições de tipos de dados vem da necessidade de se utilizar estas informações armazenadas. Imagine um atributo que armazene a data de nascimento da pessoa. Se ele for um campo livre, alguns usuários informarão no formato adequado: 01/02/1970. Outros poderão informar: Primeiro de Fevereiro de 1970. Ainda outros poderão utilizar: 01 de Fev. de 70. Em um universo de milhares de dados, como juntar estas informações posteriormente? Imagine a necessidade de emitir um relatório com todas as pessoas com mais de 40 anos. Como realizar o filtro, considerando que os dados não estão no mesmo formato?

Análise mais este exemplo: uma tabela que armazena produtos, e contém o valor de venda do produto. Imagine que o valor de venda não tem formato definido. Alguns usuários informam no formato numérico: 90,00. Outro em texto: Noventa. No final do mês, o gerente quer um relatório de quanto foi vendido no período. Não será possível encontrar um meio automatizado de interpretar estes dados despadronizados. Por isso a necessidade da definição dos tipos de dados em cada coluna.

A seguir, os principais tipos de dados a serem utilizados em nas tabelas:

TIPO DE DADO	APLICAÇÃO
TINYINT	Um inteiro muito pequeno. De -128 a 127
SMALLINT	Um inteiro pequeno. De -32768 a 32767
INTEGER ou INT	Um inteiro de tamanho normal. De -2147483648 a 2147483647
DECIMAL(N, P)	Um número decimal, onde N é número de dígitos e P o número de casas após a vírgula.
DATE	Suporta uma data. No formato YYYY-MM-DD
DATETIME	Uma combinação de data e hora. No formato: YYYY-MM-DD HH:MM:SS
TIME	Suporta uma hora. No formato HH:MM:SS
CHAR(T)	Cadeia de caracteres. Em T deve ser determinado o tamanho da coluna.
VARCHAR(T)	Cadeia de caracteres. Em T deve ser determinado o tamanho da coluna.

Quadro 2 - Principais tipos de dados no MySQL  
Fonte: Do autor (2019)

A diferença dos tipos char e varchar é em relação a alocação de espaço. Quando um atributo é do tipo char, o banco já reserva o espaço para armazenar todo o tamanho definido para o campo. No caso de um atributo varchar, o SGBD fará esta alocação dinamicamente, de acordo com o tamanho do dado armazenado. Dessa forma, você pode pensar que é mais vantajoso sempre utilizar o varchar, não é mesmo?

A verdade é que cada um tem sua aplicação de uso. Para as colunas em que o tamanho de todos os registros será o mesmo, é indicado que se utilizar o char (alocação de espaço fixa). Dessa forma, o banco não precisará a cada novo registro avaliar o tamanho informado no campo e realizar a alocação de espaço, diminuindo o processamento na inserção. Um exemplo de um campo texto que tem tamanho fixo é o CPF. Apesar de ser um número é considerado como texto devido a possibilidade de existência de zeros a esquerda. Como todos os CPFs possuem 11(once) dígitos (9 (nove) do número e 2 (dois) do dígito verificador), a indicação é da utilização do tipo CHAR com tamanho 11.

Já no caso de uma coluna que armazene o nome de uma rua, não é possível determinar que todos os registros terão o mesmo tamanho. Tem ruas com nome curto, e tem ruas com nome muito longo. A indicação neste caso é da utilização do tipo VARCHAR com tamanho adequado. No caso do nome de rua, o tamanho 150 é mais do que suficiente de acordo com os nomes de ruas mais comuns utilizados no Brasil.

## Chaves estrangeiras

Agora que você aprendeu como inserir uma tabela em seu modelo lógico, vamos aprender a como criar os relacionamentos. Para criar um relacionamento no modelo lógico, precisa ser analisado a direção e a cardinalidade do relacionamento. Vamos usar como exemplo o relacionamento entre Disciplina e Curso. No modelo lógico, foi definido que um curso contém uma ou mais disciplinas e que uma disciplina está contida em um ou muitos cursos. Esse tipo de relacionamento é denominado muitos para muitos. Veja a seguir como criá-lo:

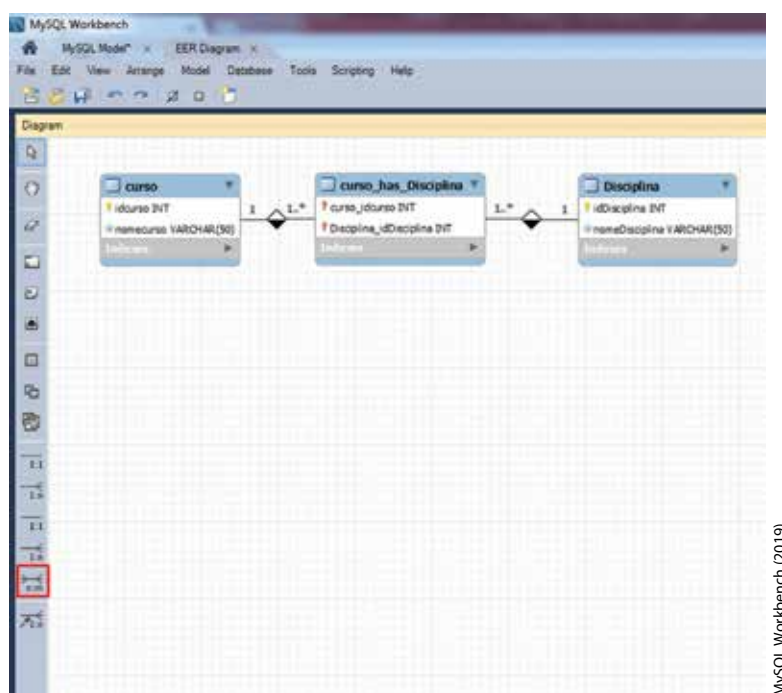





Figura 36 - Criação de um relacionamento n:m no MySQL Workbench

Para criar um relacionamento muitos para muitos, deve-se clicar no ícone , depois em uma das tabelas que compõe o relacionamento, e depois na outra. Neste caso, é criada, automaticamente, uma terceira tabela, pois este relacionamento indica que um curso pode ser vinculado a mais de uma disciplina e uma disciplina vinculada a mais de um curso.


Dando continuidade a criação do modelo lógico, pode-se verificar que a entidade Disciplina, tem um relacionamento recursivo, em que uma disciplina pode ser  pré-requisito de outra. Neste caso, deve-se clicar no ícone , e na tabela disciplina duas vezes, indicando que a origem e o destino do relacionamento, é ela mesma. Ao realizar esta operação, a ferramenta irá criar automaticamente uma nova coluna para representar o relacionamento. Para ajustar a cardinalidade deste relacionamento, considerando que uma disciplina pode

ou não ter pré-requisito, deve-se alterar a propriedade “Not Null” que indica se ela é obrigatória ou não. Verifique que ao alterar esta propriedade a cardinalidade é automaticamente alterada. Você pode também alterar o nome da coluna para uma nomenclatura que faça mais sentido, por exemplo: idPreRequisito, conforme demonstrado a seguir.



Figura 37 - Criação de um relacionamento recursivo no MySQL Workbench

Fonte: Do autor (2019)

Para criar o relacionamento entre turma e disciplina, basta utilizar a opção  pois uma turma é de uma disciplina e uma disciplina pode ser oferecida em uma ou mais turmas. Veja o resultado a seguir:

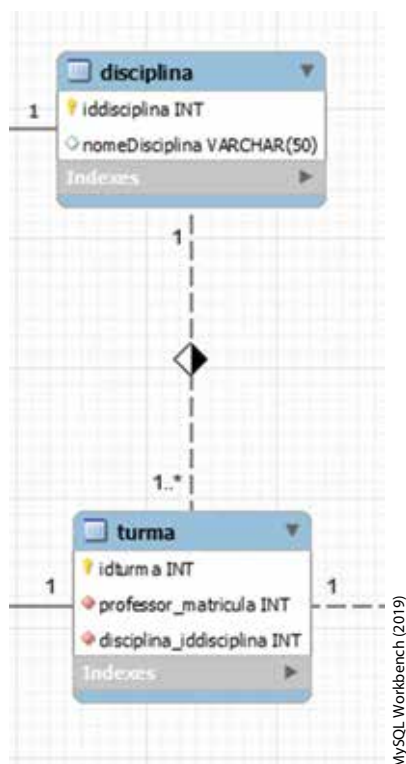


Figura 38 - Criação de um relacionamento entre Disciplina e Turma no MySQL Workbench

Fonte: Do autor (2019)

É importante ressaltar que algumas entidades no modelo conceitual se desdobrarão em mais de uma tabela e que alguns relacionamentos virarão tabelas. Veja o exemplo das entidades aluno, turma e avaliação, que no modelo lógico são três entidades relacionadas entre si. Devido às cardinalidades definidas, no modelo conceitual, existe uma tabela para turma que concentra os dados comuns a toda a turma como disciplina e professor, e uma tabela filha desta, que detalha os alunos da turma. Já para a entidade avaliação, uma tabela com o cabeçalho da avaliação, que contém o código e descrição da avaliação, e uma outra tabela com as notas da avaliação, contendo código da turma e código do aluno, bem como sua nota. Veja a seguir o modelo lógico que demonstra estas definições:

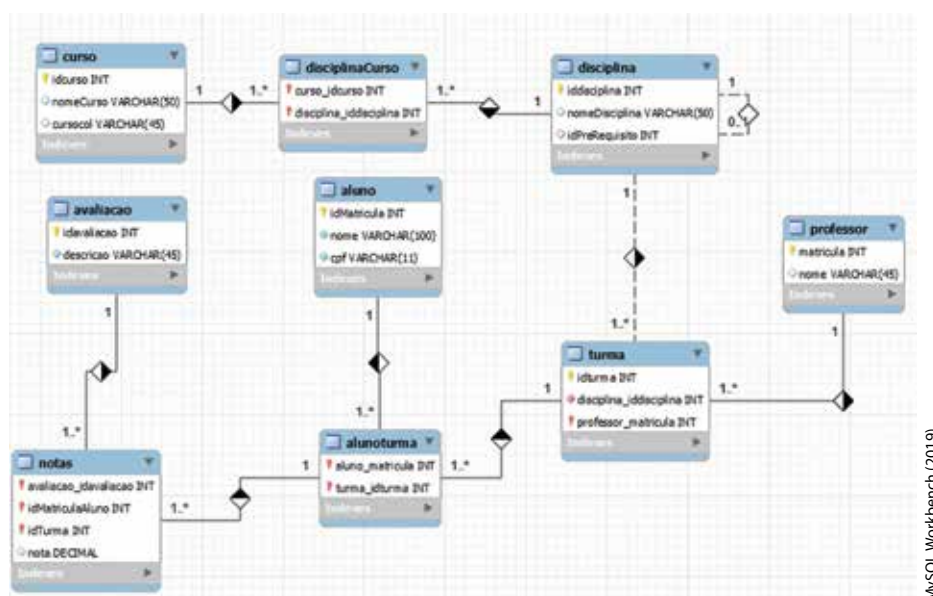


Figura 39 - Modelo Conceitual no MySQL Workbench  
Fonte: Do autor (2019)

### 3.1.2 DEPENDÊNCIA FUNCIONAL

Em uma tabela, uma coluna X depende funcionalmente de uma coluna Y (ou que a coluna X determina a coluna Y) quando, em todas linhas da tabela, para cada valor de X que aparece na tabela, aparece o mesmo valor de Y.

Por exemplo, em uma tabela de clientes, pode se dizer que o atributo CPF do cliente determina funcionalmente o nome do cliente.

É importante o entendimento desse conceito para que o conceito de normalização, que será visto a seguir, seja melhor compreendido.

### 3.1.3 NORMALIZAÇÃO

O processo de normalização visa obter a melhor modelagem de dados possível, baseado nos conceitos de forma normal.

Uma forma normal é uma regra que deve ser correspondida por uma tabela para que ela seja considerada “bem projetada”. Existe uma série de formas normais, começando da primeira, com regras cada vez mais rígidas conforme aumenta o grau de normalização. Aqui você conhecerá da 1FN até 3FN. Acompanhe!

A 1FN determina que uma tabela deve conter apenas atributo atômicos. Portanto, atributos compostos e multivalorados precisam ser eliminados. Eliminar não significa removê-los do seu modelo, e sim transformá-los. A normalização em si é este processo de transformação.

Pense no exemplo de uma tabela que tem um atributo composto de endereço (rua, número, bairro, cidade e estado). O que se espera ao executar a 1FN nesta tabela, é que cada uma dessas partes que compunham o atributo composto virem atributos independentes.

Já no caso de atributos multivalorados, pode-se pensar no exemplo do atributo e-mail. Ao realizar a modelagem conceitual foi determinado que para cada registro da tabela pudesse ter de 1 até 3 e-mails. Para atender a 1FN é necessário criar uma tabela para armazenar estes e-mails, vinculada a primeira, para que permita que para cada registro se tenha mais de um e-mail. Uma opção que pode ter sido pensada por você é a da criação de três atributos iguais (email01, email02 e email03). Essa não é uma opção indicada, uma vez que, se a necessidade mudar de 3 para 5 e-mails, será necessária uma alteração na base de dados, o banco de dados relacional resolve isso muito mais habilmente com uma nova tabela relacionada.

Para trabalhar a 2FN em uma tabela, o primeiro requisito é que ela já esteja com a 1FN aplicada. A 2FN define que os atributos comuns, os que não são chave primária, devem depender funcionalmente unicamente da chave primária da tabela. Assim como as colunas da tabela que não são dependentes funcionais dessa chave primária devem ser removidas da tabela que está sendo normalizada e deve-se criar uma nova tabela utilizando esses dados. Em um exemplo de uma tabela de veículos, além do código do veículo, da placa, número do chassi, número do renavam, valor de compra, valor de venda e lucro, contém também os atributos código do modelo e nome do modelo. Pode-se verificar que placa, número do chassi, número do renavam dependem funcionalmente do código do veículo, mas que nome do modelo depende funcionalmente do código do modelo. Nesse caso, deve-se criar uma entidade específica para os modelos, e na tabela veículo, determinar uma chave estrangeira para a tabela criada.

A 3FN também exige que as etapas anteriores estejam concluídas. Nesse caso, a 1FN e 2FN. A 3FN determina que todos os atributos sejam dependentes diretos da chave primária. Seguindo utilizando o exemplo da tabela de veículos, analise o atributo lucro. O lucro é dependente de outras duas colunas que não são chave primária. Valor de Compra e Valor de Venda: Para se chegar ao lucro, deve-se diminuir do valor de venda o valor de compra. Isso significa que este valor não precisa ser armazenado.

Toda modelagem deve passar por uma série de etapas de análise e revisão. Quando chegar no fim destas etapas, o seu modelo lógico está preparado para se tornar um modelo físico e seu banco de dados ser criado.

Na sequência você continua ampliando o seu conhecimento por meio de novo assunto: sistemas de gerenciamento de banco de dados. Acompanhe!

## 3.2 SISTEMAS DE GERENCIAMENTO DE BANCO DE DADOS

Um Sistema de Gerenciamento de Banco de Dados é um conjunto de aplicativos, programas e ferramentas que gerenciam a estrutura de um banco de dados e controlam o acesso aos dados armazenados. O SGBD é uma ponte entre o usuário e o banco de dados. A estrutura de um SGDB é armazenada como um conjunto de arquivos e o único modo de acessar os dados nesses arquivos são por meio do SGBD.

### 3.2.1 TIPOS

Diversos SGDBs foram construídos, ao transcorrer do tempo. Desde as primeiras versões do IBM DB2 e do Oracle, muita evolução ocorreu. A seguir você verifica uma lista com os bancos de dados e fabricantes mais utilizados:

- a) Oracle;
- b) MySQL;
- c) Microsoft SQL Server;
- d) Postgre SQL;
- e) IBM DB2.

Para escolher entre um SGDB e outro, é necessário levar em conta uma série de fatores, como, por exemplo: orçamento disponível, quantidade de usuários concorrentes, quantidade de dados armazenados, nível de segurança necessário, extensão territorial do acesso.

A seguir, você conhecerá alguns aspectos relativos a instalação de um SGDB.

### 3.2.2 INSTALAÇÃO

Bancos de dados são utilizados pelos mais diferentes tipos de aplicação. Desde softwares corporativos a nível mundial até pequenos aplicativos móveis. Da mesma forma, a questão de instalação e configuração depende deste tipo de aspecto. Atualmente, qualquer computador pode receber a instalação de um SGDB. Porém em relação a performance e armazenamento irá depender muito de quantidade de usuários e do tamanho dos dados armazenados.

Pensando um pouco em relação ao hardware, o tripé, processamento, armazenamento e memória é que garantirá um SGDB íntegro e rápido.

Em aplicações comerciais, é essencial que haja um servidor dedicado para o banco de dados. Isto permite que todos os recursos computacionais disponíveis estejam dedicados a esta tarefa.

Atualmente, estes servidores podem estar alocados na nuvem, ou seja, utilizando um serviço disponibilizado por uma empresa de hospedagem. Essa alternativa permite inclusive que sejam adicionados servidores adicionais de acordo com o aumento de demanda.

Isso significa que, se o seu banco de dados está instalado em um serviço na nuvem, e você tem uma demanda sazonal de performance, poderá contratar uma adição de infraestrutura de servidores para aquele período.

A seguir você conhecerá como ocorre a manipulação de banco de dados. Vamos lá!

### 3.3 MANIPULAÇÃO DE BANCO DE DADOS

Através da manipulação de banco de dados é que alcança o objetivo final de um banco de dados: que é criar a estrutura de um banco de dados e mantê-la; inserir, alterar, consultar e excluir os dados armazenados obtendo, dessa forma, o pleno uso de uma estrutura de SGDB.

A manipulação de um banco de dados pode ocorrer para realização de manutenções estruturais ou de dados, ou através de aplicações, sistemas que o utilizem como meio de armazenar as suas informações, ou seja, a atividade-fim do banco de dados, armazenar informações relativas a uma determinada área de negócio.

#### 3.3.1 FERRAMENTAS

Todos SGDBs disponibilizam ferramentas para acesso a manipulação de banco de dados. Existem também ferramentas que acessam diversos bancos de dados através de drivers de conexão.

Para acessar o banco de dados Oracle, existem as ferramentas Oracle SQL Developer da própria Oracle e o mundialmente conhecido PL/SQL Developer da All Round Automations.

Já para o Microsoft SQL Server, existem algumas opções do próprio fabricante, como o SQL Server Management Studio e o SQL Server Data Tools.

O MySQL, conforme visto anteriormente, concentra na ferramenta Workbench também a manipulação do banco físico.

Por fim, é importante comentar sobre o HeidiSQL que é um programa *freeware* que permite a conexão nos bancos MySQL, Microsoft SQL Server e PostgreSQL.

Já as aplicações, utilizam conexão através de drivers. Drivers de conexão de banco de dados são *middlewares*, ou seja, são ferramentas que permitem que uma determinada linguagem de programação (Java, C#, PHP) se conecte a um banco de dados.



**CURIOSIDADES**

O MySQL foi criado por uma parceria de suecos e um finlandês. A MySQL AB que era a empresa deles desenvolvedora do software foi adquirida em 2008 pela Sun Microsystems pelo valor de 1 bilhão. Em 2009, a Sun Microsystems foi adquirida pela Oracle o gigante dos bancos de dados. Neste momento, os usuários do MySQL ficaram apreensivos, pela possibilidade de que este o MySQL fosse descontinuado em detrimento do Oracle. O que aconteceu foi o contrário, a Oracle investiu no produto, tornando-o mais completo e robusto se tornando uma linha paralela ao Oracle atingindo um público de usuários e aplicações específico.

Na sequência será abordado um novo assunto. Prepare-se para conhecer a linguagem SQL!

### 3.3.2 LINGUAGEM SQL

As ferramentas de modelagem de dados disponibilizam funcionalidades para geração automatizada dos scripts para criação do banco de dados na linguagem A linguagem SQL foi criada nos laboratórios de pesquisa da IBM na década de 1970, e é um padrão mundial para bancos de dados relacionais desde a década de 1980. SQL é uma linguagem para criação de bancos de dados, manipulação (inserção, atualização, exclusão) e seleção de dados. Você precisa aprender SQL, pois é um padrão para bancos de dados relacionais há mais de 30 anos e sem previsão de grandes mudanças. Além do mais, conhecer SQL é um requisito básico na maioria das posições de emprego na área de sistemas de informação.

Os bancos de dados são peças centrais da maioria dos sistemas de informação. Por este motivo, o desenvolvimento de scripts na linguagem SQL bem estruturados, seguindo as melhores práticas é essencial para o sucesso do software como um todo.

As principais características da linguagem SQL você acompanha na sequência.

- a) Independência do fabricante: para os mais diversos fabricantes de banco de dados a linguagem é a mesma: SQL. Isso se deve a uma unificação de padrões que nasceu pelas mãos do American National Standards Institute (ANSI), que é revisada periodicamente. Os fabricantes adotam este padrão facilitando assim o aprendizado e uso da linguagem em suas aplicações.
- b) Possibilidade de extensões: apesar do padrão ANSI ser amplamente utilizado e incentivado, cada fabricante de banco de dados pode desenvolver extensões próprias, que são comandos e estruturas desenvolvidas especificamente pelo fabricante. Se um tipo de extensão for muito interessante e servir para os demais, é provável que na próxima discussão sobre o padrão ANSI esta extensão possa ser incluída no padrão e difundida pelos demais fabricantes de banco de dados.
- c) Linguagem SQL é case INSENSITIVE, ou seja, NÃO faz diferenciação entre letras maiúsculas e minúsculas. Isso significa que isto não precisa ser uma preocupação ao desenvolver os códigos em linguagem SQL. Obviamente por questões estéticas, não é adequado utilizar a capitalização todas em maiúsculas ou todas em minúsculas. A sugestão é que se utilize CamelCase que é a denominação em inglês para a técnica de escrever palavras compostas ou frases, onde cada palavra é iniciada com sua primeira letra em maiúscula e as demais em minúscula e unidas sem espaços. Exemplos:CodigoBarras, TelefoneCliente, NomeProduto.

- d) SQL utiliza um inglês de alto nível de fácil entendimento. Palavras-chaves são utilizadas e ao realizar a tradução literal para o português é possível obter um aprendizado rápido.
- e) Uma linguagem em que se determina o resultado esperado, e não quais são os caminhos que se deve percorrer para se chegar até ele. Isso significa que com pouca programação, resolvem-se problemas complexos. Esta característica só é possível devido aos interpretadores de comandos programados no SGDB, que são desenhados para resolver as situações propostas.

O MySQL Workbench por ser uma ferramenta completa no que tange a administração de banco de dados, proporciona que o usuário interaja com o banco de dados através da execução de scripts em SQL.

O MySQL Workbench, também tem disponível a funcionalidade de criação de um banco de dados físico através de seu modelo lógico. O nome desta funcionalidade é Forward Engineer to Database e seu acesso se dá através do menu Database.

Na primeira tela, Connection Options, você deverá informar os parâmetros de conexão a sua base de dados.

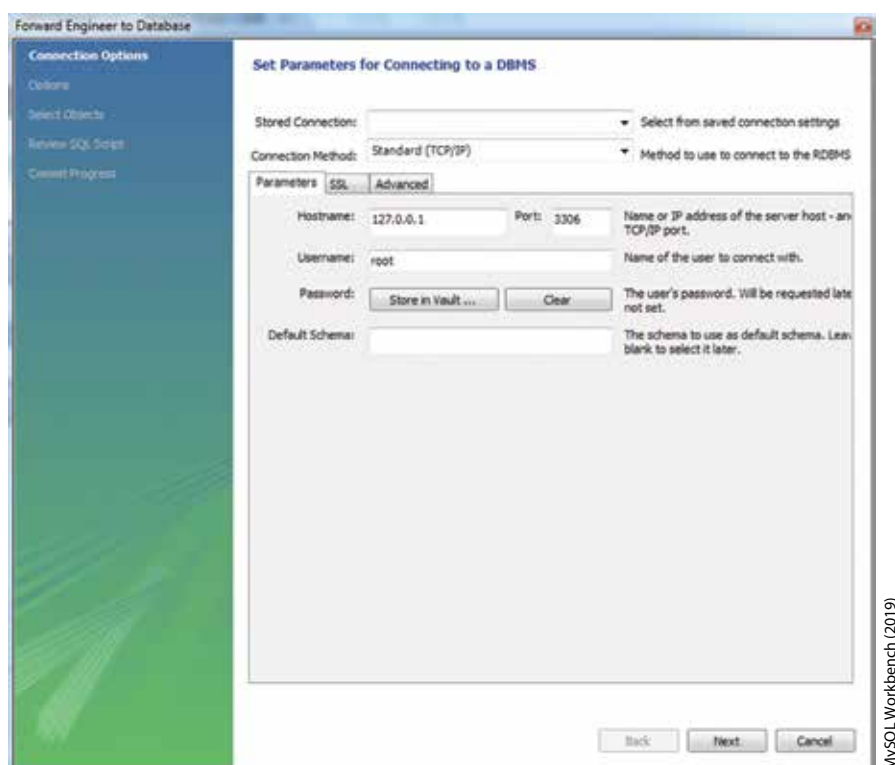


Figura 40 - Connection Options de Forward Enginner no MySQL Workbench

Na sequência, você deverá informar sua senha de acesso ao banco de dados:



Figura 41 - Informação da senha de conexão no MySQL Workbench

Na próxima tela, serão apresentadas as opções de que objetos você deseja exportar para o banco de dados. Se quiser exportar tudo, não é necessário modificar nada nesta funcionalidade.

A última tela apresenta uma prévia do SQL preparado para a criação do banco de dados. Recomenda-se que se utilize a função Save to File para armazenar o script que será executado. Isso auxiliará em futuras manutenções.

A opção Forward Enginner é um facilitador para a criação do script SQL de criação do banco de dados. Portanto, ao finalizar essas etapas, o seu banco de dados estará criado.

### Divisões da Linguagem SQL

A linguagem SQL é dividida em 5 conjuntos de comandos, cada um com um objetivo específico:

- a) DDL (Data Definition Language): tem como objetivo criar e alterar a estrutura dos bancos de dados;
- b) DML (Data Manipulation Language): tem como objetivo criar, alterar e manipular os dados contidos em um banco de dados;
- c) DQL (Data Query Language): é responsável pela consulta (recuperação) de dados inseridos no banco;
- d) DCL (Data Control Language): tem como objetivo controlar o acesso ao banco e aos dados;
- e) DTL (Data Transaction Language): é responsável pelo gerenciamento das transações ocorridas em um banco de dados.

Na sequência, você estudará cada uma destas linguagens e terá a oportunidade de conhecê-las mais detalhadamente e compreender o quanto cada uma completa a outra nas tarefas de manipulação de um banco de dados.

### 3.3.3 DDL – DATA DEFINITION LANGUAGE

A linguagem DDL é a área da linguagem SQL responsável pela criação e manutenção da estrutura dos bancos de dados. Através de DDL você irá criar e alterar tabelas e seus atributos, criar relacionamentos entre os atributos, definir restrições, criar triggers, procedures e views. A linguagem DDL é a que dará vida ao seu banco de dados. Apesar da maioria das ferramentas de modelagem de dados se integrarem diretamente ao SGDB para criação do banco de dados, esta criação é executada através de scripts SQL DDL. É essencial que você conheça estes comandos e consiga utilizá-los sempre que necessário.

#### CREATE DATABASE

O comando create database permite que você crie uma nova instância de campo de dados no seu servidor. Geralmente, é criado um database para cada assunto ou sistema que será trabalhado. A seguir, você pode verificar a sintaxe do comando create database:

```
CREATE DATABASE <nomedatabase>
```

A criação de um database não o seleciona para o uso, para isso você precisa especificar qual database deseja utilizar, o que é feito através do comando use. A seguir, você pode verificar a sintaxe do comando use:

```
USE <nomedatabase>
```

Vamos, agora, verificar um exemplo de criação e seleção de um database para uso no MySQL Workbench. A figura a seguir apresenta o SQL Editor, com os comandos adequados. Para executar qualquer comando no MySQL Workbench, você deve utilizar o botão .

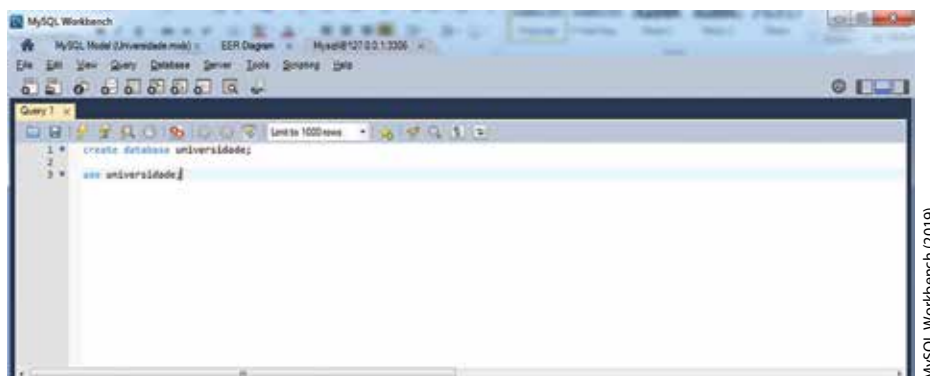


Figura 42 - Criação e Seleção de um Database no MySQL Workbench

A criação de um database você fará uma vez só, mas a seleção para uso deve ser feita a cada vez que se conecta ao banco de dados, ou exista a necessidade de mudar de database após a conexão.

Após a criação e a seleção do banco de dados, você está apto a criar as demais estruturas do banco de dados, as quais estudará a seguir.

## CREATE TABLE

O comando mais usado da linguagem DDL é o CREATE TABLE. Através dele, as tabelas são criadas com suas colunas e toda sua estrutura. A seguir, o detalhamento da sintaxe deste comando:

Ao rodar um script SQL, podem ocorrer erros. Em qualquer comando, um erro de sintaxe, fará com que ele não seja executado e que o SGDB retorne um erro. No caso do comando create table, mesmo com a sintaxe correta, podem ocorrer erros nas seguintes situações:

```
CREATE TABLE <nome da tabela> (
  <nomedacoluna1> <tipo de dado> <tamanho> <obrigatoriedade>;
  <nomedacoluna2> <tipo de dado> <tamanho> <obrigatoriedade>;
  <nomedacolunaN> <tipo de dado> <tamanho> <obrigatoriedade>;
  <nomedabase>;
  restrições de integridade);
```

a) caso já exista uma tabela com o mesmo nome no database selecionado;  
b) caso não exista não tenha nenhum database em uso. Neste caso, deve-se usar o comando USE

Ao criar uma tabela, é preciso definir para cada atributo, seu tipo, seu tamanho (se for o caso), sua obrigatoriedade e as restrições de integridade.

Exemplo de criação de uma tabela:

```
CREATE TABLE produto (
  codProduto integer not null primary key,
  descProduto varchar(100) not null,
  precoVenda decimal(10,2) not null,
  observacao varchar(500) null);
```

Ao rodar um script SQL, podem ocorrer erros. Em qualquer comando, um erro de sintaxe, fará com que ele não seja executado e que o SGDB retorne um erro. No caso do comando create table, mesmo com a sintaxe correta, podem ocorrer erros nas seguintes situações:

- a) caso já exista uma tabela com o mesmo nome no database selecionado;
- b) caso não exista não tenha nenhum database em uso. Neste caso, deve-se usar o comando USE <nomedatabase>;

## ALTER TABLE

Este comando tem como objetivo alterar a estrutura de uma tabela. Pode ser a adição de uma nova coluna, a alteração de uma coluna já existente, ou até mesmo a retirada de uma coluna. Também podem ser

alteradas as restrições de integridade da tabela, como a chave primária e a criação de chaves estrangeiras. A sintaxe básica deste comando é:

```
ALTER TABLE <nometabela>
```

Seguida da modificação que se pretende fazer, por exemplo, se quiser adicionar uma nova coluna em uma tabela já existente, a sintaxe é:

```
ALTER TABLE <nometabela> ADD column <nomecoluna> <tipo tamanho>  
<obrigatoriedade>
```

Se a necessidade for a adição de uma nova chave estrangeira, além da sintaxe básica, deve adicionar a criação da foreign key.

Sintaxe:

```
ALTER TABLE <nometabela> ADD FOREIGN KEY (<nomecolunadestino>)  
REFERENCES <nometabelaorigem> (<colunaorigem>);
```

Exemplo:

```
ALTER TABLE CURSO1 ADD FOREIGN KEY (CURSOPAI) REFEREN-  
CES CURSO1(CDCURSO);
```

A seguir, as principais modificações que podem ser realizadas:

ALTERAÇÃO	SINTAXE
Alterar a definição de uma coluna	ALTER TABLE <nometabela> MODIFY COLUMN <nomecoluna> <tipo tamanho> <obrigatoriedade>
Modificar o nome de uma coluna	ALTER TABLE <nometabela> RENAME COLUMN <nomeantigo> TO <novonome>
Modificar o nome da tabela	ALTER TABLE <nometabela> RENAME TO <novonometabela>;
Eliminar uma coluna	ALTER TABLE <nometabela> DROP COLUMN <nomecoluna>

Quadro 3 - Principais comandos de alteração de uma tabela  
Fonte: Do autor (2019)

## DROP TABLE

Este comando tem como função excluir uma tabela do banco de dados. É importante lembrar que ao excluir uma tabela do banco de dados, você está excluindo sua estrutura e os dados contidos nela. Tabelas mesmo que sem uso geralmente não são excluídas de um banco de dados. A premissa é de que um dia as informações lá contidas possam ser necessárias. É preciso levar em consideração que as tabelas na maioria das vezes armazenam dados estratégicos das organizações, não se pode correr o risco de perdê-las. Portanto, o comando drop table deve ser utilizado somente casos específicos. Um exemplo de situação em que pode ser necessária a utilização de drop table é no caso de uma reestruturação no seu banco de dados. Por alguma necessidade específica, o seu banco de dados será remodelado. Os dados que estão na tabela x, foram para uma nova tabela y, e a tabela x deixará de ser utilizada. Como os dados, estão seguros na tabela y, pode-se extinguir a tabela x. Deixe para fazer isso após garantir que a migração dos dados de x para y tenha funcionado corretamente. A sintaxe do comando DROP TABLE é bem simples, conforme a seguir:

```
DROP TABLE <nometabela>;  
Exemplo:  
DROP TABLE aluno;
```

Após a execução de um DROP TABLE, a tabela com sua estrutura e seus dados deixa de existir permanentemente.

## TRIGGERS

A melhor definição para triggers em um banco de dados é a sua própria tradução para o português. Triggers são gatilhos que são disparados no banco de dados quando alguma operação de manipulação é executada. As triggers podem ser disparadas quando ocorre uma inserção, atualização ou exclusão de dados, respectivamente pelos comandos insert, update e delete. Dentro de uma trigger, será definida uma cadeia de comandos a serem executados naquela situação.

Ao criar uma trigger, além de definir a partir de qual operação ela será disparada (insert, update ou delete), é necessário definir qual será seu momento de execução. Se antes (before) ou depois (after) da execução do comando em si. Essa diferenciação ocorre, pois pode ser necessário executar algo antes de uma determinada operação ou logo após a operação ser concluída.

As triggers podem ser utilizadas para verificações, ou seja, para checar alguma condição antes de executar uma determinada operação. Um exemplo de verificação é a validação do saldo de uma conta corrente antes da execução de um saque.

Outra aplicação do uso de triggers é na geração de reflexos após uma determinada operação. Utilizando o mesmo contexto, uma trigger após a conclusão do saque para atualizar o saldo da conta corrente deduzindo o valor sacado pode ser desenvolvida. A sintaxe de para criação de uma trigger a seguir:

```
CREATE TRIGGER <nome trigger> <momento de execução> <evento disparo> ON  
<nometabela> FOR EACH ROW  
BEGIN  
  <comandos>  
END;
```

Onde:

- a) <momento de execução>, pode ser BEFORE ou AFTER, o que significa que a trigger será executada antes ou depois da execução da operação executada em <evento de disparo>.
- b) <evento de disparo>, pode ser INSERT, UPDATE OU DELETE,

Não existe um meio para invocar uma trigger, ou seja, forçar a execução dela. A execução de triggers ocorre somente através da execução automática pela ocorrência do evento programado na trigger, seja ele de inserção, atualização ou exclusão.

## PROCEDURES

Procedures são estruturas criadas no banco de dados, identificadas por um nome pelo qual podem ser chamadas. Um procedimento pode executar uma série de instruções, receber parâmetros e retornar valores.



A utilização de procedures se faz necessária quando uma série de comandos precisa ser executada em conjunto e repetidas vezes. Mantendo essa série de comando no banco de dados, facilita a operação de execução.

Outro fator que faz com que seja relevante a utilização de procedures é quando os comandos a serem executados implicam em um processamento de dados muito grande, com um tempo de execução elevado. No caso de execução de procedures, todo o processamento fica alocado no servidor, que geralmente possui uma configuração robusta, permitindo assim que esta execução ocorra de uma forma mais adequada.

Uma procedure pode receber um ou mais parâmetros para permitir direcionar o objeto de sua execução.

Para compreender melhor o conceito de procedure, pense no exemplo de uma função do sistema que concede um percentual de aumento nas notas de uma avaliação de acordo com um determinado critério do Professor. Se a turma o atingir, recebe um percentual de aumento na nota de uma avaliação conforme sua performance. Para não ter que reescrever esta atualização das notas a todo momento, pode ser criada uma procedure, conforme a seguir:

```
delimiter $$
create procedure patualizanotas (in pcdAvaliacao int,in pcdTurma int,in
ppercentual decimal)
begin
    update notas
        set nota = nota + (nota * ppercentual/100)
        where notas.avaliacao_idavaliacao = pcdAvaliacao
        and notas.idTurma = pcdTurma;
end$$
delimiter ;
```

A necessidade de alteração do delimitador, ocorre pela utilização do delimitador comum (;) dentro da procedure. Lembrando que, ao criar uma procedure, está apenas armazenando-a para futuras execuções. Quando for necessário executá-la, basta utilizar a sintaxe a seguir:

```
call <nomeprocedure> (parâmetros);
Para o exemplo:
call patualizanotas (1,1,1,10)
```

## VIEWS

Entre as facilidades que os banco de dados proporcionam ao desenvolvedor, encontra-se a view. Com o objeto view, pode-se criar uma operação de consulta (*SELECT*) e armazená-la para uso posterior até mesmo com outras pesquisas. A partir de sua criação, uma view se comporta como uma tabela do seu banco de dados. Isso significa que ela pode ser utilizada livremente em conjunto com outras tabelas, sejam elas views ou não para compor consultas compostas e detalhadas.

Acompanhe a seguir as vantagens de utilizar VIEWS.

- a) Velocidade de acesso às informações: Uma view é armazenada no SGDB na forma “compilada”, ou seja, como o SGDB entende que como você optou por ter de criar uma view é muito provável que você a utilize novamente. Por este motivo ela já está programada de forma otimizada. Desta forma quando a view for invocada o tempo de execução será muito mais rápido.
- b) Evitar desperdiçar tempo com retrabalho: Se você construir uma view, você pode utilizá-la em qualquer parte de seu software, com a vantagem do banco de dados interpretá-la como uma tabela, portanto não há prejuízos.
- c) Mascarar complexidade do banco de dados: Você pode criar uma view somente com as tabelas que são essenciais para um determinado usuário, ocultando outras que não precisam ser visualizadas por este usuário. Outra forma de uso é o de esconder colunas que podem ter valores restritos a determinados usuários do banco de dados.

A seguir, é apresentada a sintaxe de criação de uma view:

```
CREATE VIEW <nomedaview>  
AS <estrutura do select>;
```

Para apresentar um exemplo de criação de uma view, imagine a tabela de disciplina do modelo da Universidade. Para apresentar uma lista das disciplinas e seus pré-requisitos, pode ser criada uma view:

```
CREATE VIEW DisciplinaPreReq  
AS SELECT d1.idDisciplina,  
        d1.nomeDisciplina,  
        d2.idDisciplina as codPreRequisito,  
        d2.nomeDisciplina as nomePreRequisito  
FROM disciplina d1  
LEFT JOIN disciplina d2  
ON d1.idPreRequisito = d2.idDisciplina;
```

Dessa forma, toda vez que for necessário mostra uma disciplina é quem é seu pré-requisito, basta utilizar a view no select, como no exemplo a seguir:

```
SELECT disc.nomeDisciplina,  
       disc.nomePreRequisito  
FROM DisciplinaPreReq disc  
WHERE disc.idDisciplina = 'ADM0390';
```

Veja a seguir o resultado da execução da consulta>

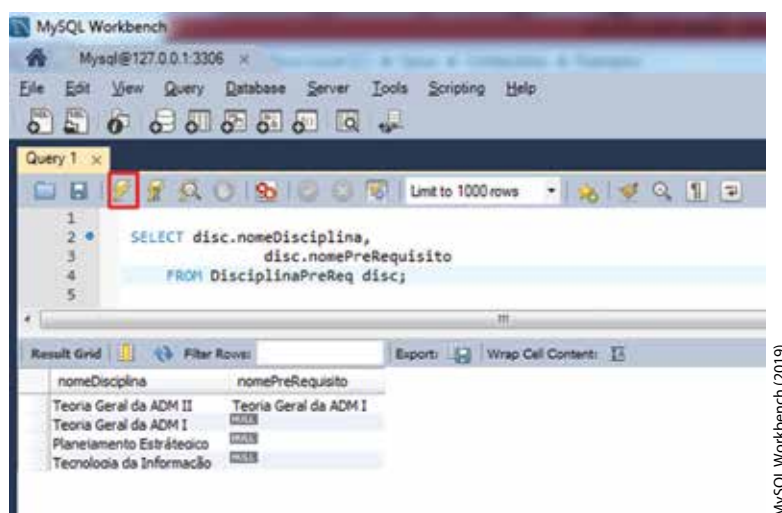


Figura 43 - Consulta com utilização de uma view no MySQL Workbench

Para ilustrar o seu estudo, a respeito de banco de dados, acompanhe o “Casos e Relatos”.



## CASOS E RELATOS

### Ferramenta de gestão

Uma equipe de desenvolvimento iniciou um projeto, criou seu banco de dados e entregou a aplicação em ambiente de produção do cliente.

Durante o período inicial do uso do aplicativo, foram encontradas algumas necessidades de ajustes. Essas necessidades envolviam modificações no banco de dados.

Ocorre que nenhum script SQL que foi executado no banco de dados do cliente foi armazenado. Os analistas apenas executavam os comandos e os descartavam. O banco de dados estava lá criado, e em funcionamento, mas havia necessidade de manutenção. Para isto foi necessário realizar o que é chamado de engenharia reversa, que é recriar estes scripts baseados no banco de dados físico. Para poder alterá-los e executá-los novamente.

Para que este problema não ocorresse, bastava que a equipe adotasse uma política de armazenamento de scripts, de preferência com a utilização de uma ferramenta de gestão de configuração, que permite a criação de histórico de versões para cada um dos arquivos gerenciados.

---

Agora chegou o momento de conhecer a linguagem DQL.

### 3.3.4 DQL – DATA QUERY LANGUAGE

A Data Query Language – DQL é responsável pela recuperação dos dados armazenados em um banco de dados. Ela é centralizada numa única estrutura, o comando SELECT, que possui uma variedade grande de composições. Um comando SELECT pode ser muito simples ou extremamente complexo. Entre os comandos SQL o SELECT é um dos mais utilizados.

O comando SELECT possui a seguinte sintaxe básica:

```
SELECT <colunas>  
FROM <nometabela1, nometabela2>  
WHERE <condições> (opcional);
```

Em um mesmo comando *select* você pode obter dados de diversas colunas e de diversas tabelas de um banco de dados. Pode, e deve inclusive, restringir o espectro de dados que deseja visualizar. As chamadas condições, são filtros aplicados nos dados selecionados afim de apresentar apenas os dados que importam. Para facilitar, você iniciará conhecendo o *select* contendo apenas uma tabela. Como exemplo, você precisa apresentar todos os dados da tabela aluno. Veja no exemplo a seguir como fazer isso:

```
SELECT * FROM aluno;
```

Perceba que foi utilizado o símbolo asterisco (\*) no lugar da definição das colunas. Também não foi definida nenhuma condição (*where*). Isto porque a proposta foi de apresentar todos os dados da tabela aluno:

o asterisco (\*) simboliza que todas as colunas das tabelas envolvidas na cláusula FROM sejam apresentadas; a ausência da cláusula *where* significa que todos os registros devem ser apresentados, sem nenhum tipo de restrição.

Agora você precisa apresentar apenas a matrícula e o nome de todos os alunos. Desta forma será necessário especificar quais colunas serão selecionadas. O comando ficará da seguinte forma:

```
SELECT idMatricula,nome FROM aluno;
```

Lembre-se que os nomes das colunas devem ser definidos exatamente com a mesma nomenclatura de quando criados e que deve-se utilizar a vírgula “,” como separador. Não há limite definido para quantidade de colunas a serem demonstradas. A ordem de apresentação das colunas fica a critério da necessidade de apresentação das colunas, não sendo mandatório obedecer a mesma ordem de como as colunas foram dispostas na criação da tabela.

### Operadores de Comparação

Muitas vezes é necessário restringir os dados que se deseja apresentar. Por exemplo, pode ser necessário resgatar os dados apenas de clientes de um determinado bairro, pacientes que tenham mais que uma determinada idade, produtos com preço abaixo de um determinado valor, entre outras especificidades que incluam a obrigação de realizar comparação entre informações.

Para atender essas situações a linguagem SQL incorporou os operadores de comparação. Os operadores de comparação da linguagem SQL estão listados na tabela a seguir:

OPERADOR	FUNCIONAMENTO	EXEMPLO
=	Igual a	nota = 10
>	Maior que	preco > 50
>=	Maior ou igual a que	nota >= 7
<	Menor que	nota < 7
<=	Menor ou igual a que	preco <= 2
<>	Diferente de	bairro <> 'Centro'

Quadro 4 - Os operadores de comparação da linguagem SQL  
Fonte: Do autor (2019)

Os operadores de comparação são utilizados em conjunto com a cláusula *where*.

Para exemplificar o uso, com as tabelas do banco criado, suponha que precisará desenvolver uma funcionalidade no sistema em que ao informar o CPF de um aluno, seja apresentado seu nome e matrícula. Para atender esta necessidade o comando a ser preparado deverá incluir a cláusula *where* e o uso de um operador de comparação:

```
SELECT matricula, nome  
FROM aluno  
WHERE cpf = '00000000000';
```

Onde '00000000000' é o possível CPF digitado pelo usuário do sistema.

Operadores de comparação geralmente são utilizados em conjunto com operadores lógicos, os quais conhecerá a seguir.

### Operadores Lógicos

Operadores lógicos são utilizados quando existe a necessidade de conjugar mais de uma condição para compor uma cláusula *where*.

Os operadores lógicos utilizados em SQL são:

OPERADOR	FUNÇÃO
AND	Retorna verdadeiro se ambas as condições forem verdadeiras
OR	Retorna verdadeiro se uma das condições for verdadeira
NOT	Retorna verdadeira se a condição a seguir for falsa

Quadro 5 - Os Operadores Lógicos da Linguagem SQL  
Fonte: Do autor (2019)

Imagine a seguinte situação: é solicitado uma consulta que exiba todos os códigos de alunos que obtiveram a nota menor que 7 em uma determinada avaliação (código 29). Esta consulta exige dois critérios de busca, nota e código de avaliação. Para resolver este caso, pode-se utilizar a seguinte consulta:

```
SELECT codAluno  
FROM notas  
WHERE notas.nota > 7  
AND notas.idAvaliacao = 29;
```

Outro tipo de operador disponível na linguagem SQL são os operadores aritméticos, dos quais você conhecerá melhor a seguir.

### Operadores Aritméticos

Os operadores aritméticos são utilizados para executar cálculos matemáticos utilizando os dados contidos em um banco de dados.

OPERADOR	FUNÇÃO
+	Soma
-	Subtração
/	Divisão
*	Multiplicação
%	Módulo
DIV	Divisão Inteira

Quadro 6 - Os Operadores Aritméticos da Linguagem SQL  
Fonte: Do autor (2019)

Na utilização de operadores aritméticos, a regra matemática da precedência das operações e dos parênteses é válida.

Imagine uma funcionalidade do sistema que apresenta o quanto faltou para o aluno obter naquela nota para chegar na média, se estiver abaixo da média. Supondo que a média seja 7, veja a consulta a seguir:

```
SELECT cdAluno, cdAvaliacao, 7 - nota  
FROM notas  
WHERE nota < 7;
```

### SELECT COM MAIS DE UMA TABELA

Na origem do modelo relacional, está a divisão dos dados entre tabelas. Essas tabelas contêm dados que estão vinculados entre si, através de chaves estrangeiras. Muitas vezes os usuários nos requisitam consultas em que é preciso apresentar resultados que envolvem dados que estão armazenados em tabelas diferentes.

Para isso, o comando SELECT que seja selecionado dados de mais de uma tabela. Para que isso funcione de forma adequada, é necessário realizar uma junção entre as tabelas envolvidas.

Uma junção é uma instrução para que o banco de dados una os dados das duas ou mais tabelas envolvidas de acordo com o critério determinado.

### INNER JOIN

O INNER JOIN é o tipo de junção mais utilizado. Em um Inner join os dados serão resgatados de ambas as tabelas que compõe a junção, apenas quando ambos os registros satisfizerem a condição determinada.

Na figura a seguir, pode ser identificado como funciona um Inner join. Os dados resgatados serão apenas os que fizerem parte da intersecção dos dados.

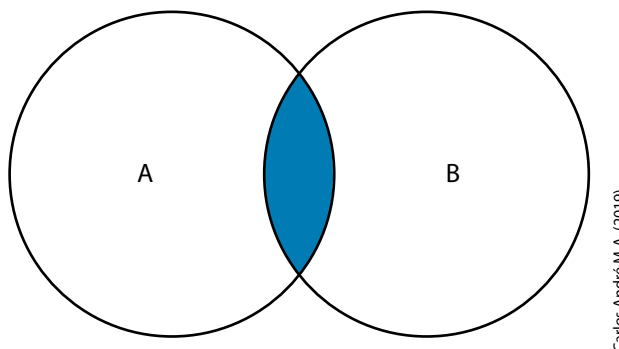


Figura 44 - Representação Gráfica de um Inner Join  
Fonte: Do autor (2019)

A sintaxe de uma instrução Select que contenha INNER JOIN é a seguinte:

```
SELECT <colunas>
FROM <tabelaA>
INNER JOIN <tabelaB>
ON <condição de junção>
WHERE <condição de filtro> (Opcional);
```

A condição de junção está diretamente vinculada a como as duas tabelas envolvidas se relacionam. Em resumo, devem ser informadas as colunas que compõe a chave estrangeira entre as tabelas.

Um exemplo que pode ser utilizado é uma consulta que exiba o nome do curso, e nome das suas disciplinas. O nome do curso está na tabela "curso". O da disciplina, na tabela "disciplina". E estas tabelas se ligam através de uma terceira tabela a "disciplinacurso". Veja como ficará o select que atende esta necessidade a seguir:

```
SELECT c.nomeCurso,
       d.nomeDisciplina
FROM curso c
INNER JOIN disciplinacurso dc
ON c.idCurso = dc.curso_idCurso
INNER JOIN disciplina d
ON d.idDisciplina = dc.disciplina_idDisciplina;
```



Você precisa atentar a algumas sintaxes novas: foi determinado um alias ou apelido para cada uma das tabelas. Isto tem como objetivo facilitar a menção a esta tabela em outras partes do select. A determinação de um alias é feita através da digitação de uma sigla ou de uma abreviação logo após o nome da tabela posicionada na cláusula FROM.

Outra novidade foi a inclusão do *alias* da tabela antes do nome da coluna na projeção. Isto é necessário para que fique claro ao interpretador de consultas do SGDB de qual tabela é aquela coluna. Muitas vezes é preciso o mesmo nome de coluna em tabelas diferentes, e se estas tabelas estiverem no mesmo select, ao executar, ocorrerá um erro. Pois o SGBD não conseguirá determinar de qual tabela deverá resgatar a informação.

### LEFT JOIN

Nem sempre a junção entre duas tabelas ocorre da forma descrita em INNER JOIN. Pode ser necessário considerar que uma das tabelas não contenha os dados. Isso ocorre quando o relacionamento tem cardinalidade mínima 0, ou seja, não obrigatório.

Dessa forma, pode ser necessário resgatar os dados que contenham ou não os registros da tabela relacionada.

Veja o exemplo a seguir:

```
SELECT d1.idDisciplina,  
       d1.nomeDisciplina,  
       d2.idDisciplina as codPreRequisito,  
       d2.nomeDisciplina as nomePreRequisito  
FROM disciplina d1  
LEFT JOIN disciplina d2  
ON d1.idPreRequisito = d2.idDisciplina;
```

Para esse caso, o banco de dados retornará os registros, independente dos dados da tabela "d2" existirem. O left significa que o sistema retornará os dados da tabela a esquerda do join, mesmo que os dados na tabela à direita não correspondam a junção. Desta forma, se existirem registros que satisfizerem a junção, serão resgatados por completo, mas se existirem registros em "d1" que não correspondem em "d2" conforme a junção determinada, os dados de "d1" serão recuperados mesmo assim. Os de "d2" serão apresentados como nulos.

## RIGHT JOIN

Para trabalhar com uma junção não obrigatória, pode ser utilizada também a sintaxe RIGHT JOIN. Ela é o inverso de LEFT JOIN. Neste caso, os dados serão retornados independentemente de existirem ou não na tabela a esquerda do JOIN. Basta existirem na tabela da direita da cláusula RIGHT JOIN.

## ORDENAÇÃO DE DADOS

Ao executar uma consulta SQL, um select na maioria dos casos, se recebe como retorno uma coleção de registros. Muitas vezes esta coleção envolve centenas e até milhares de registros. Para que a visualização de dados seja mais adequada, é indicado que estes dados sejam ordenados.

Para tanto, pode ser adicionada a cláusula ORDER BY nas consultas SELECT.

A sintaxe do SELECT com a inclusão de ORDER BY ficará da seguinte forma:

```
SELECT <colunas>
FROM <tabelas>
WHERE <condição> (Opcional)
ORDER BY <campos> ASC ou DESC;
```

Você pode definir mais de um campo para a sua ordenação. Também é possível definir se essa ordenação ocorrerá de forma ascendente ou descendente. Quando se deseja que a ordenação ocorra de forma ascendente, deve-se utilizar a palavra reservada ASC, quando de forma descendente a palavra reservada DESC. Por padrão, quando não for informado nem ASC nem DESC, a ordenação será ASC ou seja ascendente.

Imagine um caso em que o professor deseja verificar a performance dos seus alunos, verificando primeiro os alunos com nota menor para os alunos que tem nota maior em determinada disciplina.

```
SELECT al.idMatricula,
       al.nome,
       n.nota
FROM notas n
INNER JOIN aluno al
ON n.idMatriculaAluno = al.idMatricula
WHERE n.avaliacao_idAvaliacao = 1
ORDER BY n.nota ASC
```

### 3.3.5 DML – DATA MANIPULATION LANGUAGE

A Data Manipulation Language – DML é responsável pela inserção, alteração e exclusão de dados em um banco de dados.

Através de DML você conseguirá popular e manter seu banco de dados com as informações que necessita.

Um sistema de informação quando executa suas operações de cadastro e manipulação de informações está interagindo com o banco de dados através de comandos DML.

Você irá estudar estes comandos na ordem natural, primeiro a inserção de dados, depois a alteração, e depois a exclusão.

#### INSERT

O comando INSERT tem como objetivo realizar a inserção de dados em uma tabela no banco de dados. A inserção de dados ocorre em uma tabela de cada vez. Veja a sintaxe do comando INSERT, a seguir:

```
INSERT INTO <nometabela>
          (coluna1,coluna2,...)
VALUES(<valor1,<valor2,<...>);
```

A seguir você acompanha algumas considerações relevantes em relação ao comando insert.

- O nome das colunas deve ser o nome das colunas da tabela em questão.
- Todas as colunas obrigatórias devem estar relacionadas, sob pena da ocorrência de erro no caso de alguma não seja informada.
- Os valores devem ser informados na mesma ordem que as colunas foram informadas. É assim que o banco vai entender o quê atribuir aonde.
- Os valores informados devem respeitar os tipos de dados pré-determinados no momento da criação da tabela. Se uma coluna é do tipo inteiro, apenas números inteiros podem ser informados nela. Se uma coluna é do tipo data, apenas datas podem ser informadas nela, e assim com cada tipo de dado. A informação de um tipo de dado diferente pré-definido causará erro na execução do comando insert.
- Os valores informados devem respeitar todas as restrições de integridade pré-definidas. Portanto se uma coluna faz parte de uma chave estrangeira, você precisa informar valores que respeitem aquela chave estrangeira.

Para ficar mais claro, veja um exemplo de um INSERT. Nesse caso, serão inseridos 4 registros na tabela disciplina:

```
INSERT INTO disciplina (idDisciplina, nomeDisciplina, idPreRequisito) values  
(29, 'Teoria Geral da ADM I', null);  
insert into disciplina (idDisciplina, nomeDisciplina, idPreRequisito) values  
(39, 'Planejamento Estratégico', null);  
insert into disciplina (idDisciplina, nomeDisciplina, idPreRequisito) values  
(33, 'Tecnologia da Informação', null);  
insert into disciplina (idDisciplina, nomeDisciplina, idPreRequisito) values  
(11, 'Teoria Geral da ADM II', 29);
```

A seguir, a execução do script na ferramenta:

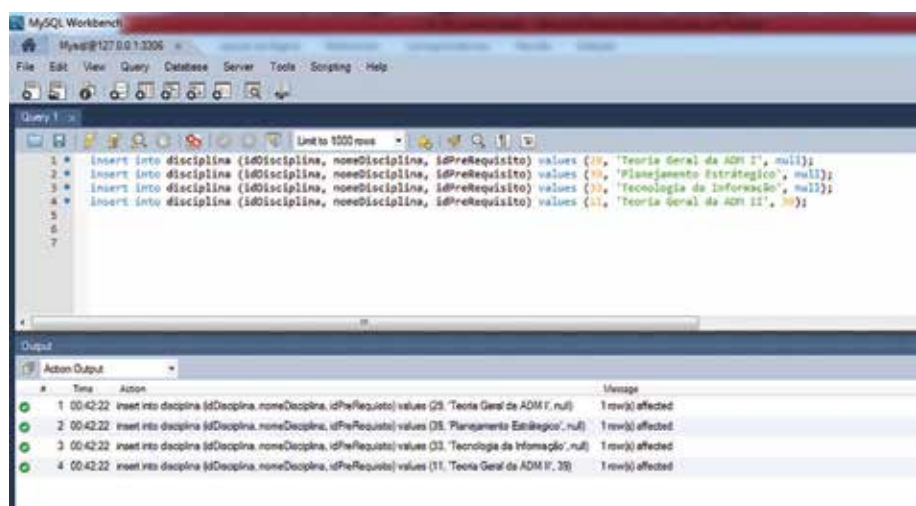


Figura 45 - Execução de um Insert no MySQL Workbench



**SAIBA  
MAIS**

Existe outra sintaxe possível para a execução de um insert, que é a inclusão de registros baseado numa consulta SELECT. Ao invés do usuário digitar os dados a serem inseridos, os dados são executados baseados no resultado de uma consulta. A sintaxe e as regras para este tipo de inserção você pode verificar em: <https://dev.mysql.com/doc/refman/8.0/en/insert-select.html>.

## UPDATE

Por muitas vezes surge a necessidade de alterar dados que estão armazenados no banco de dados. Para atender a essa necessidade, a linguagem SQL fornece o comando *update*, que tem a seguinte sintaxe:

```
UPDATE <nometabela>  
    SET <coluna1> = <valor1>, <coluna2> = <valor2>  
    WHERE <condição> (opcional, porém recomendável);
```

É importante detalhar algumas características do comando *update*. Acompanhe.

- a) Atualiza apenas uma tabela por vez.
- b) Permite atribuir novo valor apenas para as colunas que necessita naquele momento. As demais permanecem com o valor já armazenado.
- c) Permite atribuir novo valor para mais de uma coluna por vez.
- d) Permite atribuir novos valores para mais de um registro por vez.
- e) Permite definir condições, selecionando os registros que serão atualizados.
- f) Da mesma forma que na inserção, na atualização de dados todas as regras de integridade serão checadadas. Portanto não será permitido atualizar uma coluna para um valor que não seja permitido considerando as restrições de integridade.
- g) Em um comando *update* a condição *where* não é obrigatória, mas é extremamente recomendável. Se uma condição não for determinada o comando *update* atualizará as colunas determinadas de TODOS os registros daquela tabela. Na maioria das vezes a necessidade é atualizar um registro específico ou no máximo um conjunto de registros que satisfaçam uma determinada condição.
- h) Se a condição determinada (*where*) não retornar nenhum registro, a atualização não será aplicada.

Exemplo de utilização de *update*: um aluno quando cadastrado seu nome foi registrado incorretamente. O aluno de matrícula número 407 teve o nome registrado como João da Silva Santos, sendo que o nome correto dele é João dos Santos Silva.

O comando que satisfaz esta necessidade é:

```
UPDATE aluno  
    SET nome = "João dos Santos Silva"  
    WHERE matricula = 407;
```

Veja, na figura a seguir, a execução do comando UPDATE. E na sequência a execução do comando SELECT. Dessa forma, é possível checar se os dados foram atualizados corretamente.

É importante destacar na figura a seguir, a seção *Output*. Nesta seção do MySQL Workbench, são listados todos os *feedbacks* em relação a cada comando executado. Se um comando ocasionar um erro, é nesta seção que você deverá verificar a mensagem. As mensagens são instrutivas e, na maioria das vezes, o erro se relaciona a uma digitação incorreta da sintaxe de um comando ou com a violação de uma restrição de integridade. Portanto ao executar qualquer comando, é obrigatório que se verifique qual foi o retorno que o banco de dados lhe deu no Output.

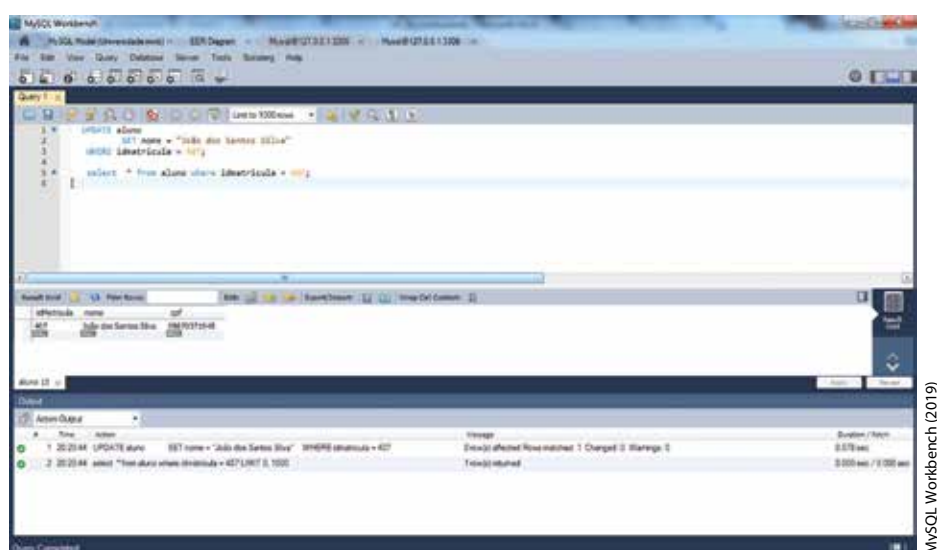


Figura 46 - Execução de Comando Update no MySQL Workbench.

## DELETE

O comando DELETE é utilizado para excluir dados de tabelas do seu banco de dados. Este comando deve ser utilizado sempre com muito cuidado. Dados excluídos não podem ser resgatados.

A sintaxe do comando DELETE é:

```
DELETE FROM <nometabela>
WHERE (opcional – porém recomendável);
```

Acompanhe algumas características importantes em relação ao comando *update*.

a) É informada apenas uma tabela por vez.

b) A condição *where*, apesar de opcional, é extremamente recomendada. Quando um comando *delete* é executado sem *where*, a exclusão atinge todos os registros da tabela. Geralmente exclusões são realizadas considerando condições específicas.

c) Ao realizar a exclusão de um registro, todas as regras de integridade são verificadas. Isso significa que não é permitido excluir um registro que está vinculado a um registro vinculado a uma chave estrangeira.

Agora você precisa conhecer um exemplo de exclusão de dados. Para excluir um professor, o de matrícula 90, você precisa executar o seguinte comando:

```
DELETE FROM professor  
WHERE matricula = 90;
```

Ao executar este comando o registro da tabela professor que tem a matrícula com código igual a 90 será excluído. Lembrando que esta exclusão só será permitida se não violar as regras de integridade. Caso haja alguma dependência em alguma tabela que tenha ligação com a tabela Professor, o SGDB retornará um erro.



#### FIQUE ALERTA

É importante atentar aos valores nulos em um banco de dados. Nulo não é zero (0), nem branco. Nulo é simplesmente a ausência de valor. Porque ao atribuir um valor zero(0) ou em branco (""), o usuário está optando por isso. No caso do nulo, o usuário não indicou nenhum valor. Portanto, ao somar um determinado valor com nulo, se tem na verdade um erro.

Para tratar esta situação, o MYSQL disponibiliza a função IFNULL. A função recebe dois parâmetros, a coluna que precisa ser analisada, e o valor que ela deve assumir quando o valor dela for nulo.

Por exemplo:

```
SELECT IFNULL(preco, 0) FROM produto;
```

Desta forma, quando o valor do campo preço for nulo, na consulta será substituído por zero.

Na sequência acompanhe novo assunto: DTL

### 3.3.6 DTL – DATA TRANSACTION LANGUAGE

Uma das preocupações ao se trabalhar com banco e dados que você precisará ter é em relação ao controle de transações. Transações são necessárias para controlar o fluxo de alterações de dados em uma base. Imagine um sistema bancário em que diversas operações ocorrem ao mesmo tempo. Um saque não pode ocorrer ao mesmo tempo que outro, pois a soma dos dois saques podem comprometer mais do que o saldo disponível, não é mesmo?

Outro ponto importante é a da suscetibilidade a erros. Ao executar uma série de operações de um mesmo conjunto, deve-se ter a possibilidade de confirmar tudo ou nada. Se uma série de comandos forem executados em conjunto, e algum deles ocasionar um erro, se tem que ter condições de anular o efeito de todos.

Para isso a linguagem DTL proporciona três comandos: BEGIN (TRANSACTION), COMMIT e ROLLBACK;

O comando BEGIN indica ao banco de dados que você pretende iniciar uma transação. A partir da execução deste comando, todas as operações DML (manipulação de dados) que você executar estarão dentro desta transação. Uma transação é concluída no momento que você executar a confirmação ou a anulação desta transação. Caso você deseje confirmar ao banco de dados que as operações executadas, você deve utilizar o comando COMMIT;. Caso deseje abortar as operações executadas, o comando ROLLBACK;. Após um BEGIN enquanto não houver um COMMIT ou ROLLBACK a transação permanece em aberto e as manipulações realizadas não estarão disponíveis na base de dados. Não é recomendável manter transações em aberto por um longo período de tempo. Imagine que estes dados podem ser necessários para outros usuários e estarão “presos” na sua transação aberta.

Na figura a seguir é demonstrada uma execução de script em que o usuário inicia uma transação, executa uma sequência de 4 comandos de inserção e aborta a operação (mediante comando rollback). Ao executar o comando select para verificar se os dados foram inseridos, é identificado que a tabela está vazia.

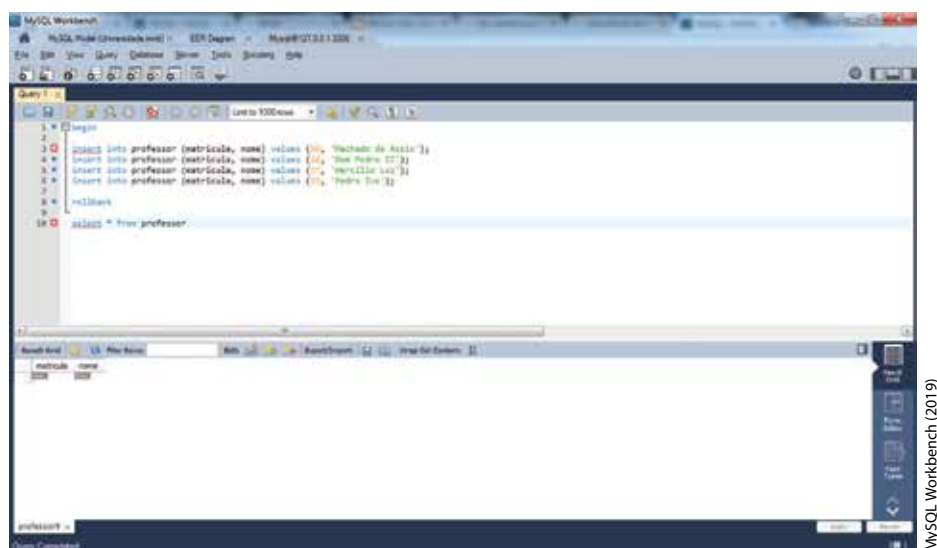


Figura 47 - Execução de uma transação com rollback no MySQL Workbench

Agora, você pode verificar um exemplo em que a transação é, ao final, confirmada. No script abaixo, o usuário inicia uma transação, executa uma sequência de 4 comandos de inserção e confirma a operação (mediante comando commit). Ao executar o comando select para verificar se os dados foram inseridos, é identificado que a tabela os dados estão persistidos lá.



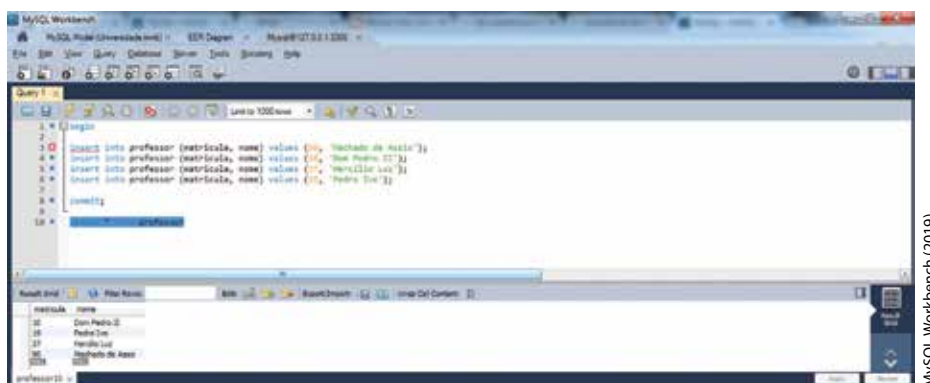


Figura 48 - Execução de uma transação com Commit no MySQL Workbench

### 3.3.7 DCL – DATA CONTROL LANGUAGE

DCL é a divisão da linguagem SQL que trata da criação, manipulação, exclusão e controle de acesso de usuários de um banco de dados. A necessidade de se criar diferentes usuários ao banco de dados se dá pelas diferentes necessidades que cada usuário tem ao acessar um banco de dados. Por exemplo, em um sistema que disponibilize um acesso de consulta para um sistema disponível publicamente na WEB, o usuário deste sistema WEB que acessará o banco de dados deverá ter o mínimo de permissões, provavelmente apenas permissão de consulta aos dados nas tabelas necessárias. Já usuários administradores, devem ter maiores permissões, para permitir que as operações sejam realizadas e a manutenção do banco de dados ocorra com sucesso.

Quando o banco de dados é instalado, um usuário e senha padrão são definidos. A partir deste acesso, que é um acesso com todos os privilégios, deve-se criar os demais usuários. Este usuário padrão deve ser preservado e de preferência não utilizado.

A regra para as permissões de usuários é de que um usuário só tenha permissão naquilo que lhe seja necessário. Isto evita acesso a operações não permitidas ou a ocorrência de danos aos dados por engano.

O comando para criação de usuários é o *CREATE USER*. A sintaxe do comando *CREATE USER* é:

```
CREATE USER <nomeusuario> IDENTIFIED BY <senha>;
```

Quando é necessário excluir um usuário o comando a ser utilizado é o *DROP USER*. Segue abaixo a sintaxe do comando *DROP USER*:

```
DROP USER <nomeusuario>;
```

Quando um usuário é criado ele não tem privilégio nenhum. Isso significa que ele não consegue fazer absolutamente nada no banco de dados.

Os comandos para manipulação destas permissões são *GRANT* e *REVOKE*.

*GRANT* tem como objetivo conceder as permissões e *REVOKE* revoga-las.

A sintaxe do comando *GRANT* inclui a determinação de qual privilégio em qual objeto e para qual usuário. Veja a seguir a sintaxe deste comando:

```
GRANT <privilegio> ON <objeto> to <usuário>;  
Para conceder a permissão de consulta em uma tabela, por exemplo:  
GRANT SELECT ON professor to usuario1;
```

Quando é preciso retirar uma permissão previamente concedida, deve-se utilizar o comando *REVOKE*.

```
REVOKE <privilegio> ON <objeto> FROM <usuário>;  
Portanto para remover um privilégio de consulta de  
um usuário, você pode utilizar o comando:  
REVOKE select ON professor from usuario1;
```

Acompanhe, a seguir, o Recapitulando do capítulo.



## RECAPITULANDO

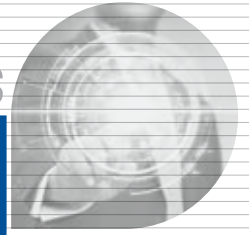
---

Neste capítulo, você teve a oportunidade de conhecer os métodos, as ferramentas e as melhores práticas para conceber um banco de dados do zero. Dessa forma, você está capacitado a construir bancos de dados para sistemas computacionais que resolvem os mais diversos problemas de negócio de organizações pequenas ou grandes, público ou privadas. É importante ressaltar a necessidade das etapas de modelagem com o intuito de garantir a melhor solução para o problema, estudando-o, dividindo-o em partes e finalmente criando o banco de dados para uso.

---







A administração exerce um papel fundamental na sociedade atual. Tem papel econômico e de desenvolvimento social. Atualmente, todas as áreas profissionais precisam lidar com administração. Os negócios passaram de locais para mundiais com a evolução da tecnologia. A administração é uma necessidade de todos os negócios, sejam públicos ou privados, pequenos ou grandes, familiares ou societários.

Outro quesito fundamental nas organizações modernas é da ampla aplicação da ética. Atualmente as empresas investem em políticas e em treinamentos de *compliance*. O termo *compliance*, é originado do verbo em inglês “to comply” que tem como significado “cumprir”. Ou seja, *compliance* é cumprir o que foi proposto, é agir em consonância com leis e regulamentos internos e externos.

Ao final desse capítulo você estará apto a:

- a) conhecer os aspectos éticos da informática na sociedade;
- b) entender o que são diretrizes empresariais.

Prepare-se e vamos lá!

## 4.1 ASPECTOS ÉTICOS DA INFORMÁTICA NA SOCIEDADE

A palavra ética tem origem no termo grego “ethos”, que significava “bom costume”, “costume superior”, ou “portador de caráter”.



guirong hao ([20-?])

A influência da informática na sociedade pode ser naturalmente percebida. Desde o início de seu uso e principalmente nos dias atuais, hoje praticamente todas as áreas profissionais utilizam, se beneficiam e dependem de tecnologia da informação.

O profissional de tecnologia é de extrema importância nesse contexto porque ele é quem faz a interface entre a sociedade e os ambientes organizacionais, com o intuito de estabelecer diretrizes para que os processos sejam realizados da forma ágil, segura, eficiente e eficaz.

Na maioria dos casos, a tecnologia é utilizada com grande benefício para a sociedade, com o intuito de reduzir a necessidade de tarefas manuais, de automatizar processos trabalhosos e delicados, e com o intuito de permitir que as organizações tenham acesso a suas informações de forma rápida e estruturada.

Infelizmente a tecnologia também pode ser utilizada para atividades ilícitas ou prejudiciais a outras organizações e pessoas. São exemplos de mau uso, o vazamento de informações, as fraudes digitais, os acessos indevidos, entre outros delitos e infrações que são realizadas utilizando a tecnologia como meio. Vale lembrar que a má intenção é sempre do ser humano que opera, programa e implanta o mecanismo digital para obter a vantagem indevida ou a ação prejudicial a um terceiro.

A ética dentro de Tecnologia da Informação precisa levar em conta o profissional da área e o usuário da tecnologia, pois ambos lidam diariamente com os sistemas, e, portanto, precisam ter responsabilidade nas suas definições em relação a princípios éticos, profissionais, regulamentações de cada setor, legislações.

Um profissional de Tecnologia da Informação deve tomar cuidado em relação às solicitações de seus usuários. Se esta solicitação é ilegal, ou infringe alguma política definida daquela organização, deve imediatamente comunicar os responsáveis para que o caso seja avaliado.



metamorworks (20-71)

Imagine que você foi designado para realizar a análise e desenvolvimento de um sistema de emissão de notas fiscais. O usuário chave solicita que o sistema faça o cálculo de imposto diferente do que está determinado na legislação, o que caracteriza uma sonegação de imposto. Você não pode ficar alheio a esta situação. Realizar o desenvolvimento desta forma é ser conivente com este crime, que é o de sonegação fiscal.

Uma das diretrizes essenciais para o para o profissional de TI é a formalização das solicitações dos usuários. Sempre realizar o registro das solicitações dos usuários, é uma forma de se resguardar em relação à eles em futuros questionamentos. Muitas vezes um profissional de TI não terá condições de delimitar se determinado escopo é ético ou legal, pois se trata de uma área muito específica, como por exemplo, o Direito, a área Médica ou Contábil. Esse tipo de registro assegura a origem das solicitações e permite que o profissional tenha esta retaguarda a seu favor.

Dentro deste contexto, surge a antiga discussão do que ético ou não. Nesse caso, aplicado a tecnologia da informação, algumas questões podem ser colocadas na sequência.

- a) É tolerado acessar os arquivos de outros funcionários?
  - b) É permitido que o responsável imediato de um profissional monitore eletronicamente as atividades dos seus comandados?
  - c) É admitido realizar cópias de softwares sem autorização, instalar em computadores pessoais ou em organizações públicas ou privadas?
  - d) É permitida a ocorrência de acesso de pessoas estranhas a informações de clientes disponíveis em bancos de dados?
  - e) Os funcionários de uma empresa podem utilizar a estrutura da TI da empresa para fins pessoais?
  - f) Pode o funcionário da empresa utilizar seu equipamento pessoal para desenvolver atividades da empresa?
- Acompanhe na sequência um “Caso e Relatos” para ilustrar seu aprendizado.



## CASOS E RELATOS

---

### Tecnologia e Sociedade

A urna eletrônica começou a ser utilizada nas eleições municipais do Brasil em 1996. A implantação que foi vista durante muito tempo como um avanço para o país, mas atualmente é questionada por diversos grupos da sociedade civil organizada por estar suscetível a falhas. Periodicamente o TSE (Tribunal Superior Eleitoral) realiza eventos públicos de testes, em que profissionais e acadêmicos tentam burlar este sistema. Foi identificado nestas seções vulnerabilidades que foram corrigidas pelo TSE, mas ainda assim restam dúvidas sobre a integridade das urnas. Neste contexto, em 2015, foi aprovada a Lei n. 13.165 que versa sobre a implantação do voto impresso nas eleições brasileiras. Nesta proposta, a urna eletrônica continua existindo, mas imprime cada voto realizado para que possa ser feitos conferências pontuais ou não. Este é um exemplo em que a tecnologia é usada a favor da sociedade, mas que por receio de fraudes, se obrigou a criar um mecanismo de validação e controle.

---

Em relação ao tema banco de dados, existem vários aspectos que podem ser considerados, éticos ou não.

Quando você trabalha com bancos de dados reais, está lidando com informações organizacionais preciosas. Como profissional de tecnologia da informação, deve trabalhar com ética no sentido de não utilizar as informações contidas nos sistemas que tem acesso em benefício próprio, nem tampouco divulgá-las a terceiros.





Também não pode modificar qualquer informação no banco de dados sem a devida autorização. Afinal, o “dono” dos dados é quem determina o que deve ser inserido em um banco de dados.

Considerando que as equipes de Tecnologia da Informação possuem acesso privilegiado às informações, as configurações, aos equipamentos, aos serviços disponibilizados aos clientes e até mesmo as senhas, é essencial que sejam definidas regras próprias para reger sua atuação, com a definição de um termo de responsabilidade específico, de confidencialidade e, sempre que houver desenvolvimento interno, termo de cessão de código fonte. Já para a organização como um todo, em nível mais estratégico, deve existir um Plano Diretor de Tecnologia da Informação dando ciência a todos de seus direitos, deveres e responsabilidades considerando cada contexto organizacional.



É necessário ficar atento às questões legais que envolvem a produção e a licença de um software. Um desenvolvedor que trabalha em uma empresa não tem direito sobre o código fonte que ele desenvolve lá. A empresa precisa firmar com ele um termo de cessão do direito intelectual sobre o código fonte desenvolvido lá. Dessa forma, garante que o desenvolvedor não irá requerer estes direitos no futuro.

Geralmente, quando tem acesso a bancos de dados reais, é necessário assinar termos de confidencialidade específicos de cada cliente para garantir a ciência das responsabilidades do acesso e uso das informações obtidas.

**SAIBA  
MAIS**

Recentemente veio a público o caso de uma empresa de marketing político ter utilizado informações do Facebook na campanha eleitoral dos Estados Unidos. O Facebook está sendo investigado e a empresa sofre com a queda de suas ações na bolsa de valores. Saiba mais no link: <https://exame.abril.com.br/tecnologia/o-que-voce-precisa-saber-sobre-o-vazamento-de-dados-do-facebook/>

## 4.2 DIRETRIZES EMPRESARIAIS

As diretrizes empresariais são determinadas por uma ferramenta denominada planejamento estratégico. O planejamento estratégico é composto de pontos básicos que dão o direcionamento inicial do negócio. Antes mesmo de fazer a análise estrutural, financeira e de custo, é primordial definir qual o papel daquela empresa perante a sociedade, qual sua finalidade e ou objetivo, quais benefícios trarão e qual diferencial irá oferecer ao mercado e ao seu público. É necessário se concentrar primeiramente, nestes três pontos: missão, visão, valores.

### 4.2.1 MISSÃO

A missão da empresa é composta daquilo que a organização pretende oferecer aos clientes em decorrência dos produtos e/ou serviços que irá oferecer ao mercado. Na missão da empresa são determinadas as características do negócio, os principais assuntos que serão atendidos por ele.

São exemplos de missão de empresa de desenvolvimento de software:

- a) desenvolver soluções tecnológicas que potencializem o valor dos negócios das organizações;
- b) atender o mercado de software para concessionários de energia elétrica, garantindo a satisfação dos clientes, colaboradores, sócios, parceiros;
- c) ser reconhecida pelo mercado no âmbito nacional como empresa referência em qualidade de software de gestão empresarial.

### 4.2.2 VISÃO

A visão descreve os objetivos que a organização pretende alcançar com as suas atividades que foram descritas na missão. A visão demonstra aonde a empresa quer chegar. São exemplos de visão:

- a) atender clientes em todas as regiões do País;
- b) ser reconhecida como uma empresa de classe mundial;
- c) ser líder no Brasil em soluções web para gestão de transportes, destacando-se na qualidade dos produtos e serviços.

**CURIOSIDADES**

Conheça a missão e visão de uma das maiores empresas de tecnologia do mundo, a Microsoft.

**Missão da Microsoft**

Permitir às pessoas e empresas, em todo o mundo, a concretização do seu potencial.

**Visão da Microsoft**

Disponibilizar às pessoas software de excelente qualidade – a qualquer momento, em qualquer local e em qualquer dispositivo.

### 4.2.3 VALORES

Os valores são os ideais norteadores daquela organização. Geralmente, são cunhados com base em valores pessoais dos fundadores daquela empresa. Estes valores irão orientar as ações da empresa em sua operação.

Exemplos de valores: Sustentabilidade, Inovação, Confiança, Gestão de Pessoas, Diversidade, Responsabilidade social.

### 4.2.4 POLÍTICA DE QUALIDADE

A política de qualidade é a determinação do que é qualidade para aquela organização. É estranho pensar que a definição de qualidade é diferente de uma empresa para outra, mas você deve conhecer empresas do mesmo ramo que entregam produtos ou serviços com qualidade diferentes.



marchmeena25 ([20-?])

Por isso é necessário definir para aquela organização qual é a sua política de qualidade. A política da qualidade deverá formatada em um documento e divulgada a todos. Mas o melhor de uma política de qualidade é quando ela que é conhecida e aplicada por todos na organização.



## RECAPITULANDO

---

Dentro de uma organização, seja ela público ou privada, mesmo atuando na área técnica é essencial que você esteja atualizado e alinhado com os aspectos empresariais. Neste Capítulo, você conheceu os conceitos de ética e de como ela se relaciona com a tecnologia no mundo moderno. Também conheceu alguns aspectos relativos ao planejamento estratégico de uma organização. Sempre que estiver atuando em uma organização é essencial que busque estes conceitos e aplique-os no seu dia a dia de trabalho

---





## REFERÊNCIAS

---

AUTOMATIONS, All Round. **All Round Automations PL/SQL Developer**. Disponível em: <<https://www.allroundautomations.com/plsqldev.html>>. Acesso em: 20 maio 2018.

BLOK, Marcella. **Compliance e Governança Corporativa**: atualizado de acordo com a lei Anticorrupção Brasileira (Lei 12.846) e o Decreto-Lei 8.421/2015. Rio de Janeiro: Freitas Bastos, 2017.

CHIAVENATO, Idalberto. **Administração para não administradores**: a gestão de negócios ao alcance de todos. 2. ed. Baruer: Manole, 2011.

ELMASRI, Ramez; NAVATHE, Sham. **Sistemas de banco de dados**. 6. ed. São Paulo (SP): Pearson, c2011. 788 p.

HEIDISQL. **HeidiSQL is a powerful and easy client for MySQL, MariaDB, Microsoft SQL Server and PostgreSQL. Open source and entirely free to use**. Disponível em: <<https://www.heidisql.com/>>. Acesso em: 20 maio 2018.

HEUSER, Carlos A. **Projeto de banco de dados**. 6. ed. Porto Alegre: Bookman, c2009. xii, 282 p. (Livros didáticos (Universidade Federal do Rio Grande do Sul); v. 4.

ORACLE. **Oracle SQL Developer Downloads**. Disponível em: <<http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html>>. Acesso em: 20 maio 2018.





## MINICURRÍCULO DO AUTOR

---

### DIEGO MARTINS POLLA DE MORAES

Diego Martins Polla de Moraes é especialista em Gerenciamento de Projetos de TI pela Universidade do Sul de Santa Catarina – Unisul desde 2012, e bacharel em Sistemas de Informação pela Escola Superior de Criciúma – ESUCRI, desde 2009. Atua na área de desenvolvimento de software desde 2003. É Analista de Sistemas da Indra Brasil, desenvolvendo softwares para a gestão pública. É docente na área de tecnologia da informação desde 2012, atuando em disciplinas de Modelagem de Dados, Sistemas de Gerenciamento de Banco de Dados, Engenharia de Software, Gestão de TI e Planejamento de Projetos. Atualmente é docente no Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas e Professor Orientador do MBA SMART em Gestão de Projetos Ágil do SENAI/SC.



# ÍNDICE

---

## A

Abstração de dados, 9, 15, 22, 27, 38

Arquitetura, 5, 9, 20, 21

Aspectos Organizacionais, 10, 83

Atributos, 5, 9, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 39, 40, 43, 44, 45, 47, 48, 51, 52, 58, 59

## B

Banco de dados, 1, 3, 5, 9, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 30, 31, 33, 35, 37, 38, 46, 47, 52, 53, 54, 55, 56, 57, 58, 59, 61, 62, 64, 65, 66, 68, 69, 71, 73, 75, 76, 77, 78, 79, 80, 86, 87, 93, 95

brModelo, 5, 9, 37, 38, 39, 40, 41, 42

## D

Data Control Language, 9, 57, 79

Data Manipulation Language, 9, 57, 73

Data Query Language, 9, 57, 66

Data Transaction Language, 9, 57, 77

Diretrizes empresariais, 10, 83, 88

## E

Entidades, 5, 9, 20, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 38, 39, 41, 42, 43, 45, 49, 51, 52

## F

Ferramentas, 9, 13, 35, 37, 38, 49, 53, 54, 55, 56, 58, 65, 66, 74, 80, 88

## I

Identificadores, 9, 25, 29, 31, 43

Informática, 10, 13, 83, 84

Integridade referencial, 9, 32, 33, 34

## L

Linguagem SQL, 6, 9, 55, 57, 58, 67, 68, 69, 75, 79

## **M**

Missão, 10, 88, 89

Modelagem de dados, 9, 15, 20, 21, 22, 24, 30, 35, 40, 43, 52, 55, 58, 95

Modelo conceitual, 5, 9, 15, 21, 22, 24, 27, 29, 30, 31, 32, 33, 35, 37, 38, 41, 42, 43, 44, 45, 51

Modelo físico, 9, 21, 30, 35, 53

Modelo lógico, 5, 9, 21, 23, 30, 31, 32, 33, 44, 45, 46, 49, 51, 53, 56

Política de qualidade, 10, 89, 90

Relacionamentos, 5, 9, 24, 25, 27, 28, 29, 30, 31, 34, 41, 43, 49, 50, 51, 58, 71

**SENAI – DEPARTAMENTO NACIONAL  
UNIDADE DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA – UNIEP**

*Felipe Esteves Morgado*  
Gerente-Executivo

*Luiz Eduardo Leão*  
Gerente de Tecnologias Educacionais

*Catarina Gama Catão*  
Coordenação Geral do Desenvolvimento dos Livros Didáticos

**SENAI – DEPARTAMENTO REGIONAL DE SANTA CATARINA**

*Fabrizio Machado Pereira*  
Diretor do SENAI/SC

*Claudemir José Bonatto*  
Diretor de Educação do SENAI/SC

*Daniel Thiesen Horongoso*  
Gerente Executivo da Regional Sudeste SENAI/SC

*Ricardo Maximo Anzolin*  
Gerente de Operações da Regional Sudeste SENAI/SC

*Priscila Carneiro Gallasse Cesconetto*  
Supervisora de Consultoria em Educação

*Diego Martins Polla de Moraes*  
Elaboração

*Felipe Demarchi*  
Revisão técnica

*Michele Antunes Corrêa*  
*Rosecler Fernandes*  
Design educacional

*Carlos André Marques de Andrade*  
*Davi Leon Dias*  
Ilustrações e tratamento de imagens

*IStock*  
*SENAI/SC*  
Banco de imagens

*Tatiana Daou Segalin*  
Diagramação

*Tatiana Daou Segalin*  
Revisão e Fechamento de Arquivos

*Luciana Effting Takiuchi*  
*CRB – 14/937*  
Ficha Catalográfica

---

*i-Comunicação*  
Projeto Gráfico

*Editorar Multimídia Ltda.*  
Revisão Ortográfica e Gramatical

*Editorar Multimídia Ltda.*  
Normalização





*Iniciativa da CNI - Confederação  
Nacional da Indústria*

