ESCOLA TÉCNICA ESTADUAL
MONTEIRO LOBATO

Modelagem, Projeto e Gestão de Banco de Dados

Curso Técnico em Informática

Sumário

Introdução 1

A tecnologia de informação tem passado por modificações, atingindo toda a programação computacional, de dispositivos móveis a outros dispositivos eletrônicos. Em virtude do crescimento dessas informações, os usuários estão necessitando de formas mais intensas, de espaços para armazenar seus dados. Para toda aplicação desenvolvida, torna-se fundamental a presença de um banco de dados (**SGBD**), do qual tem o objetivo de armazenar os dados feitos via aplicação, possuindo uma interatividade entre base de dados, aplicação e usuário (DEVMEDIA, 2015).

De acordo com Alexandrusk (2010), a humanidade sempre procurou manter registros históricos dos eventos mais importantes para que pudessem ser utilizados posteriormente. Exemplos: Pinturas em cavernas, inscrições hieroglíficas, escritas cuneiformes e a imprensa (a partir do século XV). Os computadores inventados e aperfeiçoados a partir do século XX permitiram que os dados fossem armazenados e recuperados com grande rapidez e facilidade. No início da década de 70 surgiram os SGBDs (Sistemas de Gerenciamento de Banco de Dados). Pesquisas na área resultaram em um conjunto de técnicas, processos e notações para a modelagem ou projeto de banco de dados. Observe, a seguir, alguns conceitos importantes:

1.1 Dado

Podemos definir dado como uma sequência de símbolos quantificados ou quantificáveis. Exemplo: texto (as letras são símbolos quantificados). Também são dados: fotos, figuras, sons gravados e animação, pois todos podem ser quantificados. Um dado é necessariamente uma entidade matemática e, desta forma, é puramente sintático. Isto significa que os dados podem ser totalmente descritos através de representações formais, estruturais. Sendo ainda quantificados ou quantificáveis, eles podem obviamente ser armazenados em um computador e processados por ele.

1.2 Informação

Informação é uma abstração informal (não pode ser formalizada através de uma teoria lógica ou matemática), que está na mente de alguém, representando algo significativo para essa pessoa.

Se a representação da informação for feita por meio de dados, pode ser armazenada em um computador. Mas, o que é armazenado no computador não é a informação, mas a sua representação em forma de dados.

1.3 Conhecimento

Conhecimento pode ser caracterizado como uma abstração interior, pessoal, de algo que foi experimentado, vivenciado, por alguém. Nesse sentido, o conhecimento não pode ser descrito, o que se descreve é a informação. Também não depende apenas de uma interpretação pessoal, como a informação, pois requer uma vivência do objeto do conhecimento.

1.4 Banco de dados

Um banco de dados pode ser definido como:

- Uma coleção de dados integrados que tem por objetivo atender a uma comunidade de usuários;
- Um conjunto de dados persistentes e manipuláveis que obedecem a um padrão de armazenamento. Exemplos: lista telefônica, dicionário, etc.

1.5 Sistema de Gerenciamento de Arquivos (FMS – File Management System)

Foi a primeira forma utilizada para armazenamento de dados. Baseia-se no armazenamento dos dados de forma sequencial em um único arquivo. Apresenta como vantagem a simplicidade na forma em que os dados são estruturados no arquivo. No entanto, esse sistema não apresenta relação entre os dados, nem mecanismos de busca, classificação e recursos para evitar problemas de integridade.

1.6 Sistema de Gerenciamento de Banco de Dados (SGBD)

O Sistema de Gerenciamento de Banco de Dados é o software que incorpora as funções de definição, recuperação e alteração de dados em um banco de dados.

Um banco de dados é um conjunto organizado de dados (segundo um modelo de dados) existentes num sistema informático, disponíveis a todos os utilizadores ou processamentos da organização em que o acesso e atualização são realizados através de software específico. O Sistema de Gerenciamento de Banco de Dados (SGBD) ou Data Base Management System, permite criar base de dados, modificar base de dados, eliminar bases de dados, inserir dados na base de dados e eliminar dados da base de dados (DEVMEDIA, 2015).

Bancos de dados informatizados apresentam as seguintes vantagens:

- Compacto (elimina arquivos de papéis);
- Rápido;
- Integrado (vários aplicativos utilizam o mesmo repositório de dados);
- Compartilhado (vários usuários podem acessar);
- Seguro (controle de acesso);
- Padronizado;
- Consistente;
- Suporte a transações.

1.7 Modelo de dados

Modelo de dados refere-se à descrição formal da estrutura de um banco de dados. A modelagem de dados é uma técnica usada para a especificação das regras de negócios e as estruturas de dados de um banco de dados. Ela faz parte do ciclo de desenvolvimento de um sistema de informação e é de vital importância para o bom resultado do projeto.

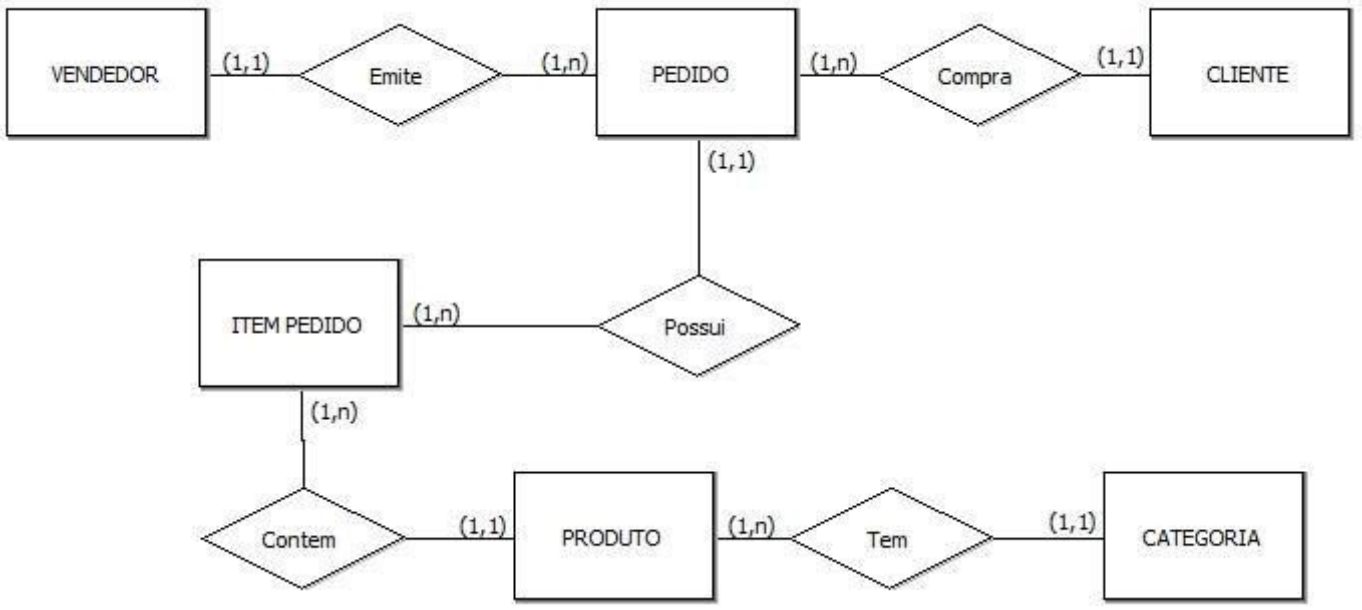
Modelar dados consiste em desenhar o sistema de informações, concentrando-se nas entidades lógicas e nas dependências lógicas entre essas entidades. Modelagem de dados ou modelagem de banco de dados envolve uma série de aplicações teóricas e práticas,

visando construir um modelo de dados consistente, não redundante e perfeitamente aplicável em qualquer SGBD moderno. A modelagem de dados está dividida em: Modelo conceitual, Modelo Lógico e Modelo Físico.

1.7.1 Modelo conceitual

A modelagem conceitual baseia-se no mais alto nível e deve ser usada para envolver o cliente, pois o foco aqui é discutir os aspectos do negócio do cliente e não da tecnologia. Os exemplos de modelagem de dados vistos pelo modelo conceitual são mais fáceis de compreender, já que não há limitações ou aplicação de tecnologia específica. O diagrama de dados que deve ser construído aqui é o Diagrama de Entidade e Relacionamento, onde deverão ser identificados todas as entidades e os relacionamentos entre elas. Este diagrama é a chave para a compreensão do modelo conceitual de dados.

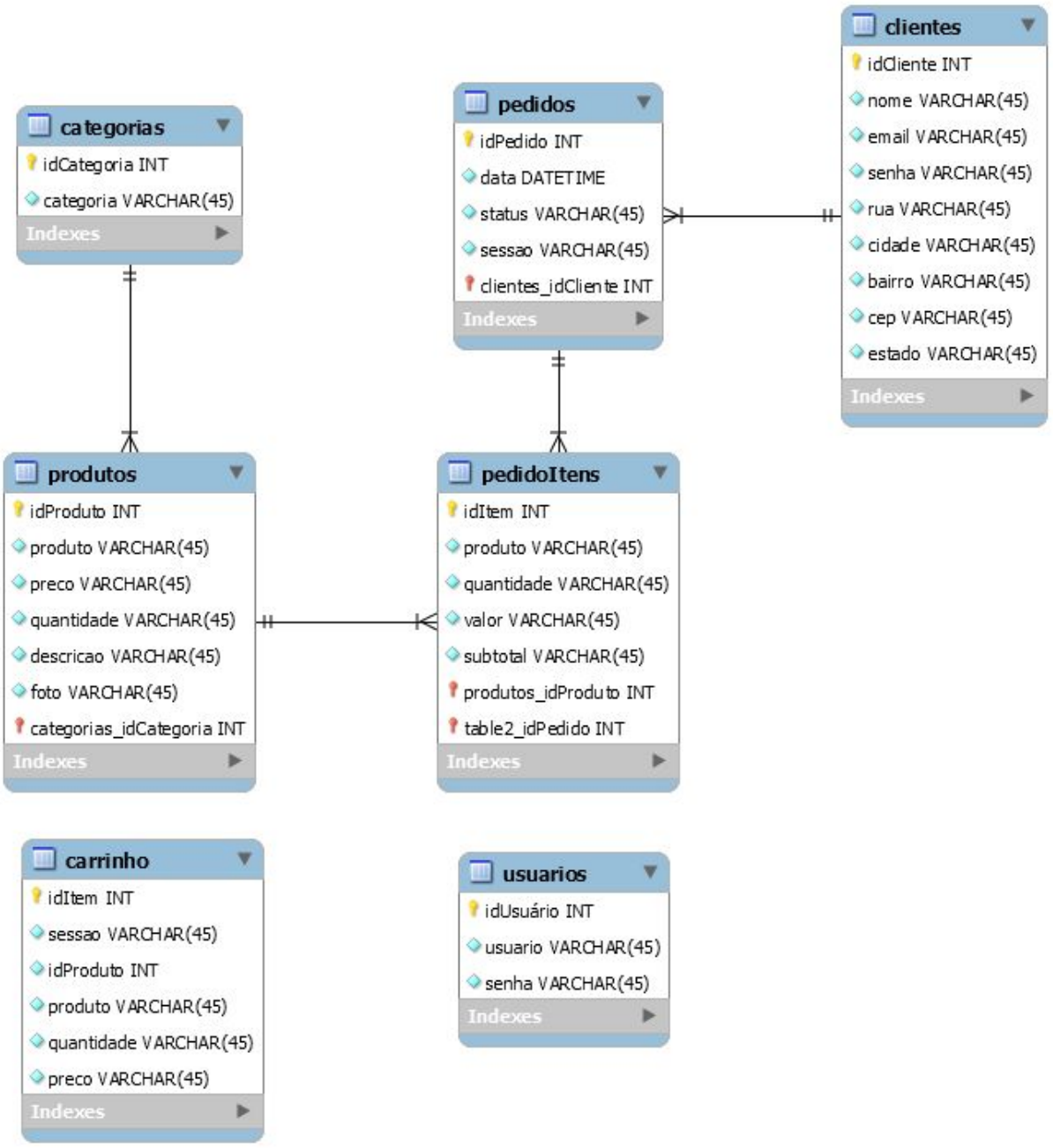
Exemplo simples de Diagrama de Entidade e Relacionamento:



1.7.2 Modelo lógico

O modelo lógico já leva em conta algumas limitações e implementa recursos como adequação de padrão e nomenclatura, define as chaves primárias e estrangeiras, normalização, integridade referencial, entre outras. Para o modelo lógico deve ser criado levando em conta os exemplos de modelagem de dados criados no modelo conceitual.

Exemplo do diagrama de banco de dados, representando o modelo lógico:



1.7.3 Modelo físico

No modelo físico fazemos a modelagem física do modelo de banco de dados. Neste caso leva-se em conta as limitações impostas pelo SGBD escolhido e deve ser criado sempre com base nos exemplos de modelagem de dados produzidos no item anterior, modelo lógico. O assunto é extenso e nem sempre de fácil compreensão, mas com um pouco de prática fica mais fácil o entendimento.

1.8 Profissões

Os profissionais que trabalham com banco de dados podem atuar em diferentes áreas específicas, logo surgem diferentes profissões. Vejamos elas:

1.8.1 Assistente de processamento de dados

Realiza implantação, conservação e alimentação do banco de dados e atualiza informações em sistema.

1.8.2 Analista de Business Intelligence

Analisa e levanta requisitos, define modelagem e modelos de dados a serem tratados e transformados e estuda a melhor ferramenta para extração de dados.

1.8.3 Analista de Banco de Dados

Administra banco de dados, envolvendo manutenção de estrutura e implementação de novos processos de software, métodos de acesso e dimensionamento de hardware e mantém segurança conforme políticas da empresa.

1.8.4 Analista de DBM

Realiza a gestão do DBM(Data Base Marketing) da companhia junto ao fornecedor contratado, acompanhando de perto as melhorias e as alterações. Entende as regras de negócio da empresa e replica os conhecimentos na estrutura do DBM. Implementa e acompanha métricas de negócios e realiza melhorias nas bases de dados e nos processos da área de DBM. Atua com a criação e gestão de mailings extraídos da base para suporte nas ações mercadológicas. Elabora apresentações sobre conclusões das análises descritivas.

1.8.5 Administrador de Banco de Dados - DBA

Planeja e executa as manutenções em bancos de dados de produção. Pesquisa inovações tecnológicas para banco de dados. Garante que backups de produção estejam sendo executados. Maximiza e aprimora a performance do banco de dados. Garante a segurança dos bancos de dados de produção. Administra servidores de bancos de dados de produção. Identifica riscos de atrasos nos trabalhos. Presta suporte aos usuários, orienta analistas e desenvolvedores na otimização de performance das aplicações referente a área de banco de dados.

1.9 Modelagem de Banco de Dados

A modelagem de dados visa fornecer um modelo de dados referente à descrição formal da estrutura de um banco de dados. É uma técnica usada para a especificação das regras de negócios e as estruturas de dados de um banco de dados. Sendo assim, faz parte do ciclo de desenvolvimento de um sistema de informação e é de fundamental importância para o sucesso do projeto. Ao modelarmos os dados estamos desenhando o sistema com suas informações destacando as entidades lógicas e as dependências lógicas entre essas. Ao fazermos isto estamos usando uma série de aplicações teóricas e práticas, procurando construir um modelo de dados consistente, não redundante e perfeitamente aplicável em qualquer SGBD.

1.9.1 Modelo Hierárquico (HDS – Hierarquical Database System)

Na década de 1960 surgiu a primeira linguagem de banco de dados, chamada DL/I, desenvolvida pela IBM e a North American Aviation. Esta linguagem organizava os dados de cima para baixo, como uma árvore. Cada registro era dividido em partes denominadas segmentos. Este modelo de banco de dados se assemelha a um organograma com um segmento raiz e um

número qualquer de segmentos subordinados. Os segmentos são arranjados em estruturas com um segmento superior ligado a um segmento subordinado em um relacionamento “pai-filho”. Um segmento “pai” pode ter mais de um “filho”, mas um segmento “filho” só pode ter um “pai”. A desvantagem apresentada é rigidez da estrutura de dados, que obriga refazer todo o banco de dados, caso o seguimento raiz ou os seguimentos que possuem dependentes sejam alterados. O sistema comercial mais divulgado do modelo hierárquico foi o IMS (Information Management System) da IBM Corporation.

1.9.2 Modelo em Rede (NDS - Network Database System)

Definidos pelo DBTG (DataBase Task Group) do comitê do CODASYL (Conference on Data Systems Language) a partir de 1971. Esse modelo é uma extensão do modelo hierárquico. Os registros são organizados no banco de dados por um conjunto arbitrário de gráficos. Em outras palavras, um “filho” pode ter mais de um “pai”. Esta metodologia torna a pesquisa mais rápida e mais flexível, pois não depende de um único nó raiz como vetor de inicialização de pesquisa. Entretanto, o modelo em rede ainda apresenta os mesmos problemas com relação ao projeto de estrutura do modelo hierárquico. Qualquer alteração feita em uma classe de dados implica na criação de uma nova estrutura para suportar aquela alteração. No modelo em rede um dos sistemas mais conhecidos é o CA IDMS da Computer Associates.

1.9.3 Modelo Relacional (Relational Model)

Foi apresentado por Edgard F. Codd (IBM) em seu artigo, A Relational Model of Data for Large Shared Data Banks (1970). Foi o evento mais importante na história recente da área de banco de dados. O objetivo do modelo é representar os dados de forma mais simples, através de um de conjuntos de tabelas inter-relacionadas. Este modelo abandona os conceitos anteriores, tornando os bancos de dados mais flexíveis, tanto na forma de representar as relações entre os dados, como na tarefa de modificação de sua estrutura, sem ter que reconstruir todo o banco de dados. Os primeiros produtos relacionais começaram a aparecer no final da década de 1970. Hoje a maioria dos sistemas de banco de dados é relacional:

- IBM: DB2
- Microsoft: SQL Server
- Oracle: 9i, 10g, 11g
- MySQL
- PostgreSQL

A principal linguagem de manipulação de dados em sistemas de bancos de dados relacionais é o SQL (Structured Query Language).

1.9.4 Modelo Orientado a Objetos

Surgiu em meados de 1980 para armazenamento de dados complexos, não adequados aos sistemas relacionais. Exemplos: GIS (Geographical Information System) e CAD/CAM/CAE.

O modelo de banco de dados orientado a objetos é baseado nos conceitos de orientação a objetos já difundidos em linguagens de programação como o SmallTalk e o C++. Seu objetivo principal é tratar os tipos de dados complexos como um tipo abstrato (objeto).

A filosofia do modelo de dados orientado a objetos consiste em agrupar os dados e o código que manipula estes dados em um único objeto, estruturando-os de forma que possam ser agrupados em classes. Isso significa que os objetos de banco de dados agrupados podem usar o mesmo mecanismo de herança para definir superclasses e subclasses de objetos, criando assim hierarquias.

O OMDG (Object Database Management Group) definiu um padrão de estrutura para bancos de dados orientados a objetos. O grupo propôs um padrão conhecido como ODMG-93, atualmente revisado e denominado ODMG 2.0.

Quando os bancos de dados orientados a objetos foram introduzidos, algumas das falhas perceptíveis do modelo relacional pareceram ter sido solucionadas e acreditava-se que tais bancos de dados ganhariam grande parcela do mercado.

Hoje, porém, acredita-se que os bancos de dados orientados a objetos serão usados em aplicações especializadas, enquanto os sistemas relacionais continuarão a sustentar os negócios tradicionais, onde as estruturas de dados baseadas em relações são suficientes. Utiliza-se geralmente o diagrama de classes UML como esquema para o modelo de dados orientado a objetos.

1.9.5 Sistemas Objeto Relacionais

Fornecedores de bancos de dados relacionais adicionaram a seus produtos capacidade de incorporar objetos mais complexos (imagem, som e vídeo) além de recursos de orientação a objetos. No entanto, isso não os torna sistemas puramente orientados a objetos, apesar de sua denominação ORDMS – Object-Relacional Database Management System (Sistema de Gerenciamento de Banco de Dados Objeto-Relacional).

Esses sistemas na realidade implementam uma camada de abstração de dados em cima de métodos relacionais, o que torna possível a manipulação de dados mais complexos. Seguem, portanto, as especificações da SQL3 que fornecem capacidades estendidas e de objetos adicionadas ao padrão SQL.

Todas as características relacionais permanecem, ou seja, as tabelas continuam a existir, porém elas possuem alguns recursos adicionais. Anteriormente, as tabelas apenas podiam conter valores atômicos em seus atributos, agora pode-se definir novos tipos de dados e usá-los para receber valores complexos.

Alguns Sistemas de Gerenciamento de Banco de Dados Objeto-Relacionais: Informix, IBM DB2, Oracle 10g.

1.9.6 Sistemas de banco de dados NoSQL

Os bancos NoSQL se referem a uma classe definida de banco de dados não-relacionais, que rompem com uma longa história de banco de dados focados nas propriedades ACID.

São criados para modelos de dados específicos, possuem esquemas flexíveis para a criação de aplicativos modernos. Os bancos de dados NoSQL são amplamente reconhecidos por sua facilidade de desenvolvimento, funcionalidade e performance em escala. Eles usam vários modelos de dados, incluindo documento, gráfico, chave-valor, memória e pesquisa.

Durante décadas, o modelo de dados predominante usado para desenvolvimento de aplicativos foi o modelo usado por bancos de dados relacionais, como Oracle, DB2, SQL Server, MySQL e PostgreSQL. Somente em meados dos anos 2000 que outros modelos de dados

começaram a ser adotados e ter um uso mais significativo. Para diferenciar e categorizar essas novas classes de bancos e modelos de dados, o termo “NoSQL” foi criado. Muitas vezes, o termo “NoSQL” é usado de forma intercambiável com “não relacional”.

Este modelo de banco de dados é ideal para muitos aplicativos modernos, como dispositivos móveis, Web e jogos, que exigem bancos de dados flexíveis, escaláveis, de alta performance e altamente funcionais para proporcionar ótimas experiências aos usuários. Os NoSQL’s geralmente fornecem esquemas flexíveis que permitem um desenvolvimento mais rápido e interativo, os tornando ideais para dados semiestruturados e não estruturados. Os bancos de dados possuem escalabilidade e geralmente são projetados para serem escalados horizontalmente usando clusters distribuídos de hardware, em vez de escalá-los verticalmente adicionando servidores caros e robustos. Alguns provedores de nuvem, lidam com essas operações nos bastidores, como um serviço totalmente gerenciado. Possui uma alta performance e é otimizado para modelos de dados específicos (como documento, chave-valor e gráfico) e padrões de acesso que permitem maior performance do que quando se tenta realizar uma funcionalidade semelhante com bancos de dados relacionais. Também fornecem APIs e tipos de dados altamente funcionais criados especificamente para cada um de seus respectivos modelos de dados.

1.10 Abstração de Dados

A arquitetura ANSI/SPARC (American National Standards Institute – Standards Planning And Requirements Committee) se divide em três níveis: interno, conceitual e externo, embora outros nomes também sejam utilizados:

1.10.1 Nível Interno (OU DE ARMAZENAMENTO):

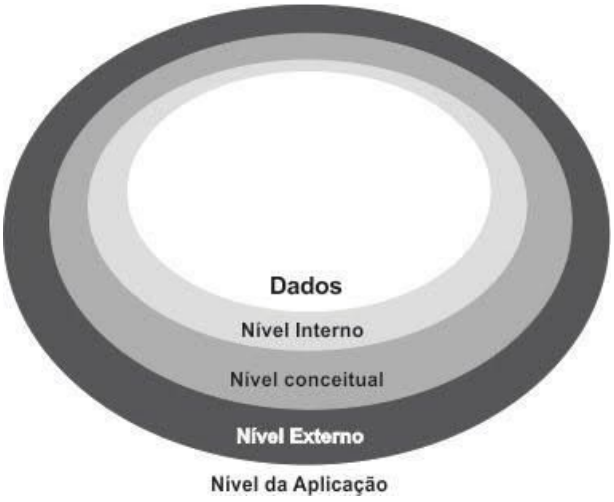
É o nível mais baixo de abstração e o mais próximo do armazenamento físico. Descreve como os dados estão de fato armazenados.

1.10.2 Nível Conceitual (LÓGICO OU LÓGICO DE COMUNIDADE):

Descreve quais dados estão armazenados e quais os relacionamentos entre eles. É uma visão do conteúdo total do banco de dados.

1.10.3 Nível Externo (LÓGICO DO USUÁRIO):

É o nível mais alto de abstração e o mais próximo dos usuários. É aquele que se ocupa do modo como os dados são vistos por usuários individuais.



1.10.4 Independência de Dados

É a capacidade de modificar a definição dos esquemas em determinado nível, sem afetar o esquema do nível superior:

Independência de dados física: é a capacidade de modificar o esquema físico sem que, com isso, qualquer programa ou aplicação precise ser reescrito.

Independência de dados lógica: é a capacidade de modificar o esquema lógico sem que, com isso, qualquer programa ou aplicação precise ser reescrito. A independência lógica é mais difícil de ser alcançada, uma vez que as aplicações são mais fortemente dependentes da estrutura lógica dos dados do que de seu acesso.

Exercícios:

1) Elabore um texto (mais ou menos vinte linhas) discorrendo sobre a importância dos registros de dados para uma empresa e para a sociedade em geral.

2) O que você entende por: informação, dado e conhecimento?

3) O que é um banco de dados? O que é um SGBD (Sistema Gerenciador de Banco de Dados)?

4) Cite cinco razões para utilizar um banco de dados computadorizado.

5) Fale brevemente sobre os seguintes modelos de banco de dados:

a. hierárquico

b. em rede

c. relacional

6) Quais são os três níveis de abstração de dados? Comente brevemente sobre eles.

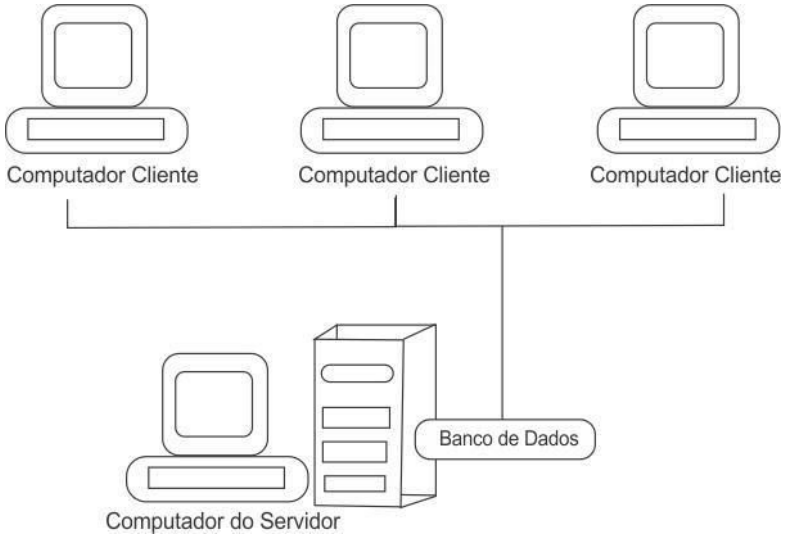
Arquiteturas de Sistemas de Bancos de Dados

2

Nesta seção são apresentadas as principais arquiteturas de SGBDs e fundamentos da modelagem de dados.

2.1 Sistemas Centralizados

Sistemas centralizados são aqueles executados em grandes computadores centrais (mainframes). Os programas de aplicação e os de interface com os usuários, bem como as funcionalidades do SGBD (Sistema Gerenciador de Banco de Dados) são todos processados no sistema central. Os usuários acessam o sistema central via terminal, sem poder de processamento, através de uma rede de comunicação.



2.2 Sistemas Cliente Servidor

A estrutura fundamental dos sistemas cliente-servidor consiste de estações de trabalho (normalmente PCs) conectadas via rede aos servidores que têm funcionalidades específicas: servidor de arquivo, de impressão, web, SGBD, etc.

As máquinas clientes (estações de trabalho) oferecem aos usuários as interfaces apropriadas para utilizar os servidores, bem como poder de processamento para executar aplicações locais.

Sistemas cliente-servidor de três camadas para aplicações Web. Possui uma camada intermediária entre o cliente e o servidor de banco de dados.

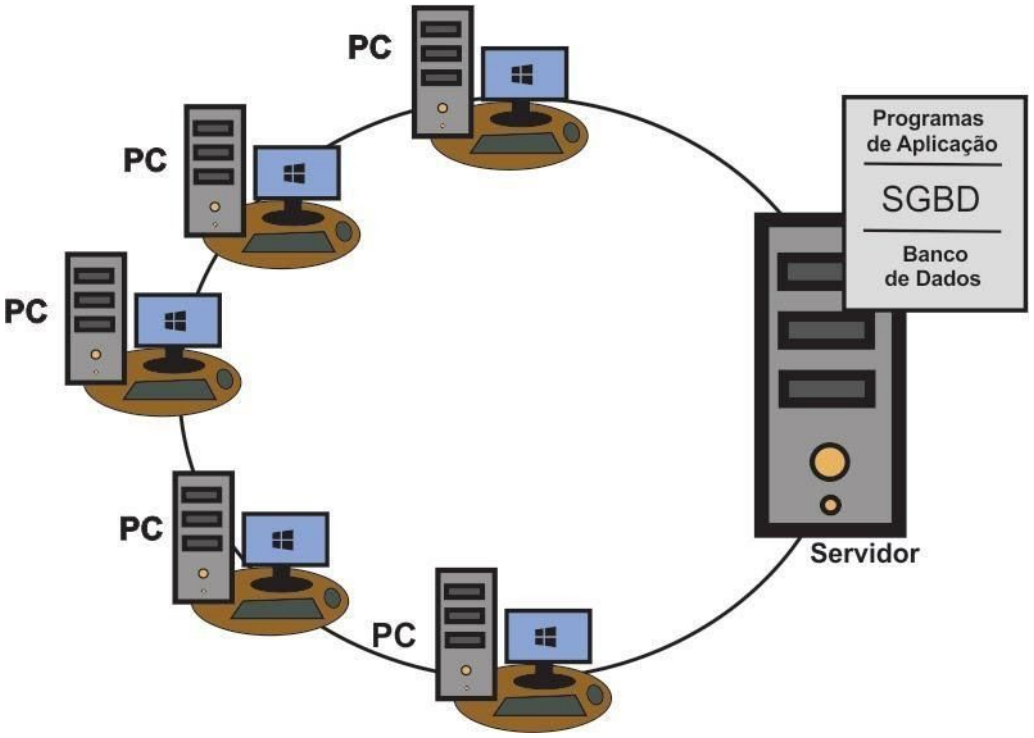
Essa camada intermediária é chamada de servidor de aplicações ou servidor web e pode armazenar regras de negócio (procedimentos ou restrições) que são usadas para acessar os dados no servidor de banco de dados.

2.3 Sistemas Paralelos

Suprem a demanda de aplicações que geram consultas em bancos de dados muito

grandes ou que tenham de processar uma quantidade enorme de transações por segundo.

Sistemas paralelos imprimem velocidade ao processamento e à I/O (input/output) por meio do uso em paralelo de diversas CPUs e discos. Há diversos modelos arquitetônicos: memória compartilhada, disco compartilhado, etc.



2.4 Sistemas Distribuídos

Em um sistema distribuído o banco de dados é armazenado em diversos computadores. Os computadores comunicam-se uns com os outros por intermédio de redes de alta velocidade ou linhas telefônicas. Eles não compartilham memória principal ou discos. Os computadores em um sistema de banco de dados distribuídos podem variar, quanto ao tamanho e funções, desde estações de trabalho até sistemas de grande porte (mainframes)

3 Modelagem de Banco de Dados

A modelagem de dados objetiva prover um modelo de dados referente à descrição formal da estrutura de um banco de dados. É um método utilizado para a especificação das regras de negócios e as estruturas de dados de um banco de dados. Sendo assim, faz parte do ciclo de desenvolvimento de um sistema de informação e é de vital importância para o sucesso do projeto.

Modelar dados significa criar um desenho do sistema de informações, concentrando-se nas entidades lógicas e nas dependências lógicas entre essas entidades. Esta etapa envolve uma série de aplicações teóricas e práticas, visando construir um modelo de dados consistente, não redundante e perfeitamente aplicável em qualquer SGBD.

O projeto de um banco de dados ocorre geralmente observando-se as seguintes etapas:

3.1 Levantamento e Análise de Requisitos

É a primeira etapa do projeto de um sistema de aplicação em banco de dados. O analista entrevista o(s) usuário(s) do banco de dados para fazer o levantamento dos requisitos de dados.

Esses requisitos devem ser especificados em um formulário de forma detalhada e completa. É importante definir também os requisitos funcionais da aplicação, isto é, as operações (transações) definidas pelo usuário que serão aplicadas ao banco de dados.

3.2 Modelo Conceitual

É a próxima etapa do projeto de um sistema de aplicação em banco de dados. Representa ou descreve a realidade do ambiente do problema, constituindo-se em uma visão global dos principais dados e relacionamentos, independente das restrições de implementação. É uma descrição em alto nível (macro definição), mas que tem a preocupação de capturar e retratar toda a realidade de uma organização. O resultado de um modelo conceitual é um esquema que representa a realidade das informações existentes, assim como as estruturas de dados que representam estas informações.

3.3 Modelo Lógico

Tem seu início a partir do modelo conceitual, levando em consideração três abordagens principais: Relacional (atualmente o mais utilizado), Hierárquica e Rede. O modelo lógico descreve as estruturas que estarão contidas no banco de dados, mas sem considerar ainda nenhuma característica específica de SGBD, resultando em um esquema lógico de dados.

3.4 Modelo Físico

Parte do modelo lógico e descreve as estruturas físicas de armazenamento de dados, tais como: tamanhos de campos, índices, tipos de dados, nomenclaturas, etc.

Este modelo detalha o estudo dos métodos de acesso do SGDB, para elaboração dos índices de cada informação colocada nos modelos conceitual e lógico.

É a etapa final do projeto de banco de dados, na qual será utilizada a linguagem de definição de dados (DDL), para a realização da montagem do mesmo no nível de dicionário de dados.

Exercícios:

1) Descreva brevemente os seguintes sistemas: centralizados, cliente-servidor, paralelos e distribuídos.

a. Centralizados

b. Cliente-servidor

c. Paralelos

d. Distribuídos

2) Quais etapas devem ser observadas na modelagem de dados? Explique cada uma delas.

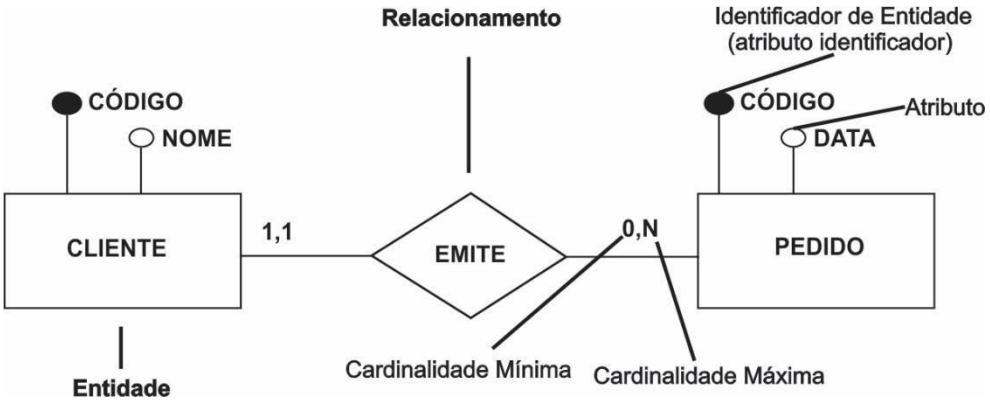
3) Como as corporações estão utilizando os dados para obter vantagens competitivas? Apresente um exemplo.

Modelo Conceitual

Nesta seção são apresentados conceitos fundamentais da modelagem de dados a partir de sua primeira etapa: o modelo conceitual.

No modelo conceitual é feita a descrição do banco de dados de forma independente de implementação em um SGBD. Registra que dados podem aparecer no banco de dados, mas não registra como estes dados estão armazenados no SGDB (Sistema Gerenciador de Banco de Dados).

4.1 Diagrama Entidade-Relacionamento



O MER (Modelo Entidade-Relacionamento Relacionamento), foi definido por Peter Chen em 1976, e teve como base a teoria relacional criada por Edgard F. Codd (1970).

Um MER é um modelo formal, preciso, não ambíguo. Isto significa que diferentes leitores de um mesmo MER devem sempre entender exatamente o mesmo. Tanto é assim, que um MER pode ser usado como entrada de uma ferramenta CASE (Computer Aided Software Engineering) na geração de um banco de dados relacional.

4.1.1 Entidades

Representam um conjunto de objetos (tudo que é perceptível ou manipulável) da realidade modelada sobre os quais deseja-se manter informações no banco de dados.

4.1.2 Atributos

Dados que são associados a cada ocorrência de uma entidade ou de um relacionamento.

ATRIBUTO IDENTIFICADOR OU ATRIBUTO DETERMINANTE - Atributo ou conjunto de atributos e relacionamentos cujos valores distinguem uma ocorrência da entidade das demais.

4.1.3 Relacionamentos

Relacionamentos representam a interação entre as entidades que indicam a dinâmica dos negócios que estão sendo modelados.

Os relacionamentos são identificados por verbos, já que identificam ações que uma

entidade exerce sobre a outra.

4.1.4 Cardinalidade

Número (mínimo, máximo) de ocorrências de entidade associadas a uma ocorrência da entidade em questão através do relacionamento.

4.1.3.1 Cardinalidade mínima

É o número mínimo de ocorrências de entidade que são associadas a uma ocorrência da mesma (auto-relacionamento) auto ou de outra(s) entidade(s) entidade através de um relacionamento.

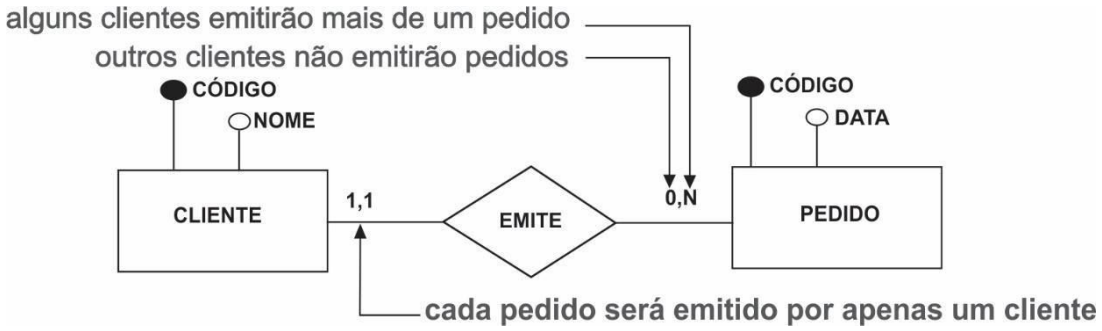
A cardinalidade mínima 1 recebe a denominação de associação obrigatória, já que ela indica que o relacionamento deve obrigatoriamente associar uma ocorrência de entidade a outra. A cardinalidade mínima 0 (zero) recebe a denominação de associação opcional.

4.1.3.2 Cardinalidade máxima

É o número máximo de ocorrências de entidade que são associadas a uma ocorrência da mesma ou de outra entidade através de um relacionamento. Apenas duas cardinalidades máximas são relevantes: a cardinalidade máxima 1 e a cardinalidade máxima n (muitos).

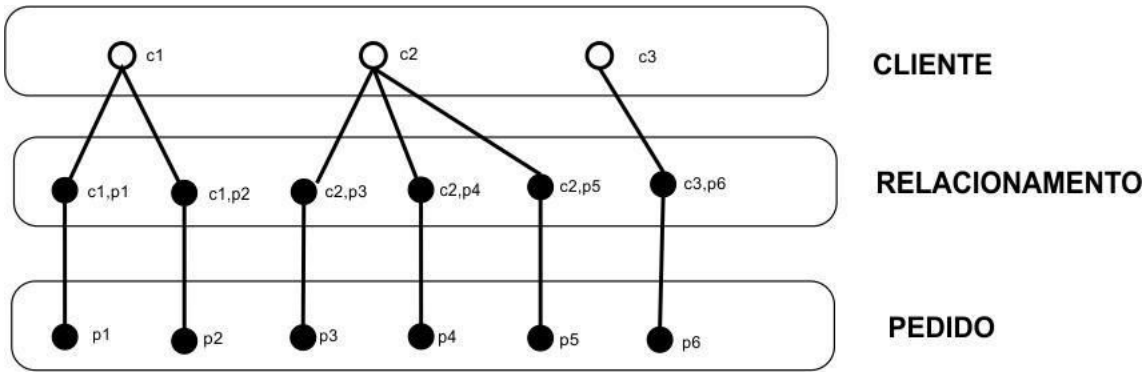
4.2 Interpretação do Diagrama Entidade Relacionamento

A figura a seguir apresenta os detalhes necessários à interpretação de um DER (Diagrama Entidade Relacionamento).



4.2.1 Diagrama de Ocorrências

Para fins didáticos, pode ser útil construir um diagrama de ocorrências. Neste as ocorrências de entidades são representadas por círculos brancos e ocorrências de relacionamentos por círculos pretos. As ocorrências de entidades participantes de uma ocorrência de relacionamento são indicadas pelas linhas que ligam o círculo preto aos círculos brancos.



Exercícios:

1) Explique o que é um modelo conceitual de dados.

2) Quem apresentou pela primeira vez o MER (Modelo Entidade Relacionamento) e baseando-se em que?

3) Explique com suas palavras cada um dos itens a seguir:

a. Entidade

b. Atributo

c. Identificador de entidade (ou atributo identificador)

d. Cardinalidade

4) O que é um diagrama de ocorrências?

5) Quais as vantagens de desvantagens de utilizar-se o RG ou o CPF de um cliente como atributo identificador? Por que muitas empresas atribuem um número ou código próprio para identificar seus clientes?

6) Elabore o DER (Diagrama Entidade Relacionamento) e o Diagrama de Ocorrências para cada caso.

Várias empresas possuem frotas de veículos que são identificados através da placa (XYZ-1234). São registrados também os fabricantes e modelos de cada veículo. Os funcionários são identificados através do número de matrícula. São mantidos registros do nome e CPF de cada funcionário. Criar o DER (Diagrama Entidade Relacionamento) para cada um dos casos descritos a seguir:

a) Empresa A: Cada veículo (sem exceção) é dirigido por um apenas funcionário. Todos os veículos estão alocados aos funcionários. Cada funcionário pode utilizar apenas um veículo e todos os funcionários têm veículos pertencentes à frota da empresa.

b) Empresa B: Cada veículo (sem exceção) é dirigido por um apenas funcionário. Todos os

veículos estão alocados aos funcionários. Cada funcionário pode utilizar apenas um veículo, porém alguns funcionários não têm veículos pertencentes à frota da empresa.

c) Empresa C: Cada veículo pode ser dirigido por um ou mais funcionários. Todos os veículos estão alocados aos funcionários. Alguns funcionários podem utilizar mais um veículo e todos os funcionários têm veículos pertencentes à frota da empresa.

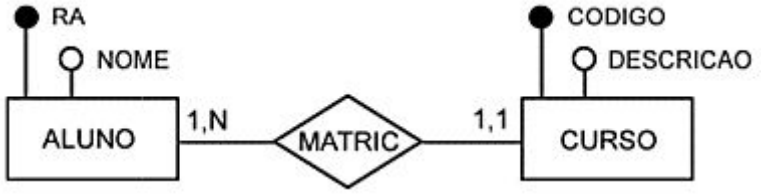
d) Empresa D: Cada veículo (sem exceção) é dirigido por um apenas funcionário. Todos os veículos estão alocados aos funcionários. Alguns funcionários podem utilizar mais de um veículo, porém alguns funcionários não têm veículos pertencentes à frota da empresa.

e) Empresa E: Cada veículo pode ser dirigido por um ou mais funcionários. Todos os veículos estão alocados aos funcionários. Cada funcionário pode utilizar apenas um veículo e todos os funcionários têm veículos pertencentes à frota da empresa.

f) Empresa F: Alguns veículos podem ser dirigidos por mais de um funcionário. Porém, outros veículos não podem ser alocados aos funcionários. Cada funcionário pode utilizar apenas um veículo, porém alguns funcionários não têm veículos pertencentes à frota da empresa.

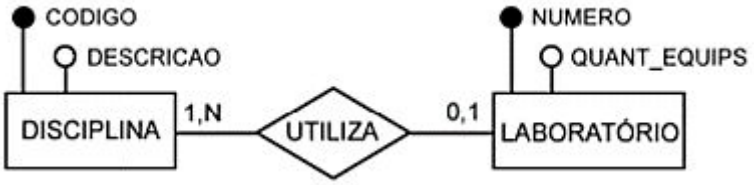
7) Tomando como base os diagramas a seguir elabore um texto breve (similar aos apresentados nas questões acima) para explicar cada caso.

a)



b)

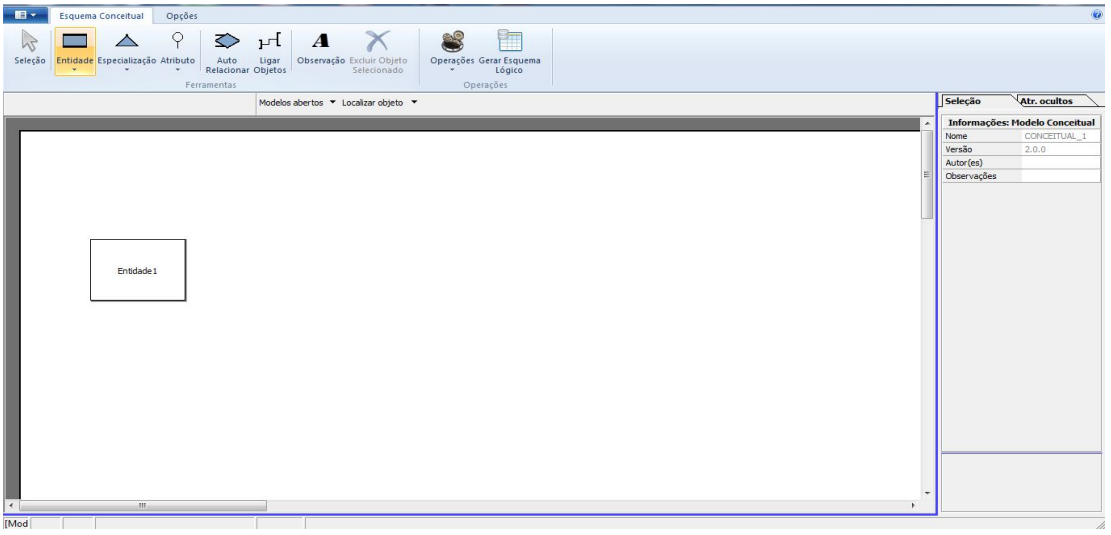
c)



4.3 Ferramenta

Uma ferramenta que pode ser utilizada é a BrModelo, ferramenta freeware voltada para ensino de modelagem em banco de dados. Desenvolvida por Carlos Henrique Cândido como

trabalho de conclusão do curso de pós-graduação graduação em banco de dados da (UNVAG-MT e UFSC). Disponível em: http://sis4.com/brModelo/download.aspx



4.4 Graus de Relacionamentos

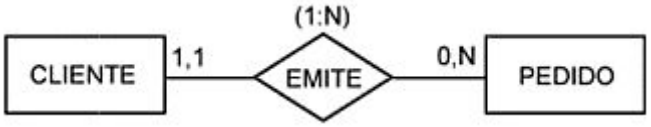
O modelo conceitual apresenta vários graus de relacionamentos: binário, auto relacionamento e ternário(n-ário).

O grau de um relacionamento refere-se ao número de entidades que participam de um relacionamento. Observe a seguir os diversos graus de relacionamentos:

4.4.1 Relacionamento Binário

Um relacionamento binário é aquele envolve duas ocorrências de entidade. Podemos classificar os relacionamentos binários em:

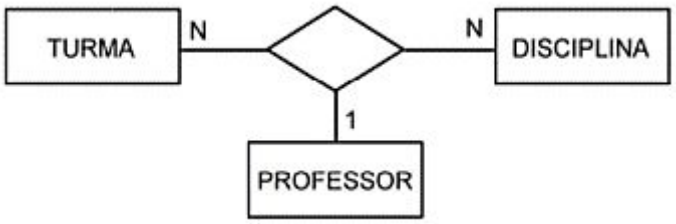
- 1:1 (um-para-um): cada ocorrência de uma entidade relaciona-se relaciona com uma e somente uma ocorrência da outra entidade.
- 1:N (um-para-muitos): uma ocorrência da entidade 1 relaciona-se relaciona com muitas ocorrências da entidade 2, mas cada ocorrência da entidade 2 somente pode estar relacionada com uma ocorrência da entidade 1.
- N:N (muitos-para para-muitos): em ambos os sentidos encontramos um ou mais relacionamentos de um-para-muitos, isto é, uma ocorrência da entidade 1 relaciona-se com muitas ocorrências da entidade 2 e vice e versa.



4.4.2 Relacionamento Ternário (N-ÁRIO)

Denominamos ternários os relacionamentos entre três conjuntos de entidades. Relacionamentos com quatro ou mais conjuntos de entidades são chamados de n-ários. Porém, sua utilização não é recomendada devido a sua complexidade. Sugere-se que sejam “quebrados”

em relacionamentos binários e/ou ternários. No exemplo a seguir queremos garantir que a seguinte situação seja representada de forma apropriada: o professor x ministra a disciplina y para a turma z. Esta condição deve ser representada através de um relacionamento ternário.



Observe que no exemplo apresentado, cada par de ocorrências (turma, disciplina) está associado à no máximo um professor.

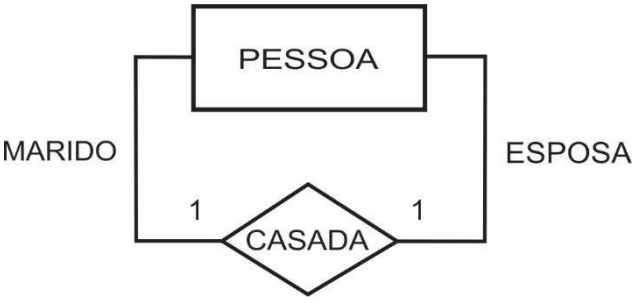
A um par (turma, professor) podem estar associadas muitas disciplinas, ou em outros termos, um professor pode ministrar a uma determinada turma várias disciplinas. A um par (disciplina, professor) podem estar associados muitas turmas, ou em outros termos, um professor pode ministrar uma determinada disciplina a várias turmas.

4.4.3 Auto Relacionamento

Relacionamento entre ocorrências de uma mesma entidade.

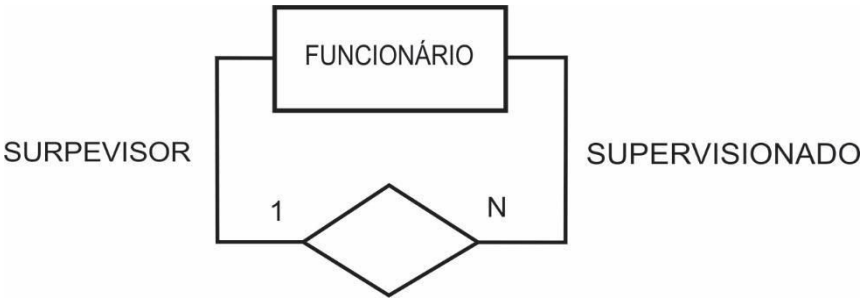
4.4.3.1 Auto Relacionamento 1:1

O diagrama abaixo representa a seguinte situação: uma ocorrência de pessoa exerce o papel de marido e outra ocorrência de pessoa exerce o papel de esposa.



4.4.3.2 Auto Relacionamento 1:N

Abaixo temos representada a seguinte situação: uma ocorrência de funcionário exerce o papel de supervisor e outras ocorrências de funcionário exercem o papel de supervisionado.



4.4.3.3 Auto Relacionamento N:N

E, finalmente, temos representada a seguinte situação: algumas ocorrências de produto exercem o papel de composto e outras ocorrências exercem o papel de componente.

Exercícios:

1) Explique cada um dos graus de relacionamentos abaixo e apresente um exemplo de cada.

a) Relacionamento binário

b) Relacionamento ternário

c) Auto-relacionamento

2) Explique a classificação dos relacionamentos binários quanto à sua cardinalidade máxima (1:1, 1:N e N:N).

4.5 Generalização/Especialização não Exclusiva, Entidade Associativa

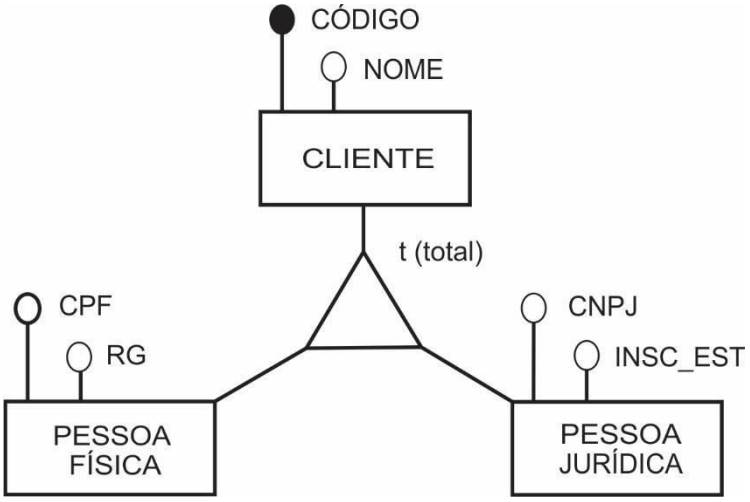
Conceitos sobre generalização/especialização, entidade associativa e atributos opcionais, compostos e multivalorados.

4.5.1 Generalização/Especialização

Através deste conceito é possível atribuir propriedades particulares a um subconjunto das ocorrências especializadas de uma entidade genérica.

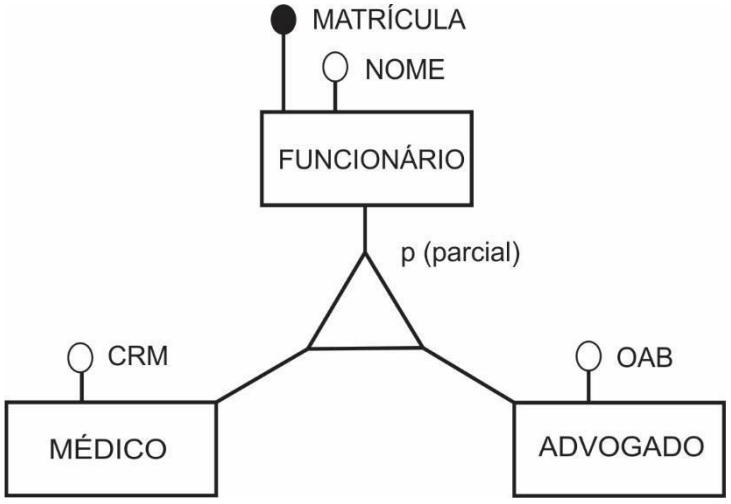
Especialização total: para cada ocorrência da entidade genérica existe sempre uma ocorrência em uma das entidades especializadas.

O exemplo abaixo apresenta uma especialização total: os clientes de uma empresa serão apenas pessoas físicas ou jurídicas.



Especialização parcial: nem toda ocorrência da entidade genérica possui uma ocorrência correspondente em uma entidade especializada.

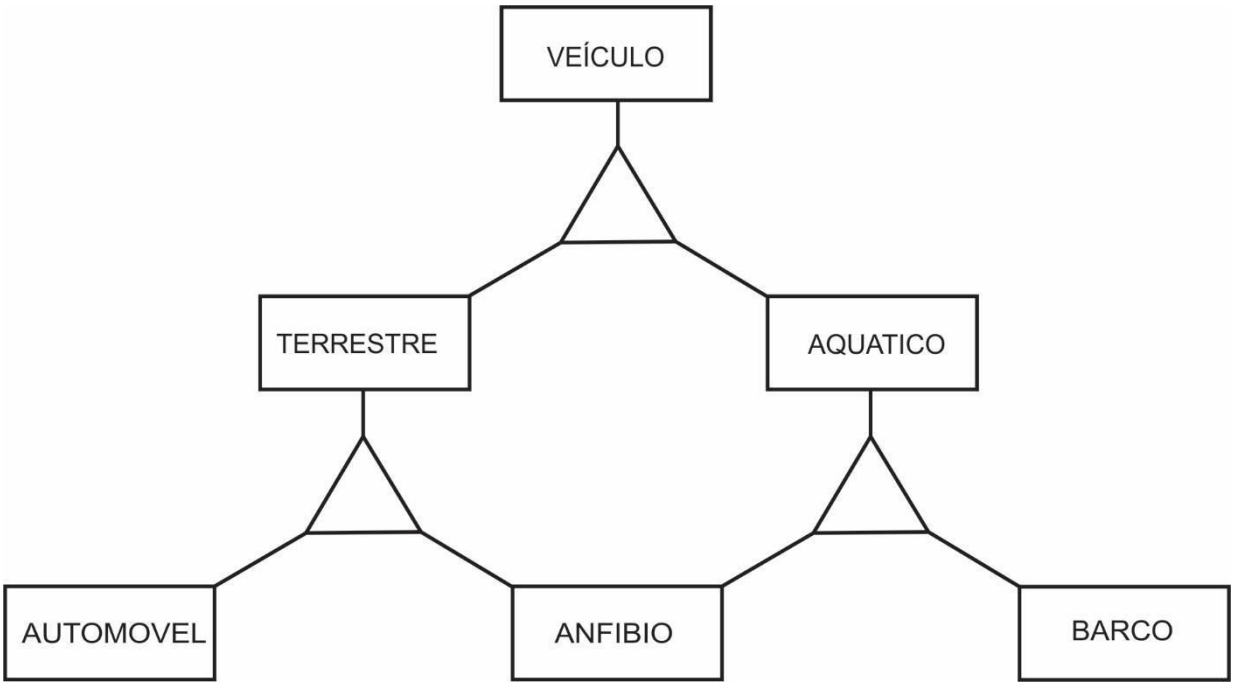
O exemplo a seguir apresenta uma especialização parcial: os funcionários da empresa poderão ter outras profissões além das apresentadas no diagrama abaixo.



4.5.2 Múltiplos Níveis de Herança Múltipla

É admissível que uma mesma entidade seja especialização de diversas entidades genérica (herança múltipla).

No diagrama abaixo o exemplo de herança múltipla aparece na entidade ANFÍBIO (que herda tanto de TERRESTRE quanto de AQUÁTICO).



4.5.3 Herança de Propriedades

Herdar propriedades significa que cada ocorrência da entidade especializada possui, além de suas propriedades (atributos, relacionamentos e generalizações/especializações) também as propriedades da ocorrência da entidade genérica correspondente.

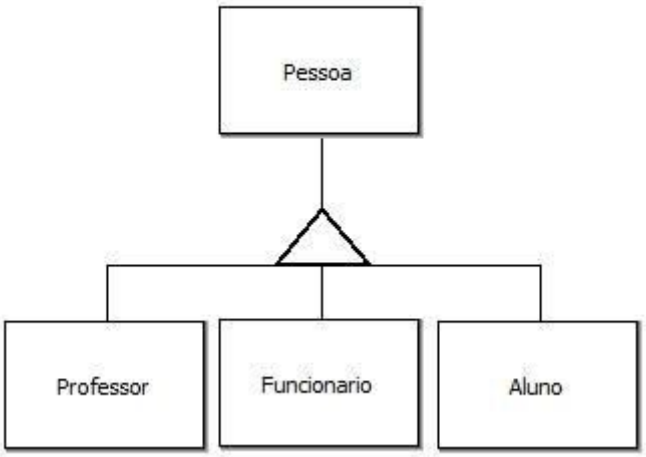
4.5.4 Generalização/Especialização não Exclusiva

Significa que uma ocorrência de entidade genérica aparece, para cada hierarquia generalização/especialização, no máximo uma vez.

4.5.5 Generalização/Especialização não Exclusiva

Neste caso, uma ocorrência da entidade genérica pode aparecer em múltiplas especializações.

No exemplo a seguir, considera-se o conjunto de pessoas vinculadas a uma universidade. Neste caso a especialização não é exclusiva, já que a mesma pessoa pode aparecer em múltiplas especializações. Uma pessoa pode ser professor de um curso e ser aluno em outro curso (pós-graduação, por exemplo). Por outro lado, uma pessoa pode acumular o cargo de professor em tempo parcial com o cargo de funcionário, ou, até mesmo, ser professor de tempo parcial em dois departamentos diferentes, sendo, portanto, duas vezes professor.

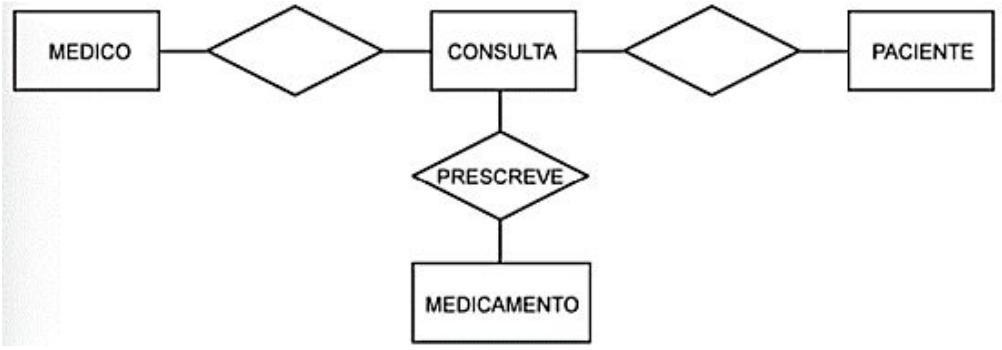


O principal problema que este tipo de generalização/especialização apresenta é que neste caso as entidades especializadas não podem herdar o identificador da entidade genérica. No caso, o identificador de pessoa não seria suficiente para identificar professor, já que uma pessoa pode ser múltiplas vezes professor.

4.5.6 Entidade Associativa

Um relacionamento é uma associação entre entidades. Na modelagem,Entidade-Relacionamento, não foi prevista a possibilidade de associar dois relacionamentos entre si.

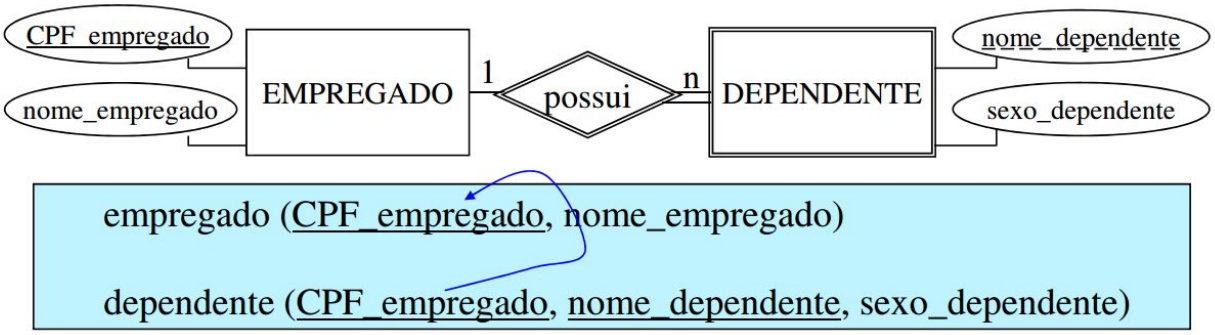
Uma entidade associativa nada mais é que a redefinição de um relacionamento, que passa a ser tratado como se fosse também uma entidade.



4.5.7 Entidade Fraca

Entidade cuja existência depende de estar associada, via um relacionamento (relacionamento de identificação), com uma outra entidade (entidade forte);

Exemplo:



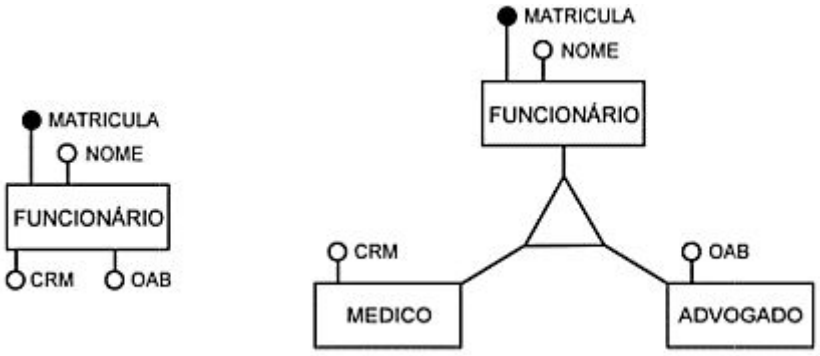
- Considere o relacionamento dependência entre os conjuntos de entidades EMPREGADO e DEPENDENTE.
- A entidade DEPENDENTE contém os dependentes dos EMPREGADOS da empresa. Por isso é uma entidade fraca, e está relacionada a uma entidade forte.
  - Pode se identificar que está relacionada com uma entidade forte: Pelo atributo chave da entidade forte. E o atributo da própria entidade fraca: Chave parcial.

4.6 Atributos

4.6.1 Atributos Compostos

Atributos opcionais são aqueles que se aplicam apenas a determinadas ocorrências de uma entidade, e não a outras. Exemplo: FUNCIONÁRIO e os registros profissionais em diferentes entidades de classe: CRM, CREA, OAB, etc.

Atributos opcionais muitas vezes indicam subconjuntos da entidade que devem ser modelados através de especialização.

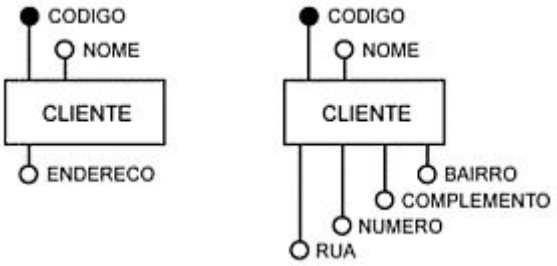


4.6.2 Atributos Compostos

Chamamos que atributos compostos àqueles nos quais o conteúdo é formado por vários itens menores.

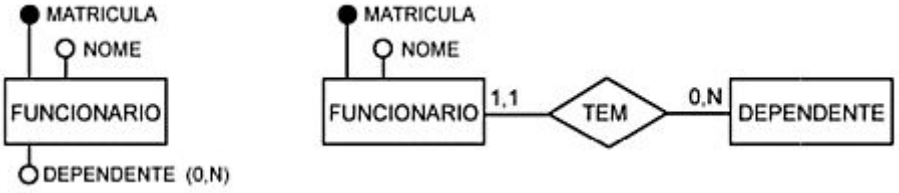
Exemplo: ENDEREÇO é composto por: nome do logradouro, número, complemento

(exemplo: apartamento), bairro, cidade, estado, etc.



4.6.3 Atributos Multivalorados

Denominamos atributos multivalorados àqueles nos quais o conteúdo é formado por mais de um valor. Exemplo: FUNCIONÁRIO e DEPENDENTE, DEPENDENTE este último como atributo. Para atributos multivalorados recomenda-se a solução apresentada na figura a seguir.



Exercícios:

1) O que você entende por:

a) Generalização/Especialização

b) Especialização total

c) Especialização parcial

d) Especialização exclusiva

e) Especialização não exclusiva

2) O que é uma entidade associativa?

3) Explique o que são:

a) Atributos opcionais

b) Atributos compostos

c) Atributos multivalorados

4) Apresente um exemplo para cada um dos seguintes casos:

a) Generalização/Especialização

b) Entidade com atributo opcional

c) Entidade com atributo composto

d) Entidade com atributo multivalorado

5) Diretor

Um diretor dirige apenas um departamento. Um departamento tem apenas um diretor. Determine o do diagrama ER.

6) Autor

Um autor escreve vários livros. Um livro pode ter vários autores. Apresente o diagrama ER.

7) Berçário

Um berçário deseja importar suas operações. Quando um bebê nasce, algumas informações são armazenadas sobre ele: nome, peso, altura, mãe e médico. A respeito da mãe, deve ser o nome, endereço, telefone e data de nascimento. Do médico deve ser informado o CRM, nome e telefone.

Apresente o diagrama ER para o modelo.

8) Sistema escolar

Uma Escola tem várias turmas. Uma turma tem vários professores, sendo que um professor pode ministrar aulas em mais de uma turma. Uma turma tem sempre aulas na mesma sala, mas uma sala pode estar associada a várias turmas (com horários diferentes).

9) Sistema de pedidos

Uma firma vende produtos de limpeza, e deseja melhor controlar os produtos que vende, seus clientes e os pedidos. Cada produto é caracterizado por um código, nome do produto, categoria (ex. detergente, sabão em pó, sabonete, etc), e seu preço. A categoria é uma classificação criada pela própria firma. A firma possui informações sobre todos seus clientes. Cada cliente é identificado por um código, nome, endereço, telefone, status ("bom", "médio", "ruim"), e o seu limite de crédito.

Guarda-se igualmente a informação dos pedidos feitos pelos clientes. Cada pedido possui um número e guarda-se a data de elaboração do pedido. Cada pedido pode envolver de um a vários produtos, e para cada produto, informa-se a quantidade.

10) Sistema de ordens de serviço

Sistema de controle e gerenciamento de execução de ordens de serviço em uma oficina mecânica: Clientes levam veículos à oficina mecânica para serem consertados ou para passarem por revisões periódicas. Cada veículo é designado a uma equipe de mecânicos que identifica os serviços a serem executados e preenche uma ordem de serviço (OS) e prevê uma data de entrega. A partir da OS, calcula-se o valor de cada serviço, consultando-se uma tabela de referência de mão-de-obra. O valor de cada peça necessária à execução do serviço também é computado. O cliente autoriza a execução dos serviços e a mesma equipe responsável pela avaliação realiza os serviços. Clientes possuem código, nome, endereço e telefone. Veículos possuem código, placa e descrição. Cada mecânico possui código, nome, endereço e

especialidade. Cada OS possui um número, uma data de emissão, um valor e uma data para conclusão dos trabalhos. Uma OS pode ser composta de vários itens (serviços) e um mesmo serviço pode constar em várias ordens de serviço. Uma OS pode envolver vários tipos de peças e um mesmo tipo de peça pode ser necessária em várias ordens de serviço.

11) Empresa de distribuição

Uma empresa de distribuição possui vários cinemas, em diversas localidades; Cada cinema possui uma identificação única, um nome fantasia, um endereço completo, incluindo rua, avenida, bairro, município, estado e sua capacidade de lotação; Os filmes podem ser dos mais variados tipos e gêneros; Cada filme é registrado com um título original, e se for filme estrangeiro, possuirá também o título em Português, o gênero, sua duração, sua impropriedade e seu país de origem, informações sobre os atores que compõem seu elenco, e seu diretor. Existirá um único diretor para cada filme; Alguns cinemas apresentam mais de um filme em cartaz, sendo nestes casos, sessões alternadas com um filme e outro; As sessões possuem horários que variam de acordo com a duração do filme, havendo sempre um intervalo de aproximadamente 15 minutos entre elas; Os atores de um filme podem, obviamente, atuar em diversos filmes, assim como o diretor de um filme pode também ser ator neste filme ou ainda mais, ser ator em outro filme. Um ator possui as seguintes características: um número de identificação, um nome, uma nacionalidade e uma idade; As sessões de cinema devem ter seu público registra do diariamente, para que se permita a totalização dos assistentes quando o filme sair de cartaz, ou a qualquer instante;

12) Sistema de Controle Bancário

Faça o esquema conceitual para um sistema de controle bancário. Para cada agência do sistema deseja-se armazenar seu número, cidade e dados sobre os funcionários que ali trabalham, tais como nome, endereço, código e salário. Cada cliente cadastrado em uma agência específica pode possuir várias contas bancárias. Para os clientes deseja-se armazenar o nome, o RG e a cidade na qual residem, além de suas contas bancárias. Dados importantes para as contas dos clientes da agência são o número da conta, o saldo e informações sobre o conjunto de transações (número_transação, data, valor) associadas à conta.

13) Companhia

Faça o esquema conceitual para o banco de dados de uma companhia. A companhia é organizada em departamentos. Cada departamento tem um nome e um número. Além disto, um departamento controla vários projetos, cada um dos quais com um nome, um número de identificação e o período de tempo no qual deve ser desenvolvido. Na referida companhia, cada projeto somente pode ser desenvolvido por um departamento específico.

Existem somente três tipos de funcionários que trabalham na companhia: pesquisador, secretário e de limpeza. Para os pesquisadores, deseja-se armazenar: o nome, o endereço, o sexo, a data de aniversário, o salário e a área de atuação. Para os secretários, deseja-se armazenar: o nome, o endereço, o sexo, a data de aniversário, o salário e o grau de escolaridade. Já para os funcionários de limpeza, deseja-se armazenar: o nome, o endereço, o sexo, a data de aniversário, o salário, o cargo e a jornada de trabalho. Os cargos dos funcionários responsáveis pela limpeza são hierárquicos. Assim, deseja-se armazenar também, para cada funcionário de limpeza, informações sobre o funcionário de limpeza que o gerencia. Os funcionários da companhia são identificados por meio de um código de identificação, e podem estar associados a apenas um

único departamento. Funcionários que são pesquisadores podem trabalhar em diversos projetos, independentemente desses projetos estarem sendo desenvolvidos no mesmo departamento no qual o empregado está associado. Deve-se armazenar o número de horas semanais trabalhadas por cada pesquisador em cada projeto no qual ele trabalha. Deve-se armazenar também informações sobre os dependentes de cada funcionário para propósitos de ajuda família. Deve-se armazenar o nome, o sexo e a data de aniversário, além do grau de parentesco com o funcionário.

14) Agência de Turismo

Deseja-se criar um BD para uma agência de turismo, contendo informações sobre recursos oferecidos pelas cidades que fazem parte da programação de turismo da agência. As informações a serem mantidas sobre cada cidade referem-se a hotéis, restaurantes e pontos turísticos.

Sobre os hotéis que a cidade possui deseja-se guardar o código, o nome, o endereço, a categoria (sem estrela, 1 estrela, 2 estrelas, ...), os tipos de quartos que os formam (por exemplo, luxo, superluxo, master, ...), o número dos quartos e o valor da diária de acordo com o tipo do quarto.

Sobre cada cidade deve-se armazenar seu nome, seu estado e a população. Além disso, quando uma nova cidade é cadastrada no banco de dados da agência, um código é a ela oferecido.

Cada restaurante da cidade possui um código que o identifica, um nome, um endereço e o tipo de sua categoria (por exemplo, luxo, simples, ...). Além disso, um restaurante pode pertencer a um hotel e um hotel somente pode ser associado a um restaurante. Diferentes pontos turísticos da cidade estão cadastrados no sistema: igrejas, casas de show e museus. A agência de turismo somente trabalha com estes três tipos de pontos turísticos. Nenhum outro é possível. Além da descrição e do endereço, igrejas devem possuir como característica a data e o estilo de construção. Já casas de show devem armazenar o horário de início do show (igual para todos os dias da semana) e o dia de fechamento (apenas um único dia na semana), além da descrição e do seu endereço. Finalmente, os museus devem armazenar o seu endereço, descrição, data de fundação e número de salas. Um museu pode ter sido fundado por vários fundadores. Para estes, deve-se armazenar o seu nome, a data de nascimento e a data da morte (se houver), a nacionalidade e a atividade profissional que desenvolvia. Além disso, um mesmo fundador pode ter fundado vários museus. Quando qualquer ponto turístico é cadastrado no sistema, ele também recebe um código que o identifica. O mesmo é válido para fundadores. Finalmente, casas de show podem possuir restaurante. Quando o cliente da agência reserva um passeio para uma casa de show, ele já sabe se esta possui restaurante e qual o preço médio da refeição, além da especialidade (comida chinesa, japonesa, brasileira, italiana, ...). Dentro de uma casa de show, apenas um único restaurante pode existir.

Faça o esquema conceitual para o banco de dados acima descrito. Defina restrições de participação total e parcial de forma apropriada.

Considerações: os atributos endereço e data não precisam ser decompostos. Eles podem ser considerados como atributos atômicos; considere hotel como apenas um único objeto físico, e não como uma cadeia de hotéis. O mesmo vale para restaurante e ponto turístico.

15) Sistema de Saúde

Construa um Diagrama Entidade-Relacionamento para um sistema de saúde ideal, considerando que:

- Hospitais são formados por um ou mais ambulatórios e cada um destes está em um único Hospital;
- Médicos clinicam em um único Hospital, cada um deles agregando vários Médicos;
- Hospitais solicitam exames clínicos em vários Laboratórios, cada um destes pode ter solicitações de vários Hospitais;
- Pacientes consultam com vários Médicos, e estes são consultados por vários Pacientes;
- Hospitais possuem ambulatórios, onde são atendidos vários Pacientes, enquanto estes só podem ser atendidos em um único Ambulatório;
- O pessoal de apoio do hospital é alocado em cada Ambulatório, e cada um destes conta com vários integrantes do Pessoal de apoio;
- Os Pacientes podem realizar vários Exames, e cada Exame é realizado por um único Paciente;
- Os Exames são realizados em Laboratórios, que por sua vez podem realizar quantos exames forem necessários;
- Cada Paciente pode receber vários Diagnósticos, e cada Diagnóstico pertence a um único Paciente;

16) Aeroporto

O aeroporto da cidade de Bem Longe, resolveu organizar a suas informações num sistema de banco de dados. Para tal começaram por organizar a informação sobre os aviões que "frequentam" o aeroporto. Cada avião tem um número de registo, e cada avião é de um modelo específico. O aeroporto pode acolher um certo número de modelos de aviões, e cada modelo tem um código de modelo (ex. DC-10, A320), bem como uma capacidade e um peso. Um certo número de técnicos trabalha no aeroporto. É necessário guardar o seu número de identificação, endereço, número de telefone e salário. Cada técnico é perito num ou mais modelos de aviões, e vários técnicos podem ser peritos em modelos iguais. Os controladores aéreos necessitam de ser sujeitos a um exame médico anual. Para cada controlador é necessário guardar a data do seu exame mais recente. Todos os empregados do aeroporto (incluindo os técnicos) pertencem a um sindicato. É necessário guardar o número de membro para cada empregado. Pode-se assumir que cada empregado é identificável pelo seu número de identificação. O aeroporto tem um certo número de testes que são usados periodicamente para verificar o estado dos aviões. Cada teste tem um número atribuído pela Associação Nacional de Aeroportos (ANA), bem como um nome e uma pontuação máxima. A ANA exige que o aeroporto mantenha informação sobre cada vez que um avião é sujeito a um determinado teste por um determinado técnico. Para cada teste efetuado, a informação a guardar é a sua data de efetuação, o número de horas gastas pelo técnico, e a pontuação obtida pelo avião.

Desenhe o diagrama de entidades e relacionamentos para este problema.

17) Sistema vídeo locadora

Uma locadora de vídeos possui aproximadamente 5000 DVDs. O objetivo do sistema é manter um controle das locações efetuadas pelos clientes. Cada DVD possui um código exclusivo e contém somente um filme. Para cada filme, é necessário saber seu título e sua categoria (comédia, drama, aventura, etc.). Cada filme recebe um identificador próprio. Há pelo menos um DVD de cada filme.

Os clientes frequentemente desejam encontrar os filmes estrelados pelos seus atores prediletos. Por isso, é necessário manter a informação dos atores que estrelam em cada filme. Nem todo filme possui atores (exemplo: documentários). Para cada ator os clientes às vezes desejam saber o nome real, além do nome artístico e a data de nascimento. A locadora possui aproximadamente 3000 clientes cadastrados. Somente clientes cadastrados podem alugar DVDs. Para cada cliente é necessário saber seu nome, seu telefone, seu email e seu endereço. Cada cliente recebe um número de associado. Um cliente pode alugar vários DVDs em um instante do tempo. É necessário manter os registros históricos das locações com as datas de retirada e entrega dos DVDs.

18) Sistema escola

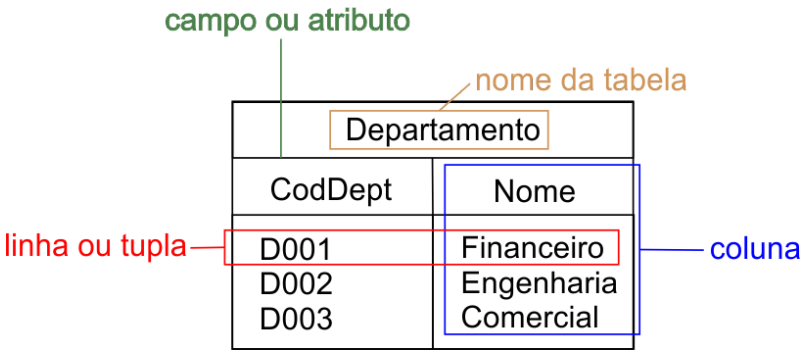
Uma escola de informática oferece vários cursos livres com duração entre trinta e sessenta dias. Cada curso recebe um código identificador. Professores são contratados para ministrar um ou mais cursos e, portanto, é necessário saber quais cursos cada professor está habilitado a ministrar. Os professores recebem um número de matrícula. A escola deseja manter também registrado o nome, endereço, telefone, email de todos os seus professores. Há várias turmas para cada curso. Cada turma, identificada por um código, tem apenas um professor e está alocada em apenas uma sala. Porém, uma sala pode ser alocada para mais de uma turma em diferentes períodos. Um aluno pode matricular-se simultaneamente em vários cursos e, portanto, pertencer a mais de uma turma. No momento da matrícula o aluno recebe um RA (válido para um ou mais cursos). A escola mantém registrado o nome, endereço, telefone, email, RG e CPF de todos os seus alunos.

5 Modelo Relacional

Apresentação da próxima etapa do projeto de banco de dados: o modelo lógico (relacional). A próxima etapa do projeto de banco de dados envolve o chamado modelo lógico. Atualmente, grande parte dos sistemas de banco de dados utiliza o modelo relacional. Um banco de dados relacional é composto por tabelas (também denominadas relações). Observe a seguir alguns conceitos importantes para pleno entendimento do modelo relacional:

5.1 Tabela

Estrutura bidimensional composta por linhas (tuplas) e campos (ou atributos).



5.2 Chaves e Índices

Chave: É uma ou mais colunas que distinguem uma linha das demais dentro de uma tabela, sendo esta chamada de chave primária ou para relacionar com outra tabela, chamada de chave estrangeira. Essas chaves é que determinam a unicidade de cada registro dentro de uma tabela.

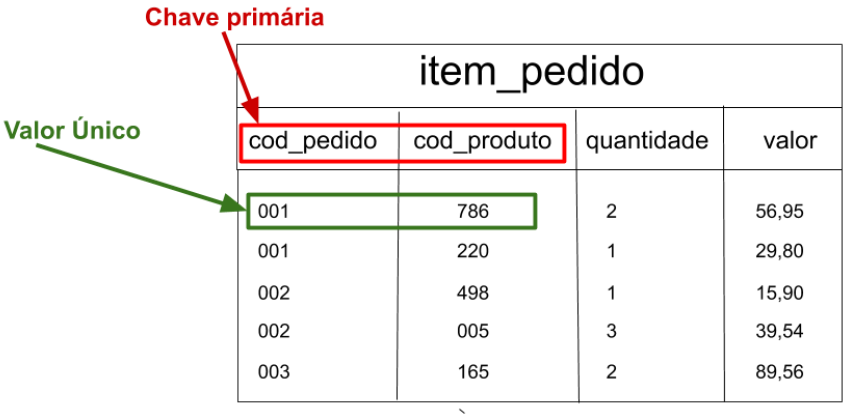
Índices: O conceito de índice está associado a um recurso físico usado para otimizar uma consulta no banco de dados. É um recurso físico, ou seja, um índice é uma estrutura de dados, (endereços), que existe fisicamente no banco de dados.

5.2.1 Chave Primária (PK – Primary Key)

Atributo através do qual seja possível identificar determinado registro. Uma chave primária não pode ser repetida, ou seja, o conjunto de valores que constituem a chave primária deve ser único dentro de uma tabela.

Chave primária simples: apenas um atributo (campo) compõe a chave primária.

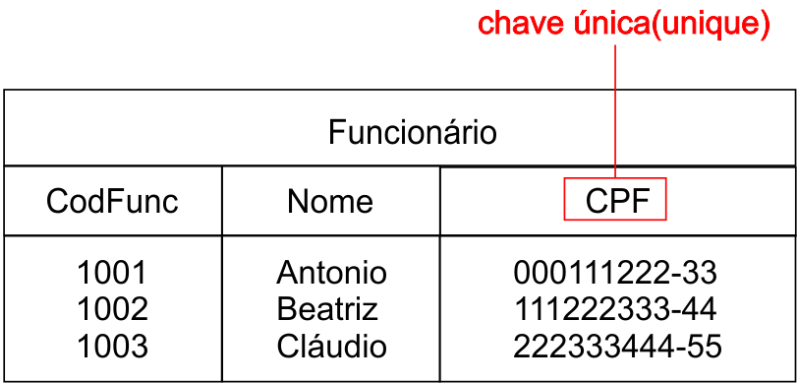
Chave primária composta: mais de um atributo compõe a chave primária.



5.2.2 Chave única (Unique)

Utilizada quando determinado campo não deve ser repetido e não é chave primária. Aumenta a consistência do banco de dados.

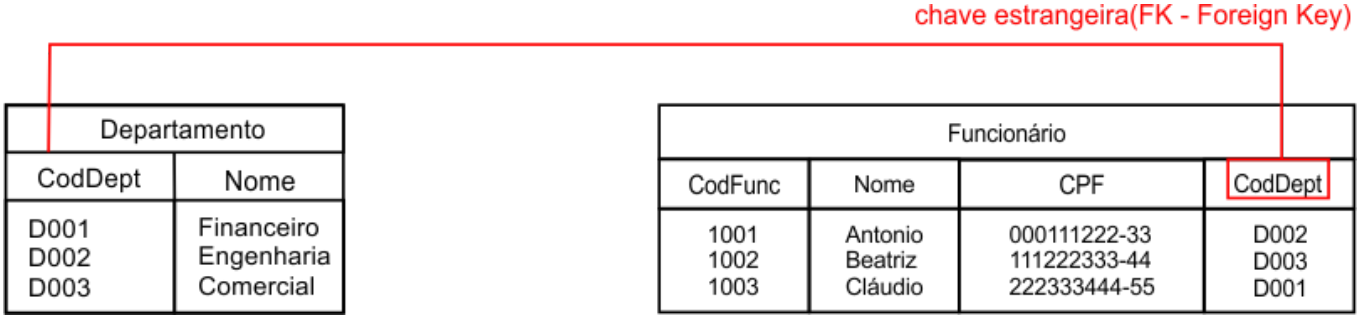
Exemplo: Cadastro de funcionários. Cada funcionário recebe um código único, que é a chave primária. Para maior segurança e consistência podemos optar que o campo CPF também seja único, evitando que o mesmo funcionário seja cadastrado duas vezes.



5.2.3 Chave Estrangeira (FK – Foreign Key)

Utilizada quando queremos que o valor de um atributo seja validado a partir par do valor de atributo de outra tabela. Criamos assim uma relação de dependência (um relacionamento) entre as tabelas.

Exemplo: Antes de efetuar a alocação de um funcionário em um departamento, departamento é necessário que o departamento em questão conste na tabela de departamentos.

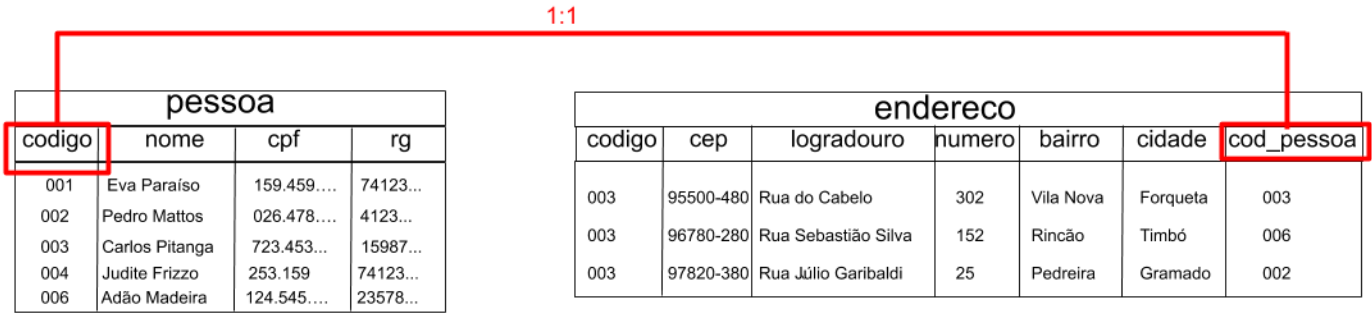


5.3 Relacionamentos

Associação estabelecida entre campos comuns de duas tabelas. Dessa forma permitimos o estabelecimento de correspondência entre registros de diferentes tabelas. Os relacionamentos apresentam a seguinte classificação quanto à sua cardinalidade:

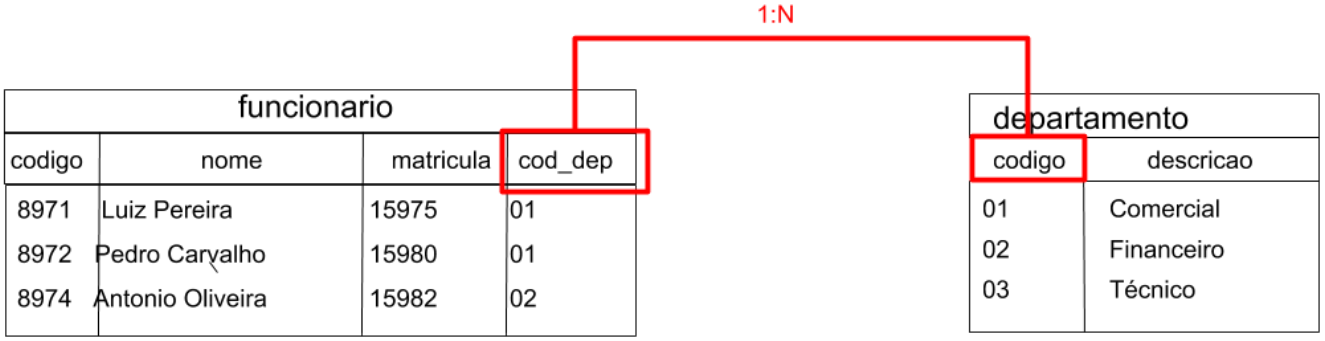
5.3.1 Relacionamento um-para-um (1:1)

Cada ada ocorrência de uma tabela relaciona-se se com uma e somente uma ocorrência da outra tabela.



5.3.2 Relacionamento um-para-muitos (1:N)

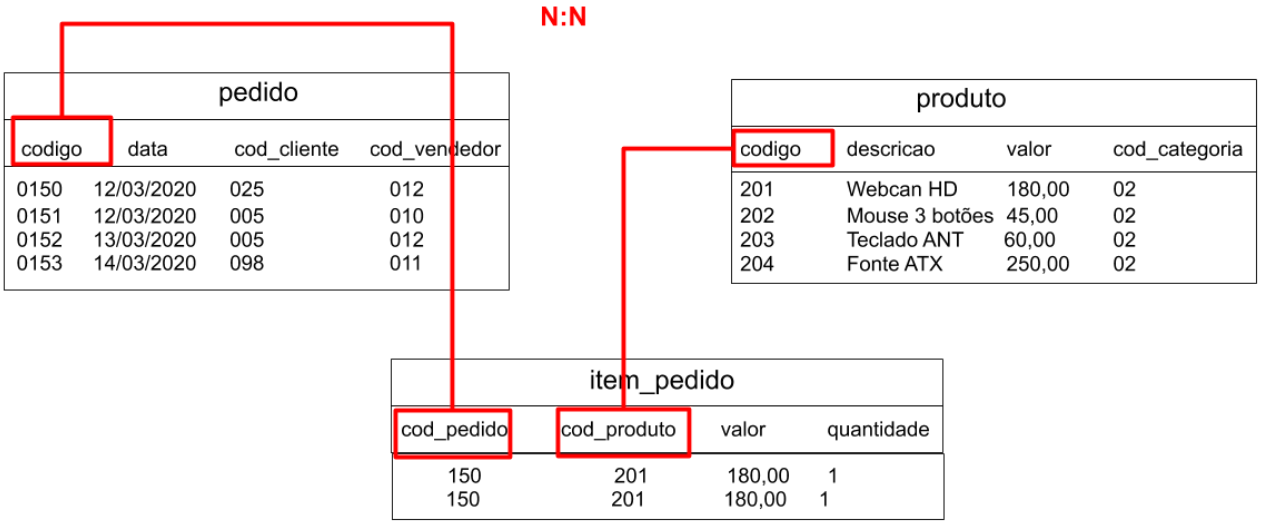
Uma ocorrência da tabela pai relaciona-se se com muitas ocorrências da tabela filho, mas cada ocorrência da tabela filho somente pode estar relacionada com uma ocorrência da tabela pai.



5.3.3 Relacionamento muitos-para-muitos (N:N)

Apresenta em ambos os sentidos um ou mais relacionamentos de um-para-muitos. No modelo relacional não é possível efetuar este tipo de relacionamento de forma direta. Neste caso, deve-se construir uma terceira tabela (tabela de associação ou tabela de detalhes). Essa tabela deve possuir chave primária composta de dois campos e as chaves estrangeiras provenientes das duas tabelas originais.

Concluindo, um relacionamento de muitos-para-muitos deve ser dividido em dois relacionamentos de um-para-muitos com uma terceira tabela.



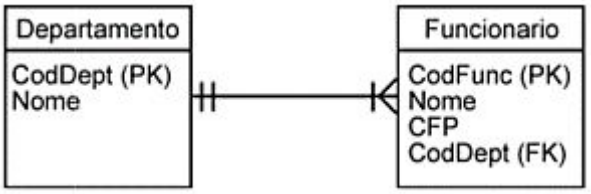
5.3.4 Notação resumida para modelos lógicos relacionais

Notação compacta, útil para discussões sobre a estrutura geral do banco de dados, utilizada quando não se deseja entrar no nível maior de detalhamento.

Exemplo:

Departamento (CodDept, Nome)
Funcionario (CodFunc, Nome, CPF, CodDept)
CodDept referencia departamento

A notação resumida acima representa o seguinte relacionamento entre as tabelas Departamento e Funcionário:



Observe que através da notação resumida não é possível determinar se o relacionamento é do tipo 1:1 ou 1:N (como no caso representado na figura acima).

5.4 Integridade de dados

Impor a integridade de dados garante a qualidade dos dados em um banco de dados. Os dados devem refletir corretamente a realidade representada pelo banco e também devem ser consistentes entre si.

5.4.1 Integridade de domínio

Zela pelos valores ideais e necessários para um atributo. Para isso definimos algumas regras de validação por meio de expressões compostas de valores constantes.

Exemplos:

- Não permitir um estoque negativo

- Impedir uma data de nascimento superior à data atual
- Não permitir que o valor de um produto seja negativo

5.4.2 Integridade de entidade

Tem o objetivo de validar os valores permitidos a partir de valores já inseridos na própria entidade. Após uma "auto-consulta" a entidade vai permitir ou não a gravação do novo registro.

Exemplos:

- Não permitir duas pessoas com o mesmo CPF
- Impedir a locação um DVD que já está locado

5.4.3 Integridade referencial

Zela pela consistência dos registros de uma entidade a partir de valores provenientes de outras entidades, isto é, determinado registro vai "depender" diretamente de um registro de outra tabela. Exemplos:

- Um registro em uma tabela pai pode ter um ou mais registros em uma tabela filho.
- Um registro em uma tabela filho sempre tem um registro coincidente em uma tabela pai.
- Para a inclusão de um registro em uma determinada tabela filho, é necessário que exista um registro pai coincidente.
- Um registro pai só poderá ser excluído se não possuir nenhum registro filho.

5.4.4 Constantes (RESTRIÇÕES)

Observe a seguir as principais constantes ou restrições utilizadas nos bancos de dados relacionais, principalmente durante o processo de criação das tabelas, para implementar os tipos de integridade anteriormente descritos:

Principais tipos de restrições

TIPO	ARMAZENA
PRIMARY KEY	Identifica a chave primária da tabela
FOREIGN KEY	Identifica a chave estrangeira
UNIQUE	Indica que os valores na coluna não podem ser repetidos
CHECK	Especifica os valores que uma coluna pode assumir
NOT NULL	Indica que o campo não pode receber valores nulos

5.5 Nomenclatura de Tabelas e de campos

Os sistemas gerenciadores de bancos de dados geralmente impõem certas restrições quanto aos caracteres válidos para denominar tabelas, campos (colunas), bem como outros objetos do banco de dados. Observe a seguir o que deve ser evitado:

- Não utilizar caracteres especiais (exceto o underscore "_");

- Começar com uma letra e não com um número;
- Evitar acentuação e “ç”;
- Não utilizar espaços.

5.6 Tipos de dados

Durante a criação das tabelas do banco de dados é necessário informar informa que tipo de dados cada coluna deverá armazenar. A tabela a seguir apresenta alguns tipos de dados compatíveis com o Oracle, um dos principais sistemas de gerenciamento de banco de dados atualmente utilizado:

Tipos de dados (Oracle)

TIPO	ARMAZENA
CHAR	Caractere
VARCHAR	Cadeia de caracteres variável
INT	Número inteiro
DATE	Data

Exercícios:

1) O que é um banco de dados relacional?

2) O que é uma tabela no contexto dos bancos de dados relacionais?

3) Explique o que você entende por cada um dos seguintes termos:

a) Chave primária

b) Chave única

c) Chave estrangeira

4) O que são relacionamentos? Como podemos classificá-los quanto à cardinalidade?

5) Que solução deve ser adotada no modelo relacional para relacionamentos com cardinalidade N:N (muitos para muitos)?

6) O que você entende por:

a) Integridade de domínio

b) Integridade de entidade

c) Integridade referencial

7) Explique o que há de errado nos casos apresentados a seguir:

a)

Aluno	
RA	Nome
111222333	Beatriz
222333444	Antonio
111222333	Cláudio

b)

Turma	
CodTurma	Nome
111222333	Beatriz
222333444	Antonio
111222333	Cláudio

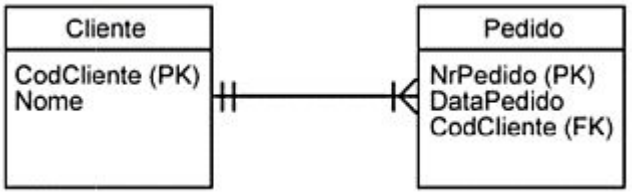
Aluno		
RA	Nome	CodTurma
111222333	Beatriz	TINF 1A1
222333444	Antonio	TINF 1D1
111222333	Cláudio	TINF 1C1

c)

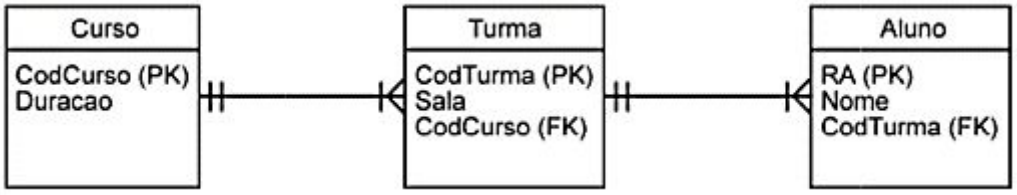
Filme	
Cod-Filme	Título
111222333	Tarzan
222333444	Avatar
111222333	Crepúsculo

8) Utilize a notação resumida para representar os casos a seguir conforme o modelo relacional:

a)



b)



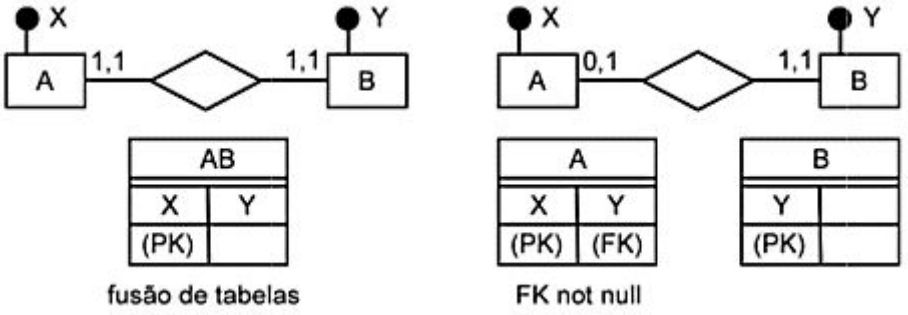
Mapeamento do Modelo Conceitual para o Lógico

6

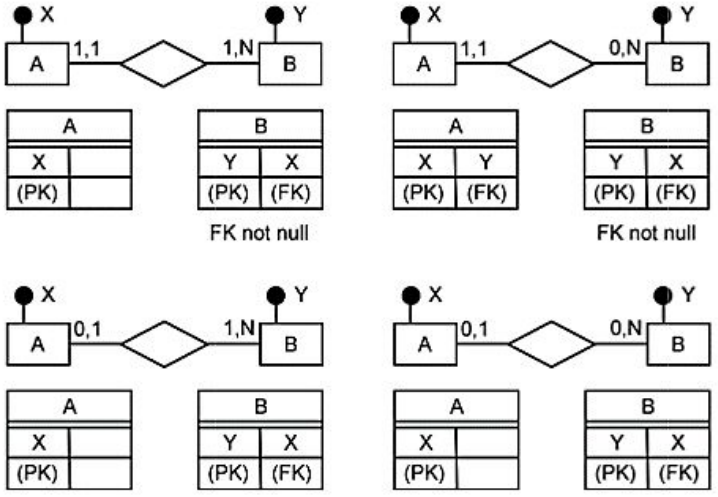
Apresentação dos mapeamentos do modelo conceitual para o lógico relacional dos diversos tipos de relacionamentos.

6.1 Relacionamentos Binários

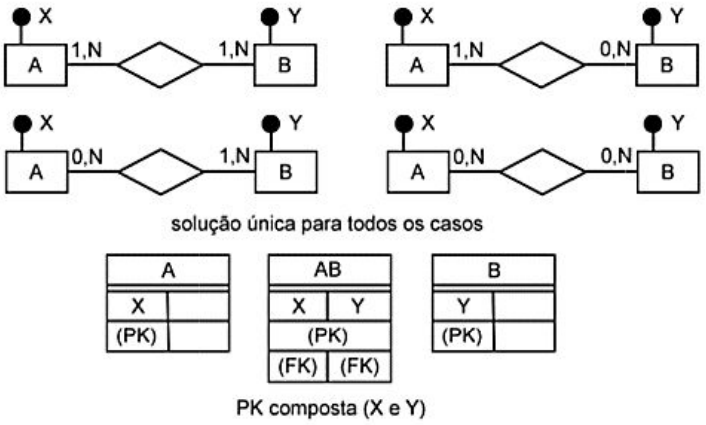
Cardinalidade máxima 1:1



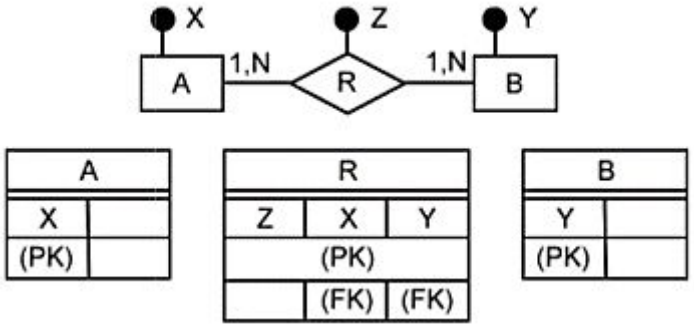
Cardinalidade máxima 1:N



Cardinalidade máxima N:N

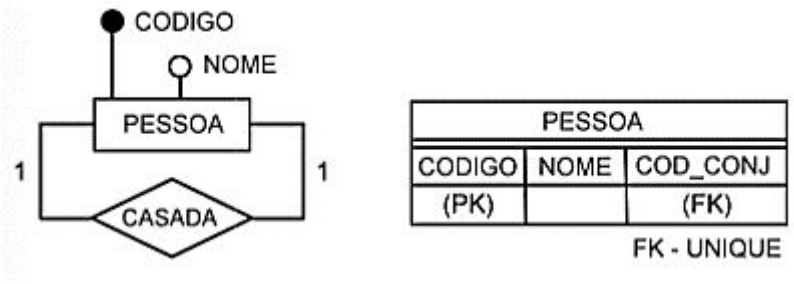


Cardinalidade máxima N:N Relacionamento com atributo identificador

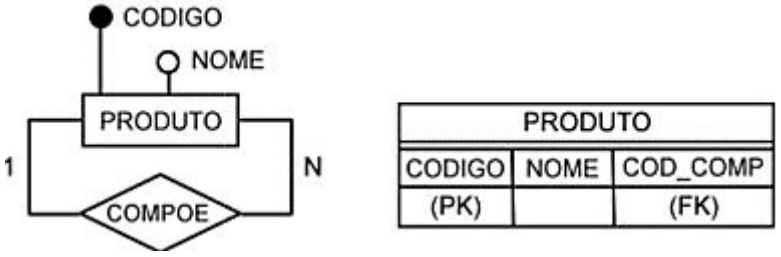


6.2 Auto Relacionamento

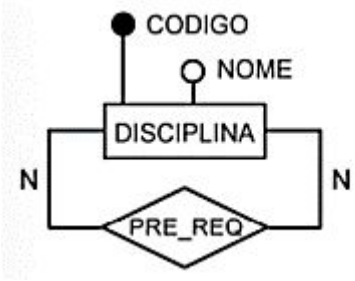
Cardinalidade máxima 1:1



Cardinalidade máxima 1:N



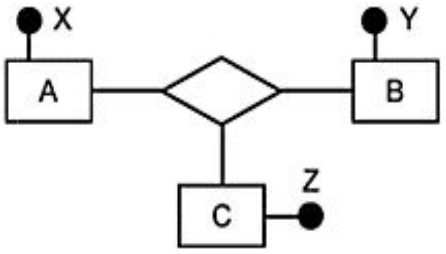
Cardinalidade máxima N:N



DISCIPLINA	
CODIGO	NOME
(PK)	

PRE_REQUISITO	
CODIGO	COD_PRE
(PK)	

6.3 Relacionamentos Ternários



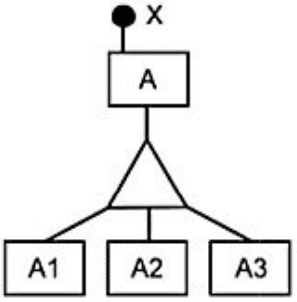
A	
X	
(PK)	

B	
Y	
(PK)	

C	
Z	
(PK)	

ABC		
Z	X	Y
(PK)		
(FK)	(FK)	(FK)

6.4 Generalização/Especialização



A	
X	colunas comuns para A1, A2 e A3
(PK)	

A1	
X	colunas exclusivas para A1
(PK)	

A2	
X	colunas exclusivas para A2
(PK)	

A3	
X	colunas exclusivas para A3
(PK)	

Exercício:

1) Faça o mapeamento do modelo conceitual para o modelo lógico (relacional) dos seguintes cases anteriormente estudados:

a) Sistema vídeo locadora

b) Sistema escola

c) Sistema empresa

7 Normalização

7.1 Conceitos preliminares

Apresentação de conceitos necessários para compreender melhor o processo de normalização de tabelas.

NORMALIZAÇÃO

- Conceito introduzido em 1970 por Edgard F. Codd;
- Processo matemático formal com fundamento na teoria dos conjuntos.

O processo de normalização aplica uma série de regras sobre as tabelas de um banco de dados para verificar se estas foram corretamente projetadas.

Os objetivos principais da normalização de tabelas são os seguintes:

- Garantir a integridade dos dados, evitando que informações sem sentido sejam inseridas;
- Organizar e dividir as tabelas da forma mais eficiente possível, diminuindo a redundância e permitindo a evolução do banco de dados.

São seis as formas normais mais utilizadas:

- 1FN – 1a Forma Normal
- 2FN – 2a Forma Normal
- 3FN – 3a Forma Normal
- FNBC – Forma Normal de Boyce e Codd
- 4FN – 4a Forma Normal
- 5FN – 5a Forma Normal

Nota: As três primeiras formas normais atendem à maioria dos casos de normalização. Uma forma normal engloba todas as anteriores, isto é, para que uma tabela esteja na 2FN, ela obrigatoriamente deve estar na 1FN e assim por diante. Normalmente após a aplicação das regras de normalização, algumas tabelas acabam sendo divididas em duas ou mais tabelas. Este processo colabora significativamente para a estabilidade do modelo de dados e reduz consideravelmente as necessidades de manutenção.

CHAVES

Chave candidata: Atributo ou conjunto de atributos que são únicos para cada registro. Para cada tabela podemos ter uma ou várias chaves desse tipo. Exemplo: codigo e cpf.

Chave primária: Entre as chaves candidatas, escolhemos uma para ser o identificador principal da tabela. Este atributo passa a ser chamado de chave primária (PK – Primary Key).

Chaves alternativas: São as chaves candidatas que não foram definidas como chave primária.

Chave estrangeira: É o atributo ou conjunto de atributos que faz a ligação com a chave primária de outra tabela.

7.1.1 Dependência Funcional (DF)

Sempre que um atributo X identifica um atributo Y, dizemos que entre eles há uma dependência funcional. Temos, portanto, que X é o determinante e que Y é o dependente.

A representação é: X → Y (lê-se X determina Y ou Y é dependente de X).

cidade → estado

No exemplo a seguir, estado é funcionalmente dependente de cidade ou ainda cidade determina estado.

CIDADE	ESTADO
Taquara	Rio Grande do Sul
Natal	Rio Grande do Norte
Niterói	Rio de Janeiro
Florianópolis	Santa Catarina

TRANSITIVIDADE

Se um atributo X determina Y e se Y determina Z, podemos dizer que X determina Z de forma transitiva, isto é, existe uma dependência funcional transitiva de X para Z.

cidade → estado

estado → pais

cidade → pais (cidade determina pais de forma transitiva)

CIDADE	ESTADO	PAIS
Taquara	Rio Grande do Sul	Brasil
Miami	Florida	EUA
Maldonado	Rocca	Uruguai

7.1.2 Dependência Funcional Irredutível à Esquerda

O lado esquerdo de uma dependência funcional é irredutível quando o determinante está em sua forma mínima, isto é, quando não é possível reduzir a quantidade de atributos determinantes sem perder a dependência funcional.

{cidade,estado} → país (não está na forma irredutível à esquerda, pois podemos ter somente o estado como determinante)

CIDADE	ESTADO	PAIS
Taquara	Rio Grande do Sul	Brasil
La Paloma	Rocca	Uruguai

estado → país (está na forma irredutível à esquerda)

ESTADO	PAIS
Rio Grande do Sul	Brasil
Rocca	Uruguai

Obs: Nem sempre estar na forma irredutível à esquerda significa possuir um determinante com apenas uma coluna.

7.1.3 Dependência Multi valorada (DMV)

A DMV é uma ampliação da Dependência Funcional (DF). Na DMV o valor de um atributo determina um conjunto de valores de um outro atributo.

É representada por X → → (X multidetermina Y ou Y é multidependente de X)

DF: {CPF} → {Nome} Temos somente um nome para cada CPF

DMV: {CPF} → {Dependente} Temos vários dependentes para cada pessoa

CPF	DEPENDENTE
538.657.931-84	Gustavo Figueroa
538.657.931-84	Betânia Almeida
538.657.931-84	Carlos Cunha

Exercícios:

Responda às seguintes perguntas:

1) O que você entende por normalização de tabelas?

2) Quais os principais objetivos da normalização de tabelas?

3) O que ocorre normalmente após a aplicação das regras de normalização?

4) Explique os seguintes conceitos:

a) Dependência Funcional

b) Transitividade

c) Dependência Funcional Irredutível à Esquerda

d) Dependência Multivalorada

7.2 Formas Normais

Veremos a aplicação das três primeiras Formas Normais em uma tabela não normalizada.

É muito comum que os funcionários dos diversos departamentos de uma empresa utilizem tabelas frequentemente geradas em planilhas eletrônicas (exemplo: Excel) para armazenamento de dados. Embora esta solução seja útil para várias situações, à medida que a quantidade de dados cresce, podem ocorrer problemas relacionados à manutenção dos dados. O problema torna-se ainda mais grave ao tentar-se passar os dados de uma planilha eletrônica para uma ou mais tabelas em um sistema de banco de dados sem observar-se algumas regras ou normas básicas. Neste processo é muito importante a aplicação de um conjunto de normas ou regras conhecidas como Formas Normais.

Uma empresa de engenharia pode, por exemplo, utilizar os seguintes formulários para controle de seus projetos:

NR_PROJ	NOME_PROJ	LOCAL_PROJ	
001	Alfa	São Paulo	
002	Beta	Jundiaí	
ID_FUNC	**NOME_FUNC**	**CARGO**	**VL_HORA**
101	Antonio	Analista Pleno	R$ 35,00
102	Beatriz	Analista Pleno	R$ 35,00
103	Claudio	Analista Senior	R$ 50,00

Observe a seguir a planilha elaborada para controle dos vários projetos da empresa:

NR_PROJ	NOME_PROJ	LOCAL_PROJ	ID_FUNC	NOME_FUNC2
			101	Antonio
001	Alfa	São Paulo	102	Beatriz
			103	Claudio
			102	Beatriz
2	Beta	Jundiaí	103	Claudio
			104	Daniela

Porém, à medida que a quantidade de projetos e funcionários alocados neles cresce, observou-se que seria necessário utilizar um sistema de banco de dados.

Para garantir a integridade e controlar a redundância dos dados aplicou-se à tabela acima as seguintes Formas Normais:

7.2.1 1FN: Primeira Forma Normal

Uma tabela está na 1FN (Primeira Forma Normal) quando não possui tabelas aninhadas. A tabela para controle de projetos apresenta a seguinte tabela aninhada:

ID_FUNC	NOME_FUNC	CARGO	VL_HORA
101	Antonio	Analista Pleno	35,00
102	Beatriz	Analista Pleno	35,00
103	Claudio	Analista Senior	50,00
102	Beatriz	Analista Pleno	35,00
103	Claudio	Analista Senior	50,00
104	Daniela	Analista Senior	50,00

Não se deve simplesmente separar a tabela acima do restante da tabela de controle de projetos, porque, neste caso, não seria mais possível determinar em quais projetos cada funcionário trabalhou. Para que isso não ocorra, é preciso incluir a coluna NR_PROJ na tabela que será denominada PROJETO_FUNCIONARIO:

PROJETO_FUNCIONARIO				
NR_PROJ	ID_FUNC	NOME_FUNC	CARGO	VL_HORA
001	101	Antonio	Analista Pleno	35,00
001	102	Beatriz	Analista Pleno	35,00
001	103	Claudio	Analista Senior	50,00
002	102	Beatriz	Analista Pleno	35,00
002	103	Claudio	Analista Senior	50,00
002	104	Daniela	Analista Senior	50,00

Consequentemente, a segunda tabela apresentará a seguinte estrutura:

PROJETO		
NR_PROJ	NOME_PROJ	LOCAL_PROJ
001	Alfa	São Paulo
002	Beta	Jundiaí

7.2.2 2FN: Segunda Forma Normal

Uma tabela está na 2FN (Segunda Forma Normal) quando, além de estar na Primeira Forma Normal, não contém dependências parciais.

Uma dependência funcional parcial ocorre quando uma coluna depende apenas de uma parte da Chave Primária COMPOSTA (Dependência Funcional Irredutível à Esquerda).

Portanto, toda tabela que está na Primeira Forma Normal e que possui Chave Primária SIMPLES (formada por uma coluna) já está na Segunda Forma Normal.

Analisando a tabela PROJETO_FUNCIONARIO nota-se que as colunas (ou atributos) NOME_FUNC, CARGO e VL_HORA dependem apenas de uma parte da Chave Primária, ou seja, do ID_FUNC. Portanto ao aplicarmos a 2FN (Segunda Forma Normal) teremos:

FUNCIONARIO			
ID_FUNC	NOME_FUNC	CARGO	VL_HORA
101	Antonio	Analista Pleno	35,00
102	Beatriz	Analista Pleno	35,00
103	Claudio	Analista Senior	50,00
104	Daniela	Analista Senior	50,00

A tabela PROJETO_FUNCIONARIO apresentará, portanto, a seguinte estrutura após a aplicação da Segunda Forma Normal:

PROJETO_FUNCIONARIO	
NR_PROJ	ID_FUNC
001	101
001	102
001	103
002	102
002	103
002	104

7.2.3.3FN: TERCEIRA FORMA NORMAL

Uma tabela está na 3FN (Terceira Forma Normal) quando, além de estar na 2FN (Segunda Forma Normal), não contém dependências transitivas.

Uma dependência funcional transitiva ocorre quando uma coluna, além de depender da Chave Primária da tabela, depende também de outra(s) coluna(s) da tabela. (Veja o tópico da aula anterior: Dependência Funcional Transitiva.)

A tabela FUNCIONARIO apresenta uma dependência funcional transitiva. Observe que o VL_HORA não depende diretamente do ID_FUNC. VL_HORA depende diretamente do CARGO. Portanto ao aplicar-se a 3FN (Terceira Forma Normal) teremos uma tabela que pode ser denominada CARGO_SALARIO com a seguinte estrutura:

CARGO_SALARIO	
CARGO	VL_HORA
Analista Pleno	35,00
Analista Senior	50,00

A tabela FUNCIONARIO após a aplicação da Terceira Forma Normal apresentará a estrutura a seguir:

FUNCIONARIO		
ID_FUNC	NOME_FUNC	CARGO
101	Antonio Alves	Analista Pleno
102	Beatriz Bernardes	Analista Pleno
103	Claudio Cardoso	Analista Senior
104	Daniela Dantas	Analista Senior

Observe a seguir quais foram as tabelas geradas após a aplicação das três primeiras Formas Normais (FN1, FN2 e FN3) e compare com a tabela controle de projeto anteriormente apresentada.

PROJETO		
NR_PROJ	NOME_PROJ	LOCAL_PROJ
001	Alfa	São Paulo
002	Beta	Jundiaí

FUNCIONARIO		
ID_FUNC	NOME_FUNC	CARGO
101	Antonio	Analista Pleno
102	Beatriz	Analista Pleno
103	Claudio	Analista Senior
104	Daniela	Analista Senior

PROJETO_FUNCIONARIO	
NR_PROJ	ID_FUNC
001	101
001	102
001	103
002	102
002	103
002	104

CARGO_SALARIO	
CARGO	VL_HORA
Analista Pleno	35,00
Analista Senior	50,00

IMPORTANTE: O exemplo apresentado tem objetivo exclusivamente didático para esclarecimento dos conceitos envolvidos na aplicação de cada uma das três primeiras Formas Normais. Outros detalhes deveriam ser levados em consideração para o desenvolvimento de um sistema completo. Exemplo: armazenar os valores históricos dos salários, quantidade de horas de cada funcionário nos respectivos projetos, etc.

Exercício:

1) Explique quando uma tabela está em conformidade com cada uma das seguintes Formas Normais:

a) 1FN (Primeira Forma Normal)

b) 2FN (Segunda Forma Normal)

c) 3 FN (Terceira Forma Normal)

2) Aplique as três primeiras Formas Normais à tabela PEDIDOS:

PEDIDOS							
NR_PEDIDO	DATA_PEDIDO	ID_CLIENTE	NOME_CLIENTE	COD_PROD	NOME_PROD	QUANT	VL_UNIT
001	10/01/2011	1003	Ernesto	P-31	Caderno	2	15,00
				P-42	Caneta	1	3,00
				P-67	Lápis	5	1,00
002	11/01/2011	1007	Fabiana	P-42	Caneta	2	3,00
				P-67	Lápis	3	1,00
				P-85	Lapiseira	1	5,00

3) Aplique as três primeiras Formas Normais à tabela de DEPARTAMENTOS:

DEPARTAMENTOS						
COD_DEPT	LOCAL	ID_GERENTE	NOME_GERENTE	TIPO_FONE	COD_AREA	NR_FONE
1011	São Paulo	35215	Geraldo	Residencial	12	5555-1234
				Comercial	11	5555-4321
				Celular	11	5555-9876
1021	Rio de Janeiro	47360	Horacia	Residencial	21	5555-5678
				Comercial	22	5555-3659
				Celular	21	5555-2345

4) Aplique as três primeiras Formas Normais à tabela CURSOS:

CURSOS							
COD_CURSO	NOME_CURSO	COD_TURMA	NR_SALA	COD_DISC	NOME_DISC	ID_PROF	NOME_PROF
1005	TADS	1005_3A3	230	3523	Algoritmos	105	Ildemar
				5282	Banco de Dados	118	Joselia
				8346	Empreendedorismo	126	Kleudir
		1005_3B3	231	3523	Algoritmos	133	Lucimar
				5282	Banco de Dados	118	Joselia
				8346	Empreendedorismo	126	Kleudir
1250	FEGAIRC	1250_4A1	380	4639	Cálculo	133	Lucimar
				6395	Lógica Digital	142	Marcelo
				9578	Redes de Dados	158	Nilmara
		1250_4B1	381	4639	Cálculo	133	Lucimar
				6395	Lógica Digital	165	Osvaldo
				9578	Redes de Dados	158	Nilmara

Álgebra Relacional

8

Apresentação das operações da álgebra relacional: seleção, projeção, produto cartesiano, diferença, união, intersecção, junção e divisão.

- Desenvolvida para descrever operações sobre uma base de dados relacional;
- Cada operador toma uma ou duas relações como sua entrada e produz uma nova relação como sua saída;
- Linguagem da consulta teórica, usuários não a utilizam diretamente;
- É usada internamente em todos os SGBDRs (Sistemas Gerenciadores de Bancos de Dados Relacionais).

8.1 Características

Constituída de cinco operadores fundamentais:

- Seleção σ (sigma)
- Projeção π (pi)
- Produto cartesiano X
- Diferença –
- União ∪

Três operadores derivados:

- Intersecção ∩
- Junção
- Divisão :

SELEÇÃO

R

A	B
a1	b1
a2	b2

$\sigma_{(A='a1')}(R)$

A	B
a1	b1

Produz uma nova relação apenas com as tuplas (linhas) da primeira relação (tabela) que satisfazem a uma determinada condição (também chamada de predicado).

8.2 Projeção

Produz uma nova relação com apenas alguns atributos da primeira relação, removendo as tuplas duplicadas.

R

A	B
a1	b1
a2	b2

$\pi_{(B)}(R)$

B
b1
b2

8.3 Produto Cartesiano

Produz uma nova relação com todas as possíveis tuplas resultantes da combinação de duas tuplas, uma de cada relação envolvida na operação.

R

A	B
a1	b1
a2	b2

S

C	D
c2	d2
c3	d3

(R X S)

A	B	C	D
a1	b1	c2	d2
a1	b1	c3	d3
a2	b2	c2	d2
a2	b2	c3	d3

8.4 Diferença

Produz uma nova relação com todas as tuplas da primeira relação que não aparecem na segunda relação. As duas relações devem ter o mesmo número de atributos (colunas) e mesmos domínios para as colunas correspondentes.

R

A	B
a1	b1
a2	b2

S

A	B
a2	b2
a3	b3

(R - S)

A	B
a1	b1

8.5 União

Produz uma nova relação composta por todas as tuplas da primeira relação seguidas por todas as tuplas da segunda relação. Tuplas comuns às duas relações aparecerão apenas uma vez no resultado. As duas relações devem ter o mesmo número de atributos (colunas) e mesmos domínios para as colunas correspondentes.

R

A	B
a1	b1
a2	b2

S

A	B
a2	b2
a3	b3

(R ∪ S)

A	B
a1	b1
a2	b2
a3	b3

8.6 Intersecção

Produz uma nova relação com a intersecção das tuplas da primeira relação com as tuplas da segunda, ou seja, apenas com as tuplas que aparecem nas duas relações.

As duas relações devem ter o mesmo número de atributos e mesmos domínios para as colunas correspondentes.

R

A	B
a1	b1
a2	b2

S

A	B
a2	b2
a3	b3

R ∩ S

A	B
a2	b2

8.7 Junção

Produz uma nova relação com as tuplas resultantes da combinação de duas tuplas, uma de cada relação envolvida na operação que satisfazem a uma determinada condição.

R

A	B
a1	b1
a2	b2

S

C	D
b2	d2
b1	d3

R X S [B = C]

A	B	C	D
a1	b1	b1	d3
a2	b2	b2	d2

8.7.1 Junção Natural

Junção na qual há uma igualdade predefinida entre os atributos de mesmo nome presentes na primeira e na segunda relação (atributos de junção). Estes atributos só aparecem uma vez no resultado.

R

A	B
a1	b1
a2	b2

S

C	D
b2	d2
b1	d3

R * S

A	B	D
a1	b1	d3
a2	b2	d2

8.8 Divisão

Produz uma relação S contendo todas as tuplas de A (dividendo) que aparecem em R (mediador) com todas as tuplas de B (divisor).

A

A
a1
a2
a3
a4
a5

R

A	B
a1	b1
a1	b2
a1	b3
a1	b4
a2	b1
a2	b2
a3	b2
a4	b2
a4	b4

B

B
b1

S

A
a1
a2

B

B
b2
b4

S

A
a1
a4

B

B
b1
b2
b3
b4

S

A
a1

8.9 Operadores

Os seguintes operadores são utilizados na álgebra relacional:

OPERADORES DE COMPARAÇÃO	
OPERADOR	**DESCRIÇÃO**
=	igual
<	menor
<=	menor ou igual
>	maior
>=	maior ou igual
<> ou ≠	diferente

OPERADORES LÓGICOS	
OPERADOR	**DESCRIÇÃO**
∧	e (and)
∨	ou (or)
¬	não (not)

OPERADOR DE ATRIBUIÇÃO	
OPERADOR	**DESCRIÇÃO**
←	"recebe"

O operador de atribuição armazena o resultado de uma expressão algébrica em uma variável de relação. Permite, portanto, o processamento de uma consulta por etapas.

Exercício:

1) Associe as operações da álgebra relacional com as suas respectivas descrições:

	OPERAÇÃO
1	Seleção
2	Projeção
3	Produto cartesiano
4	Diferença
5	União
6	Intersecção
7	Junção
8	Junção natural
9	Divisão

	DESCRIÇÃO
	Produz uma relação S contendo todas as tuplas de A (dividendo) que aparecem em R (mediador) com todas as tuplas de B (divisor).
	Produz uma nova relação com as tuplas resultantes da combinação de duas tuplas, uma de cada relação envolvida na operação que satisfazem a uma determinada condição.
	Produz uma nova relação composta por todas as tuplas da primeira relação seguidas por todas as tuplas da segunda relação. Tuplas comuns às duas relações aparecerão apenas uma vez no resultado.
	Produz uma nova relação com todas as possíveis tuplas resultantes da combinação de duas tuplas, uma de cada relação envolvida na operação.
	Produz uma nova relação apenas com as tuplas (linhas) da primeira relação que satisfazem a uma determinada condição.
	Produz uma nova relação com apenas alguns atributos da primeira relação, removendo as tuplas duplicadas.
	Produz uma nova relação com todas as tuplas da primeira relação que não aparecem na segunda relação.
	Produz uma nova relação com a intersecção das tuplas da primeira relação com as tuplas da segunda, ou seja, apenas com as tuplas que aparecem nas duas relações.
	Junção na qual há uma igualdade predefinida entre os atributos de mesmo nome presentes na primeira e na segunda relação. Estes atributos só aparecem uma vez no resultado.

SGBD MYSQL

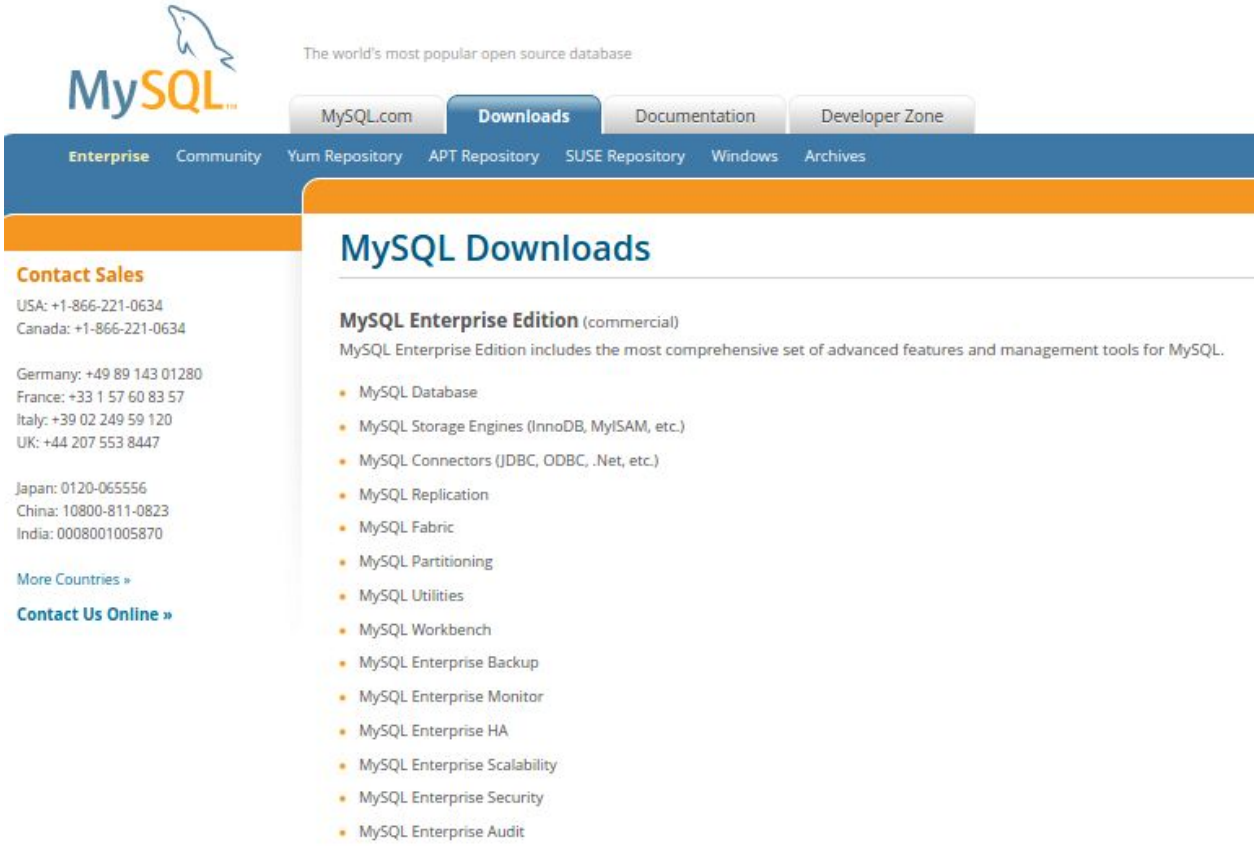
9

O **MYSQL** é um sistema de gerenciamento de banco de dados (**SGBD**), que utiliza a linguagem SQL (Linguagem de Consulta Estruturada, do inglês Structured Query Language) como interface. É atualmente um dos bancos de dados mais populares, com mais de 10 milhões de instalações pelo mundo.

9.1 Instalação

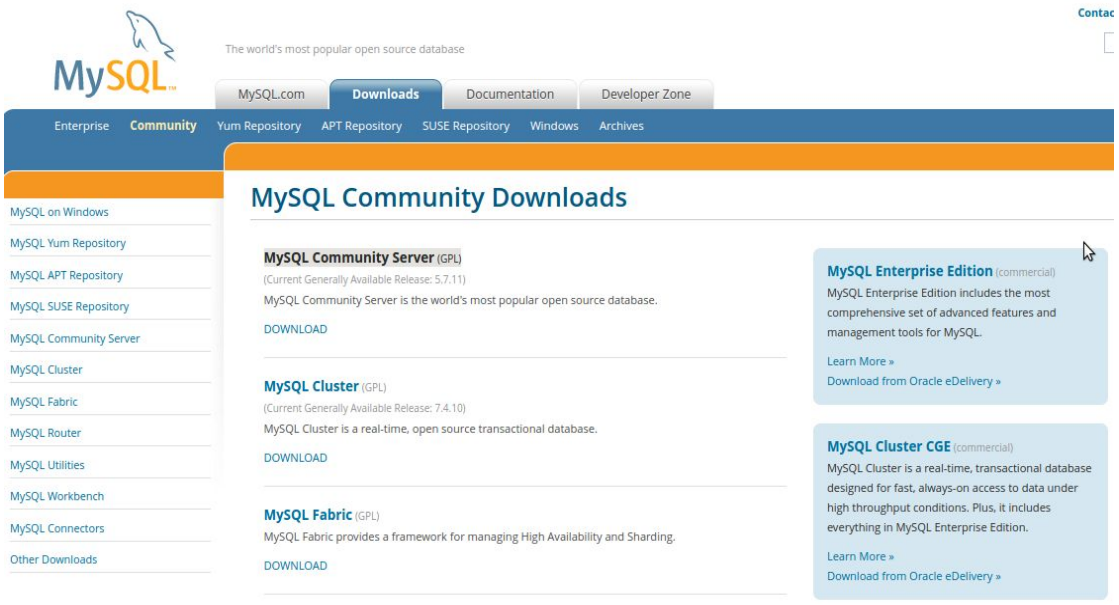
Aqui será apresentado o processo de instalação do MySQL, um dos bancos de dados gratuitos mais utilizados da atualidade.

Primeiramente deve ser acessado o site oficial do MySQL, através do seguinte link: http://www.mysql.com/downloads/

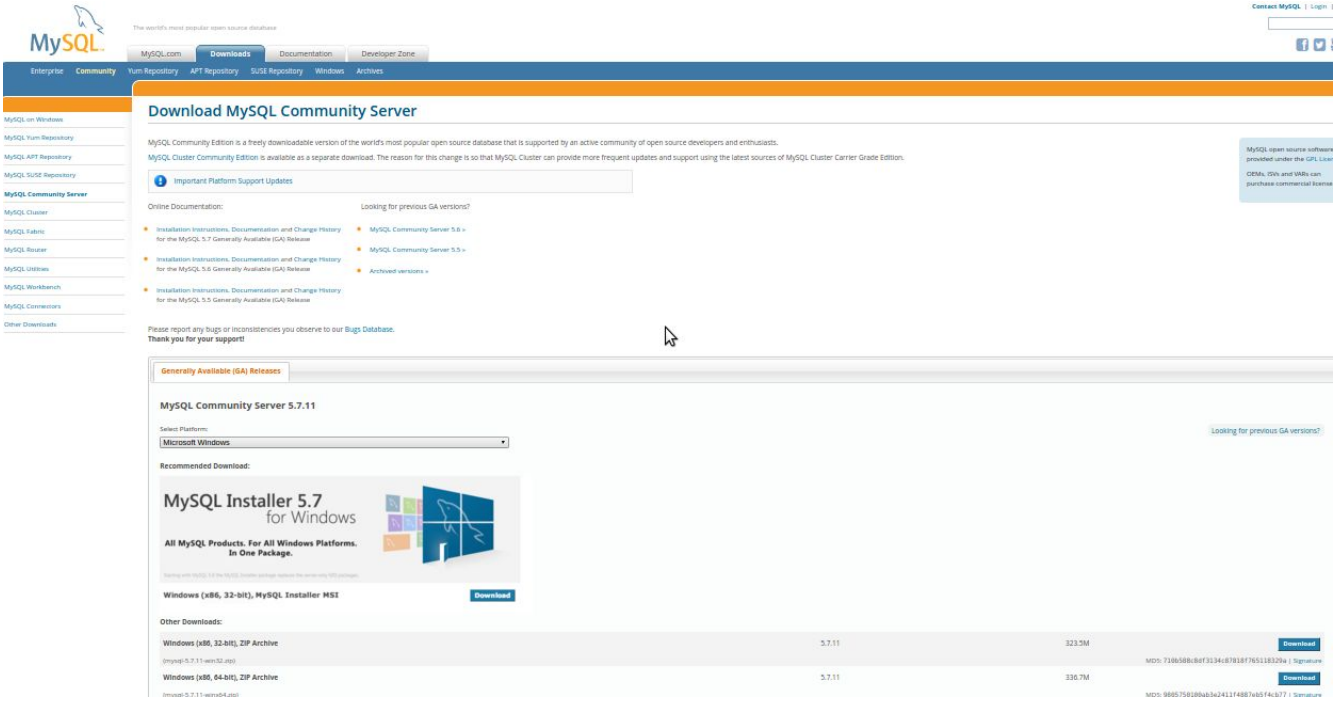


Acessar o link: Community (GPL) Downloads »

Acessar o link: **MySQL Community Server** (GPL)

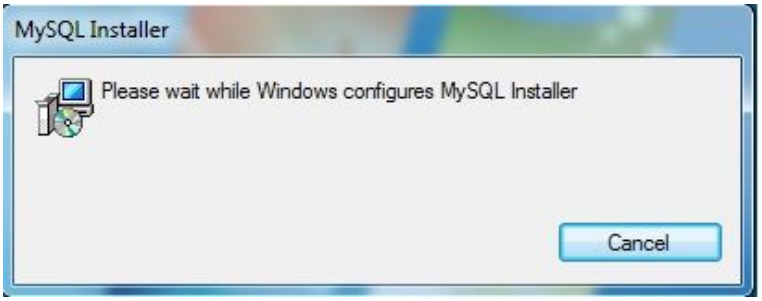


Na tela seguinte seleciona-se o sistema operacional desejado. E posteriormente aciona-se o link equivalente a arquitetura do computador em o MySQL será instado.

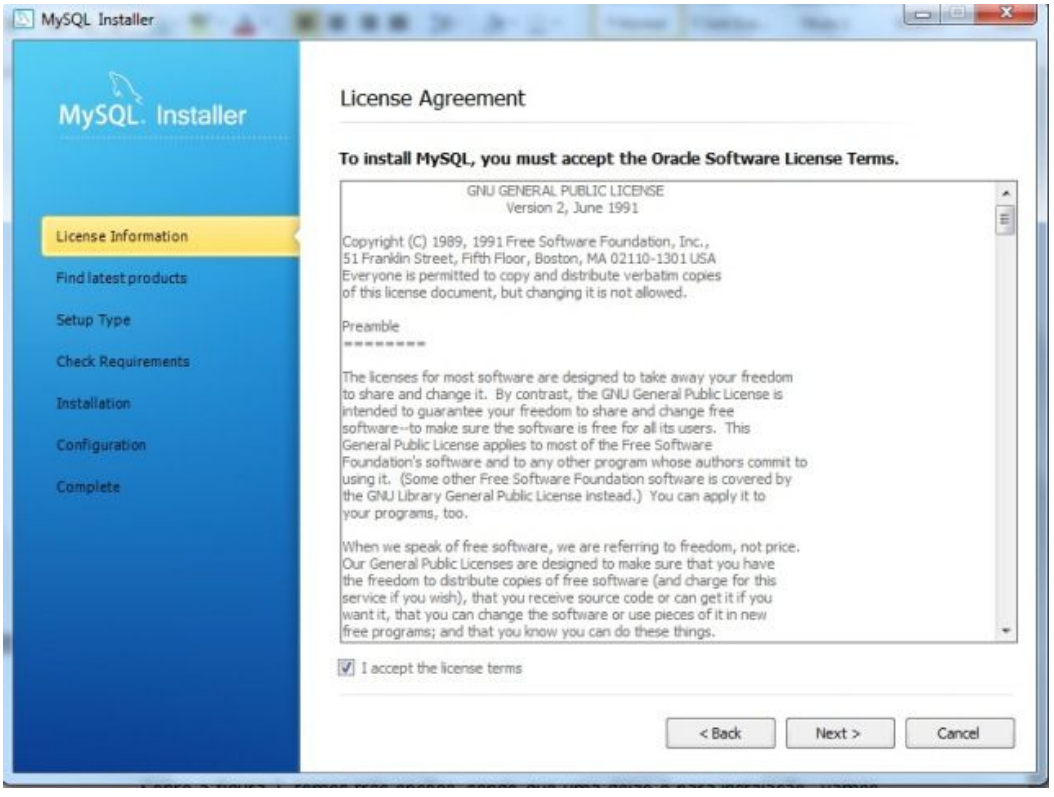


Na tela seguinte não é necessário fazer login ou realizar um novo cadastro. Basta clicar no link:

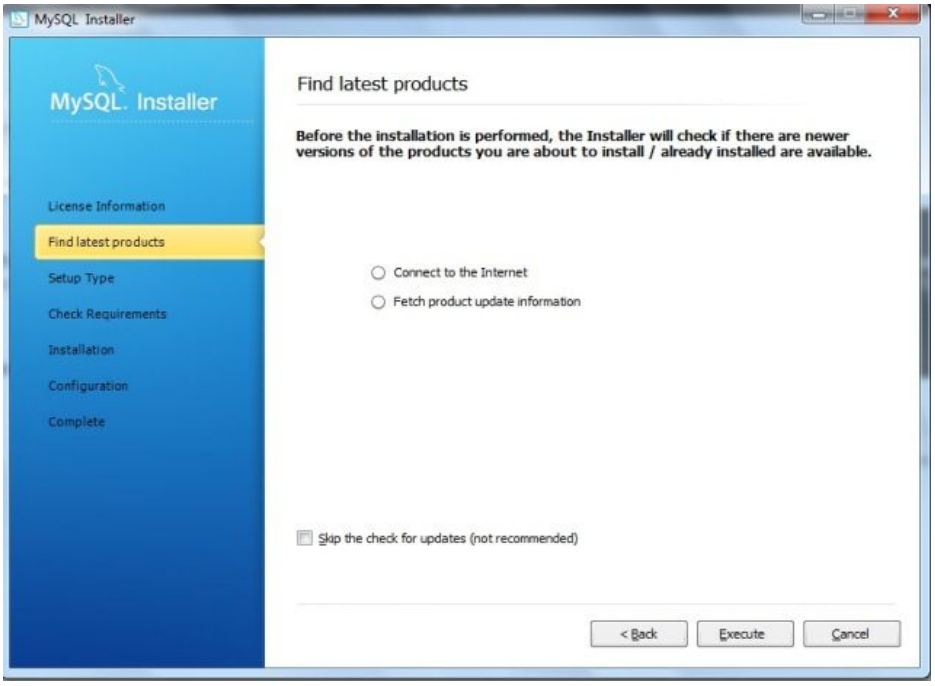
Após o download ter sido realizado. Executa-se o arquivo para inicializar a instalação.



Surgem três opções, sendo que uma delas é para instalação, para seguir adiante clica-se em INSTALL MYSQL PRODUCTS, aparecerá a tela de acordo de licença do produto, conforme apresentado a seguir.



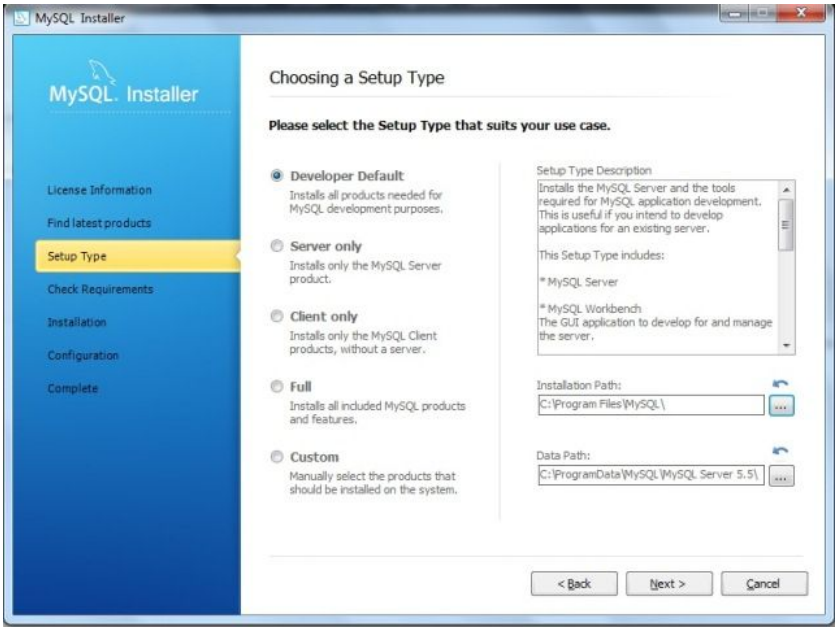
Na janela, representada na figura anterior, clica-se em I ACCEPT THE LICENSE TERMS e posteriormente em NEXT, e a janela para atualizar ou encontrar produtos recentes aparecerá, como mostra a figura a seguir:



Na figura anterior, temos as seguintes opções:

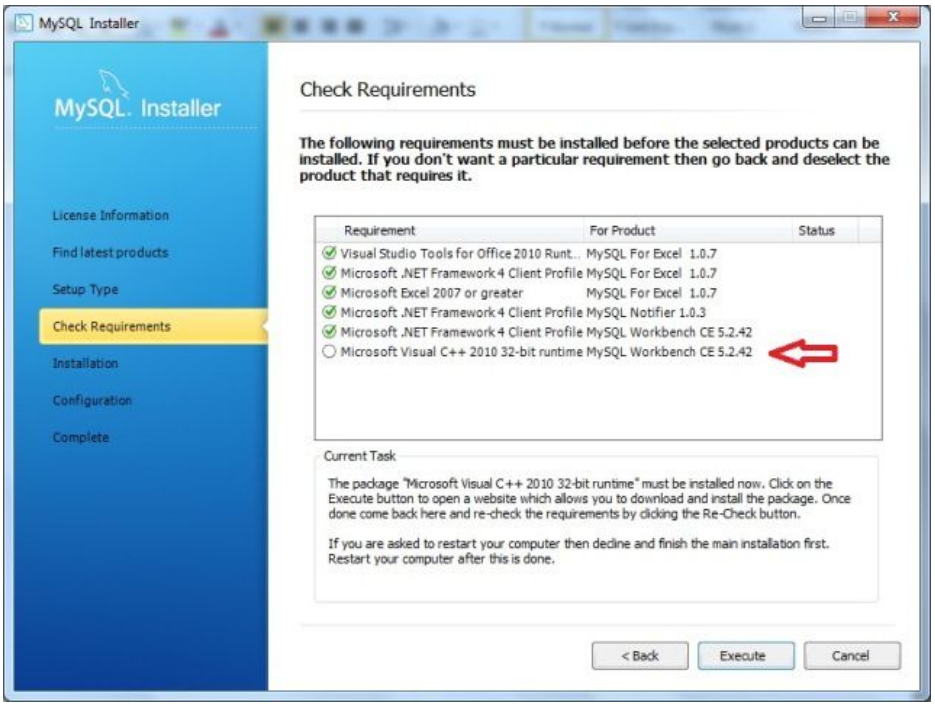
- Conectar a internet;
- Buscar informações de atualização de produto.

Estas duas opções servem para o momento da instalação, o próprio instalador do MySQL verifica se há versões mais recentes do produto, caso não ache necessária esta opção, simplesmente marque SKIP THE CHECK FOR UPDATES, ou se preferir, clique no botão EXECUTE. Após clicar no botão EXECUTE aparecerá uma nova janela, com essa deve-se ter um certo cuidado, como pode ser visto na figura a seguir.

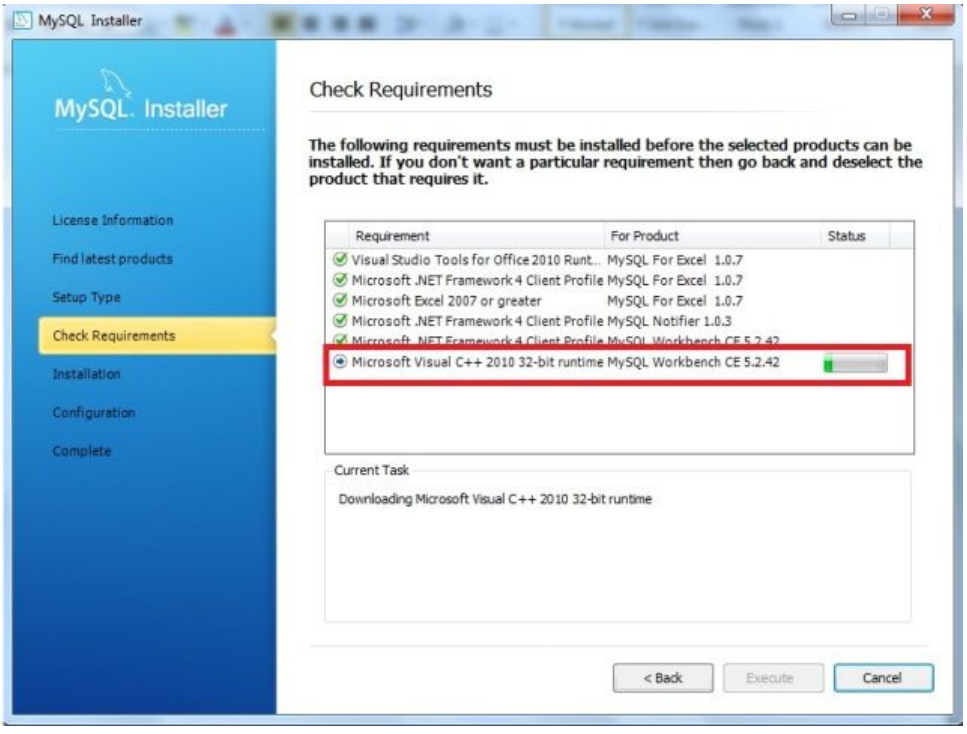


Na janela acima é apresentado várias opções, neste caso, isso vai depender de cada profissional e para que ele vai usar o MySQL. Note que no lado esquerdo o aplicativo apresenta informações bem detalhadas do que será instalado. Marca-se a opção escolhida e clica-se em

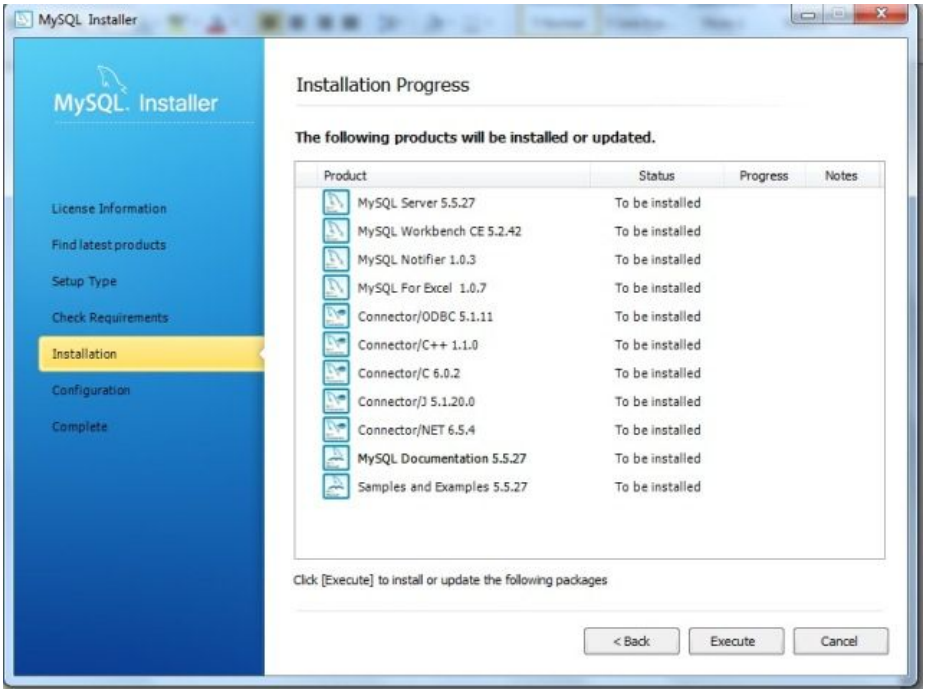
NEXT. Depois deste processo, uma nova janela surgirá, conforme a figura a seguir:



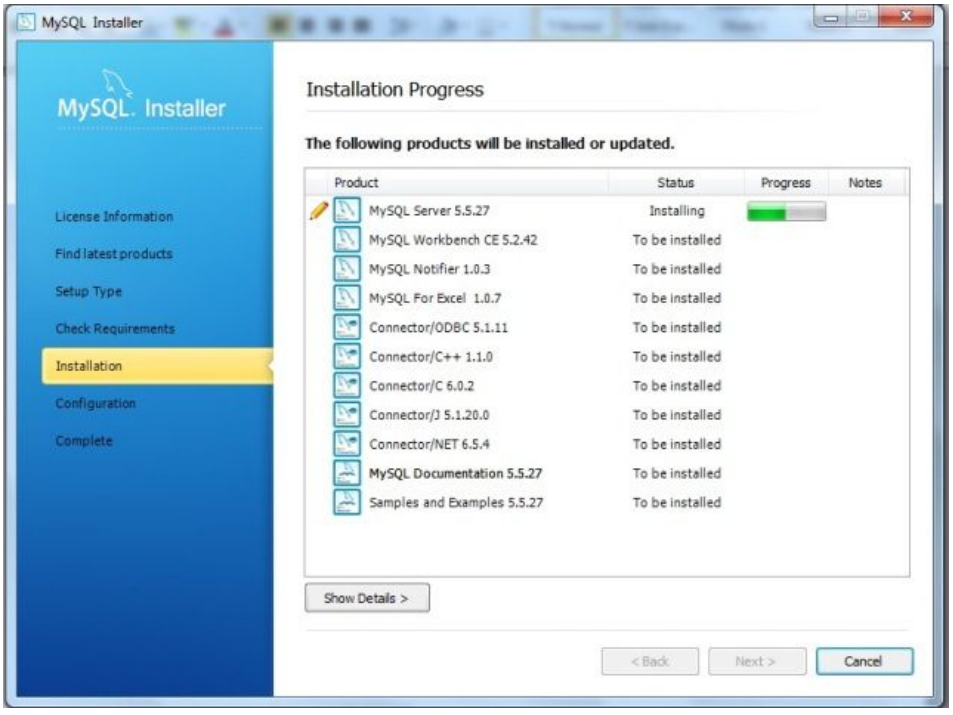
Conforme a figura anterior, existe um componente detectado que não está instalado no computador, mas não há problema, clica-se em EXECUTE que o próprio assistente vai baixá-lo, como mostra a figura a seguir.



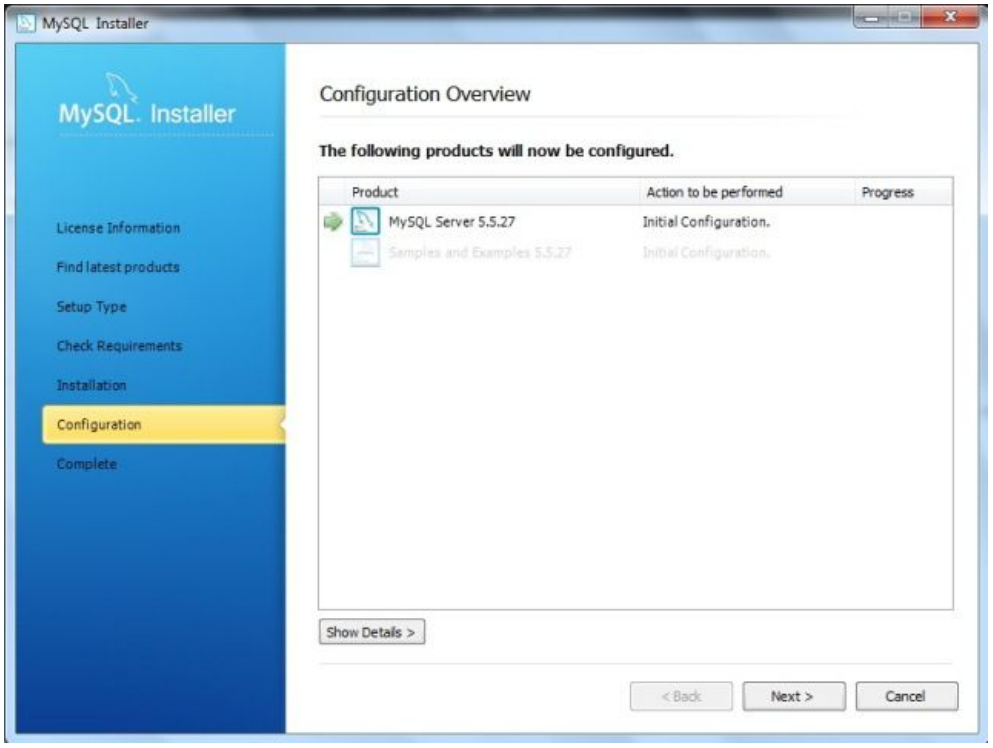
Em seguida, após baixar o aplicativo que seria necessário para dar continuidade ao processo. Deve-se dar continuidade ao processo, clicando em NEXT. A próxima janela mostrará tudo que será instalado, isso vai depender de quais opções foram escolhidas no processo mostrado na figura anterior. Na figura a seguir é apresentado o resultado.



Neste passo clica-se em EXECUTE, e conforme a seguir o processo que irá ocorrer com cada uma das opções a serem instaladas.

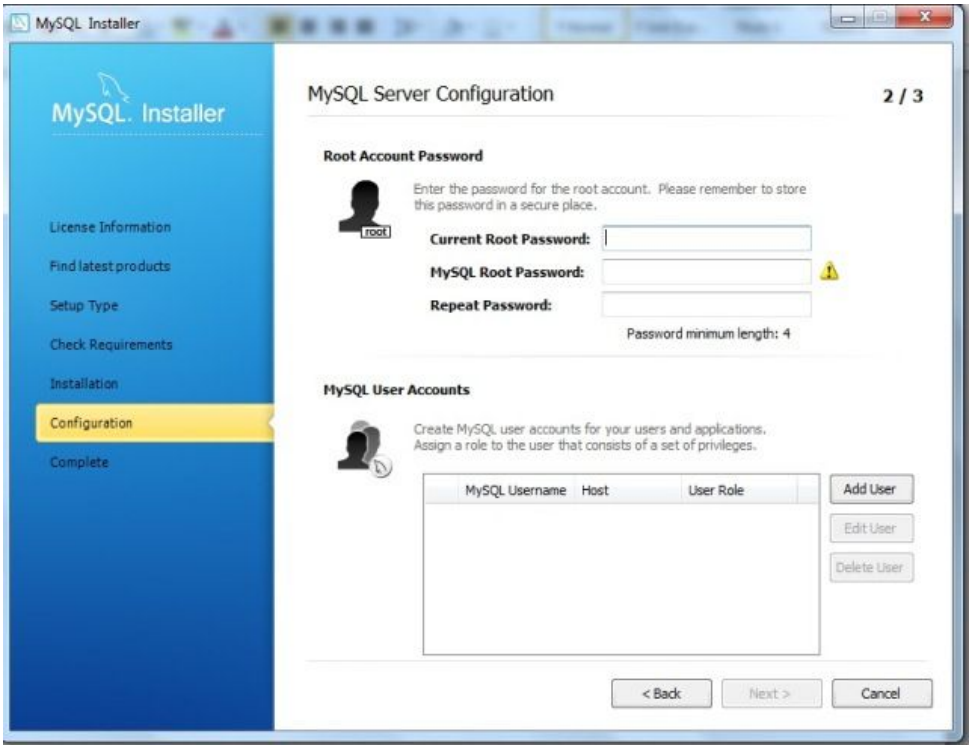


Após o término deste processo, o botão NEXT ficará habilitado, para continuar até a próxima janela CONFIGURATION OVERVIEW (visão geral da configuração), apresentada na figura a seguir.



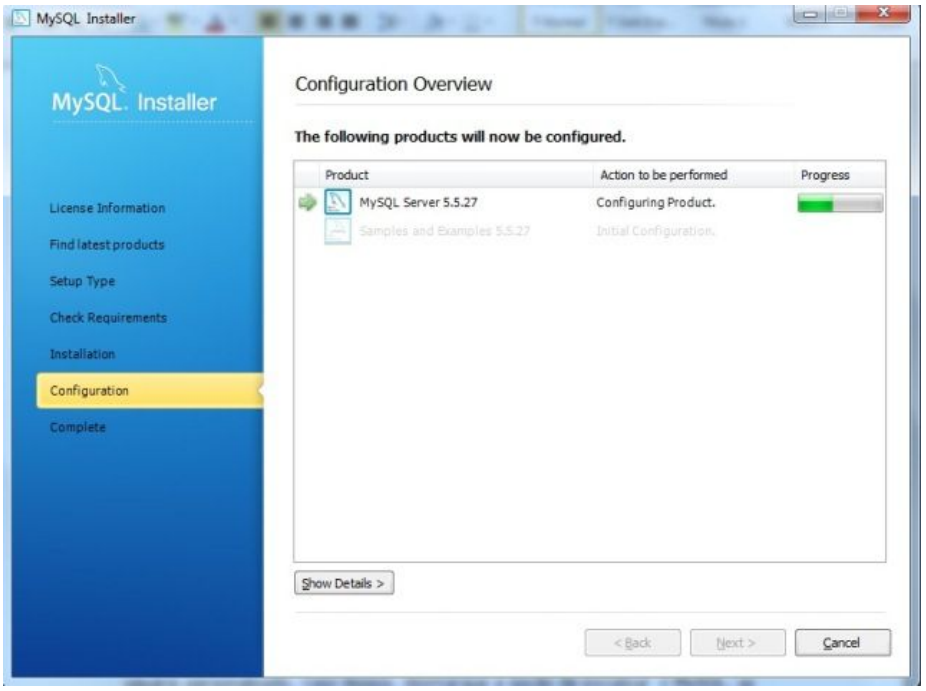
Após dar continuidade ao processo, em seguida aparecerá mais uma janela, como mostra a figura a seguir.

Na configuração do MySQL, é possível mudar a porta utilizada e o tipo de configuração do servidor: Developer Machine, Server Machine e Dedicated Machine, para este caso, vai ser deixado do jeito que está clicando em NEXT para ir ao próximo passo. Após clicar em NEXT, aparecerá uma janela para inserir uma senha para o administrador, como demonstrado na figura a seguir.

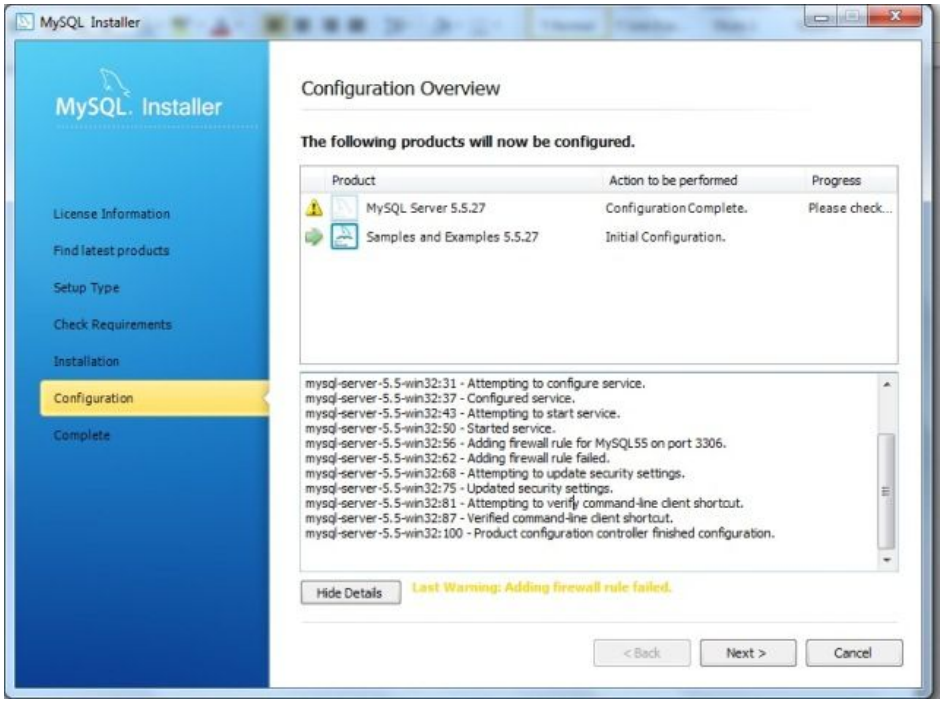


Após definir a senha do administrador clique em NEXT, novamente aparecerá uma nova janela, conforme a figura a seguir.

Na figura acima tem o WINDOWS SERVICE NAME, a opção de inicializar o MySQL com o sistema operacional e por último executar como conta do sistema padrão ou usuário personalizado. Caso deseje, desmarque a opção de inicializar o MySQL, já as outras configurações, deixe como está, clique em NEXT para o próximo passo, como mostra a figura a seguir.



Após o término, clique em NEXT para seguir para o próximo passo, continuando a configuração do servidor, como apresentada na figura a seguir.



Após este processo aparecerá uma janela informando a finalização do processo de

instalação do MySQL.

9.2 Console

Para testar a instalação do MySQL,demonstrada na seção 9.2, nada melhor do que acionar o aplicativo.

Para acessar o MySQL através do terminal de console, deve ser executado no prompt de comando do Windows a seguinte instrução:

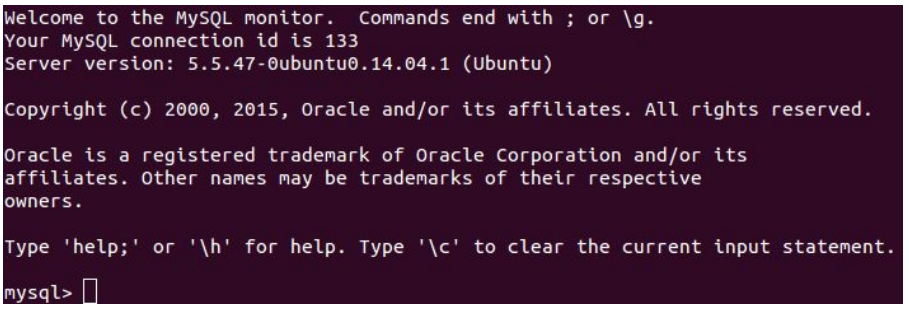
mysql -h localhost -u root -p

Entendendo o comando acima.

mysql	-- evoca o servidor
-h	-- diz que o próximo dado é referente ao host
localhost	-- informa o host
-u	-- diz que o próximo dado é referente ao usuário
root	-- informa o usuário
-p	-- diz que o próximo dado é referente a senha

Recomenda-se não digitar, porque pode ter alguém bisbilhotando. Quando o comando é executado, o console solicita a senha do usuário informado. Este é o momento certo para digitar, pois a senha não é mostrada.

A seguir é apresentado a tela referente ao console do MySQL:



10 SQL-STRUCTURED QUERY LANGUAGE

Apresentação dos principais comandos da linguagem SQL relacionando-os com a álgebra relacional.

A linguagem SQL (Structured Query Language – Linguagem de Consulta Estruturada) é uma linguagem declarativa utilizada por Sistemas Gerenciadores de Bancos de Dados Relacionais como: Oracle, SQL Server, MySQL, PostgreSQL, Firebird e outros.

Devido à sua ampla utilização por diversos SGBDs, surgiram vários “dialetos” para os comandos, que geraram a necessidade de criação de um padrão para a linguagem. Essa tarefa foi realizada pela American National Standards Institute (ANSI) em 1986 e ISO em 1987. Em 1992, foi realização uma revisão da linguagem que recebeu o nome de SQL-92.

Em 1999 e 2003, ocorreram novas revisões. Na revisão de 1999, foram adicionados padrões para expressões regulares, consultas recursivas, triggers e algumas características de orientação a objeto. Na revisão de 2003, foram introduzidas características relacionadas a XML, sequencias padronizadas e colunas com valores de auto numeração.

Uma dificuldade encontrada na utilização da linguagem SQL por parte dos desenvolvedores ou administradores de bancos de dados é a diferença entre os comandos nos diversos SGBDs, mesmo após a definição dos padrões. Porém, as diferenças não são grandes.

Os comandos da linguagem SQL são subdivididos em algumas categorias de comandos como: DDL, DML e DCL.

- **DDL** (Data Definition Language – Linguagem de Definição de Dados) - Os comandos DDL são usados para definir a estrutura do banco de dados, organizando em tabelas que são compostas por campos (colunas). Comandos que compõem a DDL: CREATE, ALTER, DROP.
- **DML** (Data Manipulation Language – Linguagem de Manipulação de Dados) - Os comandos DML permitem realizar operações de inserção, alteração, exclusão e seleção sobre os registros (linhas) das tabelas. Comandos que compõem a DML: INSERT, UPDATE, DELETE e SELECT. Alguns autores definem que o comando SELECT faz parte de uma subdivisão chamada DQL (Data Query Language – Linguagem de Consulta de Dados).
- **DCL** (Data Control Language – Linguagem de Controle de Dados) - Os comandos DCL são usados para gerenciar usuários e permissões de acesso ao Sistema Gerenciador de Banco de Dados. Comandos que compõem a DCL: GRANT e REVOKE.

Alguns autores ainda definem uma subdivisão da linguagem SQL chamada DTL (Data Transaction Languagem – Linguagem de Transação de Dados). Uma transação pode ser compreendida como um conjunto de comandos que é executado de forma atômica, ou seja, ou todos os comandos são executados com sucesso ou nenhum dos resultados obtidos por eles será mantido no banco de dados.

Para exemplificar na prática os exemplos das próximas seções será utilizado o console do MySQL.

10.1 DDL no SGBD MySQL

10.1.1 Tipos de Dados

Nas linguagens de programação, há quatro tipos primitivos de dados que são: inteiro, real, literal (texto) e lógico. Em cada SGBD, existem tipos de dados derivados desses tipos primitivos. No SGBD MySQL, é possível destacar os seguintes:

Tipo de Dado	Descrição
INTEGER	Representa um número inteiro
VARCHAR	Texto de tamanho variável. Máximo de 255 caracteres.
CHAR	Texto de tamanho fixo (preenche com espaços em branco os caracteres não preenchidos). Máximo de 255 caracteres.
DATE	Data
DATETIME	Data/Hora
TEXT	Texto de tamanho variável. Máximo de 65535 caracteres.
DECIMAL(p, d)	Número real, sendo que p define a precisão e d define o número de dígitos após o ponto decimal.

No MySQL, os campos ainda possuem atributos que definem a validação dos valores, tais como:

Atributo	Descrição	Aplica-se a
UNSIGNED	Sem sinal. Define que serão aceitos apenas números positivos.	Números inteiros
BINARY	Usado para diferenciar maiúsculas de minúsculas.	Texto

CREATE DATABASE

Este comando é utilizado para criar um banco de dados.

Exemplo:

CREATE DATABASE loja;

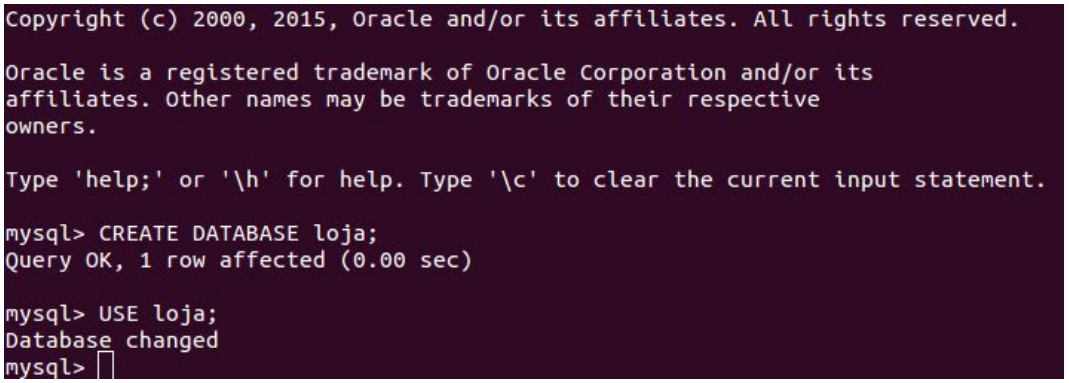
#cria um banco de dados chamado “loja”.

Comando USE

Sempre que for necessário manipular as tabelas de um banco de dados no SGBD MySQL, será necessário selecionar o banco de dados que se deseja manipular. Para isso, deve ser utilizado o comando USE.

Exemplo:

USE loja;
#acessa o banco de dados chamado “loja”
Demonstração dos comandos apresentados anteriormente, no console do MySQL.



CREATE TABLE

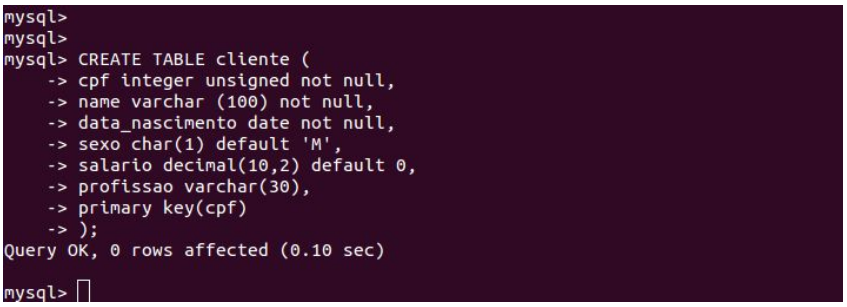
Sintaxe:

CREATE TABLE nome_da_tabela (..
definição dos campos
...
);

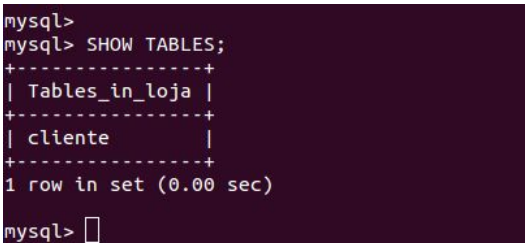
Exemplo:

CREATE TABLE cliente(
cpf integer unsigned not null,
nome varchar(100) not null,
data_nascimento date not null,
sexo char(1) default ‘M’,
salario decimal(10,2) default 0,
profissao varchar(30),
primary key(cpf)
);

A seguir a execução da instrução no console do MySQL.



Para consultar as tabelas existentes no banco de dados, utiliza-se o comando *SHOW TABLES*.



Cláusula DEFAULT

A cláusula DEFAULT permite definir um valor padrão para um campo, que será utilizado caso não seja informado nenhum valor para esse campo na inserção de um registro na tabela.

Sintaxe:

...
sexo char(1) default 'M',
...

No exemplo acima, caso o campo "sexo" da tabela não seja preenchido com um valor durante a inserção de um registro, será assumido o valor 'M' para o campo.
Para campos do tipo NUMÉRICO, o valor DEFAULT é escrito sem aspas.

Exemplo:

...
salario decimal(10,2) default 0,
...

CONSTRAINTS

a) NOT NULL: define que um campo da tabela é obrigatório (deve receber um valor na inserção de um registro);

b) PRIMARY KEY: define que um campo ou conjunto de campos para garantir a identidade de cada registro. Quando um campo é definido como chave primária, seu valor não pode se repetir em registros diferentes. Cada tabela só pode ter uma única chave primária.

- **CHAVE PRIMÁRIA SIMPLES:** composta por um único campo. Exemplo: se for definido que em um sistema de hotéis não podem existir dois clientes com o mesmo CPF, portanto este campo deverá ser definido como CHAVE PRIMÁRIA.
- **CHAVE PRIMÁRIA COMPOSTA:** formada por dois ou mais campos. Exemplo: se for definido em um sistema de Agências bancárias que não podem existir duas contas com o mesmo número da mesma agência, então esses dois campos formarão uma CHAVE PRIMÁRIA COMPOSTA, pois a combinação deles não pode se repetir.

Número da Conta	Número da Agência
1234	123
1234	567
3432	123

Observe que o número da conta pode se repetir individualmente, e o mesmo vale para o número da agência, porém a combinação desses dois campos não pode se repetir, garantindo que não existirão duas contas com o mesmo número na mesma agência.

Sintaxe:

Criação de uma chave primária simples

```
CREATE TABLE conta(
        numero integer not null primary key,
        saldo integer default 0,
        agencia_numero integer not null
        )
```

Ou

```
CREATE TABLE conta(
        numero integer not null,
        saldo integer default 0,
        agencia_numero integer not null,
        primary key(numero)
)
```

Ou

```
CREATE TABLE conta(
        numero integer not null,
        saldo integer default 0,
        agencia_numero integer not null,
        constraint pk_conta primary key(numero)
)
```

Criação de uma chave primária composta

```
CREATE TABLE conta(
        numero integer not null,
        saldo integer default 0,
        agencia_numero integer not null,
        primary key(numero,agencia_numero)
)
```

Ou

```
CREATE TABLE conta(
        numero integer not null,
        saldo integer default 0,
        agencia_numero integer not null,
        constraint pk_conta primary key(numero,agencia_numero)
)
```

c) FOREIGN KEY

Uma chave estrangeira é definida quando se deseja relacionar tabelas do banco de dados.

Sintaxe:

```
CREATE TABLE conta(
        numero integer not null,
        saldo integer default 0,
        agencia_numero integer not null,
        primary key(numero,agencia_numero),
        foreign key(agencia_numero) references agencias(numero)
)
```

Ou

```
CREATE TABLE conta(
        numero integer not null,
        saldo integer default 0,
        agencia_numero integer not null,
        primary key(numero,agencia_numero),
        constraint fk_contaagencia foreign key(agencia_numero) references
        agencias(numero)
)
```

Na criação da chave estrangeira do exemplo anterior, pode-se ler da seguinte forma: O campo agencia_numero da tabela contas faz referência ao campo número da tabela agencias.

d) UNIQUE

Uma constraint UNIQUE define que o valor de um campo ou de uma sequência de campos não pode se repetir em registros da mesma tabela. Essa constraint é criada de forma implícita quando é definida uma chave primária para uma tabela. Como só é possível ter uma chave primária por tabela, a utilização de constraints UNIQUE é uma solução quando se deseja restringir valores repetidos em outros campos.

Exemplo:

```
CREATE TABLE cliente(
        cpf integer not null,
```

nome varchar(100) not null,
data_nascimento date not null,
sexo char(1) default 'M',
salario decimal(10,2) default 0,
profissao varchar(30),
rg integer not null,
estado char(2) not null,
primary key(cpf),
constraint un_rgestado unique(rg,estado)
);

A criação da constraint acima garante que um número de RG não se repetirá em um mesmo estado.

DROP TABLE

O comando DROP TABLE é usado para apagar uma tabela do Banco de dados.

Sintaxe:

DROP TABLE nome_da_tabela;

ALTER TABLE

Para não se apagar uma tabela e recriá-la, é possível fazer alterações em sua estrutura por meio do comando ALTER TABLE. Isso é importante pois a execução do comando DROP TABLE apaga (obviamente) todos os registros da tabela, já a execução do comando ALTER TABLE não exclui nenhum registro.

Adicionar um campo

Sintaxe:

ALTER TABLE
nome_da_tabela
ADD
nome_do_campo tipo_de_dado atributos

***Exemplo*:**

ALTER TABLE
cliente
ADD
endereco varchar(90) not null;

Alterar o tipo de dado de um campo

Sintaxe:

ALTER TABLE
nome_da_tabela
MODIFY

nome_do_campo tipo_de_dado

Exemplo:

ALTER TABLE
cliente
MODIFY
endereco varchar(200);

Renomear um campo e modificar o tipo

Sintaxe:

ALTER TABLE
nome_da_tabela
CHANGE COLUMN
nome_do_campo novo_nome tipo atributos;

Exemplo para mudar apenas o nome (o tipo do campo é mantido):

ALTER TABLE
cliente
CHANGE COLUMN
data_nascimento datanasc date;

Exemplo para mudar o nome e o tipo do campo:

ALTER TABLE
cliente
CHANGE COLUMN
data_nascimento datahoranasc datetime;

Renomear uma tabela

Sintaxe:

ALTER TABLE
nome_da_tabela
RENAME TO
novo_nome_da_tabela

Exemplo:

ALTER TABLE
cliente
RENAME TO
pessoas_fisicas

Apagar um campo

Sintaxe:

ALTER TABLE
nome_da_tabela
DROP COLUMN
nome_do_campo

Exemplo:

ALTER TABLE
cliente
DROP COLUMN
endereco;

Adicionar uma PRIMARY KEY

Sintaxe:

ALTER TABLE
nome_da_tabela
ADD CONSTRAINT
nome_da_constraint
PRIMARY KEY
(campo1[,campo2,campo3,...,campoN])

Exemplo:

ALTER TABLE
cliente
ADD CONSTRAINT
pk_cpf
PRIMARY KEY
(cpf)

Apagar uma PRIMARY KEY

Sintaxe:

ALTER TABLE
nome_da_tabela
DROP PRIMARY KEY

Ou

ALTER TABLE
nome_da_tabela
DROP CONSTRAINT
nome_da_constraint_da_primary_key

Exemplo:

ALTER TABLE
cliente
DROP PRIMARY KEY;

Ou

ALTER TABLE
cliente
DROP CONSTRAINT
pk_cpf;

Adicionar uma FOREIGN KEY

Sintaxe:

ALTER TABLE
nome_da_tabela
ADD CONSTRAINT
nome_da_constraint
FOREIGN KEY
(campo1[,campo2,campo3,...,campoN])
REFERENCES
nome_da_tabela(campo1[,campo2,campo3,...,campoN]);

Exemplo:

ALTER TABLE
conta
ADD CONSTRAINT
fk_contaagencia
FOREIGN KEY
(agencia_numero)
REFERENCES
agencias(numero);

Adicionar uma constraint UNIQUE

Sintaxe:

ALTER TABLE
nome_da_tabela
ADD CONSTRAINT
nome_da_constraint
UNIQUE

(campo1[,campo2,campo3,...,campoN])

Exemplo:

ALTER TABLE
cliente
ADD CONSTRAINT
un_rgestado
UNIQUE
(rg,estado)

Apagar uma CONSTRAINT qualquer

Sintaxe:

ALTER TABLE
nome_da_tabela
DROP CONSTRAINT
nome_da_constraint

Exemplo:

ALTER TABLE
conta
DROP CONSTRAINT
fk_contaagencia

Comando SHOW TABLES

Para visualizar todas as tabelas em um banco de dados, utilize o comando SHOW TABLES.

Exemplo:

SHOW TABLES;

Comando DESC

Para visualizar a estrutura de uma tabela, utilize o comando DESC (ou DESCRIBE).

Exemplo:

DESC clientes;

10.2 DML no SGBD MySQL

10.2.1 Inserir registros em uma tabela

Para inserir dados em uma tabela utiliza-se o comando **insert**. No MySQL o **insert** consiste em inserir dados em uma tabela definida, pode-se inserir todos os campos da tabela, ou inserir somente campos definidos na instrução de inserção. A sintaxe deste comando segue o padrão a seguir:

```
INSERT INTO
        tabela (campo1,campo2)
VALUES
        ('valor1','valor2');
```

A seguir é apresentado um exemplo utilizando a tabela cliente para a inserção de dados:

```
INSERT INTO cliente
        (cpf, nome, data_nascimento, sexo, salario, profissao)
VALUES
        ('125.853.456-15', '1962-07-27', 'Pedro Boh','M', 1058.25, 'Operário');
```

ou

```
INSERT INTO cliente
        (cpf, nome, data_nascimento)
VALUES
        ('125.853.456-15', '1962-07-27','Pedro Boh');
```

Obs: Os campos identificados com não nulos, devem receber obrigatoriamente valores durante a inserção.

10.2.2 Consulta a dados de uma tabela

Para realizar uma consulta no bando de dados, utiliza-se o comando SELECT. Este comando consiste em selecionar uma coleção de registros de acordo com uma determinada condição. A seguir é representado a sintaxe básica deste comando.

```
SELECT
        lista_atributos
FROM tabela
        [WHERE condição]
```

No exemplo a seguir é realizado uma consulta que retorna todos os registros da tabela **cliente**.

```
SELECT  * FROM cliente
```

No exemplo acima a consulta retorna todos os registros incondicionalmente. Por este motivo não foi apresentado uma condição na consulta. Já no exemplo a seguir, a consulta está sendo condicionada aos registros dos clientes do sexo feminino(F).

```
SELECT * FROM cliente
WHERE sexo='F'
```

O comando **select** proporciona facilidades para projeção de informações através da eliminação de duplicatas, retorno de valores calculados com o uso de operadores aritméticos, invocação de funções de agregação.

Consulta com eliminação de duplicatas.

O exemplo a seguir apresenta uma consulta que retorna os diferentes valores de salário armazenados na tabela cliente, não exibindo valores repetidos.

SELECT salario
DISTINCT FROM cliente

Valores calculados

No exemplo a seguir é apresentado uma consulta que retorna o saldo de uma conta corrente, em reais, convertido para dólares. Considerando a cotação do dólar em R$2,50.

SELECT numero, saldo*2.5 AS saldo_dolar
FROM conta

Invocação de funções de agregação

As funções de agregação agrupam valores de acordo com alguns campos e retornam um valor baseado no conjunto de valores dos campos agregados, como uma soma, ou o menor valor entre o conjunto de valores.

As principais funções de agregação são:

- **COUNT** (contador de ocorrências [de um atributo])
- **MAX / MIN** (valores máximo / mínimo de um atributo)
- **SUM** (somador de valores de um atributo)
- **AVG** (média de valores de um atributo)

No exemplo a seguir é apresentado uma consulta que retorna o saldo médio encontrado na tabela conta.

SELECT AVG(saldo)
AS saldo_medio
FROM conta

10.2.3 Excluir registros de uma tabela

A exclusão de registros no banco de dados MySQL, é uma atividade rotineira. No MySQL a sintaxe para excluir um registro é a seguinte.

DELETE FROM tabela [WHERE condição]

10.2.4 Alterar registros em uma tabela.

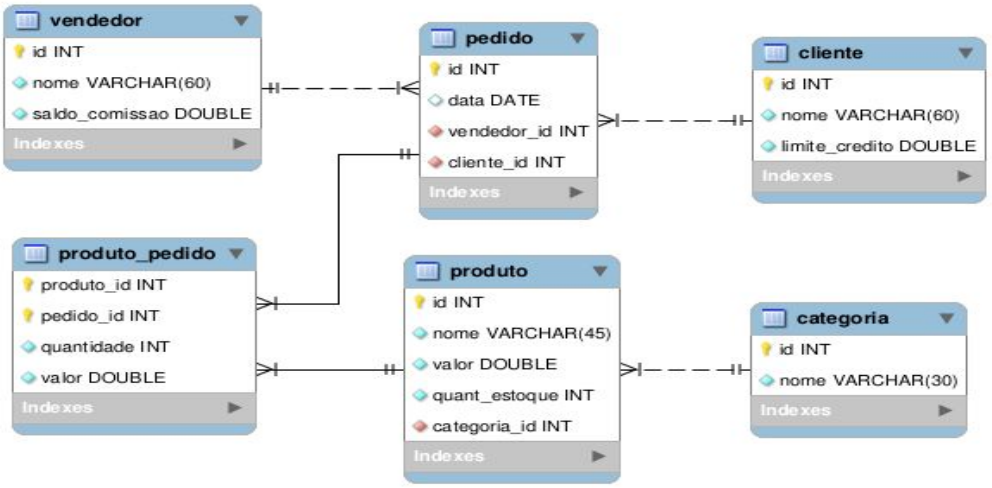
A edição de registros no banco de dados MySQL, também é uma atividade rotineira. No MySQL a sintaxe para editar um registro é a seguinte.

UPDATE FROM tabela SET coluna=valor [WHERE condição]

10.2.5 Consultas com JOIN

Joins (junções) são um recurso presente nos bancos de dados relacionais, através da qual é possível juntar o conteúdo de duas tabelas através de um critério. É um conceito que muitas vezes quem está iniciando no mundo dos bancos de dados relacionais tem dificuldade de entender. As joins mais utilizadas são: **INNER**, **LEFT**, **RIGHT**, **CROSS**.

Para os exemplos aqui apresentados serão utilizadas as tabelas cliente, produto, pedido e produto_pedido. Conforme modelo:



Segue a seguir o script para criação do banco de dados com as respectivas tabelas.

```
CREATE SCHEMA IF NOT EXISTS `loja` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci ;
USE `loja` ;

-- -----------------------------------------------------
-- Table `loja`.`cliente`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `loja`.`cliente` (
  `id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(60) NOT NULL,
  `limite_credito` DOUBLE NOT NULL DEFAULT 100.00,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_general_ci;
```

```sql
-- -----------------------------------------------------
-- Table `loja`.`vendedor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `loja`.`vendedor` (
  `id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(60) NOT NULL,
  `saldo_comissao` DOUBLE NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `loja`.`categoria`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `loja`.`categoria` (
  `id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(30) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_general_ci;


-- -----------------------------------------------------
-- Table `loja`.`produto`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `loja`.`produto` (
  `id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(45) NOT NULL,
  `valor` DOUBLE NOT NULL,
  `quant_estoque` INT NOT NULL,
  `categoria_id` INT UNSIGNED NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC),
  INDEX `fk_produto_categoria_idx` (`categoria_id` ASC),
  CONSTRAINT `fk_produto_categoria`
    FOREIGN KEY (`categoria_id`)
    REFERENCES `loja`.`categoria` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_general_ci;
```

```
-- -----------------------------------------------------
-- Table `loja`.`produto_pedido`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `loja`.`produto_pedido` (
  `produto_id` INT UNSIGNED NOT NULL,
  `pedido_id` INT UNSIGNED NOT NULL,
  `quantidade` INT NOT NULL,
  `valor` DOUBLE NOT NULL,
  PRIMARY KEY (`produto_id`, `pedido_id`),
  INDEX `fk_produto_has_pedido_pedido1_idx` (`pedido_id` ASC),
  INDEX `fk_produto_has_pedido_produto1_idx` (`produto_id` ASC),
  CONSTRAINT `fk_produto_has_pedido_produto1`
    FOREIGN KEY (`produto_id`)
    REFERENCES `loja`.`produto` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_produto_has_pedido_pedido1`
    FOREIGN KEY (`pedido_id`)
    REFERENCES `loja`.`pedido` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_general_ci;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

INSERT INTO `cliente`

VALUES (1,'Luciano de Almeida Soares',600),(2,'Amélia Ribeiro',350),(3,'Mateus Müller',750),(4,'Pedro Machado',50),(5,'Manuel Carcalho',250),(6,'Marcia Bitencurt',1200),(7,'Alcinéia Eltz',950),(8,'Luiz Inácio Magalhães',1100);

INSERT INTO `vendedor`

VALUES (1,'Carlos Aguiar',0),(2,'Pedro de Alcantara',250),(3,'Carmen Flores',125),(4,'Frederico Cardozo',359);

INSERT INTO `categoria`

VALUES (1,'Acessórios'),(2,'Equipamentos'),(3,'Software'),(4,'Serviços');

INSERT INTO `produto`

VALUES (1,'Mouse Mtek',22,25,1),(2,'Teclado Mtek ABNT',22,30,1),(3,'Monitor led 22\" LG',489,5,2),(4,'Controle de estoques FARTECH',319,20,3),(5,'Pen drive 16GB',55,15,1),(6,'Pen drive 8GB',35,16,1),(7,'Pen drive 4GB',20,12,1),(8,'Montor led 17\" LG',369,7,2),(9,'Notebook Acer Aspire E1-572-6',1599,8,2),(10,'Notebook CCE U25',999,6,2),(11,'SisFin Sistema de Gestão Financeira',419,15,3);

INSERT INTO `pedido`

VALUES

(1,'2016-02-03',1,2),(2,'2016-02-03',3,5),(3,'2016-03-01',4,6),(4,'2016-03-02',2,7),(5,'2016-03-05',1,8),(6,'2016-03-08',4,1),(7,'2016-03-10',2,4),(8,'2016-03-15',4,3);

INSERT INTO `produto_pedido`

VALUES(1,4,1,22),(2,2,1,22),(3,1,1,498),(4,5,1,319),(5,1,1,55),(5,5,1,55),(7,3,2,20),(8,7,1,369),(9,4,1,1599),(10,6,1,999);

INNER JOIN

Inner join (junção interna) é um tipo de junção na qual somente serão selecionados os resultados, cujas colunas informadas forem iguais nas duas tabelas. No exemplo a seguir serão retornados os registros da tabela **cliente** e da tabela **pedido**, que contenham registros em comum.

```
SELECT c.nome, p.data, p.vendedor_id
FROM cliente c
INNER JOIN pedido p ON p.cliente_id=c.id
```

A consulta a seguir retorna os registros em comum em um relacionamento contendo mais de duas tabelas.

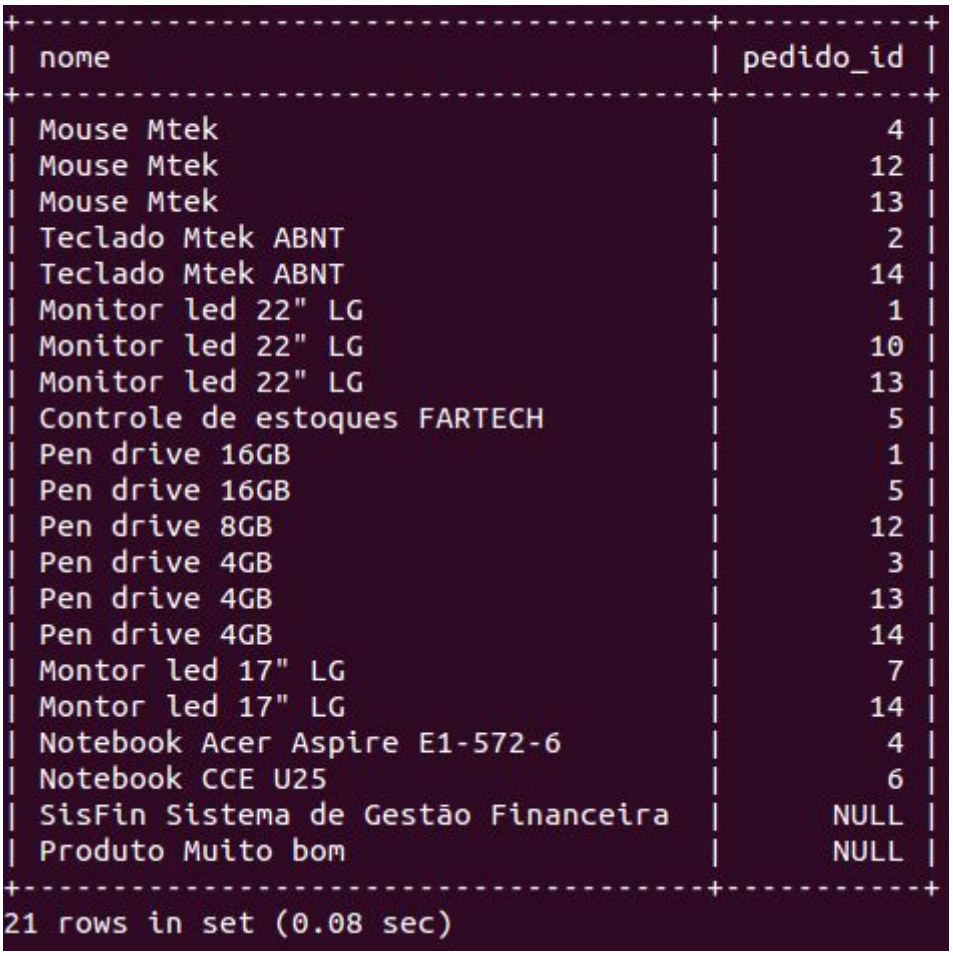
```
+---------------------------+------------+-------------+
| nome                      | data       | vendedor_id |
+---------------------------+------------+-------------+
| Luciano de Almeida Soares | 2016-03-08 |           4 |
| Luciano de Almeida Soares | 2000-12-12 |           1 |
| Amélia Ribeiro            | 2016-02-03 |           1 |
| Amélia Ribeiro            | 2000-12-12 |           2 |
| Mateus Müller             | 2016-03-15 |           4 |
| Mateus Müller             | 2000-12-12 |           3 |
| Mateus Müller             | 2012-01-12 |           3 |
| Pedro Machado             | 2016-03-10 |           2 |
| Pedro Machado             | 2012-12-01 |           4 |
| Pedro Machado             | 2015-06-01 |           4 |
| Manuel Carcalho           | 2016-02-03 |           3 |
| Marcia Bitencurt          | 2016-03-01 |           4 |
| Alcinéia Eltz             | 2016-03-02 |           2 |
| Luiz Inácio Magalhães     | 2016-03-05 |           1 |
+---------------------------+------------+-------------+
14 rows in set (0.16 sec)
```

LEFT JOIN

Left Join (junção esquerda) é semelhante ao INNER JOIN, porém aqui **TODOS** os dados da tabela a esquerda serão selecionados, independentes se possuírem ou não alguma relação com a tabela da direita.

```
SELECT p.nome,pp.pedido_id
FROM produto p
LEFT JOIN produto_pedido pp ON pp.produto_id=p.id
```

A consulta a seguir retorna todos registros do conjunto a esquerda relacionados aos conjuntos da direita. Se estas relações existirem.



nome	pedido_id
Mouse Mtek	4
Mouse Mtek	12
Mouse Mtek	13
Teclado Mtek ABNT	2
Teclado Mtek ABNT	14
Monitor led 22" LG	1
Monitor led 22" LG	10
Monitor led 22" LG	13
Controle de estoques FARTECH	5
Pen drive 16GB	1
Pen drive 16GB	5
Pen drive 8GB	12
Pen drive 4GB	3
Pen drive 4GB	13
Pen drive 4GB	14
Montor led 17" LG	7
Montor led 17" LG	14
Notebook Acer Aspire E1-572-6	4
Notebook CCE U25	6
SisFin Sistema de Gestão Financeira	NULL
Produto Muito bom	NULL

21 rows in set (0.08 sec)

RIGHT JOIN

Right Join (juncão direita) é idêntica a LEFT JOIN, com a diferença que neste caso todos os dados da coluna da direita serão selecionados, independentemente se os dados da coluna da direita tiverem ou não relação com os dados da coluna da esquerda.

CROSS JOIN

Cross Join (junção cruzada) é o tipo de junção em que para cada registro da primeira tabela, todos os registros da segunda tabela serão relacionados, ou seja, todos os registros da primeira tabela são consultados e é feito um produto cartesiano dos registros da primeira tabela com os registros da segunda tabela.

Exercícios:

A partir da base dedados loja, utilizada como exemplo na seção anterior forneça as consultas solicitadas.

1) Lista dos pedidos com os respectivos clientes e vendedores.

2) Lista de clientes que realizaram compras acima de R$100,00.

3) Lista de vendedores que emitiram pedidos contendo produtos da categoria 'software'.

4) Lista de vendedores e os respectivos totais de vendas.

5) Lista de clientes com as respectivas datas em que realizaram compras.

6) Quais os vendedores, que tiraram pedidos com o sabão em pó 'Ace', após Fevereiro.

7) Listar os itens dos pedidos informado a data do pedido, o id do pedido, o nome do cliente, nome do vendedor, o nome do produto e a categoria do produto.

8) Apresente uma consulta com todos os produtos já adquiridos pelo cliente 'pedro Bó'.

9) Lista dos clientes que compraram produtos classificados como acessórios.

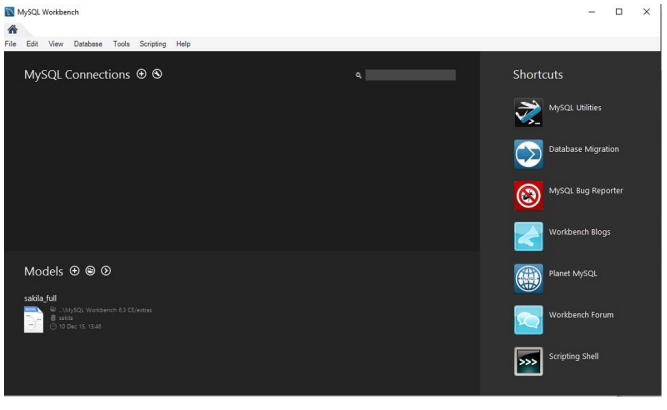
10) Listar todos os pedidos com os respectivos itens, informando ainda o vendedor e o cliente deste pedido.

10.3 Ferramentas

Existem diversas ferramentas de facilitam o trabalho de um administrador de banco de dados, dentre as principais ferramentas está o MySQLWorkbench, que foi incluído na instalação demonstrada na seção 9.1.

Para acessar o MySQLWorkbench acesse o seguinte caminho: Programas->mysql->MySQLWorkbench

Quando a ferramenta é carregada visualiza-se a tela apresentada a seguir.



As funcionalidades desta ferramenta serão tratadas nas próximas seções. Esta ferramenta é muito utilizada para criar o Diagrama Entidade Relacionamento (DER), no nível lógico. Ou seja, voltado ao SGBD em que será implantado. No caso o MySQL, pois trata-se de uma ferramenta exclusiva desta SGBD. Além da elaboração do modelo lógico também é disponibilizado uma série de recursos que facilitam a administração do SGBD. No link a seguir é possível encontrar um vídeo que demostra os principais recursos desta ferramenta https://www.youtube.com/watch?v=WTyqp5PjdGI

Exercícios:

Construa o Diagrama Entidade Relacionamento para os casos descritos a seguir.

1) Sistema de pedidos

Uma firma vende produtos de limpeza, e deseja melhor controlar os produtos que vende, seus clientes e os pedidos. Cada produto é caracterizado por um código, nome do produto, categoria (ex. detergente, sabão em pó, sabonete, etc), e seu preço. A categoria é uma classificação criada pela própria firma. A firma possui informações sobre todos seus clientes. Cada cliente é identificado por um código, nome, endereço, telefone, status ("bom", "médio", "ruim"), e o seu limite de crédito. Guarda-se igualmente a informação dos pedidos feitos pelos clientes. Cada pedido possui um número e guarda-se a data de elaboração do pedido. Cada pedido pode envolver de um a vários produtos, e para cada produto, informa-se a quantidade.

2) Sistema de ordens de serviço

Sistema de controle e gerenciamento de execução de ordens de serviço em uma oficina mecânica: Clientes levam veículos à oficina mecânica para serem consertados ou para passarem por revisões periódicas. Cada veículo é designado a uma equipe de mecânicos que identifica os serviços a serem executados e preenche uma ordem de serviço (OS) e prevê uma data de entrega. A partir da OS, calcula-se o valor de cada serviço, consultando-se uma tabela de referência de mão de obra. O valor de cada peça necessária à execução do serviço também é computado. O cliente autoriza a execução dos serviços e a mesma equipe responsável pela avaliação realiza os serviços. Clientes possuem código, nome, endereço e telefone. Veículos possuem código, placa e descrição. Cada mecânico possui código, nome, endereço e especialidade. Cada OS possui um número, uma data de emissão, um valor e uma data para conclusão dos trabalhos. Uma OS pode ser composta de vários itens (serviços) e um mesmo serviço pode constar em várias ordens de serviço. Uma OS pode envolver vários tipos de peças e um mesmo tipo de peça pode ser necessária em várias ordens de serviço.

3)Empresa de distribuição

A empresa de distribuição possui vários cinemas, em diversas localidades; Cada cinema possui uma identificação única, um nome fantasia, um endereço completo, incluindo rua, avenida, bairro, município, estado e sua capacidade de lotação; Os filmes podem ser dos mais variados tipos e gêneros; Cada filme é registrado com um título original, e se for filme estrangeiro, possuirá também o título em Português, o gênero, sua duração, sua impropriedade e seu país de origem, informações sobre os atores que compõem seu elenco, e seu diretor. Existirá um único diretor

para cada filme; Alguns cinemas apresentam mais de um filme em cartaz, sendo nestes casos, sessões alternadas com um filme e outro; As sessões possuem horários que variam de acordo com a duração do filme, havendo sempre um intervalo de aproximadamente 15 minutos entre elas; Os atores de um filme podem, obviamente, atuar em diversos filmes, assim como o diretor de um filme pode também ser ator neste filme ou ainda mais, ser ator em outro filme. Um ator possui as seguintes características: um número de identificação, um nome, uma nacionalidade e uma idade; As sessões de cinema devem ter seu público registra do diariamente, para que se permita a totalização dos assistentes quando o filme sair de cartaz, ou a qualquer instante;

4) Sistema de Controle Bancário

Faça o esquema conceitual para um sistema de controle bancário. Para cada agência do sistema deseja-se armazenar seu número, cidade e dados sobre os funcionários que ali trabalham, tais como nome, endereço, código e salário. Cada cliente cadastrado em uma agência específica pode possuir várias contas bancárias. Para os clientes deseja-se armazenar o nome, o RG e a cidade na qual residem, além de suas contas bancárias. Dados importantes para as contas dos clientes da agência são o número da conta, o saldo e informações sobre o conjunto de transações (número_transação, data, valor) associadas à conta.

5) Companhia

Faça o esquema conceitual para o banco de dados de uma companhia. A companhia é organizada em departamentos. Cada departamento tem um nome e um número. Além disto, um departamento controla vários projetos, cada um dos quais com um nome, um número de identificação e o período de tempo no qual deve ser desenvolvido. Na referida companhia, cada projeto somente pode ser desenvolvido por um departamento específico.

Existem somente três tipos de funcionários que trabalham na companhia: pesquisador, secretário e de limpeza. Para os pesquisadores, deseja-se armazenar: o nome, o endereço, o sexo, a data de aniversário, o salário e a área de atuação. Para os secretários, deseja-se armazenar: o nome, o endereço, o sexo, a data de aniversário, o salário e o grau de escolaridade. Já para os funcionários de limpeza, deseja-se armazenar: o nome, o endereço, o sexo, a data de aniversário, o salário, o cargo e a jornada de trabalho. Os cargos dos funcionários responsáveis pela limpeza são hierárquicos. Assim, deseja-se armazenar também, para cada funcionário de limpeza, informações sobre o funcionário de limpeza que o gerencia. Os funcionários da companhia são identificados por meio de um código de identificação, e podem estar associados a apenas um único departamento.

Funcionários que são pesquisadores podem trabalhar em diversos projetos, independentemente desses projetos estarem sendo desenvolvidos no mesmo departamento no qual o empregado está associado. Deve-se armazenar o número de horas semanais trabalhadas por cada pesquisador em cada projeto no qual ele trabalha.

Deve-se armazenar também informações sobre os dependentes de cada funcionário para propósitos de ajuda família. Deve-se armazenar o nome, o sexo e a data de aniversário, além do grau de parentesco com o funcionário.

6) Agência de Turismo

Deseja-se criar um BD para uma agência de turismo, contendo informações sobre recursos

oferecidos pelas cidades que fazem parte da programação de turismo da agência. As informações a serem mantidas sobre cada cidade referem-se a hotéis, restaurantes e pontos turísticos.
Sobre os hotéis que a cidade possui deseja-se guardar o código, o nome, o endereço, a categoria (sem estrela, 1 estrela, 2 estrelas, ...), os tipos de quartos que os formam (por exemplo, luxo, superluxo, master, ...), o número dos quartos e o valor da diária de acordo com o tipo do quarto.
Sobre cada cidade deve-se armazenar seu nome, seu estado e a população. Além disso, quando uma nova cidade é cadastrada no banco de dados da agência, um código é a ela oferecido.
Cada restaurante da cidade possui um código que o identifica, um nome, um endereço e o tipo de sua categoria (por exemplo, luxo, simples, ...). Além disso, um restaurante pode pertencer a um hotel e um hotel somente pode ser associado a um restaurante. Diferentes pontos turísticos da cidade estão cadastrados no sistema: igrejas, casas de show e museus. A agência de turismo somente trabalha com estes três tipos de pontos turísticos.
Nenhum outro é possível. Além da descrição e do endereço, igrejas devem possuir como característica a data e o estilo de construção. Já casas de show devem armazenar o horário de início do show (igual para todos os dias da semana) e o dia de fechamento (apenas um único dia na semana), além da descrição e do seu endereço. Finalmente, os museus devem armazenar o seu endereço, descrição, data de fundação e número de salas. Um museu pode ter sido fundado por vários fundadores. Para estes, deve-se armazenar o seu nome, a data de nascimento e a data da morte (se houver), a nacionalidade e a atividade profissional que desenvolvia. Além disso, um mesmo fundador pode ter fundado vários museus. Quando qualquer ponto turístico é cadastrado no sistema, ele também recebe um código que o identifica. O mesmo é válido para fundadores. Finalmente, casas de show podem possuir restaurante. Quando o cliente da agência reserva um passeio para uma casa de show, ele já sabe se esta possui restaurante e qual o preço médio da refeição, além da especialidade (comida chinesa, japonesa, brasileira, italiana, ...). Dentro de uma casa de show, apenas um único restaurante pode existir.
Faça o esquema conceitual para o banco de dados acima descrito. Defina restrições de participação total e parcial de forma apropriada.

Considerações: Os atributos endereço e data precisam ser decompostos. Considere hotel como apenas um único objeto físico, e não como uma cadeia de hotéis. O mesmo vale para restaurante e ponto turístico.

Gestão de Usuário

11

O MySQL possui um mecanismo que permite limitar o acesso de um usuário a apenas um banco, tabela ou coluna, além de poder controlar o acesso de acordo com o host a partir de onde está sendo feita a conexão com o servidor. Pode-se ainda, conceder privilégios diferentes para cada host de onde o usuário possa estabelecer a conexão. Assim, é possível que determinados comandos possam ser executados somente quando o usuário estiver em um host específico, por exemplo o mesmo host do servidor MySQL (localhost).

O MySQL armazena as informações dos seus usuários em 4 tabelas que estão localizadas no banco de dados **mysql**. Estas tabelas são a **user**, **db**, **tables_priv** e **columns_priv**. A tabela user armazena as informações de todos os usuários do banco e os privilégios globais deste usuário. A tabela db armazena os privilégios dos usuários específicos de um banco de dados. Finalmente, as tabelas tables_priv e columns_priv armazenam os privilégios associados a tabelas e colunas, respectivamente. Como estas tabelas possuem as informações dos usuários, bem como os seus privilégios, recomenda-se que apenas o administrador do banco de dados tenha acesso ao banco mysql (usuário **root**).

Para criar usuários e conceder privilégios no MySQL, utiliza-se o comando GRANT. Ao executar um comando GRANT para um usuário que não existe, o mesmo será criado. Um GRANT para um usuário já existente adicionará os novos privilégios aos já concedidos anteriormente. A sintaxe resumida do comando GRANT é exibida a seguir:

GRANT priv [(*colunas*)] [, priv [(*colunas*)]] ...
ON {*.* | db.* | *db.tabela*}
TO *usuario* [IDENTIFIED BY '*senha*']
[, *usuario* [IDENTIFIED BY 'senha']] ...
[WITH [GRANT OPTION |
MAX_QUERIES_PER_HOUR *contador* |
MAX_UPDATES_PER_HOUR *contador* |
MAX_CONNECTIONS_PER_HOUR *contador*]]

No comando acima os **[ ]** indicam que o comando é opcional. O primeiro item a ser informado é(são) o(s) privilégio(s) a ser(em) concedido(s) ao(s) usuário(s). A lista de privilégios existentes no MySQL é descrita a seguir:

Privilégio	Descrição
ALL [PRIVILEGES]	Todos os privilégios exceto **GRANT OPTION**
ALTER	Permite executar **ALTER TABLE**
CREATE	Permite executar **CREATE TABLE**
CREATE TEMPORARY TABLES	Permite executar **CREATE TEMPORARY TABLE**

DELETE	Permite executar **DELETE**
DROP	Permite executar **DROP TABLE**
EXECUTE	Permite executar stored procedures (MySQL 5.0)
FILE	Permite executar **SELECT ... INTO OUTFILE** e **LOAD DATA INFILE**
INDEX	Permite executar **CREATE INDEX** e **DROP INDEX**
INSERT	Permite executar **INSERT**
LOCK TABLES	Permite executar **LOCK TABLES** em tabelas que você tenha o privilégio**SELECT**
PROCESS	Permite executar **SHOW FULL PROCESSLIST**
REFERENCES	Ainda não está implementado
RELOAD	Permite executar **FLUSH**
REPLICATION CLIENT	Permite ao usuário obter a localização do Master ou Slave
REPLICATION SLAVE	Necessário para a replicação Slave (leitura dos eventos do log binário do Master)
SELECT	Permite executar **SELECT**
SHOW DATABASES	exibe todos os bancos de dados
SHUTDOWN	Permite executar mysqladmin shutdown
SUPER	Permite executar **CHANGE MASTER**, **KILL** , **PURGE MASTER LOGS** e**SET GLOBAL**. Permite conectar-se ao servidor uma vez, mesmo que o max_connections tenha sido atingido
UPDATE	Permite executar **UPDATE**
USAGE	Sinônimo para "no privileges"
GRANT OPTION	Permite ao usuário repassar os seus privilégios

Uma vez informados os privilégios do usuário, você deverá indicar o nível ao qual o privilégio se aplica, sendo possível especificar três níveis:

.	Privilégio global
db.*	Qualquer tabela do banco db
db.tb	Apenas a tabela tb do banco de dados db. Para especificar apenas algumas colunas de uma determinada tabela, estas deverão ser listadas ao lado do privilégio (**priv (*colunas*)**)

Depois do nível você deverá indicar o usuário, ou a lista de usuários, para os quais os

privilégios se aplicam. No MySQL o usuário é constituído de um nome mais o host de onde ele poderá acessar o servidor (user@host). Caso você não informe o host para o usuário, o MySQL assumirá "%", isto é, todos os hosts. A senha do usuário é opcional, mas é recomendado sempre informá-la no momento de criação do usuário, por questões de segurança. Para adicionar privilégios a um usuário existente o **IDENTIFIED BY** poderá ser omitido. No exemplo a seguir é criado um usuário com o nome teste que pode se conectar somente do host onde o servidor está em execução (localhost), o usuário só poderá fazer **SELECT** nas colunas nome e idade da tabela pessoa, que se encontra no banco de dados rh. A senha do usuário é 12345.

mysql>GRANT SELECT (nome, idade) ON rh.pessoa TO teste@localhost
IDENTIFIED BY "12345";

Para listar os privilégios deste usuário utilize o comando:

mysql>SHOW GRANTS FOR teste@localhost;

Você pode especificar um conjunto de hosts utilizando o caracter "%", neste caso é possível dar acesso a um usuário dentro de uma faixa de IPs ou DNS. Usuários anônimos também podem ser criados informando um nome com o caracter espaço (" "). No exemplo a seguir, o usuário remoto poderá executar UPDATE e INSERT em qualquer tabela do banco de dados rh, sendo possível a conexão ao servidor a partir de qualquer máquina no domínio cimol.com.br:

mysql>GRANT UPDATE, INSERT ON rh.* TO remoto@"%.cimol.com.br"
IDENTIFIED BY "remoto";

Ou utilizando IP:

mysql>GRANT UPDATE, INSERT ON rh.* TO remoto@"192.168.13.%" IDENTIFIED BY "remoto";

Um usuário anônimo com os mesmos privilégios do usuário remoto seria criado da seguinte forma:

mysql>GRANT UPDATE, INSERT ON rh.* TO " "@"%cimol.com.br"
IDENTIFIED BY "anonimo";

Finalmente, a opção GRANT OPTION é utilizada para que o usuário possa conceder os seus privilégios para outros usuários do banco. Além disto, você poderá limitar os recursos do usuário com as opções

MAX_QUERIES_PER_HOUR, **MAX_UPDATES_PER_HOUR** ou **MAX_CONNECTIONS_PER_HOUR**.

Para remover um privilégio do usuário utilize o comando REVOKE mostrado abaixo:

REVOKE priv [(colunas)] [, priv [(colunas)]] ...
ON {*.* | db.* | db.tabela}
FROM usuario [, usuario] ...

Lembre-se de que a parte **ON** do **REVOKE** deverá coincidir com a parte **ON** do **GRANT** que você deseja remover, caso isto não se verifique o comando **REVOKE** não terá efeito algum. Além disto, o comando **REVOKE** remove apenas os privilégios do usuário, mas o usuário continuará existindo. A remoção do usuário deverá ser feita

com um**DELETE** explícito na tabela de usuários do MySQL, após terem sido removidos todos os seus privilégios com o comando **REVOKE**. Observe que para remover o usuário você deverá ter privilégio para executar **DELETE** na tabela user do mysql. Geralmente, esta tarefa é executada pelo administrador do banco (**root**).

mysql>DELETE FROM mysql.user WHERE user="teste" AND host="localhost";

Feito isto, você terá que executar um comando **FLUSH PRIVILEGES** para que o MySQL possa atualizar os privilégios que estão em memória.

mysql>FLUSH PRIVILEGES;

Aqui estamos descrevendo os comandos para a manipulação de usuários, sendo que esta é a maneira de entendermos como funciona a criação de usuários no MySQL. Uma forma mais intuitiva para executar esta tarefa pode ser encontrada no MySQL Administrator, que foi apresentado no último artigo.

11.1 Problemas para a conexão com o MySQL a partir do localhost:

Ao instalar o MySQL são criados o usuário root com todos os privilégios (administrador), podendo se conectar somente do host local sem senha, e o usuário anônimo com privilégios apenas no banco de dados test, que pode se conectar apenas do host local sem senha. Para autentiticar um usuário durante a conexão com o servidor, o MySQL armazena em memória a listas de todos usuários, hosts e senhas cadastrados no banco e as ordena do host mais específico para o menos específico. Caso existam dois hosts iguais, os usuários mais específicos virão antes do usuário anônimo. Feito isto, a cada conexão será pesquisada nesta lista a primeira ocorrência que coincida o host, usuário e senha informados no momento da tentativa de conexão. Tomemos o caso em que após a instalação do MySQL, criamos um usuário que possa se conectar a partir de qualquer host dentro do domínio onde se encontra o servidor MySQL:

mysql>GRANT ALL ON rh.* TO user@"192.168.0.%" IDENTIFIED BY "12345";

Neste caso, o MySQL cria em memória a lista dos usuários da seguinte forma:

Usuário Senha
root@localhost
" "@localhost
user@"192.168.0.%" 12345

Este usuário será capaz de se conectar ao servidor MySQL a partir de qualquer máquina no domínio, exceto o host local, pois neste caso, ao pesquisar na lista pelo user@localhost, a primeira ocorrência encontrada é a do usuário anônimo, já que qualquer usuário pode ser considerado anônimo pelo MySQL. O detalhe é que as senhas destes usuários não coincidem, o que gera um erro de "Acesso Negado". Para evitar a situação acima você deverá excluir o usuário anônimo ou adicionar o mesmo usuário para se conectar remoto e a partir do host local, da seguinte forma:

mysql>GRANT ALL ON rh.* TO user@"192.168.0.%" IDENTIFIED BY "12345",
user@localhost IDENTIFIED BY "12345";

Assim a ordenação ficaria:

Usuário Senha
root@localhost
user@localhost 12345
" "@localhost
user@"192.168.0.%" 12345

Desta forma, o primeiro registro encontrado não é mais o usuário anônimo e sim o user@localhost, portanto a conexão será estabelecida com sucesso. Para evitar este tipo de transtorno é melhor remover o usuário anônimo após a instalação do servidor, eliminando também eventuais problemas de segurança.

Opções avançadas 12

12.1 Views

Para quem trabalha com desenvolvimento de sistemas e administração de dados diretos em bases de dados concentradas em um SGBD como o MySQL, Oracle ou SQL Server, sabe o quanto é chateante ter que escrever e reescrever determinadas consultas todos os dias ou mesmo mais de uma vez no mesmo dia. Muitas destas consultas são derivadas de várias tabelas o que nos dá um re-trabalho ao montar todas aquelas JOIN´s, utilizar esse ou aquele índice setado para esta ou aquela tabela para que também a performance de tal consulta tenha um tempo razoavelmente atraente.

Sabemos também que a cada momento em que reescrevemos uma mesma consulta, conseguir os mesmos resultados de antes é uma tarefa séria, já que existem várias formas para se escrever uma mesma consulta, levando em conta a ordem em que as tabelas aparecem, a ordem dos campos e tudo mais. Tudo isso implica em ganho ou perda de performance. Estamos falando tanto em performance que parece que nosso artigo tomou outros rumos, mas não. De fato, quando temos um grande trabalho de ajuste de performance em uma consulta, podemos rapidamente transformar esta em uma View, que a partir disso, permanecerá armazenada no servidor de bancos de dados em forma de tabela, para que possamos consultá-la todas as vezes que precisarmos, sem ter que reescrever a mesma, aproveitando todo o trabalho de refino de performance supracitado.

12.1.1 Mas, o que é uma View?

Uma View é um objeto que pertence a um banco de dados, definida baseada em declarações SELECT´s, retornando uma determinada visualização de dados de uma ou mais tabelas. Esses objetos são chamados por vezes de “virtual tables”, formada a partir de outras tabelas que por sua vez são chamadas de “based tables” ou ainda outras Views. E alguns casos, as Views são atualizáveis e podem ser alvos de declaraçãoINSERT, UPDATE e DELETE, que na verdade modificam sua “based tables”.

Os benefícios da utilização de Views, além dos já salientados, são:

- Uma View pode ser utilizada, por exemplo, para retornar um valor baseado em um identificador de registro;
- Pode ser utilizada para promover restrições em dados para aumentar a segurança dos mesmos e definir políticas de acesso em nível de tabela e coluna. Podem ser configurados para mostrar colunas diferentes para diferentes usuários do banco de dados;
- Pode ser utilizada com um conjunto de tabelas que podem ser unidas a outros conjuntos de tabelas com a utilização de JOIN´s ou UNION.

12.1.2 Criando Views

Para definir Views em um banco de dados, utilize a declaração CREATE VIEW, a qual tem a seguinte sintaxe:

CREATE [OR REPLACE] [ALGORITHM = algorithm_type] VIEW

VIEW view_name [(column_list)]
AS select_statement
[WITH [CASCADED | LOCAL] CHECK OPTION]

view_name: é o nome que damos ao objeto View que criarmos. Esse nome poderá ser não qualificado, quando criado no banco de dados corrente ou totalmente qualificado quando criarmos em um banco de dados que não está definido no contexto atual (db_name.view_name).

column_list: recurso para que possamos sobrescrever os nomes das colunas que serão recuperadas pela declaração SELECT;

select_statement: é a sua declaração SELECT e indica a forma na qual você deseja que os dados sejam retornados. Tal declaração deverá indicar a forma a qual você deseja retornar os dados, podendo ser utilizado funções, JOIN´s e UNION. Podem ser utilizadas quaisquer tabelas ou views contidas no servidor de bancos de dados MySQL, observando a questão de nomes totalmente qualificados ou não.

OR REPLACE: pode ser utilizada para substituir uma View de mesmo nome existente no banco de dados ao qual ela pertence. Pode-se utilizar ALTER TABLE para o mesmo efeito;

ALGORITHM: essa cláusula define qual algoritmo interno utilizar para processar a View quando a mesma for invocada. Estes podem ser UNDEFINED (cláusula em branco),MERGE ou TEMPTABLE.

WITH CHECK OPTION: todas as declarações que tentarem modificar dados de uma view definida com essa cláusula serão revisadas para checar se as modificações respeitarão a condição WHERE, definida no SELECT da View.

Ao criar uma View, um arquivo com extensão ".frm", com o mesmo nome desta também é criado no diretório do banco de dados, sob o diretório de dados do MySQL (datadir).

12.1.3 Definindo Views

Para iniciar com a mão na massa, a seguir é definida uma View simples para listar produtos da tabela produto, do banco de dados loja, como segue na figura 01.

CREATE VIEW vw_produto AS SELECT * FROM produto

A seguir a execução da view, recém criada, com o comando SELECT

SELECT * FROM vw_produto

Resultado

#	id	nome	valor	quant_estoque	categoria_id
1	1	Mouse Mtek	22	25	1
2	2	Teclado Mtek ABNT	22	30	1
3	3	Monitor led 22" LG	489	5	2
4	4	Controle de estoques FARTECH	319	20	3
5	5	Pen drive 16GB	55	15	1

Ao executar o comando SHOW TABLES no banco de dados World, visualiza-se que uma tabela adicional foi criada, que é a View recém criada. Para uma conceituação mais ampla, uma View é um mapeamento lógico de várias tabelas contidas em um ou mais bancos de dados que

por sua vez estão em um servidor MySQL. No caso da View criada, tem uma tabela virtual (vw_produto) baseada em uma tabela chamada de base (produto). Um bom exemplo para utilizar a tal lista de colunas é criar a mesma View, mas agora sobrescrevendo o nome da coluna “name” para “produto”, como é demostrado a seguir.

CREATE OR REPLACE VIEW vw_produto (produto) AS SELECT nome FROM produto

Como a View vw_produto teve que ser sobrescrita, pois já existia uma outra com o mesmo nome. Para tanto, usou-se o OR REPLACE para facilitar o trabalho.

Agora o resultado para o comando SELECT, apresentado anteriormente, será o seguinte:

#	produto
1	Mouse Mtek
2	Teclado Mtek ABNT
3	Monitor led 22" LG
4	Controle de estoques FARTECH
5	Pen drive 16GB

É possível criar Views mais sofisticadas, com um comando SELECT mais trabalhado, pode-se utilizar as cláusulas WHERE, GROUP BY, HAVING e ORDER BY. Alguns SGBD’s comerciais não permitem a utilização de ORDER BY em meio ao SELECT na definição de uma View, a exemplo do SQL Server, da Microsoft. A Figura a seguir mostra uma consulta mais trabalhada para a criação de uma View, envolvendo as tabelas produto e categoria do banco de dados loja.

```
CREATE OR REPLACE VIEW vw_produto
AS SELECT p.nome AS produto, c.nome AS categoria
FROM produto p
INNER JOIN  categoria c
WHERE p.categoria_id=c.id
```

Resultado da execução da consulta com select.

#	produto	categoria
1	Mouse Mtek	Acessórios
2	Teclado Mtek ABNT	Acessórios
3	Pen drive 16GB	Acessórios
4	Pen drive 8GB	Acessórios
5	Pen drive 4GB	Acessórios
6	Monitor led 22" LG	Equipamentos
7	Montor led 17" LG	Equipamentos
8	Notebook Acer Aspire E1-572-6	Equipamentos
9	Notebook CCE U25	Equipamentos
10	Controle de estoques FARTECH	Software
11	SisFin Sistema de Gestão Financeira	Software

12.1. 4 Atualizando Views

Views podem ser constituídas facilmente, como visto anteriormente. Mas para ter-se Views que podem receber declarações de atualização tais como UPDATE e DELETE, que de fato

alteram as tabelas base (“based tables”), deve-se ter alguns cuidados na criação do objeto de visualização. Uma View criada com funções agregadas, por exemplo, não poderá receber atualizações, pois os dados logicamente estão agregados ou agrupados e não teram correspondências diretas para uma exclusão ou atualização.

DROP VIEW: Para excluir uma View, basta utilizar o comando DROP VIEW view_name. Trabalhar com View ou visualizações nada mais é que um papeamento lógicos de outras tabelas em uma nova, definido com comandos SELECT’s. Foi visto como criar Views, como alterar ou sobrescrever as mesmas e trabalhar com restrições ao atualizá-las.

Exercícios:

1) Criar um usuário chamado aluno, com direitos de acesso na base de dados aeroporto no servidor local, 'localhost'.

2) Criar uma visão que disponibilize somente a lista dos aeroportos com as respectivas cidades.

3) Criar uma visão que disponibilize a lista de clientes e as cidades para as quais já viajaram.

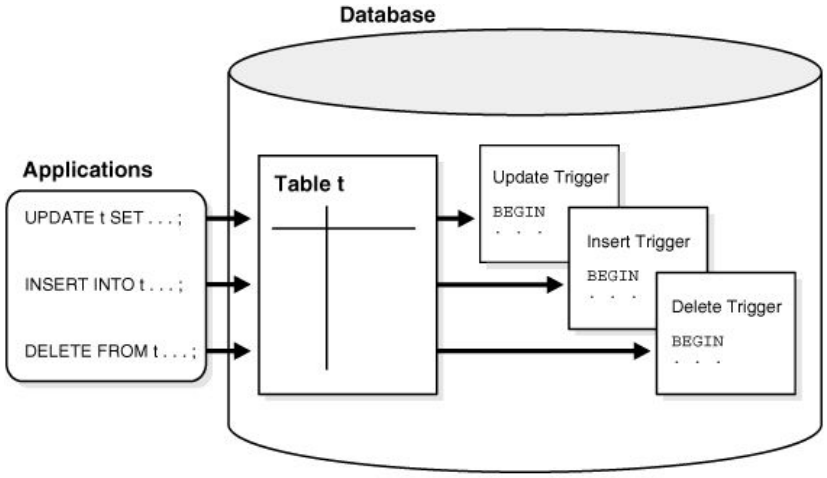
4) Permitir que o usuário 'aluno', tenha acesso as visões criadas nos exercícios anteriores.

5) Conectar ao banco de dados através do usuário aluno. Em seguida, apresente a instrução capaz de retornar a lista dos aeroportos e suas respectivas cidades.

6)Alterar a permissão de acesso do usuário 'aluno' as visões do banco 'aeroporto'. Permitir apenas leitura 'SELECT'.

12.2 Triggers

Trigger é um recurso de programação executado sempre que o evento associado ocorrer. Trigger é um tipo especial de procedimento armazenado, que é executado sempre que há uma tentativa de modificar os dados de uma tabela que é protegida por ele. É muito utilizada para ajudar a manter a consistência dos dados ou para propagar alterações em um determinado dado de uma tabela para outras. Um bom exemplo é um gatilho criado para controle de quem alterou a tabela, nesse caso, quando a alteração for efetuada, a trigger é "disparada" e grava em uma tabela de histórico de alteração, o usuário e data/hora da alteração.



Os principais pontos positivos sobre os triggers são: Parte do processamento que seria executado na aplicação passa para o banco, poupando recursos da máquina cliente;Facilita a manutenção, sem que seja necessário alterar o código fonte da aplicação;
Já contra sua utilização existem as seguintes considerações: Alguém que tenha acesso não autorizado ao banco de dados poderá visualizar e alterar o processamento realizado pelos gatilhos;Requer maior conhecimento de manipulação do banco de dados (SQL) para realizar as operações internamente.

A seguir é explicado o processo de criação de triggers, a sintaxe utilizada e o significado de cada instrução.

A sintaxe dos comandos para criar um novo trigger no MySQL é a seguinte:

```
CREATE TRIGGER nome momento evento
ON tabela
FOR EACH ROW
BEGIN
/*corpo do código*/
END
```

Onde se tem os seguintes parâmetros:

- nome: nome do trigger, segue as mesmas regras de nomeação dos demais objetos do banco.
- momento: quando o trigger será executado. Os valores válidos são BEFORE (antes) e AFTER (depois).
- evento: evento que vai disparar o trigger. Os valores possíveis são INSERT, UPDATE e DELETE. Vale salientar que os comandos LOAD DATA e REPLACE também disparam os eventos de inserção e exclusão de registros, com isso, os gatilhos também são executados.
- tabela: nome da tabela a qual o gatilho está associado.
- Não é possível criar mais de um trigger para o mesmo evento e momento de execução na mesma tabela. Por exemplo, não se pode criar dois triggers AFTER INSERT na mesma tabela.

12.2.1 Os registros NEW e OLD

Como os triggers, são executados em conjunto com operações de inclusão e exclusão, é necessário poder acessar os registros que estão sendo incluídos ou removidos. Isso pode ser feito através das palavras NEW e OLD. Em triggers executados após a inserção de registros, a palavra reservada NEW dá acesso ao novo registro. Pode-se acessar as colunas da tabela como atributo do registro NEW, como veremos nos exemplos.

O operador OLD funciona de forma semelhante, porém em triggers que são executados com a exclusão de dados, o OLD dá acesso ao registro que está sendo removido.

12.2.2 Utilização do trigger

Para exemplificar e tornar mais clara a utilização de triggers, é simulado a seguinte situação: um mercado que, ao realizar vendas, precisa que o estoque dos produtos seja automaticamente reduzido. A devolução do estoque deve também ser automática no caso de remoção de produtos da venda.

Como se trata de um ambiente hipotético, teremos apenas duas tabelas de estrutura simples, cujo script de criação é mostrado na listagem a seguir.

```
CREATE TABLE produtos (
   referencia  VARCHAR(3) PRIMARY KEY,
   descricao   VARCHAR(50) UNIQUE,
   estoque INT NOT NULL
);

INSERT INTO Produtos VALUES ('1', 'Lasanha', 10);
INSERT INTO Produtos VALUES ('2', 'Morgango', 5);
INSERT INTO Produtos VALUES ('3', 'Farinha', 15);

CREATE TABLE itensvenda (
   venda INT,
   produto VARCHAR(3),
   quantidade  INT
);
```

Ao inserir e remover registro da tabela itensvenda, o estoque do produto referenciado deve ser alterado na tabela produtos. Para isso, serão criados dois triggers: um AFTER INSERT para dar baixa no estoque e um AFTER DELETE para fazer a devolução da quantidade do produto.

```
DELIMITER;;

CREATE TRIGGER tgr_itensvenda_insert AFTER INSERT
ON itensvenda
FOR EACH ROW
BEGIN
UPDATE produtos SET estoque = Estoque - NEW.quantidade
WHERE referencia = NEW.produto;
END;
```

```
CREATE TRIGGER tgr_itensvenda_delete AFTER DELETE
ON itensvenda
FOR EACH ROW
BEGIN
   UPDATE produtos SET estoque = estoque + OLD.quantidade
WHERE referencia = OLD.produto;
END;

DELIMITER;
```

No primeiro trigger, foi utilizado o registro NEW para obter as informações da linha que está sendo inserida na tabela. O mesmo é feito no segundo trigger, onde se obtém os dados que estão sendo apagados da tabela através do registro OLD. Tendo criado os triggers, pode-se testá-los inserindo dados na tabela itensvenda. Nesse caso, é simulado uma venda de número 1 que contem três unidades do produto 1, uma unidade do produto 2 e cinco unidades do produto 3.

Insere dados na tabela.

```
INSERT INTO itensvenda VALUES (1, '1',3);
INSERT INTO itensvenda VALUES (1, '2',1);
INSERT INTO itensvenda VALUES (1, '3',5);
```

Nota-se que o estoque dos produtos foi corretamente reduzido, de acordo com as quantidades "vendidas". Agora para testar o trigger da exclusão, será removido o produto 1 dos itens vendidos. Com isso, o seu estoque deve ser alterado para o valor inicial, ou seja, 10.

```
DELETE FROM itensvenda WHERE venda = 1 AND produto = '1';
```

Em ambientes reais, triggers podem ser utilizados para operações mais complexas, por exemplo, antes de vender um item, verificar se há estoque disponível e só então permitir a saída do produto.