

# INTRODUÇÃO A BANCO DE DADOS

## Apresentação

Para construirmos um banco de dados temos que iniciar um projeto. Em qualquer projeto de banco de dados, a primeira etapa consiste na modelagem dos dados.

A modelagem dos dados, muitas vezes, é tratada simplesmente visando a implementação de estruturas de dados, tendo assim, uma forma errada de enxergar a modelagem dos dados. Uma grande parte de literaturas existentes no mundo dos bancos de dados mostram grande ênfase ao processo de implementação lógica, utilizando a abordagem relacional na maioria dos casos, e “pulando” o item primordial para a modelagem, que consiste no processo de construir um modelo conceitual.

As maiores dificuldades em começar o projeto pela fase conceitual devem-se ao fato da enorme preocupação com os aspectos tecnológicos, que consequentemente inibem a capacidade de “enxergar o mundo que modelam a sua frente”. Não podemos ficar presos às exigências tecnológicas, tais como: normalização, redundância, chaves de acesso e outras, pois isso afeta a abstração do mundo real.

A modelagem de dados não é útil somente ao projeto de banco de dados, é uma ferramenta de grande valia em outras áreas, tais como: O&M, processos de reengenharia para definição de regras de negócio etc.

O sucesso de qualquer projeto de banco de dados inicia-se com uma modelagem conceitual como ponto de partida para o processo de modelagem lógica dos dados.

## Objetivos

Esta disciplina visa apresentar os conceitos fundamentais de Banco de Dados e apresentar, de forma didática e prática, o desenvolvimento da modelagem de dados.

A construção de bases de dados pode ser dividida em três diferentes partes: modelagem conceitual, projeto lógico e projeto físico. Este curso tem como objetivo abranger somente duas primeiras etapas, tais como:

- Modelagem Conceitual, utilizando a abordagem entidade-relacionamento.
- Projeto Lógico, utilizando a abordagem relacional.

# INTRODUÇÃO

## O Ciclo de Vida dos Bancos de Dados

Assim como qualquer software, os bancos de dados também passam por um ciclo de vida cheio de mudanças. O projeto de um banco de dados é um processo interativo e incremental.

## Análise de Requisitos

Todo banco de dados nasce a partir do momento que pessoas possuem necessidades. É com essas necessidades que temos que nos preocupar quando iniciamos um novo projeto de banco de dados. O usuário final é nosso consumidor final do banco de dados que estará utilizando o sistema através dos programas aplicativos. O usuário do sistema é um consumidor direto do banco de dados, pois ele o utiliza para construir o sistema que será usado pelo usuário final.

Qualquer um desses usuários, tanto o final quanto o do sistema, possui necessidades específicas que devemos conhecer antes de iniciar o projeto.

## Modelagem de Dados

Após as necessidades dos usuários, temos que iniciar a modelagem formal do problema em questão. A função da modelagem é ajudar a organizar a maneira de pensar sobre os dados, esclarecendo assim seu significado e para que eles servirão na prática.

Em um projeto de banco de dados, a primeira fase é a modelagem dos dados, a qual chamamos de Modelo Conceitual. O modelo conceitual, além de facilitar a comunicação durante o desenvolvimento do banco de dados, é uma maneira segura devido ao fato de que reduz a complexidade a um nível que o projetista possa entender e construir estruturas de dados cada vez mais complexas, por isso a modelagem possui essa enorme importância.

## Projeto do Banco de Dados

O projeto de banco de dados inicia-se assim que começamos a pensar nas necessidades e como as coisas se relacionam umas com as outras. Quando acres-

## INTRODUÇÃO A BANCO DE DADOS



centamos algo novo no projeto, estamos modelando e quando relacionamos uma coisa com a outra, estamos projetando. Portanto, existe um processo de interação cíclica e incremental entre modelagem e projeto.

Depois que o modelo de dados conceitual estiver pronto, iremos transformá-lo em um esquema, que chamaremos de projeto lógico. Neste ponto, podemos utilizar uma ferramenta CASE (Veja Apêndice B), que fornece uma geração automática a partir do modelo, gerando o esquema relacional, facilitando também a consistência entre o modelo e o esquema que devem estar sempre em sincronia. Após qualquer alteração, o esquema do banco de dados também deve ser atualizado, ou seja, após retornar aos requisitos e novamente à modelagem, também existirão mudanças no esquema do banco de dados.

Após o projeto lógico, passamos ao projeto físico, que também não é uma seqüência de passos, mas um processo interativo, portanto, quando estamos desenvolvendo o projeto físico podemos perceber que precisamos voltar e realizar alterações no projeto lógico e, muitas vezes, até no modelo conceitual.

O projeto físico visa construir os métodos de acesso e as estruturas de armazenamento no modelo físico do banco de dados. A ênfase no projeto físico é pensar na melhoria do desempenho do sistema, ou seja, otimização e desempenho e não mais na modelagem.

O projeto ainda não acabou até que consideremos todos os riscos que o banco de dados poderá sofrer e também que métodos de gerenciamento podemos utilizar para reduzir ou até evitar esses riscos. Um risco é a probabilidade existente de acontecer algo que gere consequências negativas. Dentro da área de banco de dados, temos os seguintes riscos: desastres, falhas de hardware, falhas e defeitos de software, perdas acidentais e ataques propostais aos dados contidos no servidor. Após determinar o nível de tolerância a falhas, temos que gerenciar os riscos para que fiquem neste nível aceitável. Por exemplo, se não podemos, de nenhuma maneira, tolerar nem um minuto o banco de dados fora do ar, precisamos utilizar recursos avançados de tolerância a falhas de produtos SGBD modernos.

Nesta disciplina, iremos nos aprofundar nas duas primeiras etapas de um projeto de banco de dados: modelagem conceitual e projeto lógico.

## Qualidade, Revisões e Testes do BD

Requisitos, projeto e construção asseguram a qualidade do banco de dados. Para garantir a qualidade dos requisitos e do projeto, utilizamos técnicas de revisão. Para garantir a qualidade da construção, utilizamos os testes.

Os testes devem acontecer juntamente com o projeto e a construção do banco de dados, assim como em qualquer desenvolvimento de softwares aplicativos, utilizando também a abordagem interativa e incremental. Para qualquer revisão que seja feita no desenvolvimento do projeto, deve-se também realizar a verificação dos testes.

## Certificação do Banco de Dados

Quando estamos testando os resultados, estamos compreendendo os riscos de uso do banco de dados. Essa “compreensão” gera uma capacidade de comunicar esse risco para todas as pessoas que queiram utilizar o banco de dados. A certificação é essa capacidade de comunicar o risco a outros interessados no uso do banco de dados.

A certificação do banco de dados permite que um banco de dados seja reutilizável. A parte funcional da certificação consiste na documentação clara dos esquemas conceituais e físicos. Sem a compreensão de como o banco de dados funciona, ninguém poderá reutilizá-lo.

## Manutenção e Otimização do Banco de Dados

A manutenção e a otimização constituem o estágio final do ciclo de vida de um banco de dados. Mesmo assim, após colocar o banco de dados para funcionar, continuamos a ter um processo interativo e incremental, já que um banco de dados sofre muitas alterações ao longo de sua existência.

# CONCEITOS BÁSICOS

Antes de iniciarmos o aprendizado do projeto de banco de dados, iremos apresentar os conceitos básicos de banco de dados. Tais conceitos são necessários à compreensão do projeto de banco de dados, além de fornecer uma visão geral deste processo.

## Banco de Dados

Há muitas definições para bancos de dados. Cada autor costuma dar uma:

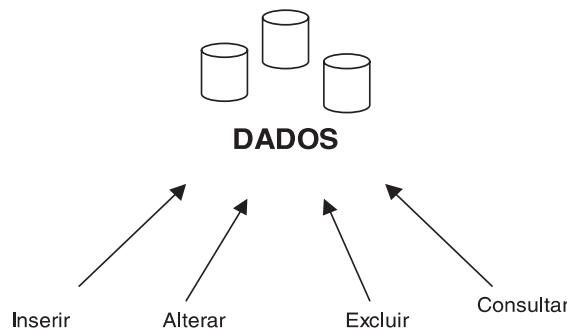
- Segundo *Palmer*, banco de dados é uma *coleção de dados, organizados e integrados*, que constitu-

em uma representação natural de dados, sem imposição de restrições ou modificações para todas as aplicações relevantes, sem duplicação de dados.

- Segundo *Date*, um banco de dados é uma *coleção de dados persistentes utilizados pelos sistemas de aplicação de uma determinada empresa*.

Há autores que fazem distinção entre Dado e Informação.

- Dado:** refere-se aos valores fisicamente registrados no banco de dados.
- Informação:** refere-se ao significado desses valores para determinado usuário.



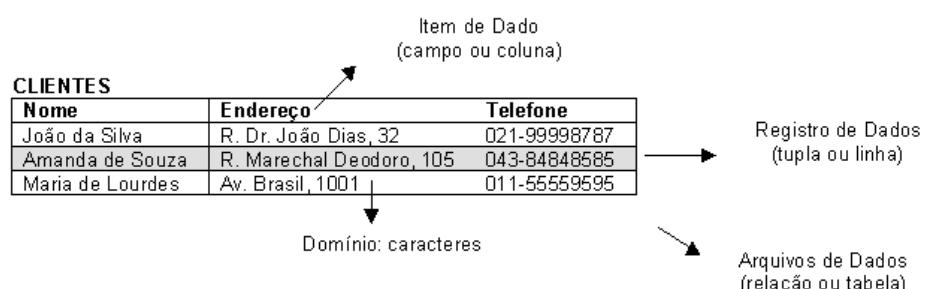
Em outras palavras, **banco de dados são dados armazenados, ou seja, são dados inter-relacionados, organizados e armazenados de tal forma que possibilitem facilmente sua manipulação: inserir, alterar, excluir e consultar**.

O Banco de Dados pode ser utilizado por sistemas aplicativos de uma única pessoa ou uma grande empresa.

Exemplos:

- Dados de controle de estoque de uma loja
- Dados de controle de uma faculdade

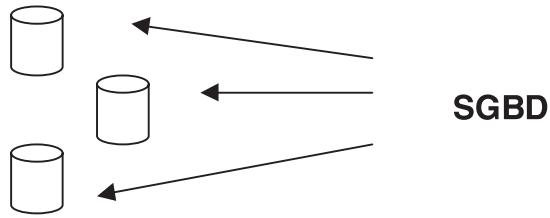
## Elementos Básicos de um Banco de Dados





## Sistema Gerenciador de Banco de Dados (SGBD)

É um software cujo objetivo é criar e manter um Banco de Dados.



### Objetivos de um SGBD

Um dos principais objetivos de um SGBD é manter um ambiente que seja adequado e eficiente para recuperar e armazenar dados provendo aos seus usuários uma visão abstrata dos mesmos.

Um Sistema Gerenciador de Banco de Dados possui os seguintes objetivos:

- Prover um **ambiente que suporte o armazenamento e consultas ao banco de dados**, independente do tamanho.
- **Controlar as requisições dos usuários** para acesso ao Banco de Dados.
- **Só alterar dados mediante solicitação do SGBD**.
- **Prover segurança das informações** contra quedas no sistema, acesso não-autorizado etc.
- **Fornecer linguagens de acesso aos dados** que facilitem a consulta ao Banco de Dados sem a necessidade de conhecer detalhes do armazenamento dos mesmos.
- **Facilitar a realização das operações básicas: inserir, recuperar, atualizar e excluir os dados**.

### Desvantagens de Sistemas de Arquivos

Não devemos confundir um SGBD - *Sistema Gerenciador de Banco de Dados* com a parte do Sistema Operacional chamada de Gerenciador de Arquivos. Veja algumas das desvantagens de um Gerenciador de Arquivo frente a um SGBD:

- **Redundância e inconsistência de dados**  
Arquivos com formatos diferentes, diferentes linguagens de programação, elementos de informação duplicados em diversos arquivos.
- **Dificuldade no acesso aos dados**  
Dados recuperados de forma inconveniente e inefficiente.
- **Isolamento de dados**  
Anomalias de acesso concorrente. Dados acessados por diferentes programas aplicativos, supervisão difícil.
- **Problemas de segurança**  
Difícil definição de visibilidade para usuários.

### • Problemas de integridade

Restrição de integridade nos valores dos atributos.

### Regras que um SGBD deve Possuir

Abaixo, são definidas algumas regras que um SGBD precisa possuir. Qualquer sistema de manipulação de dados que não possuir “todas” essas regras não pode ser definido como sendo um SGBD.

#### Regra 1: Auto-Contenção

Um SGBD contém os dados em si juntamente com toda a descrição dos seus dados, seus relacionamentos e formas de acesso, também chamado de Meta-Base de Dados. No caso de Gerenciadores de Arquivos, em algum momento, os programas aplicativos declaram estruturas (algo que ocorre tipicamente em C, COBOL e Pascal). Por exemplo, quando definimos a forma do registro no programa, não estamos trabalhando com um SGBD.

#### Regra 2: Independência dos Dados

Teremos “independência dos dados”, quando as aplicações estiverem realmente sem nenhuma definição de dados, ou seja, definições de dados ou estruturas de acesso nunca deverão estar contidos nos programas da aplicação.

Em Gerenciadores de Arquivos, quando criamos uma nova forma de acesso, um novo índice, precisamos alterar o código do aplicativo, o que difere de um SGBD.

#### Regra 3: Abstração dos Dados

Um SGBD autêntico deve fornecer ao usuário somente uma representação conceitual dos dados, ou seja, ele “esconde” maiores detalhes sobre sua forma de armazenamento real. Usamos a abstração de dados, chamada Modelo de Dados para fornecer esta representação conceitual. Neste modelo, um esquema das tabelas, seus relacionamentos e suas chaves de acesso são exibidos ao usuário, porém nada é afirmado sobre a criação dos índices, ou como serão mantidos.

Não estamos trabalhando com um SGBD se, por exemplo, inserirmos um pedido em um cliente inexistente e esta entrada não for “automaticamente” rejeitada.

#### Regra 4: Visões (Views)

Visualizar os dados de forma personalizada pelos usuários também é uma característica de um SGBD.

Uma visão consiste de um subconjunto de dados, necessariamente derivados dos existentes no Banco de Dados, porém estes não deverão estar explicitamente armazenados, ou seja, se precisamos replicar uma estrutura com a finalidade de criar um acesso de forma diferenciada por outros aplicativos, estamos trabalhando com Sistema de Arquivos.

#### Regra 5: Transações

Controlar os acessos múltiplos e simultâneos evitando assim atualizações incorretas. A integridade referencial

- **Visão Abstrata**

O sistema deve esconder detalhes físicos de armazenamento dos dados.

## Aspectos Funcionais de um SGBD

Um SGBD deve possuir Consistência de Dados, para isso o Sistema Gerenciador de Banco de Dados avalia os dados recebidos utilizando regras de integridade, garantindo assim que as características de Validade, Completeza e Consistência, sempre estejam corretas.

### Validade

É o conjunto de valores válidos para um determinado dado. Todo dado possui um domínio de valores. Os dados são válidos quando pertencem ao domínio de valores possíveis naquele caso.

Alguns Bancos de Dados fazem este teste, em outros é testado pelo aplicativo.

### Completeza

Compreende o preenchimento de todos os dados que são essenciais. Todos os dados que precisam ser conhecidos devem estar disponíveis quando necessário. Se eu tenho determinado dado na estrutura e esse dado é essencial para a minha aplicação se ele não for preenchido corretamente, eu não tenho completeza.

Geralmente, nos bancos de dados indicam-se quais os campos que podem ou não deixar de ser preenchidos.

### Consistência

É a principal regra da corretude. Sempre que a mesma informação é gravada, mesmo que em locais diferentes, ela deve possuir o mesmo valor.

Todas as informações do banco de dados devem ser coerentes umas com as outras, por exemplo:

- Se a idade de uma pessoa é armazenada em dois lugares diferentes no banco de dados, tenho que ter o controle para que a idade seja igual nos dois lugares.
- Quando tenho um campo que guarda a soma de outros vários campos, tenho que ter o controle para que o valor deste campo esteja sempre correspondendo à soma dos outros campo.
- Quando um campo de código de cliente for preenchido com um determinado número, por exemplo, 42, será que existe algum cliente cadastrado com o código 42?

*Integridade* significa a segurança de que os dados estarão sempre corretos e disponíveis para consulta.

Um banco de dados deve ser capaz de garantir a integridade dos dados e formas de recuperá-la se acontecer algum problema.

Quanto maior for a aplicação, o volume de dados manipulados e o número de pessoas que manipulam esses dados, o banco de dados se torna mais necessário.

definida pelo esquema do banco de dados deve ser completamente gerenciada pelo SGBD, sem nenhum auxílio do programa aplicativo, para isso, um SGBD deve possuir ao menos uma instrução que permita a gravação de uma série de modificações simultâneas e uma instrução capaz de cancelar esta série de modificações.

Por exemplo, ao cadastrar um pedido para um cliente que deseja reservar cinco itens de nosso estoque, sendo que os itens estão indisponíveis naquele momento e portanto, são efetivamente reservados. Porém, suponha que exista um bloqueio financeiro (por exemplo, uma duplicata do cliente em atraso) que impede a venda. Nesses casos, a transação deverá ser desfeita com apenas uma instrução ao Banco de Dados, sem precisar nenhuma modificação extra nos dados.

Em Sistemas de Arquivos, precisamos corrigir as reservas, através de acessos complementares, o que um SGBD não necessitaria.

### Regra 6: Acesso Compartilhado Automático

O sistema operacional, para evitar as alterações sobrepostas, só permite um acesso de cada vez, porém isso torna os sistemas de acesso aos dados muito inefficientes. Por exemplo, imagine se só um cliente de cada vez pudesse acessar o banco de dados de uma agência bancária. Isso não seria muito viável. Para corrigir esta situação, os SGBD permitem acesso concorrente compartilhado aos dados de um arquivo, totalmente gerenciado pelo mesmo para evitar que um sistema sobreponha suas alterações às de outro sistema.

Quando um recurso é compartilhado por mais de um sistema, o sistema operacional pode cair em uma situação chamada de deadlock, cuja responsabilidade deve ser do SGBD e não da aplicação, como no caso dos Gerenciadores de Arquivos.

### Requisitos de um SGBD

Um bom SGBD deve conter os seguintes requisitos:

- **Consistência de Dados**

A informação, mesmo que armazenada em diversos locais do banco de dados, as cópias devem ser sempre iguais.

- **Acesso aos Dados**

Deve ser rápido e preciso.

- **Acessos Concorrentes**

Permitir manipulação simultânea aos dados, garantindo a sua consistência.

- **Segurança**

Prover segurança no acesso aos dados através de grupos de usuários e senhas de acesso.

- **Integração**

Obedecer as regras de integridade no armazenamento dos dados.

- **Recuperação**

Recuperar o sistema da maneira mais rápida quando do acontecimento de falhas.



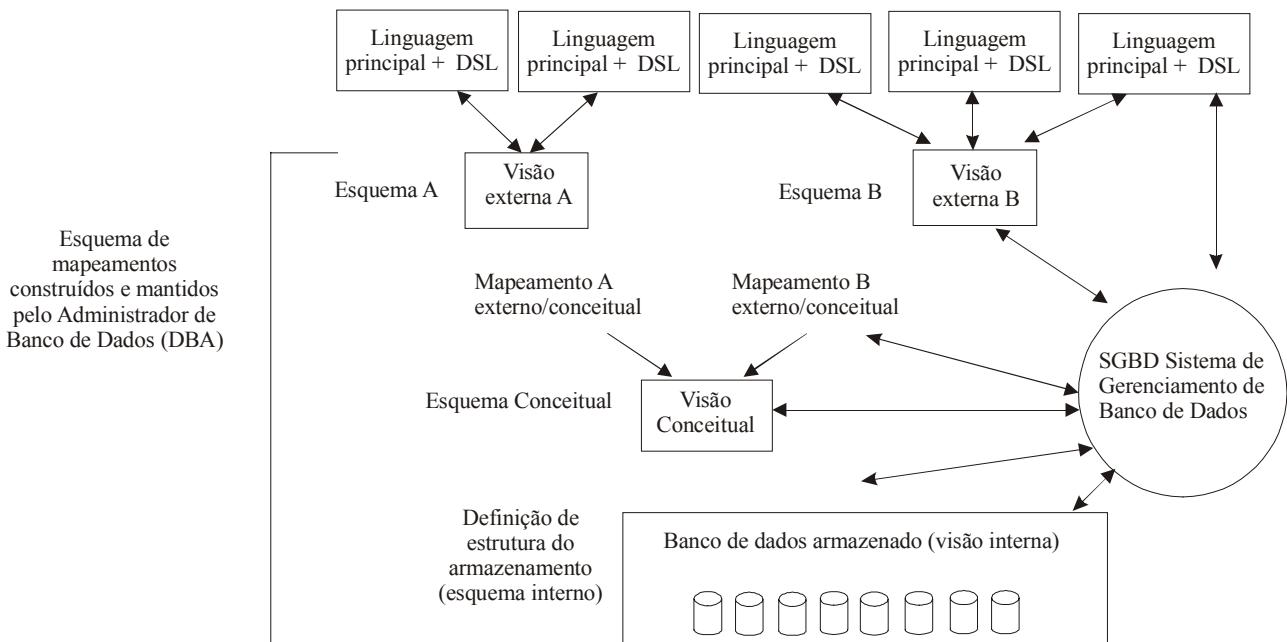
## Arquitetura de um SGBD

### Estrutura

Na estrutura de um Banco de Dados, temos:

- **Nível Interno**, onde é descrita a estrutura de armazenamento físico dos dados.
- **Nível Intermediário**, onde temos a descrição lógica dos dados.
- **Nível Externo**, onde são descritas as visões para grupos de usuários.

O Banco de Dados garante a Independência Lógica e Física dos Dados, portanto, podemos alterar o esquema conceitual dos dados, sem alterar as visões dos usuários ou mesmo alterar o esquema interno, sem tudo alterar seu esquema conceitual.



## Vantagens da Abordagem Banco de Dados

Já vimos as características que um SGBD possui e quais as suas vantagens com relação ao Sistema de Arquivos. Agora veremos quais as vantagens de um Sistema de Banco de Dados.

### Problemas no Uso de Arquivos

#### Alterações na Estrutura dos Dados

Os programas gravam seus dados em disco utilizando estruturas próprias, sendo necessário conhecer a sua estrutura para acessá-los.



Na manipulação de dados num sistema de arquivos, os programas dependem da estrutura dos dados, ou seja, se a estrutura for alterada através da inclusão ou remoção de campos, ou mesmo alterar o tamanho dos campos, isso afeta diretamente os programas. Para corrigirmos tal problema, os programas devem ser mo-

dificados para ficarem compatíveis com as mudanças realizadas nos arquivos de dados e, após essa atualização, temos que recompilar novamente os programas.

Se vários programas utilizam os mesmos dados, todos esses programas devem conhecer e manipular as mesmas estruturas.

#### PROGRAMA A

#### PROGRAMA B



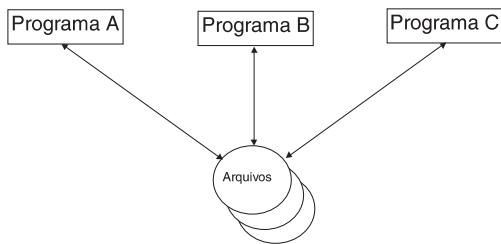
Se a estrutura dos dados for alterada, todos os programas precisarão também de alteração, mesmo que esta alteração ocorra em dados que ele não utiliza.

#### Controle de Acesso aos Dados

Na estrutura de arquivos, outro problema que temos é o acesso indiscriminado dos dados por todos os programas, este acesso descontrolado acaba trazendo complicações aos dados.

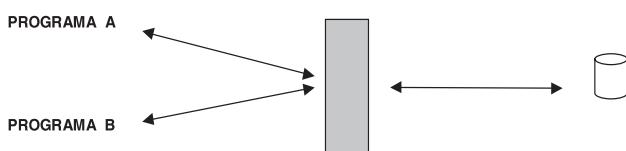
## INTRODUÇÃO A BANCO DE DADOS

- não precisa ser alterado quando as estruturas do banco de dados for modificada, mesmo se ele utilize ou não esses dados.



Num Sistema de Arquivos, temos diversos programas que acessam diretamente os dados, portanto dificilmente podemos manter uma segurança na sua permissão. Por outro lado, qualquer manutenção na estrutura desses afeta diretamente todos os programas de acesso.

Solução: colocar um sistema entre os dados e os programas, cuja função será de conversão do formato de gravação dos dados para o formato específico utilizado pelos programas.

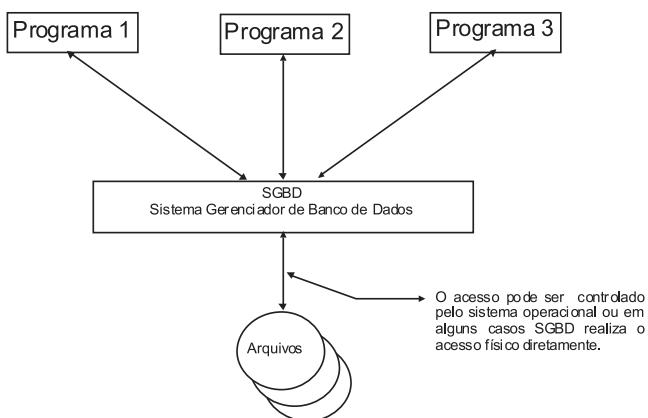


## Arquitetura de um Banco de Dados

Num Banco de Dados, o acesso aos dados é controlado por um SGBD, que, por sua vez, é controlado pelo Sistema Operacional. Porém, o SGBD pode realizar, se necessário, um acesso aos dados independente do sistema operacional.

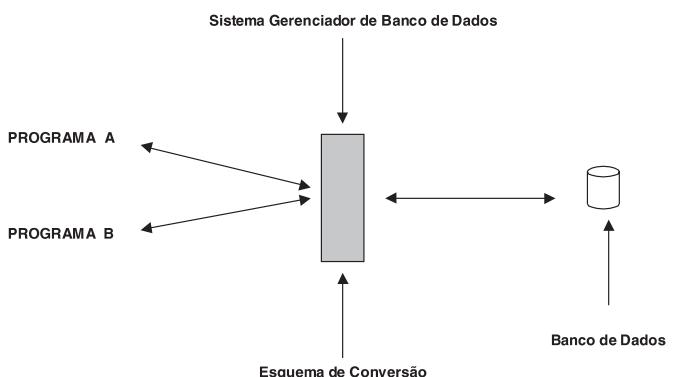
A **independência dos dados** é o maior benefício dos Bancos de Dados.

Como o acesso aos dados é controlado pelo gerenciador, não precisamos alterar os programas que não acessam aqueles que forem incluídos ou alterados nas tabelas do banco de dados. O próprio SGBD cuida em organizar a estrutura destes arquivos.



Cada programa:

- vê somente os dados que lhe interessam;
- não precisa saber como esses dados estão armazenados fisicamente no banco de dados;



## Comparação entre Sistemas de Arquivos e Banco de Dados

### Vantagens dos Sistemas de Arquivos

Num Sistema de Arquivos, cada usuário pode definir e implementar seus próprios arquivos através da tecnologia que domina e está disponível, o que normalmente é mais barato e não precisa de treinamentos. Enquanto no Banco de Dados o preço do produto é elevado, além de ser um sistema que requer aprendizado e gerenciamento.

### Vantagens dos Bancos de Dados

Num Sistema de Arquivos, cada usuário tem que definir e implementar os arquivos necessários para sua aplicação específica, ou seja, como não tem um padrão, o sistema de arquivos aceita, por exemplo, que o mesmo dado possa ser armazenado em formatos diferentes, através de programas escritos em linguagens diferentes e programadores diferentes.

Utilizando um SGBD, temos as vantagens de independência e consistência de dados. Retiramos dos programas aplicativos a complexidade de gerenciamento de estruturas de acesso aos dados.

O SGBD restringe e controla a autorização de acesso aos dados para os diferentes usuários, comprometendo assim a segurança e o isolamento dos dados.

Através do SGBD podemos facilmente proteger os dados contra perda através da utilização dos recursos de backup existentes no SGBD.

O controle centralizado dos dados facilita a utilização, atualização, manutenção da integridade, reorganização dos dados, redução de informações redundantes, eliminação de inconsistências entre os dados, compartilhamento dos dados, entre outros.



## Histórico dos Bancos de Dados

**1954** – Surgem os sistemas operacionais e as linguagens de alto nível.

**1958** – Comitê CODASYL – formado por universidades, fabricantes de computadores, clientes e governo, cujo objetivo era a definição de como manipular bases de dados garantindo integridade e independência de dados em relação ao hardware.

**1963** – Modelo de Rede – não chegou a ser lançado, devido às dificuldades de modelagem.

**1968** – Modelo Hierárquico – lançado o banco de dados hierárquico IMS (Information Management System) da IBM.

**1970** – Modelo Relacional – baseado na teoria de conjuntos. Garantia uma integridade matemática dos dados. Resultou de um trabalho de universidade individual de um pesquisador da IBM.

**1976** – Implementação do Modelo Relacional – surge o MER (Modelo Entidade-Relacionamento).

**1983** – DB2 – Banco de dados Relacional da IBM, na verdade, apenas uma casca relacional em cima do velho IMS.

**1988** – Modelo de Dados Orientados a Objetos – Considerado uma evolução natural do Modelo Entidade-Relacionamento.

## Exemplos de Bancos de Dados

- Sybase
- Dataflex
- Unify
- Oracle
- Gupta
- Access
- DBase
- FoxPro
- O2
- ObjectStore
- Informix
- DB2
- CA-OpenIngres
- ZIM
- Postgress
- SQL-Server
- Paradox
- Orion
- Jasmine
- E outros....

## Abstrações de Dados

Os usuários de um SGBD possuem visões abstratas dos dados, ou seja, não precisam saber como os dados são armazenados e mantidos.

## Mundo Real

É o mais alto nível de abstração. Os “objetos” do mundo real são seres, fatos, coisas e os organismos sociais. Podemos considerar, por exemplo, o departamento de uma empresa como algo do mundo real, pois é um organismo social, cabe ao projetista delimitar o que lhe interessa do mundo real para utilizá-lo na modelagem dos dados.

## Modelo Descritivo

É o nível das informações informais. Por exemplo, os relatórios escritos em uma linguagem natural (português, inglês etc.). Esse tipo de informação deve ser totalmente inteligível para as pessoas que interagem normalmente com ele, sem a necessidade de maiores conhecimentos.

## Modelo Conceitual

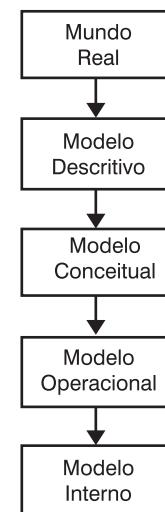
Modelo formal, baseado em símbolos. O modelo descritivo é também conceitual, porém só utilizamos essa denominação para os modelos de terceiro nível, ou seja, modelo conceitual formal.

## Modelo Operacional

Nível dos dados. Os dados devem ser expressos de tal forma que um computador os possa receber e tratar. Os modelos de dados do nível operacional podem ser: modelo relacional, de redes ou hierárquico.

## Modelo Interno

Último nível. É a representação interna dos dados e dos programas. Pode estar em linguagem de máquina (código objeto executável) ou em uma linguagem intermediária (códigos objetos interpretáveis).



Alguns autores costumam fazer outra divisão destes níveis de abstração:

### Nível Externo

É o mais alto nível de abstração, descrevendo apenas parte do banco de dados, é o nível do usuário individual. Cada usuário tem uma linguagem à sua disposição: para o programador de aplicação, pode ser uma linguagem convencional de programação; para o usuário final, pode ser uma linguagem de consulta ou uma linguagem de propósitos especiais, talvez baseada em formulários ou menu, modelada às necessidades do usuário e suportada por um programa de aplicação on-line.

O nível externo representa a visão externa, por exemplo, um usuário do departamento de pesso-



## Servir de Elo de Ligação com Usuários

É função do DBA servir de elo de ligação com os usuários, a fim de garantir a disponibilidade dos dados de que estes necessitam, e preparar - ou auxiliá-los na preparação dos necessários esquemas externos. Deve ainda ser definido o mapeamento entre qualquer esquema externo e o esquema conceitual.

Comunicar-se com os usuários, entender suas necessidades e identificar os dados a serem armazenados no banco de dados.

Conhecer as informações que precisam ser armazenadas de acordo com as atividades e necessidades da organização.

Escolher as estruturas apropriadas para representação e armazenamento desses dados.

Essas tarefas são realizadas antes mesmo da implementação do banco de dados.

## Administradores de Bancos de Dados (DBA)

O administrador de banco de dados, também chamado de DBA (Database Administrator), possui as seguintes responsabilidades:

### Criar a Base de Dados

Através do “esquema conceitual” criar a base de dados (Projeto Lógico), implementando as decisões tomadas pelo “Projetista de Banco de Dados”, também chamado de “Administrador de Dados”. A forma compilada daquele esquema é utilizada pelo SGDB para responder às solicitações de acesso. A forma não-compilada atua como documento de referência para os usuários do sistema.

### Decidir a Estrutura de Armazenamento e a Estratégia de Acesso

Desenvolver o “projeto físico” do banco de dados, estabelecendo a estrutura de armazenamento, método de acesso e como os dados serão representados internamente. Além disso, deve definir o mapeamento associado entre os níveis interno e conceitual.

Autorizar o acesso ao banco de dados, estabelecendo grupos de usuários e o tipo de acesso que cada grupo possui, tais como: leitura, atualização e exclusão, pela utilização de senhas de acesso.

### Definir os Controles de Segurança e Integridade

Solucionar problemas tais como: violação da segurança, restrições de integridade.

Realizar procedimentos de backup e recuperação quando necessário.

### Monitorar o Desempenho e Atender às Necessidades de Modificações

Atender às alterações de requisitos, monitoração de desempenho, tempo de resposta lento etc.

al pode ver o banco de dados como uma coleção de eventos de registros do departamento, e uma coleção de eventos de registros de empregados (podendo estar desinformado das ocorrências de registros de fornecedores e peças vistas pelos usuários do departamento de compra).

### • Nível Conceitual

É a visão do conjunto de usuários. O nível conceitual é a representação de todo o conteúdo de informações do banco de dados. De maneira geral, podemos dizer que a visão conceitual é a visão dos dados “como realmente são”, e não como os usuários são forçados a vê-los devido às restrições.

Neste nível, temos a visão do conteúdo total do banco de dados, ou seja, temos o esquema conceitual, que, neste caso, deve incluir uma grande quantidade de aspectos, como controles de segurança e de integridade.

### • Nível Interno

Sabemos como os dados estão realmente armazenados, são complexas estruturas de dados.

A visão interna é descrita por meio do esquema interno, que não só define os vários tipos de registros armazenados como também especifica os índices que existem, como os campos armazenados são representados, a seqüência física dos registros armazenados, e assim por diante.

## Usuários de um SGBD

Em um banco de dados de grande porte, há diversas pessoas responsáveis pelo projeto, uso e pela manutenção do ambiente do sistema de banco de dados.

A forma mais comum de interação Usuário e Banco de Dados dá-se através de sistemas específicos que, por sua vez, acessam o volume de informações geralmente pela linguagem SQL.

Resumidamente, os Administradores de Banco de Dados (DBA) são responsáveis pelo controle ao acesso aos dados e pela coordenação da utilização do BD. Já os projetistas de Banco de Dados (DBP) são analistas que identificam os dados a serem armazenados em um Banco de Dados e pela forma como estes serão representados.

Os Analistas e Programadores de Desenvolvimento criam sistemas que acessam os dados da forma necessária ao Usuário Final, que é aquele que interage diretamente com o Banco de Dados.

Dentre as pessoas responsáveis em manter um ambiente de banco de dados funcionando, temos:

## Projetistas de Bancos de Dados ou Administradores de Dados

Os projetistas de banco de dados possuem as seguintes responsabilidades:



## Definir a Estratégia de Reserva e Recuperação

Coordenar e monitorar a utilização do banco de dados, estabelecendo regras para manutenção e acesso aos dados. O DBA deve definir e implementar uma estratégia de recuperação apropriada envolvendo, por exemplo, o descarregamento periódico do banco de dados na memória auxiliar de armazenamento (backup) e procedimentos para recarregá-lo, quando necessário.

Adquirir recursos de hardware e software quando necessário.

Em grandes empresas, o DBA possui um grupo de profissionais (staff) que o ajudam a realizar essas funções, ou seja, comanda uma equipe que implementa suas decisões.

## Usuários Finais

A existência de um banco de dados se deve principalmente às pessoas (usuários finais) que possuem tarefas que envolvem acesso ao banco para realização de consultas, atualizações e geração de relatórios, por intermédio dos aplicativos desenvolvidos pelos programadores ou pelos recursos próprios contidos no SGBD.

## Analistas de Sistemas e Programadores de Aplicações (Engenheiros de Software)

Os **analistas de sistemas** possuem as seguintes responsabilidades:

- determinar as necessidades dos usuários finais
- desenvolver especificações para transações padronizadas, que atendam a essas necessidades.

Os **programadores de aplicações** possuem as seguintes responsabilidades:

- implementar as especificações (do analista de sistema) como programas.
- testar, retirar erros, documentar e fazer a manutenção dessas transações.

Os engenheiros de software (analistas e programadores) devem conhecer toda a capacidade do SGBD para poderem realizar as suas tarefas.

Em um banco de dados de pequeno porte, geralmente uma única pessoa define, constrói e manipula o banco de dados.

O SGBD localiza-se entre o Banco de Dados e os usuários do sistema e todas as requisições dos usuários para acessar o Banco de Dados são realizadas pelo SGBD.

Alguns exemplos de programas utilitários para auxiliar o DBA no desempenho destas tarefas são:

- **Rotinas de carga**

Para criar uma versão inicial do banco de dados.

- **Rotinas de cópia e recuperação**

Copiar o banco de dados e recarregá-lo (cópia de segurança).

- **Rotinas de reorganização**

Para agrupar os dados no banco por razões de desempenho (Ex.: regenerar espaço ocupado por dados que se tornaram obsoletos).

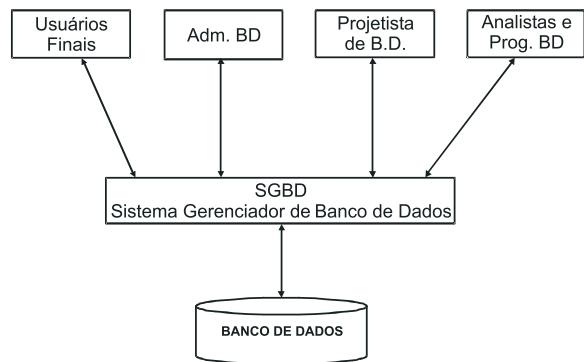
- **Rotinas estatísticas**

Relata estatísticas de desempenho, tamanhos de arquivos e distribuição de valores de dados.

- **Rotinas analíticas**

Para analisar as estatísticas geradas nas Rotinas estatísticas.

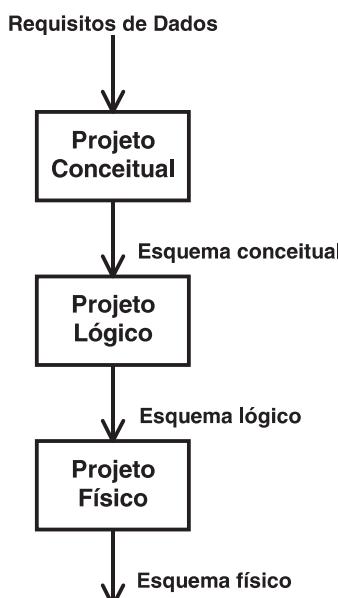
Uma das ferramentas mais importantes do DBA é o dicionário de dados (ou catálogo do sistema). Os diversos esquemas e mapeamentos (externo, conceitual etc.), especialmente, serão fisicamente armazenados no dicionário. Um dicionário abrangente também incluirá referências cruzadas das informações, mostrando, por exemplo, quais programas utilizam tal parte do banco de dados, quais departamentos necessitam de tais relatórios, quais terminais estão conectados ao sistema, e assim por diante.



# METODOLOGIA DE PROJETO DE BANCO DE DADOS

Um projeto de banco de dados é realizado em etapas, sendo que, em cada etapa, vamos considerando maiores detalhes sobre o banco de dados. Podemos dividir o projeto de um banco de dados em três partes:

- **Modelagem Conceitual** – modelo formal dos requisitos de um banco de dados.
- **Projeto Lógico** – definir as estruturas de dados que implementam os requisitos do modelo conceitual no SGBD escolhido.
- **Projeto Físico** – define os parâmetros físicos de acesso ao BD, buscando sua otimização.



## Modelagem Conceitual

Normalmente, a primeira etapa de um projeto de banco de dados consiste no desenvolvimento do esquema conceitual.

O esquema conceitual descreve o formato do conteúdo e a estrutura do banco de dados, exceto estruturas e estratégias de armazenamento, com isso é possível diferentes implementações do mesmo esquema, já que o seu desenvolvimento independe do SGBD.

Um modelo conceitual necessita ser mapeado em um esquema lógico baseado, por exemplo, no modelo relacional.

## Projeto Lógico

O termo “lógico” refere-se à possibilidade de se especificar propriedades do banco de dados que são permanentes, independentemente de qualquer situação particular, originam entradas no dicionário de dados. Até o início da década de 90, três diferentes modelos de dados foram considerados para a descrição de esquemas lógicos:

- **hierárquico**
- **em rede**
- **relacional**

## Projeto Físico

O nível físico para descrição de bancos de dados endereça assuntos de implementação específicos, entre os quais estão incluídos os métodos para acesso e gerência de dados, por exemplo, método seqüencial, indexado seqüencial ou direto; bem como a organização de arquivos.

Neste nível, descreve-se a forma como o sistema operacional deve tratar os dados do banco, incluindo a descrição dos arquivos (com os tamanhos e formatos dos campos), segmentos, blocos, buffers, ligações, ponteiros e técnicas de recuperação eficientes.

Esta disciplina tem como objetivo estudar as duas primeiras partes do ciclo de vida de um banco de dados, que são: modelagem conceitual e projeto lógico.



# MODELOS DE DADOS

O modelo de dados serve para esquematizar o banco de dados segundo uma estrutura padronizada, daí o nome de esquema de Banco de Dados.

Utilizamos um conjunto de conceitos para descrever um Banco de Dados, representando-o através de um Modelo de Dados. Não existe uma única forma de representação deste modelo, porém qualquer forma que permita a correta compreensão das estruturas de dados compreendidas no Banco de Dados pode ser considerada adequada. Vamos descrever sucintamente este modelo, pois estes serão objetos de estudo de outras disciplinas.

## Tipos de Modelos

### Modelo Orientado ao Registro

São modelos que representam esquematicamente as estruturas das tabelas de forma bastante próxima a existente fisicamente. Basicamente, são apresentados os re-

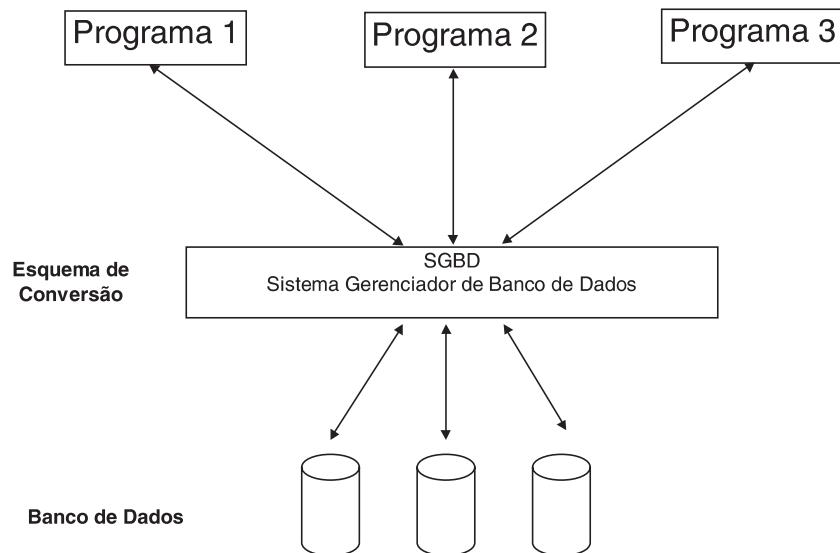
gistros de cada tabela (inclusive seus campos) e seus relacionamentos elementares. O Modelo Relacional, o Modelo de Rede e o Hierárquico são exemplos deste tipo de representação. Veja Apêndice A.

### Modelo Semântico

São modelos onde existe uma representação explícita das entidades e relacionamentos. Os **Modelos Entidade-Relacionamento** e o **Funcional** são exemplos deste tipo de abordagem.

### Modelo Orientado ao Objeto

São modelos que procuram representar as informações através dos conceitos típicos da Programação Orientada ao Objeto, utilizando o conceito de Classes que irão conter os objetos. Citamos os **Modelos O2** e o **de Representação de Objetos** como exemplos típicos desta abordagem.



## Esquema de Banco de Dados

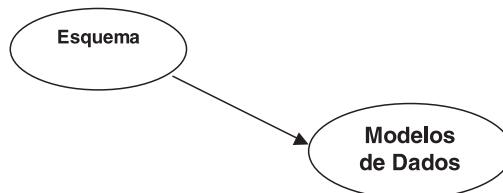
É uma forma de representar os dados seguindo uma estrutura padronizada para que o mesmo possa ser compreendido pelo SGBD utilizado.

**Exemplo:****Tabela de CLIENTES**

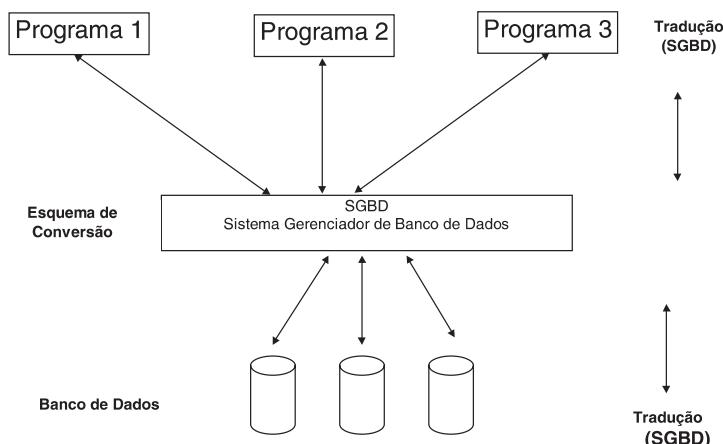
Nome	Endereço	Telefone
João da Silva	R. Dr. João Dias, 32	021-99998787
Amanda de Souza	R. Marechal Deodoro, 105	043-84848585
Maria de Lourdes	Av. Brasil, 1001	011-55559595

**Esquema**

Nome	Endereço	Telefone



Um Modelo de Dados possui vários elementos que permitem a representação através de propriedades semânticas e sintáticas pré-definidas. Utilizando essas propriedades disponíveis no modelo utilizado, podemos representar o “mundo real” definindo um “esquema de dados” que o gerenciador escolhido entenda.



Até o início da década de 90, os modelos de dados utilizados para descrição dos esquemas de dados foram: hierárquico, em rede e relacional. Veja Apêndice A para maiores informações sobre os esquemas de dados hierárquico e em rede.

Neste curso, iremos nos aprofundar no Modelo Relacional, que é um modelo mais pesquisado e usado em bases de dados.

No modelo de dados relacional, vemos os dados em tabelas (relações) que parecem simples, porém, possuem regras rigorosas para manter a integridade dos dados.

Portanto, não é uma “simples” tabela.

O conceito básico é o “template” de tabela, ou seja, um “modelo” (a relação = tabela). Essa “relação” possui um conjunto de campos, que é denominada “esquema”. O “esquema” consiste na estrutura do Banco de Dados. Cada tabela pode ser constituída por um conjunto de registros.

Um banco de dados relacional é constituído por um conjunto de tabelas, cada qual com colunas (atributos ou campos) e linhas de registros. Cada linha representa uma relação entre os atributos. Em cada tabela, geralmente, há um atributo que se denomina chave principal, que é usada para distinguir unicamente cada linha da tabela. Cada linha descreve um objeto.

Atributos têm domínios. Um domínio representa o conjunto de possíveis valores para aquele atributo. Algumas vezes, o tipo associado ao atributo, por exemplo, um inteiro ou uma data, estabelece um domínio de maneira implícita, podendo também serem definidas regras do negócio do banco de dados em questão.



# MODELAGEM CONCEITUAL UTILIZANDO A ABORDAGEM ENTIDADE-RELACIONAMENTO (ER)

A primeira etapa do projeto de um banco de dados é a construção do seu modelo.

A modelagem é uma forma abstrata de descrever os dados que serão armazenados no banco, independente da implementação no computador.

A abordagem Entidade-Relacionamento (ER), criada em 1976, por Peter Chen, é uma das técnicas mais utilizadas para descrever o modelo de dados.

O modelo de dados é representado por um modelo entidade-relacionamento (modelo ER), considerado um padrão de fato para a modelagem conceitual.

Graficamente, um modelo ER é representado pelo DER, ou seja, Diagrama Entidade-Relacionamento.

Os principais conceitos da abordagem ER são: entidade, relacionamento, atributo, generalização/especialização e entidade associativa.

A notação gráfica que iremos utilizar para diagramas ER é a notação original de Peter Chen. Iremos discutir também outras notações que são utilizadas para a construção de diagramas ER.

O Modelo Entidade-Relacionamento baseia-se em representar dados do “mundo real” através da definição de entidades e o relacionamento entre essas entidades.

## Entidade

### Definição

*Conjunto de elementos (objetos, ocorrências ou instâncias) do mundo real cujas informações se desejam guardar no banco de dados.*

O mundo real, de forma abrangente, contém diversas entidades, porém, quando estamos desenhando um modelo ER, temos que representar somente as entidades que interessam ao problema em questão.

### Exemplo

- Entidade Funcionário representa inúmeros funcionários.
- Entidade Departamento representa inúmeros departamentos.

## Terminologia Acadêmica

Os termos “Entidade” e “Conjunto de Entidades” correspondem respectivamente a:

- conjunto de entidades = conjunto de elementos (objetos). Ex.: Funcionário

- entidade = cada elemento (objeto) individual. Ex.: Maria

## Termos Utilizados no Projeto de BD

No Projeto de Banco de Dados, a definição de “Entidade” e “Conjunto de Entidades” é tratada de maneira diferenciada. Ao desenvolvermos um projeto de banco de dados sempre estaremos nos referindo ao conjunto de elementos, dificilmente a um elemento individual. Portanto, para não ficarmos “falando” conjunto de entidades a todo o momento, interpretamos uma entidade como um conjunto de elementos (objetos), por exemplo: “Funcionário” e nos referimos a uma ocorrência ou uma instância da entidade quando desejamos interpretar um elemento (objeto) individual, por exemplo: “Maria”.

## Notação



## Representação Gráfica

Representamos uma entidade por um retângulo que contém o nome da entidade, por exemplo:



A entidade pode representar um elemento concreto, como um Funcionário e também pode representar elementos abstratos, como um departamento.

Cada retângulo, ou seja, cada entidade representa um conjunto de elementos (objetos, ocorrências ou instâncias) do mundo real cujas informações se desejam guardar no banco de dados. Por exemplo: a entidade Funcionário representa um conjunto de todos os funcionários que se deseja guardar no banco de dados.

## Relacionamento

### Definição

Conjunto de associações entre entidades (objetos, ocorrências ou instâncias).

### Exemplo

Quais funcionários estão associados a quais departamentos em uma empresa.

## Terminologia Acadêmica

Os termos “Conjunto de Relacionamentos ou Instância de Relacionamento” e “Relacionamento” correspondem respectivamente a:

- Relacionamento = cada relacionamento (associação) individual.  
Ex.: Maria **trabalha na** Contabilidade
- Conjunto de Relacionamentos ou Instância de Relacionamento = conjunto de associações do mesmo tipo.  
Ex.: Funcionários **trabalham em** departamentos

## Termos utilizados no Projeto de BD

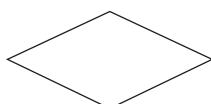
No Projeto de Banco de Dados, a definição de “Relacionamento” e “Ocorrência de Relacionamento ou Instância de Relacionamento” é tratada de maneira diferenciada. Ao desenvolvemos um projeto de banco de dados, sempre estaremos nos referindo ao conjunto de elementos e, dificilmente, a um elemento individual, portanto, para não ficarmos “falando” conjunto de relacionamentos a todo o momento, interpretamos um relacionamento como um conjunto de relacionamentos, por exemplo: “Funcionários trabalham em departamentos” e nos referimos a uma ocorrência ou uma instância de relacionamentos quando desejamos interpretar associações particulares dentro de um conjunto, por exemplo: “Maria trabalha na Contabilidade”.

Relacionamento = conjunto de associações do mesmo tipo.

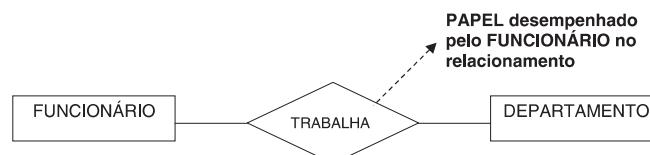
Ex.: Funcionários **trabalham em** departamentos

Ocorrência de Relacionamento ou Instância de Relacionamento = cada relacionamento (associação) individual. Ex.: Maria **trabalha na** Contabilidade

## Notação



## Representação Gráfica



## Informações Expressas no Modelo

- conjunto de funcionários (Entidade FUNCIONÁRIO)
- conjunto de departamentos (Entidade DEPARTAMENTO)
- conjunto de associações, onde cada associação liga um funcionário a um departamento (relacionamento TRABALHA)

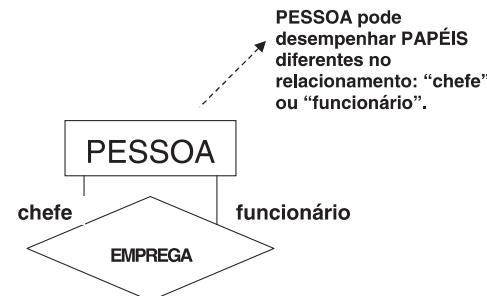
## Auto-Relacionamento

### Definição

*Relacionamento (conjunto de associações) entre instâncias (ocorrências) de uma mesma entidade, ou seja, entre uma entidade com ela mesma.*

Um relacionamento não precisa ser unicamente entre entidades diferentes, pode ser um relacionamento entre ocorrências de uma mesma entidade, o que chamamos de “auto-relacionamento”, como o exemplo a seguir:

## Representação Gráfica



No relacionamento emprega, uma instância de pessoa exerce o papel de chefe e a outra instância exerce o papel de funcionário.

## Cardinalidade de Relacionamento

Observe que até o momento o modelo não informa quantas vezes uma entidade pode ser associada a outra entidade através de um relacionamento. Para representarmos a quantidade de ocorrências associadas entre entidades, usamos o conceito de Cardinalidade de Relacionamentos.

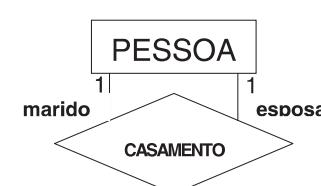
### Definição

Cardinalidade é o grau do relacionamento (grau de associação), ou seja, é o número de ocorrências, em que uma entidade pode estar associada a outra.

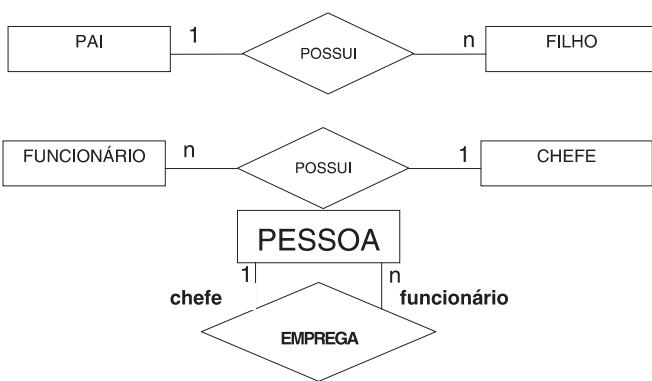
## Representação Gráfica

*A cardinalidade é representada por um número “1” ou “N”, sendo que o “N” representa vários e pode ser substituído por qualquer outra letra, como por exemplo, P, R, X etc.*

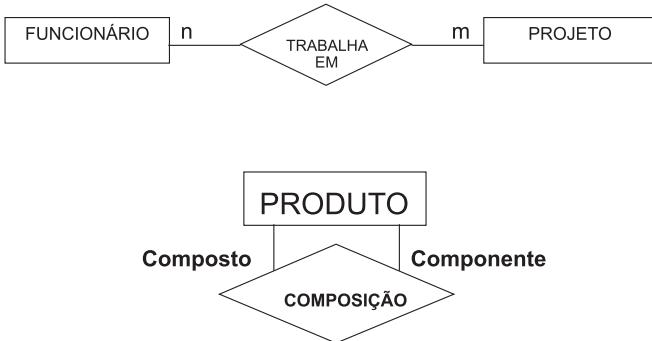
1:1



1:n , n:1



n:m



## Papel da Entidade no Relacionamento

### Definição

O papel da entidade em um relacionamento define que função uma instância da entidade cumpre dentro de uma instância do relacionamento, sendo extremamente importante quando se modelam “auto-relacionamentos”, ou seja, quando apenas uma entidade está envolvida.

### Tipos de Relacionamentos

Vimos, até o momento, relacionamentos binários, ou seja, relacionamentos que contêm duas ocorrências de entidade, inclusive os “auto-relacionamentos”, pois os mesmos, apesar de envolverem apenas uma entidade, também possuem duas ocorrências da mesma entidade.

A abordagem Entidade-Relacionamento permite, além de relacionamentos binários, também ternários, quaternários etc.

## Relacionamentos Binários

*Relacionamento (conjunto de associações) entre instâncias (ocorrências) de duas entidades.*

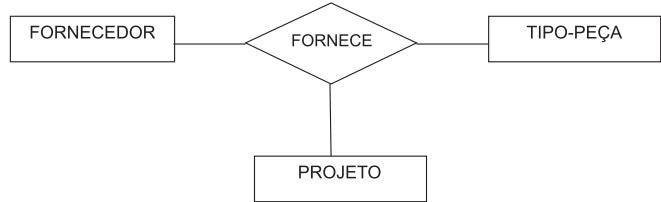
### Representação Gráfica



## Relacionamentos Ternários

*Relacionamento (conjunto de associações) entre instâncias (ocorrências) de três entidades.*

### Representação Gráfica

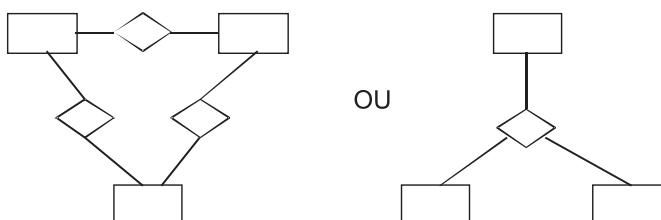


## Exercício

Dada a tabela abaixo:

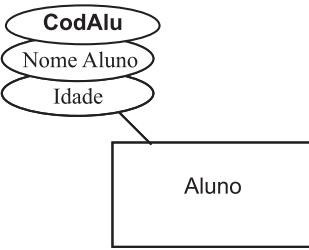
Fornecedor	Tipo-peça	Projeto
F1	TP1	PR1
F1	TP1	PR2
F1	TP2	PR1
F2	TP1	PR1
F2	TP2	PR2

Qual o melhor diagrama?



## Atributo

Cada entidade ou relacionamento pode conter propriedades de interesse, ou seja, informações que descrevem uma entidade ou relacionamento, como por exemplo: nome ou o salário do empregado.



## Definição

Propriedade ou característica de uma entidade.

### Exemplo:

Entidade Funcionário possui os atributos nome, idade, peso etc.

## Tipos de Atributos

### Simples

Ex.: endereço

### Composto

Ex.: endereço (rua, número, CEP)

### Monovalorado

Ex.: RG

### Multivalorado

Ex.: vários telefones por pessoa

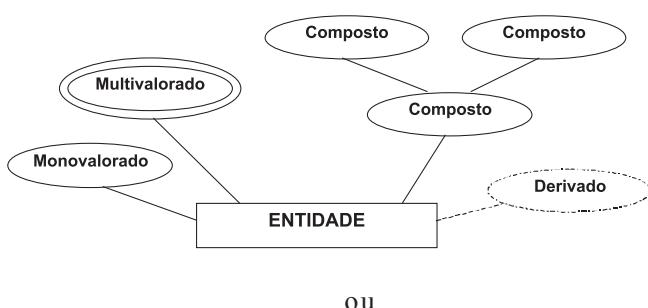
### Nulo

Ex.: valor desconhecido ou “não-aplicável”

### Derivado

Ex.: o atributo “idade” é derivado do atributo “data de nascimento”

### Notação



## Esquema Textual dos Atributos

A representação dos atributos pode sobrecarregar o Diagrama Entidade-Relacionamento, principalmente se as entidades possuírem muitos atributos. Portanto, ao invés de representar graficamente, representamos textualmente, separado do diagrama.

Quando utilizamos um software, uma ferramenta Case para construir os modelos, não precisamos nos preocupar com o armazenamento dos atributos, pois o próprio programa se encarrega de armazenar a lista de atributos para cada entidade no que chamamos de dicionário de dados.

## Cardinalidade de Atributos

Assim como os relacionamentos, os atributos também podem possuir cardinalidade.

## Definição

A cardinalidade do atributo define a quantidade de valores que podem estar associados a um elemento da entidade/relacionamento de que este atributo pertence.

## Domínio de um Atributo

## Definição

Conjunto de valores válidos para o atributo.

### Exemplo:

Cada atributo pertence a um domínio de valores, como por exemplo o domínio do atributo “meses do ano”: jan, fev, mar, abr, mai, jun, jul, ago, set, out, nov, dez.

## Instância

## Definição

Exemplo de uma ocorrência de uma entidade da vida real.

### Exemplo:

Instanciação da Entidade Funcionário, onde o funcionário é João, idade 30 anos e peso 70 kg.



## Estudos de Caso

### Administradora de Imóveis

Construa um diagrama ER (apenas entidades e relacionamentos com cardinalidades máximas) para a administradora de imóveis descrita abaixo:

A administradora trabalha tanto com administração de condomínios, quanto com a administração de aluguéis.

Uma entrevista com o gerente da administradora resultou nas seguintes informações:

- A administradora administra condomínios formados por unidades condominiais.
- Cada unidade condoninal é de propriedade de uma ou mais pessoas. Uma pessoa pode assumir diversas unidades.
- Cada unidade pode estar alugada para, no máximo, uma pessoa. Uma pessoa pode alugar diversas unidades.

\* estudo de caso retirado do livro de Projeto de Banco de Dados – CARLOS HEUSER

### Universidade – Parte 1

Construa um diagrama ER (apenas entidades e relacionamentos com cardinalidades máximas) para a universidade descrita a seguir:

Um banco de dados para uma universidade deve suportar os seguintes requisitos:

- Para um departamento, deseja-se manter seu número e nome.
- Para um orientador, armazenar seu código, nome e o número do departamento ao qual ele pertence.
- Para uma disciplina, armazenar o código da disciplina e o nome.
- Para um aluno, armazenar seu número e nome. Para cada disciplina que o aluno faz, armazenar o código da disciplina, o nome da disciplina e a média recebida. Além disso, armazenar o código e nome do orientador, para cada aluno.

\* estudo de caso retirado de um curso de especialização da UFSCar.

### Universidade – Parte 2

Dando continuidade ao modelo da universidade desenvolvido na parte 1, considere que um aluno pode estar em uma das três situações: estar cursando normalmente, estar com a matrícula trancada ou ser um aluno especial.

Para um aluno que está cursando normalmente, deseja-se manter informações sobre o nome de seu curso e o total de créditos já realizados.

Para um aluno com matrícula trancada deseja-se guardar a data de trancamento e o motivo.

Para um aluno especial, mantém-se a formação acadêmica, ano de conclusão, escola (todo aluno es-

pecial deve ser já formado em algum curso de nível superior; no caso de ser formado em mais do que um curso, somente um deles é armazenado).

Sabe-se que:

- Um aluno especial pode cursar qualquer disciplina oferecida e recebe um parecer sobre seu desempenho (ótimo, bom, regular etc.).
  - Somente os alunos que estão cursando normalmente possuem orientador.
  - Todo orientador é também um professor. Pode existir professor que não é orientador.
  - De um orientador, deseja-se guardar a área de atuação.
  - Um professor possui as informações: código, nome, endereço, categoria (auxiliar, assistente, adjunto).
  - Todo aluno especial deve ter um professor responsável por ele. Um professor pode ser responsável por vários alunos especiais.
- Eventualmente, um professor pode ser orientador de um aluno em situação normal e responsável por um aluno especial.
- \* estudo de caso retirado de um curso de especialização da UFSCar.

### Locadora de Vídeo

(adaptado do material de um curso de modelagem de dados Oracle)

Uma pequena locadora de vídeos possui ao redor de 2.000 fitas de vídeo, cujo empréstimo deve ser controlado.

Cada fita possui um número. Para cada filme, é necessário saber seu título e sua categoria (comédia, drama, aventura). Cada filme recebe um identificador próprio. Para cada fita é controlado que filme ela contém. Para cada filme há pelo menos uma fita, e cada fita contém somente um filme. Alguns poucos filmes necessitam duas fitas.

Os clientes podem desejar encontrar os filmes estrelados pelo seu ator predileto. Por isso, é necessário manter a informação dos atores que estrelam em cada filme. Nem todo filme possui estrelas. De cada ator, os clientes às vezes desejam saber o nome real, bem como a data de nascimento.

A locadora possui muitos clientes cadastrados. Somente clientes cadastrados podem alugar fitas. Para cada cliente, é necessário saber seu prenome e seu sobrenome, seu telefone e seu endereço. Além disso, cada cliente recebe um número de associado.

Finalmente, desejamos saber que fitas cada cliente tem emprestadas. Um cliente pode ter várias fitas em um instante no tempo. Não são mantidos registros históricos de aluguéis.

\* estudo de caso retirado do livro de Projeto de Banco de Dados - CARLOS HEUSER

## Céu

O céu é composto por moradores comuns, por anjos, por santos e, é claro, por Deus. Os anjos e santos desempenham funções particulares. Os anjos são divididos em anjos operários e anjos da guarda. Os anjos da guarda são alocados para olharem por mortais ainda na terra. Cada mortal tem dois anjos que o guardam, e cada anjo da guarda é alocado para apenas um mortal. Os anjos operários desempenham funções no céu, as quais são denominadas funções celestiais. Essas funções são alocadas para inúmeros anjos, enquanto um anjo pode ser responsável por inúmeras dessas tarefas. Os santos ficam o dia todo atendendo a pedidos provenientes dos mortais. Em algumas vezes, esses atendimentos são entendidos como milagres. Os moradores comuns passam o dia orando e se purificando, e têm a função de venerar santos e Deus por uma determinada quantia de horas por dia. Durante o restante do tempo, esses moradores descansam. Os dados que devem ser armazenados sobre os anjos são: cód\_anjo, cor\_asas, nome e idade; sobre os santos são: cód\_santo, cor\_vestes, tempo\_beatificação, nome e idade; sobre os moradores comuns: cód\_mor, grau\_luz e as horas que oram por dia. Além disso tudo, um anjo sempre é supervisionado por outros anjos, os quais pode supervisionar vários anjos. Sobre Deus, não se sabe muita coisa, e por isso atribui-se apenas um código para leitura.

\* estudo de caso retirado de um curso de especialização da UFSCar.

## Campeonato

A especificação refere-se ao controle de um campeonato de futebol. Participam do campeonato 24 equipes. Cada equipe possui um nome, nome de seu técnico, nome de seus 11 titulares, nome de seus 11 reservas, uniformes número 1 e 2, com a cor da camisa, das meias e do calção. Deve-se relacionar com cada equipe, as informações sobre a que país pertence. Cada país possui nome, continente, população, tamanho em km quadrados, renda-per capita e condição (país desenvolvido, em desenvolvimento ou subdesenvolvido). Devem ser guardadas as partidas, as equipes participantes, o placar, o nome do juiz principal, a localização do campo (cidade) e o nome do campo.

\* estudo de caso retirado de um curso de especialização da UFSCar.

## Administradora de Condomínios

Uma administradora deseja construir um banco de dados contendo informações sobre os condomínios sob sua responsabilidade.

Cada condomínio possui uma série de moradores (um por apartamento) que podem ser proprietários ou inquilinos. Alguns destes moradores possuem funções administrativas, tais como, síndico, sub-síndico ou conselheiro.

Cada condomínio possui um conjunto de funcionários, sendo que um deles é o chefe dos demais (o zelador).

Cada condomínio tem um histórico, mês a mês, das receitas (somente total). As receitas provêm das contribuições dos moradores, e neste sentido é importante registrar quem contribuiu e quem deixou de contribuir.

O condomínio também mantém um histórico sobre todas as despesas, que são registradas individualmente, mês a mês. As despesas correspondem a salários, taxas/impostos e serviços de manutenção. Os salários são pagos individualmente aos funcionários, sendo que não há funcionários ganhando o mesmo salário. As taxas/impostos são registradas em nome da empresa concessionária (Sabesp, Embratel, Comgás etc.) e têm um valor a ser pago e data de vencimento. Para os serviços de manutenção, registra-se a empresa, descrição do serviço, endereço da empresa, data de vencimento e valor a ser pago.

\* estudo de caso retirado de um curso de mestrado da USP.

## Cartório

Um Cartorio de Registro recebe documentos para registro. O apresentante do documento é recebido pela balconista, que registra o documento e o passa para o setor de conferência. Se o apresentante ainda não possuir cadastro no cartório, a balconista lhe entrega uma ficha para preenchimento com CPF, nome etc. Todos os setores do cartório são numerados para facilitar a identificação. O apresentante pode deixar pago um sinal, ou deixar para acertar no final do serviço. O apresentante recebe um comprovante de que ele deixou o documento no cartório. O documento recebe uma identificação, que é um número seqüencial. Ao documento são anexados alguns dados; além do número seqüencial: data de entrega do serviço, nome do apresentante, resumo do documento. Ao final da conferência, caso o documento esteja sem problemas, ele passa para o setor de registro, onde será microfilmado e receberá todos os carimbos necessários. Se, na conferência, estiver faltando alguma coisa no documento, ele é devolvido para o apresentante com uma carta explicando as providências que ele deve tomar para reapresentar o documento de novo. Depois do setor de registro, o documento passa para o setor de marcação, onde ele vai gerar vários registros no livro de índice. Após o setor de marcação, o documento passa pelo setor de conferência final, onde tudo que foi feito é conferido novamente e então entregue ao apresentante. Os apresentantes são avisados que o documento está pronto através do telefone (o cartório guarda o número de telefone de cada apresentante). O apresentante paga o serviço no caixa, se ainda não tiver pago. Cada funcionário pertence apenas a um setor, porém um setor pode ter vários funcionários. O Escrivão possui um controle da localização dos documentos dentro do cartório, que é atualizado pelos funcionários diariamente. Os funcionários possuem matrícula e nome. Este controle guarda em que setor está o documento.

## Locadora

A Locadora Fapi empresta fitas para os sócios. Os sócios assinam um recibo comprovando que pegaram a



fita. A Locadora guarda as seguintes informações a respeito dos seus sócios: código, telefone, nome e endereço. Os sócios alugam fitas e o preço do aluguel depende do filme alugado, por exemplo, “Xuxa e os Duendes” custa 1,00 real, já o filme “Harry Potter e a Câmera Secreta” custa 1,50 real. O aluguel pode ser pago quando se pega ou quando se devolve a fita. Ao ser entregue, a fita passa por testes rápidos para verificação de que não foi danificada. Na entrega, o sócio recebe um comprovante de que entregou a fita. Cada fita possui um número de série que já vem da distribuidora. As distribuidoras geralmente recebem pedidos de novas fitas quinzenalmente. Elas publicam uma listagem dos lançamentos também quinzenalmente. A Locadora mantém uma relação de distribuidoras sempre atualizada, com CGC, nome, endereço e telefone. A Locadora tem várias fitas de um mesmo filme, por exemplo: do filme “O Senhor dos Anéis”, possui duas fitas, a 39403 e a 38482. Para identificar o filme, o gerente criou uma numeração especial, por onde é possível saber o nome do filme e o preço do aluguel. Outro meio de encontrar o filme é através dos atores, atrizes e diretores: o gerente possui um cadastro de elenco com código, nome e foto dos atores, atrizes e diretores que participaram dos filmes. Outro meio de encontrar o filme é enquadrá-lo em um estilo: policial, ação, comédia etc. Existe uma tabelinha de estilos com sigla e a descrição. A Locadora pretende, a partir de agora, manter também uma relação atualizada dos estilos que cada sócio prefere. Afinal, cada sócio prefere alguns estilos em especial. Isto ajuda a enviar dicas para os sócios por mala direta.

### Loja de Peças

As Lojas de Peças atendem a pedidos dos clientes. Os pedidos são preenchidos pelo vendedor e são definidos por um número seqüencial. As lojas guardam as seguintes informações a respeito dos clientes: código, telefone, nome e endereço. O pedido é encaminhado para o departamento de faturamento. O pedido pode gerar uma nota fiscal. A nota fiscal é enviada ao comprador, que a paga em qualquer caixa. O pedido conterá as mercadorias que foram compradas. As mercadorias são entregues ao cliente no departamento de pacotes. A nota fiscal possui número e data de emissão. Quando a compra for a prazo, a nota fiscal desdobra-se em duplicatas. As duplicatas possuem um número de ordem que indica em quantas duplicatas desdobrou-se a nota fiscal. As duplicatas geralmente têm datas de vencimento diferenciadas e podem ter valores diferenciados também. O pa-

gamento das duplicatas será feito pelo cliente em bancos credenciados. Os bancos possuem nome e endereço. As mercadorias possuem código e descrição e são compradas de vários fornecedores. Cada loja mantém uma relação de fornecedores sempre atualizada, com CGC, nome, endereço e telefone.

### Agência Bancária

Uma agência bancária abre contas para clientes que tenham um bom volume de movimentação. Para o cliente abrir a conta, deve informar seu nome, o número de seu seguro social, endereço e CPF. Após o seu preenchimento, a ficha é enviada ao subgerente que faz um pedido de abertura de conta, que é enviado ao Conselho Diretor do banco. Se aprovada, a conta corrente receberá um número de identificação. Duas agências podem ter duas contas com mesmo número. A agência possui nome, cidade e código de identificação. A conta corrente passa a receber lançamentos de dois tipos (débito e crédito), sendo que estes lançamentos são identificados por um número seqüencial, data e valor do lançamento. O cliente pode requisitar extratos que conterão os lançamentos efetuados durante um certo período. Os cheques emitidos pelo cliente são recebidos no banco pelo departamento de compensação. A partir da compensação, eles se transformam em lançamentos comuns.

### Outro Planeta

No nosso planeta existem cinco tipos de seres: os xigort, as ritue, os diiwir, os queod e os diwpow. Um ritue pode sobrevoar vários queod. Cada ritue possui um pequeno mdfire. Um queod pode ser sobrevoado por vários ritue. Um diwpow é ouvido apenas por um queod. Os diiwir possuem um qzozz para se diferenciarem uns dos outros. Os Queod são codificados através dos aosjcked. As características mais marcantes dos queod são: saidf, seoir, saodf e qldoid. Um que, às vezes, casase com outro queod, deve-se guardar a data do casamento. Os xigort são identificados pelos seus ajarei. Os diwpow são muito tímidos e possuem um aoidi que os diferencia. Um diiwir pode ser morto por varias ritue. As ritue são chamadas pelas suas ejmrdic. Cada diwpow possuem zocoos e zoieids. As principais características dos xigort são seus aleri, asodfia e seus incríveis qpoeeed. Cada xigort pode cumprimentar várias ritue, porém uma ritue só pode ser cumprimentada por um xigort. Uma ritue mata vários diiwir. Um queod ouve atenciosamente apenas um diwpow.

# PROJETO LÓGICO UTILIZANDO A ABORDAGEM RELACIONAL

A segunda etapa do projeto de um banco de dados é o que chamamos de modelo lógico ou projeto lógico, na qual utilizaremos a abordagem relacional.

A abordagem utilizada no projeto lógico depende do tipo de SGBD. Existem diversos tipos de SGBD no mercado, tais como: relacional, hierárquica, redes e sistemas proprietários. Iremos utilizar a abordagem relacional que é a mais utilizada atualmente.

A abordagem Relacional é utilizada para descrever como um banco de dados relacional é organizado, ou seja, como as estruturas de dados são utilizadas e como elas se relacionam em um Sistema Gerenciador de Banco de Dados Relacional.

Os dados de um Modelo Relacional são vistos como armazenados em tabelas ou relações, daí o nome “relacional”.

## Tabelas

### Definição

Uma tabela é uma relação que contém linhas e colunas.

### Exemplo:

Coluna (atributo)			
nome do campo (nome do atributo)			
valor do campo (valor do atributo)			
Emp			linha (tupla)
CódigoEmp	Nome	CódigoDept	CategFuncional
E5	Souza	D1	C5
E3	Santos	D2	C5
E2	Silva	D1	C2
E1	Soares	D1	—

As linhas, também chamadas de “tuplas”, representam as instâncias com valores definidos entre um conjunto de valores, chamados de atributos ou campos.

Um atributo é identificado por um nome do atributo ou nome do campo e pode assumir um valor dentro de um conjunto de valores possíveis (domínio do atributo).

Vantagens e Diferenças entre uma tabela do banco de dados e um arquivo do sistema de arquivos de um computador:

- Ordenação:** Na tabela, as linhas não possuem ordem, somente se o comando de busca possuir

uma cláusula de ordenação. Em arquivos, o programador tem o controle sobre a ordem que se deseja no armazenamento, podendo, assim, recuperar registros em uma posição específica.

- Valores de Campo:** Na tabela, os valores são atômicos monovalorados, sendo que nos arquivos, os campos podem ser compostos por outros campos e esses campos podem ser multivalorados, através da utilização de arrays na linguagem Pascal por exemplo.
- Linguagens de Consulta:** Na tabela, podemos consultar a base de dados utilizando qualquer critério. Em arquivos, as consultas precisam de uma estrutura auxiliar. Para agilizar o processo, precisamos criar um caminho de acesso, utilizando índices e ponteiros, o que torna as consultas mais rápidas já que não precisam ler todos os registros de um arquivo. O caminho de acesso também existe em tabelas do banco de dados, porém não são visíveis aos usuários.

### Tipos de Chaves

Diferentemente do conceito de chave, em sistemas operacionais e arquivos convencionais, a chave é qualquer coluna utilizada para definir um índice ou qualquer outro tipo de estrutura de acesso, na abordagem relacional, não possui o mesmo significado. O significado de chave, na abordagem relacional, é definir apenas uma restrição de integridade dos dados, ou seja, é uma regra que deve ser obedecida pelo banco de dados.

As chaves, em um modelo relacional, identificam as linhas e criam relações entre as linhas das tabelas do banco de dados.

Utilizaremos as seguintes relações para explicar o conceito de chaves:

PESSOA = { código, nome, CPF, endereço, coddepto }

DEPARTAMENTO = { coddep, nomedep, chefe, salas }

### Super-Chave

A super-chave é qualquer conjunto de atributos de uma relação que identificam uma única tupla de uma relação.

Na tabela PESSOA, podemos definir as seguintes super-chaves:

```
{código, nome, CPF, endereço, coddepto}
{código, nome, CPF, endereço }
{código, nome, CPF, coddepto }
```



- .
- :
- {CPF, endereço }
- {código, nome, coddepto }
- {código, CPF }
- {código}
- {CPF }

Qualquer combinação de atributos onde apareçam os atributos código e CPF será super-chave, pois são eles que realmente identificam uma PESSOA.

## Chave

A chave é qualquer super-chave da qual eu não posso tirar nenhum atributo e ela continua sendo chave.

Na tabela PESSOA, a chave é: código ou CPF.

A chave define as super-chaves onde sobram apenas os atributos que realmente identificam as tuplas.

## Chave Primária

Chave Primária é a chave com a qual eu organizo minha base para ter maior velocidade de acesso, podendo ser composta por um ou mais atributos que identificam unicamente uma tupla ou linha de uma tabela.

A chave primária pode ser simples (um único atributo) ou a chave primária pode ser composta (dois ou mais atributos).

Na tabela PESSOA, a chave primária é simples: código

Dentre as “chaves”, escolhemos aquela que possui menor tamanho, possibilitando assim uma maior velocidade de acesso.

Para identificar as chaves primárias de uma tabela, iremos utilizar um grifo abaixo do atributo, como segue:

PESSOA = (código, nome, CPF, endereço, coddepto)

DEPARTAMENTO = (coddep, nomedep, chefe, salas)

A chave primária possui a restrição de integridade da unicidade de valores, ou seja, os valores das colunas que compõem a chave primária devem ser únicos em todo o banco de dados.

## Chave Candidata

Chave Candidata são as “outras chaves”, exceto a chave primária.

Neste exemplo, a chave candidata é: CPF

## Chave Estrangeira

A chave estrangeira é qualquer conjunto de atributos que são obrigatoriamente chaves primárias em outra tabela do banco de dados.

Neste exemplo, a chave estrangeira é o atributo coddepto da tabela PESSOA, pois o mesmo é chave primária em outra tabela, chamada DEPARTAMENTO.

É através deste conceito que implementamos os relacionamentos entre as tabelas do banco de dados.

## Restrições de Integridade

A integridade dos dados é de extrema importância para os bancos de dados, pois garante que os dados estejam íntegros e consistentes refletindo de maneira correta a realidade que foi modelada.

Para garantir consistência aos dados do banco de dados, usamos as restrições de integridade. Essas restrições devem ser garantidas automaticamente por qualquer SGBD relacional, não precisando ser definidas pelo programador.

A integridade dos dados é assegurada pelas restrições de integridades que são:

## Integridade de Domínio

Garantem que os valores aceitos para uma determinada coluna de uma tabela estejam dentro o domínio já pré-definido durante a sua criação.

## Integridade de Vazio

Garantem se preenchimento da coluna da tabela é obrigatório ou pode ser opcional, aceitando assim valores nulos, exceto para a coluna que é chave primária.

## Integridade de Chave

Garantem que os valores de chave primária e chave alternativa sejam únicos.

## Integridade Referencial

A integridade referencial deve ser imposta pelo banco de dados quando usamos uma chave estrangeira, garantindo assim que:

- Quando incluímos ou alterarmos uma linha na tabela que contém a chave estrangeira, esta deve existir como chave primária em outra tabela, ou na própria tabela, como no exemplo abaixo:

Emp	Nome	CodigoDept	CodigoEmpGerente
CódigoEmp			
E5	Souza	D1	—
E3	Santos	D2	E5
E2	Silva	D1	E5
E1	Soares	D1	E2

Chave estrangeira  
referencia chave primária na  
própria tabela

- Quando excluímos ou alterarmos o valor da chave primária, a coluna da chave estrangeira referenciada deve ser atualizada, ou seja, em caso de exclusão, sómente deve ser permitido se a chave primária a ser excluída não tiver nenhuma referência a nenhuma tabela do banco de dados e, em caso de alteração, o novo valor deve ser automaticamente atualizado para todas as suas referências no banco de dados.

## Restrições Semânticas

Existem certos tipos de integridades que não são asseguradas pelas restrições que estão implementadas automaticamente pelo SGBD, são as regras semânticas ou regras do negócio. Essas restrições dependem de cada negócio que estamos modelando, por exemplo:

Como garantir que um empregado só pode ser registrado no sistema se o mesmo possuir acima de 18 anos?

Como esse tipo de regra depende de cada negócio, as mesmas são implementadas por rotinas escritas pelo programador e não pela definição da base de dados.

## Domínios e Valores Vazios

Ao criarmos uma tabela no banco de dados, temos que determinar o tipo de cada atributo desta tabela, o que chamamos de “domínio da coluna” ou “domínio do campo” ou “domínio do atributo” e temos que especificar também se esses campos podem aceitar valores vazios, null ou nulos, o que indica que não precisa obrigatoriamente receber valores do seu “domínio” já definido.

As colunas “obrigatórias” precisam receber valores que pertençam ao seu domínio de valores pré-definidos.

As colunas “opcionais” podem permanecer com seus valores vazios ou nulos.

A única coluna que os Sistemas Gerenciadores de Bancos de Dados exigem que sejam sempre preenchidas é a chave primária, sendo que as demais chaves, incluindo as chaves estrangeiras, podem aceitar valores nulos.

## Especificação de Banco de Dados Relacional

A especificação de um banco de dados relacional é chamada de “esquema do banco de dados”, que contém pelo menos:

- tabelas;
- colunas das tabelas;
- restrições de integridade.

Existem diversas notações utilizadas para a definição do esquema, desde as mais resumidas até aquelas que abrangem maiores detalhes.

Em nosso curso, iremos adotar a seguinte notação durante a especificação dos esquemas dos bancos de dados:

Departamento(código, nome)  
Empregado(código, nome, endereço, CodDepartamento)  
CodDepartamento referencia Departamento  
Regras utilizadas nesta notação

Para cada tabela, colocamos o nome dos atributos entre parênteses.

- Os atributos-chave são sublinhados.
- Os atributos que são chaves estrangeiras são especificados logo abaixo da definição da tabela e seguem o seguinte padrão: NomeColuna referencia NomeTabela ou (NomeColuna1, NomeColuna2) referência NomeTabela quando a chave estrangeira for composta.

## Consultas à Base de Relacional

SQL é a linguagem padrão de definição e manipulação do banco de dados relacional. Os comandos para aplicação desta linguagem serão vistos em disciplinas posteriores.

Algumas vantagens das instruções SQL:

- O programador não precisa definir caminhos de acesso, pois os mesmos são decididos automaticamente pelo SGBD.
- Quando temos que consultar mais do que uma tabela ao mesmo tempo, a ligação entre as duas tabelas é feita automaticamente pela operação de junção, também chamada de “join”, que simplesmente iguala os valores dos campos associados.

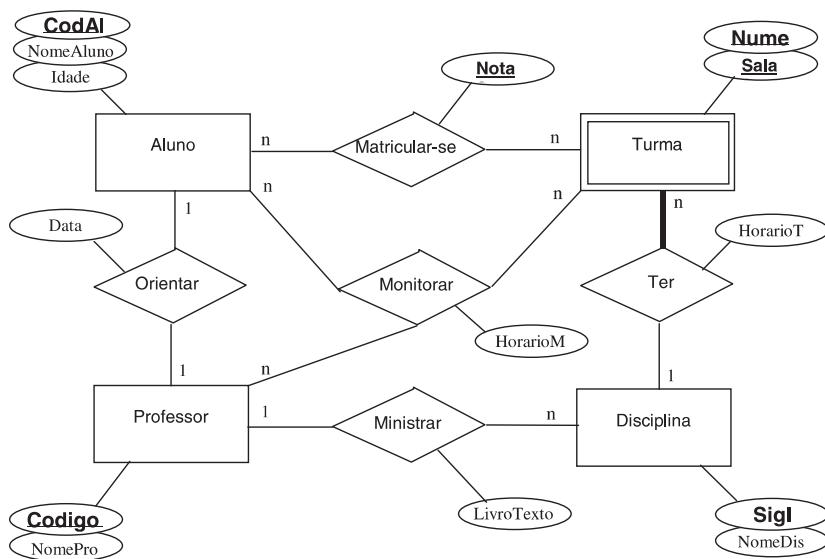


# MAPEAMENTO DO MER PARA O MODELO RELACIONAL

Este mapeamento pode ser realizado por um computador.

Ele é feito pelo banco de dados ZIM (o único banco de dados baseado no modelo Entidade-Relacionamento).

O DER abaixo será utilizado para ilustrar os passos do mapeamento:



## 1º Passo

Mapear as entidades regulares (normais).

Cada entidade é mapeada como uma relação que tem como chave primária a chave da entidade, e como atributos primos, os demais atributos.

ALUNO = {CodAlu, NomeAluno, Idade, CódigoPro, Data}

PROFESSOR = {CodPro, NomePro}

DISCIPLINA = {Sigla, NomeDis}

## 2º Passo

Mapear as entidades fracas.

Cada entidade fraca é mapeada como uma relação que tem como chave primária a chave da entidade fraca mais as chaves de todas as relações que mapeiam as entidades que participam dos relacionamentos que tornam a entidade fraca.

Os atributos primos serão os demais atributos da entidade fraca.

TURMA = {Número, Sigla, Salas}

Sigla referencia DISCIPLINA

## 3º Passo

Mapear os relacionamentos binários de cardinalidade 1:1.

Escolhe-se uma das relações que mapeiam as entidades que participam do relacionamento, e incluem-se como **atributos primos** a chave da outra relação e os atributos do relacionamento.

ALUNO = {CodAlu, NomeAluno, Idade, CódigoPro, Data}

CódigoPro referencia PROFESSOR

## 4º Passo

Mapear os relacionamentos binários de cardinalidade 1:N.

Toma-se a relação que mapeia a entidade que participa do relacionamento com cardinalidade N, e incluem-se como **atributos primos** a chave da outra relação e os atributos do relacionamento.

DISCIPLINA = {Sigla, NomeDis, CódigoPro, LivroTexto}

CódigoPro referencia PROFESSOR

TURMA = {Número, Sigla, Salas, HorarioT}

Sigla referencia DISCIPLINA

## 5º Passo

Mapear os relacionamentos binários de cardinalidade N:N.

Para cada relacionamento N:N cria-se uma relação que tem como chave a soma das chaves das relações

## INTRODUÇÃO A BANCO DE DADOS

das entidades que participam do relacionamento, e como atributos primos os atributos do relacionamento.

MATRICULAR = {CodAlu, Número, Sigla, Notas}  
 CodAlu referencia ALUNO  
 (Número, Sigla) referencia TURMA

### 6º Passo

Mapear os atributos multivalorados.

Cada atributo multivalorado cria uma nova relação, que tem a chave primária a chave da relação onde se encontrava o atributo multivalorado mais a chave da “peça” onde o atributo está ligado e o atributo como monovalorado primo.

O atributo Notas da relação MATRICULAR é multivalorado. A primeira providência é eliminá-lo da relação MATRICULAR:

MATRICULAR = {CodAlu, Número, Sigla}  
 CodAlu referencia ALUNO  
 Número referencia TURMA

Depois, cria-se uma nova relação para armazenar a informação anteriormente guardada no atributo Notas:

NOTAS = {CodAlu, Número, Sigla, NúmerodaProva, Nota}

(CodAlu, Número, Sigla) referencia MATRICULAR

O atributo Salas da relação TURMA é multivalorado. A primeira providência é eliminá-lo da relação TURMA:

TURMA = {Número, Sigla, HorarioT}  
 Sigla referencia DISCIPLINA

Depois, cria-se uma nova relação para armazenar a informação anteriormente guardada no atributo Salas:

SALAS = {Número, Sigla, NúmerodaSala}  
 (Número, Sigla) referencia TURMA

### 7º Passo

Mapear os relacionamentos ternários e quaternários.

Para cada relacionamento ternário ou quaternário, cria-se uma relação que tem como chave a soma das chaves das relações das entidades que participam do relacionamento, e como atributos primos os atributos do relacionamento.

MONITORAR = {CodAlu, CodPro, Número, Sigla, HorarioM}

CodAlu referencia ALUNO  
 CodPro referencia PROFESSOR  
 (Número, Sigla) referencia TURMA

## Esquema Relacional

As relações resultantes do mapeamento serão:

PROFESSOR = {CodPro, NomePro}

ALUNO = {CodAlu, NomeAluno, Idade, CódigoPro, Data}

CódigoPro referencia PROFESSOR  
 DISCIPLINA = {Sigla, NomeDis, CódigoPro, LivroTexto}

CódigoPro referencia PROFESSOR  
 MATRICULAR = {CodAlu, Número, Sigla}  
 CodAlu referencia ALUNO  
 (Número, Sigla) referencia TURMA

NOTAS = {CodAlu, Número, Sigla, NúmerodaProva, Nota}

(CodAlu, Número, Sigla) referencia MATRICULAR  
 TURMA = {Número, Sigla, HorarioT}  
 Sigla referencia DISCIPLINA  
 SALAS = {Número, Sigla, NúmerodaSala}  
 (Número, Sigla) referencia TURMA  
 MONITORAR = {CodAlu, CodPro, Número, Sigla, HorarioM}

CodAlu referencia ALUNO

CodPro referencia PROFESSOR

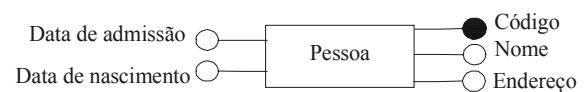
(Número, Sigla) referencia TURMA

**Exercícios**

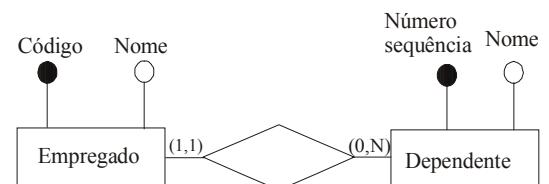
Faça o mapeamento dos seguintes MER:

\* MER retirado do livro de Projeto de Banco de Dados – CARLOS HEUSER

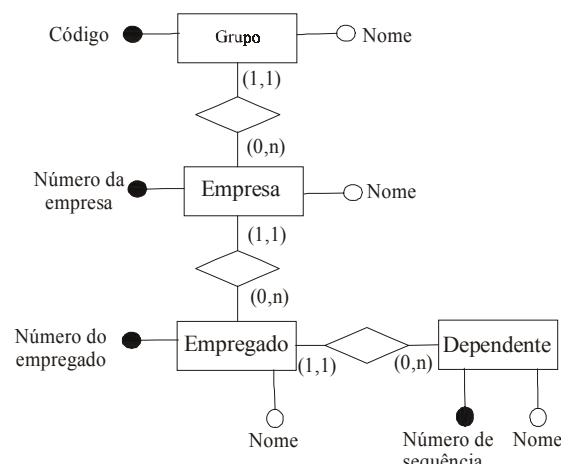
### Exercício 01



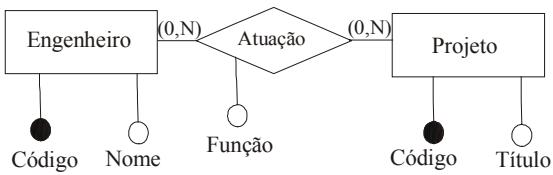
### Exercício 02



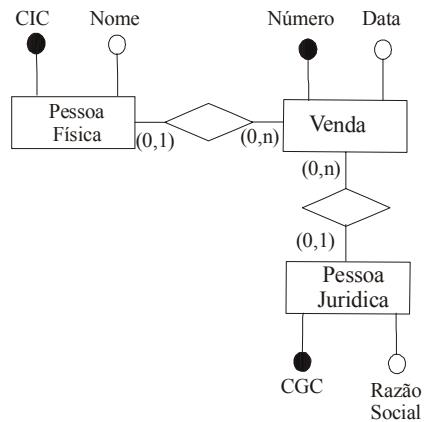
### Exercício 03



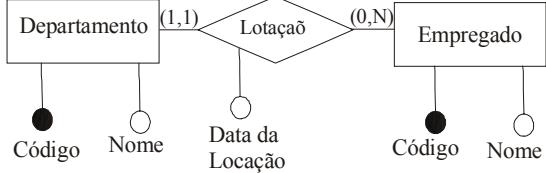
### Exercício 04



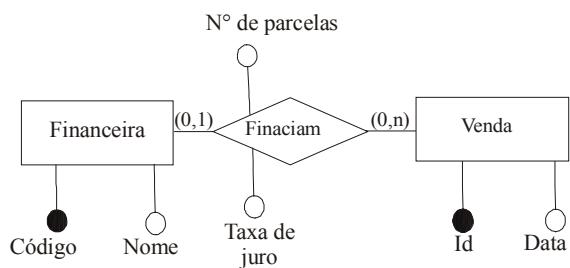
### Exercício 11



### Exercício 05



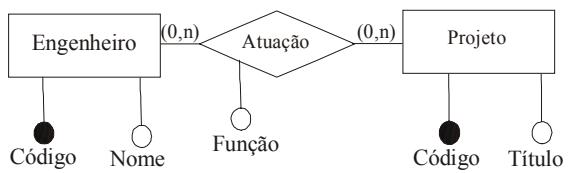
### Exercício 12



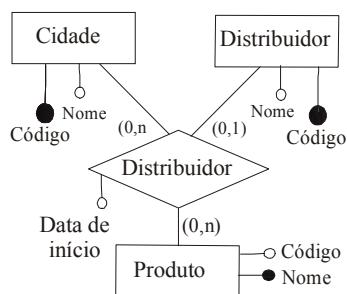
### Exercício 06



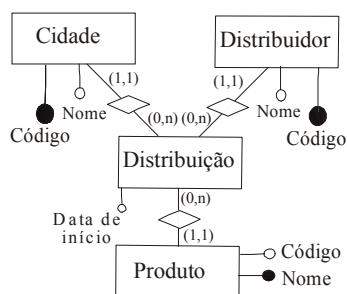
### Exercício 13



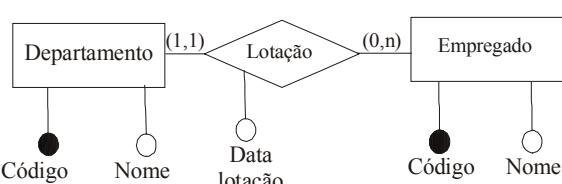
### Exercício 14

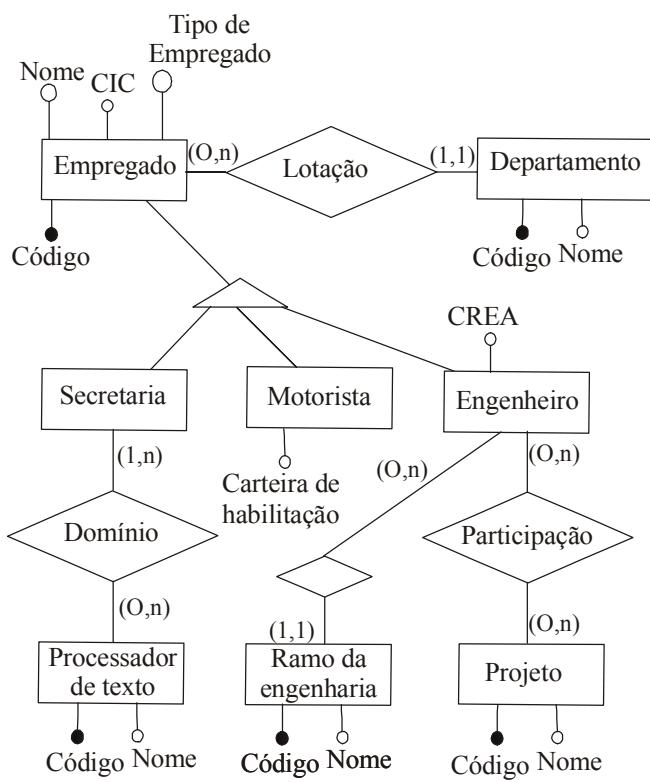
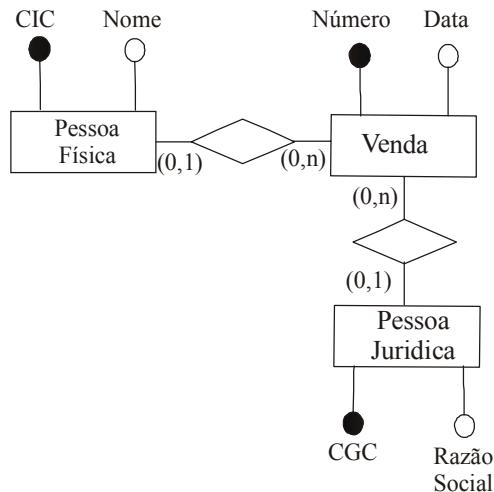
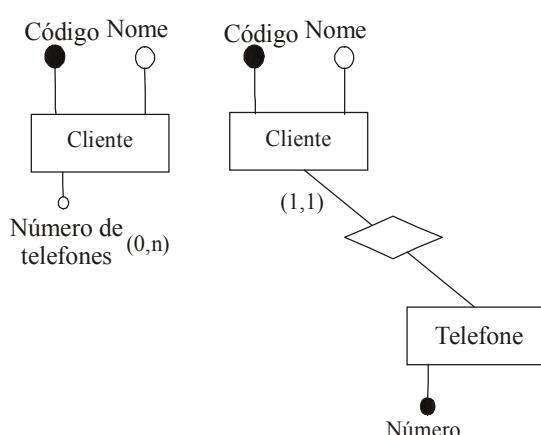
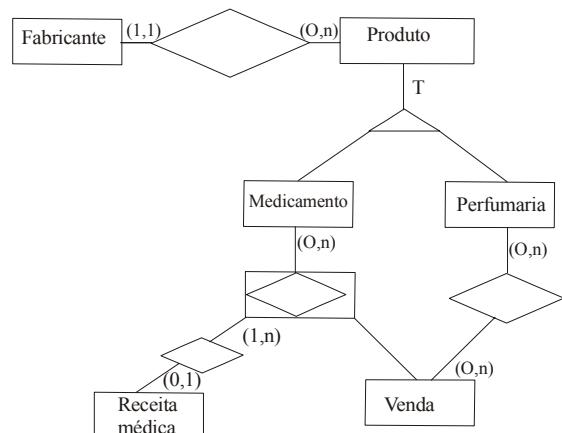


### Exercício 15



### Exercício 10

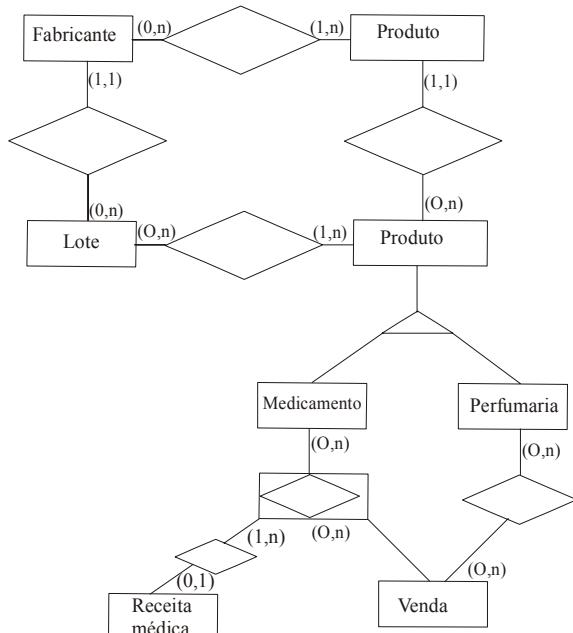


**Exercício 16****Exercício 17****Exercício 18****Exercício 19**

Considere os seguintes atributos para o MER acima?

Produto (Número, NomeComercial, TipoEmbalagem, Quantidade, PreçoUnitário)

Fabricante(CGC, Nome, Endereço)  
 Medicamento(Tarja, Fórmula)  
 Perfumaria(Tipo)  
 Venda(Data, NúmeroNota, NomeCliente, CidadeCliente)  
 PerfumariaVenda(Quantidade, Imposto)  
 MedicamentoReceitaVenda(Quantidade, Imposto)  
 ReceitaMédica(CRM, Número, Data)

**Exercício 20**

Considere os seguintes atributos para o MER acima?

Escritório(Número, Local)  
 Cliente(Número, CartMotorista, EstadoCar, Motorista, Nome, Endereço, Telefone)  
 Contrato aluguel(Número, Data, Duração)  
 Veículo(Número, DataPróximaManutenção, Placa)  
 Tipo de Veículo(Código, Nome, ArCondicionado)  
 Automóvel(Número de Portas, Direção Hidráulica, Câmbio Automático, Rádio)



## ENGENHARIA REVERSA DE MODELOS RELACIONAIS

O processo de engenharia reversa também pode ser considerado um processo de abstração de dados. A engenharia reversa parte de um modelo de implementação e resulta em um modelo conceitual. Sua finalidade é abstrair a especificação do banco de dados, ou seja, o modelo conceitual de um banco de dados já implementado.

A engenharia reversa de modelos relacionais é um caso específico da engenharia reversa de banco de dados, que consiste em um processo mais amplo de engenharia reversa de arquivos e documentos convencionais.

Nesta disciplina, daremos ênfase somente a este caso específico da engenharia reversa: engenharia reversa de modelos relacionais, ou seja, partiremos das tabelas do banco de dados e não de arquivos ou documentos, pois isso é tema que será abordado em outra disciplina do curso.

O ponto inicial de uma engenharia reversa de modelos relacionais é analisar o modelo lógico do banco de dados e gerar o modelo conceitual. Utilizaremos a abordagem relacional, a mesma abordagem adotada pelo processo de projeto de banco de dados utilizada nesta disciplina. Este é um processo inverso ao do projeto lógico, quando mapeamos o modelo conceitual para o projeto lógico.

A realização de um processo de engenharia reversa é útil nas seguintes situações:

- Quando não se tem o modelo conceitual do banco de dados devido, durante o seu desenvolvimento, a não aplicação de nenhuma metodologia para o projeto do banco de dados.
- Quando o esquema do banco de dados (projeto lógico), ou seja, as tabelas do banco de dados sofreram modificações durante o ciclo de vida do seu desenvolvimento e isso não foi sincronizado com o modelo conceitual, estando este desatualizado.

### Processo de Engenharia Reversa de um Modelo Relacional

Para concluir todo o processo de engenharia reversa de um modelo relacional, devem-se seguir os seguintes passos:

- **Passo 1:** Identificar a construção ER correspondente a cada tabela.
- **Passo 2:** Definir os relacionamentos 1:n e 1:1.
- **Passo 3:** Definir os atributos.
- **Passo 4:** Definir os identificadores de entidades e relacionamentos.

Iremos explicar cada passo utilizando um exemplo de um sistema acadêmico contendo o seguintes esquemas:

Relação de tabelas:

**Disciplina** (CodDisciplina, Nome)

**Curso**(CodCurso, Nome)

**Curriculum**(CodCurso, CodDisciplina, Obrigatoria/ Opcional)

CodCurso referencia Curso

*CodDisciplina referencia Disciplina*

**Prédio**(CodPredio, Endereco)

**Sala**(CodPredio, CodSala, Capacidade)

*CodPredio referencia Predio*

**Turma**(AnoSem, CodDisciplina, SiglaTurma, Capacidade, CodPredio, CodSala)

CodDisciplina referencia Disciplina

*(CodPredio, CodSala) referencia Sala*

**Laboratório**(CodPredio, CodSala, Equipamento)

*(CodPredio, CodSala) referencia Sala*

\* exemplo baseado no livro Projeto de Banco de Dados – CARLOS ALBERTO HEUSER

### Passo 1

#### Identificar a Construção ER Correspondente a cada Tabela

Uma tabela corresponde a:

- Entidade
- Relacionamento n:n
- Entidade especializada

Dica: Para determinar se a tabela corresponde a uma entidade, um relacionamento ou uma entidade especializada, analise a sua CHAVE PRIMÁRIA.

#### Regra 1: Quando a Chave Primária é Composta por Mais de uma Chave Estrangeira

Qualquer tabela que possuir a chave primária composta de diferentes chaves estrangeiras será um relacionamento n:n entre as 2 entidades referenciadas pela chave estrangeira. Ex.: CURRÍCULO que tem como chave primária CodCurso e CódDisciplina, ambas são referenciadas pelas tabelas Curso e Disciplina, respectivamente.

## INTRODUÇÃO A BANCO DE DADOS

- Para identificarmos o lado n da cardinalidade, temos que observar qual é a tabela que possui a chave estrangeira.

Exemplo – Temos que olhar para todas as tabelas, exceto para a tabela CURRÍCULO, já que esta já possui seu relacionamento definido n:n e a tabela LABORATÓRIO que é uma entidade especializada.

**Disciplina** (CodDisciplina, Nome)

**Curso**(CodCurso, Nome)

**Prédio**(CodPredio, Endereco)

**Sala**(CodPredio, CodSala, Capacidade)

*CodPredio referencia Predio*

**Turma**(AnoSem, CodDisciplina, SiglaTurma, Capacidade, CodPredio, CodSala)

*CodDisciplina referencia Disciplina*

*(CodPredio, CodSala) referencia Sala*

Comentários: As tabela que possui chave estrangeira são:

**Sala**(CodPredio, CodSala, Capacidade)

*CodPredio referencia Predio*

*A tabela Sala (lado n ou 1) possui um relacionamento com a tabela Predio (lado 1).*

**Turma**(AnoSem, CodDisciplina, SiglaTurma, Capacidade, CodPredio, CodSala)

*CodDisciplina referencia Disciplina*

*(CodPredio, CodSala) referencia Sala*

*A tabela Turma (lado n ou 1) possui um relacionamento com a tabela Disciplina (lado 1) e com a tabela Sala (lado 1).*

Como o esquema não informa se é 1:1 ou 1:n, temos que definir a cardinalidade correta da tabela Sala (lado 1 ou lado n) analisando os possíveis conteúdos do banco de dados.

Para analisar o que o banco de dados, neste exemplo, pode aceitar realizamos perguntas entre as entidades envolvidas.

Pergunta: O Prédio pode possuir 1 ou n salas? Se verificarmos o conteúdo do banco de dados, iremos concluir que um prédio possui n (várias) salas.

E a tabela Turma? Temos que fazer 2 perguntas já que a mesma possui 2 relacionamentos, com Disciplina e com Sala.

Pergunta 1: Uma disciplina pode ser ministrada para 1 ou mais turmas? Resposta: Uma disciplina pode ser ministrada para várias (n) turmas.

Pergunta 2: Uma sala pode ser utilizada por 1 ou mais turmas? Resposta: Uma sala pode ser utilizada por várias (n) turmas.

Segue a implementação do Passo 2:

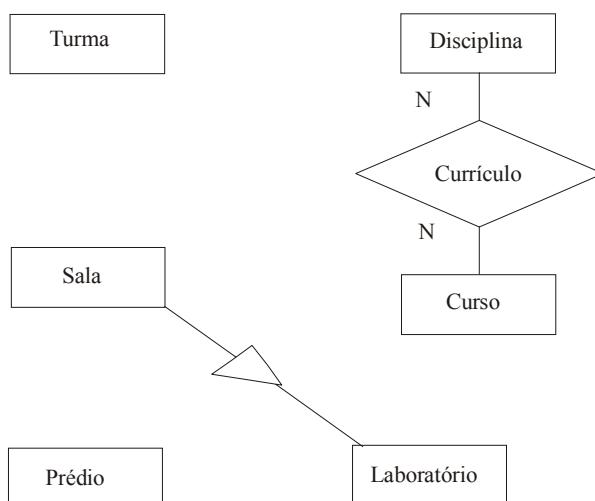
## Regra 2: Quando Toda a Chave Primária é uma Chave Estrangeira

Sempre que toda a chave primária é uma chave estrangeira temos uma entidade que forma uma especialização da outra tabela (entidade) que é referenciada pela chave estrangeira. Exemplo: LABORATÓRIO, que tem como chave primária (CodPredio, CodSala), ambas são chaves estrangeiras da tabela Salas. No Modelo Entidade-Relacionamento, isso só acontece quando temos uma especialização, neste caso LABORATÓRIO é uma especialização de SALAS.

## Regra 3: Demais Casos

Se a tabela não representa as regras 1 e 2, ou seja, não é nem relacionamento, nem entidade especializada, então, todas as demais tabelas são entidades. Exemplo: a tabela CURSO, possui a chave primária CodCurso, não possui chaves estrangeiras, portanto representa uma entidade. Da mesma maneira, as tabelas SALA, DISCIPLINA, PRÉDIO e TURMA não obedecem nem à regra 1 nem à regra 2.

Até o momento, podemos começar a desenhar o modelo conceitual, que segue:



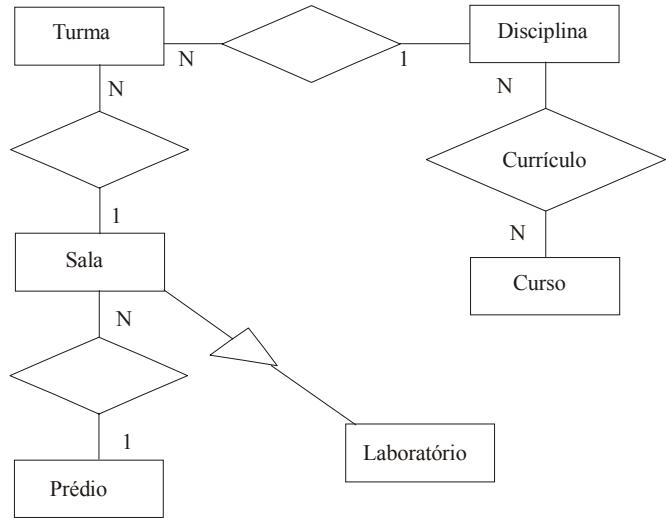
## Passo 2:

### Definir os Relacionamento 1:n e 1:1

**Regra:** Toda chave estrangeira, que não se enquadra nas regras 1 e 2 acima, ou seja, não representa um relacionamento nem uma entidade especializada, representa um relacionamento 1:1 ou 1:n.

### Dicas:

- Para definirmos a cardinalidade 1:1 ou 1:n, temos que verificar os possíveis conteúdos do banco de dados.

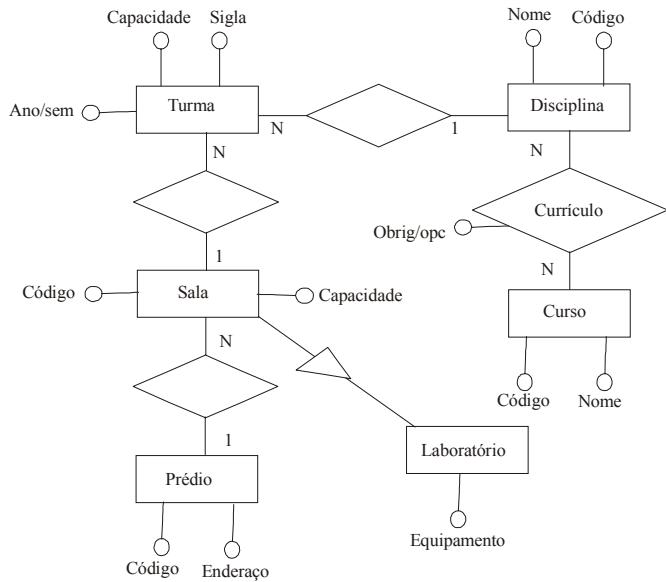


### Passo 3:

#### Definir os Atributos

Para cada coluna da tabela, **que não for chave estrangeira**, definimos como um atributo na entidade ou no relacionamento correspondente à tabela.

Exemplo: Nossa diagrama, neste passo, fica da seguinte maneira:



### Passo 4:

#### Definir os Identificadores de Entidades e Relacionamentos

Temos duas regras para definir os identificadores:

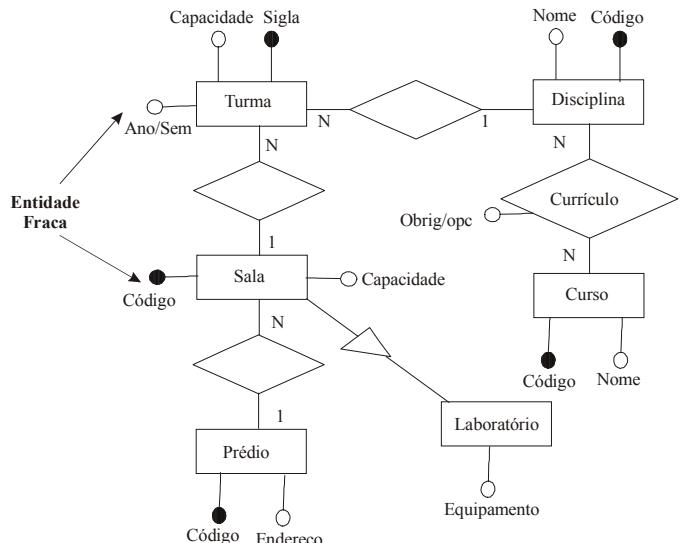
#### Regra A: A coluna da Chave Primária que Não é Chave Estrangeira

Corresponde a um atributo identificador da entidade ou do relacionamento.

#### Regra B: A Coluna da Chave Primária que é Chave Estrangeira

Corresponde a um atributo identificador externo da entidade (entidade fraca). Neste exemplo, temos a coluna CodDisciplina, que é parte da chave primária da tabela TURMA e também é chave estrangeira da tabela Disciplina.

O diagrama entidade-relacionamento completo a seguir:



#### Exercícios

1. Usando as regras de engenharia reversa, construa o diagrama ER para este banco de dados abaixo:

PEDIDO ( num , data, nome-cliente )

nome-cliente referencia CLIENTE

CLIENTE ( nome-cliente , end-cliente )

ITEM ( cod-item , descrição, preço-unit )

ENCOMENDA ( num , cód-item , quantidade )

num referencia PEDIDO

cód-item referencia ITEM

2. Usando as regras de transformação de modelos ER para modelo lógico relacional, projete um banco de dados relacional, ou seja, o projeto lógico, para o modelo Entidade Relacionamento abaixo. Para não sobrecarregar o diagrama entidade-relacionamento, os atributos estão listados abaixo e os atributos identificadores estão sublinhados.

Produto (Número, NomeComercial, TipoEbalagem, Quantidade, PreçoUnitario)

Fabricante (CGC, Nome, Endereco)

Medicamento (Tarja, Fórmula)

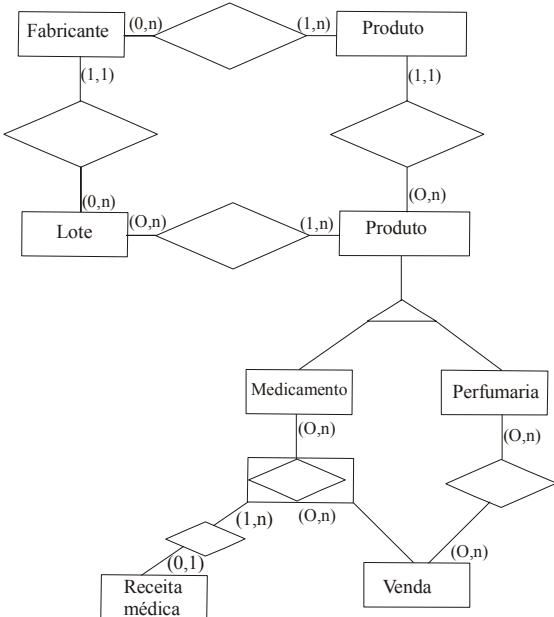
Perfumaria (Tipo)

Venda (Data, NúmeroNota, NomeCliente, CidadeCliente)

PerfumariaVenda (Quantidade, Imposto)

MedicamentoReceitaVenda(Quantidade, Imposto)

ReceitaMedica (CRM, Número, Data)



\* exercício retirado do livro de Projeto de Banco de Dados – CARLOS HEUSER

3. Usando as regras de engenharia reversa, construa o diagrama ER para este banco de dados que armazena dados sobre produtos e vendas em uma loja.

Produto (CodTipoProduto, NumProduto, Descrição, Preço)

/\* tabela de produtos de uma loja\*/

Similaridade (CodTipoProduto, NumProduto, CodTipoProdutoSimilar, NumProdutoSimilar)

(CodTipoProduto,NumProduto) referencia Produto  
(CodTipoProdutoSimilar, NumProdutoSimilar) referencia Produto

/\* tabela de similaridade de produtos, para cada produto informa quais seus produtos similares\*/  
TipoProduto(CodTipoProduto,DescriçãoTipoProduto)

/\* tabela de tipos de produtos \*/

Venda(NumNota, DataVenda, CodRegistradora, CodEmpregado)

CodRegistradora referencia Registradora

CodEmpregado referencia Empregado

/\* tabela que informa as vendas que ocorreram na loja \*/

ItemVenda (NumNota, CodTipoProduto, NumProduto, QuantItem, PrecoItem)

NumNota referencia Venda

(CodTipoProduto, NumProduto) referencia Produto

/\* tabela que informa os itens de uma venda, ou seja, que produtos, em que quantidade e com que preço foram vendidos em uma venda \*/

Registradora(CodRegistradora, SaldoRegistradora)

/\* tabela de dados de cada registradora da loja \*/

Empregado(CodEmpregado, NomeEmpregado, SenhaEmpregado)

## INTRODUÇÃO A BANCO DE DADOS

/\* tabela de dados de todos os empregados da loja \*/

\* exercício retirado do livro de Projeto de Banco de Dados – CARLOS HEUSER

4. Usando as regras de engenharia reversa, construa o diagrama ER para este banco de dados que armazena dados genealógicos.

Pessoa (PessoaID, Nome, LocalNascimentoID, DataNascimento, LocalFalecimentoID, DataFalecimento, ProfissãoID, FilhoCasamentoID, Sexo)

LocalNascimentoID referencia Local

LocalFalecimentoID referencia Local

ProfissãoID referencia Profissão

FilhoCasamentoID referencia Casamento

/\* tabela de pessoas \*/

Local(LocalID, Cidade, País)

LocalID referencia Local

/\* tabela de locais \*/

Profissão (ProfissãoID, Nome)

ProfissãoID referencia Profissão

/\* tabela de profissões \*/

Casamento (CasamentoID, PessoaMaridoID, PessoaEsposaID, DataCasamento, LocalCasamentoID)

PessoaMaridoID referencia Pessoa

PessoaEsposaID referencia Pessoa

LocalCasamentoID referencia Local

/\* tabela de casamentos \*/

\* exercício retirado do livro de Projeto de Banco de Dados – CARLOS HEUSER

## Respostas

### Exercício 1

Segue a resolução passo a passo:

#### Passo 1: Identificar a Construção ER Correspondente a cada Tabela

Uma tabela corresponde a:

- Entidade
- Relacionamento n:n
- Entidade especializada

Dica: Para determinar se a tabela corresponde a uma entidade, um relacionamento ou uma entidade especializada, analise a sua CHAVE PRIMÁRIA.

#### Regra 1: Quando a Chave Primária é Composta por Mais de uma Chave Estrangeira

Qualquer tabela que possui a chave primária composta de diferentes chaves estrangeiras será um relacionamento n:n entre as duas entidades referenciadas pela chave estrangeira. Ex.: ENCOMENDA tem como chave primária num e cód-item, ambas são referenciadas pelas tabelas Pedido e Item.

Exemplo – Com a aplicação desta regra, conseguimos montar o diagrama seguir:



### Regra 2: Quando a Chave Primária é Uma Chave Estrangeira

Sempre que toda a chave primária é uma chave estrangeira, temos uma entidade que forma uma especialização da outra tabela (entidade) que é referenciada pela chave estrangeira.

Neste exemplo, não temos nenhuma tabela deste tipo.

### Regra 3: Demais Casos

Se a tabela não representa as regras 1 e 2, ou seja, não é nem relacionamento, nem entidade especializada, então, todas as demais tabelas são entidades.

Exemplo – A única tabela que sobrou é a tabela de Cliente, que pode ser adicionada ao diagrama:



### Passo 2: Definir os Relacionamentos 1:n e 1:1

**Regra:** Toda chave estrangeira, que não se enquadra nas regras 1 e 2 acima, ou seja, não representa um relacionamento nem uma entidade especializada, representa um relacionamento 1:1 ou 1:n.

#### Dicas:

- Para definirmos a cardinalidade 1:1 ou 1:n, temos que verificar os possíveis conteúdos do banco de dados.
- Para identificarmos o lado n da cardinalidade, temos que observar qual é a tabela que possui a chave estrangeira.

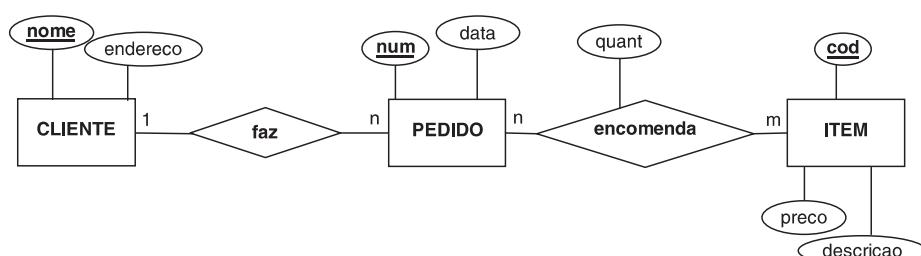
Exemplo – Temos que olhar para todas as tabelas, exceto para a tabela Encomenda, já que esta já possui seu relacionamento definido n:n.

PEDIDO ( num , data, nome-cliente )

nome-cliente referencia CLIENTE

CLIENTE ( nome-cliente , end-cliente )

O diagrama completo segue abaixo:



ITEM ( cod-item , descrição, preço-unit )

**Comentários:** A única tabela que possui chave estrangeira é a tabela PEDIDO, que se referencia à tabela CLIENTE, portanto, temos um relacionamento da tabela Pedido (lado n ou lado 1) com a tabela Cliente (lado 1).

Para definirmos a cardinalidade correta da tabela Pedido (lado 1 ou lado n), temos que fazer a seguinte pergunta: O cliente pode realizar 1 ou n Pedidos? Se verificarmos o conteúdo do banco de dados, iremos concluir que o cliente pode realizar n (vários) Pedidos.

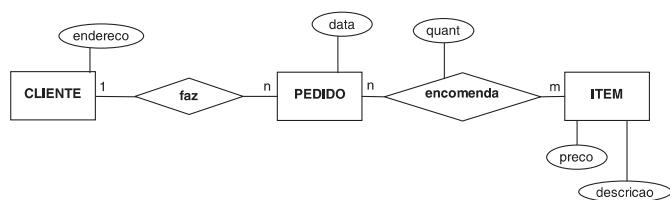
Segue a implementação do Passo 2:



### Passo 3: Definir os Atributos

Para cada coluna da tabela, que não for chave estrangeira, definimos como um atributo na entidade ou no relacionamento correspondente à tabela.

Exemplo: Nossa diagrama, neste passo, fica da seguinte maneira:



### Passo 4: Definir os Identificadores de Entidades e Relacionamentos

Temos duas regras para definir os identificadores:

**Regra A: A coluna da chave primária que não é chave estrangeira**

Corresponde a um atributo identificador da entidade ou do relacionamento.

**Regra B: A coluna da chave primária que é chave estrangeira**

Corresponde a um atributo identificador externo da entidade (entidade fraca). Nossos exemplos, não possui nenhuma tabela cuja coluna é chave primária e também chave estrangeira.

## Exercício 2:

Produto (CGC, NúmeroProd, NomeComercial, TipoEmbalagem, Quantidade, PrecoUnitario, TipoProd, Tarja, Fórmula, Tipo)

*CGC referencia Fabricante*

Fabricante (CGC, Nome, Endereco)

Venda (Data, NúmeroNota, NomeCliente, CidadeCliente)

PerfumariaVenda (CGC, NúmeroProd, NúmeroNota, Quantidade, Imposto)

*(CGC, NumeroProd) referencia Produto*

*NúmeroNota referencia Venda*

MedicamentoReceitaVenda (CGC, NúmeroProd, NúmeroNota, Quantidade, Imposto)

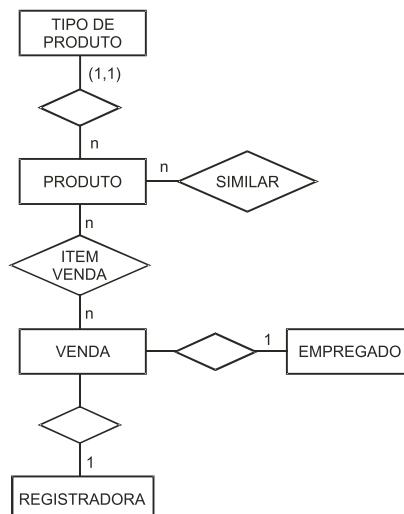
*(CGC, NumeroProd) referencia Produto*

*NúmeroNota referencia Venda*

*(CRM, Número) referencia ReceitaMédica*

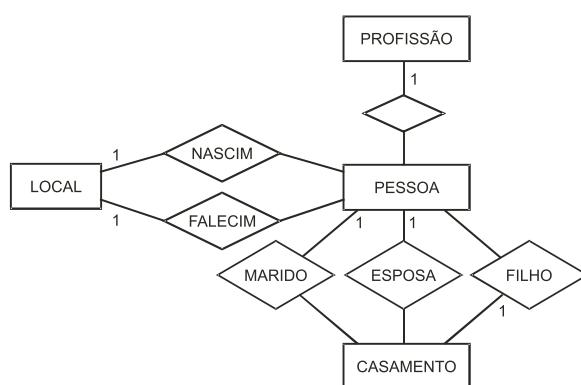
ReceitaMedica (CRM, Número, Data)

## Exercício 3:



Produto (NúmeroProd, DescricaoProd, PreçoProd)  
 TipoProd (CodigoTipoProd, DescricaoTipoProd)  
 Venda (NúmeroNF, DataVenda)  
 ItemVenda (QtdItem, PreçoItem)  
 Registradora (CodReg, SaldoReg)  
 Empregado (CodEmp, NomeEmp, SenhaEmp)

## Exercício 4:



Pessoa (PessID, PessNome, DataNasc, DataFalec, Sexo)  
 Local (LocID, Cidade, País)  
 Profissão (ProfID, ProfNome)  
 Casamento (CasamID, DataCasam)



# APÊNDICES

## Apêndice A - Modelos Hierárquicos e de Redes

Características dos modelos hierárquicos e de redes:

- Dados representados por coleções de registros (contêm dados e ligações físicas – ponteiros que representam as ligações entre os dados).
- Ambos são navegacionais. A partir de um registro, qualquer usuário pode navegar através da implementação das ligações entre os registros.
- As ligações nos registros são explícitas, isso mostra ao usuário do banco de dados uma visão que reflete a forma em que os registros estão organizados, armazenados e são localizados no meio físico.
- Não possuem Independência Física dos Dados – obriga o usuário a conhecer detalhes físicos do banco de dados, isso limita a: extensibilidade, a manutenção e a portabilidade dos aplicativos desenvolvidos.

### Modelo Hierárquico

Em um sistema hierárquico, os registros que representam entidades são conectados por ligações, formando árvores, ligações hierárquicas (não há ciclos). Cada registro pode estar ligado a vários filhos, mas conecta-se a um único registro pai. Cada registro no IMS corresponde a um conjunto de campos.

As estruturas de um banco de dados hierárquico seguem o estilo de um organograma empresarial (Diretoria-Divisão-Seção-Setor) ou de uma biblioteca (Exata-Matemática-Algebra Linear-Vetores).

Este modelo é capaz de representar este tipo de organização de forma direta, mas apresenta inconvenientes quando esta situação não aparece claramente com relações de hierarquia, como podemos visualizar no exemplo de um Sistema de Folha de Pagamento:

**Fábrica** Financeiro Comercial

**Injeção** Extrusão Pagar Receber Contábil Vendas Marketing

**Paulo** André Alessandra Sílvia Luiz Adalberto Rogério

Pedro Carlos Maria Luciana Paula Pedro João

Sabemos que Paulo é “filho” da Injeção que por sua vez é “filha” da Fábrica.

Um SGBD utilizando até os dias de hoje é o sistema IMS – Information Management System que representa um modelo hierárquico que surgiu em 1968, sendo um dos primeiros SGBDs comercializados. Permite ligações de Redes.

As linguagens usadas são DBD para descrição do banco de dados e PCB para a comunicação das aplicações com o SGBD e especificação de Visões.

### Modelo de Redes

Em um esquema em rede, os registros que representam entidades são relacionados através de ligações, podendo formar qualquer tipo de grafo, inclusive com ciclos.

Um SGBD que representa um modelo em rede é o sistema DBTG. Através da DDL do DBTG é possível declarar: o nome do esquema; tipos de registro; conjuntos que definem relacionamentos entre registros; e áreas físicas para armazenamento de registros.

Um outro SGBD em rede que surgiu a partir do DBTG é o IDMS.

## Apêndice B - Tutorial Erwin

### Apresentação

O ERwin é uma ferramenta muito poderosa para modelagem de dados de uma maneira simples e fácil.

Sua principal vantagem é que, uma vez modelado um sistema, você pode criar e manter banco de dados em muitos servidores diferentes.

Os bancos de dados aceitos pelo ERWin são:

DBMS	Connect Via
AS/400	ODBC.DLL
DB2/MVS, DB2/390	ODBC.DLL
DB2/CS, DB2/UDB	ODBC.DLL
INFORMIX	LDLLSQLW.DLL and ODBC.DLL
Ingres/OpenIngres	ODBC.DLL
InterBase	ODBC.DLL
ORACLE 6	ORA6WIN.DLL
ORACLE 7.x	ORA7WIN.DLL
PROGRESS	ODBC.DLL
Rdb	ODBC.DLL
Red Brick	ODBC.DLL
SQLBase	SQLAPIW.DLL
SQL Server	W3DBLIB.DLL
SYBASE	W3DBLIB.DLL
Teradata	ODBC.DLL
WATCOM/SQL Anywhere	ODBC.DLL

### Conceitos Iniciais

Antes de iniciar a explicação de como criar um modelo de dados no ERWin, iremos explicar conceitos iniciais como criar novos diagramas, abrir diagramas já existentes, alternar entre os modelos lógico e

físico e mudar a metodologia usada para representar os modelos de dados.

## Criando um Novo Diagrama

Ao iniciar o ERwin, uma janela contendo um novo diagrama é aberta, podendo assim iniciar a criação de um novo modelo de dados.

Para criar novos diagramas:

1. Escolha o comando “New...” no menu “File”, aparecendo a caixa de diálogo “ERwin Template Selection”, como mostra a Figura 1.

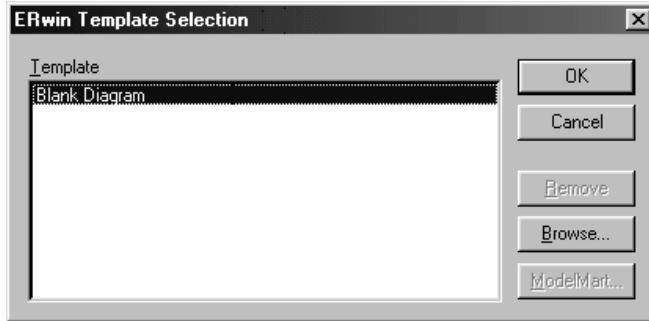


Figura 1.

2. Selecione “Blank Diagram” e clique no botão OK.

## Abrindo um Diagrama Existente

Para abrir um diagrama já existente:

1. Escolha o comando “Open...” no menu “File”
2. Selecione o arquivo desejado e pressione o botão OK.

## Alternando entre os Modelos Lógico e Físico

O ERwin possibilita a visualização de dois modelos:

**Lógico:** O modelo lógico, também chamado de Modelo Conceitual, são visualizadas as entidades e seus atributos-chave e não-chave.

**Físico:** Neste modelo visualizamos as tabelas, suas colunas e seus tipos de dados.

Para alternar entre os modelos lógico e físico:

1. Escolha o comando “Logical Model” para selecionar o Modelo Lógico ou “Physical Model” para selecionar o modelo Físico no menu “Edit”

## Representação Gráfica

### Modelo Lógico

No Modelo lógico, podemos utilizar dois tipos de metodologias:

ou use a barra de ferramentas como mostra a Figura 2 e a Figura 3.

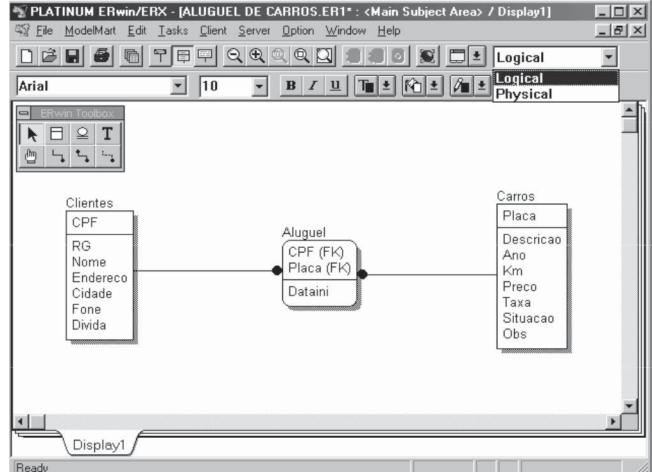


Figura 2 - “Modelo Lógico”.

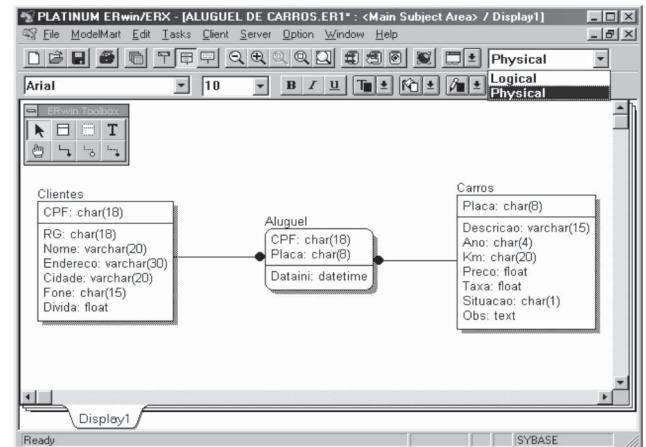
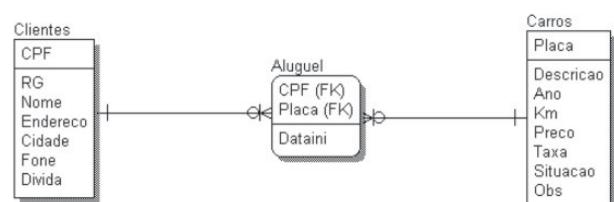
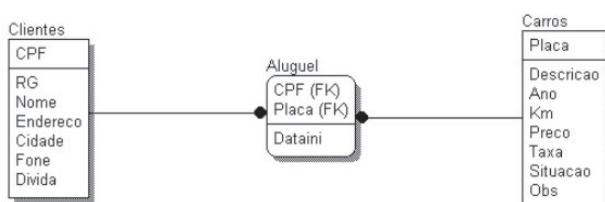


Figura 3 - “Modelo Físico”.

### Metodologias

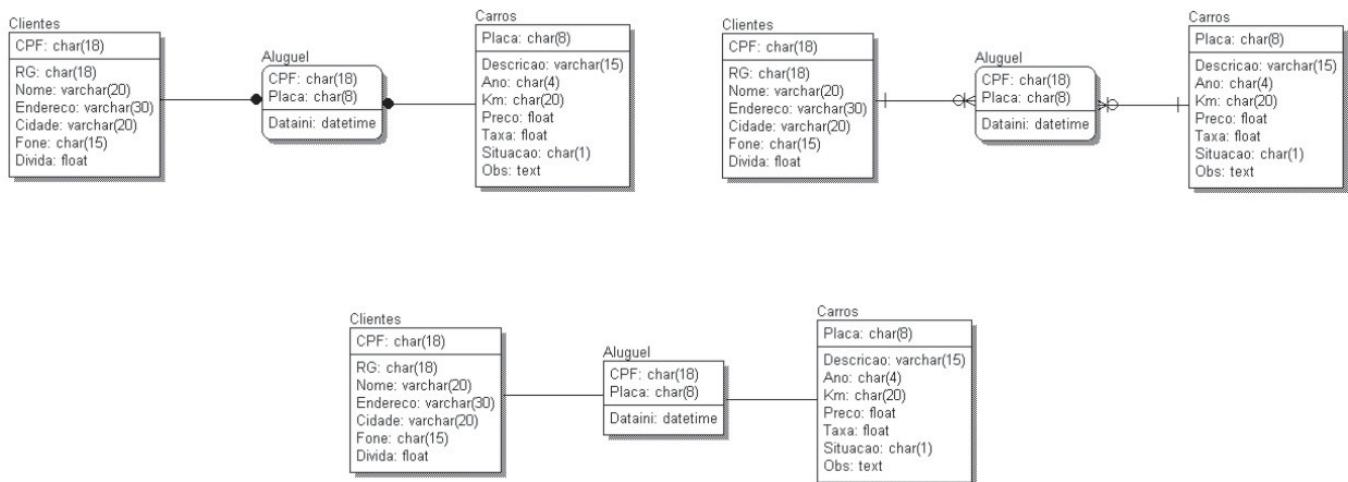
As metodologias disponíveis para representar os modelos de dados no ERWin são:

- **IDEF1X – Integration DEFinition for Information Modeling**
- **IE – Information Engineering**
- **DM – Dimensional Modeling** (só está disponível para ser utilizada no Modelo Físico)



**Modelo Físico**

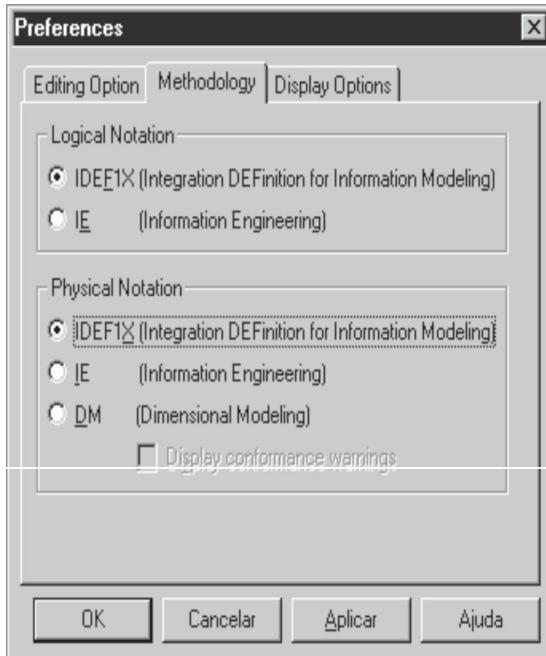
No Modelo físico, podemos utilizar três tipos de metodologias:

**Mudar a Metodologia:**

1. Escolha o comando “**Preferences...**” do menu “**Options**”

2. Clique na guia “**Methodology**”

Selecione a notação Lógica e Física desejada, como mostra a Figura 4.



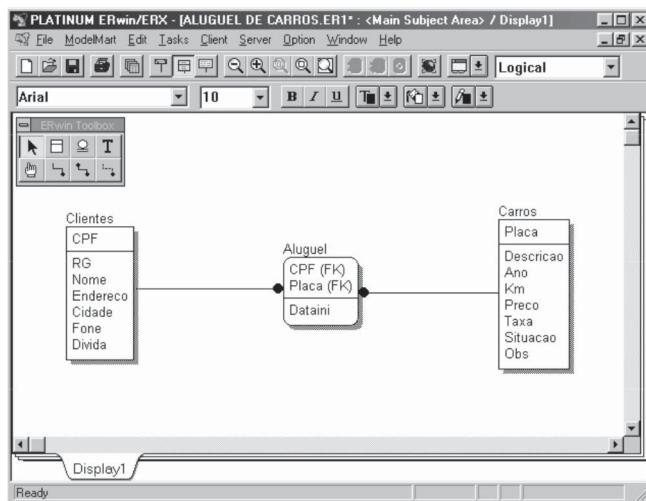
*Figura 4 - Caixa de Dialogo “Preferences”.*

**Observação:** Neste tutorial, iremos utilizar a notação IDEF1X para ambos os modelos Lógico e Físico.

**Modelagem Lógica**

Este tutorial irá se basear no exemplo do modelo de dados da figura 5.

Físico.



*Figura 5 - Modelo de “Dados Lógicos”.*

O Modelo de Dados que estamos usando como exemplo contém três entidades: Clientes, Carros e Aluguel.

Os carros são descritos pela placa, descrição, ano, modelo, km, Preço por km, Taxa Diária, Situação e Observação.

Os clientes são cadastrados pelo seu CPF, RG, nome, endereço, cidade, telefone e dívida (reservado para registrar pagamentos pendentes).

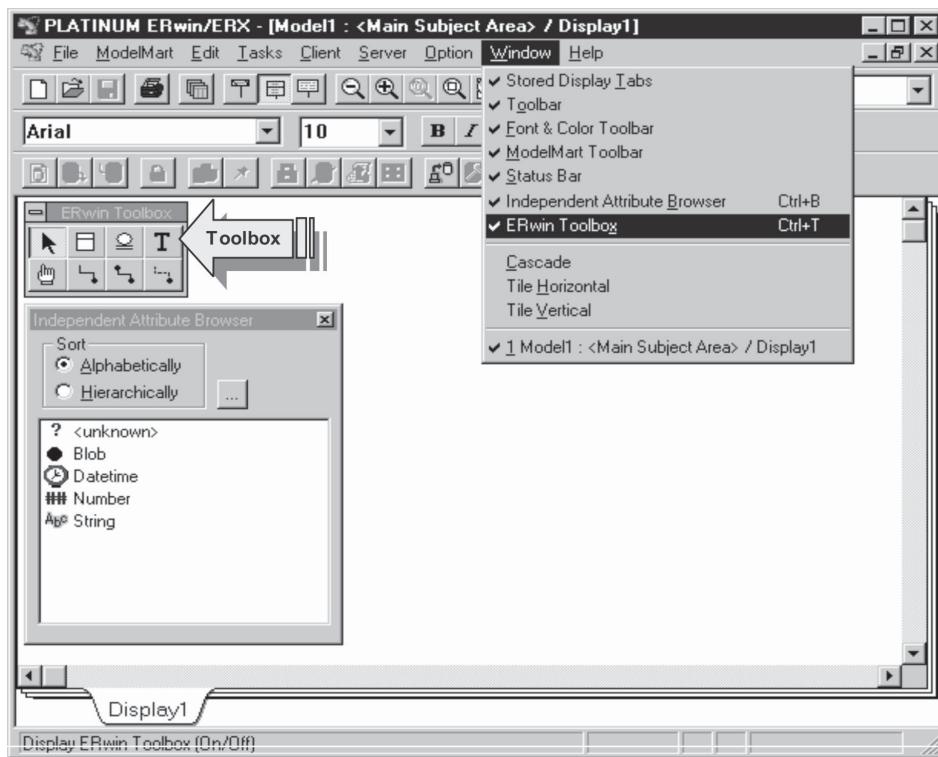
Quando um carro disponível é escolhido pelo cliente para ser alugado é registrada a data inicial do aluguel.

## Ferramenta de Trabalho

Para começar a desenhar um modelo de dados, precisamos de uma caixa de ferramentas chamada “Erwin Toolbox”.

### Exibir/Ocultar a Caixa de Ferramenta “Toolbox”:

1. Escolha o comando “ERwin Toolbox” no menu “Window” ou pressione a tecla CTRL+T
- Aparecerá a caixa de ferramentas “Erwin Toolbox” como mostra a figura 6.



*Figura 6 - Caixa de Ferramentas “Erwin Toolbox”.*

**Botões da Caixa de Ferramentas “Erwin Toolbox”**

A caixa de ferramentas “ERWin Toolbox” possui comandos para criar entidades/tabelas ou relacionamentos.

**Seleção**

Utilizado para selecionar e **mover objetos** na área de trabalho, tanto no **modelo lógico** quanto no **modelo físico**.

**Entity**

Utilizado para criar **entidades** no **modelo lógico**. Cada entidade no modelo lógico corresponde a uma **tabela** no **modelo físico**. É usado para criar tanto entidades dependentes quanto independentes, como será visto na parte **entidades e atributos**.

**Complete sub-category**

Utilizado para criar supertipos e subtipos.

**Text**

Utilizado para inserir comentários no modelo de dados.

**Attribute manipulation**

Utilizado para manipular atributos, ou seja, modificar a posição do atributo de uma entidade ou mover atributos de uma entidade para outra.

**Identifying Relationship**

Utilizado para criar **relacionamentos identificáveis**.

**Many-to-many Relationship**

Utilizado para criar **relacionamentos muitos-para-muitos**, portanto, não há a necessidade de criar uma terceira tabela para relacionamentos muitos-para-muitos.

**Non-identifying Relationship**

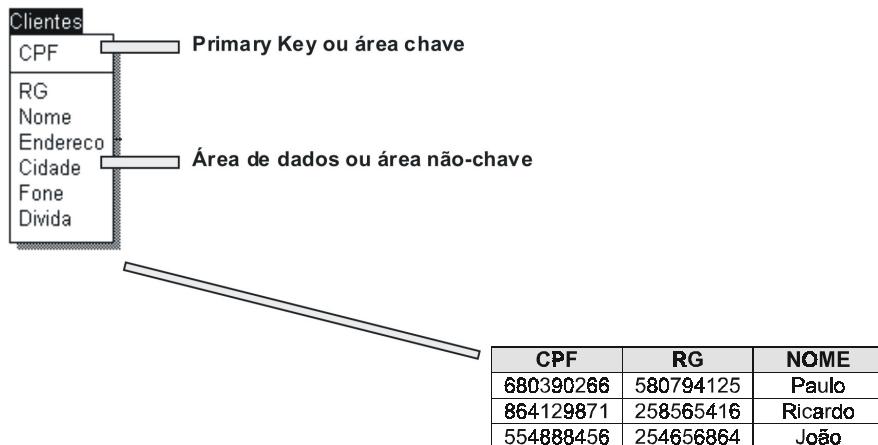
Utilizado para criar **relacionamentos não-identificáveis**.

**Entidades e Atributos****Definição**

Cada entidade corresponde a uma tabela e cada atributo corresponde a uma coluna da tabela no banco de dados.

**Representação Erwin****Atributos-Chave X Não-Chave**

No Erwin, cada entidade é dividida por uma linha horizontal que separa o atributo dentro de dois grupos (área-chave e não-chave), conforme mostra a Figura 7.



*Figura 7 - Exemplo de Entidade/Atributos X Tabela/Colunas.*

A área de cima da linha é chamada de área-chave contém a chave primária. A chave primária pode ser composta por um ou mais atributos/colunas, sendo que estes são usados para identificar uma única instância da entidade/tabela.

Os atributos não-chave são localizados abaixo da linha.

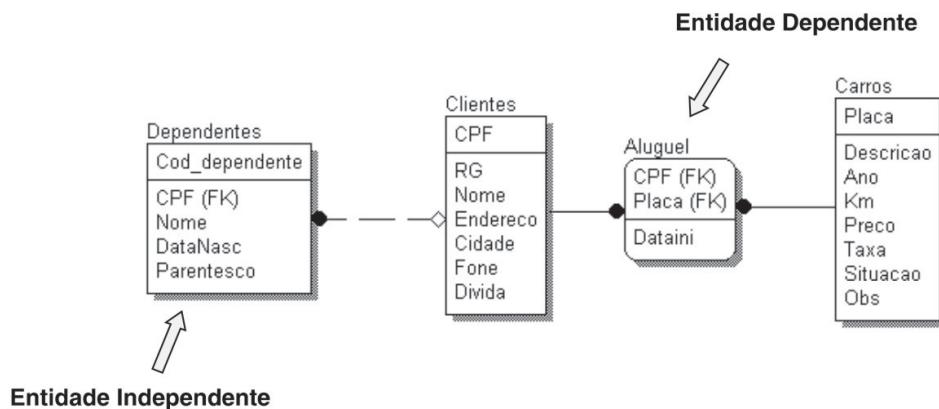
### Tipos de Entidades (Dependente X Independente)

O tipo da entidade (dependente ou independente) é determinado pelo tipo de relacionamento.

Para criar entidades dependentes, temos que criar um relacionamento do tipo  “Identifying Relationship”, ou seja, Relacionamentos Identificados

Para criar entidades independentes, temos que criar um relacionamento do tipo  “Non-Identifying Relationship”, ou seja, Relacionamentos Não-identificáveis.

A entidade dependente é representada por uma caixa com cantos arredondados e a entidade independente é representada por uma caixa retangular, conforme a Figura 8.



*Figura 8 - Entidades Dependentes x Entidades Independentes.*

A criação de relacionamentos, que irá determinar os tipos de entidades será vista com maiores detalhes no item **relacionamentos** deste tutorial.

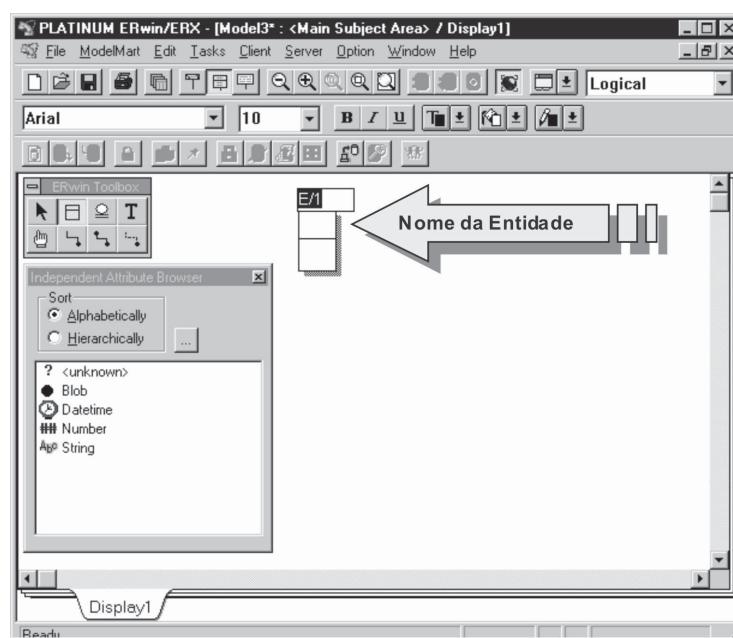
### Criando Entidades e Atributos

1. Selecione o botão “Entity”  da barra de ferramentas “Toolbox”

2. Dê um clique na área de trabalho

Aparecerá um nome-padrão para a entidade E/n, onde n é o próximo número da entidade a ser criada, conforme mostra a Figura 9.

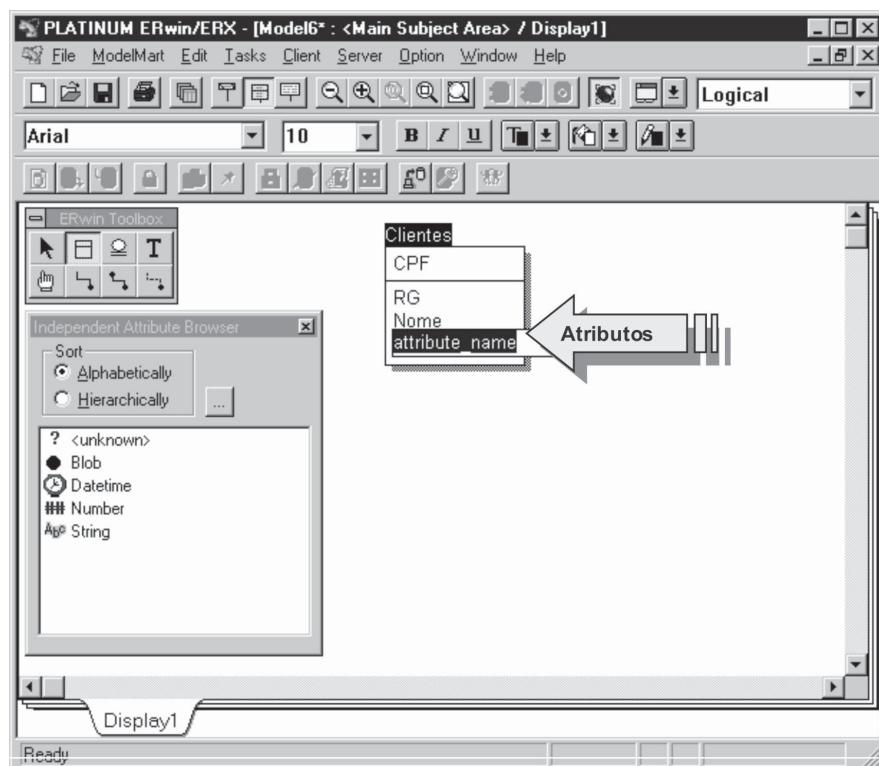
3. Digite o nome desejado para a entidade



*Figura 9 – Exemplo de criação de entidades.*

## INTRODUÇÃO A BANCO DE DADOS

4. Após digitar o nome da entidade, pressione a tecla <TAB/ENTER> para incluir os **atributos-chave** (primary key).
  5. Após digitar os atributos-chave, pressione a tecla <TAB> (a tecla TAB serve para navegar entre o nome da entidade, a área-chave e a área não-chave das entidades) para ativar a área de atributos não-chave. Digite os atributos não-chave (abaixo da linha horizontal), pressionando <ENTER> para continuar incluindo novos atributos não-chave, conforme a Figura 10.
- Após digitar o último atributo, pressione a tecla <ESC> para concluir.



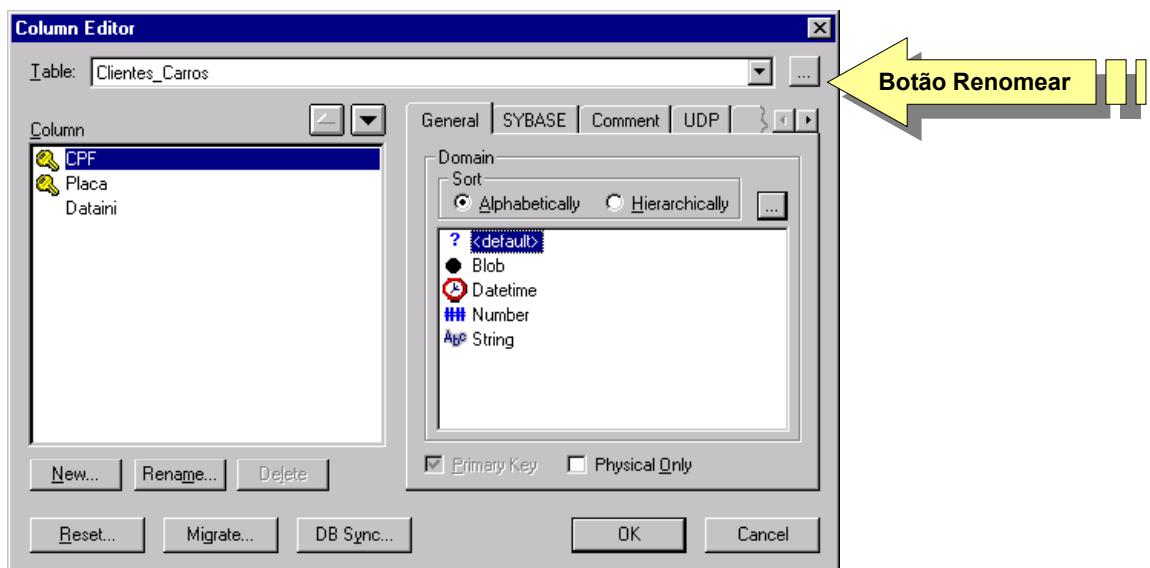
*Figura 10 - Inserindo atributos em Entidades.*

### Renomear Entidades

Para renomear entidades, tanto faz estar no modelo lógico ou físico.

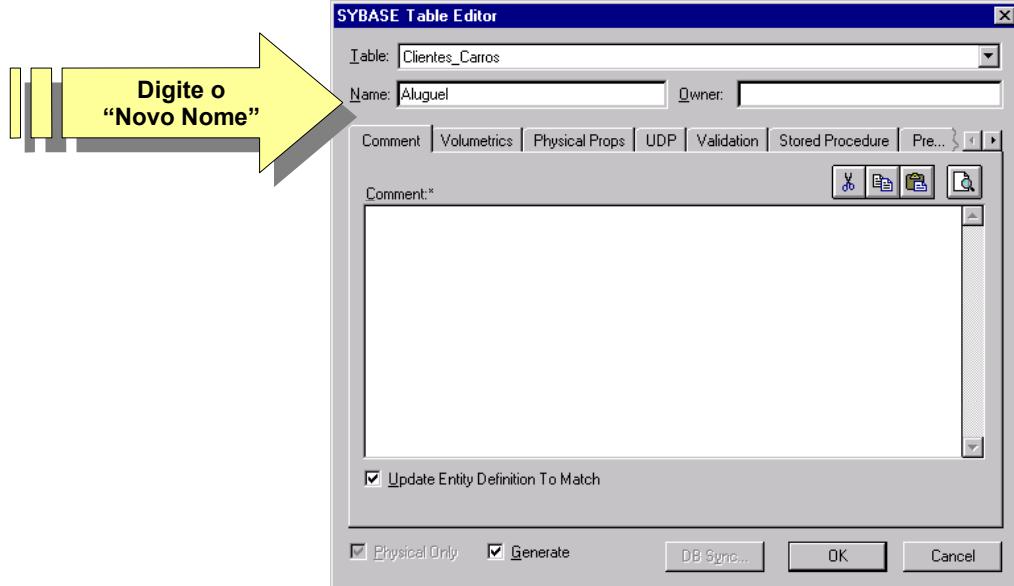
1. Dê um clique duplo na entidade cujo nome.3.2 deseja mudar.

Aparecerá a caixa de diálogo da Figura 11:



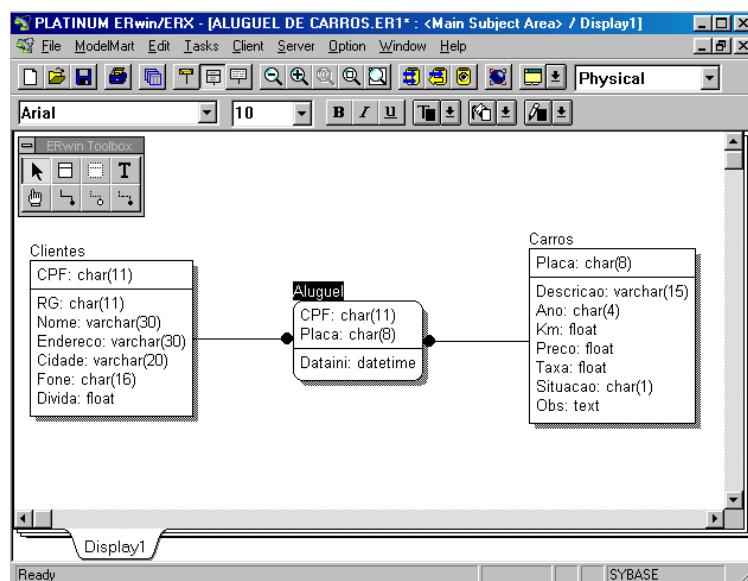
*Figura 11 - Caixa de diálogo “Column Editor”.*

Aparecerá a caixa de diálogo da Figura 12:



**Figura 12 - Caixa de diálogo SYBASE Table Editor .**

2. Digite o novo nome no item “Name” da janela acima e pressione o botão OK até fechar todas as janelas.  
Agora o Modelo ficou como mostra a Figura 13:



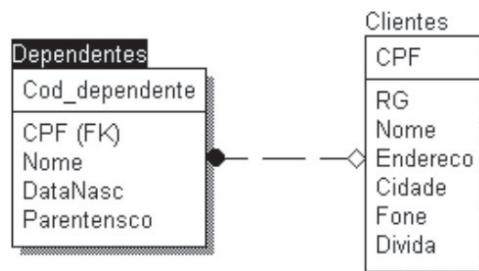
**Figura 13 - Modelo de Dados de Exemplo .**

### Propriedades de Atributos

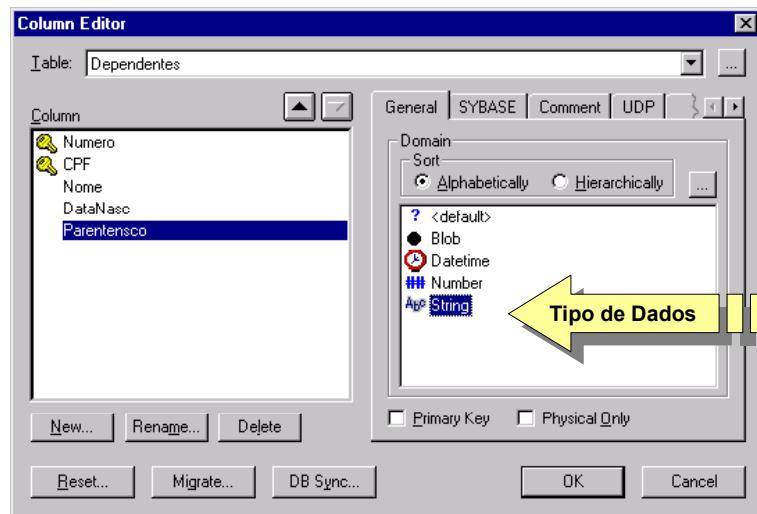
Para alterar as propriedades dos atributos como alterar o nome e o tipo de dados, tanto faz, estamos no modelo lógico ou físico.

As propriedades NULL/NOT NULL e o tamanho do atributo só podem ser alterados no modelo físico.

1. Dê um clique duplo na entidade que contém o atributo que deseja alterar.

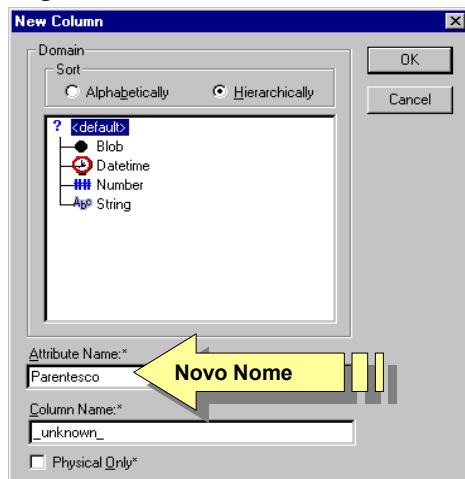


Aparecerá a caixa de diálogo da Figura 14.



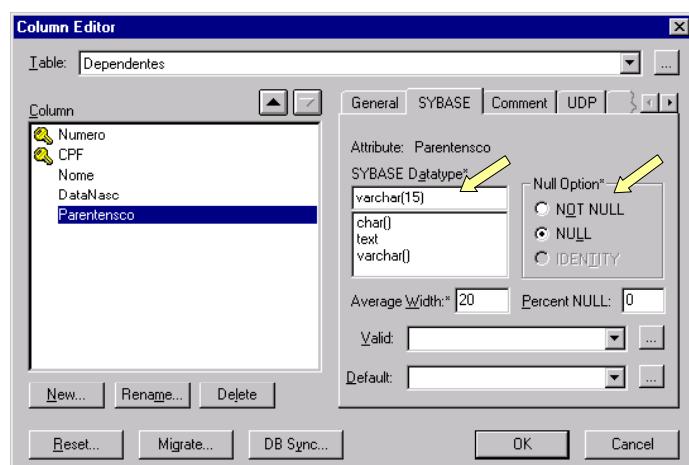
*Figura 14 - Caixa de Diálogo Column Edito*

2. Selecione o atributo que deseja alterar.
3. Selecione o Tipo do Dado, por exemplo, **String**, como mostra a figura 16.
4. Clique no botão “New...” para corrigir o nome do atributo “Parentesco” para “Parentesco”. Aparecerá a caixa de diálogo da Figura 15.



*Figura 15 - Caixa de Diálogo “New Column”.*

5. Pressione o botão OK
6. Para alterar o datatype de um atributo, após selecionar o atributo, selecionar a guia contendo o nome do banco escolhido para gerar o modelo de dados, que no nosso tutorial é a guia “SYBASE”
7. Digite o tamanho do tipo de dados desejado e selecione se o atributo será NULL / NOT NULL, como mostra a Figura 16.



*Figura 16 - Caixa de Diálogo “Column Editor”.*



## Relacionamentos

### Definição

Relacionamentos são mais complexos do que parecem. Eles compartilham parte da informação.

Um relacionamento é uma associação entre duas entidades e é representado por uma linha sólida ou tracejada.

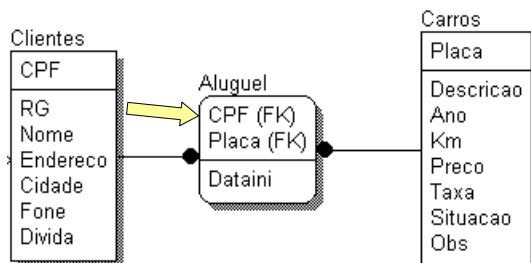
### Representação Erwin

No Erwin, um relacionamento é representado por uma linha conectando duas entidades. Dependendo da notação escolhida, a visualização será diferente (veja item 2.4 METODOLOGIA).

Existem dois tipos de relacionamentos:

### Identifying (Relacionamento Identificado)

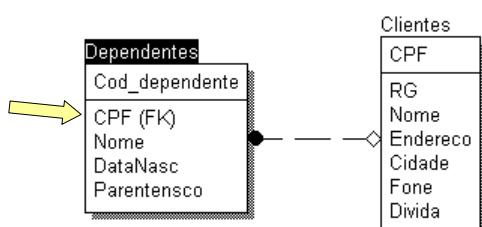
Num relacionamento Identificado, a FK migra para cima da linha, ou seja, faz parte da chave primária da entidade filha, como mostra o exemplo da Figura 17.



*Figura 17 - Exemplo de um relacionamento identificado.*

### Non-Identifying (Relacionamento Não-Identificado)

Em um relacionamento Não-Identificado, a FK migra para baixo da linha, ou seja, não faz parte da chave primária da entidade filha, como mostra o exemplo da Figura 18.

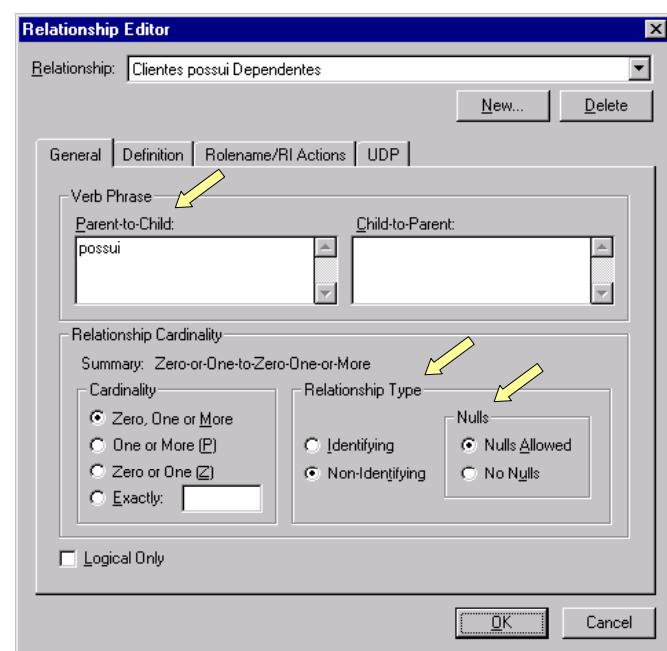


*Figura 18 - Exemplo de um relacionamento não-identificado.*

### Tipos de Relacionamentos

Para representar um relacionamento identificável, use o botão

Para representar um relacionamento não-identificável, use o botão



*Figura 19 - Caixa de Diálogo “Relationship Editor”.*

Para os tipos de relacionamentos “Não-identificáveis” é possível preencher a opção “Nulls” (Nulls Allowed ou No Nulls) para “Permite Nulos” ou “Não-Nulos”, como mostra a figura 18 acima.

No item “Verb Phase” (Parent-to-Child/Child-to-Parent) na caixa de diálogo “Relationship Editor”, é possível definir o nome do relacionamento.

### Cardinalidade

A cardinalidade é uma propriedade do relacionamento que define exatamente como muitas instâncias aparecem em uma tabela filha para cada instância correspondente em uma tabela pai.

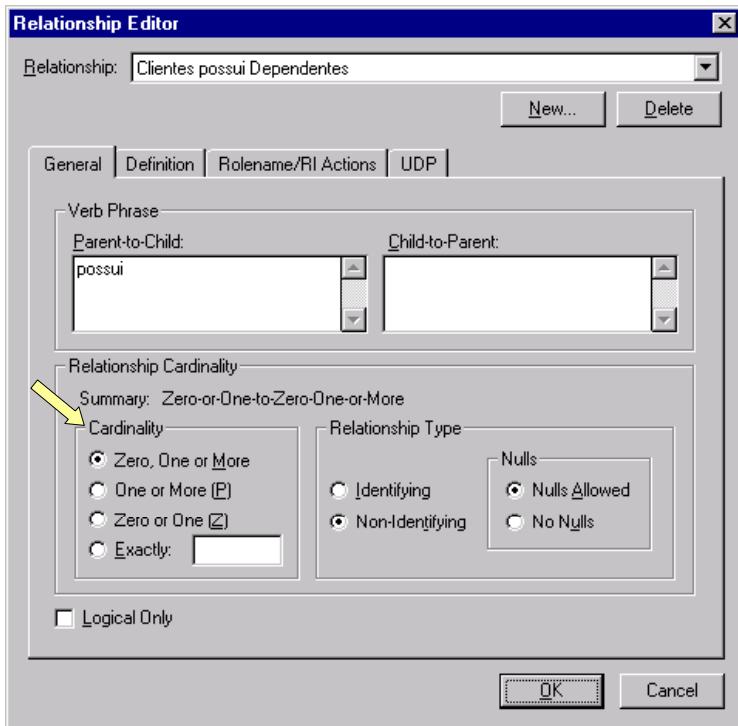
A idéia de “muitos” em um relacionamento não significa que tem que ter mais de uma instância de um filho conectado a um determinado pai.

Em lugar de “muitos” em “1 para muitos” realmente significa que tem zero, um ou mais instâncias de um filho unindo-se ao pai.

Para alterar a cardinalidade de um relacionamento, dê um clique duplo no relacionamento para ativar a caixa de diálogo “Relationship Editor”, como mostra a Figura 20.

Os tipos de cardinalidade existentes são:

- Zero, Um ou Mais
- 1 ou Mais (P)
- Zero ou Um
- Exatamente n (n indica a quantidade desejada)

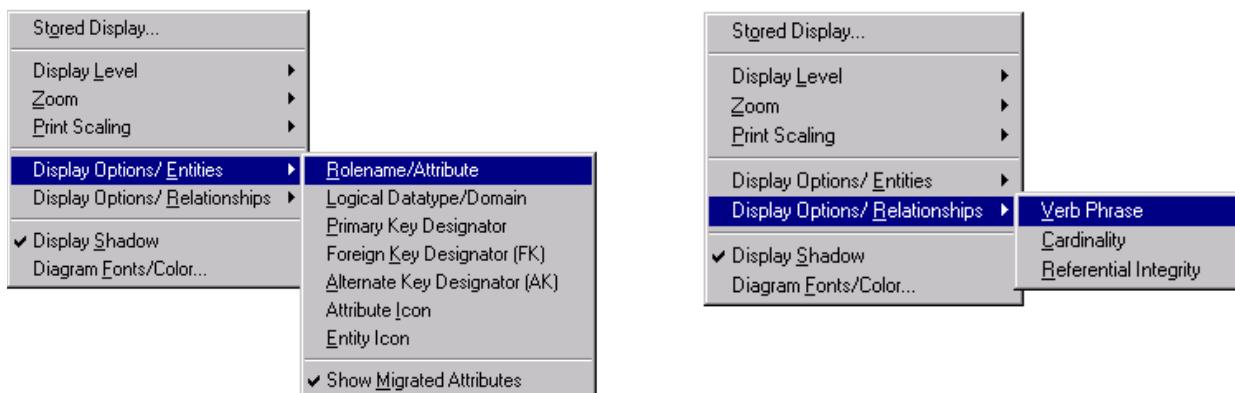


*Figura 20 - Caixa de Diálogo “Relationship Editor”.*

### Opções de Exibição de Entidades / Relacionamentos

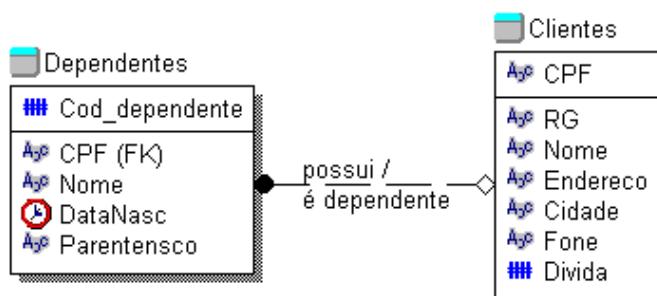
A ferramenta ERWin permite que a visualização do modelo mostre mais informações para entidades/atributos.

Para selecionar opções de visualização, dê um clique com o botão direito e escolha o comando “Display Options/ Entities” ou “Display Options/Relationships/Relationships” e escolha os subcomandos desejados, como mostra a Figura 21.



*Figura 21*

Selecionando os seguintes subcomandos: Primary Key Designator, Foreign Key Designator (FK), Attribute Icon e Entity Icon para o comando “Display Options / Entities” e os subcomandos: Verb Phrase e Cardinality do comando “Display Options/Relationships” o nosso modelo ficará como a Figura 22.



*Figura 22*

## Exemplos de Relacionamentos

Neste exemplo (Fig. 23) são mostrados vários tipos de relacionamentos, apesar da ferramenta ERWin possuir apenas três botões para desenhá-los, você pode representar todos os tipos de relacionamentos usando-se da cardinalidade.

Por exemplo, quando desejamos representar um relacionamento 1 para 1 (veja Disciplina / Ementa): a entidade Ementa é uma entidade fraca, ou seja, não possui identificação própria, sendo assim, utilizamos o tipo de relacionamento “Identificado”  e definimos cardinalidade 1, ou seja, a disciplina possui apenas 1 ementa e não “n” ementas.

Um relacionamento com cardinalidade “1 ou mais” ao invés de “0, 1 ou mais” é representado pelas entidades Orientador / Departamento, onde um Ori-

entador está em apenas 1 departamento e o departamento possui no mínimo 1 e no máximo “n” orientadores (representado pelo ERWin pela letra “P”). Como o Orientador é identificado independentemente do departamento, utilizamos um relacionamento “não-identificado” .

As entidades Orientador / Aluno possuem um relacionamento “Não-identificado” também, porém, a cardinalidade representada é “0, 1 ou mais”, ou seja, um orientador pode orientar 0, 1, ou mais alunos. Em algum momento, o professor pode estar sem nenhum orientando.

Para representar, por exemplo, que um aluno possui 0 ou 1 nota, representamos esse relacionamento como “Identificado”  e mudamos a cardinalidade para 0 ou 1 (representado pelo ERWin pela letra “Z”).

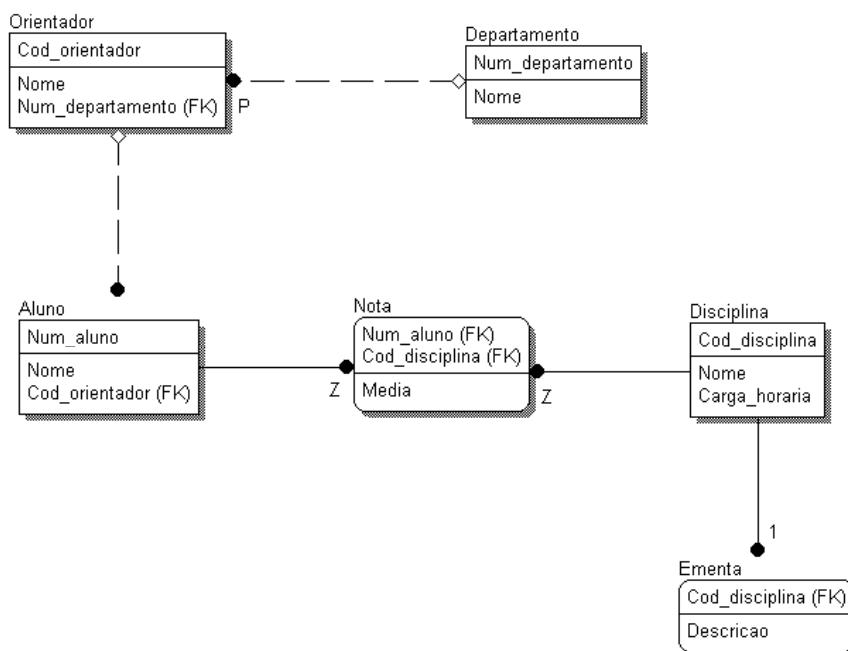


Figura 23

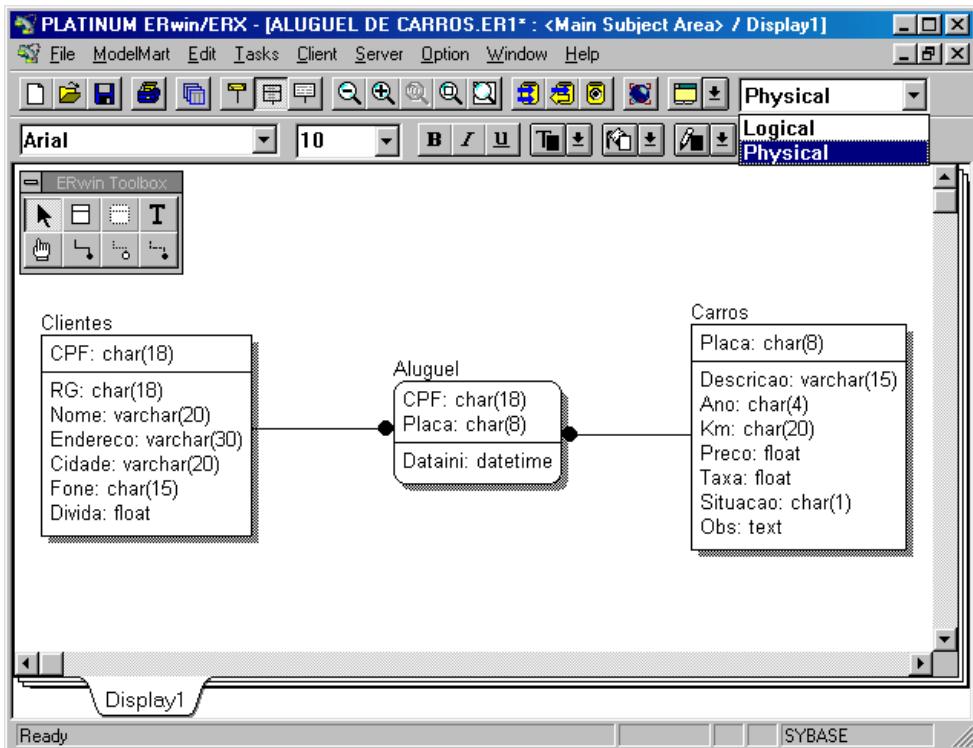
## Modelagem Física

Após criar o modelo lógico de dados, mude para a notação física, como vimos no item 2.3, o modelo aparecerá como a figura 11.

No Modelo Físico, podemos alternar mudar o tamanho do tipo de dados e também gerar um script com os comandos para criação do modelo de dados no banco desejado.

Nesta parte do tutorial, iremos modificar o nome de entidades, mudar o nome de atributos e suas características como tipo de dados, tamanho e a propriedade NULL e NOT NULL.

Em seguida, será mostrado como gerar os scripts para a criação do modelo de dados da figura 24 e a criação do modelo no banco de dados desejado.



*Figura 24*

### Scripts para Criação do Modelo de Dados

Após concluir o modelo de dados desejado, podemos criar o modelo de dados de duas maneiras:

1. Criar diretamente no banco de dados desejado
2. Criar um arquivo (script) contendo os comandos para serem executados para a criação do modelo

Neste tutorial, iremos explicar a segunda opção, ou seja, como gerar os scripts de criação do modelo de dados.

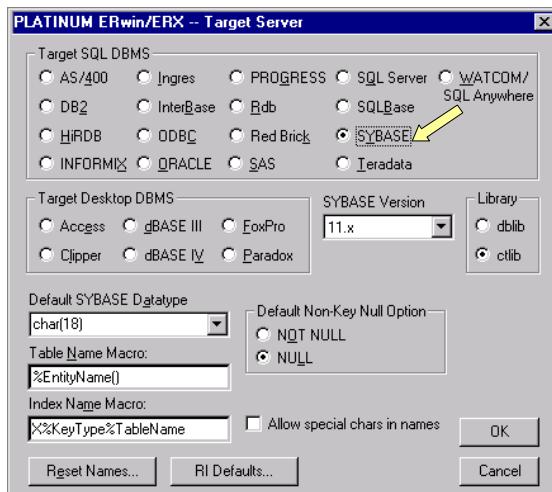
### Criação do Script

Após terminar o modelo de dados desejado:

1. Mude para o modelo físico (veja item 2.3 deste tutorial)
2. Selecione o botão “Select Target Server” ou escolha o comando “Target Server...” do menu “Server” para selecionar o banco de dados desejado.

Neste tutorial, iremos utilizar o banco SYBASE, como exemplo.

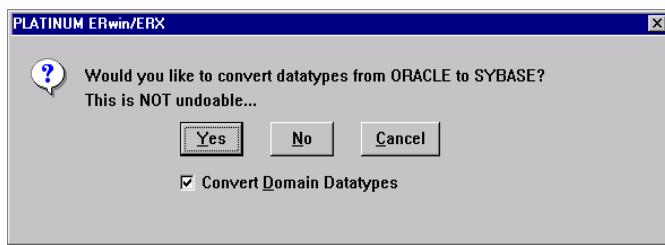
Aparecerá a caixa de diálogo a seguir:



*Caixa de Diálogo “Target Server”.*

3. Selecione o banco de dados desejado e pressione a tecla OK.

Aparecerá a seguinte caixa de diálogo:



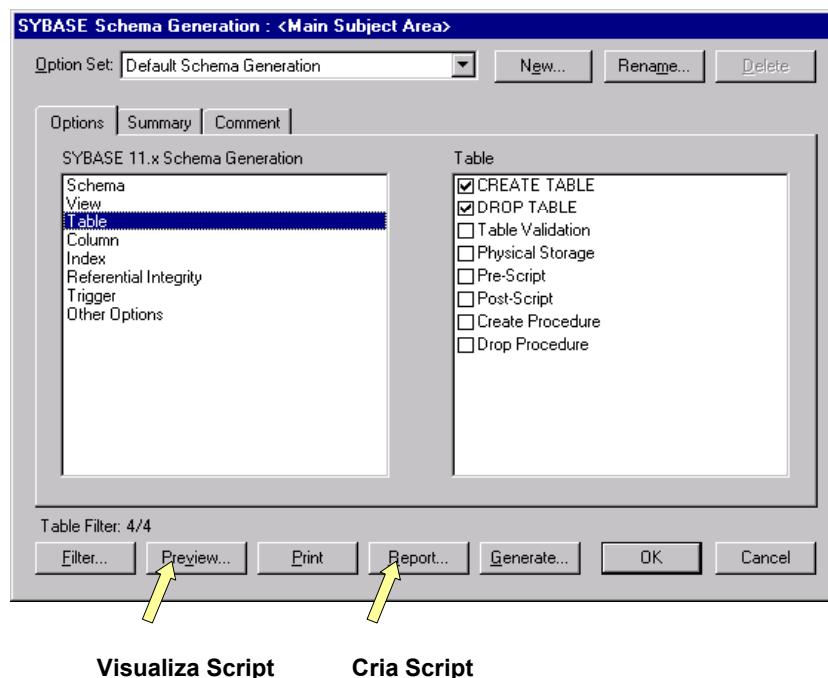
4. Pressione o botão “Yes” para converter os tipos de dados do Oracle para o Sybase.

O ERWin, por default, vem pré-definido para gerar os scripts em Oracle. Sendo assim, todas as vezes em que for alterado o banco, o ERWin perguntará se você deseja converter os tipos de dados correspondentes para o banco selecionado.

Após escolher o banco de dados desejado, iremos criar os scripts para geração do banco de dados.

5. Selecione o botão “Forward Engineer” ou selecione o comando “Forward Engineer/Schema Generation...” do menu “Tasks”

Aparecerá a caixa de diálogo da Figura 25, a seguir:



*Figura 25 - Caixa de Diálogo “Shema Generation”.*

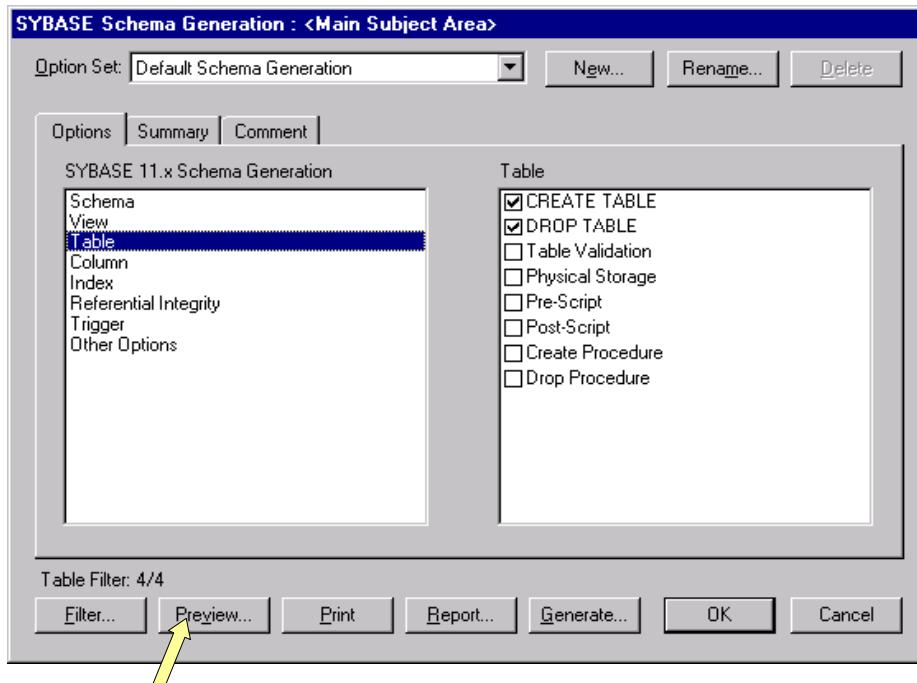
6. Na guia “Options”, selecione os itens que deseja incorporar no script.

7. Neste tutorial iremos assinalar somente algumas opções para gerar o script contendo os comandos para a criação das tabelas. No item “Table” assinale o comando: **CREATE TABLE**. No item “Column”, assinale todos os comandos: **VALIDATION**, **PHYSICAL ORDER**, **DEFAULT** e **USER DATATYPE**. E, por último, no item “Referential Integrity” assinale os comandos: **PRIMARY KEY**

(PK), **FOREING KEY (FK)** e **UNIQUE (AK)**.

8. Desmarque todos os demais itens.

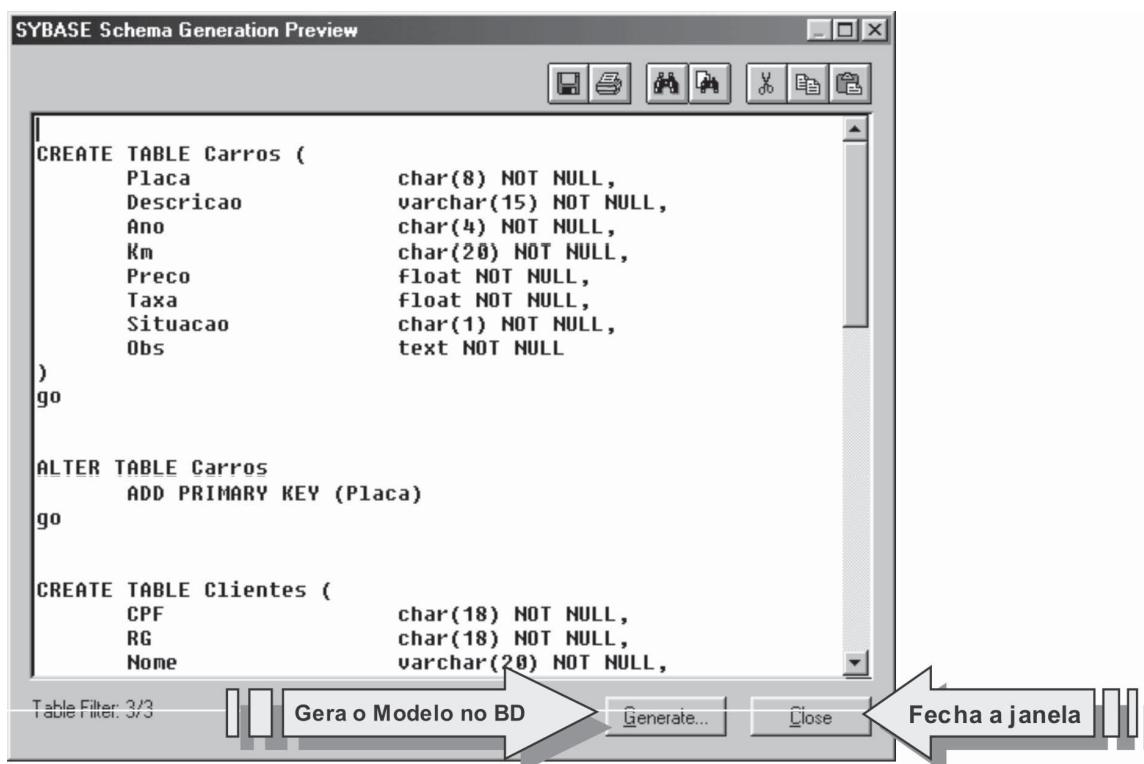
9. Para visualizar os comandos que serão colocados no script, clique no botão “Preview”, como mostra a Figura 26:



**Visualiza Script**

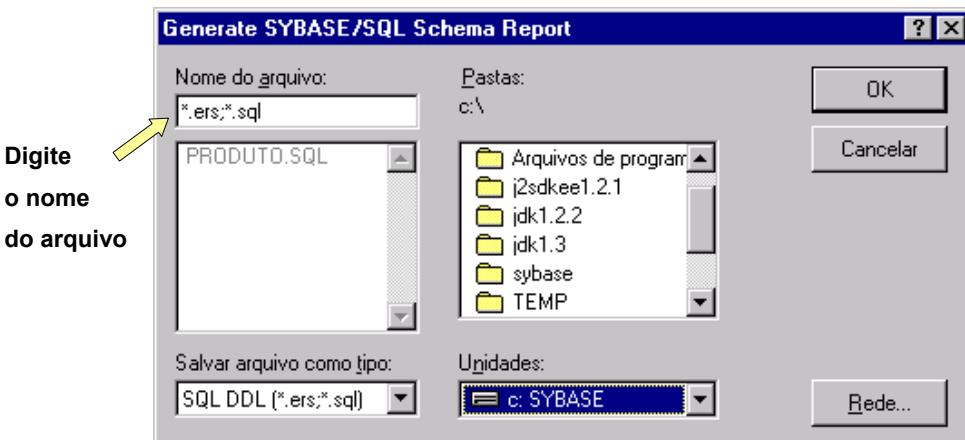
*Figura 26 - Caixa de Diálogo “Shema Generation”.*

Aparecerá a caixa de diálogo da Figura 27:



*Figura 27.*

10. Clique no botão “Generate” para gerar o modelo de dados no banco de dados selecionando ou clique no botão “Close” para fechar a janela.
  11. Para criar um script contendo os comandos que criam as tabelas do modelo, feche a janela “Preview” e clique no botão “Report...”, como mostra a figura 19.
- Aparecerá a seguinte caixa de diálogo da Figura 28:



**Figura 28 - Caixa de Diálogo “Generate SYBASE/SQL Schema Report”.**

12. Informe o drive, o diretório e o nome do arquivo com extensão (opcional).

Caso você não informe a extensão, o default é “\*.ers”.

Pronto !!!

Agora é só executar este script no SYBASE.

### Execução do Script no Banco de Dados SYBASE

O script gerado neste tutorial deve possuir o seguinte conteúdo:

```

CREATE TABLE Carros (
    Placa char(8) NOT NULL,
    Descricao varchar(15) NOT NULL,
    Ano char(4) NOT NULL,
    Km char(20) NOT NULL,
    Preco float NOT NULL,
    Taxa float NOT NULL,
    Situacao char(1) NOT NULL,
    Obs text NOT NULL
)
go
ALTER TABLE Carros
    ADD PRIMARY KEY (Placa)
go
CREATE TABLE Clientes (
    CPF char(18) NOT NULL,
    RG char(18) NOT NULL,
    Nome varchar(20) NOT NULL,
    Endereco varchar(30) NOT NULL,
    Cidade varchar(20) NOT NULL,
    Fone char(15) NOT NULL,
    Divida float NOT NULL
)
go
ALTER TABLE Clientes
    ADD PRIMARY KEY (CPF)

```

## INTRODUÇÃO A BANCO DE DADOS

```
go  
CREATE TABLE Aluguel (  
    CPF           char(18) NOT NULL,  
    Placa         char(8) NOT NULL,  
    Dataini      datetime NULL  
)  
go  
ALTER TABLE Aluguel  
    ADD PRIMARY KEY (CPF, Placa)  
go  
ALTER TABLE Aluguel  
    ADD FOREIGN KEY (Placa)  
        REFERENCES Carros  
go  
ALTER TABLE Aluguel  
    ADD FOREIGN KEY (CPF)  
        REFERENCES Clientes  
go
```

### Para Executar o Script no Sybase:

1. Abra o “Interactive SQL” Isql do banco de dados SYBASE
2. Escolha o comando “Open...” do menu **File** e selecione o arquivo do script gerado pelo ERWin e pressione o botão “Abrir”
3. Conecte-se usando o comando “Connect..” do menu ‘Command’
4. Clique no botão “Execute”, como mostra a Figura 29:

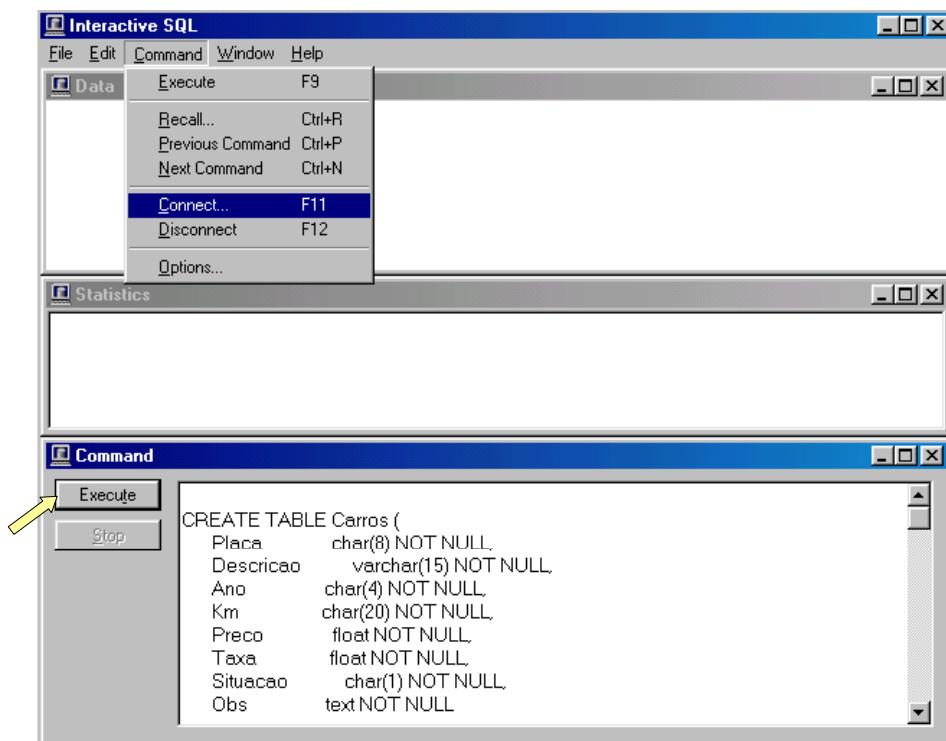


Figura 29 - Programa “Interactive SQL” - SYBASE.

## Engenharia Reversa

O ERWin também suporta Engenharia Reversa, ou seja, a partir de um banco de dados qualquer, pela engenharia reversa, é possível recuperar todo o modelo de dados.

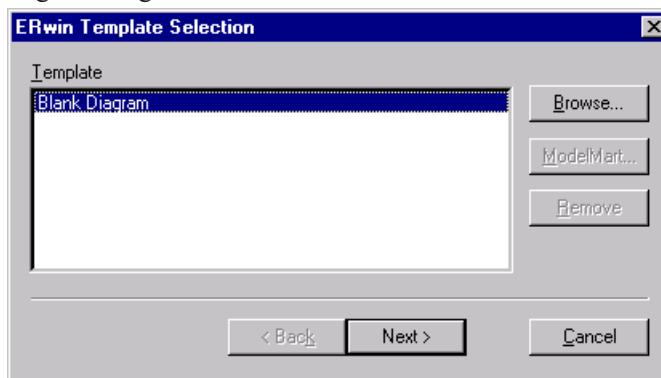
A Engenharia Reversa pode ser feita a partir de um script existente do banco de dados ou diretamente no banco de dados.

### Engenharia Reversa a Partir de um Script

Para recuperar o modelo de dados a partir de um arquivo (script):

1. Escolha o comando “Reverse Enginner...” do menu “Tasks”.

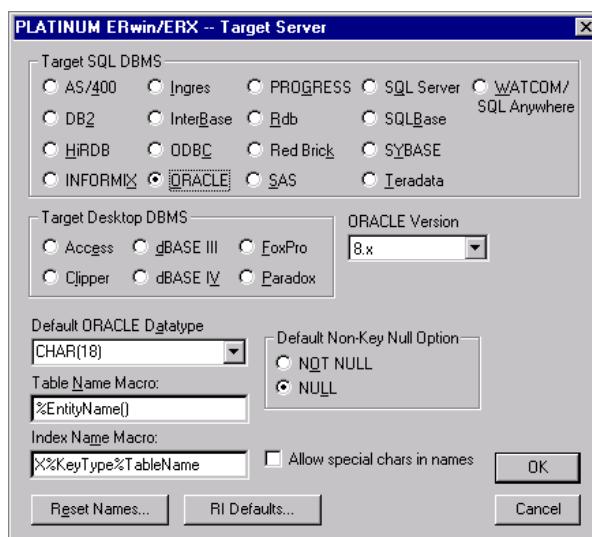
Aparecerá a caixa de diálogo da Figura 30.



*Figura 30 - Caixa de Diálogo “ERwin Template Selection”.*

2. Escolha o botão “Next >” ou selecione o botão “Browse...” e após escolha o botão “Next >” para passar para a etapa seguinte da Engenharia Reversa.

Aparecerá a caixa de diálogo da Figura 31.



*Figura 31 - Caixa de Diálogo “Select Target Server”.*

3. Selecione o banco de dados desejado e pressione o botão “Next >”.

Neste tutorial, estaremos utilizando um Script contendo os comandos para a criação de um banco de dados em Oracle. O conteúdo do script é o seguinte:

```

CREATE TABLE Disciplina (
    Cod_disciplina NUMBER NOT NULL,
    Nome VARCHAR2(30) NULL,
    Carga_horaria NUMBER NULL);
ALTER TABLE Disciplina
    ADD( PRIMARY KEY (Cod_disciplina));
CREATE TABLE Aluno

```

## INTRODUÇÃO A BANCO DE DADOS

```

(Num_aluno NUMBER NOT NULL,
Nome VARCHAR2(30) NULL);

ALTER TABLE Aluno
ADD( PRIMARY KEY (Num_aluno) ) ;

CREATE TABLE Notas (Num_aluno NUMBER NOT NULL,
Cod_disciplina NUMBER NOT NULL,
Media NUMBER(2,2) NULL);

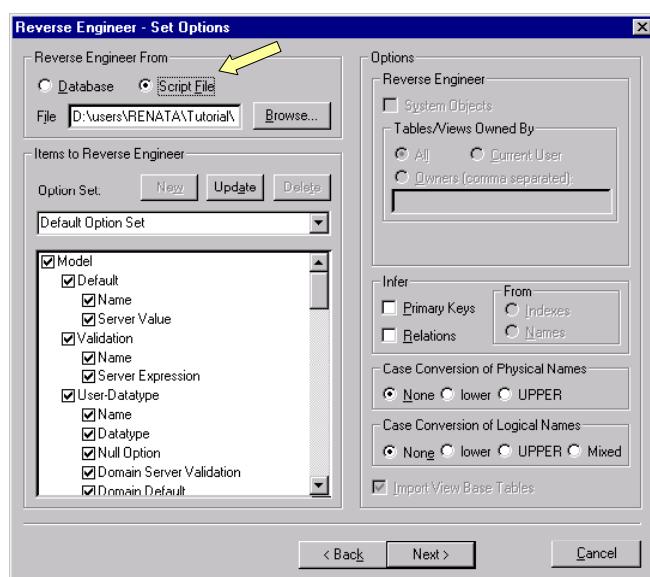
ALTER TABLE Notas
ADD ( PRIMARY KEY (Num_aluno, Cod_disciplina) ) ;

ALTER TABLE Notas
ADD( FOREIGN KEY(Cod_disciplina) REFERENCES Disciplina ) ;

ALTER TABLE Notas
ADD ( FOREIGN KEY (Num_aluno)
REFERENCES Aluno ) ;

```

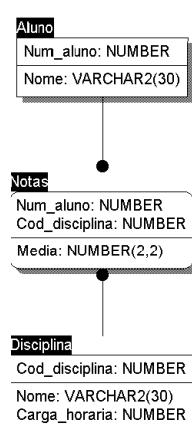
Aparecerá a caixa de diálogo da Figura 32 :



*Figura 32 - Caixa de Diálogo Engineer - Set Options.*

4. Selecione o local de onde se deseja realizar a Engenharia Reversa (“Database” ou “Script File”). Neste tópico, iremos escolher a opção “Script File”. Selecione o nome do script clicando no botão “Browse...”. Após escolher o arquivo de script, escolha o botão “Next >”.

Aparecerá no diagrama o modelo de dados contido no script, conforme a Figura 33.



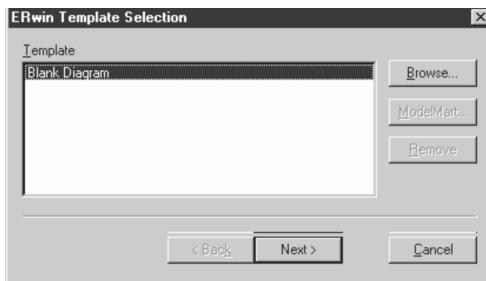
*Figura 33 - Exemplo do Modelo da Engenharia Reversa.*

## Engenharia Reversa Diretamente do BD

Iremos utilizar o banco de dados Oracle para recuperar o modelo de dados a partir de um banco de dados:

1. Escolha o comando “Reverse Enginner...” do menu “Tasks”.

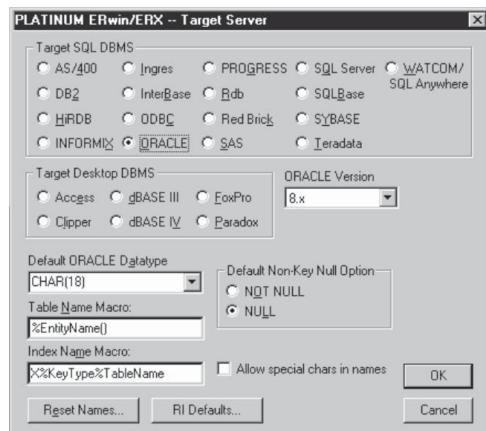
Aparecerá a caixa de diálogo da Figura 34.



*Figura 34 - Caixa de Diálogo “ERwin Templates Selection”.*

2. Escolha o botão “Next >” ou selecione o botão “Browse...” e em seguida escolha o botão “Next >” para passar para a etapa seguinte da Engenharia Reversa.

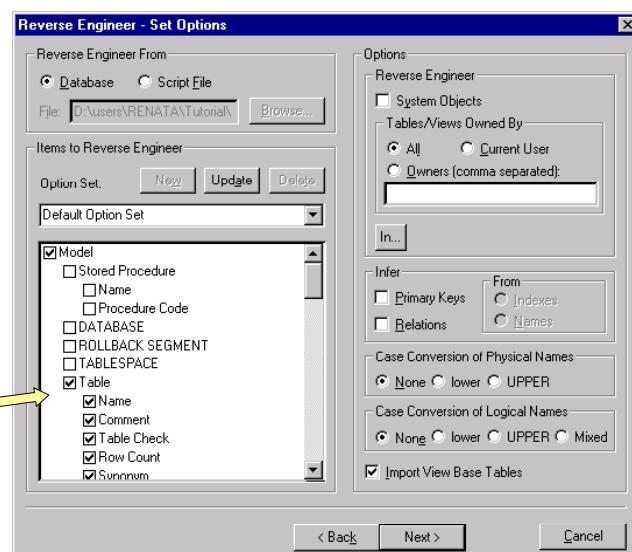
Aparecerá a caixa de diálogo da Figura 35.



*Figura 35 - Caixa de Diálogo “Select Target Server”.*

3. Selecione o banco de dados desejado (neste exemplo estamos utilizando o Banco de dados Oracle) e pressione o botão “Next >”.

Aparecerá a caixa de diálogo da Figura 36:

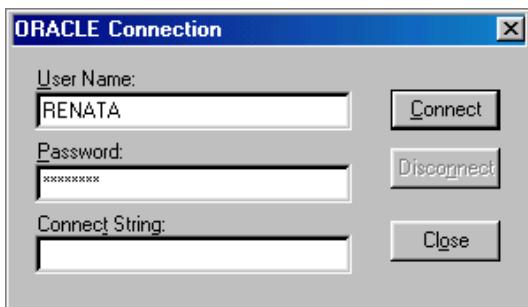


*Figura 36 - Caixa de Diálogo “Reverse Enginner - Set Options”.*

## INTRODUÇÃO A BANCO DE DADOS

4. Na caixa de diálogo “Reverse Enginner – Set options”, opção “Items to Reverse Enginner”, selecione apenas o item “TABLE” e pressione o botão “Next >”

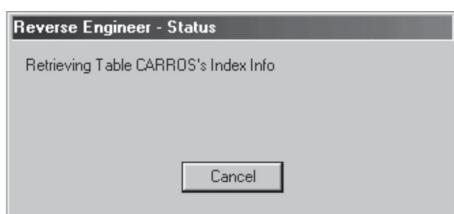
Se o banco não estiver iniciado, aparecerá a caixa de diálogo da Figura 37:



*Figura 37 - Caixa de Diálogo “ORACLE Connection”.*

5. Após iniciar o Startup do Banco, aparecerá a janela da Figura 38.

O ERWin estará recuperando todas as tabelas contidas no banco de dados.



*Figura 38 - Caixa de Diálogo “Reverse Enginner - Status”.*

## Apêndice C Introdução ao Oracle - SQL

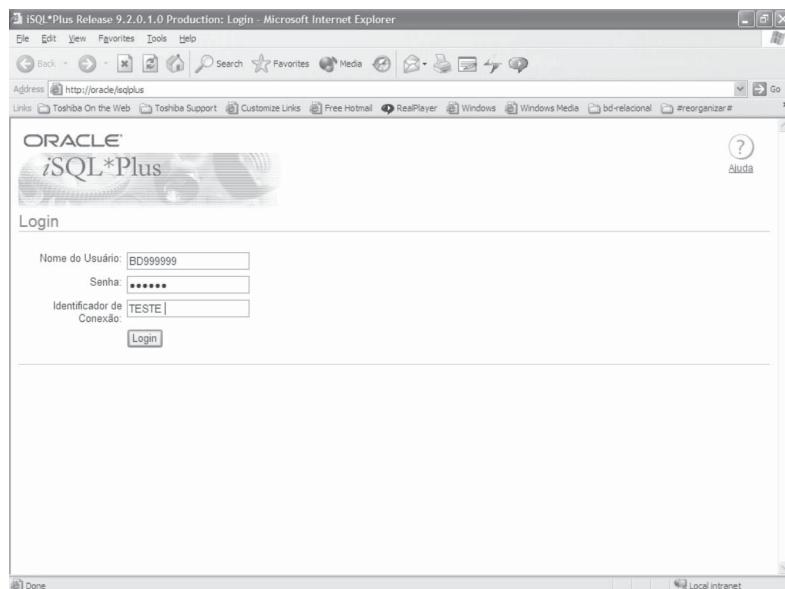
O Oracle9i é um banco de dados relacional. A linguagem SQL (Structured Query Language) é a base para qualquer banco de dados relacional.

### SQL \* Plus

A interface padrão do Oracle com a linguagem SQL é feita através do SQL \* Plus, usado para emitir consultas ad-hoc, relatórios e administração do banco de dados em modo caractere e execução de scripts.

### Conectar ao Banco de Dados

1. Abra o Internet Explorer
2. Digite: <http://oracle/isqlplus>



3. Digite o “Nome do Usuário”, senha e o nome da base de dados (TESTE).
4. Pressione o botão “Login”

## Linguagem de Definição de Dados (DDL)

CREATE – cria quase todos os objetos.

ALTER – altera as definições dos objetos e configurações do banco de dados.

DROP – remove objetos do banco de dados.

RENAME – renomeia os objetos do banco de dados.

## Objetos do Banco de Dados

Os objetos são “unidades lógicas” dentro do banco de dados que utilizamos para armazenar dados e referenciá-los como banco de dados back-end. Cada objeto se localiza dentro de um esquema do banco de dados.

O que são objetos de banco de dados?

Um *objeto de banco de dados* é tudo aquilo que utilizamos para armazenar ou referenciar dados.

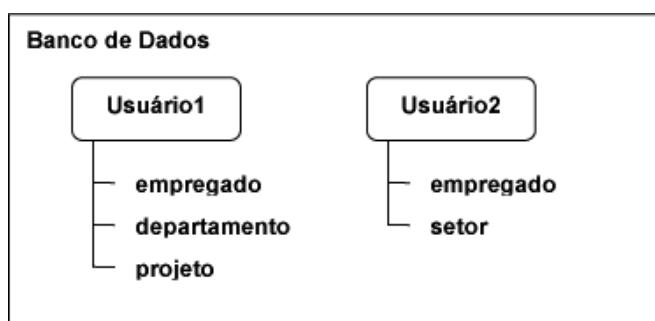
Quais são os objetos de um banco de dados?

Os objetos de um banco de dados podem ser: tabelas, visualizações, clusters, seqüências, índices e símônimos.

O que é um esquema?

Um esquema é o conjunto de objetos de um usuário do banco de dados chamados de proprietário de esquema.

A figura a seguir representa um banco de dados e os objetos que cada usuário possui. Cada usuário possui seu próprio esquema. O usuário 1 possui as tabelas empregado, departamento e projeto e o usuário 2 possui as tabelas empregado e setor. Ambos possuem as tabelas empregado. Em um mesmo banco de dados nomes de tabelas iguais só é possível se pertencerem a esquemas diferentes, portanto, nomes de tabelas em um banco de dados são sempre únicos já que cada tabela possui seu proprietário do esquema que a tabela faz parte.



O nome real da tabela empregado para o usuário 1 é **Usuário1.empregado**, porém, quando o proprietário do esquema acessa seus próprios objetos, não há necessidade de especificar o usuário, podendo este referenciar sua tabela “empregado” de duas maneiras:

- empregado

- Usuário1.empregado

Caso outro usuário fosse consultar qualquer tabela do Usuário1, além deste possuir permissão, este deve utilizar a segunda opção, colocando o nome do usuário antes da tabela desejada.

## Tabelas

Uma tabela possui linha(s) e coluna(s) para armazenar os dados. Cada tabela precisa ser armazenada em um espaço físico no banco de dados, este espaço pode ser permanente ou temporário.

A criação de tabelas é bem simples, porém, estas devem ser bem planejadas e definidas. Todas as tabelas que não possuem chaves estrangeiras são criadas antes das demais tabelas.

A criação das tabelas são determinadas de acordo com cada projeto, após a fase de Análise. Uma tabela corresponde a uma entidade no modelo conceitual.

Antes de criar as tabelas, defina as seguintes informações:

- Nome da Tabela
- Coluna(s) que comporá (ão) a chave primária
- Nome das Colunas
- Tipo de Dado das colunas
- Comprimento de cada coluna
- Quais colunas são obrigatórias seu preenchimento (Not Null)
- Chaves estrangeiras se possuir, de qual(is) tabela(s)

## Simbologia

O Oracle utiliza a seguinte simbologia para se referir aos elementos da estrutura de uma tabela.

**PK** – Primary Key = Chave Primária

**FK** – Foreign Key = Chave Estrangeira

**FK1, FK2** – Duas chaves estrangeiras na mesma tabela

**FK1, FK1** – Duas colunas formando uma chave estrangeira

**NN** – Coluna Not Null, ou seja, É Negado Nulo

**U** – Coluna Única

**U1, U1** – A combinação de duas colunas deve ser única.

**CK** – Checagem / Verificação

## Regras de Nomeação

Os nomes no Oracle das tabelas e das colunas devem seguir um certo critério:

Possuir no mínimo 1 e no máximo 30 caracteres

O primeiro caracter deve ser alfabético

## INTRODUÇÃO A BANCO DE DADOS

Deve conter somente os caracteres: a-z, A-Z, 0-9, \_ (underline), \$ e #<sup>1</sup>

Não devem ser iguais às palavras reservadas do Oracle.

Não pode ser igual ao nome de qualquer outro objeto criado pelo mesmo usuário Oracle.

### Tipos de Dados

Tipo de dado	Descrição
Number(x,y)	Valor número com no máximo x dígitos sendo y após a vírgula
Varchar2(X)	Texto (string) de tamanho variável até x caracteres, String = Conjunto de Caracteres
Varchar(x)	Igual ao Varchar2(x). Recomenda-se usar o tipo de dados Varchar2, ao invés de Varchar
Char(x)	Texto(String) de tamanho fixo com x caracteres(letras)
Date	Data e Hora entre 4712 a.C. e 4712 d.C. já pronta para enfrentar o bug de 2000!
Long	Parecido com o tipo de dados Varchar(2), com o tamanho máximo de 2GB. Com o tipo de dados Long pode-se armazenar textos longos, som e imagem de até 2Gb.

### Constraints

O Oracle possui constraints de integridade de dados para forçar a integridade dos dados.

Constraint	Descrição
Not Null	Determina que a coluna da Tabela não aceitará valores nulos, ou seja, deve ser obrigatório o preenchimento.
Unique	Determina que a(s) coluna(s) terão seus valores únicos.
Primary Key	Determina a <u>unicidade</u> de cada linha da tabela. Primary Key possui as constraints Not Null e Unique.
Foreign Key References	Informa que a coluna possui chave estrangeira, ou seja, esta coluna possui referências com a chave primária da tabela referenciada.
Check	É uma condição que deve ser atendida dependendo da regra do negócio.

## Criando Tabelas

### Sintaxe para Criação de Tabelas

```
CREATE TABLE NOME_TABELA
(NOME_COLUNA DATATYPE | 
 [ CONSTRAINT NOME_CONSTRAINT NOT NULL ]
 |
 [ CONSTRAINT NOME_CONSTRAINT UNIQUE ]
 |
 [ CONSTRAINT NOME_CONSTRAINT PRIMARY KEY ]
 |
 [ CONSTRAINT NOME_CONSTRAINT CHECK (CONDICAO)
 ] |
 [ CONSTRAINT NOME_CONSTRAINT FOREIGN KEY
      (NOME_COLUNA)
      REFERENCES
      TABELA (NOME_COLUNA) ] |
 [, NOME_COLUNA DATATYPE [CONSTRAINT...]
 ] )
```

### Sintaxe para Mudar o Nome da Tabela

```
RENAME NOME_TABELA TO NOVO_NOME
```

## INTRODUÇÃO A BANCO DE DADOS

### Sintaxe para Exibir a Estrutura da Tabela

```
DESC[RIIBE]      NOME_TABELA
```

### Sintaxe para Alteração da Tabela

Após a criação da tabela, podemos adicionar colunas, alterar o tipo de dados e o tamanho de uma coluna ou até mesmo incluir uma nova constraint a uma coluna através do comando ALTER TABLE.

```
ALTER TABLE NOME_TABELA
ADD (NOME_COLUNA DATATYPE
     [ CONSTRAINT NOME_CONSTRAINT NOT NULL ]
     | CONSTRAINT NOME_CONSTRAINT UNIQUE )
     | CONSTRAINT NOME_CONSTRAINT PRIMARY KEY
     | CONSTRAINT NOME_CONSTRAINT CHECK (CONDICÃO)
     | CONSTRAINT NOME_CONSTRAINT FOREIGN KEY
       (NOME_COLUNA) REFERENCES TABELA(NOME_COLUNA) )
     [, NOME_COLUNA DATATYPE [CONSTRAINT...]]
```

### Adicionar Colunas à Tabela

Ao adicionar colunas, podemos definir constraints no mesmo comando.

```
ALTER TABLE NOME_TABELA
ADD (NOME_COLUNA DATATYPE
     [ CONSTRAINT NOME_CONSTRAINT NOT NULL ]
     | CONSTRAINT NOME_CONSTRAINT UNIQUE )
     | CONSTRAINT NOME_CONSTRAINT PRIMARY KEY
     | CONSTRAINT NOME_CONSTRAINT CHECK (CONDICÃO)
     | CONSTRAINT NOME_CONSTRAINT FOREIGN KEY
       (NOME_COLUNA) REFERENCES TABELA(NOME_COLUNA) )
     [, NOME_COLUNA DATATYPE [CONSTRAINT ...]]
```

### Modificar Tipos de Dados de Tabelas

```
ALTER TABLE NOME_TABELA
MODIFY (NOME_COLUNA [DATATYPE] [NOT NULL]
        [, NOME_COLUNA [DATATYPE] [CONSTRAINT NOT
NULL ] ] )
```

### Remover Colunas de uma Tabela

É necessário colocar os parênteses, mesmo se for apagar uma única coluna da tabela.

```
ALTER TABLE NOME_TABELA
DROP (NOME_COLUNA [, NOME_COLUNA ] ...)
```

### Adicionar Constraints: Primary Key, Foreign Key, Unique e Check

O subcomando ADD CONSTRAINT:

NOME\_CONSTRAINT TIPO\_CONSTRAINT é usado somente para adicionar constraints de (PK, FK,

U) à colunas já existentes. A constraint NOT NULL é adicionada usando o comando MODIFY.

(NOME\_COLUNA [DATATYPE] [CONSTRAINT NOME\_CONSTRAINT].

```
ALTER TABLE NOME_TABELA
ADD CONSTRAINT NOME_CONSTRAINT UNIQUE (NOME_COLUNA)
| CONSTRAINT NOME_CONSTRAINT PRIMARY KEY (NOME_COLUNA)
| CONSTRAINT NOME_CONSTRAINT CHECK (NOME_COLUNA CONDIÇÃO)
| CONSTRAINT NOME_CONSTRAINT FOREIGN KEY
  (NOME_COLUNA) REFERENCES TABELA(NOME_COLUNA)
```

### Remover Constraints

```
ALTER TABLE NOME_TABELA
DROP CONSTRAINT NOME_CONSTRAINT
```

### Sintaxe para Remoção da Tabela

O comando **DROP TABLE nome\_tabela** elimina a estrutura da tabela inteira com todas as suas linhas. CUIDADO!!! Uma vez apagada uma tabela, não há como recuperá-la novamente.

```
DROP TABLE NOME_TABELA
```

onde:

**NOME\_TABELA** – é o nome da tabela.

**NOME\_COLUNA** – é o nome da coluna.

**DATATYPE** – é o tipo de dado da coluna.

**NOME\_CONSTRAINT** – é o nome da constraint para os dados da coluna. As constraints de integridade podem ser adicionadas após a criação da tabela, utilizando-se do comando ALTER TABLE, que veremos mais adiante.

**NOVO\_NOME** – é o novo nome dado à tabela.

### Exemplo para Criação de Tabelas

```
SQL > CREATE TABLE FUNCIONARIO
2 CD_EMPREGADO NUMBER(5),
3 TX_NOME VARCHAR2(30) CONSTRAINT FUNC_NOME_NN
NOT NULL,
4 TX_SOBRENOME VARCHAR2(30),
5 NU_SALARIO NUMBER(11,2),
6 CD_REGIAO NUMBER(3),
7 NU_COMISSAO NUMBER(4,2),
8 TX_OBSERVAÇÃO VARCHAR(255),
9 CONSTRAINT EMP_CD_PK PRIMARY KEY (CD_EMPREGADO),
10 CONSTRAINT EMP_NOME_UK UNIQUE (TX_NOME),
11 CONSTRAINT EMP_COMISSAO_CK CHECK (NU_TXCOMISSAO
IN(10,15,20)) ;
TABLE CREATED. Ou TABELA CRIADA.
```

### Exemplo para Mudar o Nome da Tabela

```
SQL > RENAME FUNCIONARIO TO EMPREGADO ;
TABLE RENAMED. Ou TABELA RENOMEADA.
```

### Exemplo para Exibir a Estrutura da Tabela

```
SQL > DESC EMPREGADO ;
Name          Null?    Type
-----+-----+-----+
CD_EMPREGADO NOT NULL NUMBER(5)
TX_NOME      NOT NULL VARCHAR2(30)
TX_SOBRENOME VARCHAR2(30)
NU_SALARIO   NUMBER(11,2)
CD_REGIAO    NUMBER(3)
NU_COMISSAO  NUMBER(4,2)
TX_OBSERVAÇÃO VARCHAR(255)
```

**Exemplo para Alterar uma Tabela Existente****Exemplo para Adicionar Colunas à Tabela**

A coluna adicionada sempre é incluída no final da tabela.

```
SQL > ALTER TABLE EMPREGADO
      ADD (DT ADMISSAO DATE
           CONSTRAINT EMP ADMISSAO_NN NOT NULL);

TABLE ALTERED. Ou TABELA ALTERADA.

SQL > DESC EMPREGADO;

Name          Null?    Type
-----        -----    -----
CD_EMPREGADO NOT NULL NUMBER(5)
TX_NOME       NOT NULL VARCHAR2(30)
TX_SOBRENOME VARCHAR2(30)
NU_SALARIO    NUMBER(11,2)
CD_REGIAO    NUMBER(3)
NU_COMISSAO   NUMBER(4,2)
TX_OBSERVAÇÃO VARCHAR(255)
DT ADMISSAO  NOT NULL DATE
```

**Exemplo para Modificar Tipos de Dados das Colunas da Tabela**

Além de modificar o tipo de dado de uma coluna, pode ser alterado se o atributo será NULL ou NOT NULL.

```
SQL > ALTER TABLE EMPREGADO
      MODIFY (TX_NOME VARCHAR2(40));

TABLE ALTERED. Ou TABELA ALTERADA.

SQL > DESC EMPREGADO;

Name          Null?    Type
-----        -----    -----
CD_EMPREGADO NOT NULL NUMBER(5)
TX_NOME       NOT NULL VARCHAR2(40)
TX_SOBRENOME VARCHAR2(30)
NU_SALARIO    NUMBER(11,2)
CD_REGIAO    NUMBER(3)
NU_COMISSAO   NUMBER(4,2)
TX_OBSERVAÇÃO VARCHAR(255)
DT ADMISSAO  NOT NULL DATE
```

**Exemplo para Remover Colunas da Tabela**

Ao remover uma coluna, todo seu conteúdo é eliminado também e não há como recuperar novamente.  
CUIDADO!

```
SQL > ALTER TABLE EMPREGADO
      DROP (TX_OBSERVACAO);

TABLE ALTERED. Ou TABELA ALTERADA.

SQL > DESC EMPREGADO;

Name          Null?    Type
-----        -----    -----
CD_EMPREGADO NOT NULL NUMBER(5)
TX_NOME       NOT NULL VARCHAR2(30)
TX_SOBRENOME VARCHAR2(30)
NU_SALARIO    NUMBER(11,2)
CD_REGIAO    NUMBER(3)
NU_COMISSAO   NUMBER(4,2)
DT ADMISSAO  NOT NULL DATE
```

**Exemplo para Adicionar Constraints à Tabela**

```
SQL > ALTER TABLE EMPREGADO
      ADD CONSTRAINT EMP_CD_REGIAO_U UNIQUE(CD_REGIAO);

TABLE ALTERED. Ou TABELA ALTERADA.

SQL > ALTER TABLE EMPREGADO
      ADD CONSTRAINT EMP_NOME_SOBRENOME_U UNIQUE(TX_NOME,
TX_SOBRENOME);

TABLE ALTERED. Ou TABELA ALTERADA.
```



```
SQL > CREATE TABLE REGIAO
      (CODIGO      NUMBER(2)    CONSTRAINT REGIAO_CODIGO_PK
       PRIMARY KEY,
       NOME        VARCHAR2(15) );
TABLE ALTERED. Ou TABELA ALTERADA.

SQL > ALTER TABLE EMPREGADO
      ADD CONSTRAINT EMP_CD_REGIAO_FK
           FOREIGN KEY (CD_REGIAO) REFERENCES
           REGIAO(CODIGO);
TABLE ALTERED. Ou TABELA ALTERADA.

SQL > ALTER TABLE EMPREGADO
      ADD CONSTRAINT EMP_SALARIO_CK
           CHECK(CD_REGIAO > 1000.00);
TABLE ALTERED. Ou TABELA ALTERADA.

SQL > ALTER TABLE EMPREGADO
      ADD CONSTRAINT EMP_COMISSAO_CK
           CHECK(NU_COMISSAO <= 0.2 * (NU_SALARIO));
TABLE ALTERED. Ou TABELA ALTERADA.
```

### Exemplo para Remover uma Constraint de uma Tabela

```
SQL > ALTER TABLE EMPREGADO
      DROP CONSTRAINT EMP_COMISSAO_CK;
TABLE ALTERED. Ou TABELA ALTERADA.
```

### Exemplo para Remover uma Tabela

```
SQL > DROP TABLE EMPREGADO ;
TABLE DROPPED. Ou TABELA ELIMINADA.
```

### Tabela User\_Constraints

A tabela *USER\_CONSTRAINT* é uma das tabelas criadas e gerenciadas pelo Oracle que contém informações sobre todas as “constraints” existentes no banco de dados.

COLUNA	DESCRÍCÃO
OWNER	Informa o nome do usuário que criou a constraint
CONSTRAINT_NAME	Nome da constraint
CONSTRAINT_TYPE	Indica o tipo da constraint. C-Check, P-Primary, U-Unique, R-Foreing Key
TABLE_NAME	Nome da tabela que a constraint pertence
SEARCH_CONDITION	É o texto contendo a condição determinada pela constraint CHECK
R_OWNER	Nome do usuário que é o dono da tabela
R_CONSTRAINT_NAME	Nome da chave primária da coluna referenciada pela constraint R-Foreing Key
DELETE_RULE	Determina a deleção da chave primária ou única conforme a chave estrangeira Esta opção é somente válida para as opções cascade e no_action
STATUS	Este valor é determinado e usado internamente pelo Oracle

- Se executarmos o comando “DESC USER\_CONSTRAINTS” veremos a estrutura desta tabela.
- Podemos utilizar o comando “SELECT” para visualizar seu conteúdo.

### Tabela User\_Cons\_Columns

A tabela *USER\_CONS\_COLUMNS* é uma das tabelas criadas e gerenciadas pelo Oracle que contém os nomes das colunas envolvidas em constraints.

Coluna	Descrição
<i>Owner</i>	Nome do usuário que criou a constraint
<i>Constraint name</i>	Nome da Constraint
<i>Table Name</i>	Nome da tabela que a constraint pertence
<i>Column Name</i>	Nome da coluna especificada na constraint.
<i>Position</i>	Posição da coluna especificada na constraint

- Se executarmos o comando “DESC USER\_CONS\_COLUMNS” veremos a estrutura desta tabela.
- Podemos utilizar o comando “SELECT” para visualizar seu conteúdo.

## Linguagem de Manipulação de Dados (DML)

Os comandos DML, ou seja, Linguagem de Manipulação de Dados (INSERT, UPDATE e DELETE) são utilizados para:

- Inserir dados na tabela (INSERT)
- Alterar dados na tabela (UPDATE)
- Remover dados da tabela (DELETE)

## Manipulando Tabelas

### Sintaxe para Inserir Dados na Tabela

DICA PARA INSERÇÃO!!!

- Determine quais colunas serão inseridos dados na tabela, usando o comando DESC.
- Ao inserir datas digite-as da seguinte forma: Ex.: ‘21-jun-1999’ ou utilize o comando SYSDATE para inserir a data e a hora corrente.
- Ao inserir valores nulos, a coluna deve permitir valores NULL
- Não incluir aspas para inserir valores numéricos.

```
INSERT INTO      NOME_TABELA [ (COLUNA1, COLUNA2...) ]
                VALUES      VALOR1, VALOR2...
```

### Sintaxe para Alterar Dados na Tabela

DICA PARA ALTERAÇÃO!!!

- Verifique o valor das linhas que deseja alterar usando o comando SELECT.
- Quando utilizar a cláusula WHERE para restringir as linhas a serem alteradas digite a palavra conforme ela foi inserida na tabela, ou seja, todas em maiúsculas, minúsculas, alternadas... e sempre entre aspas simples ‘ ’

```
UPDATE          NOME_TABELA
SET             COLUNA1=VALOR1 [, COLUNA2=VALOR2...]
WHERE           CONDIÇÃO
```

### Sintaxe para Apagar Dados na Tabela

DICA PARA REMOÇÃO!!!

- Verifique o valor das linhas que deseja remover usando o comando SELECT.
- Com a cláusula WHERE é possível remover várias linhas da tabela.
- **NUNCA omita a cláusula WHERE, pois se esta cláusula for omitida serão removidas TODAS AS LINHAS DA TABELA.**

```
DELETE FROM      NOME_TABELA
WHERE           CONDIÇÃO
```

onde:

**NOME\_TABELA** – é o nome da tabela.

**NOME\_COLUNA** – é o nome da coluna. Os nomes das colunas podem ser ignorados, porém, os valores devem ser digitados na ordem das colunas da tabela. Use o comando DESCRIBE para verificar a ordem da tabela.

**VALOR** – é o valor que está sendo inserido na coluna. Os valores do tipo VARCHAR2, CHAR e DATA devem estar entre ‘ ’ (aspas simples).

**CONDICÃO** – identifica a(s) linha(s) que devem ser alteradas.

### Exemplo para Inserir Dados na Tabela

```

SQL > DESC EMPREGADO ;
Name          Null ?    Type
-----        -----
CD_EMPREGADO NOT NULL NUMBER(5)
TX_NOME       NOT NULL VARCHAR2(40)
TX_SOBRENOME VARCHAR2(30)
NU_SALARIO    NUMBER(3)
CD_REGIAO    NUMBER(3)
NU_COMISSAO   NUMBER(4,2)
DT_ADMISSAO  NOT NULL DATE

SQL > INSERT INTO EMPREGADO
      VALUES (1,'Manoel','Oliveira', 950, null, null,
              SYSDATE);

1 ROW CREATED. Ou 1 linha criada.

SQL > INSERT INTO EMPREGADO(CD_EMPREGADO, TX_NOME,
                           TX_SOBRENOME, DT_ADMISSAO)
      VALUES (2,'Renata', ' ', '23-mai-2000');

1 ROW CREATED. Ou 1 linha criada.

SQL > SELECT TX_NOME, TX_SOBRENOME
      FROM EMPREGADO;

TX_NOME          TX_SOBRENOME
-----          -----
Manoel           Oliveira
Renata         

SQL > SELECT * FROM REGIAO;
não há linhas selecionadas

SQL > DESC REGIAO ;
Name          Null ?    Type
-----        -----
CODIGO       NOT NULL NUMBER(2)
NOME         VARCHAR2(15)

SQL > INSERT INTO REGIAO
      VALUES (1,'NORTE');
1 linha criada

SQL > INSERT INTO REGIAO
      VALUES (1,'SUL');
1 linha criada

SQL > INSERT INTO REGIAO
      VALUES (2,'LESTE');

```

### Exemplo para Alterar Dados na Tabela

<pre> SQL &gt; DESC EMPREGADO ; Name          Null ?    Type -----        ----- CD_EMPREGADO NOT NULL NUMBER(5) TX_NOME       NOT NULL VARCHAR2(40) TX_SOBRENOME VARCHAR2(30) NU_SALARIO    NUMBER(3) CD_REGIAO    NUMBER(3) NU_COMISSAO   NUMBER(4,2) DT_ADMISSAO  NOT NULL DATE  SQL &gt; UPDATE EMPREGADO       SET TX_SOBRENOME='ZOMER', DT_ADMISSAO='22-mai-2000'       WHERE TX_NOME='Renata';  1 ROW UPDATED. Ou 1 linha atualizada.  SQL &gt; SELECT TX_NOME, TX_SOBRENOME FROM EMPREGADO; TX_NOME          TX_SOBRENOME -----          ----- Manoel           Oliveira Renata          Zomer </pre>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Exemplo para Apagar os Dados da Tabela**

```

SQL > DESC EMPREGADO ;
Name          Null ?    Type
-----
CD_EMPREGADO NOT NULL NUMBER(5)
TX_NOME       NOT NULL VARCHAR2(40)
TX_SOBRENOME
NU_SALARIO
CD_REGIAO
NU_COMISSAO
DT_ADMISSAO  NOT NULL DATE

SQL > DELETE FROM EMPREGADO
      WHERE TX_NOME='Renata';

1 ROW DELETED. Ou 1 linha deletada.

SQL > SELECT TX_NOME, TX_SOBRENOME FROM EMPREGADO;
TX_NOME           TX_SOBRENOME
-----            -----
Manoel            Oliveira

```

**Notas Complementares****Valores Nulos**

Os valores nulos indicam “ausência de valor”, ou seja, vazio. Eles podem ser inseridos nas tabelas cuja coluna permite valores nulos, ou seja, colunas que não foram criadas com a opção NOT NULL. Estes podem ser de três maneiras:

Método	Descrição
<i>Explícito</i>	Quando digitamos a palavra “NULL”
<i>Explícito</i>	Quando digitamos uma StringVazia da seguinte maneira:, ou seja, significa NULL. Usando somente para valores String e Datas
<i>Implícito</i>	A maneira implícita de inserir valores nulos e omitir a coluna que deseja que seja nula na lista de colunas selecionadas pelo comando de inserção

**Recuperação de Dados****Recuperação de Dados em um Única Tabela****Sintaxe:**

```

SELECT [DISTINCT]          (COLUNAS ["Alias"])
      FROM   (TABELA)
      WHERE  (CONDIÇÃO)
      ORDER BY (EXPRESSÃO OU CHAVE)

```

onde:

**COLUNAS** – determina as colunas desejadas na pesquisa, separadas por vírgula.

**DISTINCT (opcional)** – evita que na listagem a mesma ocorrência se repita.

**AS ALIAS (opcional)** – é um apelido para a coluna. O comando AS não é obrigatório. Se o alias possui espaços, use “” (aspas duplas) que o Oracle respeita o formato digitado.

Observações:

Dentro do comando SQL um alias pode ser usado tanto pelo comando SELECT como pelo ORDER BY.

Não se pode usar alias com o comando WHERE.

Pode-se usar este método para mudar o título das colunas normais para exibição.

**TABELA** – determina a(s) tabela(s) que serão utilizadas nesta pesquisa.

**WHERE (opcional)** – determina a condição para filtrar algumas linhas dentre as selecionadas.

**ORDER BY (opcional)** – determina em que ordem deverá ser apresentada a pesquisa desejada (crescente ou decrescente) não esquecendo as colunas que deverão ser ordenadas.

## Exemplos:

### Selecionar Todas as Colunas da Tabela:

```
SQL > SELECT * FROM REGIAO ;

Código      Nome
-----
1          NORTE
2          SUL
3          LESTE
4          OESTE
```

### Selecionar Colunas Específicas:

```
SQL > SELECT * FROM REGIAO ;

Código      Nome
-----
1          NORTE
2          SUL
3          LESTE
4          OESTE
```

### Selecionar Linhas Específicas:

```
SQL > SELECT TX_NOME FROM EMPREGADO
      WHERE DT ADMISSAO > '01-JAN-2000' ;

TX_NOME
-----
Manoel

SQL > SELECT * FROM REGIAO
      WHERE NOME LIKE '%E' ;

Código      Nome
-----
1          NORTE
3          LESTE
4          OESTE
```

### Selecionar Linhas Utilizando Cálculos e Concatenação de Colunas:

```
SQL > SELECT TX_NOME || ' ' || TX_SOBRENOME NOME,
      NU SALARIO*12 "Salario Anual"
      FROM EMPREGADO;

NOME           Salario Anual
-----
Manoel Oliveira    11400
```

## Notas Complementares

### Operadores de Comparação

Operador	Descrição
=	Igual
<>	Diferente
>	Maior que
<	Menor que
>=	Maior igual
<=	Menor igual
Between<valor_inicial> AND <valor_final>	Comparação entre valores(Inicial_final)
Not Between <valor_inicial> AND <valor_final>	Não comparação entre valores(Inicial_final)
In(List)	Igual a qualquer membro da Lista
Not In(List)	Not igual a qualquer membro da Lista
Like	Realiza uma busca aproximada
Not Like	Contrário ao Like

**Titulação Padrão das Colunas (default)**

Alinhamento de títulos

Esquerda: datas e texto

Direita: numéricos

Título todo em maiúsculas (caixa alta)

Títulos de colunas data e texto podem ser parcias, mas numéricos sempre estarão completos.

Outras formas podem ser usadas através de *aliases*.

**Observações Importantes para Colunas Calculadas:**

Usar o nome da coluna como argumento variável.

Recomenda-se usar espaços entre os argumentos aritméticos.

Os valores resultantes só existem para exibição, a tabela original não é alterada.

Os operadores têm precedência (a expressão **não** é avaliada da esquerda para a direita):

Multiplicação e Divisão tem prioridade sobre adição e subtração.

Operadores com a mesma precedência são avaliados da esquerda para a direita

Pode-se usar parênteses para forçar precedência ou facilitar a leitura

Exemplo: 12 \* nu\_comissao + 100 é inteiramente diferente de 12 \* (nu\_comissao + 100)

**Linhas Duplicadas**

Normal (padrão) é apresentação de todas as linhas, inclusive as duplicadas.

Podemos usar o comando **DISTINCT** para apresentar valores exclusivos na cláusula *SELECT*

<b>SQL &gt; SELECT nome FROM departamento</b>	<b>SQL &gt; SELECT DISTINCT nome FROM departamento</b>
NOME	NOME
-----	-----
Financeiro	Administrativo
Operacional	Operacional
Operacional	Financeiro
Administrativo	Operacional
4 linhas selecionadas	3 linhas selecionadas

**Recuperação de Dados em Várias Tabelas**

Em determinadas situações será necessário consultar dados utilizando duas ou mais tabelas.

Há três maneiras de realizar consultas em múltiplas tabelas: SIMPLE JOIN, OUTER JOIN ou SELF JOIN.

Sempre que utilizar o comando *SELECT* com múltiplas tabelas, digite o nome da coluna e o nome da tabela.

São maneiras mais específicas de se usar o comando *SELECT*

**Simple Join**

Utilizaremos SIMPLE JOIN quando a especificação de ligação entre as tabelas é simples.

**Sintaxe:**

```
SELECT TABELA.COLUNA, TABELA.COLUNA...
  FROM TABELA1, TABELA2...
 WHERE TABELA1.COLUNA = TABELA2.COLUNA
```

onde:

**TABELA.COLUNA** – é o nome da tabela e o nome da coluna que será selecionada.

**TABELA1.COLUNA = TABELA2.COLUNA** – indica a condição de ligação das relações.

## Exemplo do “Simple Join” – Ligação Simples:

```

SQL > SELECT
  FUNCIONARIO.TX_NOME, FUNCIONARIO.CD_REGIAO, REGIAO.TX_REGIAO
  2 FROM FUNCIONARIO, REGIAO
  3 WHERE FUNCIONARIO.CD_REGIAO = REGIAO.CD_REGIAO;

  TX_NOME          CD_REGIAO TX_REGIAO
  -----
ANTONIO            1           NORTE
LUIZ              1           NORTE
JOANA             2           SUL
MARIA             3           LESTE
JOAO              3           LESTE
MARCIO            4           OESTE
ELAINE            4           OESTE

7 ROWS SELECTED.

SQL > SELECT F.TX_NOME, F.CD_REGIAO, R.TX_REGIAO
  2 FROM FUNCIONARIO F, REGIAO R
  3 WHERE F.CD_REGIAO = R.CD_REGIAO;

  TX_NOME          CD_REGIAO TX_REGIAO
  -----
ANTONIO            1           NORTE
LUIZ              1           NORTE
JOANA             2           SUL
MARIA             3           LESTE
JOAO              3           LESTE
MARCIO            4           OESTE
ELAINE            4           OESTE

7 ROWS SELECTED.

```

## Claúsula Distinct

Quando um select retornar linhas repetidas devido a consultas de múltiplas tabelas, utilize a cláusula DISTINCT para suprimir as linhas repetidas.

```

SQL > SELECT DISTINCT FUNC.CD_FUNCIONARIO "COD. FUNC.",
  2                   REG.CD_REGIAO "COD. REGIAO",
  3                   REG.TX_REGIAO "NOME REGIAO"
  4   FROM FUNCIONARIO FUNC, REGIAO REG
  5 WHERE FUNC.CD_REGIAO = REG.CD_REGIAO;

  COD. FUNC      COD. REGIAO NOME REGIAO
  -----
1                  1           NORTE
2                  1           NORTE
3                  2           SUL
4                  3           LESTE
5                  3           LESTE
7                  4           OESTE
8                  4           OESTE

7 ROWS SELECTED.

```

```

SQL > SELECT F.TX_NOME, F.CD_REGIAO, R.TX_REGIAO
  2 FROM FUNCIONARIO F, REGIAO R
  3 WHERE F.CD_REGIAO = R.CD_REGIAO
  4 AND F.CD_REGIAO=1;

  TX_NOME          CD_REGIAO TX_REGIAO
  -----
ANTONIO            1           NORTE
LUIZ              1           NORTE

2 ROWS SELECTED.

```

## Outer Join

Utilizaremos OUTER JOIN quando não há uma ligação direta das linhas de uma tabela com as linhas de outra tabela.

```

SELECT TABELA.COLUNA, TABELA.COLUNA...
FROM TABELA1, TABELA2....
WHERE TABELA1.COLUNA = TABELA2.COLUNA (+)

ou

SELECT TABELA.COLUNA, TABELA.COLUNA...
FROM TABELA1, TABELA2....
WHERE TABELA1.COLUNA(+) = TABELA2.COLUNA

```

onde:

**TABELA.COLUNA** – é o nome da tabela e o nome da coluna que será selecionada.

**TABELA1.COLUNA = TABELA2.COLUNA** – indica a condição de ligação das relações.

(+) – o operador “mais” indica que se o dado existir somente em uma das tabelas, ele deverá retornar as linhas, mesmo com a falta do outro valor.

#### Exemplo do “OuterJoin” – Ligação Indireta:

	COD. CLIENTE	NOME CLIENTE	COD. PEDIDO	PEDIDO
1		PATRICIA		
2		CRISTINA	35	
2		CRISTINA	50	
2		CRISTINA	51	
5		EDUARDO	25	
6		ANTONIO		
7		VERA	30	

7 ROWS SELECTED.

#### Self Join

Utilizaremos SELF JOIN quando precisamos ligar uma tabela com ela mesma.

#### Exemplo do “Self-Join” – Auto-Ligação:

	COD. FUNCIONARIO	NOME FUNCIONARIO	NOME GERENTE
1		ANTONIO	
2		LUIZ	ANTONIO
3		JOANA	ANTONIO
4		MARIA	
5		JOAO	
6		MARCIO	MARIA
7		ELAINE	JOAO

7 ROWS SELECTED.

- INTRODUÇÃO A BANCO DE DADOS
- COUGO, P., *Modelagem Conceitual e Projeto de Bancos de Dados*. Rio de Janeiro: Campus, 1997.
- MULLER, R. J., *Projeto de Banco de Dados, usando UML para modelagem de dados*. São Paulo: Berkeley Brasil, 2002.
- ULLMAN, J. D., MOLINA, H. G. e WIDOW, J. *Implementação de Bancos de Dados*. Rio de Janeiro: Campus, 2001.
- SILBERSCHATZ, A., KORTH, H. F. e SUDARSHAN, S. *Sistema de Banco de Dados*. São Paulo: Makron Books, 1999.
- SETZER, W. W. 3. ed. *Banco de Dados – Conceitos, Modelos, Gerenciadores, Projeto Logico e Projeto Físico*. 3. ed. São Paulo: Ed. Edgard Blucher, 2000.

## Bibliografia

- HEUSER, C. A., *Projeto de Banco de Dados*. 3. ed. Porto Alegre: Ed. Sagra Luzzatto, 2000.
- ABREU, M. M. F., *Projeto de Banco de Dados – Uma visão prática*. São Paulo: Erica, 1996.
- DATE, C. J., *Introdução a Sistemas de Banco de Dados*. 7. ed. Rio de Janeiro: Campus, 2000.
- ELMASRI, R. e NAVATHE, S. B., *Sistemas de Banco de Dados – Fundamentos e Aplicações*. 3. ed. Rio de Janeiro: LTC, 2002.
- HARRINGTON, J. L., *Projetos de Banco de Dados Relacionais: teoria e prática* – 2. ed. Rio de Janeiro: Campus, 2002.