

# Modelagem de Dados no Access

Neste e nos próximos artigos vou falar um pouco sobre um assunto que muitos desconhecem: a Modelagem de Dados.

Primeiramente, o que é modelagem de dados? Para quem não sabe, a modelagem de dados é um processo no qual você "projeta" ou "planeja" a sua base de dados de forma que você possa aproveitar os recursos do Gerenciador de Banco e também para que possa construir um banco de dados consistente, que reaproveite recursos, que exija menos espaço em disco e, sobretudo, que possa ser bem administrado.

Assim, como no processo de software, a modelagem de dados é um processo que possui etapas a serem seguidas, mas que podem ser superadas, dependendo do tipo de banco que se pretende construir. O documento principal da modelagem de dados é o Diagrama de Entidade-Relacionamento - DER (leia-se: dér) ou Modelo de Entidade-Relacionamento (MER). Neste documento, são representadas as entidades e os relacionamentos entre elas. As entidades são os "embriões" das tabelas do banco. Até avançarmos esta fase da modelagem, elas recebem esta nomenclatura.

Esta primeira fase é o que chamamos de Modelagem Lógica. É quando determinamos o fluxo de dados entre as entidades, isto é, como o próprio nome diz, quando determinamos a lógica do banco que iremos contruir.

O relacionamento entre as entidades é um quesito que deve ser especialmente analisado. No modelo lógico, toda entidade deve estar relacionada à outra. Quando sobram entidades sem relacionamento, é sinal de que há algum problema. Podem ser entidades que estão sobrando, ou seja, que na verdade não deveriam existir, ou alguma entidade pode estar relacionada à qual não deveria.

Dependendo do tipo de base de dados que se deseja, pode-se aproveitar ferramentas do próprio SGBD e, desta forma, economizar linhas de código.

Suponha que façamos um controle de bens domésticos. Certamente para este sistema não há previsão de migrar a base de dados para uma plataforma maior, como o SQL Server, ou Oracle, certo? Então por que não aproveitar alguns recursos do SGBD Access para controlar os seus dados? Isto deve ser levado em conta quando se modela um banco. Mas há também casos onde se prevê uma migração ou, quem sabe, se está apenas pensando em um módulo de um sistema.

Neste caso, quanto menos dependência do gerenciador do banco, melhor. Pode-se implementar rotinas no próprio sistema e torná-lo "universal" a qualquer tipo de banco de dados, seja ele proprietário (Access, SQL Server, Oracle) ou livre (MySQL, PostgreSQL).

Em plataformas como a Oracle, há módulos de modelagem de dados próprios, totalmente integrados ao banco de dados. Como sabemos, este não é o caso do Access. Mesmo assim, podemos fazer uma análise, por mais simples que seja o banco de dados, antes de colocar Access para rodar.

Após terminarmos a modelagem lógica, partimos para a modelagem física. Nesta etapa, vamos determinar as tabelas, campos e relacionamentos que efetivamente vamos contruir em nosso banco de dados. Para isto, vamos repensar o modelo lógico que criamos e adequá-lo para a "realidade".

Após esta visão geral, vamos nos aprofundar mais na modelagem voltada para o Access.

# Modelagem de Dados 1: Entidades

Olá pessoal! Depois da visão geral ([artigo anterior](#)), vamos por a mão na massa e iniciar o Modelo Lógico do nosso banco de dados. Para que os conceitos fiquem bem visíveis, vou exemplificar para vocês um banco de dados para um sistema de Biblioteca.

O start da modelagem se dá a partir das **ENTIDADES**. Uma entidade é uma representação de um conjunto de informações sobre determinado conceito do sistema. Toda entidade possui **ATRIBUTOS**, que são as informações que referenciam a entidade.

Para exemplificar no sistema de controle de Biblioteca, partimos do conceito principal que é o empréstimo de obras por usuários da biblioteca. A partir deste conceito inicial, vamos ramificando e descobrindo novos conceitos. Podemos iniciar nosso raciocínio da seguinte forma:

"Uma biblioteca possui **Obras literárias** que podem ser tomadas em **empréstimos** pelos **usuários** credenciados."

Podemos rapidamente enxergar um *cadastro de livros*, um *cadastro de usuários* e um *registro de empréstimos*, certo? É essa visão que temos que ter ao modelarmos um banco, isto é, devemos detectar as informações que devemos armazenar.

Para identificar se aquele conceito pode ser uma entidade você deve apenas se perguntar: "*Eu desejo armazenar quais informações sobre este conceito ?*" Se houverem informações a serem armazenadas, você tem uma **ENTIDADE**. Exemplificando: Eu desejo armazenar os seguintes dados do livro: Título, Autor, Editora, Ano, Edição e Volume. Temos então a entidade Livro.

A cada uma destas informações que armazenamos, damos o nome de **ATRIBUTO**. Um atributo pode ser uma informação única, bem como pode ser um conjunto de informações. Exemplificando: Sobre empréstimos, eu tenho os seguintes atributos: *Código, livro emprestado, usuário que emprestou, data de empréstimo e data de devolução*. O atributo "livro emprestado" refere-se ao livro, porém sabemos que há informações que devem ser armazenadas sobre livros como vimos antes. Temos aí um exemplo de um atributo que é um conjunto de outros atributos: todos os atributos da entidade Livro, formam um atributo da entidade Empréstimo.

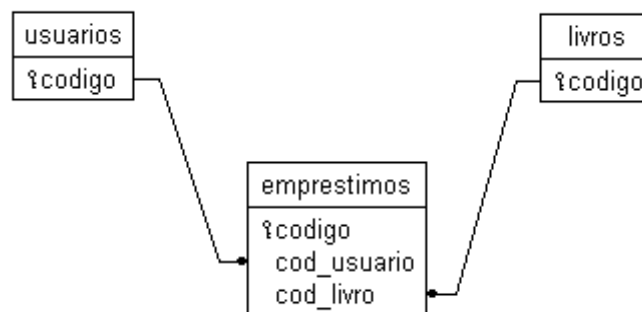
Outra situação é a seguinte: deseja-se armazenar 2 números de telefone para cada usuário. Telefone é um dos atributos da entidade Usuário. Neste caso, não temos referência à nenhuma outra entidade, ou seja, temos mais de uma informação para o mesmo atributo. A este atributo damos o nome de **ATRIBUTO MULTIVALORADO**, pois temos 2 valores para o mesmo atributo em uma mesma entidade. Sabemos que nos bancos de dados não é possível armazenar mais de uma informação no mesmo campo. Por isso, veremos mais à frente uma solução para os atributos multivalorados.

Quando os atributos de uma entidade formam o atributo de outra, poderemos dizer que existem uma referência entre as entidades. Naquele atributo da entidade Empréstimos vamos armazenar apenas uma referência à entidade Livro. O mesmo ocorrerá com relação à entidade Usuário. Pelo simples fato de existir esta referência de uma entidade em outra, temos então o que chamamos de **RELACIONAMENTO**.

Neste começo, temos um rascunho do nosso Diagrama de Entidade-Relacionamento:

**Entidades:** Usuário, Livro, Empréstimo

**Relacionamentos:** Usuário - Empréstimo, Livro - Empréstimo



## Modelagem de Dados 2 - Os Relacionamentos

Agora que definimos nossas entidades, vamos falar sobre os relacionamentos entre elas.

Na matéria anterior vimos como identificar os relacionamentos entre entidades, porém devemos observar alguns aspectos importantes que sinalizam erros na modelagem:

**01.** Quando "sobram" entidades sem relacionamentos;

**02.** Quando ocorrem "ciclos fechados" de relacionamentos;

Exemplo: Usuário relaciona-se com Empréstimo que relaciona-se com Livro que relaciona-se com Usuário que relaciona-se com Empréstimo, etc...

**03.** Entidades com muitos atributos relacionando-se com entidades com apenas alguns atributos;

**04.** Muitas entidades relacionando-se à uma mesma entidade;

É importante atentar para esses erros para que não haja acúmulo de inconsistências e que não torne a modelagem um processo problemático. Mas não se preocupe em resolver tudo de uma vez só, pois mais a frente veremos formas de VALIDAR nossa modelagem, de maneira que os erros não passem despercebidos.

Determinados os relacionamentos, temos que verificar o número de referências de uma entidade em outra, ou seja, agora vamos verificar a CARDINALIDADE dos relacionamentos. Vejamos as possibilidades:

### - Relacionamento Um-Para-Um (1:1)

Uma instância da entidade A relaciona-se a uma instância da entidade B

### - Relacionamento Um-Para-Vários (1:N)

Uma instância da entidade A relaciona-se a várias instâncias da entidade B

### - Relacionamento Vários-Para-vários (N:M)

Várias instâncias da entidade A relacionam-se a várias instâncias da entidade B

Estas 3 cardinalidades apresentadas acima são implicitamente, CARDINALIDADES MÁXIMAS, mas pode-se determinar a CARDINALIDADE MÍNIMA, que pode ser descrita desta forma:

### - Relacionamento Um-Para-Vários (0,1:1,N)

Nenhuma ou uma instância da entidade A relaciona-se a uma ou várias instancias da entidade B.

Exemplo de relacionamento Um-Para-Vários com cardinalidade mínima:

Usuario - Empréstimo (1,1:0,N)

- Um usuário pode estar relacionado a nenhum ou a vários empréstimos.
- Um empréstimo deve estar relacionado a somente um usuário.

**Conclusão:** Pode ser que um usuário nunca faça empréstimos, assim como pode haver usuário que faça vários empréstimos, porém um empréstimo obrigatoriamente tem que ser feito por um único usuário.

Exemplo de relacionamento Vários-Para-Vários com cardinalidade mínima:

Empréstimo - Livro (0,N:1,N)

- Um empréstimo pode estar relacionado a um ou a vários livros.
- Um livro pode estar relacionado a nenhum ou a vários empréstimos.

**Conclusão:** Pode ser que um livro nunca seja emprestado, assim como pode haver livros que tenham sido emprestado várias vezes, porém um empréstimo deve conter pelo menos um livro ou pode conter vários.

A Cardinalidade Mínima pode ser incluída no Modelo Lógico, mas é pouco utilizada por ser, muitas vezes, redundante e óbvia, mas muito é útil no que refere à expôr a clareza dos relacionamentos entre entidades.

Se você está em dúvida quanto á quem é o lado "Um" e quem é o lado "Vários" no seu relacionamento, use as seguintes dicas:

Relacionamentos 1 para N: no 1 o campo é a PK. No N o campo é a FK.

Relacionamentos 1 para 1: no 1 o campo é a PK. No outro 1 o campo é também uma PK.

Relacionamentos N para N: no primeiro N o campo é FK. No outro N o campo é também uma FK.

- Veja em qual das entidades está o atributo que faz referencia à outra entidade. Esta entidade onde está o atributo é o lado "Vários" a outra é o lado "Um", trata-se então de um relacionamento Um-Para-Vários.

- Se ambas as entidades tiverem atributos que referenciam uma à outra, então ambas possuem a cardinalidade "Varios", isto é, trata-se de um relacionamento Vários-Para-Vários.

Estas 2 regrinhas funcionam como fórmulas. Apenas aplique-as ao seu relacionamento, sem pensar muito. Mesmo que fique um pouco confuso para você, mais pra frente verá que elas são válidas.

### **Descrição do nosso Diagrama de Entidade-Relacionamento:**

Entidades: Usuário, Empréstimo, Livro

Relacionamentos: Usuario-Emprestimo(1:N), Emprestimo-Livro(N-N)

Na próxima matéria mostrarei para vocês o Diagrama de Entidade-Relacionamento propriamente dito e as ferramentas de software que você pode utilizar na modelagem do seu banco de dados.

## **Modelagem de Dados - Validação do modelo ER**

Nesta terceira parte da nossa série, vou falar sobre a validação do Diagrama Entidade-Relacionamento: o DER. Tendo definido as entidades, seus atributos e relacionamentos, vamos fazer uma verificação em nosso modelo ER em busca de falhas. É nesta fase que vamos validar nosso modelo ER e corrigir falhas em relacionamentos e possíveis entidades que deveriam ou não existir.

Nesta parte, também começamos a visualizar o modelo físico do banco de dados, onde as entidades viram tabelas e os atributos viram campos. Abaixo estão descritos os erros mais frequentes

ocorridos no modelo:

**Associações Incorretas:** No modelo ER sempre pensamos nas associações entre entidades não entre atributos. Seria incorreto por exemplo associar ao Livro (voltando ao nosso modelo Biblioteca) o nome do usuário que o tomou emprestado. Não se pode associar o nome do usuário ao livro, e sim o usuário como um todo. Verifique se não há associações entre atributos no seu modelo. Faça sempre associações entre Entidades.

Usar uma entidade como atributo de outra: No modelo lógico, as associações ainda se dão de forma abstrata. É incorreto colocar na entidade Empréstimo o atributo Usuário, sendo que Usuário é uma entidade associada e não um atributo e isso vale para qualquer outra entidade relacionada. Essa regra costuma gerar muita controvérsia porque costumamos confundir o conceito com o modelo físico onde criamos as Chaves - mas este assunto veremos mais à frente.

Usar o número incorreto de entidades em um relacionamento: Verifique sempre os casos de mais de uma entidade associada à mesma entidade. Nestes casos, o que surge é um relacionamento redundante e desnecessário.

Depois de verificados e corrigidos os erros nas associações (ou relacionamentos) devemos verificar se o modelo está completo. Nele devem ser expressadas todas as entidades, com seus atributos e relacionamentos. Para isso, verifique se é possível obter todas as informações desejadas a partir do modelo construído e também se todas as transações de dados podem ser executadas. Se tudo estiver ok, partimos para o próximo passo.

**Verificar redundâncias:** Um modelo de banco de dados deve ser "mínimo", ou seja, não apresentar nenhum tipo de redundância. Para verificar a redundância nas transações com dados, analise a função de cada relacionamento e reveja os que fizerem operações muito semelhantes ou iguais. Verifique também se há ciclos associativos, por exemplo:

**"A é associado a B que é associado a C que é associado a A"**

Neste caso, há uma redundância na parte "C que é associado a A", pois C já é associado a A através da entidade B, ou seja, este exemplo representa uma associação desnecessária e pode ser descartada sem nenhum prejuízo para o modelo. O correto seria: "A é associado a B que é associado a C"

Além disso, podem haver atributos redundantes nas entidades. Atributos redundantes são aqueles que podem ter uma nomenclatura diferente porém eles armazenam os mesmo dados, a mesma informação. Verifique a existência destes atributos e elimine-os do modelo.

Outro item considerável é o aspecto de tempo do banco de dados. Quando construímos modelo ER, pensamos somente na situação momentânea do banco de dados. Para corrigir isso, devemos verificar atributos e relacionamentos que podem ser alterados durante a utilização do banco de dados.

Um exemplo ótimo de atributo, porém fora do nosso caso da Biblioteca, é o Empregado.

Suponha a existência de uma entidade Empregado e o atributo Salário. A partir dessa entidade, pode-se criar uma entidade Salário com o atributo Data. Desta forma, obtem-se um histórico de salários do empregado.

Um exemplo de associação pode ser a associação entre Empregado e Departamento. Adicionando um atributo Data neste relacionamento, teremos um histórico de departamentos por onde o Empregado passou.

Também é importante levar em conta o armazenamento de informações antigas. Para evitar o crescimento demasiado do banco de dados, podem ser eliminadas informações antigas ou podem ser reinseridas no banco de outra forma. Por isso, é importante pensar no caso de remoção de dados da base, que outras informações seriam comprometidas. Deve-se planejar como guardar informações antigas ou que com o passar do tempo não venham mais a ser utilizadas, como, por exemplo, informações que serão usadas somente para cálculos estatísticos ou visões.

Pode ocorrer também a existência de entidades isoladas, isto é, sem nenhuma associação. A ocorrência destas entidades não é de todo incorreto, mas, na prática, elas acabam sendo descartadas, por isso verifique se ela não faz parte de alguma associação incorreta ou que não tenha sido feita.

Lembre-se que por mais simples que seja seu modelo ER, você deve fazer a validação completa. Este é o momento de corrigir erros e falhas de modelagem porque quando identificadas na modelagem física você acaba tendo que voltar a esse passo para fazer a arrumação do problema.

No proximo artigo, começamos a falar sobre a Modelagem Física.

## Modelagem de Dados (Parte 04) - Abordagem Relacional

A grande maioria dos Sistemas Gerenciadores de Banco de Dados (ou simplesmente SGBD's) são Relacionais. Os bancos Relacionais são compostos por Tabelas ou Relações. Este termo "Relações" é mais usado na literatura original sobre abordagem relacional (daí a denominação "Relacional"), enquanto que "Tabela" é um termo mais prático e mais usado em produtos comerciais. Para não haver confusão de conceitos, vou usar a terminologia "Tabela".

Uma Tabela é um conjunto não ordenado de linhas ou (tuplas - terminologia original) e cada linha possui campos (atributos). Cada campo é identificado por um nome de campo (ou nome de atributo) e o conjunto de linhas de uma tabela, que possuem o mesmo nome chama-se coluna. Para muitos, nada disso é novidade, mas não nos custa nada lembrar os conceitos.

Modelo lógico: relação, atributo, tupla

Modelo físico: tabela, campo, registro

### Algumas características específicas de uma tabela de banco de dados:

- As linhas da tabela não são ordenadas. A recuperação de linhas em uma tabela acontece arbitrária, ou seja, a recuperação será da forma como está no banco e pronto, a não ser que a ordem seja especificada pelo programador na sua consulta.

- Os valores de campo de uma tabela de banco de dados são mono-valorados. Pode-se apenas ter 1 valor em um campo, não sendo possível armazenar "coleções" como Arrays (ou Vetores) por exemplo.

Obs.: No PostgreSQL um campo pode armazenar informações do tipo array.

- A linguagem de consulta ao banco de dados permite o acesso por qualquer critério envolvendo os campos de uma ou mais linhas. Em um arquivo de dados por exemplo é necessário um índice ou esquema de ponteiros para buscar algum valor. Na verdade índices também existem nos bancos de dados, mas isso não é considerado pelo programador nas consultas às tabelas.

Um item primordial quando falamos sobre relacionamento entre linhas de tabelas é a CHAVE. A chave primária é formada por um ou mais campos de uma tabela e serve como identificador de uma linha. Porém a chave primária deve ser sempre mínima, ou seja, ser composta pelo menor número de campos possível (no mínimo 1). Veremos mais adiante o uso de chaves com mais de um campo, que recebem o nome de "chave composta".

Vistos os conceitos básicos de um banco relacional, temos que considerar alguns pontos antes de iniciarmos a transformação de um modelo ER em um modelo físico para o banco de dados:

- O modelo ER não considera implementação com nenhum SGBD. É muito comum, surgirem modificações no esquema lógico para possibilitar a criação do modelo físico e usar os recursos do SGBD.

- Ao construir o seu banco físico, você deve considerar fatores como:

- Diminuir o número de acessos a disco: A cada consulta à tabela, todos os campos da linha são carregados para a memória, mesmo que você utilize apenas 1 deles)

- Evitar junções: Em muitas consultas ao banco, são feitas junções de dados em várias linhas de tabelas. Utilize todos os dados necessários de preferência em uma única linha diminuindo o número de junções, pois uma junção acaba envolvendo vários acessos a disco.

- Diminuir o número de chaves primárias: Fatalmente voce acabará juntado algumas tabelas que você dividiu durante a modelagem lógico. Isso porque a presença de chaves em locais diferentes, armazenando a mesma informação acaba virando mais espaço em disco utilizado e mais processamento.

- Evitar campos opcionais: Tecnicamente um campo vazio no banco de dados não ocupa espaço em disco, devido às técnicas de compressão de dados existentes nos SGBDs de hoje. Mas o problema surge quando a obrigatoriedade ou não do preenchimento de um campo depende do valor de outro campo.

Este fator acaba sendo resolvido via programação, ou seja, pelo sistema que vai acessar aquele banco e isso deve ser evitado. Algumas medidas de integridade de dados podem ser tomadas já no banco, deixando o mínimo de campos que podem assumir o valor NULL.

A transformação do modelo ER em um modelo relacional segue as seguinte etapas:

- Tradução de Entidades e Atributos
- Tradução de Relacionamentos e respectivos atributos
- Tradução de generalizações/especializações

Inicialmente, a tradução de entidades é razoavelmente óbvia: uma entidade gera uma tabela. Cada atributo da entidade gera uma coluna e os atributos identificadores da entidade tornam-se chave primária. Essa é uma tradução inicial. No decorrer da transformação entre modelos, algumas tabelas ainda poderão ser fundidas ou divididas.

Mesmo assim, não recomendável apenas transcrever os nomes de atributos como nomes dos campos. Lembre que agora estamos falando de tabela física, de campos que serão acessados via programação ou seja, temos agora uma nova visão da situação. É nessa fase que, por questões de boa prática e organização no processo de modelagem deseja-se que sejam definidos alguns "padrões" para nomes de campos, abreviaturas e sempre primar por nomes curtos para os campos, ou seja das colunas da tabela.

Para deixar mais claro a forma como voce deve fazer isso, vou mostrar um exemplo:

Tem-se a entidade Pessoa com os atributos: Código, Nome, Endereço e Data de Nascimento. Código é o atributo identificador. Um exemplo de "tradução" dos campos seria:

Tabela: Pessoa - Campos: CodPessoa, NomePess, EnderecoPess, DataNascPess

Utilizei o seguinte padrão:

- Para a chave primária, utilizei o prefixo "Cod" + o nome da tabela.
- Para os demais campos utilizei o mesmo nome + o sufixo "Pess" em todos eles, identificando-os como sendo da tabela Pessoa.

Mas há uma boa justificativa para esse sufixo "Pess" nos nomes:

Ipoteticamente em uma instrução SQL pode haver uma junção da tabela Pessoa com a tabela Departamento. Departamento também possui um campo Nome. É recomendável não utilizar o mesmo nome de campo em tabelas diferentes para não gerar a seguinte situação:

SELECT Pessoa.Nome, Departamento.Nome FROM Pessoa, Departamento (...);

Na seleção SQL de campos o mesmo nome, deve-se especificar o nome da tabela de origem, o que pode tornar a cláusula muito longa, visto que consultas muitos mais complexas que essa são feitas com muita frequência.



```
SELECT NomePess, NomeDept FROM Pessoa, Departamento (...);
```

A utilização do sufixo para os mesmos campos de tabelas diferentes, nos poupa o trabalho de incluir o nome da tabela de origem, tornando a cláusula SQL mais limpa e legível, além de tornar o campo reconhecido em qualquer consulta: (Todos os campos com final "Pess" serão reconhecidos como sendo da tabela Pessoa)

Esse exemplo mostra que vários fatores estão envolvidos na transformação para o modelo relacional. No próximo artigo seguiremos com a próxima etapa da tradução: Relacionamentos e Atributos..

## Parte 05 (Transformação entre Modelos)

Olá caros leitores! Vamos em frente com a série de artigos sobre Modelagem de Dados. Essa parte da transformação entre modelos é a parte mais interessante da modelagem. Aqui começa "de verdade" a surgir, finalmente, o banco de dados propriamente dito. Em especial a tradução dos relacionamentos (assunto que começo a abordar hoje) é o que deixa mais claro os conceitos vistos na modelagem conceitual, ou modelagem lógica.

Um relacionamento nada mais é do que uma "ligação" entre duas entidades (agora, Tabelas) que juntas criam informações complexas. Como o assunto agora é tabela física do banco de dados, nos temos que construir nossa base de forma "conectada". Para isso nós usamos as famosas CHAVES. As chaves tem variações e diferentes finalidades. De forma rápida vou explicar um pouco sobre cada tipo de chave:

**Chave Primária:** A chave primária é o que chamávamos na modelagem lógica de "atributo identificador". Geralmente é um campo da tabela que armazena uma informação única, que não se repete em nenhum outro registro daquela mesma tabela. Desta forma ele serve como identificador daquele registro.

**Chave Composta:** A chave composta é formada pela chave primária e por alguma outra informação que também é única na tabela. Os dois campos juntos, formam uma chave composta. Este tipo de chave é usado com mais frequência em tabelas provenientes de relacionamentos N:N [vários para vários] (veremos mais à frente).

**Chave Estrangeira:** Esta chave é um campo em uma tabela que armazena o conteúdo da chave primária de outra tabela. Chave estrangeira é sinônimo de relacionamento entre tabelas. Se há relacionamento há chave estrangeira.

A primeira coisa que define a forma como você vai traduzir o relacionamento é a cardinalidade das entidades (assunto abordado no começo da série).

**Cardinalidade 1:N (Um-Para-Vários):** Indica a adição de um campo na tabela correspondente ao "lado N". Esse campo será uma chave estrangeira e vai armazenar a chave primária da tabela do "lado 1". Veja um esquema relacional que exemplifica esse tipo de cardinalidade:

Funcionario (CodFuncionario, NomeFunc)

Ramal (CodRamal, CodFuncionario, NumeroRamal)

CodFuncionario referencia Funcionario

**Observe:** Segundo a notação de Esquema Relacional, campo sublinhado indica Chave Primária. No esquema relacional da tabela Ramal temos dois campos sublinhados (indicando que os dois forma a chave primária). O campo CodFuncionario no entanto é uma chave estrangeira, porque ele armazena um valor que identifica um registro na tabela Funcionario.

**Cardinalidade N:N (Vários-Para-Vários):** Este tipo de relacionamento gera uma terceira tabela. Neste caso nenhum dos lados do relacionamento vai armazenar chave estrangeira (como no 1:N). O que vai acontecer é que cria-se uma nova tabela, que vai armazenar as chaves de ambos os lados. Veja um esquema relacional que exemplifica esse tipo de cardinalidade:

Funcionario(CodFuncionario, NomeFunc)

Projeto(CodProjeto, TituloProjeto)

Participacao(CodFuncionario, CodProjeto)

CodFuncionario referencia Funcionario

CodProjeto referencia Projeto

**Observe:** Segundo este esquema relacional, um funcionário pode participar de vários projetos, sendo que em um projeto pode-se ter vários funcionários. Neste caso cria-se mais uma tabela, que vai juntar as duas chaves do relacionamento. No exemplo, cria-se a tabela "Participacao" que indica o funcionário e qual projeto ele participa.

**Cardinalidade 1:1 (Um-Para-Um):** Esse tipo de cardinalidade em grande maioria dos casos não justifica um relacionamento. Mas pode ser implementado por questões de organização dos dados. Veremos futuramente na parte de Normalização, como resolver relacionamentos 1:1..

No próximo artigo, continuamos a fazer a tradução de relacionamentos . Até breve.

## Parte 06 (Generalizações / Especializações)

Caros colegas. Estive ausente da coluna nesses 3 meses por motivos profissionais, mas aqui estou, e darei continuidade à série sobre Modelagem.

No artigo anterior vimos como traduzir os relacionamentos. Para concluir essa etapa, vamos às recomendações:

- Tradução de relacionamentos é orientada sempre pela cardinalidade mínima.
- Se houverem relacionamentos 1:1, verifique se não é melhor fundir as 2 tabelas em uma única.
- A chave primária é colocada sempre na tabela que corresponde ao lado N do relacionamento 1:N.
- Relacionamentos N:N sempre geram uma terceira tabela, com as chaves primárias das 2 tabelas originais.
- Se necessário, escreva o esquema relacional do seu banco de dados, antes de gerar suas tabelas.

A tradução de generalizações/especializações tem alguns aspectos particulares que devemos analisar.

Primeiramente vamos supor uma entidade com especializações:

**Entidade: Pessoa** - Atributos: Código, Nome, Endereço e Telefone

**Entidade: Pessoa Física** - Atributos: Todos os atributos de Pessoa, CPF, RG

**Entidade: Pessoa Jurídica** - Atributos: Todos os atributos de Pessoa, CNPJ, IE, Razão Social

As entidades Pessoa Física e Jurídica são especializações da entidade Pessoa. Como representar isso no banco de dados?

Bem, nesse caso temos 2 alternativas:

1) Criar uma única tabela para todas as especializações e incluir um campo diferenciador: Seria

juntar todos os tipos de Pessoa, em uma única tabela e acrescentar mais um campo para identificar a Pessoa. Exemplo:

**Pessoa: Código, TipoDePessoa, Nome, Endereço, Telefone, CPF, RG, CNPJ, IE, RazaoSocial**

2) Criar uma tabela para cada especialização e definir mais um campo identificador

**Pessoa: Código, Nome, Endereço, Telefone**

**Pessoa\_Fisica: CodPessoa, CPF, RG**

**Pessoa\_Juridica: CodPessoa, CNPJ, IE, RazaoSocial**

A vantagem da primeira alternativa é que não precisaremos fazer junções da tabela generalizada (Pessoa) com a tabela especializada (Pessoa Física ou Jurídica) quando precisarmos de informações específicas. Outra vantagem é que a chave primária da tabela Pessoa fica armazenada somente 1 vez no banco de dados. A desvantagem é que, ao fazermos uma consulta no banco de dados, a linha inteira (todos os campos) são carregados na memória, mas sabemos que haverá campos em branco, dependendo do tipo de Pessoa cadastrada.

Na segunda alternativa, há a necessidade de fazer junções quando formos obter todas as informações de uma Pessoa. Porém, a vantagem é que teremos somente os dados necessários sem a necessidade de carregar todos os campos na memória, gerando mais acessos ao banco de dados. As chaves primárias de Pessoa são repetidas nas tabelas especializadas e, quando houver atualização das informações de uma pessoa, haverá a necessidade de criar uma instrução para cada tabela especializada.

A escolha de um dos tipos de tradução para generalizações/especializações irá depender do projeto que está sendo construído e dos recursos disponíveis para quem está modelando. Nada impede que as duas alternativas sejam usadas no mesmo projeto de banco de dados, uma alternativa pra cada caso de tabelas generalizadas.

Concluídas as etapas de tradução entre modelos, temos o banco de dados formado, com suas tabelas, campos e relacionamentos.

No próximo artigo falarei sobre como refinar o seu modelo relacional e também sobre a Normalização do seu banco de dados.

### **Observações:**

*Nesta série de artigos, eu faço uma análise de alto nível sobre a Modelagem de Dados sem aprofundar-me nos processos de modelagem. Se for do seu interesse, vale a pena pesquisar o conteúdo específico sobre cada etapa que descrevi nessa série.*

*Se você me enviou e-mails e não obteve resposta, peço a gentileza de reenviar. Faço questão de responder a todas as mensagens, apesar do pouco tempo hábil.*

## **Final (Normalização)**

Caros leitores. Estou de volta com a última parte desta série, onde falarei sobre Normalização.

Normalização é um processo baseado nas chamadas formas normais. Uma forma normal é uma regra e deve ser aplicada na construção das tabelas do banco de dados para que estas fiquem bem projetadas. Segundo autores, existem 4 formas normais. Neste artigo vou falar sobre as 3 primeiras, sendo as principais.

Com o banco de dados construído, devem-se aplicar as 3 formas normais em cada tabela, ou grupo de tabelas relacionadas. As formas têm uma ordem e são dependentes, isto é, para se aplicar a segunda norma, deve-se obrigatoriamente ter aplicado a primeira e assim por diante.

Então, vamos às normas:

### **1 Forma Normal: Verificação de Tabelas Aninhadas.**

Para uma tabela estar na primeira forma normal ela não deve conter tabelas aninhadas. Um jeito fácil de verificar esta norma é fazer uma leitura dos campos das tabelas fazendo a pergunta: Este campo depende de qual?.

Vamos exemplificar, com a tabela Venda. Este é o esquema relacional da tabela:

**Venda**(Codvenda, Cliente, Endereco, Cep, Cidade, Estado, Telefone, Produto, Quantidade, Valorunitario, Valorfinal).

O raciocínio é o seguinte: A tabela Venda, deve armazenar informações da venda. Pois bem, verificando o campo Cliente, sabemos que ele depende de CodVenda, afinal para cada Venda há um cliente. Vendo o campo Endereço, podemos concluir que ele não depende de Codvenda, e sim de Cliente, pois é uma informação referente particularmente ao cliente. Não existe um endereço de venda, existe sim um endereço do cliente para qual se fez a venda. Nisso podemos ver uma tabela aninhada. Os campos entre colchetes, são referentes ao cliente e não á venda.

**Venda** (Codvenda, [Cliente, Endereço, Cep, Cidade, Estado, Telefone, Produto, Quantidade, Valorunitario, Valorfinal]).

A solução é extrair estes campos para uma nova tabela, adicionar uma chave-primária à nova tabela e relaciona-la com a tabela Venda criando uma chave-estrangeira.

Ficaria desta forma:

**Cliente** (Codcliente, Nome, Endereço, Cep, Cidade, Estado, Telefone).

**Venda** (Codvenda, Codcliente, Produto, Quantidade, Valorunitario, Valorfinal).

Agora aplicamos novamente á primeira forma normal as 2 tabelas geradas. Uma situação comum em tabelas de cadastro é o caso Cidade-Estado. Analisando friamente pela forma normal, o Estado na tabela Cliente, depende de Cidade. No entanto Cidade, também depende de Estado, pois no caso de a cidade ser Curitiba o estado sempre deverá ser Paraná, porém se o Estado for Paraná, a cidade também poderá ser Londrina. Isso é o que chamamos de Dependência funcional: é onde aparentemente, uma informação depende da outra. No caso Cidade-Estado a solução é simples:

Extraímos Cidade e Estado, de Cliente e geramos uma nova tabela. Em seguida, o mesmo processo feito anteriormente: adicionar uma chave-primária à nova tabela e relaciona-la criando uma chave-estrangeira na antiga tabela.

**Cidade** (Codcidade, Nome, Estado).

**Cliente** (Codcliente, Codcidade, Nome, Endereço, Cep, Telefone).

**Venda** (Codvenda, Codcliente, Codcidade, Produto, Quantidade, Valorunitario, Valorfinal).

Seguindo com o exemplo, a tabela Cliente encontra-se na 1 forma normal, pois não há mais tabelas aninhadas. Verificando Venda, podemos enxergar mais uma tabela aninhada. Os campos entre colchetes são referente á mesma coisa: Produto de Venda

**Venda** (Codvenda, Codcliente, Codcidade, [Produto Quantidade, [Valorunitario, Valorfinal]).

Na maioria das situações, produtos têm um valor previamente especificado. O Valorunitário depende de Produto. Já a Quantidade (campo entre Produto e Valorunitario) não depende do produto e sim da Venda.

**Cidade** (Codcidade, Nome, Estado).

**Cliente** (Codcliente, Codcidade, Nome, Endereço, Cep, Telefone).

**Produto** (Codproduto, Nome, Valorunitario).

**Venda** (Codvenda, Codcliente, Codcidade, Codproduto, Quantidade, Valorfinal).

Passando a tabela Venda pela primeira forma normal, obtivemos 3 tabelas. Vamos á próxima forma

## 2 Forma Normal: Verificação de Dependências Parciais

Para uma tabela estar na segunda forma normal, além de estar na primeira forma ela não deve conter dependências parciais. Um jeito de verificar esta norma é refazer a leitura dos campos fazendo a pergunta: Este campo depende de toda a chave? Se não, temos uma dependência parcial.

Vimos antes o caso Cidade-Estado que gerava uma dependência funcional. É preciso entender este conceito para que você entenda o que é Dependência Parcial.

Após a normalização da tabela Venda, acabamos com uma chave composta de 4 campos:

**Venda** (Codvenda, Codcliente, Codcidade, Codproduto, Quantidade, Valorfinal).

A questão agora é verificar se cada campo não-chave depende destas 4 chaves. O raciocínio seria assim:

1. O primeiro campo não-chave é Quantidade.
2. Quantidade depende de Codvenda, pois para cada venda há uma quantidade específica de itens.
3. Quantidade depende de Codvenda e Codcliente, pois para um cliente podem ser feitas várias vendas, com quantidades diferentes.
4. Quantidade não depende de Cidade. Quem depende de Cidade é Cliente. Aqui está uma dependência parcial.
5. Quantidade depende de Codproduto, pois para cada produto da Venda á uma quantidade certa.

Quantidade depende de 3 campos, dos 4 que compõe a chave de Venda. Quem sobrou nessa história foi Codcidade. A tabela Cidade já está ligada com Cliente, que já está ligado com Venda. A chave Codcidade em Venda é redundante, portanto podemos eliminá-la.

**Venda** (Codvenda, Codcliente, Codproduto, Quantidade, Valorfinal).

O próximo campo não-chave é Valorfinal. Verificando Valorfinal, da mesma forma que Quantidade, ele depende de toda a chave de Venda. Portanto vamos á próxima norma.

## 3 Forma Normal: Verificação de Dependências Transitivas

Para uma tabela estar na segunda forma normal, além de estar na segunda forma ela não deve conter dependências transitivas. Um jeito de verificar esta norma é refazer a leitura dos campos fazendo a pergunta: Este campo depende de outro que não seja a chave? Se Sim, temos uma dependência transitiva.

No exemplo de Venda, temos um caso de dependência transitiva:

Na tabela Venda, temos Valorfinal. Este campo é o resultado do valor unitário do produto multiplicado pela quantidade, isto é, para um valor final existir ele DEPENDE de valor unitário e quantidade. O Valorunitário está na tabela Produto, relacionada à Venda e Quantidade está na própria Venda. Valorfinal depende destes 2 campos e eles não são campos-chave, o que nos leva a pensar: Se temos valor unitário e quantidade, porque teremos valor final? O valor final nada mais é que o resultado de um cálculo de dados que já está estão no banco, o que o torna um campo redundante.

Quando for necessário ao sistema obter o valor final, basta selecionar o valor unitário e multiplicar pela quantidade. Não há porque guardar o valor final em outro campo. Aqui a solução é eliminar o

campo Valorfinal.

**Cidade** (Codcidade, Nome, Estado).

**Cliente** (Codcliente, Codcidade, Nome, Endereco, Cep, Telefone).

**Produto** (Codproduto, Nome, Valorunitario).

**Venda** (Codvenda, Codcliente, Codproduto, Quantidade).

Em tese, agora temos todas as tabelas normalizadas. Ainda restou o caso do campo Estado na tabela Cidade, mas eu deixarei para uma outra ocasião, pois o objetivo aqui é mostrar conceitualmente o processo de normalização do banco de dados.

É muito comum, no processo de normalização enxergarmos todas as formas normais ao mesmo tempo. Enquanto separamos as tabelas aninhadas, já conseguimos ver as dependências transitivas e logo mais encontramos uma dependência parcial, tudo assim, ao mesmo tempo. Isso é normal. Só tome cuidado, para não deixar nada passar batido.

Reforçando o recado do artigo anterior: tem muito mais a ser visto sobre as etapas que eu mostrei nessa série. Aqui foi só uma demonstração do que se deve levar em conta ao modelar e construir um banco de dados íntegro, correto e que aproveita os dados da forma mais eficiente possível.

#### **Links dos originais:**

[http://imasters.uol.com.br/artigo/4468/uml/modelagem\\_de\\_dados\\_no\\_access/](http://imasters.uol.com.br/artigo/4468/uml/modelagem_de_dados_no_access/)

[http://imasters.uol.com.br/artigo/4629/bancodedados/modelagem\\_de\\_dados\\_1\\_entidades/](http://imasters.uol.com.br/artigo/4629/bancodedados/modelagem_de_dados_1_entidades/)

[http://imasters.uol.com.br/artigo/4799/access/modelagem\\_de\\_dados\\_2\\_-\\_os\\_relacionamentos/](http://imasters.uol.com.br/artigo/4799/access/modelagem_de_dados_2_-_os_relacionamentos/)

[http://imasters.uol.com.br/artigo/5111/bancodedados/modelagem\\_de\\_dados\\_-\\_validacao\\_do\\_modelo\\_er/](http://imasters.uol.com.br/artigo/5111/bancodedados/modelagem_de_dados_-_validacao_do_modelo_er/)

[http://imasters.uol.com.br/artigo/5265/bancodedados/modelagem\\_de\\_dados\\_parte\\_04\\_-\\_abordagem\\_relacional/](http://imasters.uol.com.br/artigo/5265/bancodedados/modelagem_de_dados_parte_04_-_abordagem_relacional/)

[http://imasters.uol.com.br/artigo/5403/bancodedados/modelagem\\_de\\_dados\\_-\\_parte\\_05\\_transformacao\\_entre\\_modelos/](http://imasters.uol.com.br/artigo/5403/bancodedados/modelagem_de_dados_-_parte_05_transformacao_entre_modelos/)

[http://imasters.uol.com.br/artigo/6167/bancodedados/modelagem\\_de\\_dados\\_-\\_parte\\_06\\_generalizacoes\\_especializacoes/](http://imasters.uol.com.br/artigo/6167/bancodedados/modelagem_de_dados_-_parte_06_generalizacoes_especializacoes/)

[http://imasters.uol.com.br/artigo/7020/bancodedados/modelagem\\_de\\_dados\\_-\\_final\\_normalizacao/](http://imasters.uol.com.br/artigo/7020/bancodedados/modelagem_de_dados_-_final_normalizacao/)

[http://imasters.uol.com.br/artigo/7020/bancodedados/modelagem\\_de\\_dados\\_-\\_final\\_normalizacao/](http://imasters.uol.com.br/artigo/7020/bancodedados/modelagem_de_dados_-_final_normalizacao/)

Modelagem de dados e administração

Não diagramação e mapeamento

SQL não é relacional e contém muitos limites arbitrários

...a diferença entre um mau e um bom programador é se considera o código ou as estruturas de dados mais importantes. Maus programadores preocupam-se com código.

Bons programadores preocupam-se com estruturas de dados e seus relacionamentos.

Torvalds, Linux.

Mostra-me teus fluxogramas e esconda-me tuas tabelas, e continuarei no escuro.

Mostra-me tuas tabelas, e... não precisarei de teus fluxogramas: serão óbvios.

-- Brooks, Frederick Phillips, Jr.: The Mythical Man-Month.

Dicionários de dados são essenciais.

qDiagramas podem, e devem, ser gerados automaticamente.

qFerramentas de modelagem podem unificar modelos de diversas bases.

qGlossários são úteis.

qDicionários de tipos de dados são essenciais.

qDicionário geral de dados dão mais trabalho.

qDiagramas te promovem: AutoDoc!

## O Modelo Relacional de Dados - Parte 01 - Júlio Battisti

### Introdução

Objetivo: Em uma série de quatro artigos, apresentarei alguns conceitos básicos sobre Bancos de Dados, mais especificamente sobre o Modelo Relacional de Dados.

Para a melhor utilização, ou seja, para uma utilização eficiente de bancos de dados como o Microsoft Access, SQL Server, ORACLE, DB2 ou qualquer outro banco de dados relacional, é importante o conhecimento e correto entendimento dos conceitos apresentados nesta série de artigos. Vou abordar os seguintes Conceitos:

- . Entidades e atributos
- . Chave primária
- . Relacionamentos entre entidades (tabelas)
- . Integridade Referencial
- . Normalização de tabelas
- . Um Problema Proposto
- . Arquitetura do Microsoft Access.

**Nota:** Os exemplos apresentados utilizarão telas do Microsoft Access. Porém os princípios básicos do modelo relacional aplicam-se a qualquer banco de dados baseado no modelo relacional de dados. Estes bancos de dados são algumas vezes denominados: SGBDR - Sistemas Gerenciadores de Banco de Dados Relacionais.

Entidades e Atributos:

Toda a Informação de um banco de dados relacional é armazenada em Tabelas, que na linguagem do modelo relaciona, também são chamadas de Entidades. Por exemplo, posso ter uma Tabela "Clientes", onde seriam armazenadas informações sobre os diversos clientes.

Sobre cada um dos clientes podem ser armazenadas diversas informações tais como:

- .Nome
- .RG
- .CPF
- .Rua
- .Bairro
- .Telefone
- .CEP
- .Data de Nascimento

Essas diversas características de cada Cliente são os "**Atributos**" da entidade Cliente, também chamados de campos da tabela Cliente.

"O Conjunto de todos os Atributos de um cliente e os valores dos atributos é o que forma o Registro do Cliente".

Com isso temos uma Tabela que é constituída por um conjunto de Registros (uma linha completa com informações sobre o cliente) e cada Registro formado por um conjunto de atributos (Nome, Endereço, etc).

Resumindo:

**Entidade ou Tabela:** Um conjunto de Registros.

**Campos ou Atributos:** Características Individuais da tabela.

Considere o Exemplo da figura abaixo, mostro uma tabela com cadastro de Clientes com os seus diversos Campos (atributos):



	Código do Cliente	Nome da Empresa	Nome do Contato	Cargo do Contato
+	FAMIA	Familia Arquibaldo	Aria Cruz	Assistente de Marketing
+	COMMI	Comércio Mineiro	Pedro Afonso	Assessor de Vendas
+	RICAR	Ricardo Adocicados	Janete Limeira	Assistente do Agente de V
+	TRADH	Tradição Hipermercados	Anabela Domingues	Representante de Vendas
+	QUEDE	Que Delícia	Bernardo Batista	Gerente Financeiro
+	QUEEN	Queen Cozinha	Lúcia Carvalho	Assistente de Marketing
+	LAUGB	Laughing Bacchus Wine Cellars	Yoshi Tannamuri	Assistente de Marketing
+	MEREP	Mère Paillard	Jean Fresnière	Assistente de Marketing
+	BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	Gerente Financeiro
+	SIMOB	Simons bistro	Jytte Petersen	Proprietário
+	VAFPE	Vaffeljernet	Palle Ibsen	Gerente de Vendas
+	BOLID	Bólido Comidas preparadas	Martín Sommer	Proprietário

Registro: 32 de 91

Código exclusivo de cinco caracteres baseado no nome do cliente.

Figura 1: Tabela Cliente e seus Campos - CódigoDoCliente, NomeDaEmpresa e assim por diante

No exemplo da figura anterior temos entidade: "Clientes" e seus diversos atributos: "Código do Cliente", "Nome da Empresa", "Nome do Contato", "Cargo do Contato", "Endereço", etc. Em cada linha temos um conjunto de atributos e seus valores. Cada linha forma um Registro. Cada Coluna é um atributo da Tabela Clientes.

Um dos grandes desafios em se projetar um Banco de Dados com sucesso é a correta Determinação das Entidades que existirão no Banco de Dados, bem como dos Atributos de Cada Entidade.

### Chave Primária

Objetivo: Neste item falarei sobre o conceito de Chave Primária e a sua importância no Modelo Relacional de dados.

### Chave Primária

O Conceito de "**Chave Primária**" é fundamental para o correto entendimento de como funciona um Banco de Dados baseado no modelo relacional. Vamos entender o que significa um campo ser a Chave Primária de uma Tabela e como tornar um Campo a Chave Primária de uma Tabela.

*"Ao Definirmos um Campo como sendo uma Chave Primária, estamos informando ao Microsoft Access que não podem existir dois registros com o mesmo valor no campo que é a Chave Primária, ou seja, os valores no campo Chave Primária precisam ser únicos".*

Por exemplo, se defino um campo "Número da Identidade", da tabela Clientes, como sendo um campo do tipo Chave Primária, estou dizendo que não podem ser cadastrados dois clientes com o mesmo valor no campo "Número da Identidade". Na prática estou garantindo que não possam ser cadastrados dois clientes com o mesmo Número de Identidade".

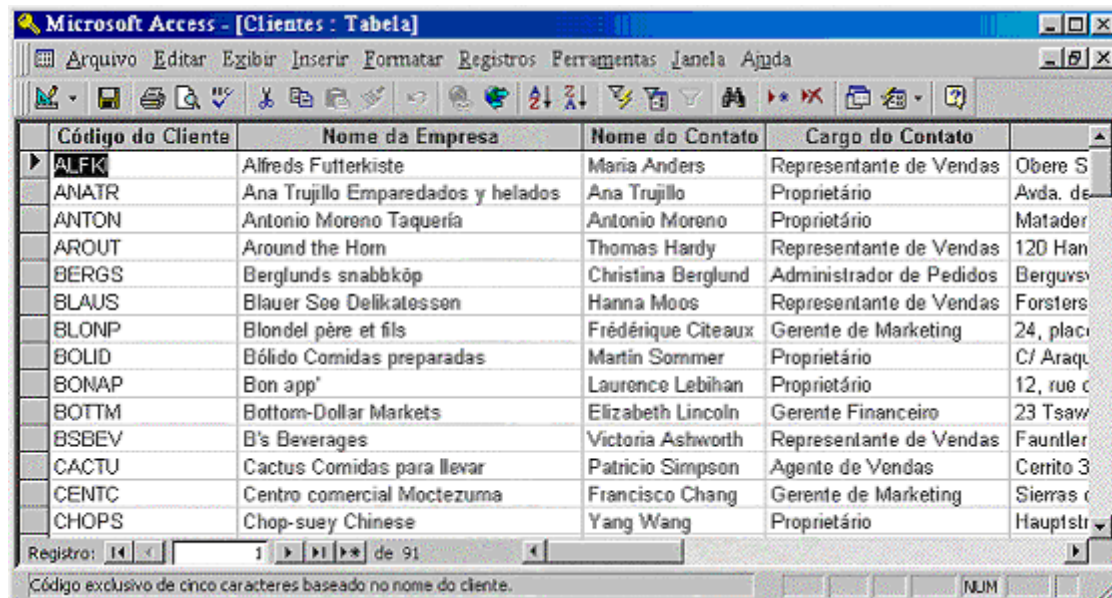
Em outras palavras poderíamos dizer que o Campo Chave Primária identifica de Maneira Única cada Registro de uma Tabela, isto é, de posse do valor da Chave Primária somente localizaremos um registro com aquele valor no campo Chave Primária. Outros exemplos de campos que podem ser definidos como chaves primária:

- Campo CPF em uma tabela de cadastro de clientes.
- Campo CNPJ em uma tabela de cadastro de fornecedores.
- Matrícula do aluno em uma tabela de cadastro de alunos.

- . Código da Peça em uma tabela de cadastro de peças.
- . Matrícula do funcionário em uma tabela de cadastro de funcionários.
- . Número do pedido em uma tabela de cadastro de pedidos

Este é um conceito muito importante, pois conforme veremos mais adiante os conceitos de Integridade Referencial e Normalização estão diretamente ligados ao conceito de Chave Primária.

Na próxima figura apresento um exemplo da tabela Cliente onde o Campo "Código do Cliente" é definido como uma Chave Primária. Observe que não existem dois clientes com o Mesmo Código.



Código do Cliente	Nome da Empresa	Nome do Contato	Cargo do Contato	
ALFK	Alfreds Futterkiste	Maria Anders	Representante de Vendas	Obere S
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Proprietário	Avda. de
ANTON	Antonio Moreno Taquería	Antonio Moreno	Proprietário	Matader
AROUT	Around the Horn	Thomas Hardy	Representante de Vendas	120 Han
BERGS	Berglunds snabbköp	Christina Berglund	Administrador de Pedidos	Berguvs
BLAUS	Blauer See Delikatessen	Hanna Moos	Representante de Vendas	Forsters
BLONP	Blondel père et fils	Frédérique Citeaux	Gerente de Marketing	24, plac
BOLID	Bólido Comidas preparadas	Martin Sommer	Proprietário	C/ Araqu
BONAP	Bon app'	Laurence Lebihan	Proprietário	12, rue c
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	Gerente Financeiro	23 Tsaw
BSBEV	B's Beverages	Victoria Ashworth	Representante de Vendas	Fauntler
CACTU	Cactus Comidas para llevar	Patricio Simpson	Agente de Vendas	Cerrito 3
CENTC	Centro comercial Moctezuma	Francisco Chang	Gerente de Marketing	Sierras c
CHOPS	Chop-suey Chinese	Yang Wang	Proprietário	Hauptstr

Figura 2: Campo "Código do Cliente" definido como Chave Primária.

Após ter definido um campo como sendo a Chave Primária da tabela, o próprio banco de dados (quer seja Access, SQL Server, ORACLE ou qualquer outro), garante que não sejam inseridos dados duplicados no campo que é a chave primária.

Por exemplo, se o usuário tentar cadastrar um pedido com o mesmo número de um pedido já existente, o registro não será cadastrado e uma mensagem de erro será emitida.

Um último detalhe importante para lembrarmos é que a Chave Primária pode ser formada pela combinação de Mais de Um Campo. Podem existir casos em que um único campo não é capaz de atuar como chave primária, pelo fato deste apresentar valores repetidos. Nestes casos podemos definir uma combinação de 2 ou mais campos para ser a nossa chave primária.

Além disso, uma tabela somente pode ter uma Chave Primária, seja ela simples ou composta. Ou seja, não pode definir dois ou mais campos de uma tabela para serem cada um, uma chave primária separada. Não confundir com o caso de uma chave primária composta, onde a união de dois ou mais campos é que forma a única chave primária da tabela. Ou seja, cada tabela pode ter uma única chave primária.

## Conclusão

Nesta primeiro artigo da série, você aprendeu sobre os conceitos de entidades (tabelas), atributos (campos) e registros. Estes são os elementos básicos do modelo relacional de dados. Em seguida falei sobre o conceito de Chave Primária. Falei sobre as características do campo chave primária e da possibilidade de existir uma chave primária composta por dois ou mais campos. No próximo artigo da série você aprenderá sobre "Relacionamentos entre Tabelas".



# O Modelo Relacional de Dados – Parte 02

## Introdução

Na primeira parte deste artigo falei sobre Entidades (tabelas), Atributos (campos) e sobre o conceito de Chave Primária. Nesta segunda parte vou abordar os seguintes tópicos:

- . Relacionamentos entre entidades (tabelas) - conceito
- . Relacionamentos entre entidades (tabelas) - tipos

**Nota:** Os exemplos apresentados utilizarão telas do Microsoft Access e o arquivo de exemplos Northwind.mdb, o qual é instalado juntamente com o Microsoft Access. Este arquivo está disponível, por padrão, no seguinte caminho:

<C:\Arquivos de programas\Microsoft Office\Office\Samples>

Porém os princípios básicos do modelo relacional aplicam-se a qualquer banco de dados baseado no modelo relacional de dados. Estes bancos de dados são algumas vezes denominados: SGBDR - Sistemas Gerenciadores de Banco de Dados Relacionais.

## Relacionamentos entre Tabelas

Neste item vou apresentar o conceito de relacionamento entre tabelas, este um conceito fundamental para o Modelo Relacional de Dados. Também falarei sobre os diferentes tipos de relacionamentos existentes. Serão apresentados exemplos práticos.

Conforme descrito na Parte I, um banco de dados é composto por diversas tabelas, como por exemplo: Clientes, Produtos, Pedidos, Detalhes do Pedido, etc. Embora as informações estejam separadas em cada uma das Tabelas, na prática devem existir **relacionamentos** entre as tabelas. Por exemplo: Um Pedido é feito por um Cliente e neste Pedido podem existir diversos itens, itens que são gravados na tabela Detalhes do Pedido. Além disso cada Pedido possui um número único (Código do pedido), mas um mesmo Cliente pode fazer diversos pedidos e assim por diante.

Em um banco de dados, precisamos de alguma maneira para representar estes relacionamentos da vida Real, em termos das tabelas e de seus atributos. Isto é possível com a utilização de "Relacionamentos entre tabelas", os quais podem ser de três tipos:

- . Um para Um
- . Um para Vários
- . Vários para Vários

### Relacionamento do Tipo Um para Um:

Esta relação existe quando os campos que se relacionam são ambos do tipo Chave Primária, em suas respectivas tabelas. Cada um dos campos não apresenta valores repetidos. Na prática existem poucas situações onde utilizaremos um relacionamento deste tipo. Um exemplo poderia ser o seguinte: Imagine uma escola com um Cadastro de Alunos na tabela Alunos, destes apenas uma pequena parte participa da Banda da Escola. Por questões de projeto do Banco de Dados, podemos criar uma Segunda Tabela "Alunos da Banda", a qual se relaciona com a tabela Alunos através de um relacionamento do tipo Um para Um. Cada aluno somente é cadastrada uma vez na Tabela Alunos e uma única vez na tabela Alunos da Banda. Poderíamos utilizar o Campo Matrícula do Aluno como o Campo que relaciona as duas Tabelas.

**Importante:** O campo que relaciona duas tabelas deve fazer parte, ter sido definido, na estrutura das duas tabelas.

Na tabela Alunos da Banda poderíamos colocar apenas o Número da Matrícula do aluno, além das informações a respeito do Instrumento que ele toca, tempo de banda, etc. Quando fosse necessário buscar as informações tais como nome, endereço, etc, estas podem ser recuperadas através do

relacionamento existente entre as duas tabelas, evitando, com isso, que a mesma informação (Nome, Endereço, etc) tenha que ser duplicada nas duas tabelas, inclusive aumentando a probabilidade de erros de digitação.

Na Figura a seguir vemos o exemplo de um Relacionamento do tipo Um para Um entre as tabelas Alunos e Alunos da Banda.

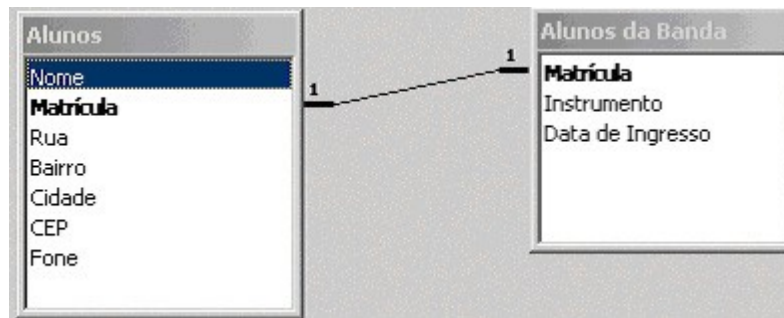


Figura 1: Relacionamento Um para Um entre as Tabelas Alunos e Alunos da Banda

Com a criação deste relacionamento estamos evitando a repetição desnecessária de informações em diferentes tabelas.

### Relacionamento do Tipo Um para Vários:

Este é, com certeza, o tipo de relacionamento mais comum entre duas tabelas. Uma das tabelas (o lado **um** do relacionamento) possui um campo que é a **Chave Primária** e a outra tabela (o lado **vários**) se relaciona através de um campo cujos valores relacionados **podem se repetir várias vezes**.

Considere o exemplo entre a tabela Clientes e Pedidos. Cada Cliente somente é cadastrado uma única vez na tabela de Clientes (por isso o campo Código do Cliente, na tabela Clientes, é uma chave primária, indicando que não podem ser cadastrados dois clientes com o mesmo código), portanto a tabela Clientes será o lado um do relacionamento. Ao mesmo tempo cada cliente pode fazer diversos pedidos, por isso que o mesmo Código de Cliente poderá aparecer várias vezes na tabela Pedidos: **tantas vezes quantos forem os pedidos que o Cliente tiver feito**. Por isso que temos um relacionamento do tipo Um para Vários entre a tabela Clientes e Pedidos, através do campo Código do Cliente, indicando que **um mesmo Cliente pode realizar diversos (vários) pedidos**.

Na próxima figura vemos um exemplo de um Relacionamento **Um para Vários** entre as Tabelas Clientes e Pedidos do banco de dados Pedidos.mdb, através do campo código do cliente:

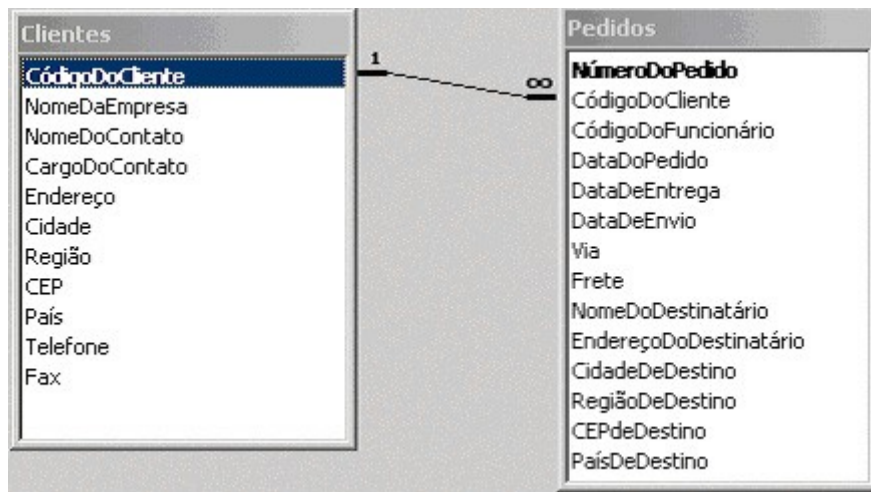


Figura 2: Relacionamento Um para Vários entre as Tabelas Clientes e Pedidos

Observe que o lado Vários do relacionamento é representado pelo símbolo do infinito (∞). Esta é a representação utilizada no Microsoft Access. Diferentes representações poderão ser utilizadas por outros bancos de dados.

No lado Um do relacionamento o campo é definido como uma Chave Primária (Campo CódigoDoCliente na tabela Clientes) e no lado Vários não (campo CódigoDoCliente na tabela Pedidos), indicando que no lado vários o Código do Cliente pode se repetir várias vezes, o que faz sentido, uma vez que um mesmo cliente pode fazer diversos pedidos.

**Importante:** Observe que o campo que é o lado vários do relacionamento não pode ser definido como chave primária. Lembrando do conceito de Chave Primária, apresentado na Parte I: Chave Primária é o campo no qual não podem haver valores repetidos. Ora, se o campo está no lado "vários", significa que ele poderá ter o seu valor repetido em vários registros. Por exemplo, na tabela pedidos, poderá haver vários registros para o mesmo cliente. Se o campo terá que ter valores repetidos, então ele não pode ser definido como chave primária.

No Banco de Dados NorthWind.mdb, Northwind.mdb, o qual é instalado juntamente com o Microsoft Access. Este arquivo está disponível, por padrão, no seguinte caminho:

C:\Arquivos de programas\Microsoft Office\Office\Samples

Existem diversos outros exemplos de relacionamentos do tipo Um para Vários, conforme descrito na Próxima Tabela:

Tipo	Lado Um	Lado Vários
Um para Vários	CódigoDoFornecedor na tabela <b>Fornecedores</b>	CódigoDoFornecedor na tabela <b>Produtos</b>
Um para Vários	CódigoDaCategoria na tabela <b>Categorias</b>	CódigoDaCategoria na tabela <b>Produtos</b>
Um para Vários	CódigoDoProduto na tabela <b>Produtos</b>	CódigoDoProduto na tabela <b>Detalhes do</b>



Pedido		
Um para Vários	CódigoDoFuncionário na tabela <b>Funcionários</b>	CódigoDoFuncionário na tabela <b>Pedidos</b>
Um para Vários	NúmeroDoPedido na tabela <b>Pedidos</b>	NúmeroDoPedido na tabela <b>Detalhes do Pedido</b>
Um para Vários	CódigoDaTransportadora na tabela <b>Transportadoras</b>	Via na tabela <b>Pedidos</b>
Um para Vários	CódigoDoCliente na tabela <b>Clientes</b>	CódigoDoCliente na tabela <b>Pedidos</b>

Em um dos próximos artigos mostrarei como implementar, na prática, estes relacionamentos. Algumas observações importantes sobre relacionamentos:

- O Nome dos Campos envolvidos no Relacionamento, não precisa ser, necessariamente, o mesmo, conforme indicado pelo relacionamento entre os campos CódigoDaTransportadora e Via, na tabela anterior. O tipo dos campos é que precisa ser o mesmo, por exemplo, se um dos campos for do tipo Texto, o outro também deverá ser do tipo Texto.
- Sempre o Lado um do Relacionamento deve ser uma chave primária, já o lado vários não pode ser uma chave Primária.
- De Preferência, antes de Criar os Relacionamentos verifique se o tipo dos campos a serem relacionados é o mesmo, além de características como máscaras de entrada e formato.

### Relacionamento do tipo Vários para Vários:

Este tipo de relacionamento "**aconteceria**" em uma situação onde em ambos os lados do relacionamento os valores poderiam se repetir. Vamos considerar o caso entre Produtos e Pedidos. Posso ter **Vários Pedidos** nos quais aparece um determinado produto, além disso **vários Produtos** podem aparecer no mesmo Pedido. Esta é uma situação em que temos um Relacionamento do Tipo Vários para Vários.

Na prática não é possível implementar um relacionamento deste tipo, devido a uma série de problemas que seriam introduzidos no modelo do banco de dados. Por exemplo, na tabela Pedidos teríamos que repetir o Número do Pedido, Nome do Cliente, Nome do Funcionário, Data do Pedido, etc para cada item do Pedido.

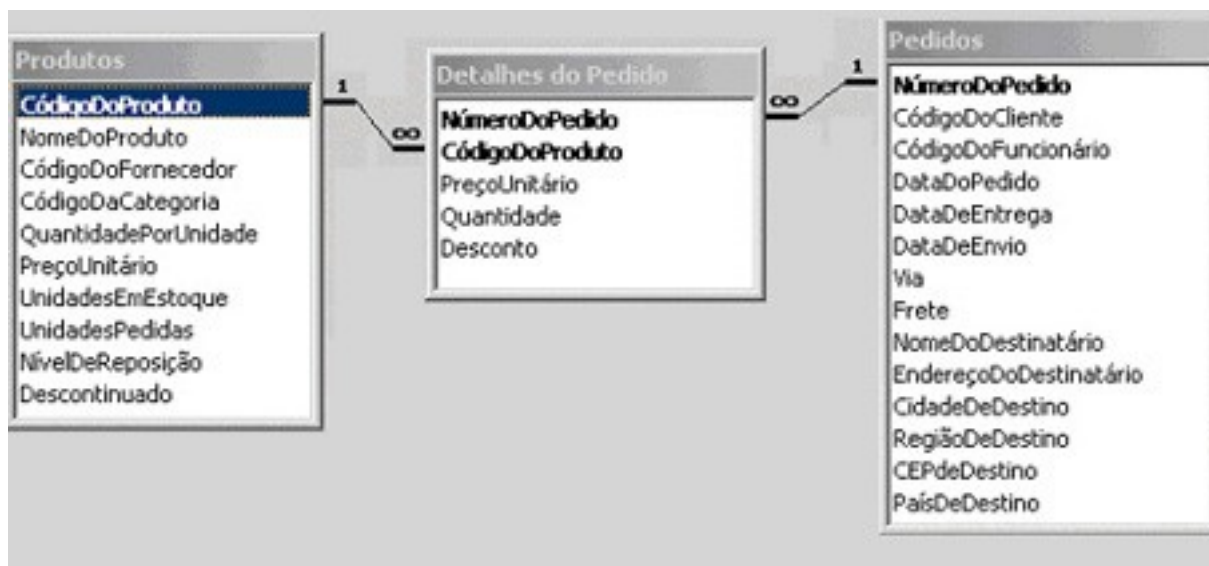
Para evitar este tipo de problema é bastante comum "**quebrarmos**" um relacionamento do tipo Vários para Vários em dois relacionamentos do tipo Um para Vários. Isso é feito através da criação de uma nova tabela, a qual fica com o lado Vários dos relacionamentos. No nosso exemplo vamos criar a tabela Detalhes do Pedido, onde ficam armazenadas as informações sobre os diversos itens de cada pedido, aí ao invés de termos um relacionamento do tipo Vários para Vários, teremos dois relacionamentos do tipo um para vários, conforme descrito pela próxima tabela:

Tipo	Lado Um	Lado Vários
------	---------	-------------

Um para Vários	CódigoDoProduto na tabela <b>Produtos</b>	CódigoDoProduto na tabela <b>Detalhes do Pedido</b>
Um para Vários	NúmeroDoPedido na tabela <b>Pedidos</b>	NúmeroDoPedido na tabela <b>Detalhes do Pedido</b>

Na figura abaixo temos a representação dos dois relacionamentos Um para Vários, resultantes da quebra do relacionamento vários-para-vários:





### Tabela Detalhes do Pedido ficou com o lado Vários dos Relacionamentos

Esta situação em que um relacionamento um para Vários é "quebrado" em dois Relacionamentos do tipo Um para Vários é bastante comum. Diversas vezes utilizamos esta técnica para eliminar uma série de problemas no Banco de Dados, tais como informação repetida e inconsistência de Dados.

Agora que já conhecemos os Tipos de Relacionamentos existentes, na próxima Parte III, falarei sobre o conceito de Integridade Referencial como uma maneira de Garantir a Consistência dos Dados.

### Conclusão:

Nesta segundo artigo da série, você aprendeu sobre os conceitos de relacionamentos, sem dúvidas um dos conceitos mais importantes do Modelo Relacional. Implementar um banco de dados sem se preocupar com um projeto cuidados dos relacionamentos, é garantia certa de "encrenca" mais adiantes. São consultas que retornam dados inesperados, são relatórios que retornam valores "absurdos" e por aí vai.

É fundamental que os profissionais de desenvolvimento e de administração de banco de dados entendem o quão impotante é planejar o banco de dados, cuidadosamente, definindo quais tabelas farão parte do banco de dados, quais os campos de cada tabela, quais campos serão chave primária e qual o relacionamento entre as tabelas. Muitos acham que esta é uma "perda de tempo", que o bom mesmo é sentar e começar a implementar o banco de dados, depois "a gente vê o que dá". Ledo engano. Não é perda de tempo, muito pelo contrário, é um ganho de tempo e principalmente de qualidade no produto final. Quanto tempo é perdido depois, tentando corrigir erros e inconsistências que muitas vezes são decorrentes de um banco de dados mal projetado? Uma boa leitura a todos e até a Parte III.

## O Modelo Relacional de Dados - Parte 03

**Objetivo:** Na primeira parte deste artigo falei sobre Entidades (tabelas), Atributos (campos) e sobre o conceito de Chave Primária. Na segunda parte falei sobre Relacionamentos e tipos de relacionamentos. Nesta terceira parte vamos aprender sobre um dos conceitos mais importantes do modelo relacional de dados: Integridade Referencial. Também mostrarei um exemplo prático de como configurar Relacionamentos e Integridade Referencial no Microsoft Access.

Nota: Os exemplos apresentados utilizarão telas do Microsoft Access e o arquivo de exemplos Northwind.mdb, o qual é instalado juntamente com o Microsoft Access. Este arquivo está disponível, por padrão, no seguinte caminho:

C:\Arquivos de programas\Microsoft Office\Office\Samples

Porém os princípios básicos do modelo relacional aplicam-se a qualquer banco de dados baseado no modelo relacional de dados. Estes bancos de dados são algumas vezes denominados: SGBDR - Sistemas Gerenciadores de Banco de Dados Relacionais.

#### Integridade Referencial

A Integridade Referencial é utilizada para garantir a Integridade dos dados entre as tabelas relacionadas. Por exemplo, considere um relacionamento do tipo Um-para- Vários entre a tabela Clientes e a tabela Pedidos (um cliente pode fazer vários pedidos). Com a Integridade Referencial, o banco de dados não permite que seja cadastrado um pedido para um cliente que ainda não foi cadastrado. Em outras palavras, ao cadastrar um pedido, o banco de dados verifica se o código do cliente que foi digitado já existe na tabela Clientes. Se não existir, o cadastro do pedido não será aceito. Com o uso da Integridade Referencial é possível ter as seguintes garantias (ainda usando o exemplo entre as tabelas Clientes e Pedidos):

- Quando o Código de um cliente for alterado na Tabela Clientes, podemos configurar para o banco de dados atualizar, automaticamente, todos os Códigos do Cliente na Tabela Pedidos, de tal maneira que não fiquem Registros Órfãos, isto é , registros de Pedidos com um Código de Cliente para o qual não existe mais um correspondente na Tabela Clientes. Essa ação é conhecida como **"Propagar atualização dos campos relacionados"**.

- Quando um Cliente for excluído da Tabela Clientes, podemos configurar para que o banco de dados exclua, automaticamente, na tabela Pedidos, todos os Pedidos para o Cliente que está sendo Excluído. Essa opção é conhecida como **"Propagar exclusão dos registros relacionados"**.

Essas opções, conforme mostrarei logo em seguida, podem ser configuradas quando da Definição dos Relacionamentos (no exemplo prático mais adiante utilizarei o Microsoft Access, mas estes conceitos são válidos para qualquer banco de dados). Estas opções não são obrigatórias, isto é, podemos optar por não Atualizar ou não Excluir em cascata. A Opção de **"Propagar atualização dos campos relacionados"** é utilizada na maioria das situações, já a opção de **"Propagar exclusão dos registros relacionados"** deve ser estudada caso a caso. Por exemplo, se nos quiséssemos manter um histórico com os Pedidos de cada Cliente, não utilizaríamos a opção "Propagar exclusão dos registros relacionados"; caso não nos interessasse manter um histórico dos pedidos, poderíamos utilizar esta opção.

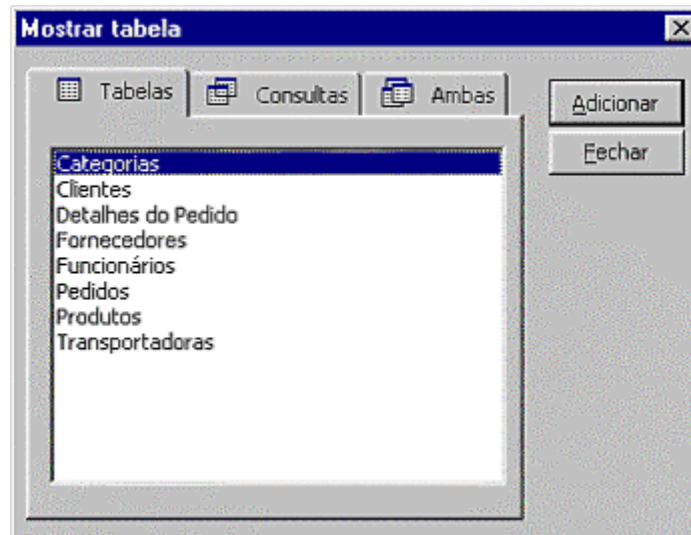
#### Exemplo prático: Como Criar e Configurar Relacionamentos no Microsoft Access:

Para Definir Relacionamentos no Microsoft Access siga os passos indicados a seguir:

**01.** Abra o banco de dados onde estão as tabelas nas quais serão definidos os relacionamentos.

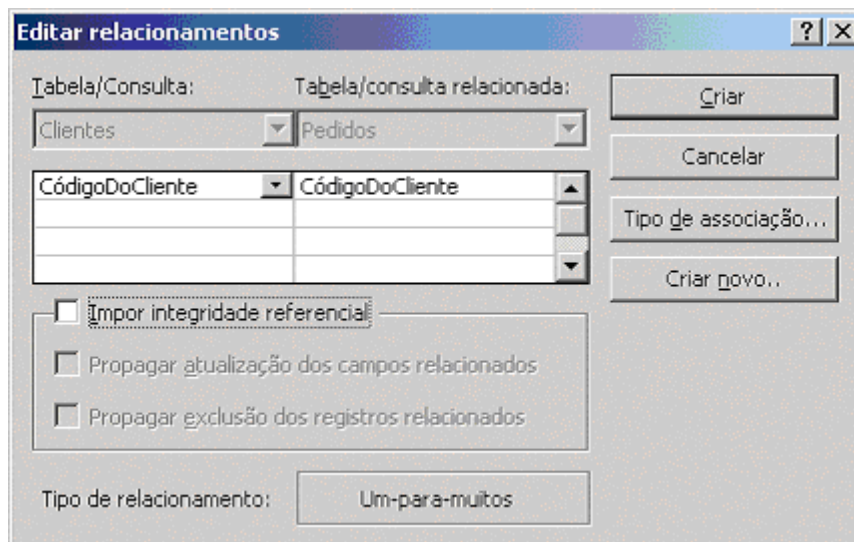
**02.** Selecione o comando Ferramentas -> Relacionamentos.

**03.** Surgirá a Janela indicada na próxima Figura. Nesta Janela você adicionará as Tabelas que farão parte de algum dos relacionamentos. Para Adicionar uma Tabela, basta marcá-la e dar um clique no botão "Adicionar". Você pode adicionar todas as tabelas de uma única vez. Para isto dê um clique na primeira, libere o mouse, pressione a tecla SHIFT e fique segurando SHIFT pressionado e dê um clique na última tabela. Com isso todas serão selecionadas, agora ao dar um clique no botão Adicionar, todas as tabelas selecionadas serão adicionadas. Caso não queira adicionar todas mas somente algumas e de uma maneira intercalada, ao invés de usar a tecla SHIFT, utilize a tecla CTRL. Com a tecla CTRL uma tabela é selecionada a medida que você vai clicando com o mouse sobre o nome da tabela.



Dê um clique para marcar a Tabela e depois dê um clique no botão Adicionar

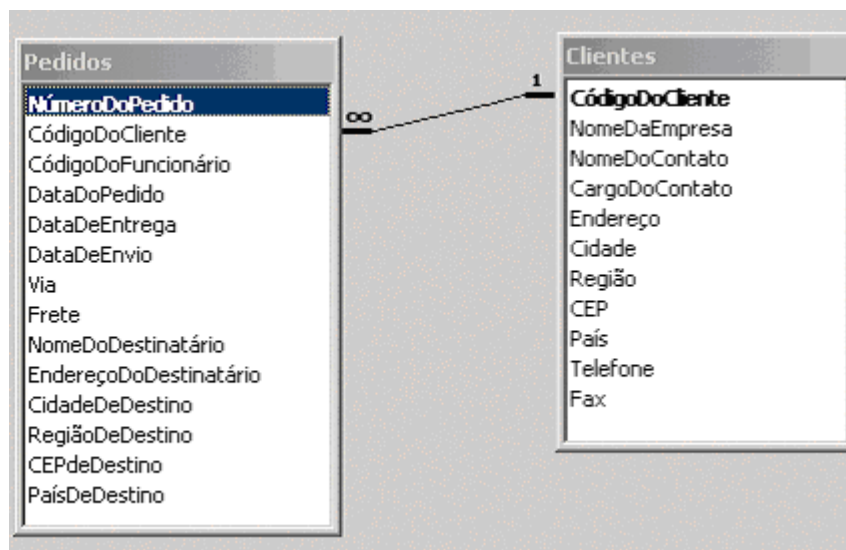
**04.** Após ter adicionado as tabelas, para criar um relacionamento, basta arrastar um campo de uma tabela sobre o campo da outra tabela na qual será estabelecido o relacionamento. Por exemplo, para estabelecer o relacionamento Um-para-Vários entre as tabelas **Clientes** e **Pedidos**, arraste o campo "CódigoDoCliente" da tabela Clientes, sobre o campo "CódigoDoCliente" da Tabela Pedidos. Ao largar um campo sobre o Outro, o Microsoft Access abre uma janela conforme indicado na figura a seguir (Definindo as características do Relacionamento):



### Definindo as Características do Relacionamento

**05.** Observe que, por padrão, o campo "**Importar Integridade Referencial**" não está marcado. Ao marcá-lo serão habilitadas as Opções de "Propagar atualização dos campos relacionados" e "Propagar exclusão dos registros relacionados". Observe, também, que o Microsoft Access já definiu este relacionamento como sendo do tipo Um-para-Vários. Isso acontece porque o Microsoft Access identifica o campo CódigoDoCliente na tabela Clientes como sendo do tipo chave primária e na tabela Pedidos como não sendo chave primária, o que automaticamente transforma o Relacionamento como sendo do tipo Um para Vários. Se em ambas as tabelas o campo CódigoDoCliente fosse definido como Chave Primária, o relacionamento, automaticamente, seria do tipo Um-para-Um.

**06.** Após marcar as Opções desejadas, basta dar um clique no botão "Criar" e pronto, o Microsoft Access cria o Relacionamento, o qual é indicado através de uma linha entre as duas tabelas (Clientes e Pedidos), com o número 1 no lado da Chave Primária e o Sinal de infinito no lado Vários. Caso você precise alterar as características de um determinado relacionamento, basta dar um duplo clique sobre a linha do relacionamento, que o Microsoft Access abrirá a janela indicada na figura anterior, para que você possa fazer as alterações desejadas. Na Figura a seguir indico o relacionamento já criado entre as tabelas Pedidos e Clientes:



### Relacionamento entre Pedidos e Clientes

**07.** Observe também que os campos Chave Primária aparecem em Negrito no Diagrama dos Relacionamentos.

**08.** Este diagrama que exibe as Tabelas e os Relacionamentos entre as tabelas é conhecido como "**Diagrama Entidades x Relacionamentos (DER)**".

**09.** Antes de fechar o Diagrama Entidades x Relacionamentos, dê um clique no botão com o desenho do disquete para salvar as alterações que foram feitas. A qualquer momento você pode acessar o Diagrama Entidades x Relacionamentos para fazer alterações ou para revisar os relacionamentos, para isto basta ir no menu Ferramentas e clicar em Relacionamentos.

### Conclusão

Nesta terceiro artigo da série, você aprendeu sobre o conceitos de Integridade Referencial, sem dúvidas um dos conceitos mais importantes do Modelo Relacional, juntamente com os conceitos de Relacionamentos, abordados na Parte II. Na Parte IV falarei sobre os princípios básicos de normalização de tabelas. Até lá.

### O Modelo Relacional de Dados - Parte 04

Na primeira parte deste artigo falei sobre Entidades (tabelas), Atributos (campos) e sobre o conceito

de Chave Primária. Na segunda parte falei sobre Relacionamentos e tipos de relacionamentos. Na terceira parte falei sobre um dos conceitos mais importantes do modelo relacional de dados: Integridade Referencial. Também mostrei um exemplo prático de como configurar Relacionamentos e Integridade Referencial no Microsoft Access. Nesta quarta parte sobre os fundamentos do Modelo Relacional de dados, falarei sobre Normalização de banco de dados e as três principais formas normais.

**Nota:** Os exemplos apresentados utilizarão telas do Microsoft Access e o arquivo de exemplos Northwind.mdb, o qual é instalado juntamente com o Microsoft Access. Este arquivo está disponível, por padrão, no seguinte caminho:

<C:\Arquivos de programas\Microsoft Office\Office\Samples>

Porém os princípios básicos do modelo relacional aplicam-se a qualquer banco de dados baseado no modelo relacional de dados. Estes bancos de dados são algumas vezes denominados: SGBDR - Sistemas Gerenciadores de Banco de Dados Relacionais.

### Normalização de tabelas

**Objetivo:** O objetivo da normalização é evitar os problemas provocados por falhas no Projeto do Banco de Dados, bem como eliminar a "mistura de assuntos" e as correspondentes repetições desnecessárias de dados. Uma Regra de Ouro que devemos observar quando do Projeto de um Banco de Dados baseado no Modelo Relacional de dados é a de "**não misturar assuntos em uma mesma Tabela**". Por exemplo na Tabela Clientes devemos colocar somente campos relacionados com o assunto Clientes. Não devemos misturar campos relacionados com outros assuntos, tais como Pedidos, Produtos, etc. Essa "Mistura de Assuntos" em uma mesma tabela acaba por gerar repetição desnecessária dos dados bem como inconsistência dos dados.

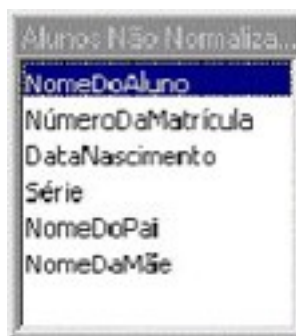
O Processo de Normalização aplica uma série de Regras sobre as Tabelas de um Banco de Dados, para verificar se estas estão corretamente projetadas. Embora existam 5 formas normais (ou regras de Normalização), na prática usamos um conjunto de 3 Formas Normais.

Normalmente após a aplicação das Regras de Normalização, algumas tabelas acabam sendo divididas em duas ou mais tabelas, o que no final gera um número maior de tabelas do que o originalmente existente. Este processo causa a simplificação dos atributos de uma tabela, colaborando significativamente para a estabilidade do modelo de dados, reduzindo-se consideravelmente as necessidades de manutenção. Vamos entender o Processo de Normalização na Prática, através de exemplos.

#### Primeira Forma Normal:

"Uma Tabela está na Primeira Forma Normal quando seus atributos não contém grupos de Repetição".

Por isso dissemos que uma Tabela que possui Grupos de Repetição não está na Primeira Forma Normal. Considere a estrutura da Tabela Indicada na Próxima Figura:



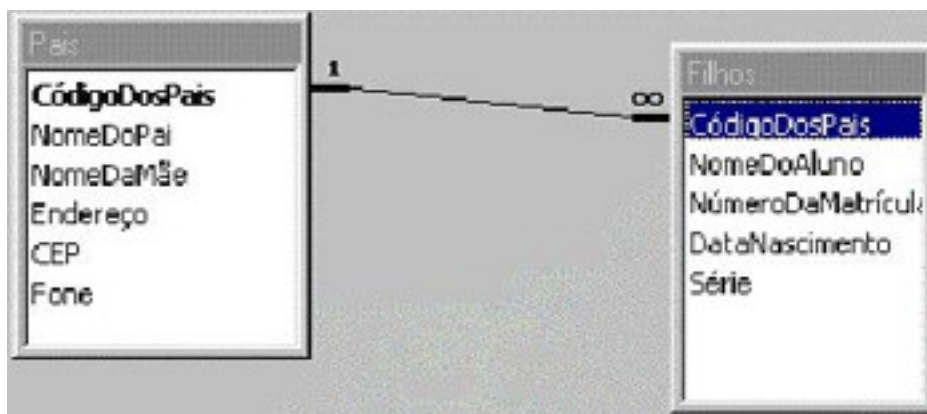
NomeDoAluno
NúmeroDaMatricula
DataNascimento
Série
NomeDoPai
NomeDaMãe

### Tabela que não está na Primeira Forma Normal

Uma tabela com esta estrutura apresentaria diversos problemas. Por exemplo se um casal tiver mais de um filho, teremos que digitar o Nome do Pai e da Mãe diversas vezes, tantas quantos forem os filhos. Isso forma um Grupo de Repetição. Além do mais pode ser que por erro de digitação o Nome dos Pais não seja digitado exatamente igual todas as vezes, o que pode acarretar problemas na hora de fazer pesquisas ou emitir relatórios.

Este problema ocorre porque "Misturamos Assuntos" em uma mesma tabela. Colocamos as informações dos Pais e dos Filhos em uma mesma tabela. A solução para este problema é simples: Criamos uma tabela separada para a Informação dos Pais e Relacionamos a tabela Pais com a Tabela Filhos através de um relacionamento do tipo Um para Vários, ou seja, um casal da Pais pode ter Vários Filhos.

Observe na figura abaixo as duas tabelas: Pais e Filhos, já normalizadas.



### Informações sobre Pais e Filhos em Tabelas Separadas

As duas tabelas Resultantes da Aplicação da Primeira Forma Normal: Pais e Filhos estão na Primeira Forma Normal, a Tabela Original, a qual misturava informações de Pais e Filhos, não estava na Primeira forma Normal

### Segunda Forma Normal:

Ocorre quando a chave Primária é composta por mais de um campo. Neste caso, devemos observar



se todos os campos que não fazem parte da chave dependem de todos os campos que compõem a chave. Se algum campo depender somente de parte da chave composta, então este campo deve pertencer a outra tabela. Observe o Exemplo Indicado na Tabela da Figura abaixo:

Cursos	
NúmeroDaMatrícula	CódigoDoCurso
Avaliação	
DescriçãoDoCurso	

### Tabela com uma Chave Primária Composta. Não está Na Segunda Forma Normal

A Chave Primária Composta é formada pela combinação dos Campos "NúmeroDaMatrícula" e "CódigoDoCurso". O Campo Avaliação depende tanto do CódigoDoCurso quanto do NúmeroDaMatrícula, porém o campo DescriçãoDoCurso, depende apenas do CódigoDoCurso, ou seja, dado o código do curso é possível localizar a respectiva descrição, independentemente do NúmeroDaMatrícula. Com isso temos um campo que não faz parte da Chave Primária e depende apenas de um dos campos que compõem a chave Primária Composta, por isso que dizemos que esta tabela não está na Segunda Forma Normal.

A Resolução para este problema também é simples: "Dividimos a Tabela que não está na Segunda Forma Normal em duas outras tabelas, conforme indicado pela figura abaixo, sendo que as duas tabelas resultantes estão na Segunda Forma Normal.



### Informações sobre Avaliações e Cursos em Tabelas Separadas

**Obs.:** A Distinção entre a Segunda e a Terceira forma normal, que veremos logo em seguida, muitas vezes é confusa. A Segunda Forma normal está ligada a ocorrência de Chaves Primárias compostas.

#### Terceira Forma Normal:

Na definição dos campos de uma entidade podem ocorrer casos em que um campo não seja dependente diretamente da chave primária ou de parte dela, mas sim dependente de um outro campo da tabela, campo este que não a Chave Primária.

Quando isto ocorre, dizemos que a tabela não está na Terceira Forma Normal, conforme indicado pela tabela da figura abaixo:



Funcionários não n...
<b>NúmeroDaMatricula</b>
NomeFuncionário
CódigoDoCargo
DescriçãoDoCargo

Tabela com um Campo dependente de Outro campo que não a Chave Primária. Não está na Terceira Forma Normal

Observe que o Campo **DescriçãoDoCargo** depende apenas do Campo **CódigoDoCargo**, o qual não faz parte da Chave Primária. Por isso dizemos que esta tabela não está na terceira forma normal. A Solução deste problema também é simples. Novamente basta dividir a tabela em duas outras, conforme indicado pela figura a seguir. As duas tabelas resultantes estão na Terceira Forma Normal.

Funcionários	Cargos
<b>NúmeroDaMatricula</b>	<b>CódigoDoCargo</b>
NomeFuncionário	DescriçãoDoCargo
CódigoDoCargo	

Tabelas Resultantes que estão na Terceira Forma Normal

Com isso podemos concluir que como resultado do Processo de Normalização, iremos obter um número maior de tabelas, porém sem problemas de redundância e inconsistência dos dados.

#### Conclusão:

Neste quarto artigo da série você aprendeu sobre o conceitos de Normalização e formas normais. Sem dúvidas um dos conceitos mais importantes do Modelo Relacional. Na quinta e última parte falarei sobre os princípios básicos para projeto de um banco de dados. Até lá!

## O Modelo Relacional de Dados - Parte 05

**Objetivo:** Na primeira parte deste artigo falei sobre Entidades (tabelas), Atributos (campos) e sobre o conceito de Chave Primária. Na segunda parte falei sobre Relacionamentos e tipos de relacionamentos. Na terceira parte falei sobre um dos conceitos mais importantes do modelo relacional de dados: Integridade Referencial. Também mostrei um exemplo prático de como

configurar Relacionamentos e Integridade Referencial no Microsoft Access. Na quarta parte sobre os fundamentos do Modelo Relacional de dados, falei sobre Normalização de banco de dados e as três principais formas normais.

Nesta quinta e última parte falarei sobre o projeto de banco de dados e ressaltarei a importância de fazer a Modelagem do Banco de Dados, antes de partir para a implementação prática. Falarei também sobre a Arquitetura do Microsoft Access. O entendimento do Modelo Relacional e da Arquitetura do Microsoft Access são fundamentais.

Muitos dos e-mails que recebo, com dúvidas sobre o Access, são relativo à dúvidas que seriam solucionadas com o conhecimento do Modelo Relacional e da Arquitetura do Access. O que acontece, muitas vezes, é que o usuário parte diretamente para o uso do Access, criando tabelas, consultas, formulários e relatórios, sem antes ter feito um projeto cuidadoso da estrutura do banco de dados, implementando relacionamentos, fazendo a normalização e impondo Integridade referencial. Trabalhar desta maneira é como fazer um prédio sem ter antes feito um estudo do terreno, ter feito a planta e os cálculos estruturais. Você moraria em um prédio construído desta maneira? Eu não.

Nota: Os exemplos apresentados utilizarão telas do Microsoft Access e o arquivo de exemplos Northwind.mdb, o qual é instalado juntamente com o Microsoft Access. Este arquivo está disponível, por padrão, no seguinte caminho:

C:\Arquivos de programas\Microsoft Office\Office\Samples

Porém os princípios básicos do modelo relacional aplicam-se a qualquer banco de dados baseado no modelo relacional de dados. Estes bancos de dados são algumas vezes denominados: SGBDR - Sistemas Gerenciadores de Banco de Dados Relacionais.

Normalização de tabelas

Objetivo: O objetivo da normalização é evitar os problemas provocados por falhas no Projeto do Banco de Dados, bem como eliminar a "mistura de assuntos" e as correspondentes repetições desnecessárias de dados.

### **Projetando um Banco de Dados**

Neste item você aprenderá a projetar um Banco de Dados. Você aplicará os conhecimentos sobre Tabelas, Campos, Relacionamentos, Chave Primária e Normalização, vistos anteriormente, nas primeiras partes deste artigo.

Antes de começar a trabalhar com o Microsoft Access, é preciso fixar bem os conceitos vistos nas partes anteriores, aplicando-os no Projeto de um Banco de Dados. Um banco de dados bem projetado fornece um acesso conveniente às informações desejadas. Com uma boa estrutura, gasta-se menos tempo na construção de um banco de dados e, ao mesmo tempo, assegura-se resultados mais rápidos e precisos. Nunca é demais lembrar que jamais devemos misturar assuntos em uma mesma tabela.

Etapas na estruturação e projeto de um Banco de dados:

- Determinar qual o objetivo do banco de dados: Isto ajuda na determinação de quais os dados devem ser armazenados. É fundamental ter bem claro qual o objetivo a ser alcançado com o banco de dados. É fazer o acompanhamento das despesas, a evolução das vendas ou outro objetivo qualquer.
- Determinar as tabelas necessárias: Após definirmos os objetivos do Banco de Dados, as informações devem ser definidas e separadas em assuntos diferentes, tais como "Clientes", "Empregados", "Pedidos", pois cada um irá compor uma tabela no banco de dados. Lembre-se da regrinha número um: "Não misturar assuntos na mesma tabela", ou seja, uma coisa é uma coisa e outra coisa é outra coisa.

- Determinar os Campos de cada Tabela: Definir quais informações devem ser mantidas em cada tabela. Por exemplo, a tabela Clientes poderia ter um campo para o Código Do Cliente, outro para o Nome Do Cliente e assim por diante.
- Determinar a Chave Primária de cada tabela, sendo que pode haver tabelas onde não exista uma chave primária: Determinar, em cada tabela, quais campos serão utilizados como Chave Primária. Esta é uma etapa importantíssima para a definição dos Relacionamentos que vem a seguir. Pode haver tabelas onde não exista uma chave primária.
- Determinar os Relacionamentos: Decidir como os dados de uma tabela se relacionam com os dados de outras tabelas. Por exemplo, Clientes podem Fazer Vários Pedidos, então existe um relacionamento do tipo Um-para-vários entre a tabela Clientes (lado um) e a tabela Pedidos (lado vários). Fornecedores podem fornecer Vários Produtos, etc.
- Refinar a Estrutura do Banco de Dados: Antes de inserir muitos dados, ou até mesmo antes de inserir qualquer dado, verificar se a estrutura contém erros, isto é, verificar se os resultados obtidos são os desejados. Isto, normalmente, pode ser obtido através do processo de Normalização. Caso necessário, deve-se alterar a estrutura do banco de dados.

Com uma boa estrutura, gasta-se menos tempo na construção e manutenção do banco de dados e, ao mesmo tempo, assegura-se resultados mais rápidos e precisos.

### **Dicas para determinação dos campos de uma Tabela:**

- Relacionar diretamente cada campo ao assunto da tabela: Se um campo descreve o assunto de uma tabela diferente, este campo deve pertencer a outra tabela. O mesmo acontece quando uma informação se repete em diversas tabelas. Este é um indício de que existem campos desnecessários em algumas tabelas.
- Não Incluir dados Derivados ou Calculados: Não é recomendado armazenar o resultado de cálculos nas tabelas. O correto é que o cálculo seja executado quando necessitarmos do resultado, normalmente em uma consulta.
- Incluir todas as informações necessárias: Como é fácil esquecer informações importantes, deve-se ter em mente todas as informações coletadas desde o início do processo e perguntar se com elas é possível obter todos os resultados desejados.
- Armazenar todas as informações separadamente: Existe uma tendência em armazenar informações em um único campo. Por exemplo, o nome do curso e o tempo de duração em uma mesmo campo. Como as duas informações foram combinadas em um único campo, ficará difícil conseguir um relatório classificado pelo tempo de duração dos cursos.

### **Como selecionar o campo que será a Chave Primária?**

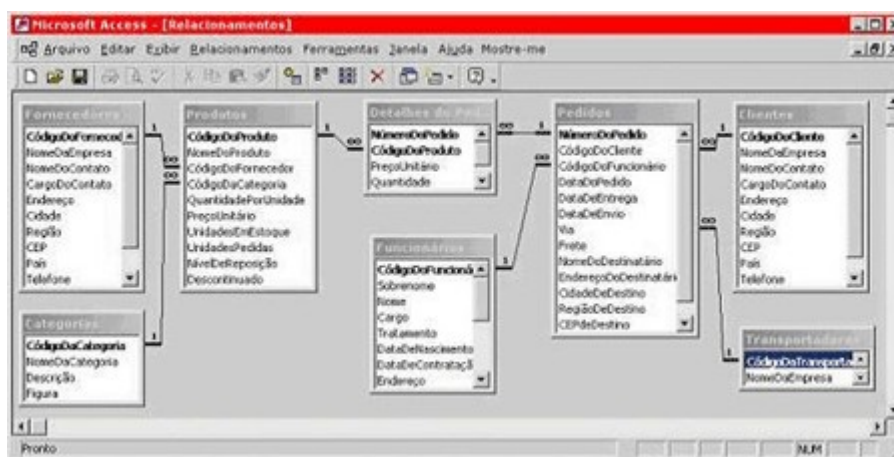
Um bom Sistema Gerenciador de Banco de Dados (SGBD) é aquele que encontra e nos fornece, rapidamente, todas as informações necessárias que nele estejam armazenadas, mesmo que estas informações estejam em diferentes tabelas. Para que isto seja possível é necessário incluir um campo ou conjunto de campos que identifiquem de modo único cada registro de uma tabela. Esta informação é chamada Chave Primária. Deve-se ter certeza que este campo (ou conjunto de campos) seja sempre diferente para cada registro, por não ser permitido valores duplicados em um campo de chave primária.

Ao escolher campos de Chave Primária, considere os seguintes detalhes:

- Não é permitido duplicidade de valores ou nulos (informações desconhecidas).
- Caso não exista um identificador único para uma determinada tabela, pode-se usar um campo que numere os registros sequencialmente.

- Pode-se utilizar o valor deste campo para encontrar registros.
- O tamanho da chave primária afeta a velocidade das operações, portanto, para um melhor desempenho, devemos utilizar o menor tamanho que acomode os valores necessários que serão armazenados no campo.

Agora que já revisamos diversos conceitos importantes sobre banco de dados vamos colocá-los em prática, através de um exercício de Projeto de Banco de Dados. Será apresentada uma determinada situação e você deverá projetar o Banco de Dados para atender a Situação Solicitada. Projetar o banco de dados significa fazer um diagrama Entidade x Relacionamentos onde são indicadas quais tabelas farão parte do banco de dados, quais os campos de cada tabela, qual o campo que será a Chave Primária nas tabelas que terão Chave Primária e quais os relacionamentos entre as tabelas. Na figura a seguir temos um exemplo de um diagrama Entidades x Relacionamentos:



**Clique na imagem para vê-la em tamanho real:**

**Nota:** Os campos que aparecem em **negrito** representam a Chave Primária de cada tabela.

**Exercício:** Imagine que você está projetando um Banco de Dados para uma Escola. Este Banco de Dados deverá conter informações sobre os Alunos, os Pais dos Alunos, As matérias em que cada aluno está matriculado (imagine que alunos da mesma série podem estar matriculados em diferentes matérias), as notas do aluno em cada matéria e em cada bimestre, bem como todo o histórico do aluno na escola. O histórico inclui as notas do aluno em cada matéria em cada um dos anos em que ele esteve na escola. O banco de dados deve manter um cadastro de alunos, dos pais dos alunos, das disciplinas ofertadas, da nota de cada aluno em cada disciplina, e em que disciplina cada aluno está matriculado.

O Sistema deverá ser capaz de fornecer, a qualquer momento, a situação atual do aluno em termos de suas notas, bem como todo o seu histórico. O Sistema não deve permitir que seja cadastrado um aluno sem antes serem cadastrados os seus pais. Além disso todo aluno terá um número de

matrícula que é único. Cada disciplina também terá um código único.

O Sistema deve ser capaz de emitir relatórios com as notas por turma e por bimestre, além das médias para cada disciplina.

Projete um Banco de Dados capaz de atender a estas necessidades. O Resultado final do seu trabalho será o "Diagrama Entidades x Relacionamentos", com as Tabelas, Campos de Cada Tabela, Chaves Primárias e Relacionamentos entre as tabelas.

Ao Final do Processo, aplique o processo de Normalização para verificar se a estrutura apresenta algum tipo de Problema.

### **Arquitetura do Microsoft Access**

Neste item iremos analisar a Arquitetura do Microsoft Access Veremos os diversos elementos que podem fazer parte de um Banco de Dados do Microsoft Access, bem como os relacionamentos entre estes diversos elementos. Veremos também alguns exemplos práticos de solução de problemas.

#### **Abordarei os seguintes tópicos:**

- Os diversos elementos do Microsoft Access e a Relação entre os eles.
- Alguns Exemplos de Situações do dia-a-dia.

#### **Os Diversos Elementos do Access e a Relação entre eles:**

Um banco de dados é uma coleção de informações relacionadas a um determinado assunto ou finalidade, como controle de pedidos dos clientes ou manutenção de uma coleção de CDs e assim por diante. Se o seu banco de dados não está armazenado em um computador, ou se somente partes dele está, você pode estar controlando informações de uma variedade de fontes, tendo que coordená-las e organizá-las você mesmo.

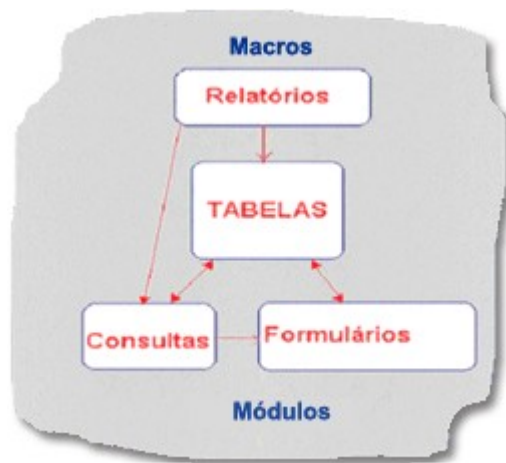
Um arquivo .mdb é um Banco de Dados do Microsoft Access. Esse banco de dados contém diversos elementos: Tabelas, Consultas, Formulários, Relatórios, Macros, Páginas de dados e Módulos.

Utilizando o Microsoft Access, você pode gerenciar todas as suas informações a partir de um único arquivo de banco de dados. Dentro do arquivo, divida seus dados em compartimentos de armazenamento separados denominados tabelas (ou entidades); visualize, adicione e atualize os dados da tabela utilizando formulários on-line; localize e recupere apenas os dados desejados utilizando consultas; e analise ou imprima dados em um layout específico utilizando relatórios.

Para armazenar seus dados, crie uma tabela para cada tipo de informação que você registra. Para reunir os dados de várias tabelas em uma consulta, formulário ou relatório, você define relacionamentos entre as tabelas. Aqui nos temos dois fatos de grande importância:

- Todos os dados ficam armazenados em Tabelas. Quando uma Consulta exibe os resultados com base em um Critério, na verdade ele está buscando os dados em uma determinada tabela. Quando um formulário exibe um determinado registro, ele também está buscando estes dados em uma determinada tabela. No Microsoft Access, o único local onde os dados ficam armazenados é nas tabelas.
- Mesmo que as informações estejam separadas em diferentes tabelas (Clientes, Pedidos, Detalhes do Pedido, etc) é possível reuni-las em Consultas, Relatórios e Formulários. Por exemplo, posso criar um Relatório de Vendas por Cliente, classificados pelo País de Destino.

Na Figura a seguir, vemos os diversos elementos que formam um Banco de Dados do Microsoft Access, bem como o Relacionamento entre os diversos elementos.



### Os Diversos Elementos de um Banco de Dados do Microsoft Access.

Nunca é demais salientar que o único local onde ficam armazenados os dados é nas tabelas. Por isso que ao construirmos uma consulta, formulário ou relatório, o Microsoft Access solicita os dados para a tabela na qual a consulta, formulário ou relatório está baseado.

#### Algumas Observações sobre os Elementos do Microsoft Access:

- Observe o Relacionamento que existe entre os Elementos. Uma consulta é baseada em uma tabela, isto é, os dados que a consulta exibe são buscados a partir de uma ou mais tabelas. Se os dados forem alterados na consulta, na verdade estas alterações são refletidas diretamente na tabela. Por isso uma seta de dupla mão entre tabelas e consultas. As mesmas observações são válidas para a relação entre formulários e tabelas.
- Um relatório também pode ser baseado diretamente em uma consulta, assim como um Formulário também pode ser baseado diretamente em uma Consulta. Quando o Formulário (ou Relatório) é aberto, o Microsoft Access executa a consulta, a qual busca os dados na Tabela, e retorna os dados para o Formulário (Ou Relatório).
- Observe que as Macros e Módulos foram colocados ao redor, envolvendo os demais elementos. Isto significa que posso ter Macros e Módulos interagindo com qualquer elemento de um banco de dados do Microsoft Access. Por exemplo, posso criar uma macro que Maximize um formulário quando o formulário é aberto. Posso criar um módulo para calcular o Dígito Verificador de um campo CPF, de tal forma que quando um CPF é digitado, o CPF não é aceito se estiver com o Dígito Verificador incorreto.
- Uma situação bastante comum é o caso em que precisamos de um relatório, porém os dados necessários não estão na forma necessária nas tabelas. Neste caso podemos criar uma consulta que



selecione os dados necessários e faça as consolidações necessárias e criamos o Relatório baseado nesta consulta e não diretamente na tabela.

Estes seis elementos: Tabelas, Consultas, Formulários, Relatórios, Macros e Módulos podem ser criados e gerenciados a partir da Janela Principal do Banco de Dados do Microsoft Access, conforme indicado a seguir:

Janela "Banco de Dados", dando acesso aos diversos elementos do Microsoft Access.

### **Alguns exemplos e situações do dia-a-dia:**

Para ilustrar o relacionamento entre os diversos elementos do Microsoft Access, vamos considerar algumas situações usuais do dia-a-dia. Nossas situações serão baseadas no arquivo Northwind.mdb, o qual é instalado juntamente com o Microsoft Access, conforme descrito na Introdução.

**Situação 01:** Vamos supor que seja solicitado um Relatório com os totais por Pedido. Como atender esta demanda?

**Solução:** Os dados necessários não estão disponíveis diretamente na tabela Pedidos. Criaremos uma consulta baseada nas tabelas Pedidos e Detalhes do Pedido. Esta consulta fará o cálculo do total por Número de Pedido. Nosso Relatório será baseado nesta consulta. Quando o Relatório for aberto, a consulta será acionada, buscará os valores nas tabelas Pedidos e Detalhes dos Pedidos, realizará os cálculos e fornecerá os valores para o Relatório. Aprenderemos a criar este tipo de consulta no decorrer deste curso.

**Situação 02:** Como fazer um relatório que exiba o total de Vendas por país de Destino e por Ano?

**Solução:** Novamente os dados necessários não estão disponíveis diretamente nas tabelas Pedidos e Detalhes do Pedido. Crio uma consulta do tipo Tabela de Referência Cruzada, baseada na Consulta que calcula os totais por Pedido. Nesta consulta adiciono um campo para o Ano do Pedido e outro campo para o País de Destino. Crio o meu relatório baseado nesta consulta. Ao ser aberto o Relatório é acionada a consulta do tipo Tabela de Referência Cruzada, a qual por sua vez aciona a Consulta na qual ela é baseada, a qual busca os dados nas tabelas. O Caminho inverso é percorrido, até que os dados são fornecidos para o Relatório, o qual exibe os mesmos na tela ou imprime na Impressora. Aprenderemos a criar este tipo de consulta no decorrer deste curso.

**Situação Desafio:** Os dados sobre o cabeçalho do Pedido ( tabela Pedidos) e os diversos itens de cada Pedido (Tabela Detalhes do Pedido) estão separados. Como reunir estas informações em um formulário de tal maneira que o mesmo se pareça com uma Nota Fiscal?

**Solução:** Desafio!

### **Conclusão:**

Bem amigo leitor, com este artigo, encerramos a série sobre o Modelo Relacional de dados. A recomendação mais enfática que eu posso fazer é a seguinte: "Não pense que fazer modelagem de dados é perda de tempo". Muito pelo contrário: É ganho de tempo e de qualidade em nossos programas. Sem a modelagem, você vai criando o banco na tentativa e erro, quando uma consulta não dá certo o programador faz um "gambiarra", adapta aqui, adapta ali e assim vai. Isso não é programação. Perdoem-me por ser enfático neste ponto. Isso é remendo, é tentativa e erro. Por isso que temos muitos programas com péssimo desempenho, funcionalidades que deixam a desejar, sem documentação, impossíveis de serem alterados a não ser por quem os criou.

É meu mais sincero desejo que eu possa ter colaborado para, pelo menos, despertar uma reflexão no amigo leitor. Não deixe de acompanhar a série de artigos sobre TCP/IP, a qual passarei a publicar quinzenalmente. Um forte abraço.

**Links dos originais:**

[http://imasters.uol.com.br/artigo/2419/bancodedados/o\\_modelo\\_relacional\\_de\\_dados\\_-\\_parte\\_01/](http://imasters.uol.com.br/artigo/2419/bancodedados/o_modelo_relacional_de_dados_-_parte_01/)

[http://imasters.uol.com.br/artigo/2455/bancodedados/o\\_modelo\\_relacional\\_de\\_dados\\_%E2%80%93\\_parte\\_02/](http://imasters.uol.com.br/artigo/2455/bancodedados/o_modelo_relacional_de_dados_%E2%80%93_parte_02/)

[http://imasters.uol.com.br/artigo/2477/bancodedados/o\\_modelo\\_relacional\\_de\\_dados\\_-\\_parte\\_03/](http://imasters.uol.com.br/artigo/2477/bancodedados/o_modelo_relacional_de_dados_-_parte_03/)

[http://imasters.uol.com.br/artigo/2521/bancodedados/o\\_modelo\\_relacional\\_de\\_dados\\_-\\_parte\\_04/](http://imasters.uol.com.br/artigo/2521/bancodedados/o_modelo_relacional_de_dados_-_parte_04/)

[http://imasters.uol.com.br/artigo/2530/bancodedados/o\\_modelo\\_relacional\\_de\\_dados\\_-\\_parte\\_05/](http://imasters.uol.com.br/artigo/2530/bancodedados/o_modelo_relacional_de_dados_-_parte_05/)