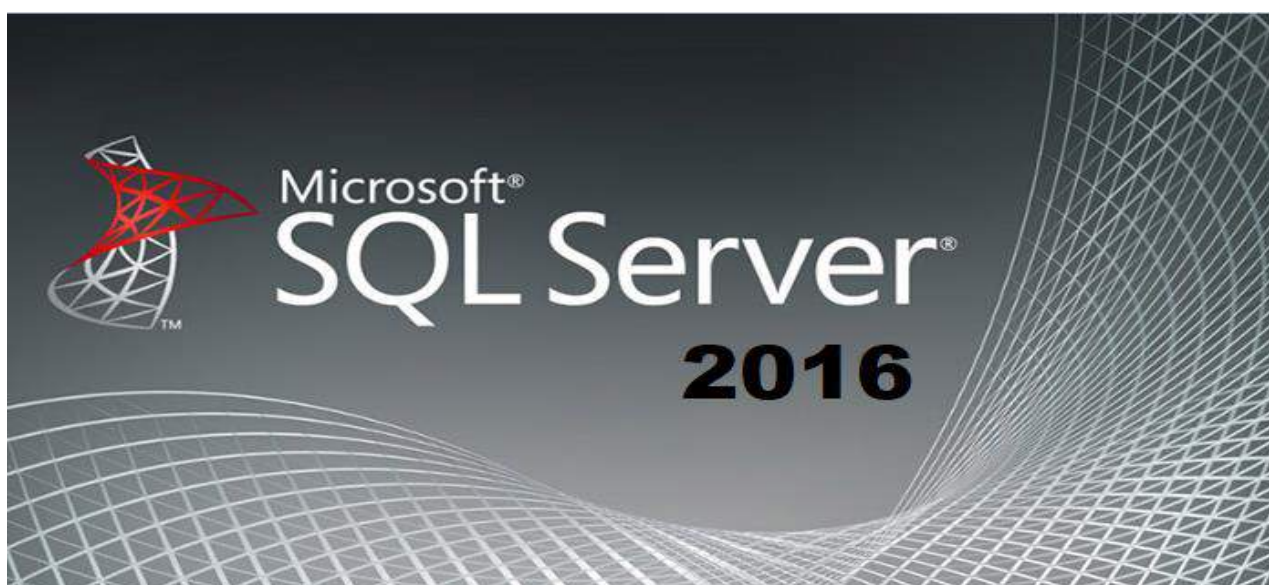


**APOSTILA: modelagem, projeto e gerenciamento de
Banco de Dados utilizando a linguagem SQL.
Projetos práticos de Banco de Dados
Implementados no SQL Server 2012/2014/2016**

**Autor e docente: Francisco Antonio de Almeida
Prof. Xyko**



Material recomendado para: Cursos técnicos e tecnológicos de informática/computação, área e concursos da tecnologia da informação.

Notas de Aulas das disciplinas ministradas nas Instituições de Ensino: IFSP-Brt e Fatec Fio Preto - Junho/2018

1) Fundamentos e Conceitos Básicos de Banco de Dados

Esta apostila é o resultado das Notas de Aulas das disciplinas: Banco de Dados, Administração de Banco de Dados, Modelagem de Banco de Dados e Tecnologias de Banco de Dados ministradas pelo **Prof. Francisco Antonio de Almeida (Xyko)** nas instituições de ensino: **Fatec de Rio Preto -SP** e no **IFSP-Brt** de 2010 a 2017.

Inicialmente serão apresentadas as principais definições (conceitos) dos Fundamentos de Banco Dados que é base para todo aprendizado na fascinante teoria e prática de Banco de Dados.

No Passado as Empresas armazenavam seus dados em grandes armários de aço com inúmeras pastas suspensas e ordenadas na sequência alfabética. A análise, o estudo e a manipulação destes dados eram demorados e sujeita a muitos erros – dependia apenas da habilidade do usuário.

Nos dias atuais, trabalhar com Base de Dados (Banco de Dados) pode ser associado à mesma ideia do passado, mas o conceito de **Armazenar Dados** evoluiu para o **Armazenamento Digital dos dados** com o surgimento dos Sistemas Gerenciadores de Banco de Dados (SGBD). Atualmente os Dados das Empresas são armazenados em Bases de Dados e gerenciados por **SGBDs** operados nos Computadores ligados às redes de computadores. Este processo aumentou a Rapidez do Processamento e reduziu a quantidade de erros na manipulação dos Dados das Empresas.

Com os **Dados Armazenados digitalmente** nos Computadores é possível fazer uma série de análises, comparações, simulações, debates, estudos e desta forma contribuir para as tomadas das decisões de Negócios pelas Empresas. SGBD nas empresas é um diferencial de competitividade.

Para que os computadores possam fazer as simulações e comparações com as Bases de Dados é preciso ter instalado um software especializado em Gerenciar grandes Bases de Dados que são os SGBDs.

1.1 - Vantagens e Benefícios dos Banco de Dados.

As principais vantagens dos Banco de Dados Gerenciados pelos SGBDs são:

- Os Dados podem ser compartilhados entre várias aplicações. Todos os Sistemas trabalham com a mesma Base de dados (o mesmo dado);
- Exemplo de trabalhar com o mesmo dado: Com um Banco de Dados o Banco Comercial Xyko S/A cadastrada o Cliente preferencial Xyko uma única vez e todos os Produtos (subsistemas) usam os mesmos dados – Conta Corrente, Poupança, Seguro, Empréstimos, etc;
- Controlar e Reduzir a Redundância dos Dados armazenados. O SGBD possui Mecanismos para gerenciar a redundância dos Dados do Banco de Dados.
- Inconsistência pode ser evitada (até certo ponto) – é possível impor restrições aos dados a serem armazenados;
- Suporte a transações pode ser fornecido – ou “tudo acontece” e grava ou “nada acontece” e não grava;

Projeto, Modelagem e Gerenciamento de Banco de Dados.

Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

- Integridade pode ser mantida – através dos relacionamentos entre as tabelas entre as chaves primárias e chaves estrangeiras. Dados integrados – todos os sistemas acessam a mesma base de dados.
- Segurança pode ser reforçada – podemos criar uma política de usuários com permissões controladas – cada usuário só acessa parte dos dados necessários ao seu trabalho;
- Padrões podem ser reforçados – neste material usamos fortemente a padronização.
- Políticas de Backups e Restauração são facilitadas com uso do SGBD para gerenciamento dos Dados armazenados nos Banco de Dados;
- Com uso das Redes de Computadores, os dados podem ser acessados a distância;
- Os Banco de Dados dão suporte a aplicações denominadas *Business Intelligent* – Estratégias de Negócios.
- **Resumindo** com Banco de Dados Gerenciado por um SGBD aumenta a Confiabilidade, Integridade, Disponibilidade e o cruzamento dos Dados.

Lembrar que: a Base de Dados é o maior patrimônio das Empresas, logo é preciso “armazenar de forma correta” estes Dados que serão a Base para as tomadas de decisões de Negócio.

Na figura 1 – é mostrado um banco de dados dando suporte á vários sistemas de informática – temos apenas uma única base de dados.

Processamento de dados com Banco de Dados

- ❑ Cada informação armazenada uma única vez
 - eventual redundância controlada pelo sistema em computador e invisível ao usuário

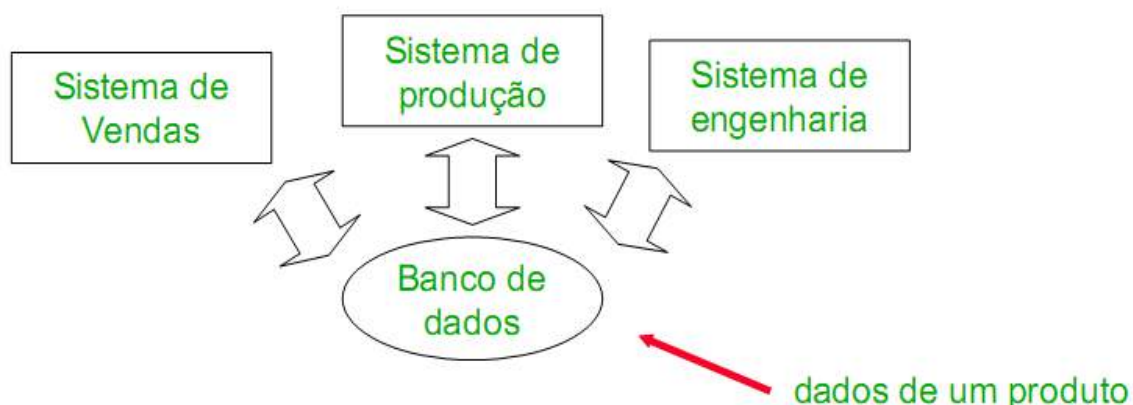


Figura 1: Banco de Dados dando suporte a vários sistemas informatizados [Euser, 2009]

Observe na Figura 1 que uma única de base de dados dá suporte para todos os sistemas. Neste caso o dado é único, ou seja, é o mesmo para todos os sistemas. Quando os dados de um produto são atualizados – todos os sistemas consultam o “**mesmo**” dado.

1.2- Desvantagens e Dificuldades dos Sistemas Isolados.

Sistemas computacionais isolados apresentam várias dificuldades (desvantagens):

- Dados redundantes – repetidos em vários sistemas;
- Dados não compartilhados – os sistemas não compartilham dados entre si. Cada sistema tem sua base de dados;
- Muitas inconsistências – O mesmo dado gravado de forma diferente nos vários sistemas (um endereço de um cliente pode ter sido atualizado num sistema e não ter sido atualizado nos demais);
- Difícil de impor e manter segurança e padronização – os Dados estão em vários Banco de Dados dos Sistemas;
- Difícil manter integridade dos dados – os dados não são totalmente confiáveis;
- Difícil compartilhar os dados entre os sistemas isolados;
- Políticas me Backups precisam ser individualizada para cada sistema isolado;

Na figura 2 é apresentada uma situação onde vários sistemas isolados acessam de forma individual sua base de dados.

Processamento de dados sem Banco de Dados

- ❑ Dados de diferentes aplicações não estão integrados
- ❑ Dados estão projetados para atender uma aplicação específica

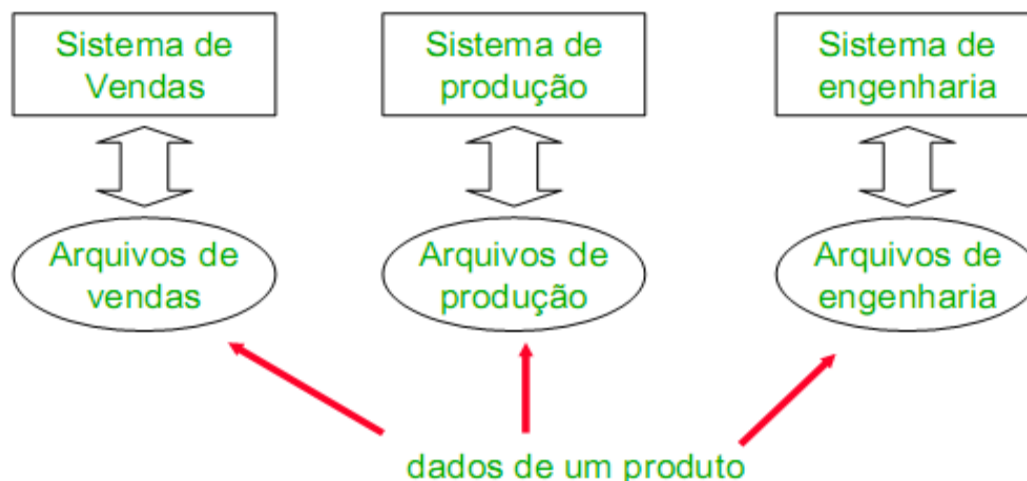


Figura 2: Processamento de Dados em Sistemas independentes sem uma Base de Dados [Euser, 2009]

1.2.1) Principais problemas com os sistemas isolados:

- Redundância dos dados – dados de um único cliente pode ser cadastrado várias vezes nos vários sistemas. A Redundância acarreta: Retrabalho com digitação; Erros de digitação; Dificuldades na Extração dos Dados.

Projeto, Modelagem e Gerenciamento de Banco de Dados.

Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

- Falta de Integridade – pode ocorrer de atualizar os dados de um Sistema e não atualizar os mesmos dados nos demais sistemas;
- Dados não integrados – os Sistemas não comunicam uns com os outros – são Sistemas Isolados; Dificuldade de Cruzar dados entre os vários Sistemas. A consistência dos Dados é Difícil – fica difícil saber qual dado é o correto.
- Dados não confiáveis. Cada Sistema pode ter dados diferentes de Clientes, Produtos, produção Vendas, etc. Decisões podem ser tomadas com base em dados errados.

Fazer a Gestão, administração destes sistemas “**sem uma Base de Dados**” é extremamente precário e ineficiente para Empresas modernas onde a Competição é um fator decisivo de permanência no mercado.

Atualmente Computadores, Bases de Dados e SGBDs são ferramentas que acrescentam um diferencial competitivo e indispensável para as Empresas.

1.3 Aplicação acessa o SGBD que gerencia os Banco de Dados.

Nas empresas informatizadas existem vários Sistemas de Informação que são compostos de vários aplicativos (programas) que são desenvolvidos utilizando várias linguagens de Programação tais como: Delphi, Java, Visual Basic, C#, dentre outras. Quando criamos um Aplicativo, programamos o Aplicativo para Acessar o SGBD que utiliza a Linguagem SQL para acessar a(s) Base de Dado(s) conforme descrito na Figura 3.

Sistema de Gerência de Banco de Dados (SGBD)

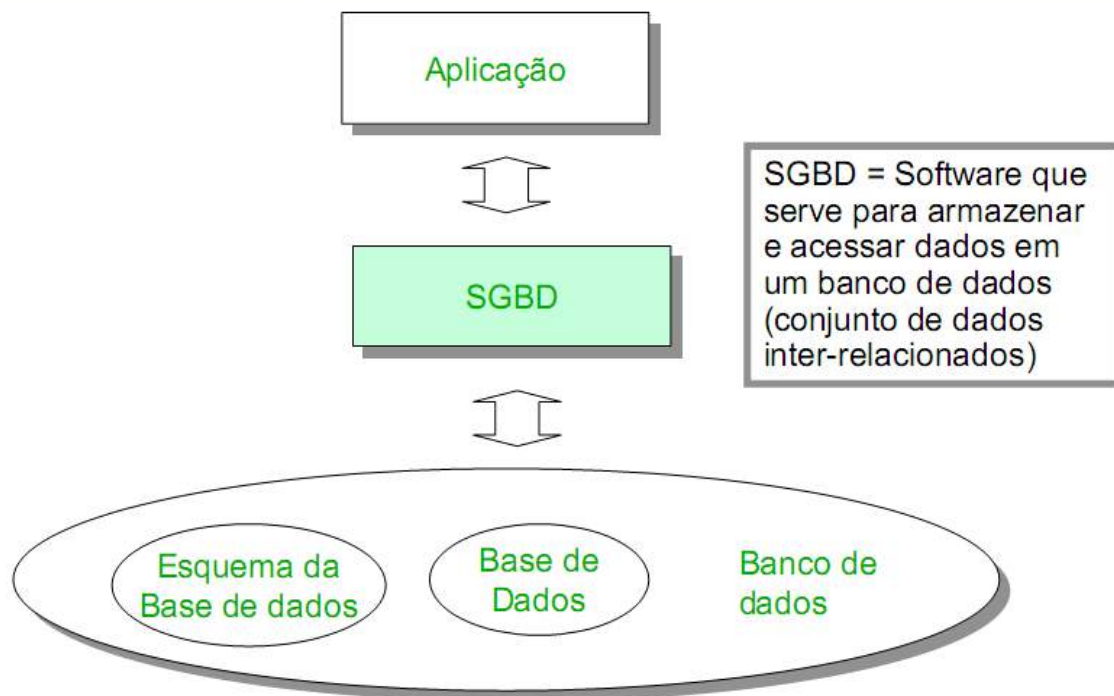


Figura 3: Acesso dos Aplicativos ao SGBD que gerencia várias Bases de DADOS [Euser, 2009]

Projeto, Modelagem e Gerenciamento de Banco de Dados.

Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

Quando vamos ao supermercado e fazemos uma compra, o caixa do supermercado está operando um terminal de computar em rede que possui um Aplicativo que acessa o SGBD que contém o Banco de Dados dos Produtos do Supermercado. Este processo permite totalizar nossa compra, baixar o estoque do supermercado e relacionar todos os produtos comprados num cupom fiscal – tudo transparente ao usuário.

Quando pagamos esta mesma compra com cartão de crédito, o caixa do supermercado utiliza outro aplicativo que via rede acessa a operadora do Cartão de Crédito e é efetuada a transação – Faz um lançamento no nosso cartão de crédito e um lançamento na Conta do Supermercado – mais uma vez transparente ao usuário. Tudo utilizando Banco de Dados conectados por redes de computadores.

Os conhecimentos necessários para desenvolver os aplicativos (programas) usando as linguagens de programação são responsabilidade das disciplinas de Programação que não são abordados neste material.

Na disciplina de Gerenciamento de Banco de Dados (GBD), ou Modelagem de Banco de Banco de Dados (MDB), ou Tecnologia de banco de dados (TBD) **não vamos desenvolver um aplicativo** que acessa o SGBD que gerencia o Banco de Dados. Na disciplina GBD vamos trabalhar diretamente com o SGBD (que tem uma interface para interação com o usuário) que é a ferramenta para Criar e Gerenciar o Banco de Dados.

Na disciplina de **Gerenciamento de Banco de Dados (GBD), ou Modelagem e Projeto de Banco de Dados (MBD) ou Tecnologias de Banco de Dados (TBD) ou Banco de Dados I e II** o aluno vai aprender a:

- Projetar o Banco de dados (fazer a Modelagem) com segurança;
- Criar uma Base Dados com Integridade;
- Dar Manutenção no Banco de Dados utilizando a Ferramenta SQL com o SGBD SQL Server 2008 ou 2012 ou 2014 ou 2016;
- Utilizar a Linguagem dos SGBD o SQL para: criar BD, Criar as Tabelas, cadastrar os Dados, fazer consultas dos dados armazenados, fazer Back-up, restaurar back-ups, criar usuários, etc;
- Configurar o ambiente do SGBD para administrar diversas Base de Dados;
- Projetar, implementar, e gerenciar Banco de Dados Relacional com a linguagem dos SGBDs SQL

1.4) Conceitos (definições) importantes sobre Banco de Dados:

- a) **Dado:** Valor de um campo armazenado, matéria-prima para obtenção de informação. Computador sempre processa Dados.
- b) **Informação:** Dados processados de acordo com solicitação de consultas e análises. Após o Processamento dos Dados pelo computador, os Seres Humanos fazem uma análise, um estudo, dão uma interpretação aos Dados e estes se tornam para nós Informação. Informação é o significado, a interpretação dos Dados para as Pessoas.
- c) **Processamento de Dados:** É o processo que ocorre quando introduzimos Dados nos Computadores através dos aplicativos para serem organizados ou reorganizados para um propósito – geralmente uma tomada de decisão.

- d) **Banco de Dados:** Conjunto de dados inter-relacionados que objetivam atender as necessidades de um conjunto de usuários (Empresa). É uma grande Base de Dados de uma Empresa. Um Banco de Dados representa algum aspecto do mundo real, algumas vezes chamado de "mini-mundo". Exemplo: para dar suporte a Gestão escolar é preciso ter um Banco de Dados que pode ser chamado BDAcademico.
- e) **Sistema Gerenciador de Banco de Dados (SGBD):** É um conjunto de Softwares que auxilia na, criação e manutenção de um Banco de Dados. Como a quantidade de Dados de um Banco de Dados é enorme, é preciso de um Software especializado em Gerenciar Bancos de Dados. **Exemplos** de SGBD: Oracle, SQL Server 2008, MYSQL, PostGreSQL, DB2, dentre outros.
- f) **Minimundo:** faz parte do mundo real. É o local que está inserida a Empresa que necessita de uma solução de Armazenamento de Dados (um Banco de Dados). Exemplo um Banco de Dados para uma Universidade que faz parte do mundo real. Neste exemplo o Minimundo é a Universidade (vamos propor uma solução para armazenar os dados da Universidade em uma Base de Dados).
- g) **DBA** – Analista de Banco de Dados – Profissional que projeta, cria e Gerencia Banco de Dados.
- h) **Modelagem de Dados:** descrição narrativa e formal do Projeto de um Banco de Dados – Para criar um Banco de Dados, primeiro criamos um Modelo que representa os dados deste Banco. Depois com uso do SGBD criamos fisicamente o Banco de Dados.
- i) **Bando Dados Relacional** – Neste contexto as Entidades serão as Tabelas (ou Relações da Matemática) do Banco de Dados com Colunas (atributos) e Linhas (Registros). Vamos trabalhar com Banco de Dados Relacional neste material.

1.5 - Etapas para Criar o Projeto do Banco de Dados.

O **Desenvolvimento de Sistemas** (aplicativos) segue algumas etapas pré-definidas tais como: Análise dos Dados, Projeto do Sistema, Desenvolvimento do Sistema (codificação), Testes, Implantação do Sistema (colocar em operação) e Manutenção (dar suporte ao Sistema). Portanto quando foi Desenvolvido o Software (aplicativo) do Supermercado foi seguido basicamente as etapas especificada acima.

Da mesma forma na etapa de **projetar o Banco de Dados**, é preciso seguir algumas Etapas bem definidas, tais como: **Projeto (Modelo) Conceitual, Projeto (Modelo) Lógico e Projeto (Modelo) Físico**, conforme especificado na Figura 4.

1.5.1) O que deve ser feito no Projeto (Modelo) conceitual.

Todo Projeto de Banco de Dados deve começar pelo Projeto (Modelo) Conceitual dos Dados do Banco de Dados a ser Criado (vide Figura 4). Mas o que realmente um DBA deve fazer nesta Etapa?

A seguir alguns Procedimentos para melhor entender o Modelo Conceitual:

- **Fazer o Levantamento e Análise de Requisitos:** O objetivo desta etapa é identificar a realidade a ser modelada, ou seja, o mundo o qual se deseja modelar (Mini-mundo), em detalhes suficientes para a construção do modelo do banco de dados em questão – Aqui vamos criar um Protótipo do Banco de Dados.

Projeto, Modelagem e Gerenciamento de Banco de Dados.

Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

- **Fazer uma Descrição mais Abstrata da base de Dados** – o DBA vai até Empresa, observa como o Negócio da Empresa funciona e precisa criar um Modelo para o Banco de Dados que simule o Negócio da Empresa. Esta fase apesar de parecer simples é uma das mais complexas do Banco de Dados e deve ser treinada.
- Ter cuidado com conteúdo transferido entre o cliente e o analista, que por ser muito grande, está sujeito a interpretações errôneas.
- **Não contém detalhes de Implementação** – aqui pensamos em criar uma solução de Armazenamento de Dados para a empresa.
- **É Independente de tipo de SGBD usado** – A solução de armazenamento de dados deve ser implementada em qualquer SGBD relacional a ser definido posteriormente.
- **É Ponto de partida do projeto da base de dados.**
- A ideia é representar a realidade através de uma visão global e genérica dos dados e seus relacionamentos.
- Criar o MER (Modelo Entidade Relacionamento) ou também chamado DER (Diagrama Entidade Relacionamento - DER).
- **O MER é composto basicamente de: Entidades, os Atributos, os Relacionamentos e as Cardinalidades.**
- Resumindo: No Modelo Conceitual tem que identificar e especificar: Entidades, Atributos, Relacionamentos e as Cardinalidades, Criar o DER/MER para visualizar o projeto do BD.
- Aqui o **DESAFIO** é Criar um Ambiente Automatizado para armazenar a Base de Dados da Empresa.

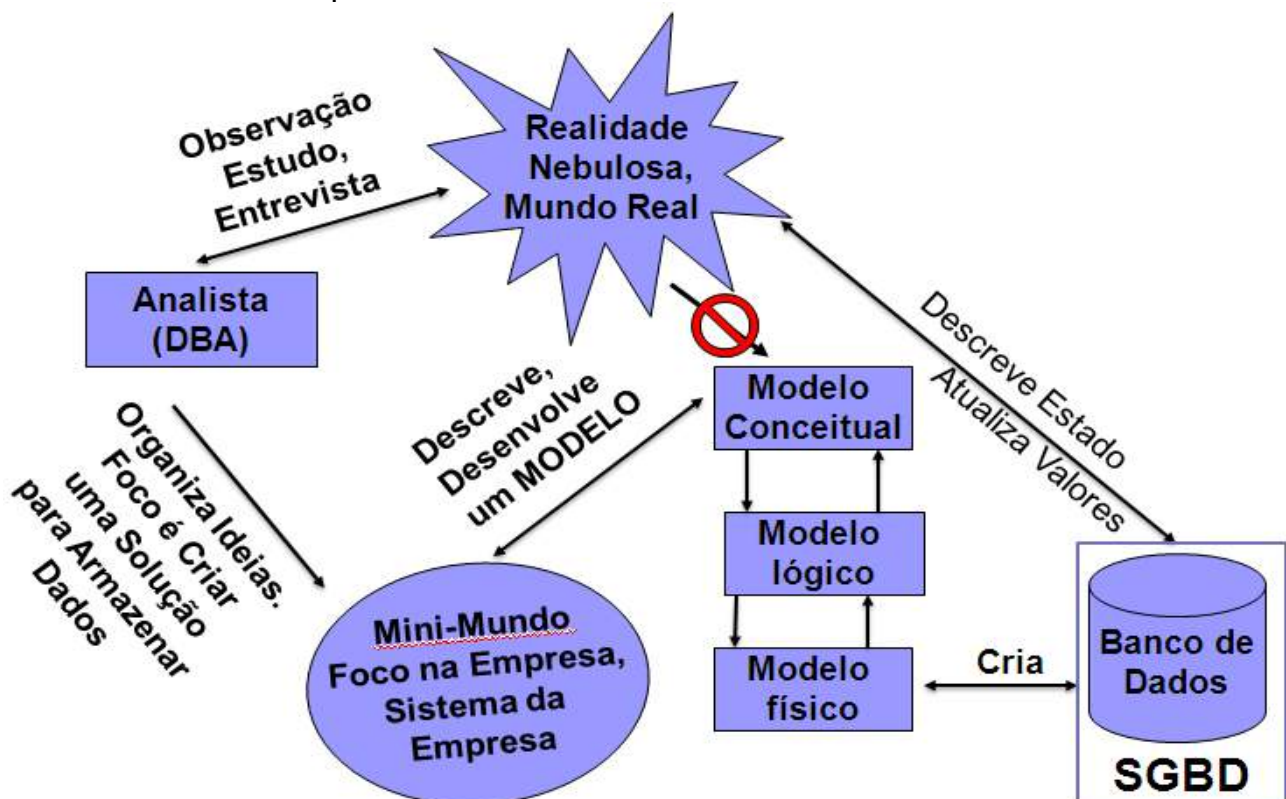


Figura 4: Etapas necessárias para criar um Banco de Dados – Adaptado de (Euser, 2009)

1.5.1.1 - Identificar as Entidades – (O que são Entidades?)

- Entidades são Objetos Distintos do Mini-Mundo, pode ser Real (uma Pessoa) ou abstrato (um Departamento), e que estamos interessados armazenar dados sobre ele.
- **Exemplos:** Pessoa, Carro, Produto, Aluno, Professor, Cliente, Fornecedor, Paciente, Médico, Avião Voo, Aeroporto, Consulta, Pedido, Nota Fiscal, ... (depende do Mini-Mundo que é o Foco).
- **Exemplo de Entidades de um Sistema Acadêmico:** Estado, Cidade, Aluno, Matrícula, Professor, Curso, Disciplina, Prédio, Sala, Histórico, dentre outras.
- **Exemplo de Entidades de uma Clínica Médica:** Médico, Paciente, Consulta, Funcionário, Especialidade, Convenio Médico, Receita, Medicamento, Exame, dentre outras.
- Para Definir as **Entidades** devemos definir o **Mini-Mundo**, como por exemplo: Clínica Médica, Sistema Acadêmica, Controle de Pedidos, Controle de Locação de Carros, Controle de Locação de Imóveis, dentre outros. Vide Figura 5.
- Entidade: é um **conjunto de objetos** do mundo real sobre os quais se deseja manter Dados no Banco de Dados.
- Entidade é distinguível de outros Objetos por seus Atributos.
- Entidade é Representada através de um **retângulo** com Nome da Entidade.
- **Lembrar:** Entidades fazem parte do Mundo Real como Pessoas, Carros, Escolas, Clientes, Compras, Vendas, Produtos, etc – Lembrar o Mini-Mundo faz parte do Mundo Real.



Figura 5: As Entidades: Médico, Paciente e Consulta de Clínica Médica ou Hospital. As entidades estão presentes no mundo Real e no Mini Mundo

Analisando a Figura 5, é possível perceber que as Entidades Possuem duas Características Especiais – **Atributos e Relacionamentos:**

- **Atributos** são características das Entidades.
 - ✓ Médicos têm como Atributos: CRM, nome, Especialidade, sexo, endereço, Data de Nascimento, salário, telefone, endereço, etc;
 - ✓ Paciente tem como Atributos: Nome, sexo, Data de Nascimento, Endereço, Cpf, RG, dentre outros.
 - ✓ Consulta tem como Atributo: CRM Médico, Data Consulta, Código Paciente, Hora da Consulta, Valor consulta, dentre outras.



Projeto, Modelagem e Gerenciamento de Banco de Dados.

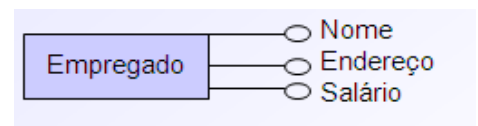
Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

- **Relacionamento** é a Associação que existe entre as Entidades – Entidades do mundo real se relacionam entre si. **Exemplos:** Cliente compra Produto. Professor Ministra Aula. Curso Possui Disciplina. Aluno se Matricula em Curso. E assim por diante.

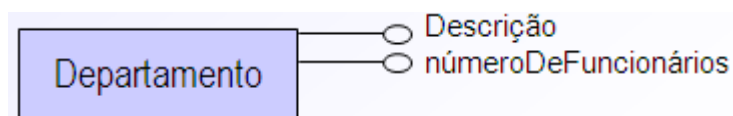
✓ **As Entidades vão ser as tabelas do Banco de Dados no SGBDR.**

1.5.1.2 – Identificar os Atributos (O que são Atributos?).

- É um Dado que é associado a cada ocorrência de uma Entidade ou de um Relacionamento.
- São Características Qualitativa e/ou Quantitativa das Entidades.
- As Entidades são descritas por seus Atributos.
- Exemplo dos Atributos da Entidade Empregado (nome, endereço, salário).



- Exemplo de Atributos da Entidade Departamento (Descrição, númeroDeFuncionários).



1.5.1.3 - Identificar os Relacionamentos (O que são Relacionamentos?).

- É uma associação entre Entidades.
- Representado através de um losângulo e linhas que ligam as Entidades relacionadas.
- Observar que as entidades do mundo real se relacionam constantemente.

Na Figura 6 temos a representação do Relacionamento entre as Entidades Empregado e Departamento. Observe que a Entidade Empregado é Representada por: Karlos, Xyko, Júlio e Thales (chamados Instâncias). A Entidade Departamento é Representada por: Informática, RH, Segurança e Vendas (Instâncias). Notar que cada Instância de cada Entidade só se relaciona com uma única Instância de Departamento – Relacionamento 1:1 (grau do relacionamento).

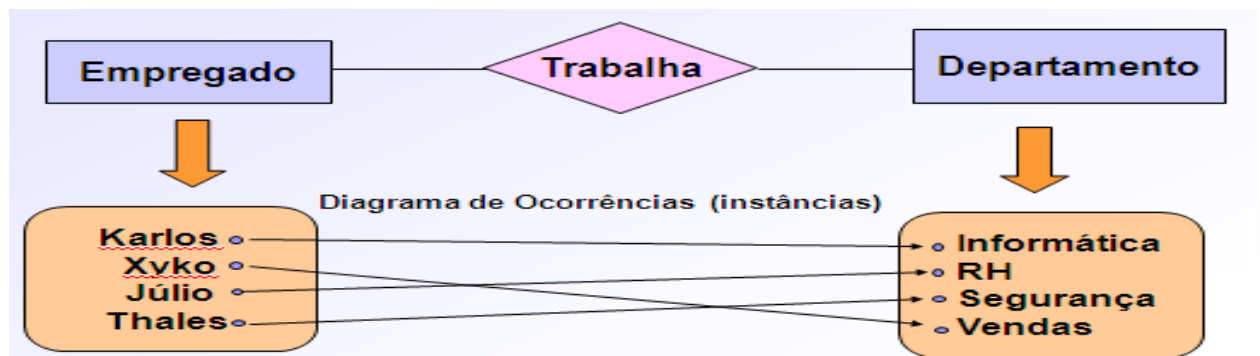


Figura 6: Modelo Conceitual do Relacionamento 1:1 entre as Entidades: Empregado e Departamento.

Na Figura 7 é apresentado um exemplo do Relacionamento entre as Entidades Aluno e Curso. Neste caso temos duas entidades e é chamado relacionamento binário. Este exemplo ocorre no Mundo Real e no Min-Mundo de Uma Escola, Faculdade ou de uma Universidade. O

Projeto, Modelagem e Gerenciamento de Banco de Dados.

Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

relacionamento entre Cliente e Produto e Entre Cliente e Cidade ocorrem no Mundo real e em Mini-Mundo como: Controle Vendas ou Sistemas de Pedidos.

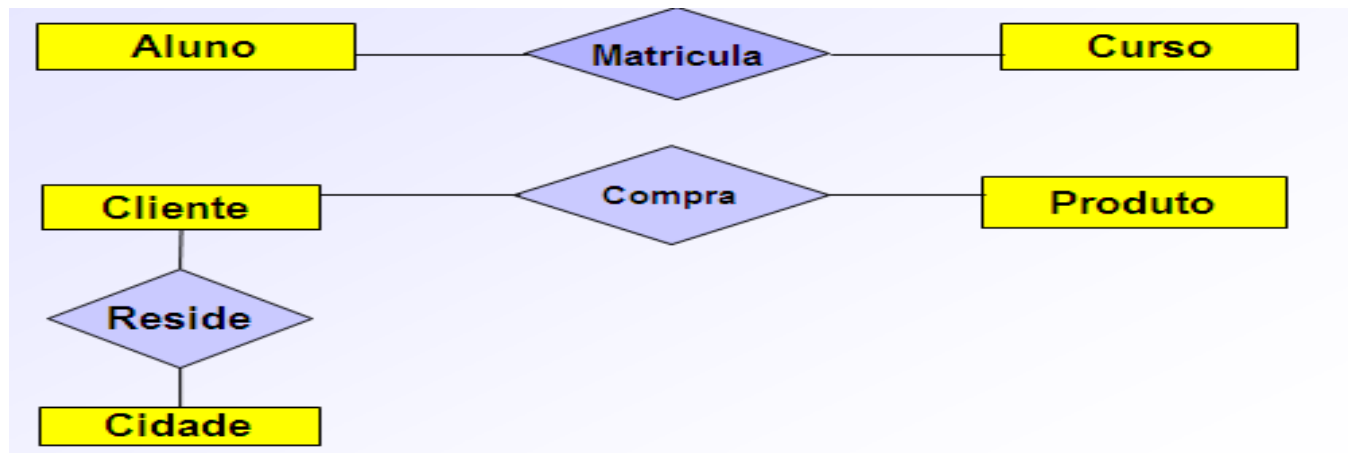


Figura 7: Representação do Grau de Relacionamento Binário entre as Entidades Aluno e Curso e o Grau de Relacionamento Ternário entre as Entidades: Cliente e Compra e Cliente e Cidade.

1.5.1.4 - Identificar as Cardinalidades (Grau do Relacionamento) – Quantidade de Ocorrência entre as Entidades.

O relacionamento entre duas Entidades pode ter Cardinalidade (Grau de Relacionamento) que depende das Regras de Negócio (contidas no contexto do Mini-Mundo). Grau de Relacionamento podem ser do tipo:

- Um-para-um ou (1:1).
- Um-para-muitos ou (1:N)
- Muitos-para-muitos ou (N:N).

a) **Exemplo da aplicação de Cardinalidade ou Grau do Relacionamento N:N.** Na Figura 8 tem um Relacionamento N:N – Muitos para Muitos. Observe que Consulta tem que identificar um Veterinário e um animal mais a data da consulta. A consulta só ocorre por ter um Veterinário e um Animal.

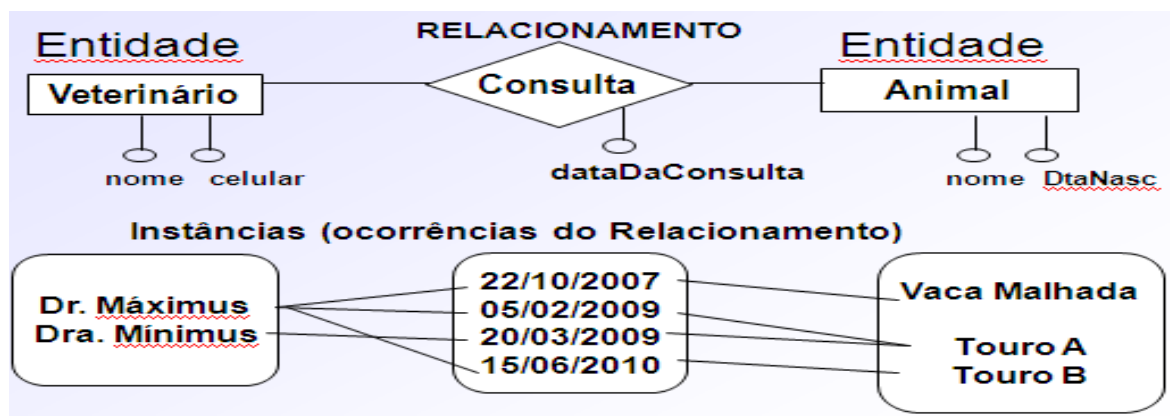


Figura 8: Um Relacionamento Muitos para Muitos (N:N).

Projeto, Modelagem e Gerenciamento de Banco de Dados.

Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

Observações Importantes da Figura 8:

- Em todo Relacionamento N:N o relacionamento tem atributos (No exemplo acima DataDaConsulta).
- O Dr. Máximus consultou três Animais (muitos).
- O Touro A teve consulta com dois Veterinários (muitos).
- Todo Relacionamentos N:N será desmembrado em dois Relacionamentos 1:N (um pra muitos).
- Em relacionamentos (N:N) sempre ocorre: Uma ocorrência de Entidade **A** está associada a **qualquer** número de ocorrências de Entidade **B**, e uma ocorrência em **B** está associada a **qualquer número** de ocorrências em **A**.
- É muito comum relacionamento N:N nos projetos de Banco de Dados.

b) Exemplo de Relacionamento Um pra Um (1:1). Uma ocorrência da Entidade **A** está associada a no **máximo uma** ocorrência da Entidade **B**, e uma ocorrência em **B** está associada a no **máximo uma** ocorrência em **A**.

A Figura 9 mostra o Relacionamento 1:1 (um pra um) entre as Entidades Empregado e Departamento.

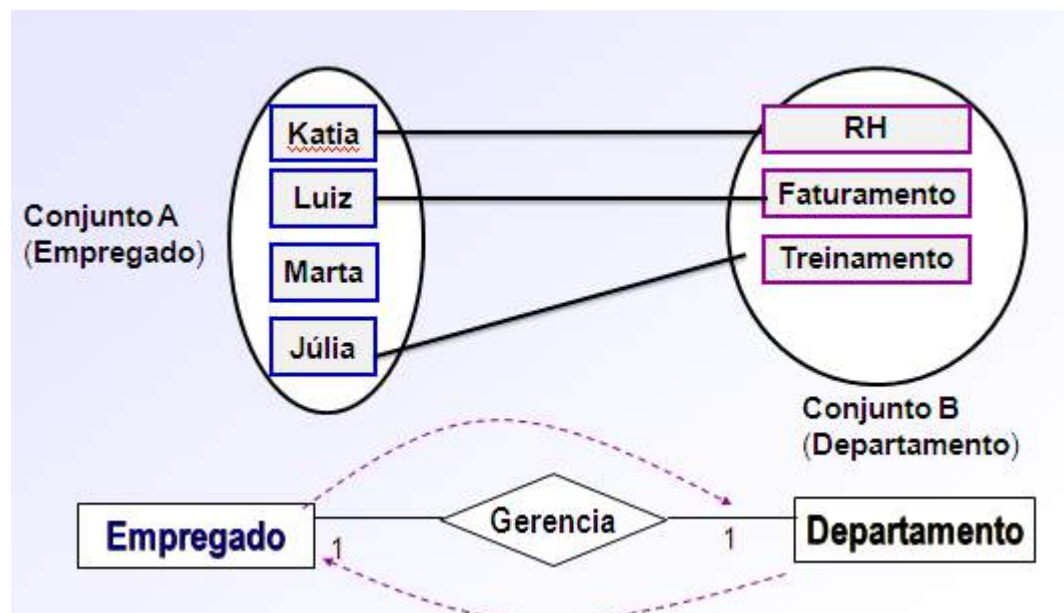


Figura 9: Representação de um Relacionamento com Cardinalidade Um pra Um (1:1).

Observações Importantes:

- A cardinalidade sempre depende da Regra de Negócio (do contexto do Mini-Mundo).
- Pode ocorrer de algum Empregado não Gerenciar Departamento – no exemplo Maria não gerencia Nenhum Departamento.
- Pode ocorrer de algum Departamento não ser gerenciado por alguma Pessoa.
- Quando ocorrer um Relacionamento entre Empregado e Departamento é sempre com cardinalidade 1:1 – neste exemplo (é a regra de negócio).
- Pergunte a seu Professor sobre mais exemplos de Relacionamento 1:1.

c) Exemplo de Relacionamento com Cardinalidade (1:N) – Um pra Muitos. Neste caso Mudou a regra de Negócio. Uma ocorrência da Entidade **A** está associada a várias ocorrências da Entidade **B**, porém uma ocorrência de **B** deve estar associada a no máximo uma ocorrência em **A**.

- Note que aqui está sendo usado exemplo quase semelhante ao da Figura 9 – ocorre que mudou a regra de Negócio – quem define é a empresa.
- Agora um Funcionário está lotado (alocado) a apenas um único Departamentos. Mas um Departamento pode ter vários Empregados. Vide Figura 10.
- Só é Possível saber quando um Relacionamento é (1:1), ou (1:N), ou (N:N) – fazendo uma visita na Empresa e fazendo a identificação das Regras de Negócio no Mini-Mundo.

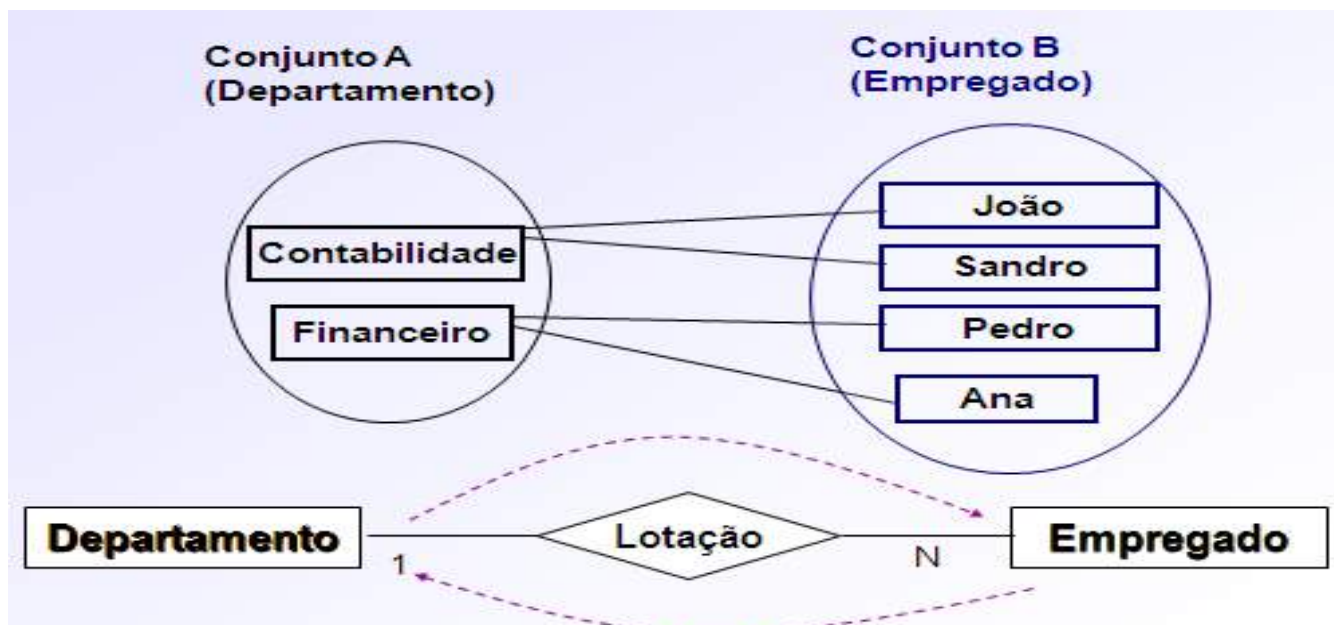


Figura 10: Representação de um Relacionamento com Cardinalidade Um pra Um (1:N).
Mudou a Regra de Negócio, muda a cardinalidade.

1.5.1.5- D.E.R (Diagrama Entidade Relacionamento ou M.E.R (Modelo Entidade Relacionamento)).

No Modelo Conceitual após identificar as Entidades, os Atributos, os Relacionamentos e definir o Grau dos Relacionamentos é preciso criar o DER (que é o Projeto inicial) do Banco de Dados. O DER é a metodologia mais usada no projeto de Banco de Dados Relacional.

d) Exemplo de um DER com: Entidades, Relacionamentos, Atributos e Cardinalidades. Aqui temos uma Visão geral de um Diagrama Entidade Relacionamento com Atributos e Cardinalidades. A Figura 11 mostra um DER de um Sistema Acadêmico com suas Entidades, Atributos, Relacionamentos, e Grau do Relacionamento.

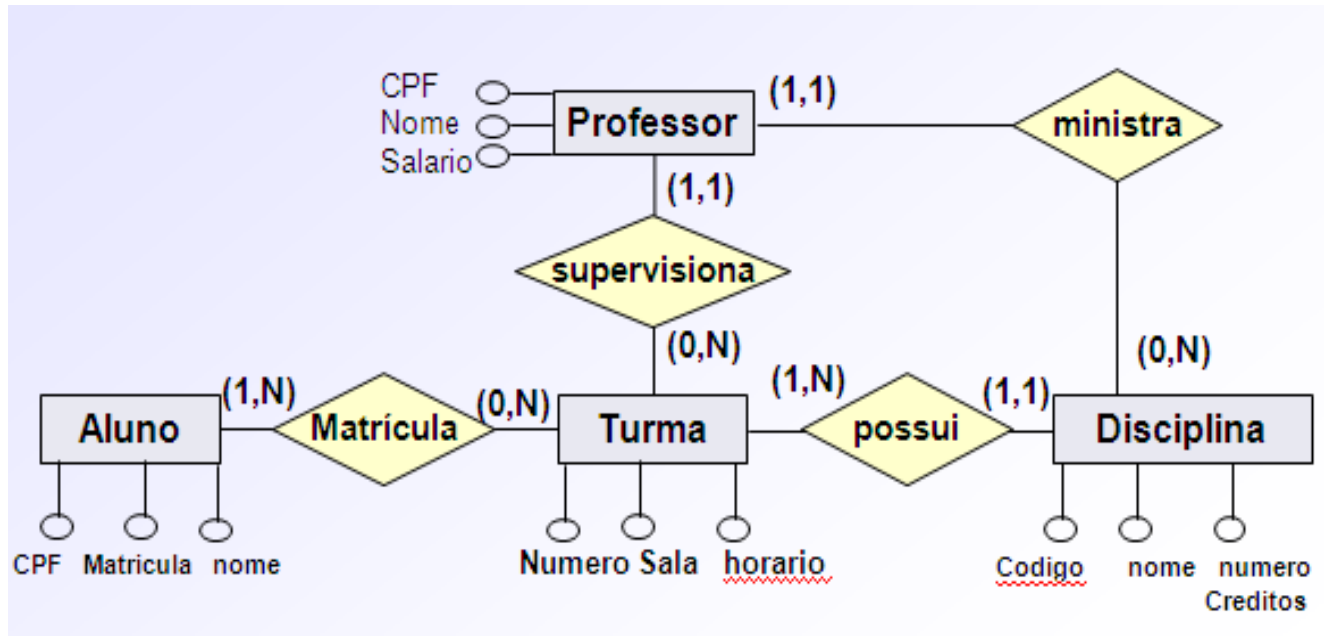


Figura 11: Visão geral de um DER com Entidades, Atributos, Relacionamentos e Cardinalidades.

1.5.1.6 – Componentes do DER/MER.

Os elementos Básicos constante em um DER são Entidades, Relacionamentos, Atributos e as Cardinalidades. Onde, Retângulo representa Entidade, Losângulo representa Relacionamento e Círculo representa os Atributos. Este tipo de Representação dificulta a inclusão dos atributos nos relacionamentos com muitos atributos. Vide Figura 12.



Figura 12: Elementos Básicos de um DER: Entidades, Atributos e Relacionamentos.

1.5.1.7 – Usando uma Tabela para melhor identificar Entidade e Atributos no Mini-Mundo.

Em um Projeto de Banco de Banco de Dados com muitas Entidades, atributos e Relacionamentos, a representação da Figura 12 pode poluir muito (dificultar o entendimento) o DER/MER. É preferível usar uma Tabela para melhor descrever as Entidades com seus atributos.

Projeto, Modelagem e Gerenciamento de Banco de Dados.

Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

Para facilitar o entendimento do DER usar uma Tabela que facilita a visualização das Entidades e Atributos. Vamos supor o Mini-Mundo Video-Locadora. PK é atributo candidatos a Chave Primária, FK é atributo candidato a chave estrangeira.

Entidade	Atributo	Observação
Cliente	IDcliente, nome, telefone, sexo, DataNascimento, Endereço, Cep, CodCidade,	IDCliente PK, CodCliente Fk
Filmes	IDFilme, DataLancamento, CodFabricante, Custo, Sinopse, CodCategoria, CodGenero	IDFilme PK, CodFabricante FK, CodGenero FK, CodCategoria FK
Genero	IDGenero, Descricao	IDGenero PK
Categoria	IDCategoria, Descricao	IDCategoria PK
Estado	IDUF, Nome	IDUF PK
Cidade	IDCidade, Nome, CodUF	IDCidade PK, CodUG FK
Funcionario	IDDuncionario, Nome, DataNsc, DataAdm, Sexo, Fone, Email, Endereco, Cep, CodCidade	IDFuncionario PK, CodCidade FK
Locacao	IDLocacao, CodCli, CodFunc, DataLocacao	IDLocacao PK, Codcli FK, CodFunc FK.
ItensLocados	CodLocacao, CodFilme, Quantidade, DataDevolucao	Codlocacao, codFilme PK, CodLocaca FK, codfilme FK

Tabela 1: Representação das Entidades com seus atributos do Mini-mundo Video Locadora.

Notas Importantes:

- **Chave Primária (PK)** é um Atributo que idêntica de forma única uma ocorrência na Entidade. Chave Primária não se repete e não pode ser nula.
- **Chave Estrangeira (FK)** é um Atributo para fazer Relacionamento entre duas Entidades. Exemplo na Entidade Cidade o atributo CodUF vem da Entidade Estado e Permite fazer o Relacionamento entre as Entidades Estado e Cidade.

1.5.2) O que deve ser feito no Projeto (Modelo) Lógico.

Como vamos implementar o Projeto de Banco de Dados em um **Modelo Relacional de Banco de Dados**, aqui as **Entidades são chamadas de Tabelas e os Atributos são chamados de Colunas**. O Modelo de Banco de Dados Relacional o objeto mais importante é Tabela que armazena os Dados em forma de Linhas e Colunas.

- O Projeto Lógico de banco de dados deve ocorrer após finalizar o Projeto Conceitual;
- Definir as Chaves Primárias (Primary Key) de cada Entidade. Por Exemplo: na Tabela TBEstado o campo IDUF é chave primária (PK);
- Definir as Chaves Estrangeiras (Foreign Key) de cada Entidade. Por Exemplo: na Tabela TBCidade o campo CodUF é chave estrangeira (FK);
- Definir para Cada Atributo (campo) de uma Entidade um **Tipo de Dados e um Tamanho para os Dados**: Exemplo:
 - ✓ Nome é **tipo Texto de 40 posições** e não pode ser nulo;
 - ✓ Salário é **tipo decimal com duas casas decimais** e não pode ter valor nulo;
 - ✓ Data-Admissao é tipo data (dd/mm/aaaa) e não pode ser nula;
 - ✓ Endereço é tipo Texto com 50 posições;

Projeto, Modelagem e Gerenciamento de Banco de Dados.

Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

- ✓ E-mail é tipo texto de 60 posições. E assim por diante.
- Tem que especificar as **Restrições para os Atributos (Campos)**. Exemplo de restrição: o atributo sexo só aceita M ou F. O atributo ano deve ser maior que 2012.
- Aqui é criado o Diagrama de Tabelas – Vide Figura 13.
- **Entidade deve ser entendida como Tabela** com Colunas e Linhas. As Colunas são os Atributos e as Linhas são os Registros (os dados a serem cadastrados nas Tabelas do Banco de Dados).
- **Todo Relacionamento N:N** deve ser desmembrado em dois Relacionamentos 1:N. No Banco de Dados Relacional não pode ser implementado um Relacionamento com Cardinalidade N:N – Muitos para Muitos;
- O Modelo Lógico pode ser descrito de forma **Textual** ou em forma de **Tabela**.
- Fazer o **Diagrama de Tabelas** com os Relacionamentos, Chaves Primárias e Chaves Estrangeiras – Tratar os Relacionamentos N:N.

Um **Exemplo de Modelo Lógico em Forma Textual** do Relacionamento N:N (Semelhante ao Relacionamento da Figura 8) entre as Entidades Aluno que cursa Disciplina que foi identificado no Modelo/Projeto Conceitual no Mini-Mundo Universidade UniXyko. Na Unixyko a **Regra de Negócio** é:

- Um Aluno pode cursar uma ou mais de uma Disciplina.
- Uma Disciplina pode ter um ou mais de um Aluno (Relacionamento N:N – Muitos para Muitos).

Assim o **Modelo Lógico** com as Entidades (Tabelas) e seus Atributos (campos) ficam:

Aluno (IDAluno, Nome, Sexo, DataNascimento, Telefone, Email).

Disciplina (IDDisciplina, Descricao, CargaHoraria).

Historico (CodAluno, CodDisciplina, Data, Nota, Situacao).

CodAluno referencia com a Entidade Aluno.

CodDisciplina referencia com a Entidade Disciplina.

Notas Importantes (da descrição em forma de Texto do Projeto Lógico):

- IDAluno sublinhado é o atributo (campo) **chave primária (PK)** da Entidade (Tabela) Aluno.
- IDDisciplina é sublinhado é o atributo (campo) **chave Primária (PK)** da Entidade (Tabela) Disciplina.
- A Entidade (Tabela) Historico foi criada para suportar o relacionamento N:N entre as Entidade Aluno e Disciplina.
- CodAluno se relaciona com a Entidade Aluno com o atributo IDaluno que é chave primária.
- CodDisciplina se relaciona com a Entidade Disciplina com o atributo IDDisciplina que é chave primária.
- Os atributos CodAluno, CodDisciplina da Entidade Historico é uma chave Primária Composta por dois campos.
- Os Atributos não tem acento, não tem cê-cedilha e não pode ser palavra composta.
- CodAluno faz Referência com a Entidade Aluno – CodAluno é Chave Estrangeira (FK).
- CodDisciplina faz referência (relaciona-se) com a Entidade Disciplina – CodDisciplina é Chave Estrangeira (FK).

Projeto, Modelagem e Gerenciamento de Banco de Dados.

Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

1.5.2.1 - Definindo os Tipos dos Atributos das Entidades (Tabelas) com: as Entidades, os Atributos, seus Tipos e Restrições.

Na Tabela 2 é descrito de forma muito bem detalhada todas as Entidades do Mini-mundo BDUiXyko em estudo com seus atributos, Tipo e Restrição de cada atributo.

Entidade	Atributo	Tipo	Restrição
Aluno	IDAluno	Número Inteiro	PK-Chave Primária – Não Nulo
	Nome	Texto 40 posições	Não Nulo
	Sexo	Texto 01 posição	M ou F - não nulo
	DataNascimento	Data	Não Nulo
	Telefone	Texto de 15 posições	Nulo
	EMail	Texto de 60 posições	Nulo
Disciplina	IDDisciplina	Número Inteiro	Chave Primária – PK–Não Nulo.
	Descricao	Texto 40 posições	Não nulo.
	CargaHoraria	Número Inteiro	Não nulo.
Historico	<u>CodAluno</u>	Número Inteiro	FK – Chave estrangeira e Não Nulo
	<u>CodDisciplina</u>	Número Inteiro	FK – Chave estrangeira e Não Nulo
	Data	Data	Não Nulo
	Nota	Número Decimal (10,2)	Não nulo – $0 \leq N \leq 10$
	Situacao	Texto 20 posições	Aprovado, Reprovado, Recuperação. Obs: CodAlulo e CodDisciplina é PK Composta

Tabela 2 – Mostra as Entidades, os Atributos, os Tipos dos Atributos e as Restrições dos atributos.

Uma visão do Diagrama de Tabelas do BDUiXyko é apresentado na Figura 13 – Observe que as Entidades já estão Relacionadas umas com as outras (conforme Regra de Negócio da UniXyko), o Relacionamento N:N entre Aluno e Disciplina foi transformado em dois Relacionamentos 1:N. Em cada Entidade (Tabela) está identificado a Chave Primária e as Restrições que controlam os Relacionamentos entre as Entidades (Tabela). Observe que Historico a PK – Chave primária - é formada por dois campos (chamado Chave Primária Composta).

A Figura 13 mostra o Diagrama de Tabelas do BDUiXyko representando o Relacionamento N:N identificado no Projeto conceitual conforme as regras de negócio da UniXyko onde: Um Aluno pode cursar muitas Disciplinas e uma Disciplina pode ter vários Alunos. Este Diagrama de Tabelas foi criado no SGBD SQL Server 2008 (ou SQL no 2012/2014/2016).

O Relacionamento é Entre Aluno e Historico (1:N – um pra muitos) é controlado pelo objeto PKCodAluno. E o Relacionamento entre Disciplina e Historico (1:N – um pra muitos) é controlado pelo objeto PKCodDisciplina.

Diagrama de Tabelas do BDUiXyko
IFSP-Barretos - Prof. Xyko
Data: 20/12/2011

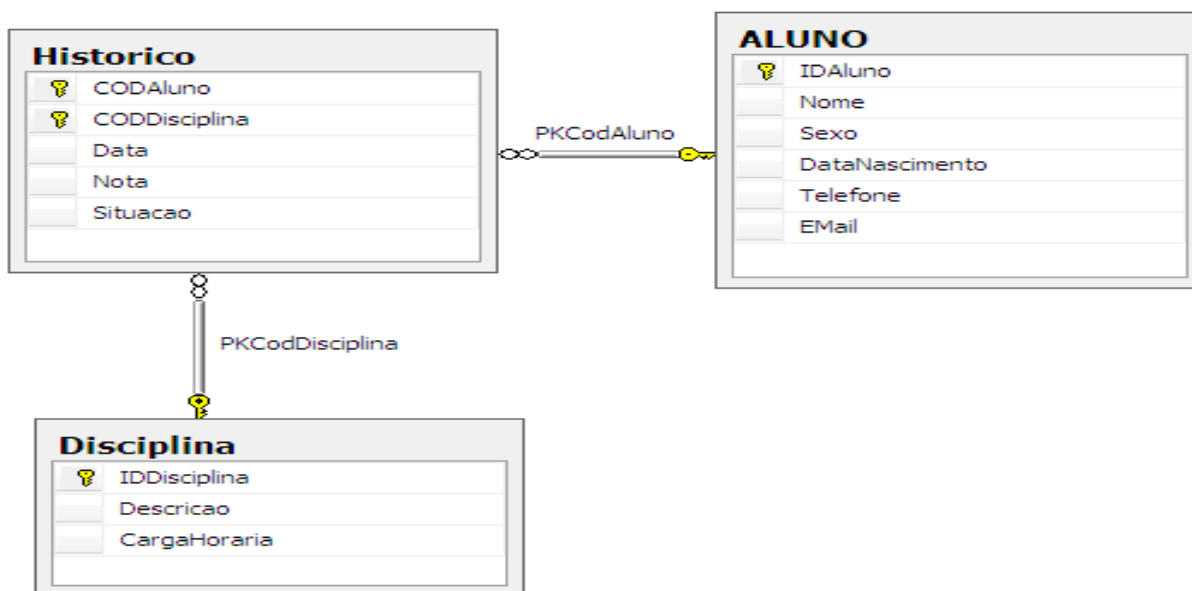


Figura 13 – Diagrama de Tabelas do Projeto Lógico do BDUiXyko com as Tabelas já relacionadas.

É extremamente importante para o Aluno (Usuário de BD) entender nesta fase do Projeto do BDUiXyko (Projeto Lógico) que a forma de armazenar Dados de um Banco de Dados Relacional conforme apresentado na Figura 13 será em forma de Tabelas (formato por linhas e colunas) conforme mostra a Figura 14.

Observe que as Colunas possuem dados com um **Tipo e um Tamanho**. Exemplo: na tabela Disciplina os campos (colunas) IDDisciplina e CargaHoraria são do **Tipo inteiro e só aceita números**. O Campo (coluna) Descrição é Tipo **texto e tem 40 posições disponíveis** para os Dados.

Tabela Disciplina.

	IDDisciplina	Descricao	CargaHoraria
1	10100	Gerenciamento de Banco de Dados	80
2	20200	Informática Básica	80
3	30300	Lógica de Programação	80
4	40400	Linguagem para Web	80
5	50500	Hardware	80
6	60600	Redes de Computadores I	80
7	70700	Administração de Redes	80

Tabela Aluno.

	IDAluno	Nome	Sexo	DataNascimento	Telefone	EMail
1	1010	Karlos Kosta Kurta	M	1980-06-15	17 - 4040-2020	kkk@gmail.com
2	1020	Krak Kent	M	1970-06-15	11 - 2020-7070	kk@gmail.com
3	2020	Senhor Asno	M	1960-07-20	41 - 4097-9070	asno@gmail.com
4	3030	Dona Anta	F	0167-04-17	41- 5690-7070	ANTA@GMAIL.COM

Figura 14 – Conteúdo das Tabelas Aluno e Disciplina com Dados do BDUiXyko .

Projeto, Modelagem e Gerenciamento de Banco de Dados.

Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

É extremamente importante para o Aluno (Usuário de BD) entender nesta fase do Projeto do BDUiXyko (Projeto Lógico) que as Tabelas estão relacionadas pelos campos Chave Primária (PK) e Chave Estrangeira (FK). O relacionamento N:N foi implementado em dois relacionamento 1:N entre Disciplina com Histórico e entre Aluno com Histórico – Vide Figura 15.

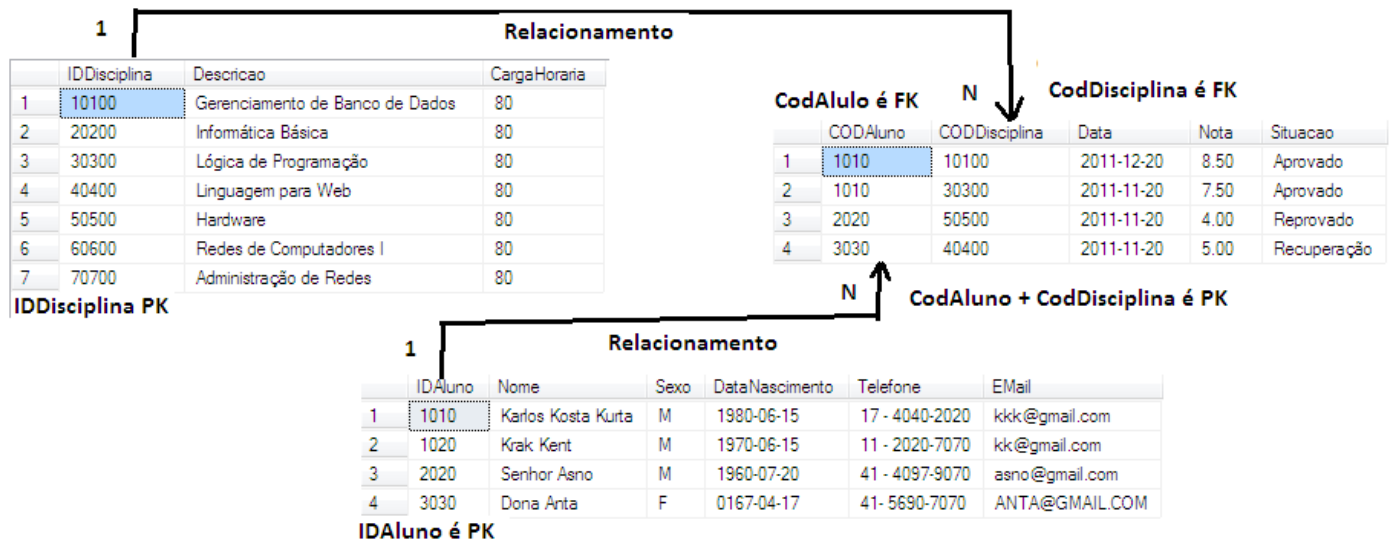


Figura 15 – Relacionamento entre as Tabelas do BDUiXyko.

Na figura 15 é possível perceber o Conceito Fundamental dos Banco de Dados Relacional – Os Dados estão armazenados em forma de Tabelas (que são conjuntos da Matemática – Conjunto não tem Repetição e a ordem não é importante). **As Colunas das Tabelas são os Atributos e as Linhas são os Registros (dados)**. Nos Bancos de Dados Relacionais as Tabelas estão relacionadas umas com as outras conforme especificado na Figura 15.

1.5.3) O que deve ser feito no Projeto (Modelo) Físico.

Agora que o Aluno já tem uma Visão Geral (Ideia formada do BD) para Armazenar os Dados identificados no Mini-Mundo, Já elaborou o Projeto/Modelo Conceitual e o Projeto/Modelo Lógico é hora de criar **Fisicamente** o Banco de Dados usando um **SGBD**. Nesta etapa deve fazer/definir (ter as preocupações):

- Qual o **SGBD** vai implementar (rodar) o Banco de Dados. Tem vários SGBD tais como: MS-Access, SQL Server 2008, Oracle, DB2, MYSql, PostGreSQL, dentre outros.
- Definir o **HARDWARE** – Servidor – que vai ser Instalado o Banco de Dados – Capacidade de Memória, Velocidade do Processar, Largura de banca da Rede, Tipo de Sistema Operacional e Segurança são itens indispensáveis para um bom desempenho do Banco de Dados;
- O Servidor certamente será instalado em uma **Rede de Computadores**;
- Banco de Dados é sempre instalado em um ambiente Cliente-Servidor – O Servidor com o BD fornece (provê) serviço para os usuários da Rede que são

Projeto, Modelagem e Gerenciamento de Banco de Dados.

Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

outros PCs/Computadores que rodam vários aplicativos (Exemplo: Software do Supermercado, Folha de Pagamento, dentre outros).

- Usando a **Linguagem SQL** dentro de um ambiente do SGBD deve: Criar o BD, Criar as Tabelas com as Chaves Primárias e Estrangeiras, Definir as Restrições para os campos da tabela, inserir Registros, fazer alterações nas Tabelas, alterar o Conteúdo dos Dados (registros), Criar Usuários, Dar Permissão aos usuários, Fazer Back-up e Restaurar Banco de Dados;
- Com A Linguagem SQL que é nata dos SGBD Relacional É Possível: Criar, Configurar e Gerenciar (dar Manutenção) os Banco de Dados que dão sustentação para as Empresas concretizarem seus negócios com base nos DADOS armazenados nos Banco de Dados.

1.5.3.1 - Ambiente do SGBD SQL Server 2008/2012/2014/2016 para criar o BDIFSP.

Na Figura 16 é apresentada a interface do SQL Server 2008/2012/2014 com alguns comandos em SQL para Criar o BDIFSP e a Tabela Aluno.

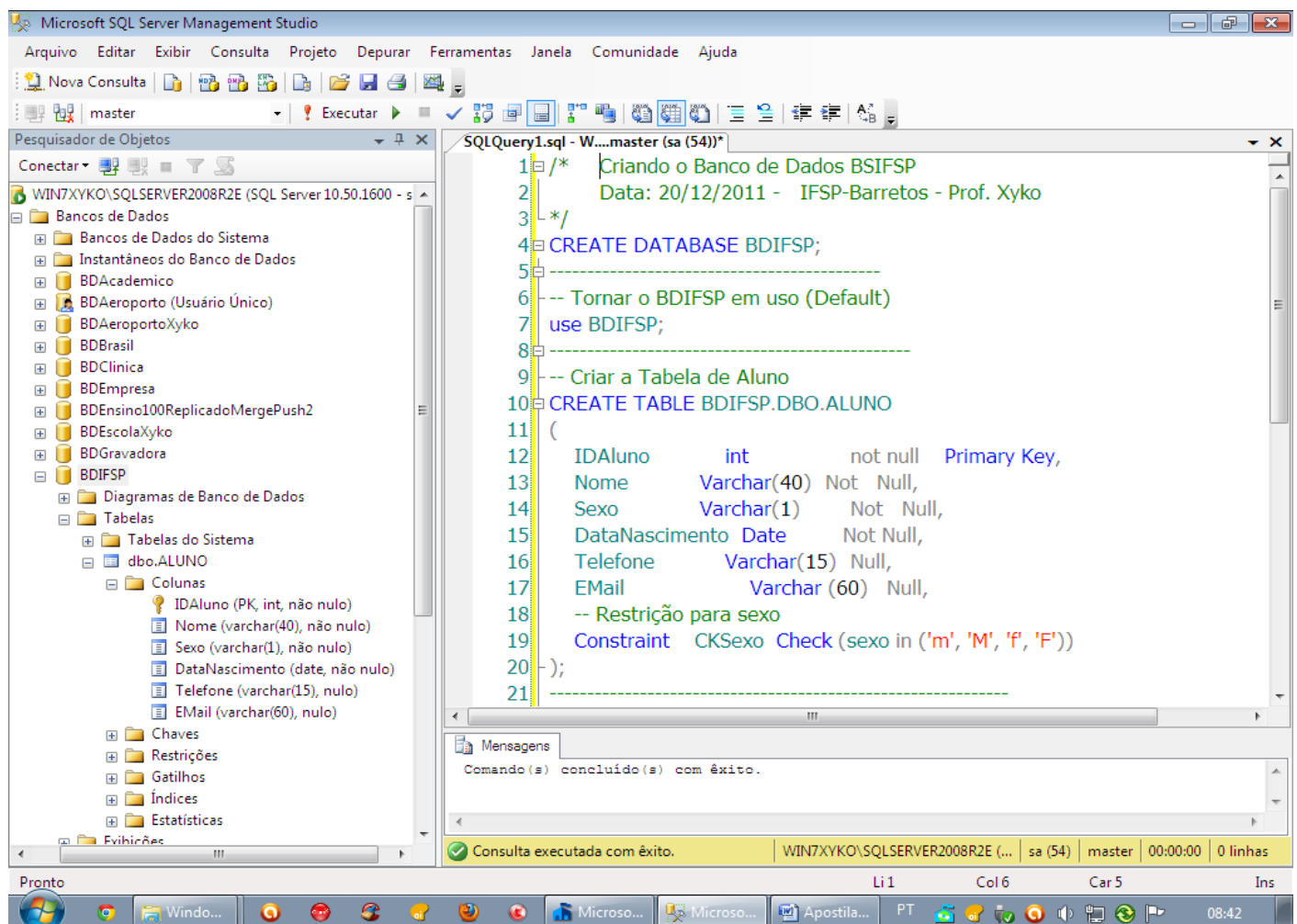


Figura 16 – Interface do SGBD Relacional SQL Server 2008/2012/2014/2016.

Projeto, Modelagem e Gerenciamento de Banco de Dados.

Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

Na Figura 16 do **lado esquerdo** é mostrado tudo que é criado no BD dentro do SGBD – tudo que é criado é chamado de Objeto. No **Lado Direito** é a área de digitação dos comandos SQL que vão criar o BD, Criar as Tabelas e demais objetos – conforme definidos no Projeto/Modelo Lógico. Na parte de Baixo (mensagens) é a tela onde mostra os Resultados da execução do SQL. Esta interface pode ser configurada para incluir ou retirar Janelas conforme necessidade do usuário. Mas basicamente as janelas iniciais são as apresentadas na Figura 16.

Na sequencia é apresentado os comandos (código SQL) que criou o BDUniXyko com as tabelas: Aluno, Disciplina e Histórico.

```
/*          Criando o Banco de Dados BDUniXyko
          Data: 11/06/2016 - IFSP-Barretos - Prof. Xyko
```

```
*/
```

```
CREATE DATABASE BDUniXyko;
```

```
-----
-- Tornar o BDIFSP em uso (Default)
use BDUniXyko;
```

```
-----
-- Criar a Tabela de Aluno
```

```
CREATE TABLE BDUniXyko.DBO.ALUNO
```

```
(
```

```
    IDAluno          int          not null  Primary Key,
    Nome             Varchar(40)   Not Null,
    Sexo             Varchar(1)    Not Null,
    DataNascimento   Date          Not Null,
    Telefone         Varchar(15)   Null,
    EMail            Varchar (60)   Null,
```

```
    -- Restrição para sexo
```

```
    Constraint CKSexo Check (sexo in ('m', 'M', 'f', 'F'))
```

```
);
```

```
-----
-- Criar a Tabela de Disciplina
```

```
CREATE TABLE BDUniXyko.DBO.Disciplina
```

```
(
```

```
    IDDisciplina     int          not null  Primary Key,
    Descricao        Varchar(40)  Not Null,
    CargaHoraria      Int          Not Null
```

```
);
```

```
-----
-- Criar a Tabela de Aluno
```

```
CREATE TABLE BDUniXyko.DBO.Historico
```

```
(
```

```
    CODAluno         int          not null,
    CODDisciplina     int          Not Null,
    Data             Date          Not Null,
    Nota             Decimal (10,2) not null,
    Situacao         Varchar(20)   Not Null,
```

Projeto, Modelagem e Gerenciamento de Banco de Dados.

Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

```
-- Restrição para poara PK - Chave Primária
Constraint PKHistorico Primary key (CodAluno, CodDisciplina),
-- Restrição de FK-Chave estrangeira
Constraint PKCodAluno Foreign Key (CodAluno)
References Aluno (IDaluno),
Constraint PKCodDisciplina Foreign Key (CodDisciplina)
References Disciplina (IDDisciplina),
-- Restrição para Situação
Constraint CKSituacao Check (Situacao in ('Aprovado',
'Reprovado', 'Recuperação')),
-- Restrição de Nota
Constraint CkNota Check (Nota Between 0 and 10.00)
```

);

-- MOSTRANDO OS REGISTROS sem critério de Filtro-Condição

```
SELECT *
FROM ALUNO;
```

-- MOSTRANDO OS REGISTROS sem critério de Filtro-Condição

```
SELECT *
FROM Disciplina;
```

-- MOSTRANDO OS REGISTROS sem critério de Filtro-Condição

```
SELECT *
FROM Historico;
```

Após criar o BD é possível verificar como o BD interpreta (entende) o nosso Projeto do BDUniXyko – que foi todo criado usando SQL no SGBD - Cheque no SQL em Diagrama – Vide Figura 17.

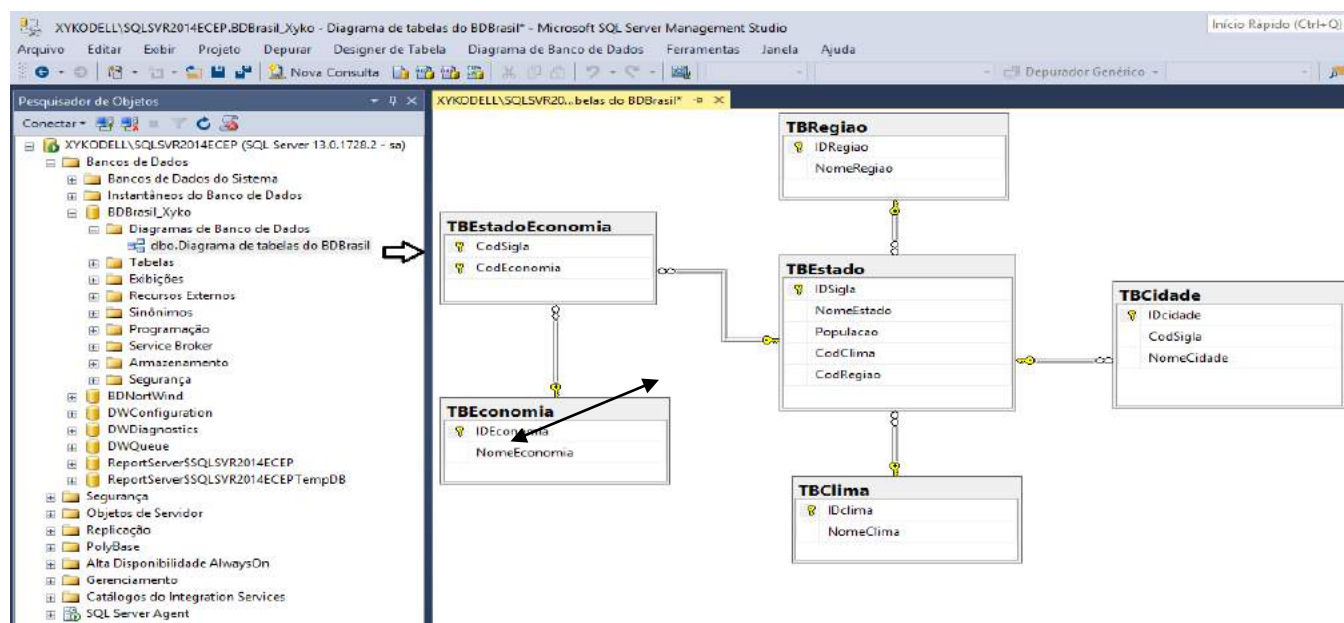


Figura 17 – Diagrama de Tabela do BDUniXyko – Uso do SGBD SQL Server 2008/2012/2014/2016.

1.5.3.2 – Resumo dos Modelos Conceitual, Lógico e Físico.

Resumindo: Para criar um Banco de Dados é preciso elaborar um Projeto para o Banco de Dados – que basicamente é composto de três etapas: Projeto conceitual, lógico e físico.

O Projeto do BD se inicia no Mini-Mundo onde é feito uma Analise de Requisitos do Banco de Dados a ser criado. Depois deve ser criado o Modelo Conceitual (aqui é criado o DER com Entidades, Relacionamentos, Atributos e Cardinalidades).

Em seguida deve ser criado o Modelo Lógico (aqui define os Campos, chaves, restrições, tipos de Dados).

Finalmente é criado o Modelo Físico (aqui usando SQL é criado fisicamente o BD com as Tabelas, os Relacionamentos, as chaves, as restrições, permite inserir dados, consultar dados e fazer todo Gerenciamento do Banco de Dados). É importante o leitor perceber que cada SGBD tem uma interface ligeiramente diferente umas das outras que são usadas digitando os comandos SQL e executando.

Na Figura 8 é apresentado um esquema que resume as Etapas do Projeto para Criar um Banco de dados Relacional.

Importante compreender que um projeto de Banco de Dados surge da necessidade da empresa (cliente) em armazenar e gerenciar seus dados, e a partir desta necessidade o DBA tem que visitar a empresa para entender o funcionamento do negócio e mapear (projetar) o banco de Dados, que posteriormente será implementado em um SGBD.

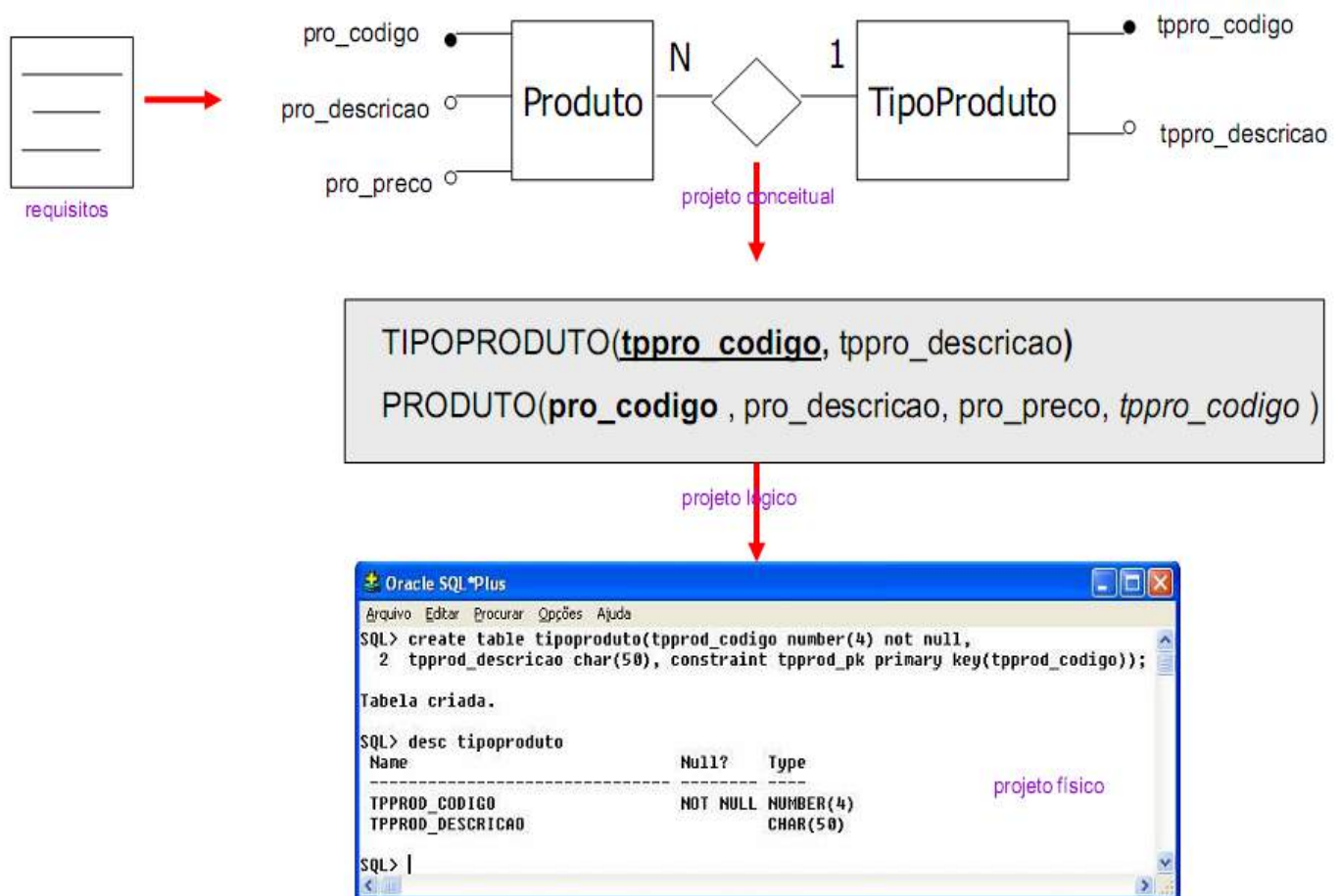


Figura 18 – Resumo das Etapas de um Projeto de Banco de Dados – Modelo Conceitual, Lógico e Físico a partir dos requisitos de armazenamento dados da empresa.

Projeto, Modelagem e Gerenciamento de Banco de Dados.

Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

Na Figura 18 o Projeto Físico foi implementado com comandos SQL no SGBD Relacional Oracle. Compare com a Figura 16 do SGBD Relacional SQL Server 2008/2012/2014/2016 – o Ambiente pode mudar, mas a finalidade é sempre a mesma: Criar, alterar e Administrar uma Base de Dados usando a linguagem SQL que é padronizada pela **ANSI**, e a maioria dos fornecedores de SGBD como: Oracle, Sybase, Microsoft, IBM, dentre outras, tem seus SGBDS no padrão **ANSI**, ou seja um código SQL padronizado **ANSI** roda em vários SGBDs ANSI.

2 – Implementação de Banco de Dados com SQL no SQL Server 2008/2012/2014/2016.

2.1- Estudo de Casos: BDEscolaXyko.


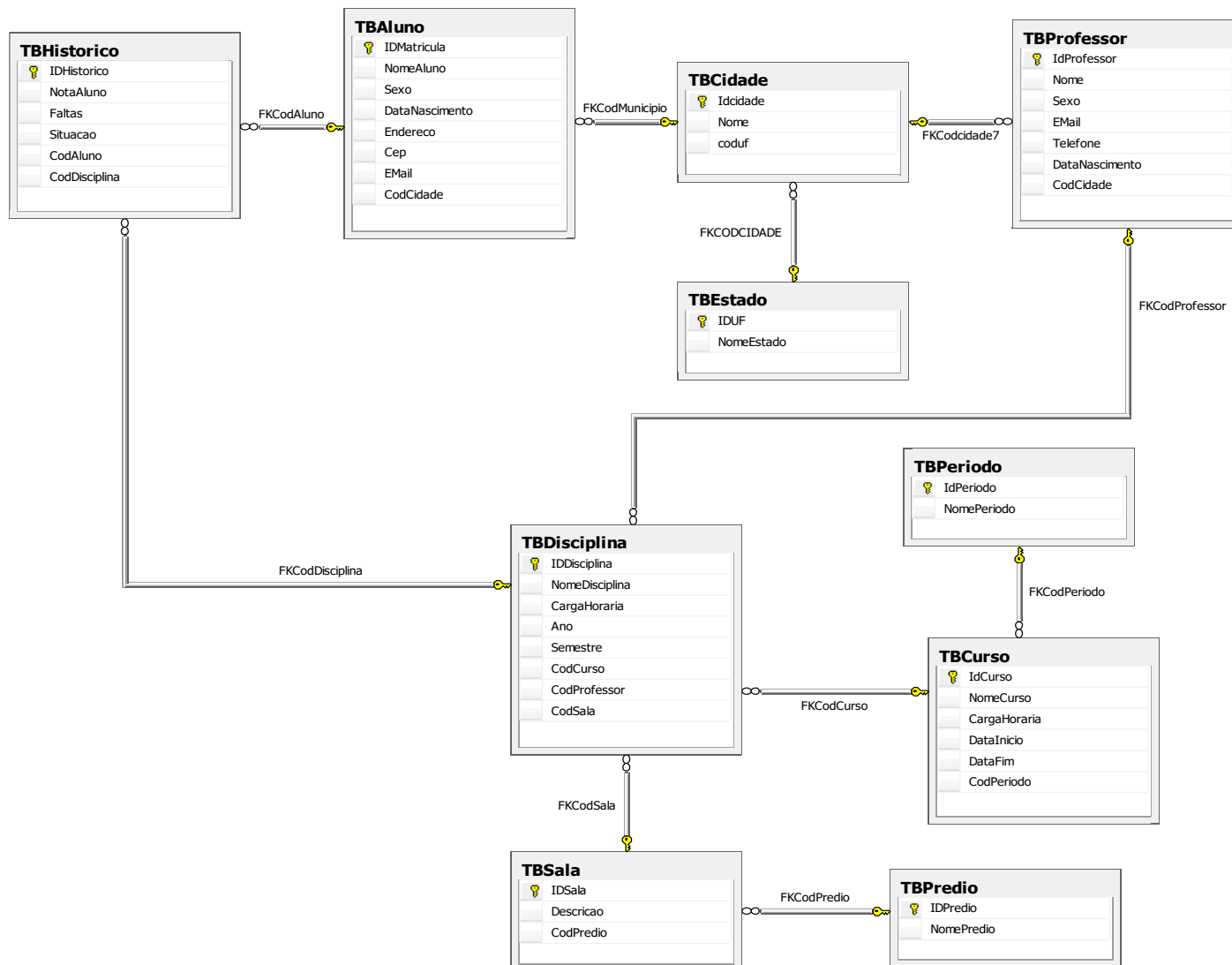
	
PROJETO DE BANCO DE DADOS. CURSO: Manutenção em Sistemas de Informação. TREINAMENTO EM SQL SERVER 2008 a 2016 - Projeto Físico completo de Banco de Dados Implementado em SQL no SQL Server 2008 a 2016.	
Modelagem de Dados e Introdução ao SQL	SQL Server 2008/2012/2014/2016
PROFESSORES : Francisco A de Almeida – Xyko	Data: Junho/2017

Diagrama de Tabelas BDEscolaXyko
IFSP-Barretos
Ano: 2011/2012
Prof. Xyko



/*

Código SQL para Criação e Inserção de Dados no banco de Dados BDEscolaXyko.

Finalidade: Visão Geral da Modelagem, Projeto de Banco de Dados e Treinamento em SQL

SGBD usado: SQL Server 2008 a 2016

• Finalidade:

- **Instalação e apresentação do ambiente do SQL Server 2008 a 2016**
- **Cria o Projeto do Banco de dados;**
- **Introdução ao SQL;**
- **Criar as Tabela;**
- **Fazer comentários em SQL**
- **Fazer Relacionamento;**
- **Inserir Dados;**
- **Fazer Consulta dos Dados Inseridos;**
- **Fazer restrições de entrada de dados na criação da tabela no Banco de Dados**

***/**

/*

Parte 1) Criar o BD, Criar as Tabelas e Inserção de Dados o mostrar os Registros Inseridos - Banco de Dados BDEscolaXyko

***/**

-- criando o banco de Dados

Create DataBase BDEscolaXyko;

-- Definindo o Banco de Dados a trabalhar.

use BDEscolaXyko;

-- definindo a inserção de datas no padrao dd/mm/aaaa

-- Padrão acostumado no Brasil

set dateformat 'dmy';

-- Criando a Tabela de Estados - TBEstado

Create Table BDEscolaXyko.dbo.TBEstado

(

IDUF char(2) not null Constraint PKIduf Primary Key,
NomeEstado Varchar(50) not null

);

```
--Inserindo Registros na Tabela TBEstado
Insert Into BDEscolaXyko.dbo.TBEstado
(IDUF, NomeEstado)
Values
('SP', 'São Paulo');
```

```
Insert Into BDEscolaXyko.dbo.TBEstado
(IDUF, NomeEstado)
Values
('MG', 'Minas Gerais');
```

```
Insert Into BDEscolaXyko.dbo.TBEstado
(IDUF, NomeEstado)
Values
('MT', 'Mato Grosso');
```

```
Insert Into BDEscolaXyko.dbo.TBEstado
(IDUF, NomeEstado)
Values
('RJ', 'Rio de Janeiro');
```

```
Insert Into BDEscolaXyko.dbo.TBEstado
(IDUF, NomeEstado)
Values
('ES', 'Espírito Santo');
```

```
-----
-- Mostrando os registros - dados da Tabela Estados
Select *
  from BDEscolaXyko.dbo.TBEstado;
```

```
-----
-- cria a tabela TBcidade
CREATE TABLE TBCidade
(
    Idcidade int not null primary key,
    Nome varchar(40) NOT NULL,
    coduf char(2) NOT NULL,
    CONSTRAINT FKCODCIDADE FOREIGN KEY(coduf)
        REFERENCES TBEstado(IdUF)
        ON UPDATE CASCADE
        ON DELETE NO ACTION
);
```

```
-----
-- fazendo o Cadastro das Cidades
Insert into BDEscolaXyko.dbo.TBCidade
        (Idcidade, Nome, coduf)
```

Projeto, Modelagem e Gerenciamento de Banco de Dados.
Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

```
values      (10, 'São José do Rio Preto', 'SP'),
            (20, 'Barretos', 'SP'),
            (30, 'Rio de Janeiro', 'RJ'),
            (40, 'Belo Horizonte', 'MG');
```

-- Mostrando os registros - dados da Tabela TBCidade

```
Select *
from BDEscolaXyko.dbo.TBCidade;
```

/* -- Criando as Tabelas TBProfessores

Note o uso de restricao de Check para sexo

***/**

Create Table BDEscolaXyko.dbo.TBProfessor

```
(
    IdProfessor integer not null Constraint PKIDProfessor Primary Key,
    Nome varchar(50) not null,
    Sexo char(1) not null Constraint CKSexo Check ( Sexo in ('m', 'M', 'f', 'F')),
    EMail varchar(70) null,
    Telefone varchar(15) null,
    DataNascimento datetime not null
);
```

-- alterando a Tabela TBProfessor a incluindo um campo

-- incluir o campo CodCidade

-- Faz o relacionamento com TBCidade

```
Alter table BDEscolaXyko.dbo.TBProfessor
add
```

```
CodCidade    int,
```

-- agora cria a restrição de FK

-- Faz o relacionamento com TBCidade

Constraint FKCodcidade7 foreign Key (CodCidade)

References TBCidade (IDCidade)

on delete no action

on update cascade;

/*

inserindo registros na Tabela TBProfessores

Note - inserindo varios registros de uma vez

***/**

Insert Into BDEscolaXyko.dbo.Tbprofessor

(IdProfessor, Nome, Sexo, EMail, Telefone, DataNascimento, CodCidade)

Values (01, 'Francisco Antonio de Almeida', 'M', 'xykoetec@gmail.com', '17-3233 9266', '16/07/1973', 10);

**Insert Into BDEscolaXyko.dbo.Tbprofessor
(IdProfessor, Nome, Sexo, EMail, Telefone, DataNascimento, CodCidade)
Values (02, 'Karlos Kosta Kurta', 'M', 'karlos@gmail.com', '17-3227 6975',
'14/10/1970', 20);**

**Insert Into BDEscolaXyko.dbo.Tbprofessor
(IdProfessor, Nome, Sexo, EMail, Telefone, DataNascimento, CodCidade)
Values (03, 'Katia Kalson', 'f', 'katia@ig.com.br', '11-4567 56871', '15/07/1955',
30);**

**Insert Into BDEscolaXyko.dbo.Tbprofessor
(IdProfessor, Nome, Sexo, EMail, Telefone, DataNascimento, CodCidade)
Values (04, 'Maria de Fatima Fernandes Gallo', 'f', 'mf@ig.com.br', '27-5415
2567', '20/11/1958', 40);**

-- mostrando os registros inseridos
Select *
from BDEscolaXyko.dbo.Tbprofessor;

/*
Criando a tabela de Periodos
Nota - pode cadastrar os periodos dos cursos oferecidos
***/**
Create Table BDEscolaXyko.dbo.TBPeriodo
(
IdPeriodo integer not null Constraint PKIDPeriodo Primary Key,
NomePeriodo varchar(50) not null
);

-- Inserindo registros na Tabela TBPeriodo

**Insert Into BDEscolaXyko.dbo.TbPeriodo
(IdPeriodo, NomePeriodo)
Values (01, 'Matutino');**

**Insert Into BDEscolaXyko.dbo.TbPeriodo
(IdPeriodo, NomePeriodo)
Values (02, 'Vespertino');**

**Insert Into BDEscolaXyko.dbo.TbPeriodo
(IdPeriodo, NomePeriodo)**

Values (03, 'Noturno');

Insert Into BDEscolaXyko.dbo.TbPeriodo
(IdPeriodo, NomePeriodo)
Values (04, 'Fins de Semana');

-- aqui omitimos os campos no insert
-- funciona, mas tem que inserir os dados na ordem que foi criado na tabela
Insert Into BDEscolaXyko.dbo.TbPeriodo
Values (05, 'Integral');

-- Mostrando os dados da tabela TBPeriodo
Select *
from BDEscolaXyko.dbo.TbPeriodo;

/*
Criando a tabela de Cursos
***/**
Create Table BDEscolaXyko.dbo.TBCurso
(
IdCurso integer not null Constraint PKIDCurso Primary Key,
NomeCurso varchar(50) not null,
CargaHoraria integer not null,
DataInicio datetime not null,
DataFim datetime null,
CodPeriodo integer not null
Constraint FKCodPeriodo Foreign Key
references BDEscolaXyko.dbo.TBPeriodo(IdPeriodo)
on delete no action
on update cascade
);

--Inserindo Registros na Tabela Cursso

Insert into BDEscolaXyko.dbo.TBCurso
(IdCurso, NomeCURSO, CargaHoraria, DataInicio,CodPeriodo)
Values (2020, 'Ciência da Computação', 600,'01/02/2000', 01);

Insert into BDEscolaXyko.dbo.TBCurso
(IdCurso, NomeCURSO, CargaHoraria, DataInicio, CodPeriodo)
Values (2030, 'Sistemas de Informação', 600,'01/02/2000', 03);

Insert into BDEscolaXyko.dbo.TBCurso
(IdCurso, NomeCURSO, CargaHoraria, DataInicio, CodPeriodo)
Values (2040, 'Engenharia da Computação', 700,'01/02/2004', 05);
Insert into BDEscolaXyko.dbo.TBCurso
(IdCurso, NomeCurso, CargaHoraria, DataInicio, CodPeriodo)

Projeto, Modelagem e Gerenciamento de Banco de Dados.
Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

Values (2050, 'Tecnologia Análise de Sistemas', 500,'01/02/2005', 02);

Insert into BDEscolaXyko.dbo.TBCurso
(IdCurso, NomeCurso, CargaHoraria, DataInicio, CodPeriodo)
Values (2070, 'Web Design', 500,'01/02/2006', 03);

Insert into BDEscolaXyko.dbo.TBCurso
(IdCurso, NomeCurso, CargaHoraria, DataInicio, CodPeriodo)
Values (2080, 'Informática para Internet', 500,'01/02/2006', 02);

Insert into BDEscolaXyko.dbo.TBCurso
(IdCurso, NomeCurso, CargaHoraria, DataInicio, CodPeriodo)
Values (2090, 'Tecnologia em Agronegócios', 500,'15/02/2007', 02);

-- consultando dados - registros da tabela TbCurso
Select *
from BDEscolaXyko.dbo.TBCurso;

-- Deletando registros
-- cuidado se não for especificado os registros na cláusula where
-- apaga todos os Registros de uma Tabela
Delete from BDEscolaXyko.dbo.TBCurso
where IDCurso in (2050, 2060);

-- Inserindo Registros na Tabela TBCidade
Insert into BDEscolaXyko.dbo.TBCidade
(IdCidade, Nome, CodUf)
Values (4010, 'São João da Boa Vista', 'SP');

-- Inserindo Registros na Tabela TBCidade
Insert into BDEscolaXyko.dbo.TBCidade
(IdCidade, Nome, CodUf)
Values (4020, 'Olimpia', 'SP');

-- Inserindo Registros na Tabela TBCidade
Insert into BDEscolaXyko.dbo.TBCidade
(IdCidade, Nome, CodUf)
Values (4030, 'Rio das Lajes', 'RJ');

-- Inserindo Registros na Tabela TBCidade
-- Registro não Cadastrado DF não esta Cadastrado
Insert into BDEscolaXyko.dbo.TBCidade
(IdCidade, Nome, CodUf)
Values (4050, 'Brasilia', 'DF');

Insert Into BDEscolaXyko.dbo.TBEstado
(IDUF, NomeESTADO)
Values ('DF', 'Distrito Ferederal');

Projeto, Modelagem e Gerenciamento de Banco de Dados.
Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

-- agora rode o insert de BRASILIA

-- Mostrando os Dados da Tabela TBCidade

Select *

from BDEscolaXyko.dbo.TBCidade;

-- Criando a Tabela de Alunos – TBAaluno

-- Aqui tem Restrição para sexo – só aceita M, m, F, ou F

Create table BDEscolaXyko.dbo.TBAaluno

(

IDMatricula integer not null Primary Key,

NomeAluno varchar(50) not null,

Sexo char(1) not null,

DataNascimento DateTime not null,

Endereco varchar(50) null,

Cep varchar(15) null,

EMail varchar(60) null,

-- restrição para sexo

Constraint CKSexo1 Check (Sexo in ('m', 'M', 'F', 'f')),

-- Restrição para Chave estrangeira

CodCidade integer not null Constraint FKCodMunicipio

Foreign Key references BDEscolaXyko.dbo.TBCidade(IDCidade)

on delete no action

on update cascade

);

-- Cadastrando Alunos na Tabela TBAalunos

Insert into BDEscolaXyko.dbo.TBAaluno

(IDMatricula, NomeAluno, Sexo, DataNascimento, Endereco, Cep, EMail,
CodCidade) Values (5000, 'Maria Cheirosa', 'F', '15/07/1975', 'Rua da Felicidade,
230', '15.010-320', 'mc@ig.com.br', 4020);

Insert into BDEscolaXyko.dbo.TBAaluno

(IDMatricula, NomeAluno, Sexo, DataNascimento, Endereco, Cep, EMail,
CodCidade)

Values (5010, 'Cecilia Fabulosa', 'F', '25/07/1978', 'Rua do Amor, 1230', '15.050-
370',
'mc@ig.com.br', 4020);

Insert into BDEscolaXyko.dbo.TBAaluno

(IDMatricula, NomeAluno, Sexo, DataNascimento, Endereco, Cep, EMail,
CodCidade)

Values (5020, 'Philipe Morramed', 'M', '25/07/1978', 'Rua dos Lirios, 8840',
'27.050-370', 'pm@terra.com.br', 4030);

Projeto, Modelagem e Gerenciamento de Banco de Dados.
Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

-- Mostrando o Conteúdo da Tabela TBAIuno

-- aqui mostra todos os campos com todos os registros

Select * from

BDEscolaXyko.dbo.TBAIuno;

-- Criando a Tabela de Predios – TBPredio

-- Sala fica em Prédio e Escola pode ter mais de um Campo - Prédio

Create table BDEscolaXyko.dbo.TBPredio

(

IDPredio integer not null

Constraint PKIDPredio Primary Key,

NomePredio Varchar(50) not Null

);

-- Cadastrando os Predios da Instituição de Ensino

-- Escola pode ter vários Prédios

Insert Into BDEscolaXyko.dbo.TBPredio

(IDPredio, NomePredio)

Values (30,'Predio da Administração');

Insert Into BDEscolaXyko.dbo.TBPredio

(IDPredio, NomePredio)

Values (40,'Predio I');

Insert Into BDEscolaXyko.dbo.TBPredio

(IDPredio, NomePredio)

Values (50,'Predio II – Escola Infantil');

Insert Into BDEscolaXyko.dbo.TBPredio

(IDPredio, NomePredio)

Values (60,'Predio III');

Insert Into BDEscolaXyko.dbo.TBPredio

(IDPredio, NomePredio)

Values (10,'Predio IV – Cursos Pós Graduação');

Insert Into BDEscolaXyko.dbo.TBPredio

(IDPredio, NomePredio)

Values (20,'Predio V');

-- Mostrando os Registros de Prédios da Instituição de Ensino

Select *

from BDEscolaXyko.dbo.TBPredio;

-- Criando a Tabela Sala - TBSala

Create table BDEscolaXyko.dbo.TBSala

(-- aqui a PK foi criada na mesma instrução – mesma linha

**IDSala integer not null Constraint PKIDSala Primary Key,
Descricao Varchar(70) not Null,
CodPredio integer not null,
Constraint FKCodPredio Foreign Key (CodPredio)
References BDEscolaXyko.dbo.TBPredio (IDPredio)
on delete no action
on update cascade**

);

**-----
-- Inserindo - Cadastrando as Salas da Instituição Ensino**

Insert Into BDEscolaXyko.dbo.TBSala

(IDSala, Descricao, CodPredio)

Values (01, 'Tecnico de Informatica I', 10);

Insert Into BDEscolaXyko.dbo.TBSala

(IDSala, Descricao, CodPredio)

Values (02, 'Tecnico de Informatica II', 10);

Insert Into BDEscolaXyko.dbo.TBSala

(IDSala, Descricao, CodPredio)

Values (03, 'Tecnico de Informatica III', 10);

Insert Into BDEscolaXyko.dbo.TBSala

(IDSala, Descricao, CodPredio)

Values (04, 'Tecnico de Agronegócios I', 20);

Insert Into BDEscolaXyko.dbo.TBSala

(IDSala, Descricao, CodPredio)

Values (05, 'Tecnico de Agronegócios II', 20);

Insert Into BDEscolaXyko.dbo.TBSala

(IDSala, Descricao, CodPredio)

Values (06, 'Tecnico de Agronegócios III', 20);

Insert Into BDEscolaXyko.dbo.TBSala

(IDSala, Descricao, CodPredio)

Values (07, 'Introdução ao AgroNegócio I', 20);

Insert Into BDEscolaXyko.dbo.TBSala

(IDSala, Descricao, CodPredio)

Values (08, 'Liderança e Empreendedorismo', 20);

**-----
-- Mostrando as Salas Cadastradas**

-- * sequinifica todos os campos com todos os registros

Select *

from BDEscolaXyko.dbo.TBSala;

- Criando a Tabela Disciplina - TBDisciplina**
- Note as Restrições do tipo Check - restringe a entrada de dados no campo**

Create table BDEscolaXyko.dbo.TBDisciplina

```
(  
    IDDisciplina integer not null,  
    -- restrição de PK – chave primária  
    Constraint PKIdDisciplina Primary Key(IDDisciplina),  
    NomeDisciplina varchar(50) not null,  
    CargaHoraria integer not null,  
    Ano integer not null,  
    -- restrição para Data – só aceita os anos de 2008 a 2013  
    Constraint CKAno CHECK (Ano in (2010, 2011, 2012,2013,2014)),  
    Semestre integer not null,  
    -- Restrição para Semestre – só aceita 1 ou 2  
    Constraint CHSemestre CHECK (semestre in (1,2)),  
    -- campos de chave exportada - Foreign Key  
    CodCurso int not null,  
    -- restrição de chave estrangeira para TBCurso  
    -- observe que é tudo uma única instrução – sem vírgula  
    Constraint FKCodCurso Foreign Key (CodCurso)  
        References BDEscolaXyko.dbo.TBcurso(IDCurso)  
        on delete no action  
        on update cascade,  
    -- restrição de chave estrangeira para TBprofessor  
    CodProfessor int not null,  
    Constraint FKCodProfessor Foreign Key (CodProfessor)  
        References BDEscolaXyko.dbo.TBprofessor(IDProfessor)  
        on delete no action  
        on update cascade,  
    CodSala int not null,  
    -- restrição de chave estrangeira para TBSala  
    Constraint FKCodSala Foreign Key (CodSala)  
        References BDEscolaXyko.dbo.TBSala(IDSala)  
        on update cascade  
        on delete no action  
);
```

-- Inserindo Registros - Cadastrando as Disciplinas

```
Insert Into BDEscolaXyko.dbo.TBDisciplina  
(IDDisciplina, NomeDisciplina, CargaHoraria, Ano, Semestre, CodCurso,  
CodProfessor, CodSala)  
Values (7010, 'Informática Básica', 60, 2010, 1, 2020, 1, 1);
```

```
Insert Into BDEscolaXyko.dbo.TBDisciplina  
(IDDisciplina, NomeDisciplina, CargaHoraria, Ano, Semestre, CodCurso,  
CodProfessor, CodSala)
```

Values (7020, 'Lógica de Programação', 80, 2011, 2, 2030, 3, 1);

Insert Into BDEscolaXyko.dbo.TBDisciplina
(IDDisciplina, NomeDisciplina, CargaHoraria, Ano, Semestre, CodCurso,
CodProfessor, CodSala)
Values (7030, 'Desenvolvimento de Software I', 80, 2012, 1 , 2030, 3, 2);

Insert Into BDEscolaXyko.dbo.TBDisciplina
(IDDisciplina, NomeDisciplina, CargaHoraria, Ano, Semestre, CodCurso,
CodProfessor, CodSala)
Values (7040, 'Desenvolvimento de Software II', 100, 2011, 1 , 2020, 3, 3);

Insert Into BDEscolaXyko.dbo.TBDisciplina
(IDDisciplina, NomeDisciplina, CargaHoraria, Ano, Semestre, CodCurso,
CodProfessor, CodSala)
Values (7050, 'Liderança e Empreendedorismo', 80, 2010, 2 , 2040, 4, 3);

Insert Into BDEscolaXyko.dbo.TBDisciplina
(IDDisciplina, NomeDisciplina, CargaHoraria, Ano, Semestre, CodCurso,
CodProfessor, CodSala)
Values (7060, 'Lógica de Programação', 80, 2010, 2, 2070, 3, 5);

-- Mostrando os Registros Cadastrados nas Disciplinas - TBDisciplina
Select *
from BDEscolaXyko.dbo.TBDisciplina;

-- Criando a Tabela de Histórico do Aluno - TBHistorico
Create table BDEscolaXyko.dbo.TBHistorico
(**-- criado a restrição de PK junto ao campo IDHistorico**
 IDHistorico integer not null Constraint PKIDHistorico Primary Key,
 NotaAluno decimal (4,2)not null,
 Faltas integer not null,
 Situacao Varchar(40) not null,
 -- criado uma restrição para situação
 Constraint CKSituacao CHECK (Situacao in
 ('Aprovado', 'Reprovado por Nota', 'Reprovado por Falta',
 'Reprovado por Falta e Nota', 'Desistente', 'Evadido')),
 CodAluno integer not null,
 -- chave estrangeira para TBAluno
 Constraint FKCodAluno Foreign Key (CodAluno)
 references BDEscolaXyko.dbo.TBAluno (IDMatricula)
 on delete no action,
 CodDisciplina integer not null,
 -- chave Estrangeira para TBDisciplina
 Constraint FKCodDisciplina Foreign KEY (CodDisciplina)
 references BDEscolaXyko.dbo.TBDisciplina (IDDisciplina)
)

```
on delete no action
on update cascade
);
```

```
-- Inserindo Registros no Historico Aluno - Cadastro TBHistorico
Insert into BDEscolaXyko.dbo.TBHistorico
( IDHistorico, CodAluno, CodDisciplina, NotaAluno, Faltas, Situacao )
Values ( 01, 5000, 7010, 7.50, 10, 'Aprovado');
```

```
Insert into BDEscolaXyko.dbo.TBHistorico
( IDHistorico, CodAluno, CodDisciplina, NotaAluno, Faltas, Situacao )
Values      ( 02, 5000, 7020, 4.50, 40, 'Reprovado por Falta e Nota');
```

```
Insert into BDEscolaXyko.dbo.TBHistorico
( IDHistorico, CodAluno, CodDisciplina, NotaAluno, Faltas, Situacao )
Values      ( 03, 5010, 7010, 4.50, 40, 'Reprovado por Falta e Nota');
```

```
Insert into BDEscolaXyko.dbo.TBHistorico
( IDHistorico, CodAluno, CodDisciplina, NotaAluno, Faltas, Situacao )
Values      ( 04, 5010, 7020, 3.50, 30, 'Evadido');
```

```
Insert into BDEscolaXyko.dbo.TBHistorico
( IDHistorico, CodAluno, CodDisciplina, NotaAluno, Faltas, Situacao )
Values      ( 05, 5010, 7030, 8.50, 12, 'Aprovado');
```

```
Insert into BDEscolaXyko.dbo.TBHistorico
( IDHistorico, CodAluno, CodDisciplina, NotaAluno, Faltas, Situacao )
Values      ( 06, 5000, 7030, 6.50, 12, 'Aprovado');
```

```
Insert into BDEscolaXyko.dbo.TBHistorico
( IDHistorico, CodAluno, CodDisciplina, NotaAluno, Faltas, Situacao )
Values      ( 07, 5000, 7040, 9.50, 15, 'Aprovado');
```

```
-----
--Mostrando os Dados da Tabela TBHistorico
Select * from
      BDEscolaXyko.dbo.TBHistorico;
```

```
-----
Select * from
      BDEscolaXyko.dbo.TBAluno;
```

----- Parte 2 – Consultas no Banco de Dados BDEscolaXyko -----

----- Questionário de Consultas -----

/* 1) Consultar todas as disciplinas realizadas pelo aluno de IDMatricula igual a 5000, suas respectivas notas, faltas, situação, Nome das Disciplinas e os professores que ministraram estas disciplinas */

-- especifica os campos

Select IDMatricula as Matricula,
 NomeAluno as Aluno,
 IDdisciplina as [Código Disciplina],
 NomeDisciplina as nome,
 NotaAluno as Nota,
 d.CargaHoraria as [Carga Horária],
 Faltas,
 Situacao,
 Nome as Professor **-- uso de alias = apelido**

-- especifica as tabelas

from BDEscolaXyko.dbo.TBAluno as a, **-- uso de alias apelido**
 BDEscolaXyko.dbo.TBDisciplina as d,
 BDEscolaXyko.dbo.TBHistorico as h,
 BDEscolaXyko.dbo.TBprofessor as p **-- o último campo não tem vírgula**

-- condições e relacionamentos

where (p.idprofessor = d.codprofessor)
and (d.IDDisciplina = h.CodDisciplina)
and (a.IDMatricula = h.CodAluno)
and IDMatricula = 5000;

/*

2) Consulta, Mostrar para cada Curso, as disciplinas que são ministradas neste Curso e os respectivos períodos */

-- especifica os campos. Pode usar alias

Select IdCurso as [Código Curso],
 NomeCurso as Curso,
 IDDisciplina as [Código Disciplina],
 NomeDisciplina as Disciplina,
 NomePeriodo as Período

-- especifica as tabelas

From BDEscolaXyko.dbo.TBPeriodo as p,
 BDEscolaXyko.dbo.TBCurso as c,
 BDEscolaXyko.dbo.TBDisciplina as d

-- especifica os relacionamentos e

-- Todas as condições

Where (p.IDPeriodo = c.CODPeriodo)
and (c.IDCurso = d.CodCurso)

/*

3) Mostrar os Cursos que ainda não tiveram disciplinas Ministradas

***/**

Select **Distinct** IDCurso as [Código Cusro],

Projeto, Modelagem e Gerenciamento de Banco de Dados.

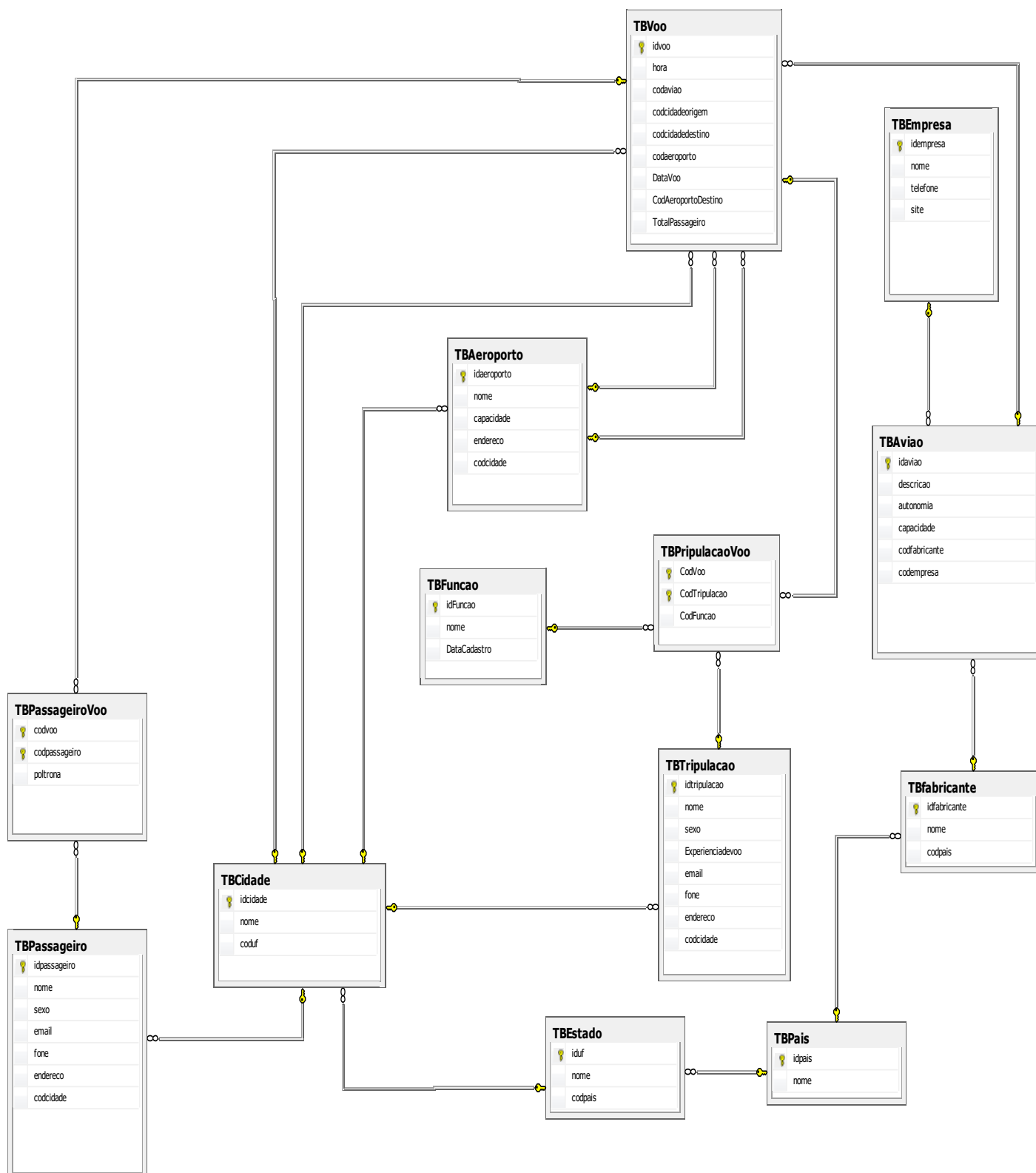
Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

```
NomeCurso as Curso
-- IDDisciplina as [Código Disciplina],
-- NomeDisciplina as Disciplina
-- especifica os relacionamentos e as condições
From      BDEscolaXyko.dbo.TBCurso as c,
          BDEscolaXyko.dbo.TBDisciplina as d
Where  IDCurso not in
      (
        Select CodCurso
        from tbdisciplina
      );

-- testando
Select *
from BDEscolaXyko.dbo.TBCurso;
```

Fica como exercícios a elaboração de mais algumas consultas por parte dos estudantes (alunos). O aprendizado de consultas é fundamental para trabalhar com a linguagem SQL de forma profissional, pois depois, as consultas são utilizadas em visões, funções, procedures e triggers.

2.2 - Estudo de caso: Implementação do Projeto do BDAeroportoXyko no SQL Server de 2008 a 2016



Projeto, Modelagem e Gerenciamento de Banco de Dados.
Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

```
/*
Projeto Banco de Dados BDAeroportoXyko
IFSP-Barretos - Prof. Xyko
Data: Junho/2017
*/

-----Codigo SQL Criando o Banco Dados -----
-- Criando o Banco de Dados
Create database BDAeroportoXyko;

-----
-- Criando a Tabela TBPais
-- chave primária criada junto com a IDPais
CREATE TABLE BDAeroportoXyko.dbo.tbPais
(
    IDPais integer Not Null Primary Key,
    nome VarChar(40) Not Null
);

-----
-- criar a tabela TBEstado
CREATE TABLE BDAeroportoXyko.dbo.TBEstado
(
    IDuf Char(2) Not Null Primary Key,
    Nome VarChar(40) Not Null,
    CodPais integer Not Null,
    -- criando a chave estrangeira - FK --
    CONSTRAINT fkcodpais Foreign Key (codpais)
        REFERENCES BDAeroportoXyko.dbo.TBPais(IDpais)
);

-----
-- criar a tabela TBCidade
CREATE TABLE BDAeroportoXyko.dbo.TBCidade
(
    IDCidade integer Not Null Primary Key,
    Nome VarChar(50) Not Null,
    CODuf Char(2) Not Null,
    --Definindo a chave estrangeira--
    CONSTRAINT fkcoduf Foreign Key (CODuf)
        REFERENCES BDAeroportoXyko.dbo.TBEstado(IDuf)
);

-----
-- criar a tabela TBFabricante
create table BDAeroportoXyko.dbo.TBFabricante
(
    IDFabricante integer not null primary key,
    Nome varchar(40) not null,
```

```

        CODpais                integer not null,
        CONSTRAINT fkcodpais1 Foreign Key (CODpais)
        REFERENCES BDAeroportoXyko.dbo.tbPais(IDpais)
    );
-----
-- criar a tabela TBEmpresa
create table BDAeroportoXyko.dbo.TBEmpresa
(
    IDempresa                integer not null primary key,
    Nome                    varchar(40) not null,
    Telefone                varchar(10) not null,
    site                    varchar(40) not null
);
-----
-- criar a tabela TBAviao
create table BDAeroportoXyko.dbo.TBAviao
(
    IDaviao                integer not null primary key,
    Descricao                varchar(40) not null,
    Autonomia                integer not null,
    Capacidade                integer not null,
    CODFabricante            integer not null,
    CODEmpresa                integer not null,
    -- restrição TBFabricante
    constraint fkcodfabricante foreign key (CODFabricante)
        references BDAeroportoXyko.dbo.TBFabricante,
    -- Restrição TBEmpresa
    constraint fkcodempresa foreign key (CODEmpresa)
        references BDAeroportoXyko.dbo.TBEmpresa
);
-----
-- criar a tabela TBPassageiro
Create Table BDAeroportoXyko.dbo.TBPassageiro
(
    IDPassageiro integer not null primary key,
    Nome varchar(40) not null,
    Sexo char not null,
    Email varchar(40) not null,
    Telefone varchar(10) not null,
    Endereco varchar(40) not null,
    CODCidade integer not null,
    constraint fkcodcidade foreign key (CODCidade)
        references BDAeroportoXyko.dbo.tbCidade(IDCidade)
);
-----
-- criar a tabela TBFuncao
```

Create Table BDAeroportoXyko.dbo.TBFuncao

(
 IDFuncao integer not null primary key,
 Nome varchar(40) not null,
 Datacadastro date not null

);

-- criar a tabela TBTripulacao

Create Table BDAeroportoXyko.dbo.TBTripulacao

(
 IDTripulacao integer not null primary key,
 Nome varchar(40) not null,
 Sexo char not null,
 Experienciadevoo integer not null,
 Email varchar(40) not null,
 Fone varchar(10) not null,
 Endereco varchar(40) not null,
 CodCidade integer not null,
 constraint fkcodcidade1 foreign key (CodCidade)
 references BDAeroportoXyko.dbo.tbCidade(IDCidade)

);

-- criar a atabela TBAeroporto

Create Table BDAeroportoXyko.dbo.TBAeroporto

(
 IDAeroporto integer not null primary key,
 Nome varchar(40) not null,
 Capacidade integer not null,
 Endereco varchar(40) not null,
 CODCidade integer not null,
 constraint fkcodcidade2 foreign key (CODCidade)
 references BDAeroportoXyko.dbo.tbCidade(idcidade)

);

-- criar a tabela TBVoo

Create table BDAeroportoXyko.dbo.TBVoo

(
 IDVoo integer not null primary key,
 Hora time not null,
 DataVoo date not null,
 TotalPassageiros integer not null,
 CodAviao integer not null,
 CodCidadeorigem integer not null,
 CodCidadedestino integer not null,
 CodAeroporto integer not null,
 CodAeroportodestino integer not null,
 constraint fkcodaviao foreign key(CodAviao)

```
references BDAeroportoXyko.dbo.tbAviao(IdAviao),
constraint fkcodcidadeorigem foreign key(CodCidadeorigem)
references BDAeroportoXyko.dbo.tbCidade(idcidade),
constraint fkcodcidadedestino foreign key(CodCidadedestino)
references BDAeroportoXyko.dbo.TBCidade(idcidade),
constraint fkcodeaeroporto foreign key(codeaeroporto)
references BDAeroportoXyko.dbo.tbAeroporto(IDAeroporto),
constraint fkcodeaeropordestino foreign key(CodAeropordestino)
references BDAeroportoXyko.dbo.tbAeroporto(IDAeroporto)
);
-----
-- criar a tabela TBTripulacaoVoo
Create Table BDAeroportoXyko.dbo.TBTripulacaoVoo
(
    CodVoo integer not null,
    CodTripulacao integer not null,
    CodFuncao integer not null,
    /* criando a chave primária */
    constraint pktbTripulacaoVoo primary key (CodVoo, CodTripulacao),
    constraint fkcodvoo foreign key (CodVoo)
        references BDAeroportoXyko.dbo.TBVoo(IDvoo),
    constraint fkcodtripulacao foreign key (CodTripulacao)
        references BDAeroportoXyko.dbo.TBTripulacao (IDTripulacao),
    constraint fkcodfuncao foreign key(codfuncao)
        references BDAeroportoXyko.dbo.TBFuncao(IDfuncao)
);
-----
-- criar a tabela TBPassageiroVoo
create table BDAeroportoXyko.dbo.TBPassageiroVoo
(
    CodVoo integer not null,
    CodPassageiro integer not null,
    /* criando a chave primária */
    constraint pktbPassageiroVoo primary key (codvoo, CodPassageiro),
    constraint fkcodvoo1 foreign key (CodVoo)
        references BDAeroportoXyko.dbo.TBVoo(idvoo),
    constraint fkcodpassageiro foreign key (CodPassageiro)
        references BDAeroportoXyko.dbo.tbPassageiro (IDPassageiro)
);
-----
/* Cadastro dos Dados nas Tabelas
   Inserindo registros nas tabelas */
-----
-- cadastro de Países
INSERT INTO BDAeroportoXyko.dbo.tbpais
(idpais, nome) Values(1, 'Brasil'),(2, 'Argentina'),(3, 'Portugal');
-- cadastro de Cidades
```


INSERT INTO BDAeroportoXyko.dbo.tbestado

(iduf, nome, codpais)

Values('SP', 'São Paulo', 1), ('RS', 'Rio Grande do Sul', 1), ('RJ', 'Rio de Janeiro', 1), ('BA', 'Bahia', 1), ('PR', 'Paraná', 1), ('SC', 'Santa Catarina', 1), ('MG', 'Minas Gerais', 1), ('AC', 'Acre', 1), ('AM', 'Amazonas', 1), ('RR', 'Roraima', 1), ('RO', 'Rondônia', 1), ('PA', 'Pará', 1), ('AL', 'Alagoas', 1), ('PI', 'Piauí', 1), ('MA', 'Maranhão', 1), ('CE', 'Ceará', 1), ('SE', 'Sergipe', 1), ('DF', 'Distrito Federal', 1), ('GO', 'Goáias', 1), ('MT', 'Mato Grosso', 1), ('MS', 'Mato Grosso do Sul', 1), ('TO', 'Tocantins', 1), ('AP', 'Amapá', 1), ('ES', 'Espírito Santo', 1), ('PE', 'Pernambuco', 1), ('RN', 'Rio Grande do Norte', 1), ('PB', 'Paraíba', 1);

--mostrando os Registros

Select * from

BDAeroportoXyko.dbo.tbestado;

--- Cadastro de cidades

INSERT INTO BDAeroportoXyko.dbo.tbcidade

(idcidade, nome, coduf)

Values(1, 'São Paulo', 'SP'), (2, 'São José do Rio Preto', 'SP'), (3, 'Porto Alegre', 'RS');

-- Cadastro de Fabricante

insert into BDAeroportoXyko.dbo.tbFabricante

(idfabricante,nome,codpais)

values(1, 'TAM', 1), (2, 'GOL', 1), (3, 'Verde e Amarelo', 1);

-- cadastro de Empresa

insert into BDAeroportoXyko.dbo.tbEmpresa

(idempresa,nome,telefone,site)

values (1, 'TAM', '4858-9668', 'www.tam.com.br'), (2, 'GOL', '5547-6635', 'www.gol.com.br'), (3, 'V&A', '4654-8552', 'www.vea.com.br');

-- Cadastro de Aviao

insert into BDAeroportoXyko.dbo.tbAviao

(idaviao,descricao,autonomia,capacidade,codempresa,codfabricante)

values (1, 'Boing 747 p/ viagens interestaduais', 18, 200, 1, 1), (2, 'Boing 666 p/ viagens nacionais e inter', 25, 300, 1, 1), (3, 'Boing 123 p/ viagens curtas', 5, 140, 2, 2);

-- cadastro de Passageiro

insert into BDAeroportoXyko.dbo.tbPassageiro

(idpassageiro,nome,sexo,endereco,telefone,email,codcidade)

Values (1, 'Thyrrie Rodrigues', 'M', 'José Figueira nº 354', '3819-7393', 'thyrrie@hotmail.com', 2), (2, 'Breno Gonçalves', 'M', 'Rubião Jr. nº 1574', '3230-4587', 'breno_crazy@hotmail.com', 2), (3, 'Wendel Hanashiro', 'M',

Projeto, Modelagem e Gerenciamento de Banco de Dados.
Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

```
'Cila nº 1254', '3232-4447', 'whanashiro@hotmail.com', 2);
```

```
-----  
-- cadastro de Tripulação
```

```
insert into BDAeroportoXyko.dbo.tbTripulacao
```

```
(idtripulacao,nome,sexo,endereco,fone,email,experienciadevoo,codcidade)
```

```
values      (1, 'João José', 'M', 'Rua que sobe', '3230-4453', 'jj@gmail.com', 500, 1),  
            (2, 'Klark Kent', 'M', 'Pequenópolis', '3232-4899', 'kk@gmail.com', 600,  
1),
```

```
(3, 'Karlos Kosta Kurta', 'M', 'Notredame', '5421-5200', 'kkk@hotmail.com', 400, 1);
```

```
-- formatar padar de inserção de datas
```

```
-- padrao dd/mm/aaaa
```

```
set dateformat 'dmy';  
-----
```

```
-- cadastro de Funcao
```

```
insert into BDAeroportoXyko.dbo.tbFuncao
```

```
(idfuncao,nome,datacadastro)
```

```
values      (1, 'Comandante', '15/10/2008'),  
            (2, 'Aeromoça', '01/02/2009'),  
            (3, 'Co-Piloto', '10/02/2009');
```

```
-----  
-- Cadastro de Aeroporto
```

```
insert into BDAeroportoXyko.dbo.tbAeroporto
```

```
(idaeroporto,nome,endereco,capacidade,codcidade)
```

```
values      (1, 'Congonhas', 'Rua 47, 270 - Leste', 1000, 1),  
            (2, 'Internacional', 'Rua 25, 543 - Centro', 1000, 3),  
            (3, 'Internacional II', 'Rua 34, 56 - Jardins', 1000, 1),  
            (4, 'São José do Rio Preto', 'Rua 82, 56 - Aeroporto', 200, 2);  
-----
```

```
-- Cadastro de VOO
```

```
insert into BDAeroportoXyko.dbo.tbVoo
```

```
(idvoo,datavoo,hora,totalpassageiros,codaeroporto,codaeroportodestino,codaviao,c  
odcidadeorigem,codcidadedestino)
```

```
Values      (1, '08/06/2010', '10:00:00', 100, 1, 2, 1, 1, 2),  
            (2, '08/06/2010', '15:00:00', 120, 1, 2, 1, 1, 3),  
            (3, '08/06/2010', '16:00:00', 110, 2, 2, 1, 1, 1);  
-----
```

```
-- cadastro de Tripulacao
```

```
insert into BDAeroportoXyko.dbo.tbTripulacaoVoo
```

```
(codvoo,codtripulacao,codfuncao)
```

```
values(1, 1, 1), (2, 2, 2), (3, 3, 3);  
-----
```

```
-- cadastro de Passageiro no voo
```

```
insert into BDAeroportoXyko.dbo.tbPassageiroVoo
```

```
(codpassageiro,codvoo)
```

```
values(1, 1), (2, 1), (3, 2);  
-----
```

Exercícios: Fazer os demais cadastros nas demais tabelas.

2.3 – Estudo de Caso - Implementação do Banco Dados BDClínica.

```
/*
Criando Banco de Dados BDClínica a ser implementado no SQL Server 2008 a 2016
IFSP – Barretos -SP – Prof. Xyko
Data: Junho/2017
*/
-- criando o Banco de Dados BDClínica
Create Database BDClínica;
-----
-- tornar o BDClínica em Uso
use BDClínica;
-----
/*
    Criando a tabela TBPais    BDClínica
*/
Create table BDClínica.dbo.TBPais
(
    idPais integer not null,
    nomePais varchar(40) not null,
    constraint PKidPais Primary Key (idPais)
);
-----
/*
    Criando a tabela TBEstado    BDClínica
*/
Create table BDClínica.dbo.TBEstado
(
    idUF char(2) not null Primary Key,
    nomeUF varchar(40) not null,
    codPais integer not null,
    -- Criando a chave estrangeira FK
    constraint FKCodPais foreign key (codPais)
    references BDClínica.dbo.TBPais(idPais)
);
-----
/*
    Criando a tabela TBCidade    BDClínica
*/
Create table BDClínica.dbo.TBCidade
(
    idCidade integer not null Primary Key,
    nomeCidade varchar(45) not null,
    codUF char(2) not null,
    -- Criando a chave estrangeira FK
```

**constraint FKCodUF foreign key (codUF)
references BDClínica.dbo.TBEstado(idUF)**

);

/*

Inserindo Países

***/**

**insert into BDClínica.dbo.TBPais
(idPais,nomePais)**

**values (1,'Argentina'), (2,'Paraguai'), (3,'Chile'),
(4,'França'), (5,'Costa do Marfim'), (6,'Japão'),
(7,'Espanha'), (8,'Portugal'), (9,'Dinamarca'),
(10,'Brasil');**

select *

from BDClínica.dbo.TBPais;

/* Inserindo Estados */

**insert into BDClínica.dbo.TBEstado
(idUF,nomeUF,codPais)**

**Values ('SP','São Paulo',10), ('MG','Minas Gerais',10),
('RS','Rio Grande do Sul',10), ('RJ','Rio de Janeiro',10),
('BA','Bahia',10), ('AM','Amazonas',10),
('AC','Acre',10), ('GO','Goias',10),
('MT','Mato Grosso',10), ('MS','Mato Grosso do Sul',10),
('SE','Sergipe',10), ('ES','Espírito Santo',10),
('PA','Para',10), ('PR','Parana',10),
('MA','Maranhão',10), ('CE','Ceara',10),
('PI','Piauí',10), ('RN','Rio Grande do Norte',10),
('RO','Rondonia',10), ('RR','Roraima',10),
('TO','Tocantins',10), ('SC','Santa Catarina',10),
('AL','Alagoas',10), ('PE','Pernambuco',10),
('AP','Amapá',10), ('PB','Paraíba',10),
('DF','Distrito Federal',10);**

select *

from BDClínica.dbo.TBEstado;

**INSERT INTO BDClínica.dbo.TBCidade
(IDCidade,nomeCidade,CodUF)**

**Values (1,'Rio Branco','AC'), (2,'Maceio','AL'),
(3,'Manaus','AM'), (4,'Macapá','AP'),
(5,'Salvador','BA'), (6,'Fortaleza','CE'),
(7,'Brasília','DF'), (8,'Vitória','ES'),
(9,'Goiânia','GO'), (10,'São Luís','MA'),
(11,'Belo Horizonte','MG'), (12,'Cuiabá','MT'),
(13,'Campo Grande','MS'), (14,'Belem','PA'),**

Projeto, Modelagem e Gerenciamento de Banco de Dados.
Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

(15,'João Pessoa','PB'),
(17,'Teresina','PI'),
(19,'Rio de Janeiro','RJ'),
(21,'Porto Velho','RO'),
(23,'Porto Alegre','RS'),
(25,'Aracaju','SE'),
(27,'Palmas','TO');

(16,'Refice','PE'),
(18,'Curitiba','PR'),
(20,'Natal','RN'),
(22,'Boa Vista','RR'),
(24,'Florianopolis','SC'),
(26,'São Paulo','SP'),

select*
from BDClinica.dbo.TBCidade

-- Criando tabela Clinica
create table BDClinica.dbo.TBClinica
(
 idClinica integer not null Primary Key,
 nomeClinica varchar(60) not null,
 codCidade integer not null,
 -- Criando chave estrangeira
 constraint FKCodCidade foreign key (codCidade)
 references BDClinica.dbo.TBCidade (idCidade)
);

insert into BDClinica.dbo.TBClinica
(idClinica,nomeClinica,codCidade)
values (1,'Clinica joice',26),
 (2,'Clinica moacir',23),
 (3,'Clinica Fernando',4);

select *
from BDClinica.dbo.TBClinica

-- Criando tabela Funcionário
create table BDClinica.dbo.TBFuncionario
(
 idFuncionario integer not null Primary Key,
 nomeFuncionario varchar (50) not null,
 endereco varchar (50) not null,
 cep char (9) not null,
 datanasc datetime not null,
 dataadmissao datetime not null,
 telefone char (9) not null,
 email varchar (30) not null,
 codcidade integer not null,
 -- criando chave estrangeira
 constraint FKCodCidade2 foreign key (codcidade)
 references BDClinica.dbo.TBCidade (idCidade)
);

```
-----
--data
--formatar inserção data
set dateformat 'DMY';
-----

insert into BDClinica.dbo.TBFuncionario
(idFuncionario,nomeFuncionario,endereco,cep,datanasc,dataadmissao,
telefone,email,codcidade)
values (1,'Joice Ferrini','Rua Antonio Pereira Braga','15061-
310','09/09/1991','01/01/2001','3224-4563','p0n31ss@hotmail.com',26),
(2,'Moacyr Miranda','Rua Haroldo Barreiro Carvalho','15045-
395','30/04/1986','02/02/2002','3237-7495','lucas_zago57@hotmail.com',13),
(3,'Fernando Hortencio','Rua Ipiranga','15025-
520','22/02/1991','03/03/2003','3231-3900','guiibes@live.com',4);
-----

select *
from BDClinica.dbo.TBFuncionario
-----

-- criando tabela paciente
Create table BDClinica.dbo.TBPaciente
(
    IDPaciente integer not null Primary Key,
    nomePaciente varchar (50) not null,
    enderecoPaciente varchar (50) not null,
    cepPaciente char(9) not null,
    telefonePaciente char(13) not null,
    emailPaciente varchar(40) not null,
    codCidade integer not null,
    sexoPaciente char(1) not null,
    rgPaciente char(12) not null,
    datanascPaciente datetime not null,
    -- criando chave estrangeira
    constraint FKCodCidade3 foreign key (codCidade)
    references BDClinica.dbo.TBCidade (idCidade)
);
-----

insert into BDClinica.dbo.TBPaciente
(IDPaciente,nomePaciente,enderecoPaciente,cepPaciente,telefonePaciente,
emailPaciente,codCidade,sexoPaciente,rgPaciente,datanascPaciente)
values (1,'Joelson da Silva ','Rua Laranja','15012-123','113221-
1243','beutrano_santos@hotmail.com',26,'M','23123123-5','05/05/1992'),
(2,'Felisbino Antonio','Rua Flores','15028-225','153241
1163','joãozinho@hotmail.com',23,'M','12413445-6','12/07/1992'),
(3,'Acacia dos Santos ','Rua Paraíso ','15024-244','183321-
3412','alininhaa@hotmail.com',4,'F','4123131-1','17/07/1991');
-----
```



```
select *  
from BDClinica.dbo.TBPaciente
```

-- criando tabela medico

Create table BDClinica.dbo.TBMedico

```
(  
    idCRM integer not null Primary Key,  
    nomeMedico varchar(50) not null,  
    cpfMedico char(14) not null,  
    telefoneMedico char(13) not null,  
    emailMedico varchar(40) not null,  
    codCidade integer not null,  
    enderecoMedico varchar(50) not null,  
    cepMedico char(9) not null,  
    --criando chave estrangeira  
    constraint FKCodCidade4 foreign key (codCidade)  
    references BDClinica.dbo.TBCidade (idCidade)  
);
```

insert into BDClinica.dbo.TBMedico

```
(idCRM,nomeMedico,cpfMedico,telefoneMedico,emailMedico,codCidade,  
enderecoMedico,cepMedico)  
values (1,'Fabio Augusto Jose','412.123.123-21','3224-  
1234','doutor_augusto@hotmail.com',26,'Rua Verde','15023-310'),  
(2,'Antonio Leonardo da Silva ','321.123.345-33','3221-  
3453','doutor_leonardo@hotmail.com',23,'Rua Ceus','15031-221'),  
(3,'Luciana Maura da Silva','982.353.121-12','3432-  
1231','doutora_maura@hotmail.com',4,'Rua Aguimar','15043-134');
```

select *
from BDClinica.dbo.TBMedico

--criando tabela especialidade

Create table BDClinica.dbo.TBEspecialidade

```
(  
    idEspecialidade integer not null Primary key,  
    nomeEspecialidade varchar(30) not null  
);
```

insert into BDClinica.dbo.TBEspecialidade

```
(idEspecialidade,nomeEspecialidade)  
values (1,'Pediatria'),  
(2,'Ginecologista'),  
(3,'Cirurgião');
```

```
select*
from BDClinica.dbo.TBEspecialidade
```

```
-----
--criando tabela medico especialidade
Create table BDClinica.dbo.TBMedicoEspecialidade
(
    codCRM integer not null,
    codEsp integer not null,
    data datetime not null,
    --criando chave estrangeira CRM
    constraint FKcodCRM foreign key (codCRM)
    references BDClinica.dbo.TBMedico (idCRM),
    --criando chave estrangeira Especialidade
    constraint FKcodEsp foreign key (codEsp)
    references BDClinica.dbo.TBEspecialidade (idEspecialidade),
    --criando Primary key
    constraint PKcodCRMcodEsp Primary Key (codCRM,codEsp)
);
```

```
-----
insert into BDClinica.dbo.TBMedicoEspecialidade
(codCRM,codEsp,data)
values (1,2,'09/09/2000'),
       (2,3,'03/03/2001'),
       (3,1,'07/07/1999');
```

```
-----
select*
from BDClinica.dbo.TBMedicoEspecialidade
```

```
-----
-- criando tabela convenio
Create table BDClinica.dbo.TBConvenio
(
    idConvenio integer not null,
    nomeConvenio varchar(30) not null,
    --criando Primary Key
    constraint PKidConvenio Primary Key (idConvenio)
);
```

```
-----
insert into BDClinica.dbo.TBConvenio
(idConvenio,nomeConvenio)
values (1,'Seguro'),
       (2,'Convenio'),
       (3,'Safe');
```

```
-----
select*
from BDClinica.dbo.TBConvenio
```

-- criando tabela consulta

Create table BDClinica.dbo.TBConsulta

```
(  
    idConsulta integer not null Primary Key,  
    dataConsulta datetime not null,  
    horaConsulta char(5) not null,  
    valorConsulta decimal (10,2) not null,  
    codPaciente integer not null,  
    codCRM integer not null,  
    codEsp integer not null,  
    codFuncionario integer not null,  
    codConvenio integer not null,  
    -- criando chave estrangeira CodPaciente  
    constraint FKcodPaciente foreign key (codPaciente)  
    references BDClinica.dbo.TBPaciente (IDPaciente),  
    -- criando chave estrangeira CodCRM  
    constraint FKcodCRM2 foreign key (codCRM)  
    references BDClinica.dbo.TBMedico (idCRM),  
    --criando chave estrangeira Esp  
    constraint FKcodEsp2 foreign key (codEsp)  
    references BDClinica.dbo.TBEspecialidade (idEspecialidade),  
    --criando chave estrangeira codFuncionario  
    constraint FKcodFuncionario foreign key (codFuncionario)  
    references BDClinica.dbo.TBFuncionario (idFuncionario),  
    --criando chave estrangeira codConvenio  
    constraint FKcodconvenio foreign key (codConvenio)  
    references BDClinica.dbo.TBConvenio (idConvenio)  
);
```

```
-----  
insert into BDClinica.dbo.TBConsulta  
(idConsulta,dataConsulta,horaConsulta,valorConsulta,codPaciente,codCRM,  
codEsp,codFuncionario,codConvenio)  
values (1,'01/06/2010','13:00',30.00,3,1,2,1,3),  
       (2,'01/06/2010','13:00',30.00,2,3,1,3,1),  
       (3,'01/06/2010','13:00',30.00,1,2,3,2,2);  
-----
```

```
select *  
    from BDClinica.dbo.TBConsulta  
-----
```

Exercícios: fazer a inserção (cadastro) de mais alguns registros nas tabelas do Banco de Dados BDClinica.

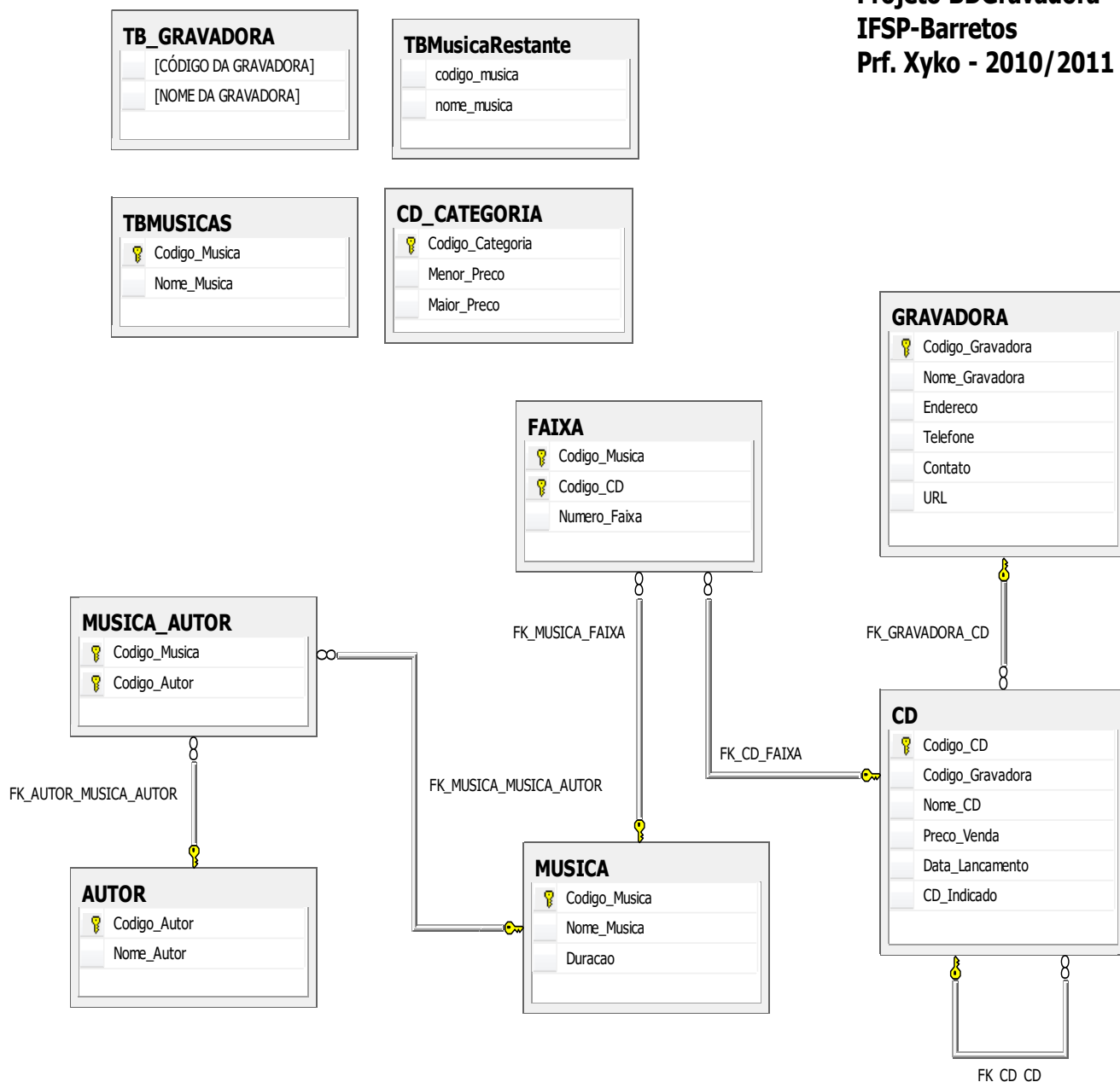
2.4 – Estudo de caso: Implementação do Projeto de banco de dados da BDGravadora a ser implementado no SQL Server de 2008 a 2016.

/*

Projeto Lógico e Físico do BDGravadora
Parte 1) Criação das tabelas com as Pks e as Fks
IFSP-Barretos – Data: junho / 2017.

*/

Projeto BDGravadora
IFSP-Barretos
Prf. Xyko - 2010/2011



Projeto, Modelagem e Gerenciamento de Banco de Dados.
Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

```
-- Criar o BD BDGravadora
Create Database BDGravadora;
-----

-- tornar o BDGravadora ativo
USE BDGravadora;
-----

-- criando a tabela Gravadora
CREATE TABLE GRAVADORA (
    Codigo_Gravadora    integer NOT NULL,
    Nome_Gravadora      VARCHAR(60) NULL,
    Endereco            VARCHAR(60) NULL,
    Telefone            VARCHAR(20) NULL,
    Contato             VARCHAR(20) NULL,
    URL                 VARCHAR(80) NULL
);
-----

-- modificando uma Tabela - Inserindo PK
ALTER TABLE GRAVADORA
    ADD CONSTRAINT PK_GRAVADORA PRIMARY KEY (Codigo_Gravadora);
-----

-- criando a tabela CD
CREATE TABLE CD
(
    Codigo_CD           integer NOT NULL,
    Codigo_Gravadora    integer NULL,
    Nome_CD             VARCHAR(60) NULL,
    Preco_Venda         decimal(14,2) NULL,
    Data_Lancamento    DATETIME NULL,
    CD_Indicado         integer NULL
);
-----

-- criando indice na Tabela
CREATE INDEX Ind_CD ON CD
(
    Codigo_Gravadora
);
-----

-- Alterando a tabela de CD
ALTER TABLE CD
    ADD CONSTRAINT PK_CD PRIMARY KEY (Codigo_CD);
-----

-- Criando a Tabela de Musica
CREATE TABLE MUSICA (
    Codigo_Musica       integer NOT NULL,
    Nome_Musica         VARCHAR(60) NULL,
    Duracao             decimal(6,2) NULL
);
```

-- alterando a tabela Musica

ALTER TABLE MUSICA

ADD CONSTRAINT PK_MUSICA PRIMARY KEY (Codigo_Musica);

-- criando a tabela Autor

CREATE TABLE AUTOR (

Codigo_Autor integer NOT NULL,
Nome_Autor VARCHAR(60) NULL

);

-- alterando a tabela autor - inserindo PK

ALTER TABLE AUTOR

ADD CONSTRAINT PK_AUTOR PRIMARY KEY (Codigo_Autor) ;

-- cria a tabela N:N Musica com Autor

CREATE TABLE MUSICA_AUTOR

(
Codigo_Musica integer NOT NULL,
Codigo_Autor integer NOT NULL
);

-- criando Indice na tabela Musica_AUTOR

CREATE INDEX IND_MUSICA_AUTOR ON MUSICA_AUTOR

(
Codigo_Musica
);

-- CRIA INDICE TABELA Musica_AUTOR

CREATE INDEX IND1_MUSICA_AUTOR ON MUSICA_AUTOR

(
Codigo_Autor
);

-- ALTERAR TABELA MUSICA_AUTOR - INSERE PK

ALTER TABLE MUSICA_AUTOR

ADD CONSTRAINT PK_MUSICA_AUTOR PRIMARY KEY (Codigo_Musica,
Codigo_Autor) ;

-- CRIA TABELA DE Faixa entre Musica e CD

CREATE TABLE FAIXA (

Codigo_Musica integer NOT NULL,
Codigo_CD integer NOT NULL,
Numero_Faixa integer NULL

);

```
-- cria indice na tabela Faixa
CREATE INDEX IND1_FAIXA ON FAIXA
(
    Codigo_Musica
);
```

```
-- deletando um Indice
drop index faixa.IND1_FAIXA;
```

```
-----
-- cria indice na tabela Faixa
CREATE INDEX IND2_FAIXA ON FAIXA
(
    Codigo_CD
);
```

```
-----
-- alterando a tabela Faixa - incluir PK - chave primaria composta
ALTER TABLE FAIXA
    ADD
    CONSTRAINT PK_FAIXA PRIMARY KEY (Codigo_Musica,
        Codigo_CD);
```

```
-----
-- Criando a tabela Categoria
CREATE TABLE CD_CATEGORIA(
    Codigo_Categoria integer NOT NULL,
    Menor_Precos decimal(14,2) NOT NULL,
    Maior_Precos decimal(14,2) NOT NULL
);
```

```
-----
-- alterando a tabela CD_categoria - incluir PK
alter table BDGravadora.dbo.cd_Categoria
add
    constraint pk_CD_Categoria primary key (Codigo_categoria);
```

```
-----
/*
criando as chaves estrangeiras
Criando os relacionamentos entre as tabelas do Banco de Dados BDGravadora
Sao as Foreign Key - FK
Como as tabelas já estão criadas, deve alterar as tabelas e inserir as FK
FK = Restricoes para manter a integridade das tabelas
*/
```

```
-----
-- altera a tabela CD
ALTER TABLE CD
    ADD
    CONSTRAINT FK_GRAVADORA_CD
    FOREIGN KEY (Codigo_Gravadora)
    REFERENCES GRAVADORA (codigo_gravadora);
```

```
-----
-- altera a tabela CD - inclui FK
ALTER TABLE CD
    ADD
    CONSTRAINT FK_CD_CD
        FOREIGN KEY (CD_Indicado)
            REFERENCES CD (codigo_cd);
-----
-- altera a atabela Musica_autor - inclui FK_autor
ALTER TABLE MUSICA_AUTOR
    ADD
    CONSTRAINT FK_AUTOR_MUSICA_AUTOR
        FOREIGN KEY (Codigo_Autor)
            REFERENCES AUTOR (codigo_autor);
-----
-- Nova FK na Tabela Musica_autor - Lembrar Relacionamento N:N
-- Tem um Chaveve primaria formada por duas Chaves estrangeiras
ALTER TABLE MUSICA_AUTOR
    ADD
    CONSTRAINT FK_MUSICA_MUSICA_AUTOR
        FOREIGN KEY (Codigo_Musica)
            REFERENCES MUSICA (Codigo_musica);
-----
-- Altera a tabela Faixa e inclui FK (duas)num unico Script
ALTER TABLE FAIXA
    ADD
    CONSTRAINT FK_CD_FAIXA
        FOREIGN KEY (Codigo_CD)
            REFERENCES CD(Codigo_cd),

    CONSTRAINT FK_MUSICA_FAIXA
        FOREIGN KEY (Codigo_Musica)
            REFERENCES MUSICA (codigo_musica);
-----
-- script = código sql para deletar/apagar uma restricao FK
-- não rode apenas para saber o comando
USE [BDGravadora]
GO
IF EXISTS
    (
        SELECT * FROM sys.foreign_keys
            WHERE object_id = OBJECT_ID(N'[dbo].[FK_CD_FAIXA]')
            AND parent_object_id = OBJECT_ID(N'[dbo].[FAIXA]')
    )
ALTER TABLE [dbo].[FAIXA] DROP CONSTRAINT [FK_CD_FAIXA]
go

```

-- mostra todas a PK e as FKs de uma Tabela

sp_helpconstraint faixa;

-- script para deletar uma PK - chave primaria

-- Não rodar – só pra estudo

USE [BDGravadora]

GO

/**** Object: Index [PK_FAIXA] Script Date: 08/12/2009 15:03:36 *****/**

IF EXISTS

(

SELECT * FROM sys.indexes

WHERE object_id = OBJECT_ID(N'[dbo].[FAIXA]')

AND name = N'PK_FAIXA'

)

ALTER TABLE [dbo].[FAIXA] DROP CONSTRAINT [PK_FAIXA]

/*

Parte 2) Script para inserir dados nas Tabelas

Etec Philadelpho Gouvea Netto

Prof. Xyko

2009

***/**

-- ABRINDO O BDGravadora

USE BDGravadora;

-- TABELA DE AUTOR

INSERT INTO AUTOR (CODIGO_AUTOR, NOME_AUTOR)

VALUES (1, 'Renato Russo');

(2, 'Tom Jobim'), (3, 'Chico Buarque'), (4, 'Dado Villa-Lobos'),

(5, 'Marcelo Bonfá'), (6, 'Ico Ouro-Preto'), (7, 'Vinicius de Moraes'),

(8, 'Baden Powell'), (9, 'Paulo Cesar Pinheiro'), (10, 'João Bosco'),

(11, 'Aldir Blanc'), (12, 'Joyce'),

(13, 'Ana Terra'), (14, 'Cartola'),

(15, 'Cláudio Tolomei'), (16, 'João Nogueira'),

(17, 'Sueley Costa'), (18, 'Guinga'),

(19, 'Danilo Caymmi'), (20, 'Tunai'),

(21, 'Sérgio Natureza'), (22, 'Heitor Villa Lobos'),

(23, 'Ferreira Gullar'), (24, 'Catulo da Paixão Cearense'),

(25, 'Zezé di Camargo'), (26, 'Niltinho Edilberto'),

(27, 'Marisa Monte'), (28, 'Carlinhos Brown'),

(29, 'Gonzaga Jr'), (30, 'Roberto Mendes'),

(31, 'Ana Basbaum'), (32, 'Caetano Veloso'),

(33, 'José Miguel Wisnik'), (34, 'Vevé Calazans'),

(35, 'Gerônimo'), (36, 'Sérgio Natureza'),

(37, 'Roberto Carlos'), (38, 'Erasmus Carlos'),

Projeto, Modelagem e Gerenciamento de Banco de Dados.
Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

```
( 39, 'Renato Teixeira' ),      ( 40, 'Chico César' ),
( 41, 'Vanessa da Mata' ),    ( 42, 'Jorge Portugal' ),
( 44, 'Renato Barros' ),      ( 45, 'Bebel Gilberto' ),
( 46, 'Cazuza' ),            ( 47, 'Dé' ),
( 48, 'Adriana Calcanhoto' ), ( 49, 'Antonio Cícero' ),
( 50, 'Paulo Machado' ),      ( 51, 'Dorival Caymmi' ),
( 52, 'João Donato' ),        ( 53, 'Ronaldo Bastos' ),
( 54, 'Barry Manilow' ),      ( 55, 'Richard Kerr' ),
( 56, 'Chris Arnold' ),       ( 57, 'David Pomeranz' ),
( 58, 'George Michael' ),     ( 59, 'S. Wonder' ),
( 60, 'Elton John' ),         ( 61, 'Arnaldo Antunes' );
```

-- verificando os dados inseridos

```
select *
from autor;
```

-- TABELA DE MÚSICA

-- rodar este bloco de uma única vez

```
INSERT INTO MUSICA (CODIGO_MUSICA, NOME_MUSICA, DURACAO )
VALUES ( 1, 'Será', 2.28 ),      ( 2, 'Ainda é Cedo', 3.55 ),
( 3, 'Geração Coca-Cola', 2.20 ), ( 4, 'Eduardo e Monica', 4.32 );
( 5, 'Tempo Perdido', 5.00 ),    ( 6, 'Índios', 4.23 ),
( 7, 'Que País é Este', 2.64 ),  ( 8, 'Faroeste Caboclo', 9.03 ),
( 9, 'Há Tempos', 3.16 );
```

-- rodar este bloco de uma única vez

```
INSERT INTO MUSICA (CODIGO_MUSICA, NOME_MUSICA, DURACAO )
VALUES ( 10, 'País e Filhos', 5.06 ), ( 11, 'Meninos e Meninas', 3.22 ),
( 12, 'Vento no Litoral', 6.05 ),    ( 13, 'Perfeição', 4.35 );
( 14, 'Giz', 3.20 ),                  ( 15, 'Dezesseis', 5.28 ),
( 16, 'Antes das Seis', 3.09 ),        ( 17, 'Meninos, Eu Vi', 3.25 ),
( 18, 'Eu Te Amo', 3.06 ),            ( 19, 'Piano na Mangueira', 2.23 ),
( 20, 'A Violeira', 2.54 ),           ( 21, 'Anos Dourados', 2.56 ),
( 22, 'Olha, Maria', 3.55 ),          ( 23, 'Biscate', 3.20 ),
( 24, 'Retrato em Preto e Branco', 3.03 ), ( 25, 'Falando de Amor', 3.20 ),
( 26, 'Pois É', 2.48 ),               ( 27, 'Noite dos Mascarados', 2.42 );
```

-- rodar este bloco de uma única vez

```
INSERT INTO MUSICA (CODIGO_MUSICA, NOME_MUSICA, DURACAO )
VALUES ( 28, 'Sabiá', 3.20 ),
( 29, 'Imagina', 2.52 ),
( 30, 'Bate-Boca', 4.41 ),
( 31, 'Cai Dentro', 2.41 ),
( 32, 'O Bêbado e o Equilibrista', 3.47 ),
( 33, 'Essa Mulher', 3.47 ),
( 34, 'Basta de Clamores Inocência', 3.38 ),
( 35, 'Beguíne Dodói', 2.14 );
```

```
( 36, 'Eu hein Rosa', 3.36 ),  
( 37, 'Altos e Baixos', 3.29 ),  
( 38, 'Bolero de Satã', 3.32 ),  
( 39, 'Pé Sem Cabeça', 2.57 ),  
( 40, 'As Aparências Enganam', 4.18 ),  
( 41, 'É o Amor', 4.19 ),  
( 42, 'Trenzinho Caipira', 3.32 ),  
( 43, 'Luar do Sertão', 3.23 ),  
( 44, 'Não tenha Medo', 3.27 ),  
( 45, 'Eu queria que você viesse', 2.57 ),  
( 46, 'Espere por mim Morena', 3.04 ),  
( 47, 'Resto de mim', 2.59 ),  
( 48, 'Gema', 2.51 ),  
( 49, 'Cacilda', 2.22 ),  
( 50, 'Agradecer e abraçar', 3.30 ),  
( 51, 'As flores do jardim da nossa casa', 3.26 ),  
( 52, 'Romaria', 3.16 );
```

-- rodar este bloco de uma única vez

```
INSERT INTO MUSICA (CODIGO_MUSICA, NOME_MUSICA, DURACAO )  
VALUES ( 53, 'A força que nunca seca', 2.17 ),  
( 54, 'Vila do Adeus', 3.06 ),  
( 55, 'Devolva-me', 3.58 ),  
( 56, 'Mais Feliz', 2.50 );
```

-- rodar este bloco de uma única vez

```
INSERT INTO MUSICA (CODIGO_MUSICA, NOME_MUSICA, DURACAO )  
VALUES ( 57, 'Inverno', 4.40 ),  
( 58, 'Mentiras', 2.58 ),  
( 59, 'Esquadros', 3.10 ),  
( 60, 'Cariocas', 3.14 ),  
( 61, 'Vambora', 4.16 ),  
( 62, 'Por isso eu Corro Demais', 2.58 ),  
( 63, 'Maresia', 4.09 ),  
( 64, 'Metade', 3.25 ),  
( 65, 'Senhas', 3.37 ),  
( 66, 'Marina', 2.55 ),  
( 67, 'Naquela Estação', 4.46 ),  
( 68, 'Mandy', 3.18 ),  
( 69, 'New York City Rhythm', 4.41 ),  
( 70, 'Looks Like We Made It', 3.32 ),  
( 71, 'Daybreak', 3.05 ),  
( 72, 'Can't Smile Without you', 3.13 ),  
( 73, 'It's a Miracle', 3.53 ),  
( 74, 'Even Now', 3.29 ),  
( 75, 'Bandstand Boogie', 2.50 ),  
( 76, 'Trying to get the feeling again', 3.50 ),
```

```
( 77, 'Some Kind of Friend', 4.02 ),  
( 78, 'Praying for Time', 3.52 ),  
( 79, 'Freedom 90', 3.52 ),  
( 80, 'They Won't Go When I Go', 3.22 ),  
( 81, 'Something to Save', 4.10 ),  
( 82, 'Cowboys and Angels', 4.12 ),  
( 83, 'Don't Let the Sun Go Down on Me', 3.45 ),  
( 84, 'Waiting for That Day', 2.58 ),  
( 85, 'Mothers Pride', 2.12 ),  
( 86, 'Heal the Pain', 3.02 ),  
( 87, 'Soul Free', 2.42 ),  
( 88, 'Waiting', 3.32 ),
```

```
-----  
-- conferindo os registros inseridos na tabela Musuca  
select * from musica;
```

```
-----  
-- TABELA GRAVADORA  
INSERT INTO GRAVADORA  
(CODIGO_GRAVADORA, NOME_GRAVADORA, ENDERECO, URL, CONTATO )  
VALUES ( 1, 'EMI', 'Rod. Pres. Dutra, s/n – km 229,8', 'www.emi-music.com.br',  
'JOÃO' );  
( 2, 'BMG', 'Av. Piramboia, 2898 - Parte 7', 'www.bmg.com.br', 'MARIA' );  
( 3, 'SOM LIVRE', NULL, 'www.somlivre.com.br', 'MARTA' );  
( 4, 'EPIC', NULL, 'www.epic.com.br', 'PAULO' );  
( 4040, 'XykoCorporation', NULL, 'www.Xyko.com.br', 'Sr. Xykao' );  
-----
```

```
-- Testando a insercao dos registros na Tabela Gravadora  
select * from Gravadora;
```

```
-----  
-- TABELA CD  
INSERT INTO CD (CODIGO_CD, CODIGO_GRAVADORA, NOME_CD, PRECO_VENDA,  
DATA_LANCAMENTO )  
VALUES ( 1, 1, 'Mais do Mesmo', 15.00, '01/10/1998' );  
INSERT INTO CD (CODIGO_CD, CODIGO_GRAVADORA, NOME_CD, PRECO_VENDA,  
DATA_LANCAMENTO )  
VALUES ( 2, 2, 'Bate-Boca', 12.00, '01/07/1999' );  
INSERT INTO CD (CODIGO_CD, CODIGO_GRAVADORA, NOME_CD, PRECO_VENDA,  
DATA_LANCAMENTO )  
VALUES ( 3, 3, 'Elis Regina - Essa Mulher', 13.00, '01/05/1989' );  
INSERT INTO CD (CODIGO_CD, CODIGO_GRAVADORA, NOME_CD, PRECO_VENDA,  
DATA_LANCAMENTO )  
VALUES ( 4, 2, 'A Força que nunca Seca', 13.50, '01/12/1998' );  
INSERT INTO CD (CODIGO_CD, CODIGO_GRAVADORA, NOME_CD, PRECO_VENDA,  
DATA_LANCAMENTO )  
VALUES ( 5, 3, 'Perfil', 10.50, '01/05/2001' );  
INSERT INTO CD (CODIGO_CD, CODIGO_GRAVADORA, NOME_CD, PRECO_VENDA,  
DATA_LANCAMENTO )
```



```
VALUES ( 6, 2, 'Barry Manilow Greatest Hits Vol I', 9.50, '01/11/1991' );  
INSERT INTO CD (CODIGO_CD, CODIGO_GRAVADORA, NOME_CD, PRECO_VENDA,  
DATA_LANCAMENTO )  
VALUES ( 7, 2, 'Listen Without Prejudice', 9.00, '01/10/1991' );
```

```
-----  
-- mostrando a conteudo dos CDs  
select * from cd;
```

```
-----  
-- atualizando, modificando o conteudo de alguns registro  
UPDATE CD  
SET CD_INDICADO = 5  
WHERE CODIGO_CD = 1;
```

```
-----  
UPDATE CD  
SET CD_INDICADO = 3  
WHERE CODIGO_CD = 2;
```

```
-----  
UPDATE CD  
SET CD_INDICADO = 1  
WHERE CODIGO_CD = 3;
```

```
-----  
UPDATE CD  
SET CD_INDICADO = 2  
WHERE CODIGO_CD = 5;
```

```
-----  
UPDATE CD  
SET CD_INDICADO = 7  
WHERE CODIGO_CD = 6;
```

```
-----  
UPDATE CD  
SET CD_INDICADO = 1  
WHERE CODIGO_CD = 4;
```

```
-----  
-- TABELA ITEMCD = Faixa  
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )  
VALUES ( 1, 1, 1 );  
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )  
VALUES ( 1, 2, 2 );  
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )  
VALUES ( 1, 3, 3 );  
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )  
VALUES ( 1, 4, 4 );  
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )  
VALUES ( 1, 5, 5 );  
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )  
VALUES ( 1, 6, 6 );  
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
```

```
VALUES ( 1, 7, 7 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 1, 8, 8 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 1, 9, 9 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 1, 10, 10 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 1, 11, 11 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 1, 12, 12 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 1, 13, 13 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 1, 14, 14 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 1, 15, 15 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 1, 16, 16 );
-- BATE-BOCA
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 2, 1, 17 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 2, 2, 18 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 2, 3, 19 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 2, 4, 20 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 2, 5, 21 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 2, 6, 22 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 2, 7, 23 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 2, 8, 24 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 2, 9, 25 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 2, 10, 26 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 2, 11, 27 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 2, 12, 28 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 2, 13, 29 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
```

```
VALUES ( 2, 14, 30 );
-- ESSA MULHER
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 3, 1, 31 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 3, 2, 32 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 3, 3, 33 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 3, 4, 34 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 3, 5, 35 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 3, 6, 36 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 3, 7, 37 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 3, 8, 38 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 3, 9, 39 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 3, 10, 40 );
-- FORÇA QUE NUNCA SECA
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 4, 1, 41 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 4, 2, 42 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 4, 3, 43 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 4, 4, 44 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 4, 5, 45 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 4, 6, 46 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 4, 7, 47 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 4, 8, 48 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 4, 9, 49 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 4, 10, 50 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 4, 11, 51 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 4, 12, 52 );
```

```
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 4, 13, 53 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 4, 14, 54 );
-- DEVOLVA-ME
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 5, 1, 55 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 5, 2, 56 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 5, 3, 57 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 5, 4, 58 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 5, 5, 59 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 5, 6, 60 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 5, 7, 61 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 5, 8, 62 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 5, 9, 63 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 5, 10, 64 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 5, 11, 65 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 5, 12, 66 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 5, 13, 67 );
-- BARRY MANILOW
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 6, 1, 68 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 6, 2, 69 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 6, 3, 70 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 6, 4, 71 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 6, 5, 72 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 6, 6, 73 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 6, 7, 74 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
```

```
VALUES ( 6, 8 ,75 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 6, 9, 76 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 6, 10, 77 );
-- LISTEN WITHOUT PREJUDICE
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 7, 1, 78 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 7, 2, 79 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 7, 3, 80 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 7, 4, 81 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 7, 5, 82 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 7, 6, 83 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 7, 7, 84 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 7, 8 ,85 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 7, 9, 86 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 7, 10, 87 );
INSERT INTO FAIXA (CODIGO_CD, NUMERO_FAIXA, CODIGO_MUSICA )
VALUES ( 7, 11, 88 );
```

```
-----
-- checando os registros inseridos
select * from faixa;
```

```
-----
-- TABELA MUSICA_AUTOR
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 1, 1 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 2, 5 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 2, 6 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 3, 4 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 3, 5 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 3, 1 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
```

```
VALUES ( 4, 1 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 4, 4 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 5, 1 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 6, 1 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 7, 1 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 8, 1 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 9, 1 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 9, 4 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 9, 5 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 10, 1 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 10, 4 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 10, 5 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 11, 1 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 11, 4 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 11, 5 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 12, 1 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 12, 4 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 12, 5 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 13, 1 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 13, 4 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 13, 5 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 14, 1 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 14, 4 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 14, 5 );
```

```
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 15, 1 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 15, 4 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 15, 5 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 16, 1 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 16, 4 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 16, 5 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 17, 2 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 17, 3 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 18, 2 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 18, 3 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 19, 2 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 19, 3 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 20, 2 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 20, 3 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 21, 2 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 21, 3 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 22, 2 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 22, 3 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 22, 7 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 23, 3 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 24, 2 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 24, 3 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 25, 2 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
```



```
VALUES ( 26, 2 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 26, 3 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 27, 3 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 28, 2 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 28, 3 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 29, 2 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 29, 3 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 30, 3 );
-- ESSA MULHER
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 31, 8 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 31, 9 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 32, 10 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 32, 11 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 33, 12 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 33, 13 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 34, 14 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 35, 10 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 35, 11 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 35, 15 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 36, 16 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 36, 9 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 37, 17 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 37, 11 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 38, 18 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
```

```
VALUES ( 38, 9 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 39, 19 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 39, 13 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 40, 20 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 40, 21 );
-- A FORÇA QUE NUNCA SECA
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 41, 25 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 44, 26 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 45, 27 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 45, 28 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 46, 29 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 47, 30 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 47, 31 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 48, 32 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 49, 33 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 50, 35 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 50, 36 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 51, 37 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 51, 38 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 52, 39 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 53, 40 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 53, 41 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 54, 30 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 54, 42 );
-- DEVOLVA-ME
```

```
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 55, 43 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 55, 44 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 56, 45 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 56, 46 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 56, 47 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 57, 48 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 57, 49 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 58, 48 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 59, 48 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 60, 48 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 61, 48 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 62, 37 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 63, 50 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 63, 49 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 64, 48 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 65, 48 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 66, 51 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 67, 52 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 67, 53 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 67, 32 );
-- BARRY MANILOW
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 68, 55 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 69, 54 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 70, 55 );
```

```
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 71, 54 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 72, 56 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 73, 54 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 74, 54 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 75, 54 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 76, 57 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 77, 54 );
-- LISTEN WITHOUT PREJUDICE
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 78, 58 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 79, 58 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 80, 59 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 81, 58 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 82, 58 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 83, 60 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 84, 58 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 85, 58 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 86, 58 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 87, 58 );
INSERT INTO MUSICA_AUTOR (CODIGO_MUSICA, CODIGO_AUTOR )
VALUES ( 88, 58 );
-----
-- checando a tabela Musica_autor
select * from musica_autor;
-----
-- TABELA CD_CATEGORIA
INSERT INTO CD_CATEGORIA (CODIGO_CATEGORIA, MENOR_PRECO,
MAIOR_PRECO)
VALUES ( 1, 5, 10);
INSERT INTO CD_CATEGORIA (CODIGO_CATEGORIA, MENOR_PRECO,
MAIOR_PRECO)
```

```
VALUES ( 2, 10.01, 12);
INSERT INTO CD_CATEGORIA (CODIGO_CATEGORIA, MENOR_PRECO,
MAIOR_PRECO)
VALUES ( 3, 12.01, 15);
INSERT INTO CD_CATEGORIA (CODIGO_CATEGORIA, MENOR_PRECO,
MAIOR_PRECO)
VALUES ( 4, 15.01, 20);
```

```
-----
-- checando categorias
select * from CD_Categoria;
```

```
-----
/*
MOSTRA INFORMAÇÕES DA TABELA DAS CONSTRAINTS
*/
USE BDGRAVADORA;
```

```
-----
SELECT CONSTRAINT_NAME AS          [NOME DA RESTRIÇÃO],
       TABLE_NAME      AS          [TABELA],
       CONSTRAINT_TYPE AS          [TIPO DA CONSTRAINT]
FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS
WHERE TABLE_NAME = 'CD';
```

```
-----
/*
MOSTRA- LISTA AS TABELAS DE UM BANCO
AS TABELAS REAIS E AS VIRTUAIS - VIEW
*/
```

```
SELECT TABLE_NAME AS [NOME DA TABELA]
FROM INFORMATION_SCHEMA.TABLES;
```

```
-----
/* 1) Mostrar o Código e o o nome dos autores cadastrados
que não gravaram musicas.
```

```
*/
Select distinct a.codigo_autor as [Código Autor],
       a.nome_autor as [Nome do Autor]
from   BDGravadora.dbo.autor as a,
       BDGravadora.dbo.musica_autor as ma
where  a.codigo_autor not in
      (
        Select codigo_autor
        from BDGravadora.dbo.musica_autor
      );
```

```
-----
/*
2) Mostrar codigo, nome dos autores e o total de musicas gravados por cada autor
```

```
*/
```

```
Select          a.codigo_autor as [Código Autor],
                  a.Nome_Autor as [Nome do Autor],
                  count (ma.codigo_autor) as [Quantidade de Músicas
Gravadas]
from  BDgravadora.DBO.Autor as a,
      BDgravadora.DBO.Musica_Autor as ma
where  a.codigo_autor = ma.codigo_autor
group by  a.codigo_autor,
          a.Nome_Autor
```

```
-----
/*
-- 3) Mostrar Código, nome das musicas não foram gravadas por autores

*/
```

```
Select          distinct m.codigo_musica as [Código musica],
                    m.nome_musica as [Nome da musica]
from            BDgravadora.dbo.musica as m,
                BDgravadora.dbo.musica_autor as ma
where          m.codigo_musica not in
(
    Select codigo_musica
    from BDgravadora.dbo.musica_autor
);
```

```
-- Testando o código acima
select * from BDgravadora.dbo.musica;
```

```
-----
/*
-- 4) Mostrar Código, nome da Musica e
    quantas vezes cada música foi gravada pelos autores
```

```
*/
Select          m.codigo_musica as [Código da Musica],
                  m.Nome_Musica as [Nome da Musica],
                  count (ma.codigo_autor) as [Quantidade de vezes que a
música foi gravada pelos autores]

from            BDgravadora.dbo.musica as m,
                BDgravadora.dbo.musica_autor as ma
where (m.codigo_musica = ma.codigo_musica)
Group by  m.codigo_musica,
          m.Nome_Musica
```

```
-----
/*
5) Mostrar Código, nome da Musica e quantas vezes cada música foi gravada pelos
autores,
mas somente para as musicas com duas ou mais gravações
```

***/**

**Select m.codigo_musica as [Código da Musica],
 m.Nome_Musica as [Nome da Musica],
 count (ma.codigo_autor) as [Quantidade de vezes que a música foi gravada
pelos autores]**

**from BDgravadora.dbo.musica as m,
 BDgravadora.dbo.musica_autor as ma
where (m.codigo_musica = ma.codigo_musica)
Group by m.codigo_musica,
 m.Nome_Musica
having count (ma.codigo_autor)>=2
order by 3;**

/*

**6) mostrar código, nome da musica e
a Quantide de CDs que a Musica foi Gravada
*/**

**Select m.codigo_musica as [Código da Musica],
 m.Nome_Musica as [Nome da Musica],
 count (f.codigo_musica) as [Quantidade de vezes
que a música foi gravada por CD]**

**from BDgravadora.dbo.musica as m,
 BDgravadora.dbo.faixa as f
where (m.codigo_musica = f.codigo_musica)
Group by m.codigo_musica,
 m.Nome_Musica**

/*

**7) mostrar código, nome da musica e
a Quantide de CDs que a Musica foi Gravada mais de uma vez
*/**

**Select m.codigo_musica as [Código da Musica],
 m.Nome_Musica as [Nome da Musica],
 count (f.codigo_musica)
 as [Quantidade de vezes que a música foi gravada por CD]**

**from BDgravadora.dbo.musica as m,
 BDgravadora.dbo.faixa as f
where (m.codigo_musica = f.codigo_musica)
Group by m.codigo_musica,
 m.Nome_Musica
having count (f.codigo_musica) >1;**

-- MOSTRANDO ALGUMAS FUNÇÕES

SELECT
 PI() AS [PI],
 SQRT(25) AS [Raiz Quadrada de 25],
 power(4,3) as [Cubo de quatro];

-- criando uma função

Create function fcubo (@num int)
 returns int
 begin
 set @num = (@num * @num * @num);
 return @num
 end

-- mostrando o resultado da função - executando

Select dbo.fcubo(5) as [cubo de 5];
Select dbo.fcubo(2) as [cubo de 2];
Select dbo.fcubo(4) as [cubo de 4];

-- cria uma função que retorna os dado de um autor específico

Create function fautor6 (@idautor int)
 returns table -- retorna uma tabela
 as
 return (
 select *
 from BDGRAVADORA.dbo.autor
 where codigo_autor = @idautor
)

-- Mostrando o dados do Autor 5

Select * from dbo.fautor6(10);

-- criando uma função que retorna os dados do autor

CREATE function FBuscaAutor()
 Returns @cli table (
 Idautor varchar(10),
 Nome varchar(30)
)

 as
 begin
 insert into @cli select codigo_autor,
 nome_autor
 from BDGRAVADORA.dbo.autor

 return
 end

-- testando

Select *

from FBuscaAutor();

**/* Usando o Banco de Dados Pubs e
comando case**

***/**

/*

Parte 3)

Criando Tabelas usando o Comando SELECT INTO Nova_Tabela

***/**

/*

**cria uma tabela chamada TBMusicas para gravar somente as musucas que
comecam de a-d. Nota que a insercao dos dados sera efetuada com um comando
select Into
que busca dados na Tabela Musica**

USO DO:

INSERT INTO

SEGUIDO DE UM SELECT

***/**

USE BDGRAVADORA;

**/* SELECT CAMPO1, CAPO2
INTO TABELA_NOVA
FROM
TABELA_JA_EXISTENTE**

CRIA A TABELA COM O SELECT

OBSERVAR QUE TBGRAVADORA NÃO EXISTE

***/**

**SELECT CODIGO_GRAVADORA AS [CÓDIGO DA GRAVADORA],
NOME_GRAVADORA AS [NOME DA GRAVADORA]
INTO TB_GRAVADORA -- AQUI CRIA A TABELA
FROM GRAVADORA
WHERE NOME_GRAVADORA LIKE '%X%'
OR NOME_GRAVADORA LIKE '%S%';**

-- VEJA O CONTEÚDO DA NOVA TABELA

SELECT * FROM TB_GRAVADORA;

/*

**AQUI CRIAMOS UMA TABELA TEMPORÁRIA #TABELA_TEMPORARIA
USO DO SELECT PARA INSERIR DADOS NA TABELA TEMPORÁRIA
AQUI TEMOS QUE CRIAR A TABELA TEMPORARIA ANTES DO SELECT**

***/**

-- CRIANDO A TABELA TEMPORARIA

CREATE TABLE #TB_TEMPORARIA

(

**IDALUNO INT NOT NULL PRIMARY KEY,
 NOME VARCHAR(50) NOT NULL**

);

-- VERIFICA A TABELA #TBTEMPORARIA NO SYSTEM DATABASES TEMPDB

INSERT #TB_TEMPORARIA

**SELECT CODIGO_AUTOR AS [IDAUTOR],
 NOME_AUTOR AS [NOME]**

FROM AUTOR

WHERE NOME_AUTOR LIKE '[J-W]%'

ORDER BY 2

-- LISTAR O CONTEÚDO DA TABELA #TB_TEMPORARIA

SELECT * FROM

#TB_TEMPORARIA;

-- cria a tabela TBMusicas - somente com dois campos

CREATE TABLE BDGravadora.dbo.TBMUSICAS (

Codigo_Musica integer NOT NULL,

Nome_Musica VARCHAR(60) NULL,

-- criando a PK

constraint pkCodMusica primary Key (codigo_Musica)

);

-- Inserindo dados na Tabela TBMusicas com dados da Tabela Musica

-- compondo o insert e select

-- Maneira 1)

insert into BDGravadora.dbo.TBMusicas

select codigo_musica,

nome_musica

from BDgravadora.dbo.musica as m

where m.nome_musica like '[a-d]%';

-- verificando o conteudo da tabela TBMusucas recém criada e que

-- foi copiado os dados da tabela Musica com filtro [so as com nome de a-d]

select * from TBMusicas;

-- outra forma de Usar o Select Into para criar uma tabela
-- Maneira 2) - A tabela que recebe os dados e criada na hora
-- nao precisa criar a nova tabela TBmusicasRestante
select codigo_musica,
nome_musica
into BDGravadora.dbo.TBMusicaRestante
from BDGravadora.dbo.musica
where nome_musica like '[e-y]%'

-- checando
select *
from BDGravadora.dbo.tbmusicaRestante
order by 2;

/*

Parte 4)

T-SQL - Programando o Banco de Dados BDGravadora
View e Funcao

Etec Philadelpho Gouvea Netto
Prof. Xyko
2009

***/**

/*

4.1) Criando funcoes escalares
funcoes que retornam valores

***/**

-- fica como exercícos práticos

/*

4.2) Criando funcoes Table (que retornam tabelas)
funcoes que retornam tabelas

***/**

create function f_MostraAutor (@codigo int)
returns table
as
return
(
select codigo_autor,
nome_autor
from bdgravadora.dbo.autor

```
        where @codigo =Codigo_Autor
    )
-- checando a funcao - executando
-- tem passar um parametro para a pesquisa
-- retorna os dados do autor passado como parametro
Select * from dbo.f_MostraAutor(4);
-----
-- Criar uma funcao que mostrando dados dos CDs Gravados
-- de uma gravadora que e passada como parametro
Create function f_Quantidade_cds_Gravados (@Cod_CD as int)
returns table
as
return (
    select c.Codigo_gravadora as [Codigo da gravadora],
           g.nome_gravadora as [Nome da Gravadora],
           count (*) as [Quantidade de Cds Gravados]
    from gravadora as g,
         cd as c
    where convert (char(10),c.codigo_gravadora) =
           convert(char(10), g.codigo_gravadora)
           and @cod_cd = c.codigo_Gravadora
    group by c.codigo_gravadora, g.Nome_gravadora
)
-----
-- executa a funcao
Select * from F_Quantidade_cds_Gravados(2);
-----
/*
```

Parte 5) Criando VIEW - QUE SAO TABELAS VIRTUAIS
NAO OCUPAM ESPACO NO HD
EXISTEM APENAS EM TEMPO DE EXECUCAO
no nome das visoes comecam com v de view - visao

sao codigos sql de select que fica gravado no BD e podem ser reutilizados

```
*/
-- mostrando quantidade de CDs gravados de
-- todas as Gravadoras
-- nao precisa passar parametros para a pesquisa

create VIEW V_Mostra_Quantidade_cds_Gravados
as
select c.Codigo_gravadora as [Codigo da gravadora],
       g.nome_gravadora as [Nome da Gravadora],
       count (*) as [Quantidade de Cds Gravados]
from gravadora as g,
     cd as c
where c.codigo_gravadora = g.codigo_gravadora
```

```
group by c.codigo_gravadora, g.Nome_gravadora  
GO
```

```
-----  
-- executa a visão  
Select * from v_Mostra_Quantidade_cds_Gravados;  
-----
```

Observação: é importante entender e fazer uso de consultas em SQL pois como visto nos exemplos práticos dos estudos de caso, as consultas são a base para visões (*view*), funções e *procedures* armazenadas no SQL

2.5 – Estudo de Caso - criação do Projeto do BDBrasil (Nível mais avançado) a ser implementado no SQL Server 2008 a 2016.

 INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SÃO PAULO	ESTUDO DE CASO	
	CURSO: Treinamento em SQL SERVER 2008 a 2016	
	IFSP-BARRETOS	
PROFESSOR : Francisco Antonio de Almeida - Xyko		
Módulo: inicial ou avançado		

EXERCÍCIO PRÁTICO DE BANCO DE DADOS

Estado	Sigla	Região	População	Economia	Clima	Cidade
Bahia	BA	Nordeste	11.855.157	Agricultura	Semi-arido Quente	Salvador
Piauí	PI	Nordeste	2.581.054	Pecuária	Quente	Bom Jardim
Rio de Janeiro	RJ	Sudeste	4.870.450	Industria	Tropical	Bom Sucesso
Rio de Janeiro	RJ	Sudeste		Serviços	Tropical	Rio Janeiro
Minas Gerais	MG	Sudeste	3.780.457	Industria	Tropical	Belo Horizonte
Minas Gerais	MG	Sudeste		Agropecuária	Tropical	Uberaba
Roraima	RR	Norte	215.790	Pecuária	Quente Úmido	Boa Vista
Sergipe	SE	Nordeste	1.492.400	Agricultura	Semi-Árido Quente	Aracaju
Alagoas	AL	nordeste	2.512.991	Agricultura	Úmido	Maceió
Paraná	PR	Sul	8.415.654	Agricultura	Tropical	Curitiba
Goiás	GO	Centro-Oeste		Pecuária	Quente	Goiatuba
Amazonas	AM	Norte	2.102.901	Industria	Equatorial	Manaus
Tocantins	TO	Centro-Oeste	1.007.410	Pecuária	Tropical	Palmas
São Paulo	SP	Sudeste	9.8111.776	Agricultura	Tropical	Olímpia
São Paulo	SP	Sudeste	9.8111.776	Industria	Tropical	Guapiaçu

- 1) (VALOR 6,00). Esta tabela com informações sobre os Estados brasileiros com suas características foi desenvolvida por alunos em uma aula de Geografia da **UNIESQUINA**. Partindo do conceito que a mesma Tabela está sendo solicitada a você, um projetista de Banco de Dados da **Fatec Rio Preto ou do IFSP**, **escreva todo código SQL** que cria as tabelas, os relacionamentos, as chaves primárias e as chaves estrangeiras – fazer o projeto Físico do **BDBrasil**.

Observações importantes para Resolução do Trabalho:

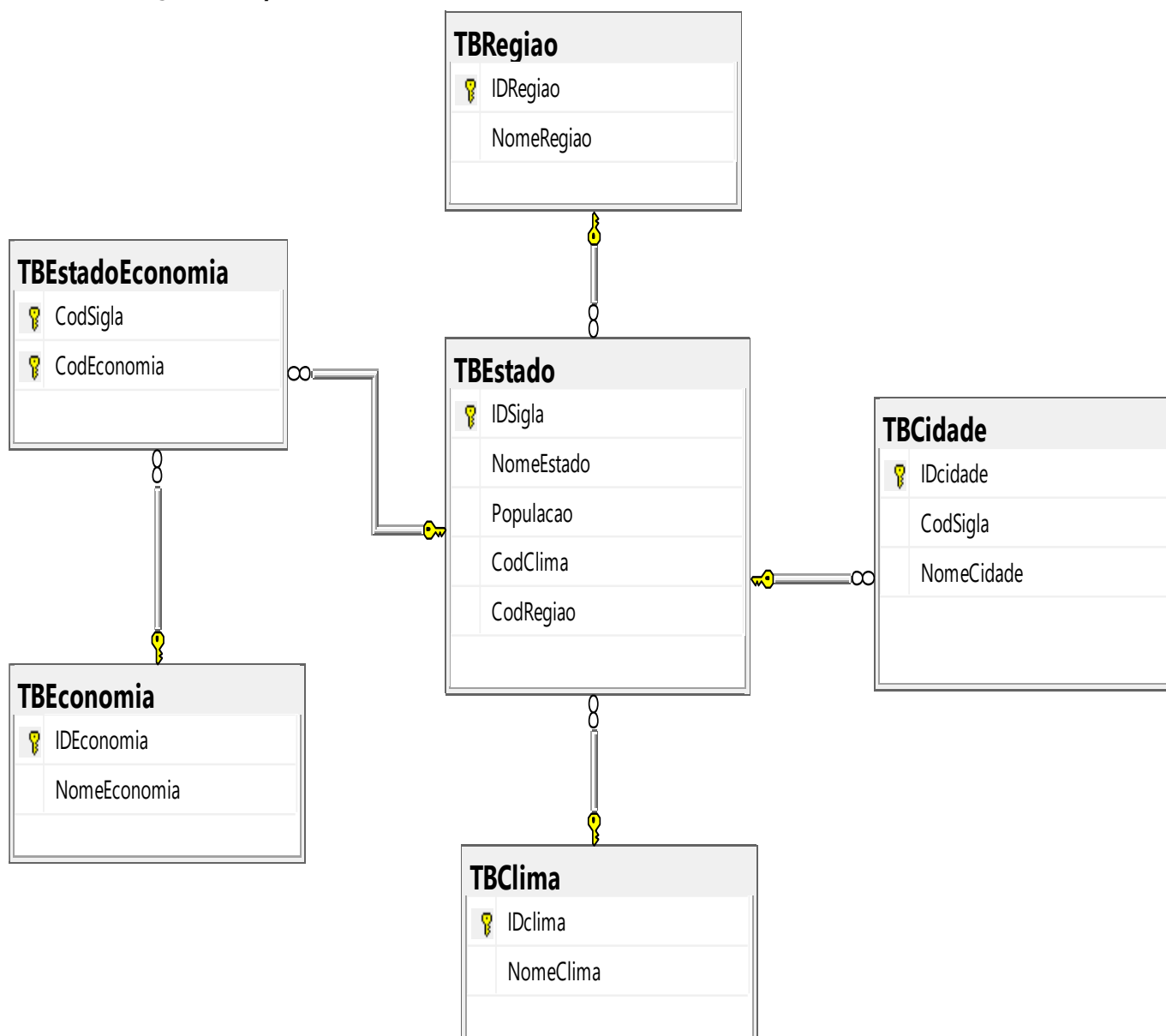
- A ideia é que o aluno consiga separar as Entidades/Tabelas e fazer os relacionamentos;**
- Observar que algumas colunas serão campos das tabelas, enquanto outras serão tabelas.**
- Observar que um Estado tem mais de uma Economia e uma Economia está em mais de um Estado – (Relacionamento N:N – que deve ser tratado na criação das Tabelas).**
- A população é uma característica do Estado e não da Região.**

Projeto, Modelagem e Gerenciamento de Banco de Dados.
Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

- 2) (Valor 2,00). Insira todos os registros acima nas respectivas tabelas criadas no item 1) usando a linguagem SQL.
- 3) (Valor 2,00). Faça a consulta em SQL que mostra a tabela acima com seus respectivos dados.

Resolução – Padrão de Respostas Esperado

Projeto do BDBrasil
IFSP - Barretos - SP
Prof. Xyko
Data: Junho / 2017



```
/*
Projeto do Banco de Dados BDBrasil
Cruso SQL Server 2008 a 2016
IFSP-Barretos - Prof. Xyko
Data: junho/2017
*/
-- Parte 1) – Criação do Projto BDBrasil.
-----
-- criando o BAnco de Dados BDBrasil
Create Database BDBrasil;
-----
-- tornar o BDBrasil em uso
use BDBrasil;
-----
-- Criando a tabela de TBclima
create table BDBrasil.dbo.TBclima
(
    IDclima int not null primary key,
    NomeClima varchar(40) not null
);
-----
-- inserindo os climas na Tabela TBclima
insert into BDBrasil.dbo.TBclima
(IDclima, NomeClima)
Values (1, 'Semi-Árido');

insert into BDBrasil.dbo.TBclima
(IDclima, NomeClima)
Values (2, 'Tropical');

insert into BDBrasil.dbo.TBclima
(IDclima, NomeClima)
Values (3, 'Quente');

insert into BDBrasil.dbo.TBclima
(IDclima, NomeClima)
Values (4, 'Quente-Úmido');

insert into BDBrasil.dbo.TBclima
(IDclima, NomeClima)
Values (5, 'Quente');

insert into BDBrasil.dbo.TBclima
(IDclima, NomeClima)
Values (6, 'Equatorial');

insert into BDBrasil.dbo.TBclima
```

```
(IDclima, NomeClima)
Values (7, 'Semi-Árido Quente');
```

```
insert into BDBrasil.dbo.TBClima
(IDclima, NomeClima)
Values (8, 'Úmido');
```

```
-- mostra o cadastro dos Climas
select *
from BDBrasil.dbo.TBClima;
```

```
-----
-- Criando a tabela de TBRegiao
create table BDBrasil.dbo.TBRegiao
(
    IDRegiao int not null primary key,
    NomeRegiao varchar(40) not null
);
```

```
-----
-- inserindo os Economias na Tabela TBRegiao
insert into BDBrasil.dbo.TBRegiao
(IDRegiao, NomeRegiao)
Values (1, 'Nordeste');
```

```
insert into BDBrasil.dbo.TBRegiao
(IDRegiao, NomeRegiao)
Values (2, 'Sudeste');
```

```
insert into BDBrasil.dbo.TBRegiao
(IDRegiao, NomeRegiao)
Values (3, 'Norte');
insert into BDBrasil.dbo.TBRegiao
(IDRegiao, NomeRegiao)
Values (4, 'Nordeste');
```

```
insert into BDBrasil.dbo.TBRegiao
(IDRegiao, NomeRegiao)
Values (5, 'Sul');
```

```
insert into BDBrasil.dbo.TBRegiao
(IDRegiao, NomeRegiao)
Values (6, 'Centro-Oeste');
```

```
-----
-- mostrar os registros de TBRegiao
select *
from BDBrasil.dbo.TBRegiao;
```

-- Criando a tabela de TBEconomia
create table BDBrasil.dbo.TBEconomia

(
 IDEconomia int not null primary key,
 NomeEconomia varchar(40) not null
);

-- inserindo os Economias na Tabela TBEconomia

insert into BDBrasil.dbo.TBEconomia
(IDEconomia, NomeEconomia)
Values (1, 'Agricultura');

insert into BDBrasil.dbo.TBEconomia
(IDEconomia, NomeEconomia)
Values (2, 'Pecuária');

insert into BDBrasil.dbo.TBEconomia
(IDEconomia, NomeEconomia)
Values (3, 'Indústria');

insert into BDBrasil.dbo.TBEconomia
(IDEconomia, NomeEconomia)
Values (4, 'Agropecuária');

insert into BDBrasil.dbo.TBEconomia
(IDEconomia, NomeEconomia)
Values (5, 'Serviços');

-- mostrar os registros de Economias
select *
from BDBrasil.dbo.TBEconomia;

-- Criando a tabela de TBEstado
create table BDBrasil.dbo.TBEstado

(
 IDSigla char(2) not null,
 NomeEstado varchar(40) not null,
 Populacao bigint null,
 CodClima int not null,
 CodRegiao int not null,
 -- crando a chave primária
 Constraint pksigla primary key(IDSigla),
 -- criando as FK Regiao
 Constraint FKCodRegiao Foreign Key (CodRegiao)
 references BDBrasil.dbo.TBRegiao (IDRegiao)
 on delete no action
);

```
        on update cascade,  
    -- criando a fk Clima  
    Constraint FKCodClima Foreign Key (CodClima)  
        references BDBrasil.dbo.TBClima (IDclima)  
        on delete no action  
        on update cascade  
);
```

```
-----  
-- mostrar os Climas e as Regiões  
select * from BDBrasil.dbo.TBClima ;  
select * from BDBrasil.dbo.TBregiao ;
```

```
-----  
-- inserindo os Estados relacionados clima e a Região  
Insert Into BDBrasil.dbo.TBEstado  
(Idsigla, NomeEstado, Populacao, CodClima, CodRegiao)  
values ('SP', 'São Paulo', 9811776, 2, 2) ;
```

```
Insert Into BDBrasil.dbo.TBEstado  
(Idsigla, NomeEstado, Populacao, CodClima, CodRegiao)  
values ('BA', 'Bahia', 11855157, 7, 1) ;
```

```
Insert Into BDBrasil.dbo.TBEstado  
(Idsigla, NomeEstado, Populacao, CodClima, CodRegiao)  
values ('PI', 'Piauí', 2581054, 5 , 1 ) ;
```

```
Insert Into BDBrasil.dbo.TBEstado  
(Idsigla, NomeEstado, Populacao, CodClima, CodRegiao)  
values ('RJ', 'Rio Janeiro', 4870450, 2 , 2 ) ;
```

```
Insert Into BDBrasil.dbo.TBEstado  
(Idsigla, NomeEstado, Populacao, CodClima, CodRegiao)  
values ('RR', 'Roraima', 215790, 4 , 3 ) ;
```

```
Insert Into BDBrasil.dbo.TBEstado  
(Idsigla, NomeEstado, Populacao, CodClima, CodRegiao)  
values ('SE', 'Sergipe', 1492400, 7 , 1 ) ;
```

```
Insert Into BDBrasil.dbo.TBEstado  
(Idsigla, NomeEstado, Populacao, CodClima, CodRegiao)  
values ('MG', 'Minas Gerais', 3870457, 2 , 2 ) ;
```

```
Insert Into BDBrasil.dbo.TBEstado  
(Idsigla, NomeEstado, Populacao, CodClima, CodRegiao)  
values ('AL', 'Alagoas', 215790, 8 , 1 ) ;
```

```
Insert Into BDBrasil.dbo.TBEstado  
(Idsigla, NomeEstado, Populacao, CodClima, CodRegiao)
```

```
values ('PR', 'PARaná',8415654, 2 , 5 ) ;
```

```
Insert Into BDBrasil.dbo.TBEstado  
(Idsigla, NomeEstado, CodClima, CodRegiao)  
values ('GO', 'Goiás',3 , 6 ) ;
```

```
Insert Into BDBrasil.dbo.TBEstado  
(Idsigla, NomeEstado, Populacao, CodClima, CodRegiao)  
values ('AM', 'Amazonas',2102901, 6 , 3 ) ;
```

```
Insert Into BDBrasil.dbo.TBEstado  
(Idsigla, NomeEstado, Populacao, CodClima, CodRegiao)  
values ('TO', 'Tocantins', 1007410, 6 , 3 ) ;
```

```
-- mostrar os Estados Cadastrados  
select * from BDBrasil.dbo.TBEstado;  
select * from BDBrasil.dbo.TBClima ;  
select * from BDBrasil.dbo.TBregiao ;
```

```
-----  
-- Criando a Tabela de Cidade - TBCidade  
create table BDBrasil.dbo.TBCidade  
(  
    IDcidade int not null,  
        CodSigla char(2) not null,  
    NomeCidade varchar(40) not null,  
        -- crando a chave primária  
    Constraint pkidcidade primary key(IDcidade),  
    -- criando as FK Estado  
    Constraint FKCodEstado Foreign Key (Codsigla)  
        references BDBrasil.dbo.TBEstado (Idsigla)  
        on delete no action  
        on update cascade  
);
```

```
-----  
-- fazendo o Cadastro de Cidades  
Insert into BDBrasil.dbo.TBCidade  
(IDCidade, NomeCidade, CodSigla)  
Values (1, 'Salvador', 'BA');
```

```
Insert into BDBrasil.dbo.TBCidade  
(IDCidade, NomeCidade, CodSigla)  
Values (2, 'Bom Jardim', 'PI');
```

```
Insert into BDBrasil.dbo.TBCidade  
(IDCidade, NomeCidade, CodSigla)  
Values (3, 'Bom Sucesso', 'RJ');
```

**Insert into BDBrasil.dbo.TBCidade
(IDCidade, NomeCidade, CodSigla)
Values (4, 'Rio de Janeiro', 'RJ');**

**Insert into BDBrasil.dbo.TBCidade
(IDCidade, NomeCidade, CodSigla)
Values (5, 'Belo Horizonte', 'MG');
Insert into BDBrasil.dbo.TBCidade
(IDCidade, NomeCidade, CodSigla)
Values (6, 'Uberaba', 'MG');**

**Insert into BDBrasil.dbo.TBCidade
(IDCidade, NomeCidade, CodSigla)
Values (7, 'Boa Vista', 'RR');**

**Insert into BDBrasil.dbo.TBCidade
(IDCidade, NomeCidade, CodSigla)
Values (8, 'Aracajú', 'SE');**

**Insert into BDBrasil.dbo.TBCidade
(IDCidade, NomeCidade, CodSigla)
Values (9, 'Maceió', 'AL');**

**Insert into BDBrasil.dbo.TBCidade
(IDCidade, NomeCidade, CodSigla)
Values (10, 'Curitiba', 'PR');**

**Insert into BDBrasil.dbo.TBCidade
(IDCidade, NomeCidade, CodSigla)
Values (11, 'Goiatuba', 'GO');**

**Insert into BDBrasil.dbo.TBCidade
(IDCidade, NomeCidade, CodSigla)
Values (12, 'Manaus', 'AM');**

**Insert into BDBrasil.dbo.TBCidade
(IDCidade, NomeCidade, CodSigla)
Values (13, 'Palmas', 'TO');**

**Insert into BDBrasil.dbo.TBCidade
(IDCidade, NomeCidade, CodSigla)
Values (14, 'Olímpia', 'SP');**

**Insert into BDBrasil.dbo.TBCidade
(IDCidade, NomeCidade, CodSigla)
Values (15, 'Guapiaçu', 'SP');**

-- Mostrar todas as cidades cadastradas

```
select *  
from BDBrasil.dbo.TBCidade;
```

-- Criando a Tabela TBEstadoEconomia

```
Create table BDBrasil.dbo.TBEstadoEconomia  
(
```

```
    CodSigla char (2) not null,
```

```
    CodEconomia int not null,
```

```
    -- definindo a PK-chave Primária
```

```
    constraint pkTbEstadoEconomia
```

```
        primary key (CodSigla, CodEconomia),
```

```
    -- fk Estado
```

```
        Constraint FKCodEstado1 Foreign Key (Codsigla)
```

```
            references BDBrasil.dbo.TBEstado (IDsigla)
```

```
            on delete no action
```

```
            on update cascade,
```

```
    -- fk Economia
```

```
        Constraint FKCodEconomia Foreign Key (CodEconomia)
```

```
            references BDBrasil.dbo.TBEconomia (IDeconomia)
```

```
            on delete no action
```

```
            on update cascade
```

```
);
```

-- Cadastro de Estado com suas Economias

```
Insert into BDBrasil.DBO.TBEstadoEconomia  
(CodSigla, CodEconomia)  
values ('BA', 1);
```

```
Insert into BDBrasil.DBO.TBEstadoEconomia  
(CodSigla, CodEconomia)  
values ('PI',2);
```

```
Insert into BDBrasil.DBO.TBEstadoEconomia  
(CodSigla, CodEconomia)  
values ('RJ',3);
```

```
Insert into BDBrasil.DBO.TBEstadoEconomia  
(CodSigla, CodEconomia)  
values ('RJ',5);
```

```
Insert into BDBrasil.DBO.TBEstadoEconomia  
(CodSigla, CodEconomia)  
values ('MG', 3);
```

```
Insert into BDBrasil.DBO.TBEstadoEconomia  
(CodSigla, CodEconomia)
```

values ('MG', 4);

**Insert into BDBrasil.DBO.TBEstadoEconomia
(CodSigla, CodEconomia)
values ('RR',2);**

**Insert into BDBrasil.DBO.TBEstadoEconomia
(CodSigla, CodEconomia)
values ('SE',1);**

**Insert into BDBrasil.DBO.TBEstadoEconomia
(CodSigla, CodEconomia)
values ('AL', 1);**

**Insert into BDBrasil.DBO.TBEstadoEconomia
(CodSigla, CodEconomia)
values ('PR', 1);**

**Insert into BDBrasil.DBO.TBEstadoEconomia
(CodSigla, CodEconomia)
values ('GO', 2);**

**Insert into BDBrasil.DBO.TBEstadoEconomia
(CodSigla, CodEconomia)
values ('AM', 3);**

**Insert into BDBrasil.DBO.TBEstadoEconomia
(CodSigla, CodEconomia)
values ('TO', 2);**

**Insert into BDBrasil.DBO.TBEstadoEconomia
(CodSigla, CodEconomia)
values ('SP', 1);**

**Insert into BDBrasil.DBO.TBEstadoEconomia
(CodSigla, CodEconomia)
values ('SP', 3);**

**-- MOSTRAR AS ECONOMIAS
SELECT *
FROM BDBrasil.DBO.TBEconomia;**

**SELECT *
FROM BDBrasil.DBO.TBEstadoEconomia;**

```
/*
Parte 2 - Consultas ao BDBrasil
Mostrar os Dados Cadastrados nas tabelas sob alguma(s) Condição(es)
*/
-- 1) Mostrar todos os Estados com o Clima Tropical
Select          IDSigla as [Unidade da Federação],
                NomeEstado as [Estado],
                NomeClima as Clima
from            BDBrasil.dbo.TBclima as c,
                BDBrasil.dbo.TBEstado as e
Where           NomeClima = 'Tropical'
and            (c.idclima = e.codclima)
order by 2;

-- mostrar os registros da Tabela TBEstado
Select *
    from TBEstado;

-- mostrar os registros da Tabela TBclima
Select *
    from TBclima;

-- atualizando a tabela de Estado - conteúdo da tabela
-- Mudar o Clima do Tocantins
Update BDBrasil.dbo.TBEstado
    set CodClima = 2
    where Idsigla = 'TO';
-----
-- 2) Mostrar todos os Estados com o Economia Pecuária e Agricultura

Select          e.IDSigla as [Unidade da Federação],
                NomeEstado as [Estado],
                NomeEconomia as Economia
from            BDBrasil.dbo.TBEstado as e,
                BDBrasil.dbo.TBEconomia as ec,
                BDBrasil.dbo.TBEstadoEconomia as ee
Where           (NomeEconomia = 'Pecuária'
                or NomeEconomia = 'Agricultura')
and            (ec.ideconomia = ee.codeconomia)
and            (ee.codsigla = e.idsigla)
order by 3;
-----

/* Consulta/Pesquisa 3.
Mostrar: Código, Nome da Região e a
Quantidade de Estados Cadastrados.
*/
```

Projeto, Modelagem e Gerenciamento de Banco de Dados.
Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

```
Select      IdRegiao as [Código Região],
            NomeRegiao as Região,
            count (E.CodRegiao) as [Total Estados na Região]
from        BDBrasil.dbo.TBRegiao as R,
            BDBrasil.dbo.TBEstado as E
Where       (R.IDregiao = E.CodRegiao)
group by    IDRegiao, NomeRegiao
```

-- checando os Resultados

-- confrontando os Resultados

```
select      NomeEstado, NomeRegiao
from        TBEstado as e,
            TBRegiao as r
Where       (r.idregiao = e.codregiao)
order by    NomeRegiao;
```

/* Consulta/Pesquisa 4.

Mostrar: Código, Nome do Clima,
Nome da Região e Total de Climas por Região

*/

```
Select      IdClima as [Código Clima],
            NomeClima as Clima,
            count (E.Codclima) as [Total Climas no Estado]
from        BDBrasil.dbo.TBClima as C,
            BDBrasil.dbo.TBEstado as E
Where       (C.IDClima = E.CodClima)
group by    IDClima, NomeClima
order by    3
```

/* Consulta - Pesquisa 5.

Mostrar: Nome Cidade,
Estado,
Nome Clima

Obs1: É fundamental perceber via Projeto que
temos 3 tabelas reacionadas (envolvidadas)

Obs2: Isto mostra a importância de Separar as Tabalas
e fazer o Relacionamento - chamado Normalização de Dados

Obs3: Perceber que o Resultado tem lógica

*/

```
Select      NomeCidade as [Cidade],
            IDSigla as Estado,
            NomeClima as Clima
```

```
from      BDBrasil.dbo.TBClima as C,  
          BDBrasil.dbo.TBEstado as E,  
          BDBrasil.dbo.TBCidade as Ci
```

```
Where      ( Ci.CodSigla = E.IDSigla )  
and        ( E.CodClima = C.IDClima )  
order by 2
```

```
-----  
/*  Consulta/Pesquisa 6.  
    Quais as Cidades tem o clima Tropical  
*/
```

```
Select      IdCidade as [Código Cidade],  
            NomeCidade as Cidade,  
            NomeClima as Clima  
from        BDBrasil.dbo.TBClima as Ci,  
            BDBrasil.dbo.TBEstado as E,  
            BDBrasil.dbo.TBCidade as C  
where       CI.NomeClima = 'Tropical'  
and         ( E.IdSigla = C.CodSigla )  
and         ( E.CodClima = CI.IDClima )  
order by 2
```

```
-----  
-- Notar que tem 08 registros (cidades com clima Tropical)  
-- checando.
```

```
-- consulta 7) Contando a quantidade de Cidades  
-- com clima Tropical  
-- Prova que a consulta anterior está correta.
```

```
Select      c.NomeClima as Clima,  
            count (CI.IDcidade) as  
            [Total de Cidades com Clima Tropical]
```

```
from        BDBrasil.dbo.TBClima as C,  
            BDBrasil.dbo.TBEstado as E,  
            BDBrasil.dbo.TBCidade as Ci
```

```
Where      nomeclima = 'Tropical'  
and        ( Ci.CodSigla = E.IDSigla )  
and        ( E.CodClima = C.IDClima )  
group by c.NomeClima
```

/* Consulta 7

Quais os Estados tem duas ou mais Economias Cadastradas?

***/**

```
select          IdSigla as [Código Estado],
                NomeEstado as Estado,
                count (distinct EE.codEconomia) as
                    [Total de Economias no Estado]
from            BDBrasil.dbo.TBEstadoEconomia as EE,
                BDBrasil.dbo.TBEstado as E,
                BDBrasil.dbo.TBEconomia as EC
where           (E.IDSigla = EE.CodSigla)
and             (EE.CodEconomia = EC.IDeconomia)
group by IdSigla, NomeEstado
-- Aqui filtra o agrupamento
having          count (distinct EE.codEconomia) >= 2
order by 2;
```

/* Consulta 8

Mostrar as Regioes e Quantos Estados tem por Regiao Cadastrados

***/**

```
Select          NomeRegiao as Regiao,
                count ( E.codRegiao) as
                    [Total Estados Cadastrados por Regiao]
from            BDBrasil.dbo.TBEstado as E,
                BDBrasil.dbo.TBRegiao as R
where           (R.IDRegiao = E.codRegiao)
group by        NomeRegiao
order by        2
```

/*

Parte 3

Programando o BDBrasil

3.1) Visoes

3.2) Funções

***/**

-- 1) Mostrar todos os Estados com o Clima Tropical
-- criando uma visão para mostrar os Estados com Clima Tropical

Create view vEstadoTropical

as

```
Select          IDSigla as [Unidade da Federação],
                NomeEstado as [Estado],
```

```
NomeClima as Clima
from      BDBrasil.dbo.TBclima as c,
          BDBrasil.dbo.TBEstado as e
Where     NomeClima = 'Tropical'
and       (c.idclima = e.codclima);
```

```
-- executar a visao vEstadoTropical
select *
  from dbo.vEstadoTropical;
```

```
-----
-- mostrar os registros da Tabela TBEstado
Select *
  from TBEstado;
```

```
-----
-- 2) Mostrar todos os Estados com o Economia Pecuária e Agricultura
-- Criar uma Visao VEconomiaPecuarariaAgricultura
```

```
create view VEconomiaPecuarariaAgricultura
as
Select      e.IDSigla as [Unidade da Federação],
            NomeEstado as [Estado],
            NomeEconomia as Economia
from        BDBrasil.dbo.TBEstado as e,
            BDBrasil.dbo.TBEconomia as ec,
            BDBrasil.dbo.TBEstadoEconomia as ee
Where       (NomeEconomia = 'Pecuária'
or          NomeEconomia = 'Agricultura')
and         (ec.ideconomia = ee.codeconomia)
and         (ee.codsigla = e.idsigla);
```

```
-- executar
select *
  from VEconomiaPecuarariaAgricultura;
```

```
-----
/* Consulta/Pesquisa 4.
   Mostrar: Código, Nome do Clima,
            Nome da Região e Total de Climas por Região
   Criando uma visão - VMostraEstadoporRegiao
*/
```

```
create view VMostraEstadoporRegiao
as
Select      IdClima as [Código Clima],
            NomeClima as Clima,
            count (E.Codclima) as [Total Climas no Estado]
from        BDBrasil.dbo.TBclima as C,
            BDBrasil.dbo.TBEstado as E
```


Where **(C.IDClima = E.CodClima)**
group by **IDClima, NomeClima;**

Select *
from VMostraEstadoporRegiao;

/* Trabalhando função
serve para retornar dados especificos de uma caonsulta
-- função retorna valor

1) Criando a função que Retorna a Quantidade de Estados,
informando o parâmetro CodRegiao
***/**

CREATE FUNCTION DBO.FUNCAO_QTDE_Estados_por_Regiao
(@CodRegiao INT)
RETURNS TABLE
AS
RETURN
(
Select **IdRegiao as [Código Região],**
NomeRegiao as Região,
count (E.CodRegiao) as [Total Estados na Região]
from **BDBrasil.dbo.TBRegiao as R,**
BDBrasil.dbo.TBEstado as E
Where **(R.IDregiao = E.CodRegiao)**
and **(idregiao = @CodRegiao)**
group by **IDRegiao, NomeRegiao**
)

-- EXECUTANDO A FUNCTION DBO.FUNCAO_QTDE_Estados_por_Regiao
SELECT *
FROM DBO.FUNCAO_QTDE_Estados_por_Regiao(3);

SELECT *
FROM DBO.FUNCAO_QTDE_Estados_por_Regiao(5);

-- 2) Mostrar todos os Estados com o Clima Tropical
-- Criando uma Visão VMostraClimaTropical

CREATE FUNCTION VMostraClimaTropical (@CodClima INT)
RETURNS TABLE
AS
RETURN
(

```
Select          IDSigla as [Unidade da Federação],
                NomeEstado as [Estado],
                NomeClima as Clima
from            BDBrasil.dbo.TBclima as c,
                BDBrasil.dbo.TBEstado as e
Where          NomeClima = 'Tropical'
and            (c.idclima = e.codclima)
and            (@CodClima = idClima)
)
-- checando TBclima
select * from TBclima;
-- executando a função
SELECT *
FROM DBO.VMostraClimaTropical(2);
```

```
-- 2) Mostrar todos os Estados com os seus Climas
-- Criando uma Visão VMostraClimaQualquer
```

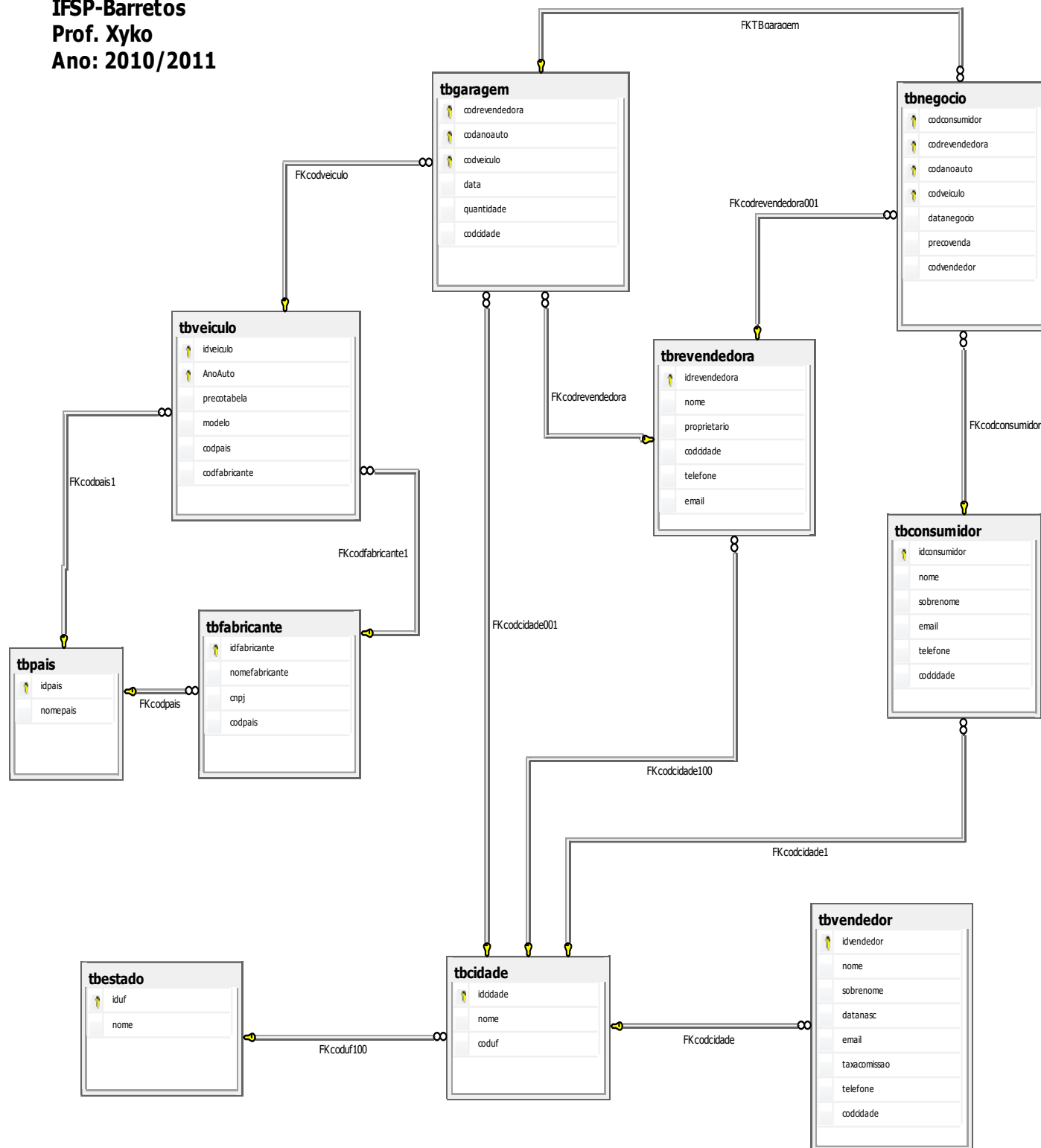
```
CREATE FUNCTION VMostraClimaQualquer
                (@CodClima INT)
    RETURNS TABLE
    AS
    RETURN
(
Select          IDSigla as [Unidade da Federação],
                NomeEstado as [Estado],
                NomeClima as Clima
from            BDBrasil.dbo.TBclima as c,
                BDBrasil.dbo.TBEstado as e
Where          (c.idclima = e.codclima)
and            (@CodClima = idClima)
)

-- executando a função
SELECT *
FROM DBO.VMostraClimaQualquer(4);
```

3 - Trabalhos Propostos – Projeto de Banco e Uso de SQL.

3.1 - Trabalho: 1) Dado o Diagrama de Tabelas (Projeto BDVeiculo) – Usando SQL Server 2008 a 2017 - Desenvolver em SQL o BDVeiculo.

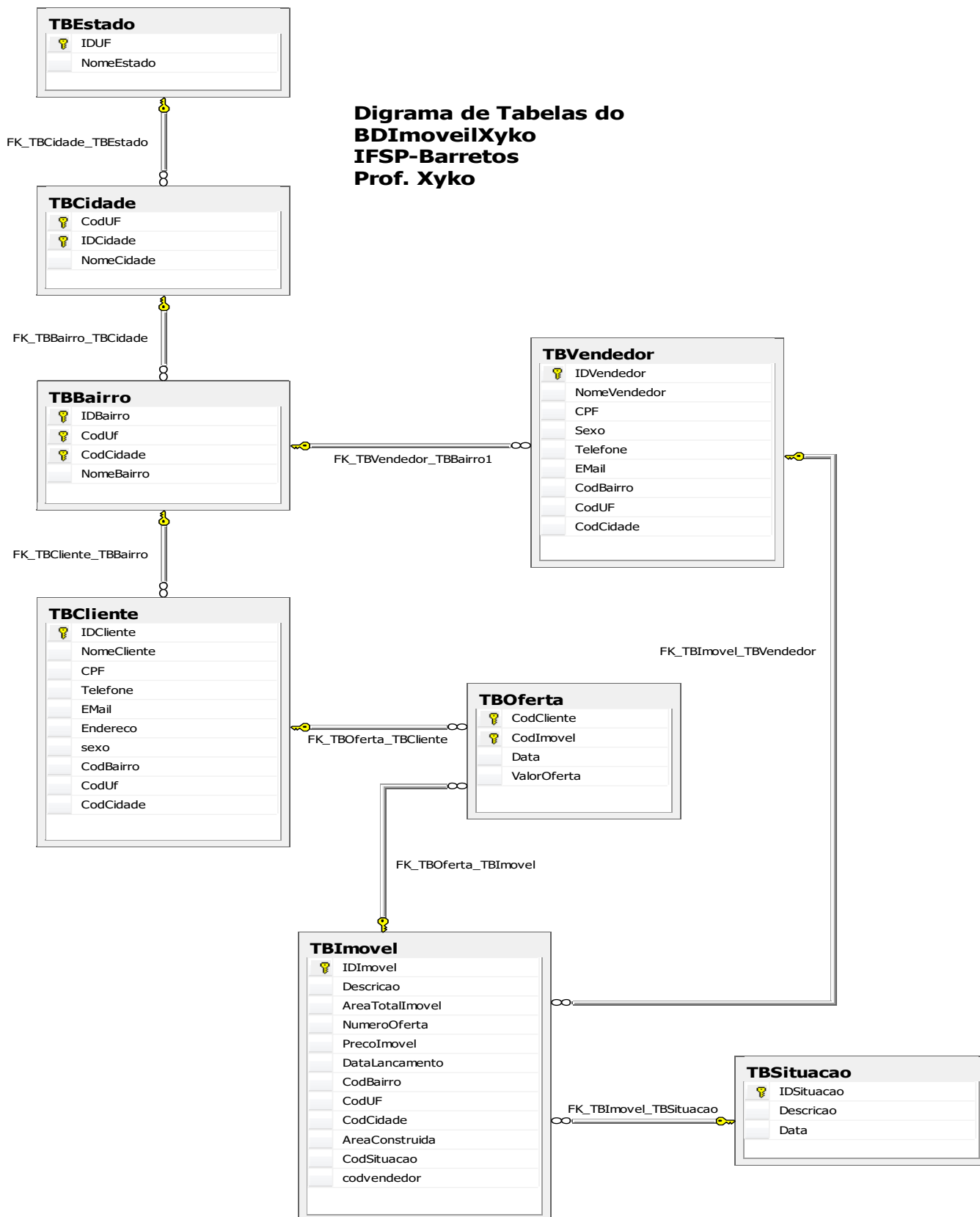
Projeto BDVeiculo
IFSP-Barretos
Prof. Xyko
Ano: 2010/2011



Projeto, Modelagem e Gerenciamento de Banco de Dados.

Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

3.2- Trabalho: 2) Dado o Diagrama de Tabelas (Projeto BDImovelXyko) – Usando SQL Server 2008 a 2017 - Desenvolver em SQL o BDImovelXyko.

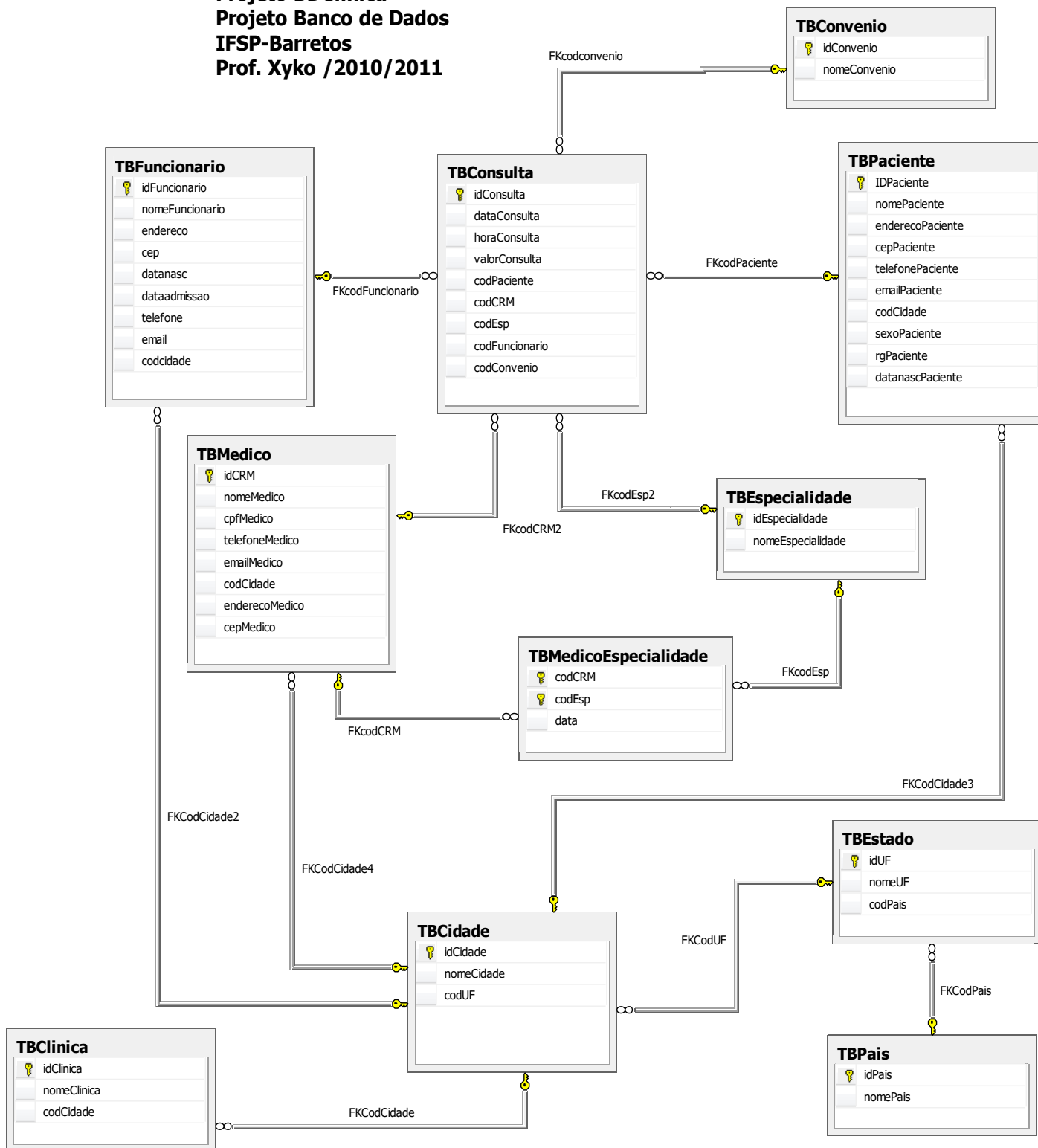


Projeto, Modelagem e Gerenciamento de Banco de Dados.

Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016


3.3- Trabalho: 3) Dado o Diagrama de Tabelas (Projeto BDClínica) – Usando SQL Server 2008 a 2017 - Desenvolver em SQL o BDClínica. Criar uma Pasta com Nome, código SQL e BKBD

Projeto BDClínica
Projeto Banco de Dados
IFSP-Barretos
Prof. Xyko /2010/2011



4 – Modelos de Provas/Exercícios de Gerenciamento de Banco de Dados

4.1- Prova 1- Ministrada em Novembro de 2011 a 2015

 INSTITUTO FEDERAL SÃO PAULO Campus Barretos	INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO- IFSP VERIFICAÇÃO DE CONHECIMENTO DE NOVEMBRO DE 2011 CURSO: MANUTENÇÃO E SUPORTE EM INFORMÁTICA - MSI	
DISCIPLINA: GERENCIAMENTO DE BANCO DE DADOS - GBD	MSI SEGUNDO MÓDULO - NOITE	
PROFESSORE : Francisco Antonio de Almeida – Xyko	DATA: 08/11/2012	
ALUNO:	NOTA:	

Justifique todas as suas respostas com argumentos Claros, Objetivos e Conclusivos.

1) (Valor: 3,00). Um Diagrama Entidade Relacionamento (DER/MER) Representa o Relacionamento entre as Entidades Empregado e Departamento, com seus respectivos atributos, e tem o relacionamento com Cardinalidade N:N, ou seja: Um Empregado pode ser alocado em 0, 1, ou vários Departamentos. Um Departamento pode ter: 0, 1 ou muitos Empregados.

- Criar o DER (lembrar: o SGBD não implementa relacionamento N:N);
- Mostrar os Comandos em SQL que Cria as Tabelas com os atributos, os relacionamentos e as respectivas Chaves Primárias e Estrangeiras.

2) (Valor: 3,00). Estude a Tabela abaixo que foi produzida por uma consulta usando SQL e **Responda:**

- Identifique: a Tabela, os Registros, Chave Primária (PK), Chave Estrangeira (FK) e os Atributos.
- Explique porque os valores 10 e 11 podem se repetir na coluna CodInstituicao?
- Qual o Relacionamento é possível identificar? Entre quais Tabelas?

Curso

CodCurso	NomeCurso	Duracao	AnoInicio	CodInstituicao
100	Sistemas de Informação	4	1999	11
120	Enfermagem	4	2000	10
015	Medicina Veterinaria	5	1998	10
210	Tecnologia em Informática	3	2004	11

Projeto, Modelagem e Gerenciamento de Banco de Dados.
Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

3) (Valor: 4,00). Na Universidade **UniXyko S/A**, a **regra de negócio** é a seguinte: um Aluno pode cursar zero, uma ou várias Disciplinas e uma Disciplina pode ter: zero, um ou vários alunos.

Entidades	Atributos	Restrições
Aluno	IDAluno, Nome, DataNascimento, Sexo, CodUF	IDaluno é chave primária, CodUF chave Estrangeira.
Estado	IDUF, NomeEstado	IDUF chave primária.
Disciplina	IDDisciplina, NomeDisciplina, CargaHoraria, CodTurno, CodDepto	IDDisciplina chave Primária, CodTurno Chave Estrangeira, CodDepto chave estrangeira
Turno	IDTurno, NomeTurno	IDTurno Chave Primária
Departamento	IDDepto, Nome	IDDepto Chave Primária

- a) Escreva o Código SQL que Cria as Tabelas Acima com os Atributos, relacionamentos e Respectivas Chaves Primárias, Estrangeiras e Restrições.

4.2 - Prova 2) – Ministrada outubro/2011 a outubro/2015

Prova 2: Gerenciamento de BANCO DE DADOS

Professor: Prof. Xyko – 2011 a 2015

Disciplina: Gerenciamento de Banco de Dados

Aluno (a): Nota:



1) – Crie as seguintes tabelas Usando SGBD SQL Server 2008 – Usando código SQL.

- TBEmpregado**

Cod_emp	Nome_emp	Cargo	Salario	Cod_Depto
PK	Texto	Texto	Decimal(10,2)	FK

- TBDepartamento**

Cod_depto	Nome	Local
PK	Texto	Texto

2 – Acrescente o atributo Endereco a tabela TBEmpregado.

3 – Insira os seguintes registros na tabela TBEmpregado:

- 456 – João das Couves, Av. Brasil S/N, Vendedor, 500, 10.
- 678 – Pedro Paulo, Av. Curitiba 2567, Digitador, 800, 20.
- 785 – Carla Maria, Rua das Flores 67, Suporte, 750, 10.
- 908 – Jose da Silva, Rua Maringá 32, Vendedor, 700, 10.

4 - Insira os seguintes registros na tabela TBDepartamento:

- 10 – Atendimento ao Cliente, Umuarama.
- 20 – Centro de Processamento de Dados, Umuarama.
- 30 – Contabilidade, Maringá.
- 40 – Vendas, Umuarama.

Projeto, Modelagem e Gerenciamento de Banco de Dados.

Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

5 – Alterar a profissão do Empregado João das Couves para a profissão de Digitador.

6 – Dar um aumento de 10% no salário para os empregados do Departamento 10.

7 – Transferir os empregados do Departamento 10 para o 40.

8 – Apagar o Empregado de Código 785 da tabela TBEmpregado.

9 – Remover um empregado do departamento 20.

10 – **Faça os seguintes Consultas (instruções) usando SQL:**

- a) Selecione todos os funcionários da tabela TBEmpregado.
- b) Selecione todos os funcionários da tabela TBEmpregado ordenados por nome.
- c) Selecione todos os funcionários que pertençam ao departamento 40.
- d) Selecione todos os funcionários agrupados por Departamento.
- e) Apresente a media dos salários dos Funcionários.
- f) Liste todos os funcionários que tenham salário no intervalo de 500 a 2500.
- g) Retorne o número de linhas da tabela Departamento.
- h) Apresente a soma de todos os salários.
- i) Apresente o maior e o menor salário de todos os Funcionários.
- j) Apresente o nome dos Funcionários, o salário e o nome do Departamento.

REVISÃO Projeto de Banco de Dados BDIFSP
Data: 20/06/2018

Com Base na Modelagem de Dados da Figura 1 abaixo, e utilizando o Software BrModelo: Projetar o Modelo Conceitual, Lógico e Projeto Físico (código SQL).

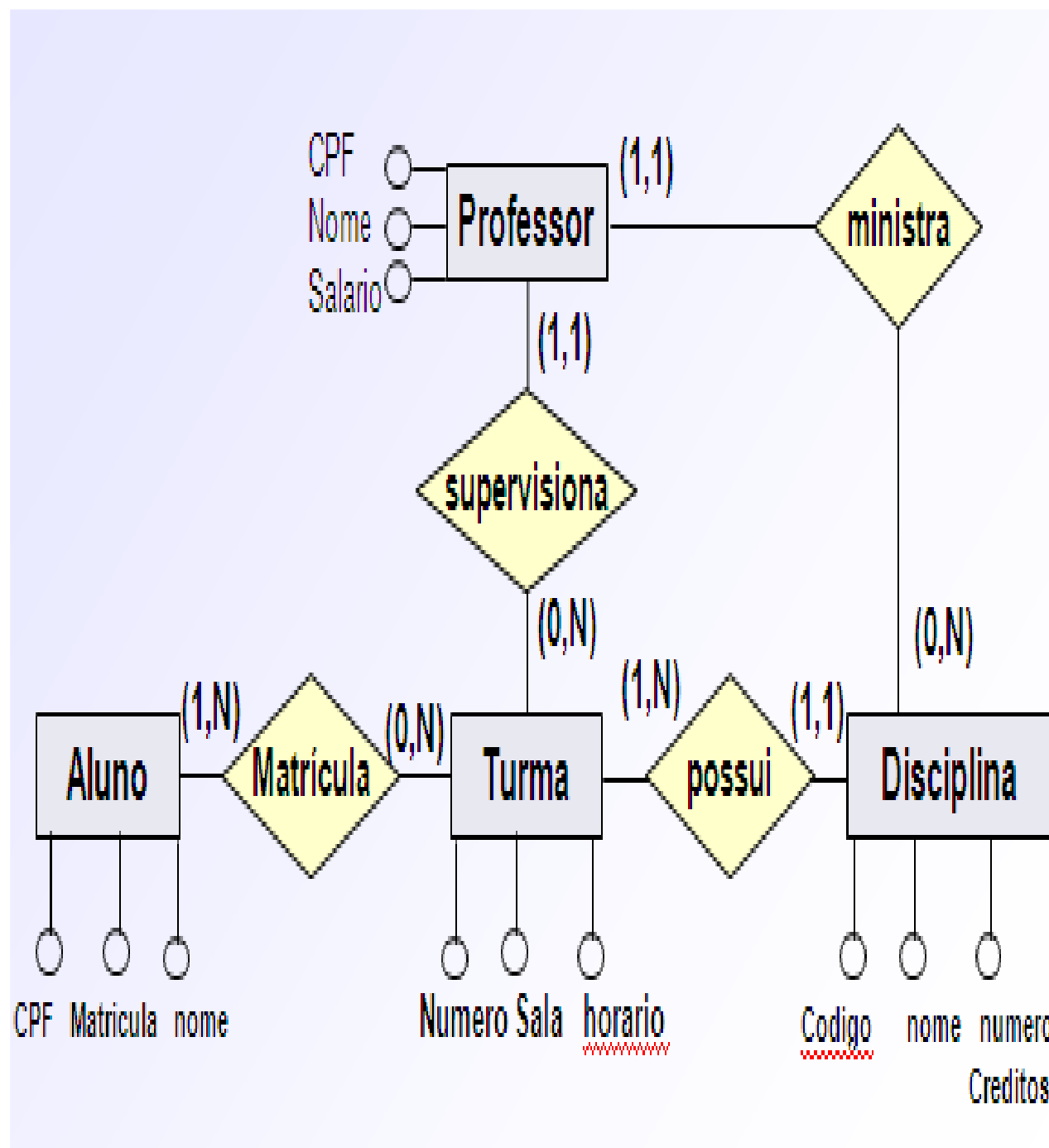


Figura 1: Visão geral de um DER com Entidades, Atributos, Relacionamentos e Cardinalidades.

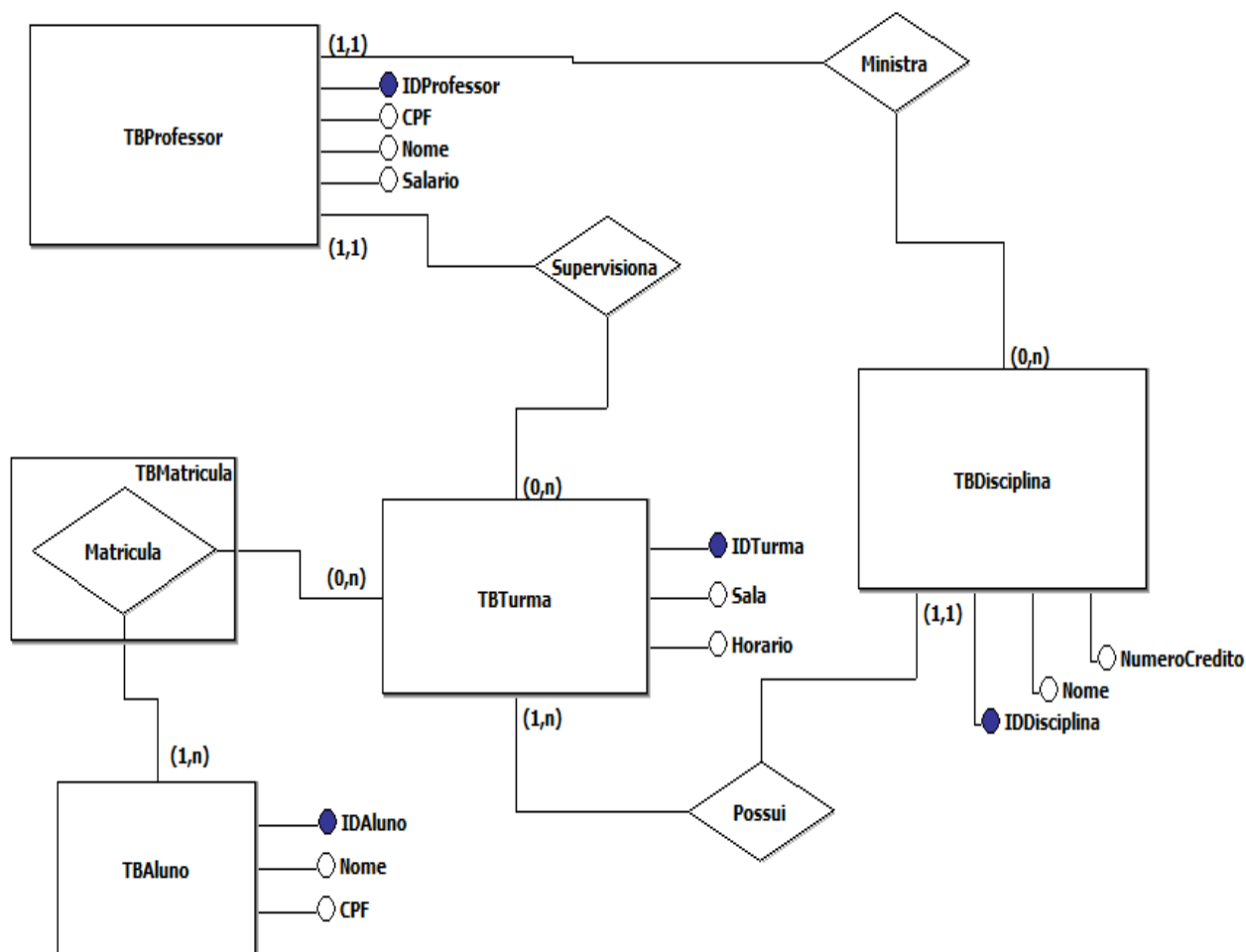
Gabarito – Padrão de Resposta Esperado

Parte a) Projeto Conceitual gerado pelo BRModelo

- Trata-se de uma visão mais de alto nível do Projeto de Banco de Dados;
- Cria-se as Entidades, define os atributos, faz os relacionamentos e define as cardinalidades dos relacionamentos;
- Nota que em toda Entidade foi definido uma chave primária para evitar repetição de dados;
- Após esta etapa será gerado pelo brModelo o Projeto Lógico (parte b).

Projeto Conceitual do BDIFSP.

Trabalhando com Entidades, atributos, relacionamentos e Cardinalidades.



Projeto, Modelagem e Gerenciamento de Banco de Dados.

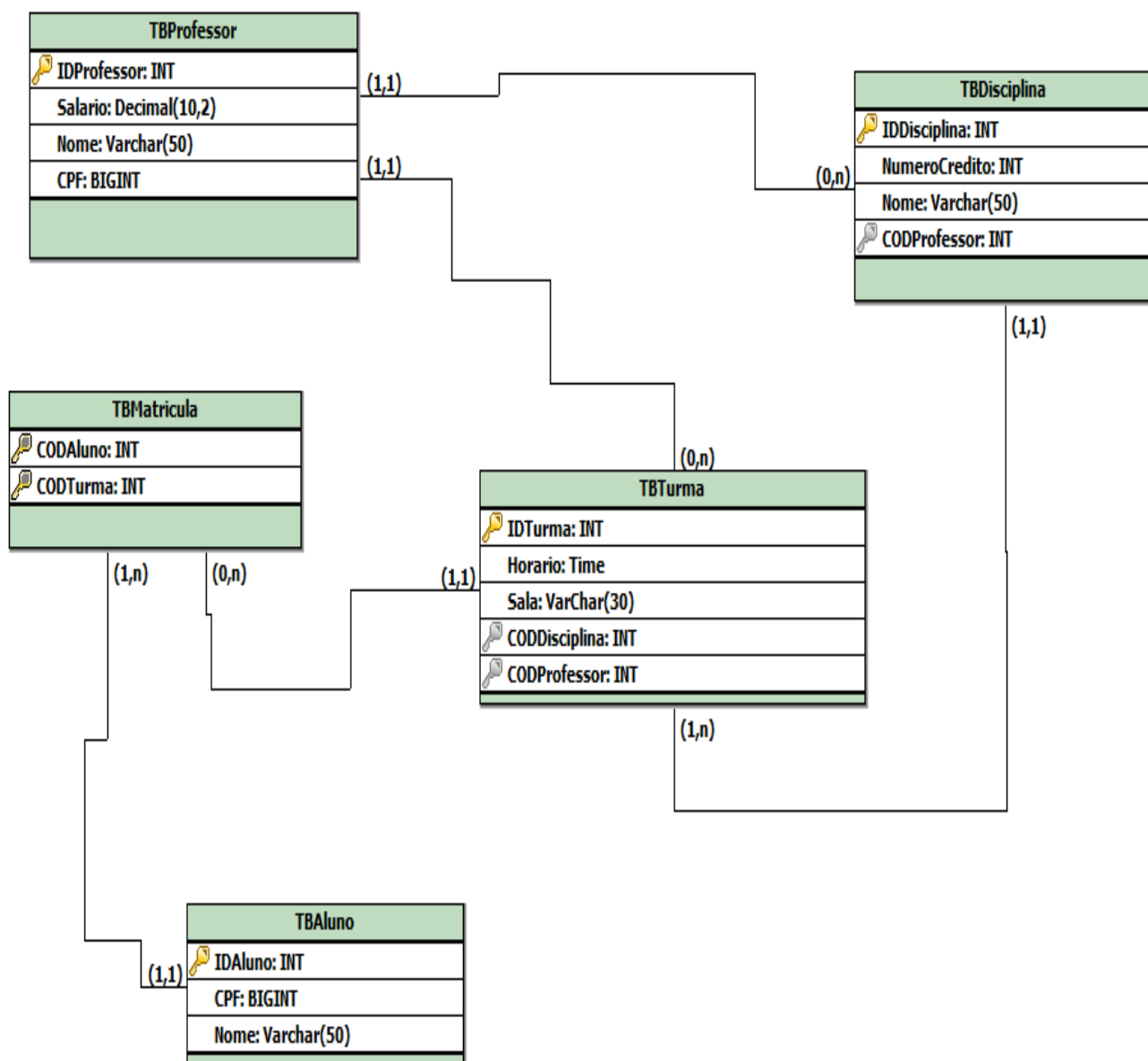
Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

Parte b) Projeto Lógico do BDIFSP criado pelo BrModelo.

- Observe que o Projeto Lógico está sendo, aos poucos, direcionado para o *software*;
- Já identificamos as tabelas, atributos e seus tipos de dados, os relacionamentos entre as tabelas, as Chaves primárias (cor dourada) e as chaves estrangeiras (cor cinza);
- Após esta etapa será gerado o código SQL pelo brModelo, que é a parte do projeto Físico a ser interpretado pelo SGBDR SQL Server 2014

Projeto LÓGICO do BDIFSP.

Trabalhando com Entidades, atributos, relacionamentos e Cardinalidades.



Projeto, Modelagem e Gerenciamento de Banco de Dados.

Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

Parte c) Projeto físico (código SQL) gerado pelo brModelo.

- Códigos SQL gerado pelo brModelo;
- Este código SQL Será interpretado pelo SGBDR SQL Server 2014 e será criado o Banco de Dados.

```
-- Geração de Modelo físico
-- Sql ANSI 2003 - brModelo.
-----
-- CRIAR O BANCO DE DADOS

CREATE DATABASE BDIFSP_XYKO
-----

-- ABRIR O bd
USE BDIFSP_XYKO
-----

CREATE TABLE TBProfessor (
IDProfessor INT NOT NULL PRIMARY KEY,
Salario Decimal(10,2),
Nome Varchar(50),
CPF BIGINT
)

CREATE TABLE TBDisciplina (
IDDisciplina INT NOT NULL PRIMARY KEY,
NumeroCredito INT,
Nome Varchar(50),
CODProfessor INT NOT NULL,
FOREIGN KEY(CODProfessor) REFERENCES TBProfessor (IDProfessor)
)

CREATE TABLE TBTurma (
IDTurma INT NOT NULL PRIMARY KEY,
Horario Time,
Sala VarChar(30),
CODDisciplina INT NOT NULL,
CODProfessor INT NOT NULL,
FOREIGN KEY(CODDisciplina) REFERENCES TBDisciplina (IDDisciplina),
FOREIGN KEY(CODProfessor) REFERENCES TBProfessor (IDProfessor)
)

CREATE TABLE TBAluno (
IDAluno INT NOT NULL PRIMARY KEY,
CPF BIGINT,
Nome Varchar(50)
)

CREATE TABLE TBMatricula (
CODAluno INT NOT NULL,
CODTurma INT NOT NULL,
PRIMARY KEY(CODAluno,CODTurma),
FOREIGN KEY(CODAluno) REFERENCES TBAluno (IDAluno),
FOREIGN KEY(CODTurma) REFERENCES TBTurma (IDTurma)
)
-----
-- Cadastrar (inserir) Registros
Insert into TBProfessor
```

Projeto, Modelagem e Gerenciamento de Banco de Dados.

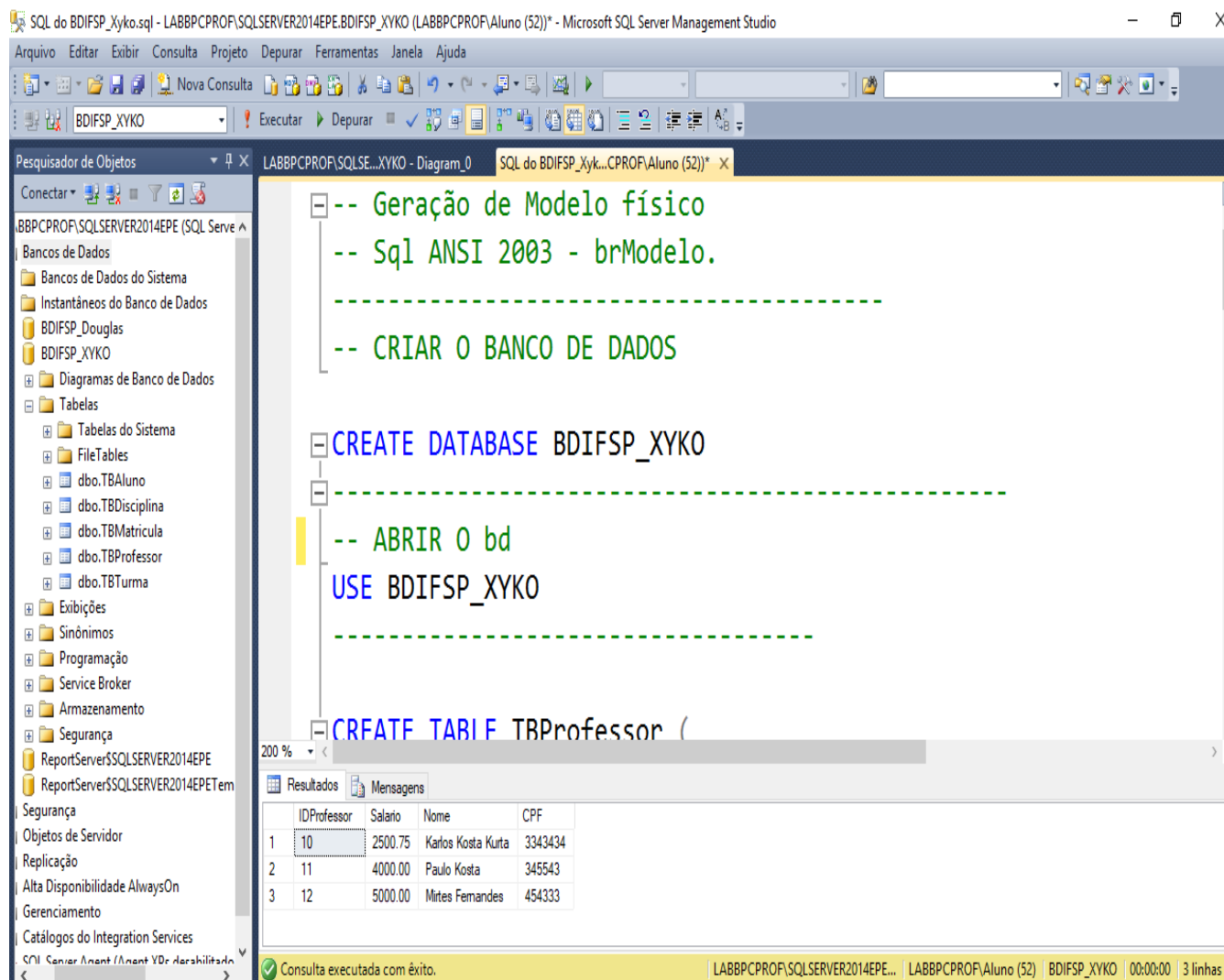
Exemplos práticos desenvolvidos para SQL Server 2008/2012/2014 ou 2016

```
Values (10, 2500.75, 'Karlos Kosta Kurta', 3343434),  
        (11, 4000, 'Paulo Kosta', 345543),  
        (12, 5000, 'Mirtes Fernandes', 454333)
```

```
-- Mostrar os Registros  
Select * from TBProfessor
```

Parte d) Aqui será aberto (conectado) o SQL Server 2014 e interpretado o código SQL para a criação do BD.

- Interface do SQL Server 2014 onde o código SQL da **parte c)** foi importado e executado;
- **Importante entender:** que após ser criado no brModelo os Projetos Conceitual e Lógico, foi criado todo código SQL pelo brModelo.



The screenshot displays the Microsoft SQL Server Management Studio (SSMS) interface. The main window shows a SQL script with the following content:

```
-- Geração de Modelo físico  
-- Sql ANSI 2003 - brModelo.  
-----  
-- CRIAR O BANCO DE DADOS  
  
CREATE DATABASE BDIFSP_XYKO  
-----  
-- ABRIR O bd  
USE BDIFSP_XYKO  
-----  
  
CREATE TABLE TBProfessor (
```

The left pane shows the 'Pesquisador de Objetos' (Object Explorer) with the database 'BDIFSP_XYKO' selected. The right pane shows the 'Resultados' (Results) window with the following data:

IDProfessor	Salario	Nome	CPF
10	2500.75	Karlos Kosta Kurta	3343434
11	4000.00	Paulo Kosta	345543
12	5000.00	Mirtes Fernandes	454333

The status bar at the bottom indicates 'Consulta executada com êxito.' (Query executed successfully.) and shows the execution time as 00:00:00 and 3 lines.

Referências Bibliográficas

ALMEIDA, Francisco Antonio de. **Aprendizado Prático da Modelagem e Projeto de Banco de Dados Relacional usando a ferramenta CASE brModelo**. Virtual Books Editora e Livraria Ltda: Patos de Minas – MG, 2016;

ALMEIDA, Francisco Antonio de. **Administração de Banco de Dados com SQL e T-SQL no SQL Server 2014 com aplicações práticas**. Virtual Books Editora e Livraria Ltda: Patos de Minas – MG, 2015, 230p;

ALMEIDA, Francisco Antonio de; BORGES JUNIOR, Sérgio Ricardo; VIANA, José Aparecido de Aguiar; SILVA, Djalma Domingos da. **Programação em Banco de Dados**. Virtual Books Editora e Livraria Ltda: Patos de Minas – MG, 2014;

ALMEIDA, Francisco Antonio de; BORGES JUNIOR, Sérgio Ricardo. **Modelagem de Dados: um estudo prático**. Virtual Books Editora e Livraria Ltda: Patos de Minas – MG, 2013;

CASTRO, Eduardo B. **Modelagem Lógica de Dados: construção básica e simplificada**. Rio de Janeiro: Campus, 2004.

DATE, C. J. **Introdução a sistemas de banco de dados**. Tradução da 7ª Edição Americana. Rio de Janeiro: Campus, 2000.

ELMASRI, R.; NAVATHE S. B. **Sistemas de Banco de Dados**. 6ª edição. Editora Pearson: São Paulo, 2011.

HEUSER, C.A. **Projeto de Banco de Dados**. 6ª Edição. Porto Alegre: Bookeman, 2009.

STANEK, W.R. **Micorsssoft SQL Server 2008 – Guia de Bolso do Administrador**. São Paulo: Booknman, 2009.

KORT, H. F.; Sudarshan, S; Silberschatz, A. **Sistema de Banco de Dados**. 5ª edição. Editora Campus: Rio de Janeiro, 2006.

PRICE, Jason. **Oracle Database 11g SQL**. São Paulo: Bookman, 2008;