Fábio J. L. Ferreira

Banco de dados

Modelagem de dados

Licença

Você pode:

Copiar, distribuir, exibir e criar obras derivadas deste material.

Com as seguintes condições:

Você deve dar crédito ao autor original e ao projeto *TI na Rede*, citando o nome do autor (quando disponível) e fazendo referência ao site oficial do projeto.

Os direitos de comercialização desta obra são de exclusividade do mantenedor oficial do projeto *TI na Rede*. Exceto nos casos em que o terceiro tenha uma autorização formal do mantenedor do projeto.

Se você alterar, transformar ou criar outra obra com base nesta, somente poderá distribuí-la sob uma licença equivalente à original.

Acreditamos que ao compartilhar o conhecimento de forma aberta e gratuita estamos, no mínimo, incentivando terceiros a buscar um aprimoramento profissional. Inevitavelmente, contribuímos ainda para o crescimento pessoal e moral dessas pessoas.

Todo e qualquer material disponibilizado por meio deste projeto é revisado, tendo como objetivo garantir um alto nível de qualidade e excelência. Não nos responsabilizamos de forma alguma por quebra de direitos autorais ou violação de propriedade intelectual que possa resultar em um ato considerado ilegal. Cada autor é responsável por responder legalmente por seu material.

Sumário

Introdução

Estamos vivendo um período de forte articulação e evolução tecnológica, no qual não somos os únicos a serem bombardeados diariamente por inovações. As empresas de todos os segmentos se veem obrigadas a implantar sistemas informatizados cada vez mais complexos, tendo como objetivo otimizar seus processos e garantir a produtividade, visando quase sempre ao lucro.

Comentário do autor: *Esse é o mundo capitalista!*

Com essa rápida evolução da tecnologia e de seus meios de comunicação, surgiu um sério problema: como guardar de forma estruturada o volume cada vez maior de dados? Para uma melhor compreensão desse questionamento, nada melhor do que pensar em uma situação hipotética:

As operadoras de telefonia móvel são obrigadas a manter algumas informações por um longo período de tempo, tais como dados pessoais dos clientes, histórico de ligações, mensagens enviadas, consumo de dados da internet etc.

Alguns desses dados são guardados para que a operadora tenha condições de faturar os clientes ao final do mês; já outros são guardados porque legalmente a Justiça ordena que determinadas informações sejam mantidas por um período Y de tempo.

Parece pouco? Pense bem: milhões de clientes, cada um enviando diariamente dezenas de mensagens, fazendo dúzias de ligações, consumindo serviços de dados etc. Ainda parece pouco?

Agora, vem a pergunta: como guardar todos esses dados para que mais tarde eles possam ser recuperados e processados conforme for necessário? Para resolver parte do problema surgiram os famosos bancos de dados, tema alvo desta obra.

Este material pode ser considerado uma introdução ao tema banco de dados; o conteúdo de nossos estudos vai do básico até o intermediário. Não é de meu interesse focar em um software ou ferramenta específica, mas sim transmitir a parte conceitual e teórica do tema alvo desta obra, um conhecimento que considero no mínimo obrigatório. Pense nesta obra como um ponto de partida para o estudo de bancos de dados.

Aproveito para deixar um alerta: *engana-se quem pensa que o mercado de TI está carente de vagas de trabalho. Na verdade, nos últimos anos esse mercado tem enfrentado um sério problema de falta de profissionais qualificados para ocupar postos de trabalho nos mais diversos níveis hierárquicos das organizações.* ***Sendo assim, sugiro que invista em si mesmo; essa é sua melhor aposta e também sua garantia de sucesso!***

Chega de blá-blá-blá e vamos direto ao que interessa. Boa leitura! ;)

Capítulo 1

Banco de dados

"O único lugar onde o sucesso vem antes do trabalho é no dicionário."
— Albert Einstein

Antes dos computadores, as empresas mantinham caixas e mais caixas de registros impressos; nelas estavam a contabilidade da empresa, os dados de clientes, os dados de funcionários etc. O problema da papelada é que com o passar do tempo ela aumenta, ocupa mais espaço, se deteriora etc.

Com o surgimento dos computadores as empresas passaram a adotá-los com uma frequência cada vez maior, desenvolveram programas que permitiam manter os registros antes impressos gravados no sistema de arquivos da máquina e, com isso, diminuíram a papelada. Os arquivos se multiplicaram rapidamente e surgiram limitações como a impossibilidade de acesso simultâneo aos dados, a inconsistência e outros fatores, que passaram a ser novos problemas.

Na busca incessante por uma solução, no final da década de 1960 e início dos anos 70 surgiram os primeiros bancos de dados — a solução que todos esperavam.

1.1 O que é um banco de dados

É um mecanismo/uma técnica capaz de armazenar de forma estruturada e organizada coleções de dados de tamanho e tipo variáveis. Em uma explicação menos técnica, um banco de dados é um conjunto de arquivos muito bem estruturado em que os dados são gravados obedecendo a padrões organizacionais predefinidos. Uma lista telefônica pode ser considerada um ótimo exemplo de banco de dados; nela os contatos estão gravados de uma forma tão bem organizada que fica fácil e rápido achá-los.

Os dados continuam sendo gravados no sistema de arquivos do computador, porém o banco de dados agora é o maestro da história. Nos tópicos a seguir entenderemos melhor essa explicação.

1.2 Dados *vs* informações

Não confunda essas duas palavras, pois elas não são sinônimas. Veja:

- **Dados**: São valores brutos, ou seja, não possuem significado relevante. Exemplo: 401.
- **Informação**: É o agrupamento, a ordenação e/ou o processamento dos dados de forma a transmitir significado dentro de um contexto. Exemplo: Moro no apartamento 401.

O número 401 é um exemplo de **dado** porque não possui nenhum significado, é somente um valor muito específico "jogado ao vento". Já a sentença Moro no apartamento 401, oferecida como exemplo de **informação**, fez com que o número passasse a ter um significado. Notou a diferença?

1.3 Sistema Gerenciador de Banco de Dados (SGBD)

O **SGBD**, em inglês **DBMS** (*Database Management System*), é um conjunto de softwares que tem como objetivo facilitar o processo de comunicação com o banco e a manipulação de dados. O SGBD sabe como o banco funciona e como os dados devem ser manipulados; por esses e outros motivos acabou se tornando um facilitador ao simplificar tarefas rotineiras.

Alguns bons exemplos de SGBD são: Oracle Database, MySQL, PostgreSQL, SQL Server etc.

Para concluir, acredito que seja relevante informar que o processo de comunicação simplificado e o acesso simultâneo aos dados são somente alguns dos vários benefícios proporcionados pela utilização dos SGBD.

1.4 Desvantagens do uso de arquivos como banco

A utilização de arquivo bruto para guardar dados possui muitas desvantagens. As principais são:

- Impossibilidade de realizar acesso simultâneo aos dados;
- Dificuldade para gravar, alterar e recuperar dados;
- Inconsistência dos dados;
- Problemas com a integridade dos arquivos.

As duas principais desvantagens da utilização de arquivo bruto e não estruturado para guardar dados são a impossibilidade de acesso simultâneo aos dados e a dificuldade para gravar, alterar ou recuperar dados.

A explicação para o primeiro item é bem simples: os sistemas de arquivos possuem limitações que impedem múltiplos acessos a um arquivo já aberto, ou seja, o arquivo fica bloqueado para acesso até que o processo de quem o está utilizando seja encerrado. Já a segunda desvantagem é resultado da ausência de um padrão organizacional: cada desenvolvedor, analista ou especialista implanta um mecanismo de armazenamento seguindo o seu próprio padrão organizacional.

1.5 Benefícios da utilização do banco de dados

A utilização do banco de dados resultou em inúmeros benefícios. Alguns dos mais relevantes são:

- Alta performance, confiabilidade e integridade da estrutura de dados;
- Transações atômicas (ver tópico seguinte - **ACID**);
- Facilidade na manipulação dos dados (alteração, inclusão, exclusão e recuperação);
- Acesso simultâneo.

Ao utilizar uma estrutura padronizada e com referências organizacionais sólidas, é possível obter um ótimo desempenho e confiabilidade por parte do banco, sem falar, é claro, no benefício de compartilhar com todos os utilizadores um padrão único de comunicação e manipulação de dados.

1.6 ACID (Atomicidade, consistência, isolamento e durabilidade)

ACID é um termo utilizado em ciência da computação para descrever as 4 propriedades que garantem a integridade das transações em um banco de dados. Veja a descrição de cada uma delas:

- **Atomicidade:**

  Essa propriedade garante que todas as transações sejam atômicas (indivisíveis), ou seja, que as transações sejam executadas em sua totalidade. Se ocorrer algum erro, todas as operações que compõem a transação serão descartadas.

- **Consistência:**

  A execução de uma transação deve levar o banco de dados de um estado consistente para outro estado de consistência, ou seja, toda transação deve respeitar as regras de integridade dos dados (tipo de dado, chave primária etc).

- **Isolamento:**

  É um recurso do banco que tem como objetivo evitar que, em um sistema multiusuário, transações em paralelo interfiram umas nas outras.

- **Durabilidade:**

  Significa que os efeitos de uma transação são permanentes, podendo ser desfeitos somente como resultado de uma transação posterior e bem-sucedida.

Ao longo desta obra essas propriedades serão constantemente relembradas e apreciadas como vantagens de se utilizar um banco de dados.

1.7 Tipos de banco

Os conceitos apresentados até aqui se aplicam a todos os tipos de banco de dados, com exceção dos bancos do tipo NoSQL — bancos que em geral quebram alguns conceitos do **ACID**.

Abaixo temos uma lista dos principais tipos de bancos:

- Hierárquico;
- Em rede;
- Relacional (conhecido também como modelo entidade relacional);
- Orientado a objetos;
- Objeto-relacional.

Nesta apostila iremos focar nos bancos do tipo relacional. Escolhi escrever sobre esse tipo pois, a cada 10 instalações de banco, ao menos 8 são do tipo relacional.

1.8 Resumo

Até aqui já sabemos que o banco de dados é um espaço altamente organizado e estruturado que serve para guardar dados, e que o SGBD é um conjunto de softwares que tem como objetivo facilitar o processo de comunicação com o banco e a manipulação dos dados, além de ser responsável por permitir acesso simultâneo aos dados.

Sabemos ainda que as palavras dados e informação não são sinônimas, e que existem diversos tipos de bancos de dados.

O termo ACID descreve as 4 propriedades que garantem a integridade de uma transação e só não se aplica basicamente aos bancos do tipo NoSQL.

Vamos à nossa primeira pausa para um cafezinho e para pôr as ideias em ordem! 5 minutos! 😉

Capítulo 2

Modelo entidade relacional (MER)

"Ensinar é aprender duas vezes."
— Joseph Joubert

No modelo entidade relacional os dados são representados em tabelas, nas quais cada coluna representa um atributo e cada linha, um registro. Veja o exemplo da entidade fictícia **curso**:

codigo	titulo	descricao

Figura 1 - Exemplo de entidade

Na representação acima temos uma entidade (tabela) com 3 atributos — **codigo**, **titulo** e **descricao**.

Em um banco de dados real o nome das entidades, bem como seus atributos, não deve conter acentuação, caracteres especiais ou espaço entre palavras; utilize *underline* (_) caso exista a necessidade de separar um nome composto. Como sugestão, todos os atributos devem ser escritos em caixa baixa ou alta, sem alternância.

2.1 Entidade

Uma entidade representa uma coleção de dados ou um conjunto de "coisas", reais ou abstratas. Digamos que a entidade é o elemento no qual podemos guardar os dados/as informações de forma organizada.

Complicado? Calma, a gente simplifica com um diagrama entidade relacional (DER):

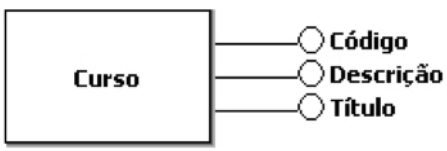


Figura 2 – Exemplo de entidade com atributos

Na imagem acima temos uma representação da entidade **Curso** e de seus atributos — falaremos sobre atributos mais à frente — **código**, **descrição** e **título**. O nome de uma entidade será sempre

escrito no singular, pois por si só ela representa um conjunto (plural).

2.2 Diagrama entidade relacional (DER)

O **DER** é uma representação visual abstrata de alto nível. Nesse tipo de representação somos livres para acentuar as palavras, utilizar o caractere de espaço e alternar entre caixa alta e baixa.

Esse tipo de representação é bem liberal e nos permite ter um entendimento inicial de como nosso banco será modelado.

Existem inúmeras formas de representar um **DER**, porém iremos adotar o modelo visual apresentado no tópico anterior (Figura 2) em razão da sua simplicidade e eficiência.

O DER é um tipo de diagrama que se encaixa na categoria de **Modelo conceitual**. Antes de explicar essa categoria, é necessário saber que existem 3 categorias de modelos de dados. São elas:

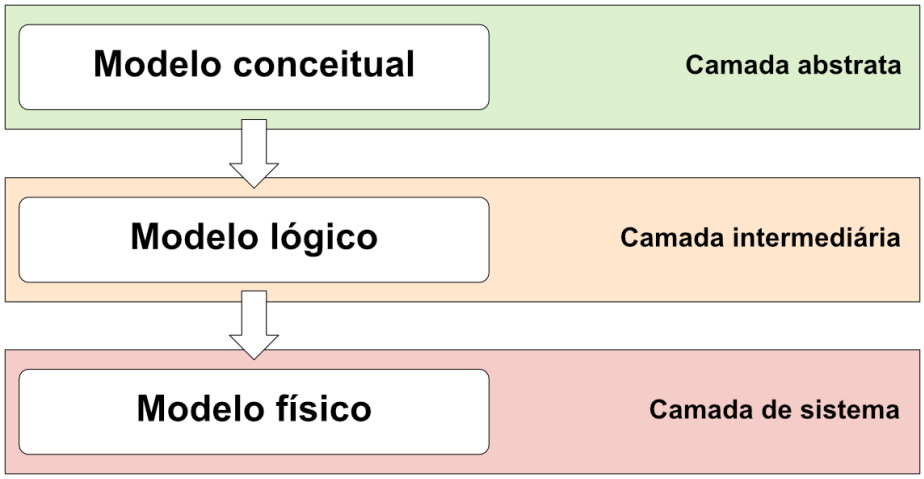


Figura 3 – Representação do conceito de camadas

- **Modelo conceitual**: É um modelo de dados independente do SGBD, ou seja, é uma representação abstrata do banco de dados. O diagrama entidade relacional (DER) se encaixa nesta camada.

- **Modelo lógico**: É um modelo de dados intermediário que está fortemente ligado ao SGBD.

- **Modelo físico**: Este modelo de dados é o que está mais próximo do hardware. Tarefas como recuperar, alterar ou gravar dados no banco são efetuadas por esta camada. Por se tratar de processos que utilizam recursos de sistema, esse nível passou a ser comumente chamado de camada de sistema.

Esses 3 modelos ou, se preferir, essas 3 camadas não são exclusivas dos bancos de dados do tipo relacional. Sendo assim, é correto afirmar que seu conceito se aplica a bancos de outros tipos.

O software não profissional que utilizei para criar o **DER** da Figura 2 se chama **BrModelo**. Download gratuito em: *http://www.sis4.com/brmodelo/download.aspx*

2.3 Relacionamento

Um relacionamento é uma associação entre ocorrências de uma ou mais entidades, ou seja, é uma associação entre registros.

Em outras palavras, estou falando que dois registros podem se completar ou se complementar ao estabelecerem algum tipo de relacionamento entre eles. Veja um exemplo:

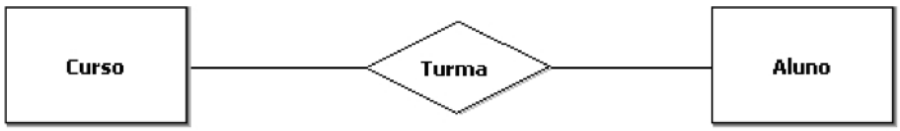


Figura 4 - Exemplo de relacionamento entre ocorrências

Na Figura 4 temos duas entidades — **Curso** e **Aluno** — que se relacionam por meio de uma outra entidade chamada **Turma**. Uma explicação mais simples seria:

- A entidade **Curso** armazena os registros do grupo/tipo curso;
- A entidade **Aluno** armazena os registros do grupo/tipo aluno;
- As entidades **Curso** e **Aluno** se relacionam por meio da entidade **Turma**.

Se pararmos um pouco para pensar, rapidamente conseguimos comparar a Figura 4 ao seguinte cenário do mundo real:

- A instituição fictícia TI na Rede oferece N cursos (entidade **Curso**);
- Essa mesma instituição possui N alunos (entidade **Aluno**);
- Obviamente essa instituição terá N turmas (entidade **Turma**).

A entidade **Turma** é uma forma de relacionar cada aluno a **N** cursos. Esse relacionamento entre curso e aluno poderia ter qualquer nome, mas por uma questão de semântica escolhemos nomeá-lo de **Turma**.

Esse entendimento pode ser um pouco confuso para quem está tendo seu primeiro contato com ele, então para simplificar farei uma demonstração mais "prática":

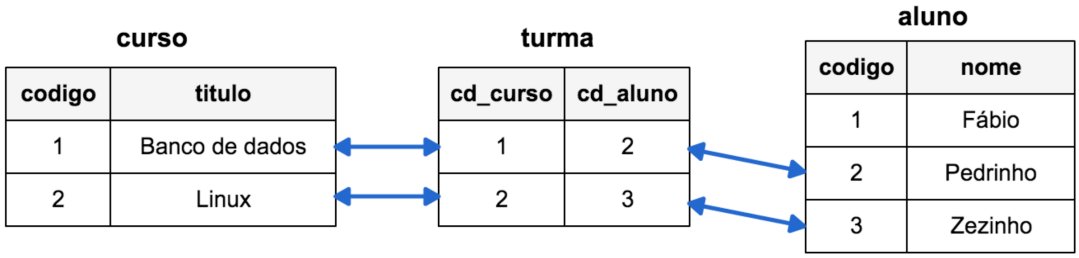


Figura 5 – Exemplo real de relacionamento

A Figura 5 não está politicamente correta — essas setas não existem —, porém, por ora esse é o desenho que irá nos permitir entender melhor o conceito de relacionamento. Então vamos detalhar o que está na figura:

- O **aluno** Fábio (código 1) não possui relacionamento;

- O **aluno** Pedrinho (código 2) se relaciona ao **curso** Banco de dados (código 1);
- O **aluno** Zezinho (código 3) se relaciona ao **curso** Linux (código 2).

Esse relacionamento se dá por meio da entidade de relacionamento **Turma**. Mais à frente veremos outras formas de montar esse mesmo relacionamento utilizando duas entidades. Relacionamentos entre três ou mais entidades também são possíveis, mas não são comuns:

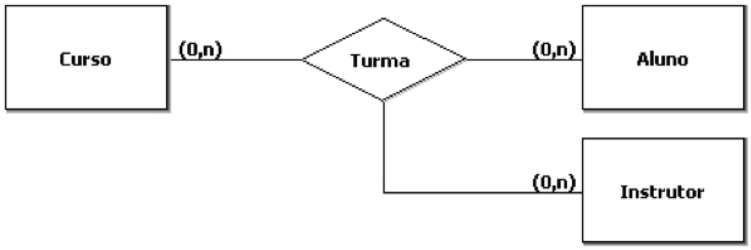
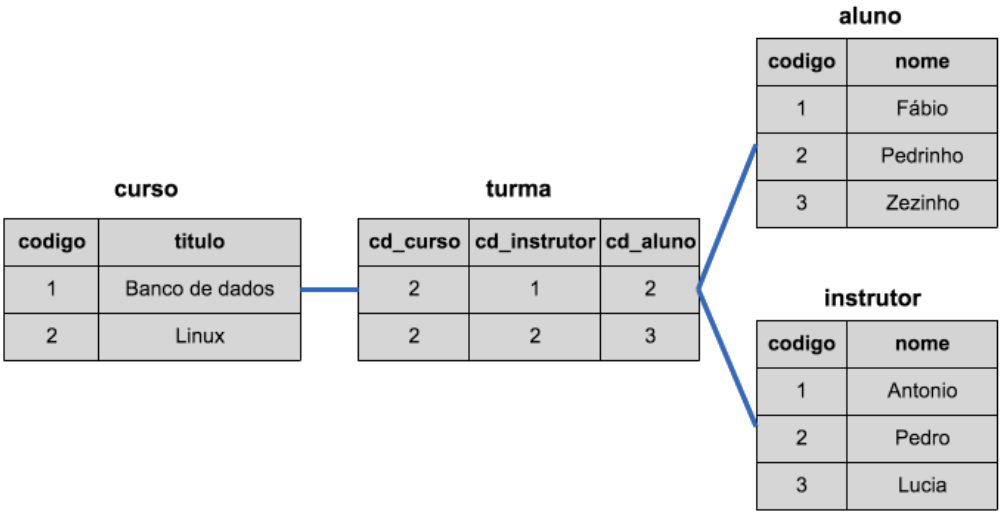


Figura 6 – DER com mais de dois relacionamentos

Uma demonstração mais próxima da realidade seria algo como:



curso

codigo	titulo
1	Banco de dados
2	Linux

turma

cd_curso	cd_instrutor	cd_aluno
2	1	2
2	2	3

aluno

codigo	nome
1	Fábio
2	Pedrinho
3	Zezinho

instrutor

codigo	nome
1	Antonio
2	Pedro
3	Lucia

Figura 7 - Exemplo real de um banco de dados

Em uma analise rápida conseguimos identificar os seguintes relacionamentos:

- O registro de código 2 da entidade **aluno** se relaciona com o registro de código 1 da entidade **instrutor** e com o registro de código 2 da entidade **curso**.
- O registro de código 3 da entidade **aluno** se relaciona com o registro de código 2 da entidade **instrutor** e com o registro de código 2 da entidade **curso**.

Logo, temos o seguinte resultado:

- O aluno **Pedrinho** está fazendo o curso de **Linux** e o seu instrutor é o **Antonio**.
- O aluno **Zezinho** está fazendo o curso de **Linux** e o seu instrutor é o **Pedro**.

Acredito que você tenha observado que cada ocorrência está se relacionando com mais duas ocorrências e que **turma** é a entidade responsável por manter esse relacionamento. No tópico sobre entidades, vimos que a entidade é uma coleção de dados. No exemplo acima temos essa

confirmação: temos várias tabelas, cada uma agrupando sua própria coleção de dados, e graças aos relacionamentos conseguimos cruzar dados e extrai-los, gerando dessa forma informações tais como quem está fazendo qual curso e qual o instrutor de determinado aluno.

2.4 Tipos de relacionamento ou cardinalidade do relacionamento

Caracterizam-se pelo número mínimo e máximo de ocorrências de uma entidade associada a ocorrências de outra entidade.

Um para um (1:1)

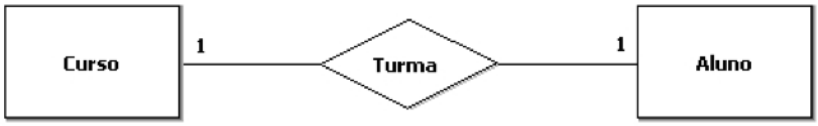


Figura 8 - Relacionamento um para um (1:1)

Uma ocorrência da entidade **Curso** se relaciona obrigatoriamente com somente uma ocorrência da entidade **Aluno** e vice-versa.

Um para muitos (1:N)

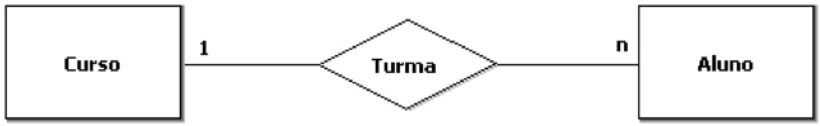


Figura 9 - Relacionamento um para muitos (1:N)

Uma ocorrência da entidade **Curso** se relaciona obrigatoriamente com uma ou muitas ocorrências da entidade **Aluno**. Já a entidade **Aluno** se relaciona com somente uma ocorrência da entidade **Curso**.

Muitos para muitos (N:N)

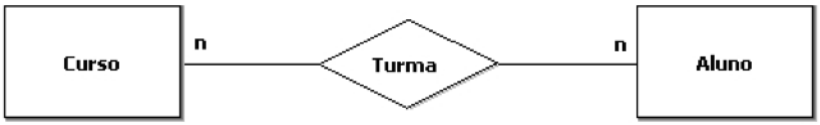


Figura 10 - Relacionamento muitos para muitos (N:N)

Uma ocorrência da entidade **Curso** se relaciona obrigatoriamente com uma ou muitas ocorrências da entidade **Aluno** e vice-versa.

A definição desses 3 tipos de relacionamento pode ser chamada ainda de cardinalidade simples. Sua utilização nos permite ter uma ideia de como as ocorrências de uma entidade se relacionam com ocorrências de outra entidade. Por essa abordagem conseguimos representar somente a cardinalidade máxima permitida, pois temos a representação de um único valor ao lado de cada entidade.

A cardinalidade é uma informação extremamente importante e que deveria obrigatoriamente estar

presente em todo e qualquer relacionamento que exista dentro de um banco de dados com modelo entidade relacional.

Cardinalidade mínima e máxima

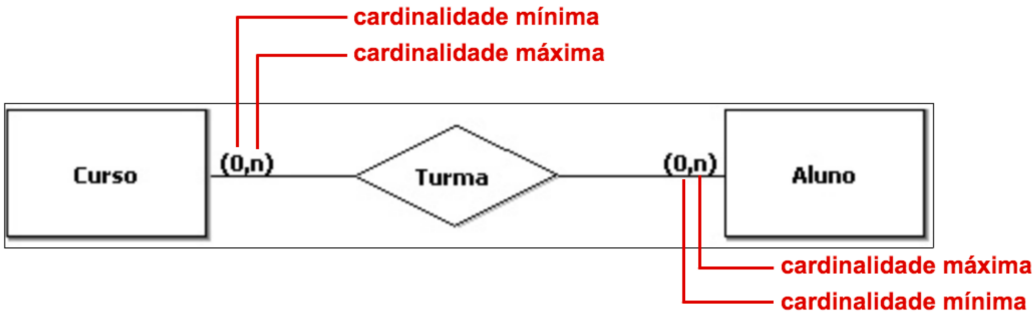


Figura 11 - Cardinalidade máxima e mínima

É importante saber que os valores que estão ao lado da entidade ditam as regras de como as outras entidades irão se comportar diante de um relacionamento.

Exemplo:

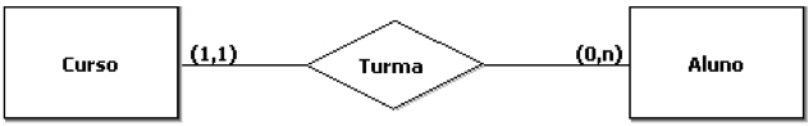


Figura 12 - Exemplo de cardinalidade máxima e mínima

Na Figura 12 temos a seguinte representação graças à cardinalidade:

- Cada ocorrência da entidade **Curso** poderá se relacionar com no mínimo nenhuma e com no máximo várias ocorrências da entidade **Aluno**;
- Cada ocorrência da entidade **Aluno** irá se relacionar com no mínimo e no máximo uma ocorrência da entidade **Curso**.

Obviamente a cardinalidade será definida de acordo com a sua regra de negócio. No meu exemplo utilizei a seguinte lógica: um curso pode ter nenhum ou vários alunos associados a ele, porém, um aluno obrigatoriamente estará associado a somente um curso.

Até aqui apresentei exemplos utilizando as entidades **Curso**, **Turma** e **Aluno**. Então vamos mudar um pouco:

Figura 13 - Exemplo de cardinalidade máxima e mínima

Na Figura 13 temos a seguinte representação graças à cardinalidade:

- Cada ocorrência da entidade **Homem** poderá se relacionar com no mínimo nenhuma e com no máximo uma ocorrência da entidade **Mulher**;

- Cada ocorrência da entidade **Mulher** poderá se relacionar com no mínimo nenhuma e com no máximo uma ocorrência da entidade **Homem**.

Em outras palavras, um homem pode não estar em nenhum relacionamento — cardinalidade 0 — ou pode se relacionar com no máximo uma mulher — cardinalidade 1.

Relacionamento opcional e relacionamento obrigatório:

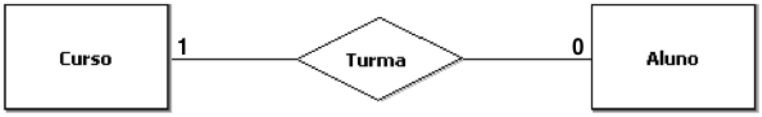


Figura 14 - Obrigatoriedade do relacionamento

Na Figura 14 temos um ótimo exemplo de relacionamento obrigatório e opcional. Ao lado da entidade **Curso** temos o valor 1; ele indica que todas as ocorrências da entidade **Aluno** devem obrigatoriamente se relacionar com alguma ocorrência da entidade **Curso**. Já ao lado da entidade **Aluno** temos o valor 0, ou seja, não é obrigatório que as ocorrências da entidade **Curso** se relacionem com as ocorrências da entidade **Aluno**.

2.5 Atributos

Os atributos são os elementos ou propriedades que diferenciam cada ocorrência dentro de uma entidade.

Por exemplo:

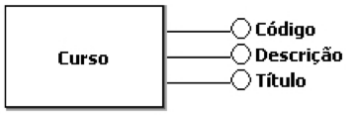


Figura 15 – Entidade e seus respectivos atributos

Outro exemplo:

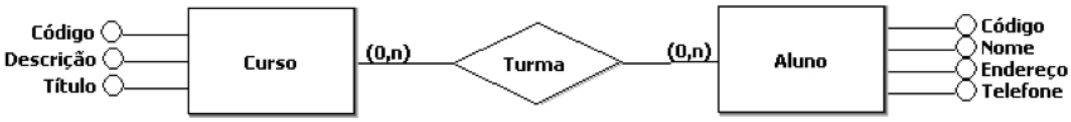


Figura 16 – Entidades, relacionamentos e atributos

Na Figura 16, tanto a entidade **Curso** quanto a **Aluno** possuem cada qual seu atributo Código, ou seja, ele é um atributo da entidade para a qual foi definido.

2.6 Identificador (chave primária e estrangeira)

Toda entidade deve possuir um atributo com valores únicos, ou seja, um identificador que diferencia uma ocorrência das demais dentro da entidade. Isso é necessário pois, ao efetuar alguma ação naquela ocorrência, o SGBD deve ser capaz de poder identificá-la como única. O termo técnico

utilizado para se referir a esse identificador se chama chave primária. Por exemplo:

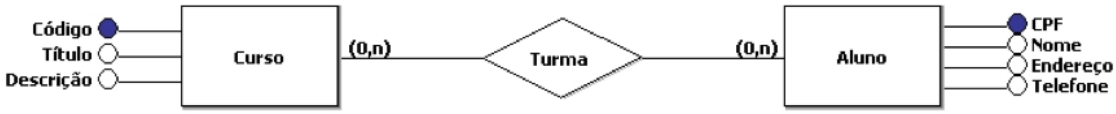


Figura 17 – Identificador

Visualmente conseguimos identificar que na Figura 17 o identificador da entidade **Curso** é o atributo Código; já na entidade **Aluno** o identificador é o atributo CPF. Esses dois identificadores representam valores que jamais irão se repetir; eles têm a missão de tornar cada ocorrência única.

Existem casos nos quais um único identificador não é capaz de tornar as ocorrências únicas; nesses casos é necessário utilizar identificadores compostos:

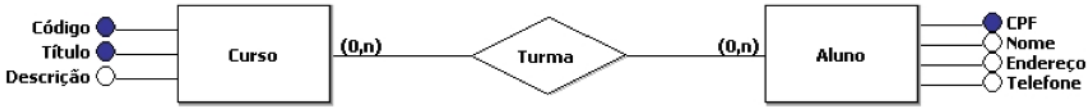


Figura 18 - Identificador composto

Na Figura 18, para a entidade **Curso**, temos dois identificadores (Código e Título), ou seja, essa entidade possui identificadores compostos. Separadamente esses atributos não serão capazes de garantir a unicidade das ocorrências, porém juntos conseguirão cumprir muito bem essa missão.

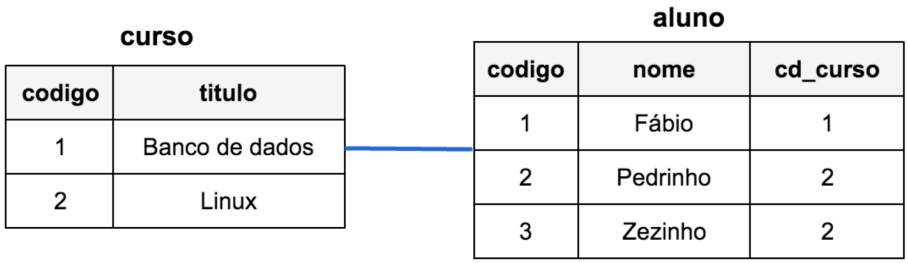
Quando a chave primária for composta por um único identificador ela será considerada uma chave simples; já quando existirem dois ou mais identificadores formando uma chave primária ela será considerada uma chave composta. Toda chave primária é obrigatória, ou seja, seu valor não pode ser nulo (vazio).

Existe ainda um conceito muito importante conhecido como chave estrangeira, que nada mais é do que uma chave primária em outra entidade com a qual a entidade atual mantém relacionamento. Sua utilização tem como resultado inúmeros benefícios; iremos utilizar a Figura 18 como objeto de nossa análise e elemento facilitador para transmitir esse conceito e, sendo assim, vamos aos benefícios gerados pela utilização de chave estrangeira:

- Consistência dos dados. Por exemplo: se você tentar cadastrar um aluno no curso X e o curso não existir, o banco irá retornar uma mensagem de erro.
- Se tentar excluir um curso que está relacionado a algum aluno, o banco irá retornar uma mensagem de erro.
- Ao tentar alterar uma chave primária, ocorrerá erro caso exista algum relacionamento ativo; isso garante a integridade e consistência do relacionamento.

Ao escolher uma chave estrangeira (chave de relacionamento), escolha sempre a chave mais simples. No exemplo da Figura 18, seria o atributo **Código**. Isso garante uma maior performance.

Assim como prometido em tópicos anteriores, irei realizar uma demonstração de relacionamento utilizando somente duas tabelas em vez de três:



curso

codigo	titulo
1	Banco de dados
2	Linux

aluno

codigo	nome	cd_curso
1	Fábio	1
2	Pedrinho	2
3	Zezinho	2

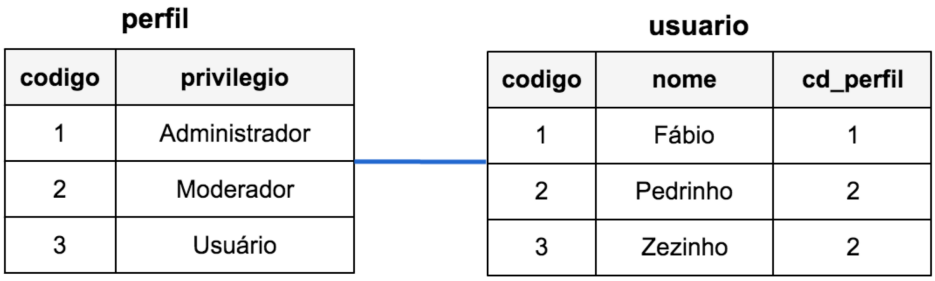
Figura 19 - Relacionamento entre duas entidades

Na Figura 19, a entidade **aluno** mantém um relacionamento com a entidade **curso** por meio da coluna cd_curso.

Em um cenário em que a cardinalidade fosse “0,1” não existiria problema em utilizar a proposta sugerida na Figura 19, porém, a partir do momento que podemos ter múltiplos relacionamentos para uma mesma ocorrência, a ideia proposta na imagem acima passa a ser desencorajada.

Uma das grandes vantagens da utilização do banco de dados é extinguir a duplicidade de registros. Se na imagem acima tivermos um relacionamento do tipo “1,n”, iremos sofrer com um sério problema de duplicidade de registros. Pense bem: sempre que um aluno tiver interesse em fazer dois ou mais cursos seremos forçados a duplicar o seu nome para poder promover um novo relacionamento, sem falar, é claro, que o nome pode ser escrito ou abreviado de outra forma, resultando em mais um problema que seria justamente a inconsistência das informações.

Mas calma aí! Esse tipo de relacionamento também pode ser muito útil quanto utilizado da forma correta e em cenários propícios. No exemplo abaixo ele se encaixa perfeitamente:



perfil

codigo	privilegio
1	Administrador
2	Moderador
3	Usuário

usuario

codigo	nome	cd_perfil
1	Fábio	1
2	Pedrinho	2
3	Zezinho	2

Figura 20 - Relacionamento entre duas entidades

Na regra de negócio acima, cada ocorrência da entidade **usuario** irá se relacionar obrigatoriamente com somente uma ocorrência da entidade **perfil**; sendo assim a proposta apresentada na Figura 20 atende a todas as expectativas para o cenário apresentado na Figura 20.

Tenha em mente que, quanto menos entidades envolvidas em “um” relacionamento, menor será o custo computacional para realizar as interações necessárias para lidar com os dados; sendo assim, a estruturação e elaboração correta do banco pode ser um diferencial valioso. Sempre que possível, evite criar entidades de relacionamento desnecessárias. Em geral elas são inúteis quando a cardinalidade máxima do relacionamento é 1.

Lembre-se do que foi dito anteriormente: tudo depende da sua regra de negócio. Em geral, uma entidade de relacionamento é utilizada quando um mesmo registro precisa se relacionar com N ocorrências de outra entidade. Veja este exemplo:

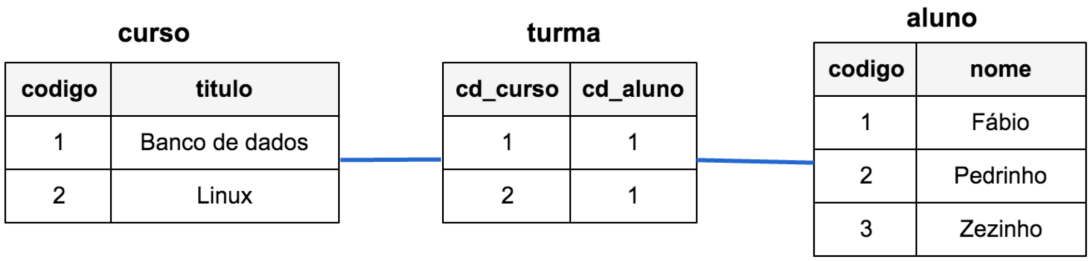


Figura 21 - Relacionamento via entidade de relacionamento

A Figura 21 nos permite ter um melhor entendimento do que foi explicado: veja que o aluno Fábio se relaciona com duas ocorrências diferentes que estão presentes na entidade **curso**. Como os relacionamentos não estão mais concentrados na entidade **aluno**, temos uma maior flexibilidade para relacionar quantas ocorrências forem necessárias sem precisar duplicar registros.

No exemplo acima os atributos codigo das entidades **curso** e **aluno**, bem como cd_curso e cd_aluno da entidade **turma** são chaves primárias. Como sabemos disso? O relacionamento se dá por meio desses atributos.

Quando temos chaves compostas em uma entidade, devemos eleger uma chave para ser utilizada como chave estrangeira (chave de relacionamento). Uma boa prática é utilizar a chave mais simples possível como objeto de relacionamento.

*Os dois atributos da entidade **turma** apontam para chaves estrangeiras, logo é normal que esses dois atributos sejam configurados como chaves primárias compostas, ou seja, esses dois atributos juntos irão garantir a unicidade das ocorrências dentro de sua entidade.*

2.7 Entidade genérica e especializada

A generalização ou especialização de uma entidade é um conceito e também uma técnica valiosa. Veja uma demonstração prática:

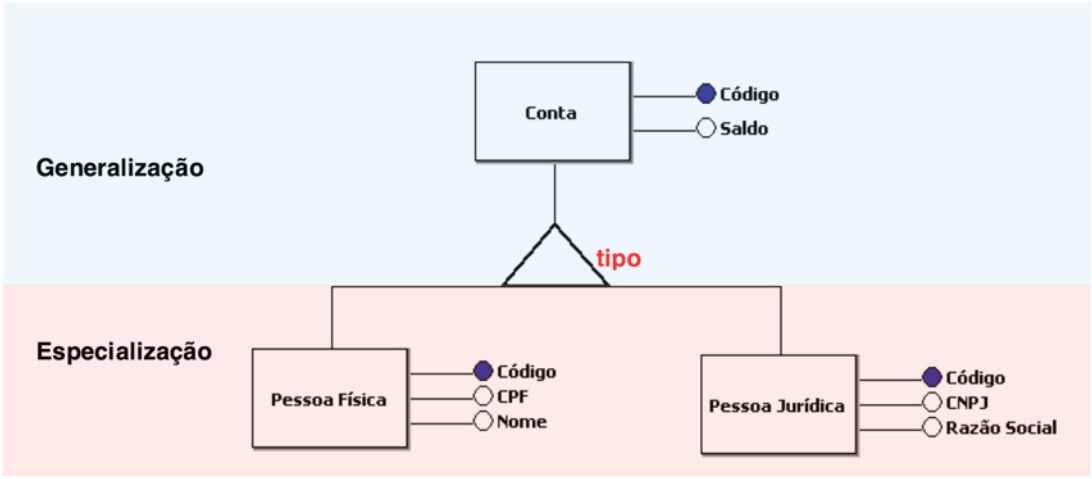


Figura 22 - Especialização e generalização

Para simplificar o entendimento, vamos inserir a Figura 22 em um contexto fictício:

A empresa fictícia TI na Rede possui uma base de dados de cadastro de clientes. Cada um de seus inestimáveis clientes possui uma **Conta** nessa empresa, e eles são divididos em dois grupos: **Pessoa Física** e **Pessoa Jurídica**. Além de compartilhar alguns atributos, cada grupo pode possuir dezenas de outros atributos próprios – isso é o que justifica a especialização.

Analisando o texto acima conseguimos identificar que a entidade **Conta** é uma entidade *genérica*, enquanto as entidades **Pessoa Física** e **Pessoa Jurídica** são entidades *especializadas*. O que isso significa? Que a entidade **Pessoa Física** é uma **Conta**, porém é uma **Conta** muito específica: além de ter atributos herdados como é o caso do atributo Saldo, possui atributos próprios como CPF e Nome. O mesmo entendimento se aplica à entidade **Pessoa Jurídica**.

Esse é um conceito que por muitas vezes pode passar batido, porém, se prestarmos atenção e conseguirmos de forma eficiente definir nosso modelo de dados, de forma a implementar ao pé da letra o conceito de generalização e especialização, teremos como resultado um banco de dados eficiente, fácil de ser compreendido e passível de algum tipo de implementação posterior sem dores de cabeça.

Considerando a Figura 22, no lugar da palavra tipo — destacada de vermelho — poderemos ter a representação de uma das seguintes letras: **C**, **P**, **T** e **X,** respectivamente representando as especializações do tipo **Compartilhada**, **Parcial**, **Total** ou **Exclusiva**.

- **Compartilhada**: Indica que uma ocorrência da entidade genérica poderá se relacionar ao mesmo tempo com ocorrências de várias entidades especializadas.

- **Parcial**: Indica que uma ocorrência da entidade genérica nem sempre irá se relacionar com ocorrências das entidades especializadas, ou seja, as ocorrências da entidade genérica poderão opcionalmente se relacionar com alguma entidade especializada.

- **Total**: Indica que nenhum registro poderá ser uma simples entidade genérica, ou seja, cada ocorrência da entidade genérica precisa obrigatoriamente se relacionar com alguma entidade especializada.

- **Exclusiva**: Indica que as ocorrências da entidade genérica irão se relacionar exclusivamente com no máximo uma entidade especializada.

Obviamente, essas especializações serão utilizadas conforme a necessidade de cada caso.

Para o cenário da Figura 22, você saberia informar qual tipo de especialização melhor se encaixa?

Se você escolheu a especialização *exclusiva,* parabenizo você por sua opção!

Vamos analisar cada especialização no contexto da Figura 22 para entender o motivo pelo qual a especialização *exclusiva* é a mais adequada para esse caso:

- Especialização compartilhada: Uma conta poderá ser ao mesmo tempo uma conta pessoa física e jurídica.

- Especialização parcial: Uma conta poderá ser somente uma conta, ou opcionalmente poderá ser uma conta pessoa física ou jurídica.

- Especialização total: Uma conta precisa obrigatoriamente ser uma conta pessoa física e jurídica.

- Especialização exclusiva: Uma conta precisa ser exclusivamente uma conta pessoa física ou jurídica. Para o meu cenário, essa é a especialização perfeita.

Se não prestarmos atenção ao cenário, fica muito fácil cair em erro ao tentar determinar uma especialização; os conceitos se aproximam bastante e chegam a ser uma “pegadinha” para mentes desatentas. Se ficou alguma dúvida, releia este tópico.

Mas calma! Se mesmo após uma releitura tudo continuar obscuro, não desanime. Com o tempo, o conceito fica mais claro e a prática leva quase à perfeição.

2.8 DER, demonstração prática

Até agora vimos muita teoria. Chegou a hora de uma demonstração prática do que foi ensinado até aqui, porém acredito que seja relevante citar que um DER pode ser desenhado de diversas formas diferentes e que todas podem estar corretas.

Além disso, os símbolos e níveis de detalhes podem variar um pouco dependendo do software utilizado e do capricho de quem elaborou o diagrama.

Veja um exemplo prático:

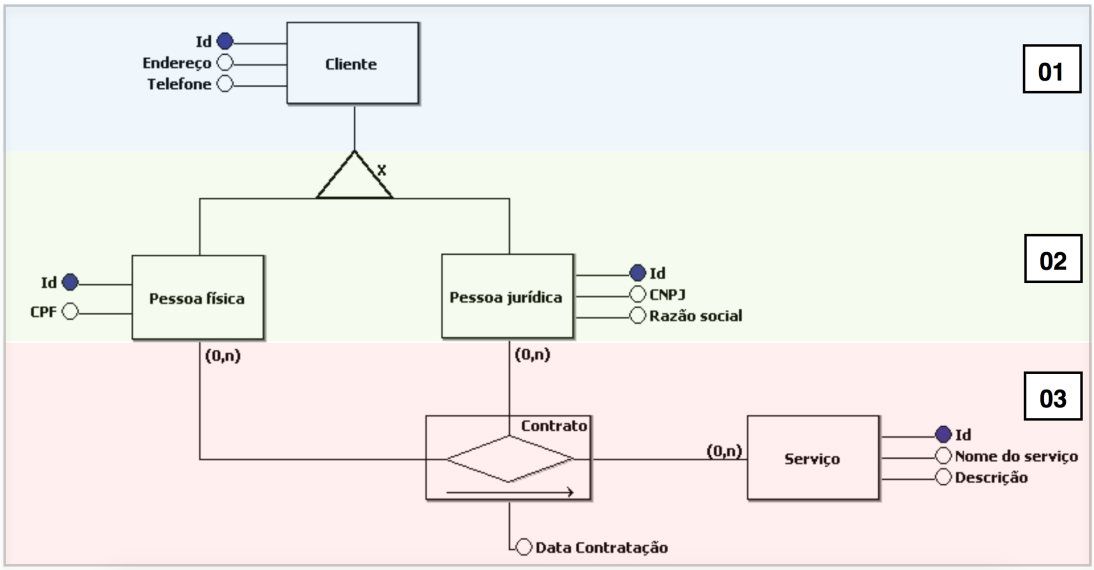


Figura 23 - Demonstração prática de generalização e especialização

Qual a historinha desse DER? A empresa TI na Rede precisa de um banco de dados no qual os clientes possam ser cadastrados para que serviços possam ser ofertados a seus futuros consumidores.

O DER foi dividido em blocos coloridos e numerados de 01 a 03 para que seja possível conduzir uma explicação mais dinâmica. Então vamos analisar cada bloco separadamente:

- **Bloco 01**: Temos uma entidade genérica chamada **Cliente** e três atributos genéricos.

- **Bloco 02**: Temos a especialização exclusiva representada pela letra **X**, ou seja, toda e qualquer ocorrência da entidade **Cliente** deve obrigatoriamente se relacionar com ocorrências da entidade **Pessoa física** ou **Pessoa jurídica**. Isso significa que um **Cliente** será sempre uma **Pessoa física** ou uma **Pessoa jurídica**; é impossível ser os dois ao mesmo tempo ou não ser nenhum dos dois.

- **Bloco 03**: Temos uma entidade de relacionamento chamada **Contrato**. Você lembra-se que foi mencionado que um DER pode ter um nível de detalhe maior ou menor dependendo de quem o desenhou? Pois é, nesse DER foi inserida uma seta dentro da entidade **Contrato** apontando para a direita. Qual minha intenção com isso? Deixar explícito que o relacionamento se dá em direção à entidade **Serviço**, ou seja, quando o DER for lido teremos o seguinte entendimento: uma **Pessoa física** ou **Pessoa jurídica** contrata um **Serviço**. Buscamos impedir, assim, a interpretação de que um **Serviço** pode contratar uma **Pessoa física**.

Acredito que os identificadores (chave primária) Id não passaram despercebidos aos olhos atentos de meus queridos leitores. Pois é, esse atributo chamado Id geralmente é utilizado quando queremos criar um código único e sequencial para representar as ocorrências dentro da entidade. Lembra que uma chave primária é um valor que não se repete e que serve para identificar e diferenciar uma ocorrência dentro de uma entidade? Em vez de ficar dando nomes diferentes ao identificador de cada nova entidade, o chamamos genericamente de Id — abreviação de identificador.

No bloco 03 temos as cardinalidades. Uma **Pessoa física** ou **Pessoa jurídica** pode estar relacionada a nenhuma ou a várias ocorrências da entidade **Serviço**; já as ocorrências da entidade **Serviço** podem se relacionar com nenhuma ou com várias ocorrências das entidades **Pessoa física** ou **Pessoa jurídica**. Como foi decidida essa cardinalidade? Bem, podemos ter em nosso banco de dados um serviço que ainda não foi contratado por ninguém, então se ele pode existir mesmo que não tenha relacionamento sua cardinalidade deve ser opcional, ou seja, igual a 0 (zero); uma **Pessoa física** ou **Pessoa jurídica** também pode existir sem que nenhum serviço tenha sido relacionado a ela e, sendo assim, sua cardinalidade mínima também pode ser opcional. Simples, não é!?

Abaixo represento de uma outra forma o mesmo DER da Figura 23:

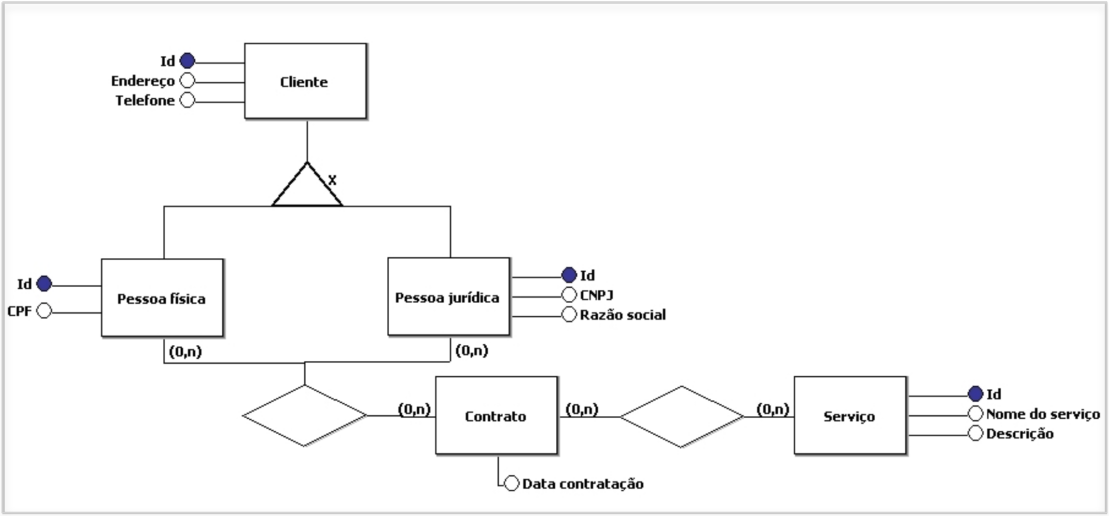


Figura 24 - Outro exemplo de generalização e especialização

A mesma coisa, concorda? A única diferença é que agora gastei mais tinta para representar o mesmo DER da Figura 23, com níveis de detalhes e de semântica visual um pouco menores; ainda assim, esses dois DER são equivalentes em termos de significado.

2.9 Julgando um DER

Como saber se um DER está correto? Devemos julgar o formato dos símbolos? Julgar a forma como se relacionam? Julgar a riqueza de detalhes?

Eita, calma! Bem, não temos como julgar um DER pelos seus símbolos pois isso pode variar bastante. Nesse tipo de diagrama é possível julgar a necessidade de determinada entidade, relacionamento e atributos com base na regra de negócio. Já a riqueza de detalhes é algo a se pensar; às vezes uma nível muito alto de detalhes pode atrapalhar mais do que ajudar.

Coloque no DER somente o que for necessário para se fazer entender a lógica ali desenhada. Esse diagrama deve conter as informações necessárias para que, ao ser lido por alguém que entenda um DER, a pessoa seja capaz de compreender a ideia ali transmitida. Além do DER existem muitas outras formas de modelar um banco – existe a modelagem textual, existe o EER e muitos outros – , porém o DER é uma das formas de representação mais simples que existe e foi justamente este o motivo pelo qual o escolhi para ser apresentado aqui nesta obra.

2.10 Resumo

Neste capítulo foram passados alguns conceitos introdutórios que podem ser considerados a base para toda e qualquer interação com bancos do tipo modelo entidade relacional; por isso, este passa a ser um capítulo de leitura obrigatória.

Os conceitos de entidade, atributos, relacionamentos e DER serão altamente utilizados sempre que formos trabalhar com bancos relacionais. Diante dessa afirmação este pode ser considerado um dos capítulos de maior importância nesta obra.

A representação gráfica do modelo DER pode variar dependendo do software utilizado, porém os conceitos e a aplicação de regras são invariantes, ou seja, não se apegue às formas gráficas, mas sim às regras e boas práticas que ditam como um DER deve ser montado.

Capítulo 3

Linguagem de consulta estruturada (SQL)

"Experiência é apenas o nome que damos aos nossos erros."
— Oscar Wilde

A linguagem SQL foi desenvolvida no início dos anos 70 dentro dos laboratórios da IBM como uma forma de mostrar a viabilidade de um modelo relacional para persistência de dados. Essa é uma linguagem declarativa, muito simples e de fácil compreensão, fatores que a tornaram a principal linguagem para manipulação de dados persistidos em bancos relacionais.

3.1 Padronização do SQL

Embora a linguagem SQL tenha sido inicialmente desenvolvida pela IBM, rapidamente outras organizações passaram a adotar padrões próprios para manipulação de dados. Daí começou o problema: os profissionais do mercado passaram a ter muita dor de cabeça, pois cada solução implementava um padrão próprio e único de SQL e, sendo assim, as operações básicas que funcionavam em um banco de dados não funcionavam em outro…

Qual a solução para esse problema? A criação e implantação de padrões sólidos. Essa missão foi realizada em 1986 pelo Instituto Nacional Americano de Padrões (ANSI) e em 1987 pela Organização Internacional para Padronização (ISO). Depois disso a linguagem recebeu outras atualizações, porém essa informação é irrelevante para o estudo desenvolvido nesta obra.

Devido à padronização do SQL, operações básicas construídas nessa linguagem passaram a possuir uma compatibilidade de praticamente 100% perante bancos de dados diferentes, mas que implementam o padrão SQL. Para aqueles que trabalham no ramo essa foi uma grande vitória.

Consultar, alterar, incluir ou deletar dados em um banco são consideradas operações triviais, mas antes de 1986/1987 essa era uma tarefa complicada, pois cada solução tinha seu padrão próprio de lidar com os dados. Atualmente essa é uma tarefa simples e comum entre as diversas soluções de banco de dados que implementam o padrão SQL, ou seja, a escrita das operações básicas é exatamente idêntica entre as diferentes soluções de banco de dados presentes no mercado atual.

3.2 Subconjuntos da linguagem SQL

A linguagem SQL é dividida em 5 subconjuntos; eles representam o grupo ou conjunto de ações que desejamos executar sobre um banco de dados.

3.2.1 DML - Linguagem de manipulação de dados

Seguem abaixo as ações e os comandos que fazem parte deste subconjunto:

Ação	Comando	Descrição
Alteração	UPDATE	Utilizado para alterar registros
Consulta	SELECT	Utilizado para consultar registros
Exclusão	DELETE	Utilizado para excluir registros
Inserção	INSERT	Utilizado para inserir registros

Esse é o subconjunto mais utilizado no dia a dia, o que se deve ao fato de ele lidar diretamente com a manipulação dos dados.

3.2.2 DDL - Linguagem de definição de dados

Seguem abaixo as ações e os comandos básicos que fazem parte deste subconjunto:

Ação	Comando	Descrição
Alteração	ALTER	Permite alterar um objeto. Exemplo: incluir ou alterar colunas.
Criação	CREATE	Permite criar um objeto. Exemplo: bancos, tabelas, visões e outros.
Exclusão	DROP	Permite apagar um objeto. Exemplo: bancos, tabela, visões e outros.

3.2.3 DCL - Linguagem de controle de dados

Seguem abaixo as ações e os comandos que fazem parte deste subconjunto:

Ação	Comando	Descrição
Autorizar	GRANT	Concede privilégios ao usuário.
Desautorizar	REVOKE	Remove privilégios do usuário.

3.2.4 DTL - Linguagem de transação de dados

Seguem abaixo as ações e os comandos que fazem parte deste subconjunto:

Ação	Comando	Descrição
Transação	BEGIN	Marca o início de uma transação.
Aplicar	COMMIT	Encerra a transação salvando todas as operações.

Desfazer	ROLLBACK	Encerra a transação desfazendo todas as operações.

3.2.5 DQL - Linguagem de consulta de dados

Embora este subconjunto seja composto por somente um comando, este é sem dúvida o comando mais utilizado dentro de um banco de dados:

Ação	Comando	Descrição
Consultar	SELECT	Utilizado para consultar dados

Obs: O comando SELECT faz parte do subconjunto DML e DQL.

3.3 Cláusulas

As cláusulas são condições utilizadas para determinar quais registros se deseja selecionar, excluir ou alterar.

Existe um número impressionante de cláusulas disponíveis para uso, porém, neste primeiro momento, irei listar aquelas com as quais tenho certeza que você terá maior contato no dia a dia:

- **FROM**: Utilizado para especificar a tabela ou as tabelas sobre as quais se deseja realizar alguma interação.
- **WHERE**: Utilizado para especificar a condição a ser considerada para selecionar os registros sobre os quais se deseja realizar alguma interação.
- **GROUP BY**: Utilizado para agrupar ou separar em grupos os registros selecionados.
- **ORDER BY**: Utilizado para ordenar os registros selecionados.
- **DISTINCT**: Utilizado para selecionar dados sem que haja repetição.
- **INNER JOIN**: Utilizado para retornar dados quando duas tabelas contiverem o mesmo valor informado na cláusula de busca.

3.4 Operadores

Em geral os operadores fazem parte da cláusula **WHERE** e são utilizados para montar expressões ou ligar condições.

3.4.1 Operadores lógicos

São operadores utilizados para fazer a junção entre condições/expressões. São eles:

- **AND**: Operador E. Avalia as condições e retorna o resultado caso ambas as condições sejam verdadeiras.
- **OR**: Operador OU. Avalia as condições e retorna o resultado caso ao menos uma seja verdadeira.

- **NOT**: Operador de negação. Retorna o valor contrário à condição.

3.4.2 Operadores relacionais

São operadores utilizados para fazer comparação entre valores. São eles:

Comando	Descrição
<	Menor que
>	Maior que
<=	Menor que ou igual a
>=	Maior que ou igual a
=	Igual a
<>	Diferente de

Além dos operadores em formato de símbolo temos os operadores reservados:

- **BETWEEN**: Utilizado para selecionar valores dentro de um intervalo determinado.
- **LIKE**: Utilizado para selecionar valores por comparação relativa à posição do seu conteúdo.
- **IN**: Permite especificar uma lista e buscar de uma só vez todos os valores informados na lista.

3.5 Funções de agregação

Utilizadas somente em conjunto com a instrução SELECT para retornar resultados agregados. As funções são:

Comando	Significado
AVG	Utilizado para calcular a média.
COUNT	Utilizado para retornar o total de registros da seleção,
MAX	Utilizado para retornar o maior valor do campo especificado.
MIN	Utilizado para retornar o menor valor do campo especificado.
SUM	Utilizado para devolver a soma para os campos informados.

3.6 Executando instruções SQL no banco

As instruções SQL que executamos em um banco de dados são comumente conhecidas como *query* ou *queries*.

Existem basicamente duas formas de se executar uma query no banco de dados; a primeira seria utilizar um software cliente instalado na sua máquina pessoal, e a segunda opção seria utilizar a console do SGBD.

Como já foi dito, este material é uma obra teórica e, diante disso, passarei somente alguns exemplos de *queries* para que você, leitor, saiba como montá-las. A sua utilização prática será introduzida em uma outra apostila, devidamente elaborada para essa finalidade mais prática.

3.7 Exemplo de *query* do subconjunto DML

Como este é um dos subconjuntos mais utilizados no dia a dia, nada mais justo do que apresentar alguns exemplos reais de uso.

Para simplificar o entendimento, irei utilizar a tabela abaixo como objeto alvo de nossas *queries*:

id	nome	perfil	idade
1	Fábio	1	26
2	Pedro	3	30
3	Maria	3	29

3.7.1 Consulta

Nessa *query* estamos realizando um **SELECT** (seleção) em toda a tabela usuario e retornando somente os registros nos quais a coluna nome for **igual** a Fábio:

- **SELECT * FROM** usuario **WHERE** nome = 'Fábio';

Nesta outra *query* realizamos uma consulta semelhante à anterior, porém dessa vez temos duas condições: só serão retornados registros nos quais a coluna perfil for **igual** a 1 e a idade maior que 18:

- **SELECT** nome **FROM** usuario **WHERE** perfil = '1' **AND** idade > 18;

Por fim, neste último exemplo realizamos uma contagem de todos os registros da tabela usuario; caso fosse de nosso interesse realizar a contagem do total de usuários de uma determinada cidade, bastaria incluir a cláusula **WHERE** e informar a cidade alvo de nossa contagem:

- **SELECT *COUNT(*)* FROM** usuario;

3.7.2 Alteração

Para alterar/atualizar um registro, basta utilizar o comando **UPDATE**, informar a tabela, a palavra reservada **SET**, a coluna ou as colunas a serem atualizadas com seus respectivos valores e

informar por último uma cláusula **WHERE**, que fica responsável por indicar quais registros serão atualizado(s):

- **UPDATE** usuario **SET** nome='Fábio Jânio', idade='27' **WHERE** nome='Fábio';

No exemplo acima estamos buscando o registro no qual o atributo nome é **igual** a Fábio, e atualizando a coluna nome com o valor Fábio Jânio e a coluna idade com o valor 27.

Obs: Só serão atualizados os registros que satisfaçam a condição da cláusula **WHERE**. Como temos três registros na tabela usuario e somente um satisfaz essa condição, somente ele será afetado por esse comando.

3.7.3 Inclusão

Essa *query* irá inserir um novo registro na tabela usuario:

- **INSERT INTO** usuario (nome, perfil, idade) **VALUES** ('Zezinho', 3, 65);

ou

- **INSERT INTO** usuario **VALUES** (null, 'Zezinho', 3, 65);

No primeiro exemplo devemos informar as colunas e seus respectivos valores na mesma ordem em que as colunas foram declaradas no INSERT.

No segundo exemplo a declaração das colunas é omitida da *query*, e por esse motivo somos obrigados a declarar o valor de todas as colunas na ordem em que foram definidas no ato da criação da tabela. Em nosso caso tivemos que informar os valores id, nome, perfil e idade, que representam respectivamente os valores null, Zezinho, 3 e 65. O valor **null** informado na coluna id é uma forma de indicar que para aquela coluna não estamos passando nenhum valor no momento do **INSERT**. Como a coluna id é um autoincremente o banco se encarregará de realizar seu preenchimento com um valor sequencial, ou seja, o valor do registro anterior + 1.

Recomendo que você utilize o modelo proposto na primeira *query*; ele é mais intuitivo.

3.7.4 Exclusão

Segue um exemplo de como excluir registros de uma tabela:

- **DELETE FROM** usuario **WHERE** nome = 'Fábio';

3.8 Atenção

Os comandos **commit** e **rollback** citados anteriormente no subconjunto dos DTL podem ser utilizados para aplicar ou desfazer as operações de inclusão (INSERT), alteração (UPDATE) e exclusão (DELETE).

Esses dois recursos são úteis? Vamos supor que você recebeu a missão de atualizar um registro **X**. Você vai lá no banco feliz da vida e aplica um UPDATE em outro registro; qual seria a solução? Executar o comando **rollback** para desfazer a besteira. Porém, se por acaso você executar um **commit**, aí já era! Sua alteração será persistida permanentemente até que um outro UPDATE seja executado para corrigir a falta de atenção.

Obs: Em geral o recurso de **autocommit** já vem ativo por *default*; sendo assim, ao executar qualquer comando **DML**, ele será automaticamente persistido no banco. Para resolver isso basta desativar o **autocommit**. Esse comando será diferente para cada SGBD diferente, por isso sugiro que pesquise esse comando na documentação do SGBD que você está utilizando.

*Recomendo extrema atenção ao utilizar os comandos **UPDATE** e **DELETE**. Sem a presença da cláusula **WHERE**, o efeito de sua execução irá afetar todos os registros da tabela.*

3.9 Resumo

Neste capítulo tivemos o que podemos chamar de experiência real de uso, ou seja, o leitor foi introduzido a uma visão real de como manipular dados dentro de um banco por meio dos comandos SQL, e de como criar, alterar e excluir bancos, entidades etc.

Todos os comandos apresentados até aqui são considerados simples e de uso cotidiano, porém é inegável que o subconjunto DML seja o mais frequentemente utilizado; diante disso oriento os meus queridos leitores a darem uma atenção maior a ele.

Com o uso frequente, muitos dos comandos apresentados acima serão automaticamente decorados, sendo assim não se preocupe caso já tenha esquecido alguns dos comandos listados nas páginas anteriores; com o uso e com o passar do tempo, eles irão se fixar na sua memória.

Mais importante do que decorar a escrita de um comando é saber o seu significado. Ao bater o olho em um SELECT, o leitor atento saberá que este é utilizado para buscar dados, e que é preciso informar em qual tabela os dados serão pesquisados e quais serão os parâmetros de busca.

Capítulo 4

Normalização de banco de dados

"Se eu enxerguei longe, foi por ter subido nos ombros de gigantes."
— Isaac Newton

A normalização é o processo pelo qual os dados são organizados no banco de tal modo que se possa protegê-los estabelecendo relacionamentos sólidos, eliminando redundâncias e valores inconsistentes.

As formas normais ditam um conjunto de regras e boas práticas a serem seguidas para que o banco seja o mais flexível e consistente possível. É importante ter em mente que as formas normais possuem ordem de precedência, ou seja, um banco só será considerado coberto pela **Segunda Forma Normal** se já estiver cumprindo os requisitos da **Primeira Forma Normal**; logo, só estará na **Terceira Forma Normal** se cumprir os requisitos da **Primeira e da Segunda Formas Normais,** e assim por diante.

Falaremos abaixo sobre as **Formas Normais 1, 2 e 3**. Existem outras, porém para este estudo inicial não se faz necessária sua menção.

4.1 Primeira Forma Normal (1FN)

Para um banco de dados estar de acordo com a 1FN, é necessário que não existam grupos de valores repetidos, ou seja, se todos os valores forem únicos (atômicos = indivisíveis) então estaremos a obedecer essa forma normal.

Veja um exemplo de tabela que quebra as definições da 1FN:

id	nome	telefone	endereco
1	Fábio	(21)XXXX-XXXX	Rua 6, Copacabana, Rio de Janeiro/RJ - 22080-040
2	Zezinho	(61)XXXX-XXXX (61)XXXX-XXXX	Quadra 102, Conjunto 8, Samambaia Sul, Brasília/DF - 72300-015
3	Maria	(11)XXXX-XXXX	Rua 47, Jardim São Paulo, São Paulo/SP - 08465-312

Figura 25 - Exemplo para análise

Analisando a tabela acima:

- Na coluna **telefone** temos uma lista de valores, ou seja, mais de um valor por registro.
- Na coluna **endereco** temos conjuntos de valores, tais como: logradouro, bairro, cidade, estado e CEP. Este conjunto de valores precisa ser separado em colunas.

Para nos adequar à **Primeira Forma Normal** devemos seguir estes passos:

a) Mover os valores que representam uma lista ou grupo para outra entidade. Em nosso exemplo, essa regra se aplica principalmente à coluna **telefone**
b) Separar conjuntos de valores em colunas de tal modo que os dados possuam “significado" atômico; em nosso exemplo essa regra se aplica principalmente a coluna **endereco**

Veja o exemplo anterior adequado à 1FN:

id	nome	logradouro	bairro	cidade	estado	cep
1	Fábio	Rua 6	Copacabana	Rio de Janeiro	RJ	22080-040
2	Zezinho	Quadra 102 Conjunto 8	Samambaia Sul	Brasília	DF	72300-015
3	Maria	Rua 47	Jardim São Paulo	São Paulo	SP	08465-312

id_usuario	telefone
2	(61)XXXX-XXXX
2	(61)XXXX-XXXX
1	(21)XXXX-XXXX
3	(11)XXXX-XXXX

Figura 26 - Tabelas de acordo com a 1FN

Para nos adequar a 1FN tivemos que realizar a separação do conjunto de dados **endereco** em atributos distintos; além disso criamos uma entidade adicional para comportar os números de telefone tendo em vista que alguns usuários podem ter mais de um telefone para contato.

Poderíamos ter estruturado nossa ideia utilizando 3 tabelas – uma para os usuários, outra para os endereços e por fim uma para os telefones –, porém minha regra de negócio sugeria que cada usuário poderia ter somente um endereço para correspondência e N números de telefone, e logo não vi necessidade em criar uma entidade para separar o endereço e promover um relacionamento desnecessário. Lembre-se de que relacionamentos desnecessários podem prejudicar a performance do banco.

Para o meu problema, a estrutura proposta acima atende a todas as minhas necessidades, porém, se por algum motivo fosse necessário permitir o cadastro de mais de um endereço por usuário, aí sim seria necessário criar uma terceira entidade chamada "endereco" e mover para ela as colunas pertinentes ao seu escopo.

4.2 Segunda Forma Normal (2FN)

Para estar de acordo com a **2FN** é necessário que a tabela esteja também na **1FN**, e que todos os atributos que não forem chave sejam dependentes de toda a chave primária e não somente de parte dela. Um dos principais benefícios desta regra de normalização é evitar a presença de valores redundantes.

A tabela (controle) abaixo não se enquadra na 2FN:

id	id_servico	servico	total_horas	valor_hora	subtotal
1	0001	Designer de Site	12	70,00	840,00
2	0002	Desenvolvimento de Software	50	90,00	4.500,00
3	0001	Consultoria	5	100,00	500,00

Figura 27 - Exemplo para análise

Analisando a tabela acima:

- A coluna **id_servico** faz referência às colunas **servico** e **valor_hora**, logo se imagina que em determinado momento os valores dessas duas colunas irão se repetir. Lembre-se que a duplicação de valores/dados é um dos elementos que quebram os conceitos da 2FN;
- As colunas **id e id_servico** são claramente chaves compostas, pois servem para identificar registros únicos, porém a coluna id faz referência somente às colunas total_horas e subtotal, enquanto **id_servico** faz referência às demais colunas. A 2FN diz que todos os atributos não chave devem depender de toda a chave e não somente de parte dela, sendo assim estamos quebrando essa regra;
- Outro problema no modelo acima é que, se em algum momento tivermos que alterar o nome de determinado serviço, será necessário aplicar *update* em N registros para alterar uma única informação.

Para nos adequar à **Segunda Forma Normal** devemos seguir estes passos:

a) Separar os atributos que possam gerar alguma redundância de valor, ou seja, mover as colunas **id_servico**, **servico** e **valor_hora** para outra entidade;
b) Estabelecer um relacionamento por meio de uma chave estrangeira.

Veja abaixo o resultado final (tabela de acordo com a Segunda Forma Normal):

id	id_servico	total_horas	subtotal
1	0003	5	500,00
2	0001	50	840,00
3	0002	12	4.500,00

id_servico	servico	valor_hora
0001	Designer de Site	70,00
0002	Desenvolvimento de Software	90,00
0003	Consultoria	100,00

Figura 28 - Tabela na 2FN

Ótimo, agora sim! Na primeira coluna temos o controle de horas e o custo total para determinado serviço. Quando um serviço for prestado, basta efetuar um INSERT na primeira tabela e criar uma referência para o serviço por meio da coluna **id_servico**. Se por acaso um dia resolvermos trocar o nome do serviço Designer de Site para Designer Web precisaremos efetuar um único *update* na

segunda tabela, que irá refletir em todas as outras, pois a referência se dá por meio do código, ou seja, por meio da coluna **id_servico,** e não pelo nome do serviço propriamente dito.

Agora podemos fazer a seguinte alusão: a primeira tabela se refere ao meu controle de horas trabalhadas, e a segunda tabela é o meu catalogo de serviços. Bom, não é? :)

4.3 Terceira Forma Normal (3FN)

Para estar de acordo com a 3FN é necessário que a tabela esteja cumprindo com a 1ª e a 2ª Formas Normais; além disso é necessário remover as colunas que possuam valores que podem ser obtidos a partir da comutação de valores de outras colunas da mesma tabela.

Vamos utilizar a mesma tabela do exemplo anterior:

id	id_servico	total_horas	valor_hora	subtotal
1	0003	5	100,00	500,00
2	0001	50	70,00	840,00
3	0002	12	90,00	4.500,00

Figura 29 - Tabela para análise

Analisando:

- A coluna **subtotal** representa o valor comutado que quebra a 3FN; ele representa a multiplicação da coluna **total_horas** por **valor_hora**.

Para nos adequar à **Terceira Forma Normal** devemos seguir estes passos:

a) Identificar todas as colunas (atributos) que são funcionalmente dependentes de outros atributos não chave;
b) Remover os atributos identificados.

id	id_servico	total_horas	valor_hora
1	0003	5	100,00
2	0001	50	70,00
3	0002	12	90,00

Figura 30 - Tabela na 3FN

Os valores que podem ser obtidos a partir da comutação de outros atributos não devem ser persistidos na tabela; para visualizá-los podemos utilizar *views* (conceito não discutido nesta obra) que efetuem um "processamento" dos dados da tabela e nos apresentem uma visão temporada desses valores.

4.4 Resumo

A normalização de banco de dados é um conceito importante e que possui um peso significativo para a consistência, flexibilidade e performance de uma base de dados.

As três formas normais apresentadas neste capítulo representam os pilares fundamentais da modelagem de um banco de dados. Falar que um banco foi corretamente modelado será considerado um tremendo engano caso este não esteja no mínimo de acordo com essas três formas normais.

As formas normais são padrões consolidados e, sendo assim, sua utilização é no mínimo recomendada e muito bem apreciada por gestores e profissionais sérios do segmento.

Conclusão

Ao desenvolver esta obra, não foi de meu interesse focar em uma ferramenta ou software específico, mas sim passar uma visão teórica e conceitual quanto à modelagem de banco de dados.

Com o volume cada vez maior de informações sendo armazenadas em meio digital, o profissional que segue à risca as regras e os conceitos que regem as boas práticas para se trabalhar com banco de dados está mais bem preparado para concorrer a uma boa vaga nesse mercado cada vez mais competitivo. Pensando nisso escrevi esta obra inicial que, com certeza, terá uma continuação focada nos principais SGBD do mercado.

A curiosidade, o interesse e o desejo incessante por compreender "tudo" à minha volta me levaram a estudar assuntos diversos ligados aos mais variados temas da tecnologia da informação. Com base na experiência adquirida e na convivência com profissionais do segmento estou buscando, sempre que possível, desenvolver obras como esta. Assim, espero que o leitor faça uma boa leitura e que este material seja de grande valia para o seu crescimento profissional e pessoal.

Esta obra chegou ao fim, mas deixo como orientação a seguinte mensagem: *não encerre por aqui os estudos iniciados por meio deste material. Procure outras obras, artigos e demais materiais para aprofundar o conhecimento já adquirido e se destacar perante os demais profissionais da área.*

Ficarei agradecido caso o leitor possa me enviar o seu feedback quanto a esta obra. Para isso deixo à disposição meus dados para contato na página seguinte.

Contatos

Fábio J L Ferreira

Em caso de dúvidas, sugestões, elogios, críticas ou demais assuntos,
favor entrar em contato por meio de uma das opções abaixo:

E-mail: *fabiojanio@tinarede.com.br*

Links do autor:

@_SDinfo

#blogfabiojanio

fabiojanio

blog

Caso os links acima não funcionem de forma adequado, utilize as ligações abaixo:

Twitter: http://www.twitter.com/_SDinfo

Facebook: http://www.facebook.com/blogfabiojanio

Linkedin: http://br.linkedin.com/in/fabiojanio

Blog: http://www.fabiojanio.com/