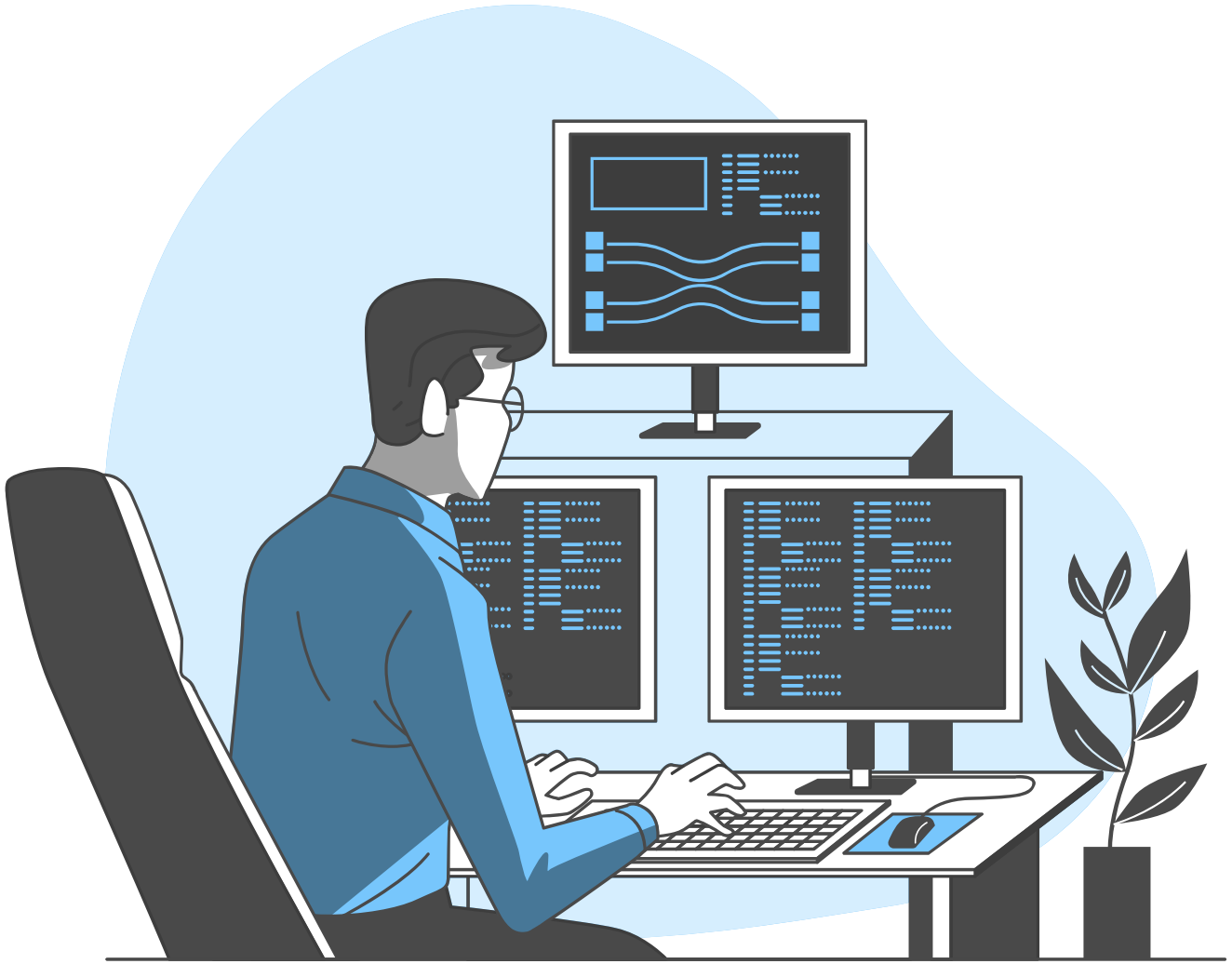
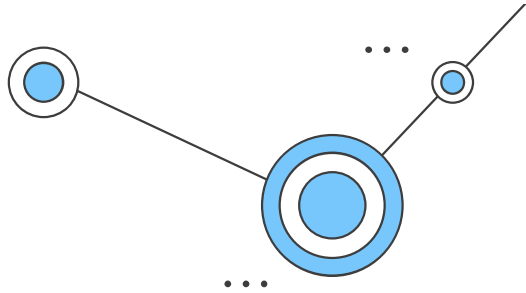


Modelagem

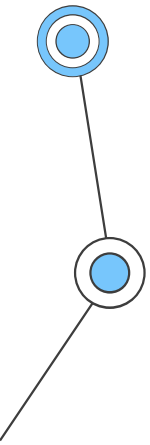
Banco de dados

01
Modelagem

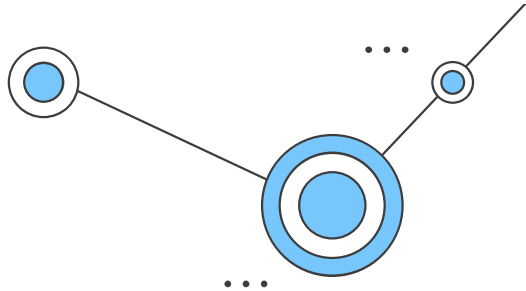
Como modelar dados?



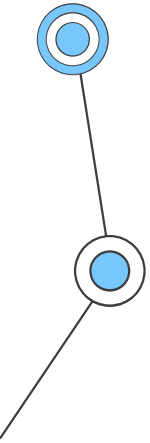
- É crucial para um desenvolvedor de aplicação ter uma noção dos fundamentos de modelagem de dados não apenas para ler os modelos de dados, mas também para trabalhar efetivamente com os DBAs responsáveis pelos aspectos relacionados aos dados do projeto.
- Fases da Modelagem:
  - Identificar os tipos de entidade;
  - Identificar atributos;
  - Aplicar convenção de nomes;
  - Identificar relacionamentos;
  - Associar chaves;
  - Normalizar para reduzir a redundância dos dados;
  - Diversificar para melhorar o desempenho.



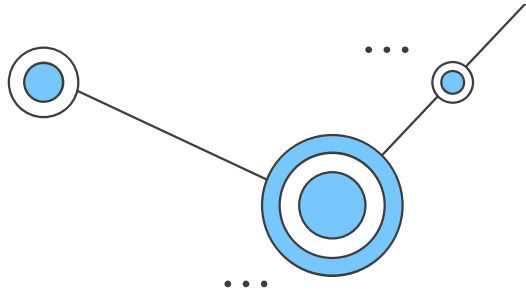
Identificar os tipos de entidade



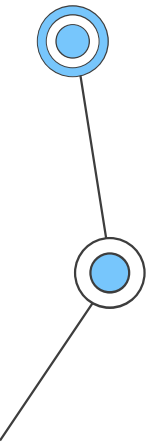
- Um tipo de entidade, ou simplesmente entidade, é conceitualmente similar ao conceito de orientação a objeto de uma classe – um tipo de entidade representa uma coleção de objetos similares. Um tipo de entidade pode representar uma coleção de pessoas, lugares, coisas, eventos ou conceitos. Exemplos de entidades em um sistema de vendas incluiria: Cliente, Endereço, Venda, Item e Taxa.
- Se estivéssemos modelando classes, esperaríamos descobrir classes exatamente com esses nomes. No entanto, a diferença entre uma classe e um tipo de entidade é que classes possuem dados e comportamentos, enquanto que tipos de entidade possuem apenas dados.



Identificar os tipos de entidade

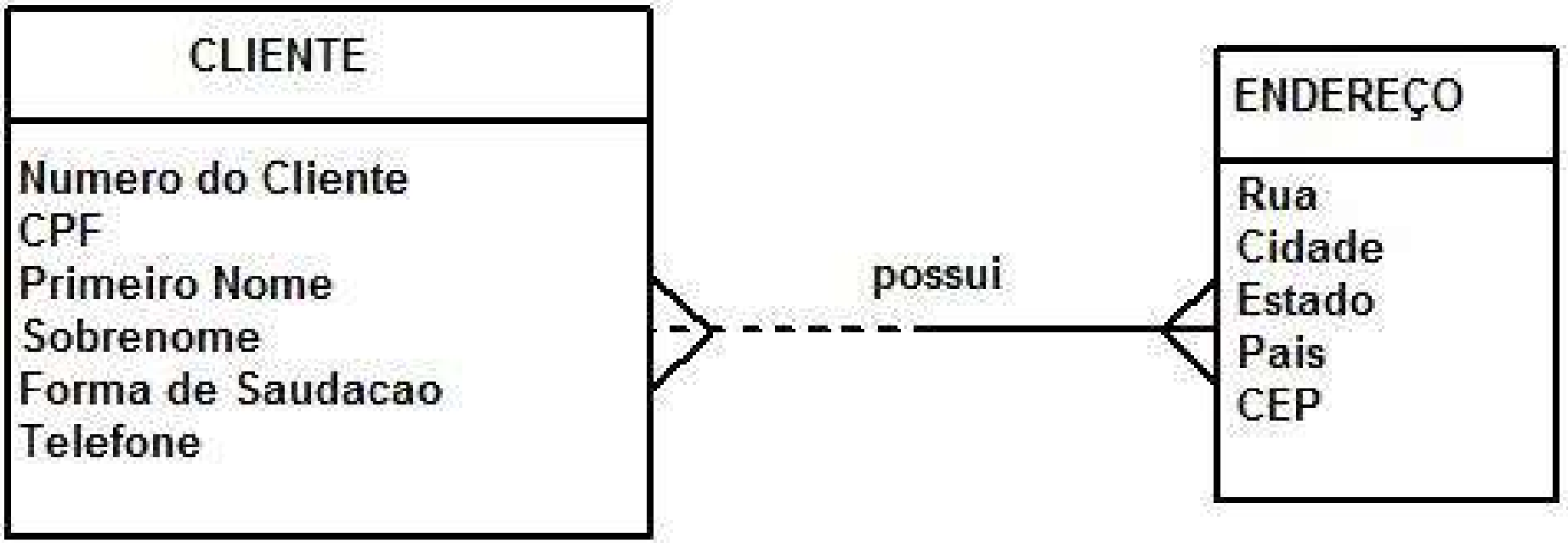


- Idealmente, uma entidade deveria ser normal, descrevendo de forma coesa uma informação do mundo real. Uma entidade normalmente descreve um conceito, tal como uma classe coesa modela um conceito. Por exemplo, cliente e venda são claramente dois conceitos diferentes, portanto, faz sentido modelá-los como entidades diferentes.



Exemplo Modelo Lógico de dados

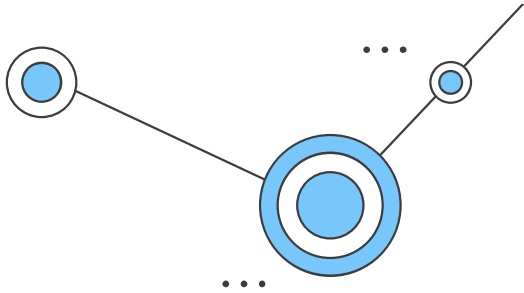
Figura 1



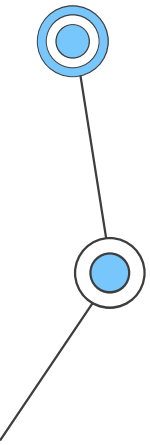
Exemplo Modelo Físico de dados

Figura 2

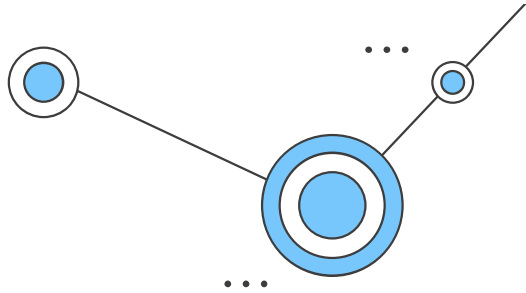
Identificar atributos



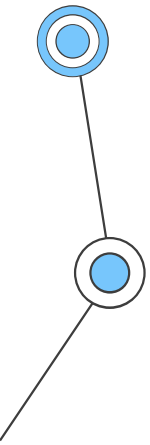
- Cada tipo de entidade terá um ou mais atributos de dados. Por exemplo, na **Figura 1** podemos ver que a entidade Cliente possui atributos como Primeiro Nome e Sobrenome e na **Figura 2** que a tabela TCLIENTE possui colunas de dados correspondentes CLI_PRIMEIRO_NOME e CLI_SOBRENOME (uma coluna é a implementação de um atributo de dados em um banco de dados relacional).



Identificar atributos



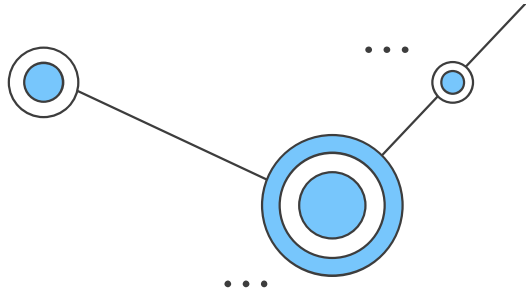
- Atributos devem ser coesos do ponto de vista do domínio da aplicação. Na Figura 1 decidimos que queríamos modelar o fato de pessoas possuírem primeiro nome e sobrenome em vez de apenas um nome (ex: “Cláudio” e “Dias” VS. “Cláudio Dias”). Usar o nível de detalhe correto pode ter um impacto significativo no esforço de desenvolvimento e manutenção. Refatorar uma simples coluna de dados em várias colunas pode ser difícil, o que pode resultar em construir o sistema com elementos desnecessários e, portanto, provoca um maior custo de desenvolvimento e de manutenção do que realmente necessário.



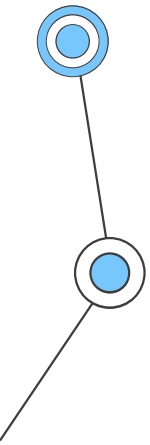
Aplicar convenções de nome

- Sua organização deve dispor de normas e diretrizes aplicáveis à modelagem de dados, algo que você deve ser capaz de obter dos administradores da empresa (se não existir você deve fazer algum lobby para incluí-lo).Essas diretrizes devem incluir as convenções de nomenclatura para a modelagem lógica e física, as convenções de nomenclatura lógica devem ser focadas na capacidade de leitura de humanos, enquanto as convenções de nomenclatura física refletirão considerações técnicas. Você pode ver claramente que diferentes convenções de nomenclatura foram aplicadas nas Figuras 1 e 2.

Aplicar convenções de nome



- A ideia básica é que desenvolvedores sigam um conjunto comum de padrões de modelagem em um projeto de software. Tal como é importante seguir convenções comuns de codificação, um código limpo que segue as diretrizes escolhidas é mais fácil de ser compreendido. Isso funciona da mesma forma para as convenções de modelagem de dados.



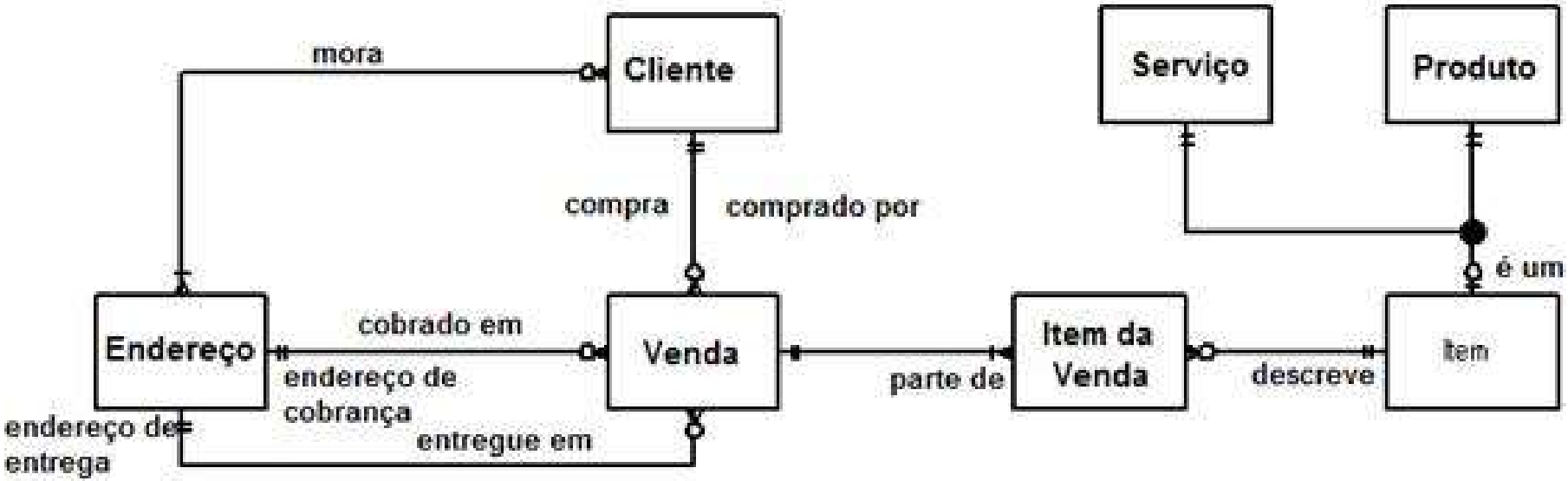
Identificar relacionamentos

- No mundo real, entidades possuem relacionamentos entre elas. Por exemplo, clientes FAZEM compras, clientes MORAM EM endereços e itens de venda SÃO PARTE DAS vendas. Todos esses termos em maiúsculo definem relacionamentos entre entidades. Os relacionamentos entre entidades são conceitualmente idênticos aos relacionamentos (associações) entre objetos.

- A Figura 4 descreve um MLD parcial para um sistema de compra online. A primeira coisa a se notar são os vários estilos aplicados aos nomes dos relacionamentos e papéis – diferentes relacionamentos requerem diferentes abordagens. Por exemplo, o relacionamento entre Cliente e Venda possui dois nomes, compra e é comprado por, mesmo o relacionamento entre essas entidades sendo apenas um. Neste exemplo, tendo um segundo nome no relacionamento, a ideia seria especificar como ler o relacionamento em cada direção. O ideal seria colocar apenas um nome por relacionamento.

Identificar relacionamentos

Figura 4

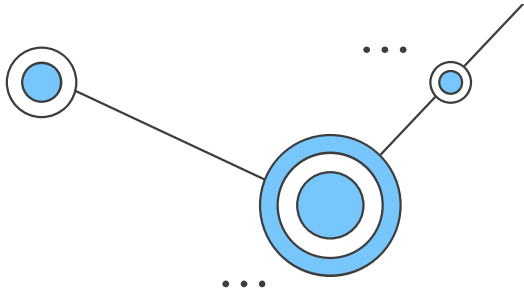


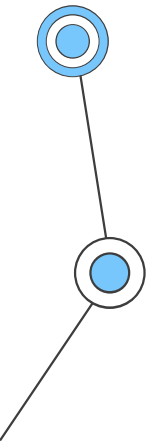
Identificar relacionamentos

- Precisamos também identificar a cardinalidade e opcionalmente de um relacionamento (a UML combina os conceitos de cardinalidade e opcionalmente de um relacionamento em um conceito único de multiplicidade). Cardinalidade representa o conceito de "quantos" enquanto opcionalmente representa o conceito de "se é obrigatória a existência da entidade". Por exemplo, não é suficiente saber que clientes fazem vendas. Quantas vendas um cliente pode realizar? Nenhuma, uma ou várias? Além disso, os relacionamentos existem nos dois sentidos: não apenas clientes fazem vendas, mas vendas são realizadas por clientes. Isso nos leva a questões como: quantos clientes podem ser envolvidos em uma dada venda e é possível ter uma venda com nenhum cliente envolvido?

- A **Figura 4** mostra que clientes fazem nenhuma ou mais vendas e que qualquer venda é realizada por um e somente um cliente. Ela também mostra que um cliente possui um ou mais endereços e que qualquer endereço possui zero ou mais clientes associados a ele.

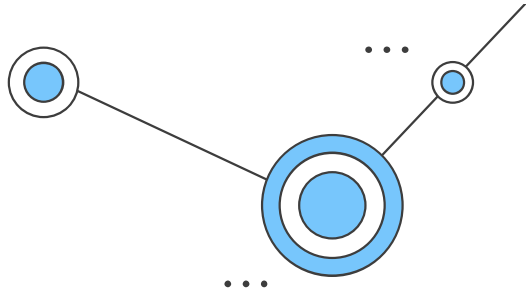
Identificar relacionamentos



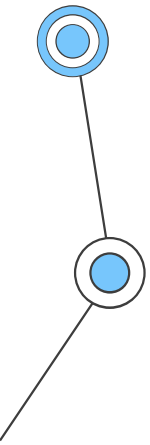


- Apesar de a UML distinguir entre diferentes tipos de relacionamentos – associações, hierarquia, agregação, composição e dependência – modeladores de dados normalmente não estão por dentro dessa questão. Subtipo, uma aplicação de hierarquia, é normalmente encontrada em modelos de dados. Agregação e composição são muito menos comum, assim como dependências, que são tipicamente uma construção de software e portanto não aparecem no modelo de dados, a menos que tenhamos um modelo físico de dados bastante detalhado que mostre como views, triggers ou stored procedures dependem de outros aspectos do esquema do banco de dados.

Associar chaves

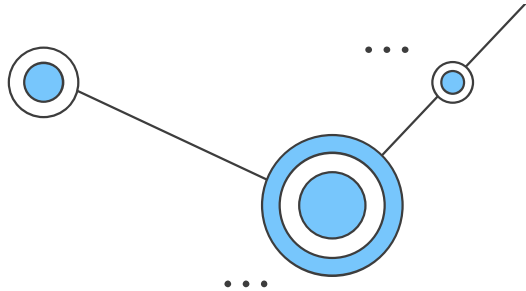


- Existem duas estratégias fundamentais para associar chaves às tabelas. Primeiro, podemos associar uma chave natural que é um ou mais atributos de dados existentes que são únicos para o conceito do negócio. Imaginemos uma tabela Cliente, por exemplo. Ela possui de imediato duas chaves candidatas, as colunas NumeroCliente e CPF. A segunda forma é introduzindo uma nova coluna, chamada chave substituta, que é uma chave que não possui qualquer significado para o negócio. Um exemplo disso seria uma coluna idEndereço de uma tabela Endereço. Endereços não possuem uma chave natural "trivial" porque seria necessário usar todas as colunas da tabela Endereço para formar uma chave. Assim, introduzir uma chave substituta é uma opção muito melhor neste caso.

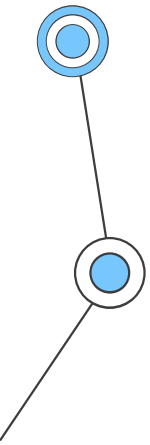


- O debate entre "natural vs. substituta" é um das grandes questões religiosas na comunidade de banco de dados. O fato é que não existe estratégia perfeita, e com o tempo percebemos que na prática algumas vezes fazem sentido usar chaves naturais e em outras situações é mais adequado o uso de chaves substitutas.

Normalizar para reduzir redundância de dados

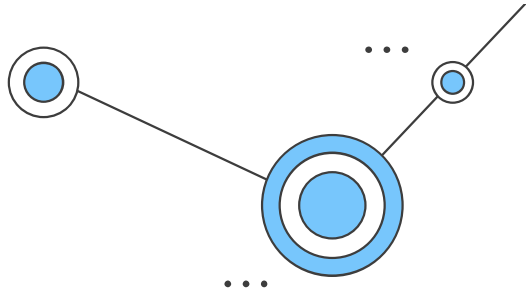


- Normalização de dados é um processo no qual atributos de dados em um modelos de dados são organizados para aumentar a coesão dos tipos de entidade. Em outras palavras, o objetivo da normalização de dados é reduzir e até eliminar redundância de dados, uma questão importante para desenvolvedores, pois é incrivelmente difícil armazenar objetos em um banco de dados relacional que mantém a mesma informação em vários lugares. A Tabela 1 resume as três principais regras de normalização descrevendo como aumentar os níveis de normalização em tipos de entidade.

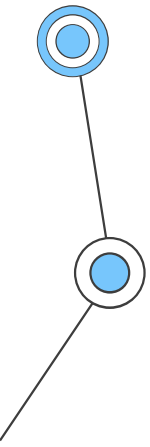


- Com respeito à terminologia, um esquema de dados é considerado estar em um nível de normalização do seu tipo de entidade menos normalizado. Por exemplo, se todos os tipos de entidade estão na segunda forma normal (2NF) ou maior, então dizemos que o esquema de dados está na 2NF.

Normalizar para reduzir redundância de dados



Nível	Regra
Primeira Forma Normal (1NF)	Uma entidade está na 1NF quando ela não contém grupos de dados repetidos.
Segunda Forma Normal (2NF)	Uma entidade está na 2NF quando ela está na 1NF e quando todos seus atributos que não são chaves primárias são completamente dependentes de sua chave primária.
Terceira Forma Normal (3NF)	Uma entidade está na 3NF quando ele está na 2NF e quando todos seus atributos são diretamente dependentes da chave primária.



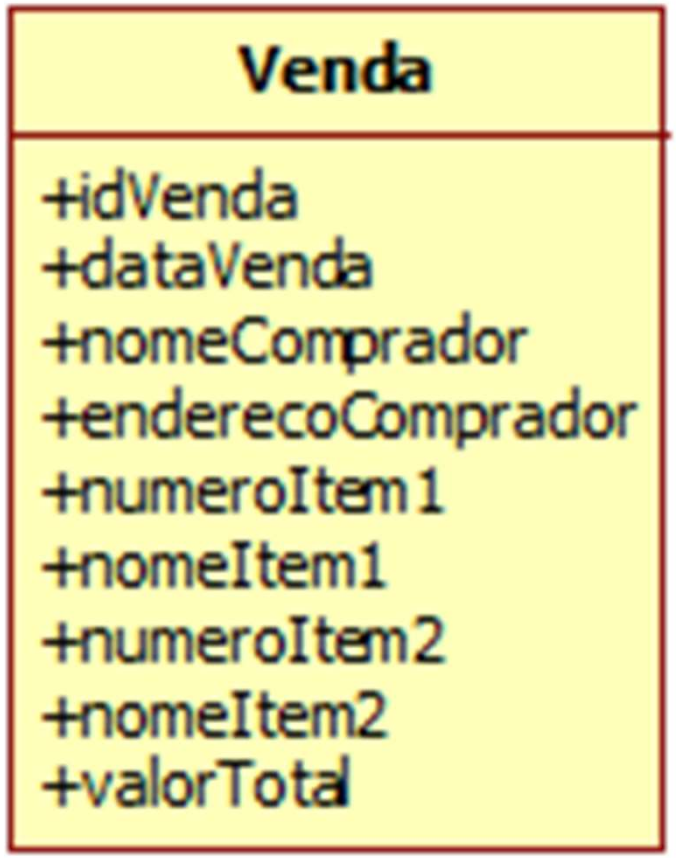
Normalizar para reduzir redundância de dados

- Por que normalização de dados? A vantagem de ter do esquema de dados altamente normalizado é que a informação é armazenada em um lugar apenas, reduzindo a possibilidade de dados inconsistentes. Além disso, esquemas de dados altamente normalizados em geral são conceitualmente mais próximos dos esquemas orientados a objeto, pois os objetivos da orientação a objetos de promover alta coesão e pouco acoplamento entre as classes resulta em soluções similares (ao menos do ponto de visa de dados). Isso geralmente torna mais simples mapear os objetos para o esquema de dados. Infelizmente, a normalização normalmente traz um custo para o desempenho. Com o esquema de dados da Figura 5 todos os dados para uma venda estão armazenados em uma linha (assumindo que vendas poderão ter até dois itens), simplificando o acesso.

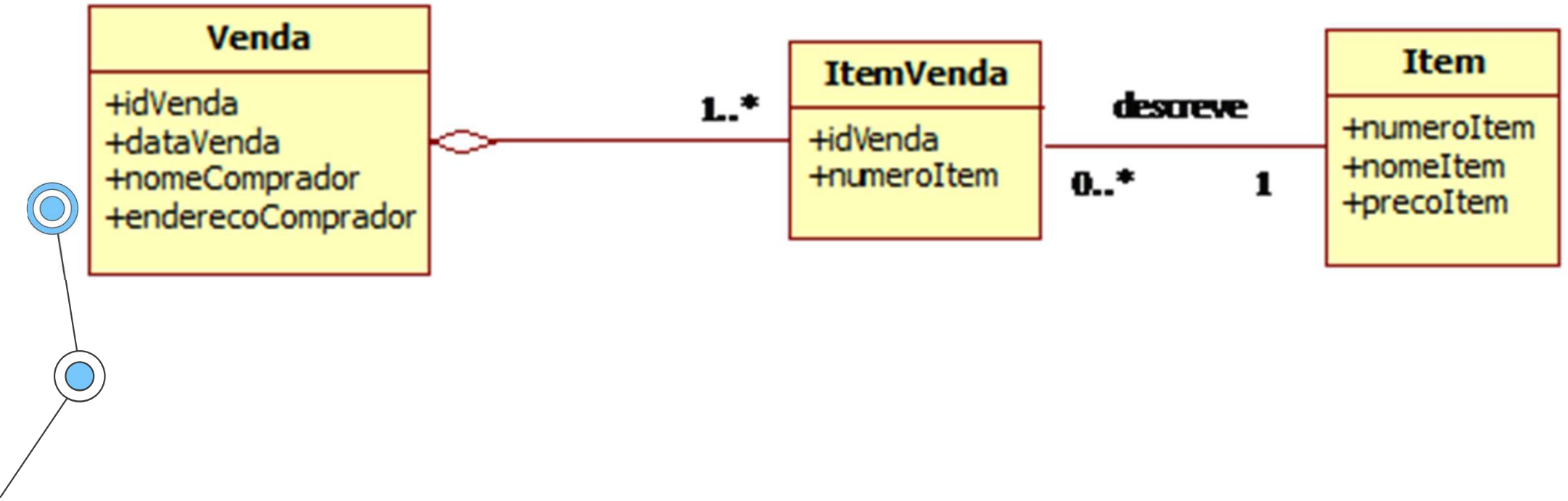
Normalizar para reduzir redundância de dados

- Com o esquema de dados da Figura 5 podemos rapidamente determinar a quantidade total de uma venda lendo uma única linha da tabela. Para fazer o mesmo com o esquema de dados da Figura 6, precisamos ler dados a partir de uma linha na tabela Venda, dados a partir de linhas na tabela ItemVenda para aquela venda e dados a partir das linhas correspondentes na tabela Item. Para esta consulta, o esquema de dados da Figura 5 provavelmente obtém melhor resultado.

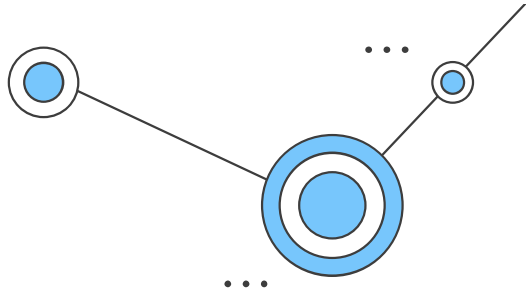
Normalizar para reduzir redundância de dados



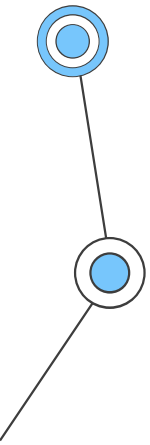
Normalizar para reduzir redundância de dados



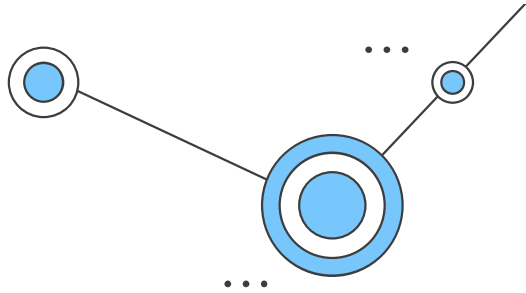
Entidades



- Os objetos ou partes envolvidas um domínio, também chamados de entidades, podem ser classificados como físicos ou lógicos, de acordo sua existência no mundo real. Entidades físicas: são aquelas realmente tangíveis, existentes e visíveis no mundo real, como um cliente (uma pessoa, uma empresa) ou um produto (um carro, um computador, uma roupa). Já as entidades lógicas são aquelas que existem geralmente em decorrência da interação entre ou com entidades físicas, que fazem sentido dentro de um certo domínio de negócios, mas que no mundo externo/real não são objetos físicos (que ocupam lugar no espaço). São exemplos disso uma venda ou uma classificação de um objeto (modelo, espécie, função de um usuário do sistema).
- As entidades são nomeadas com substantivos concretos ou abstratos que representem de forma clara sua função dentro do domínio. Exemplos práticos de entidades comuns em vários sistemas são Cliente, Produto, Venda, Turma, Função, entre outros.

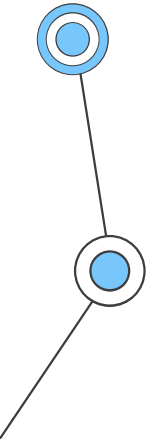


Entidades

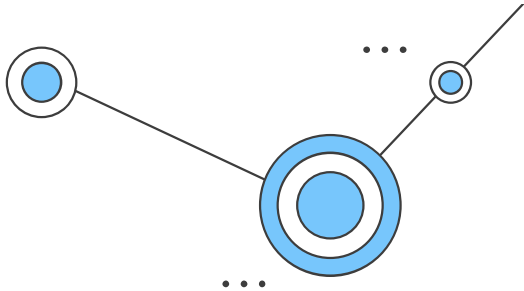


- Podemos classificar as entidades segundo o motivo de sua existência:

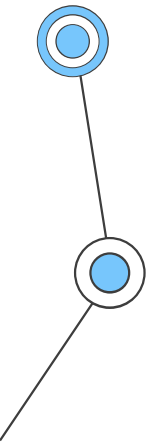
  - Entidades fortes: são aquelas cuja existência independe de outras entidades, ou seja, por si só elas já possuem total sentido de existir. Em um sistema de vendas, a entidade produto, por exemplo, independe de quaisquer outras para existir.
  - Entidades fracas: ao contrário das entidades fortes, as fracas são aquelas que dependem de outras entidades para existirem, pois individualmente elas não fazem sentido. Mantendo o mesmo exemplo, a entidade venda depende da entidade produto, pois uma venda sem itens não tem sentido.



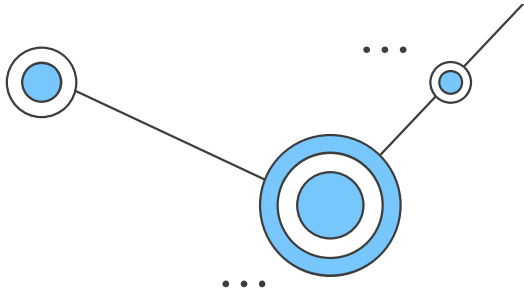
Entidades



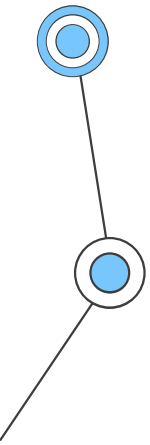
- Entidades associativas: esse tipo de entidade surge quando há a necessidade de associar uma entidade a um relacionamento existente. Na modelagem Entidade-Relacionamento não é possível que um relacionamento seja associado a uma entidade, então tornamos esse relacionamento uma entidade associativa, que a partir daí poderá se relacionar com outras entidades.
- Para melhor compreender esse conceito, tomemos como exemplo uma aplicação de vendas em que existem as entidades Produto e Venda, que se relacionam na forma muitos-para-muitos, uma vez que em uma venda pode haver vários produtos e um produto pode ser vendido várias vezes (no caso, unidades diferentes do mesmo produto).



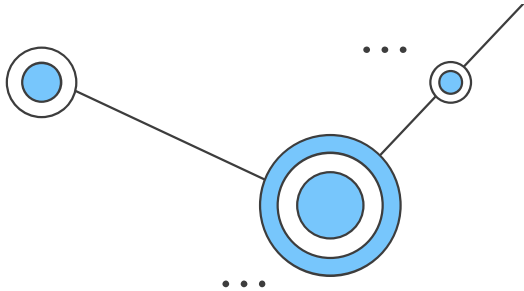
Entidades



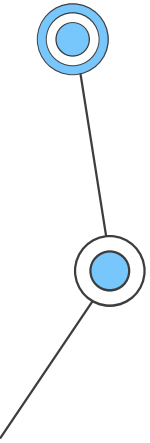
- Em determinado momento, a empresa passou a entregar brindes para os clientes que comprassem um determinado produto. A entidade Brinde, então, está relacionada não apenas com a Venda, nem com o Produto, mas sim com o item da venda, ou seja, com o relacionamento entre as duas entidades citadas anteriormente.
- Como não podemos associar a entidade Brinde com um relacionamento, criamos então a entidade associativa "Item da Venda", que contém os atributos identificadores das entidades Venda e Produto, além de informações como quantidade e número de série, para casos específicos. A partir daí, podemos relacionar o Brinde com o Item da Venda, indicando que aquele prêmio foi dado ao cliente por comprar aquele produto especificamente.



Relacionamentos

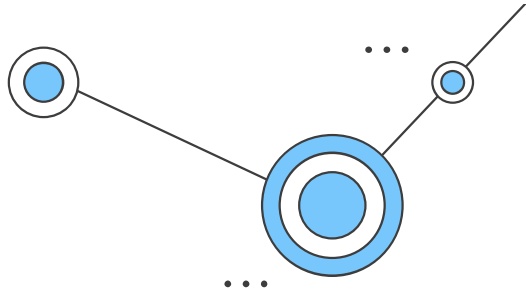


- Uma vez que as entidades são identificadas, deve-se então definir como se dá o relacionamento entre elas. De acordo com a quantidade de objetos envolvidos em cada lado do relacionamento, podemos classifica-los de três formas:



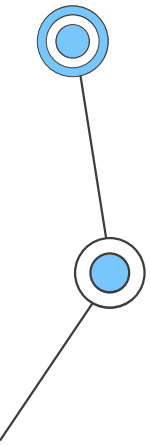
- Os relacionamentos em geral são nomeados com verbos ou expressões que representam a forma como as entidades interagem, ou a ação que uma exerce sobre a outra. Essa nomenclatura pode variar de acordo com a direção em que se lê o relacionamento. Por exemplo: um autor escreve vários livros, enquanto um livro é escrito por vários autores.

Relacionamentos



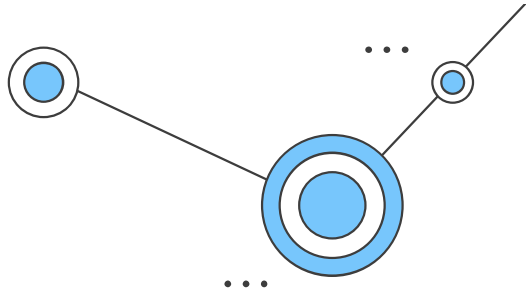
- **Relacionamento 1..1 (um para um):** cada uma das duas entidades envolvidas referenciam obrigatoriamente apenas uma unidade da outra. Por exemplo, em um banco de dados de currículos, cada usuário cadastrado pode possuir apenas um currículo na base, ao mesmo tempo em que cada currículo só pertence a um único usuário cadastrado.

- **Relacionamento 1..n ou 1..* (um para muitos):** uma das entidades envolvidas pode referenciar várias unidades da outra, porém, do outro lado cada uma das várias unidades referenciadas só pode estar ligada uma unidade da outra entidade. Por exemplo, em um sistema de plano de saúde, um usuário pode ter vários dependentes, mas cada dependente só pode estar ligado a um usuário principal. Note que temos apenas duas entidades envolvidas: usuário e dependente. O que muda é a quantidade de unidades/exemplares envolvidas de cada lado.

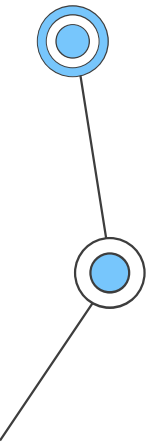


- **Relacionamento n..n ou *..* (muitos para muitos):** neste tipo de relacionamento cada entidade, de ambos os lados, podem referenciar múltiplas unidades da outra. Por exemplo, em um sistema de biblioteca, um título pode ser escrito por vários autores, ao mesmo tempo em que um autor pode escrever vários títulos. Assim, um objeto do tipo autor pode referenciar múltiplos objetos do tipo título, e vice versa.

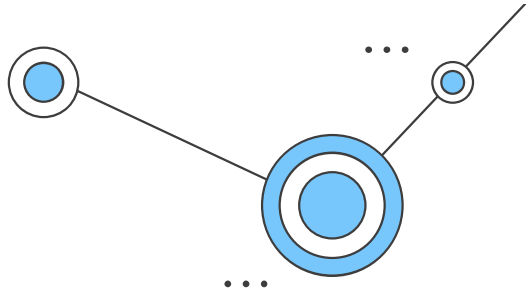
Atributos



- Atributos são as características que descrevem cada entidade dentro do domínio. Por exemplo, um cliente possui nome, endereço e telefone. Durante a análise de requisitos, são identificados os atributos relevantes de cada entidade naquele contexto, de forma a manter o modelo o mais simples possível e consequentemente armazenar apenas as informações que serão úteis futuramente. Uma pessoa possui atributos pessoais como cor dos olhos, altura e peso, mas para um sistema que funcionará em um supermercado, por exemplo, estas informações dificilmente serão relevantes.



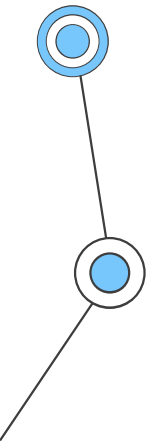
Atributos



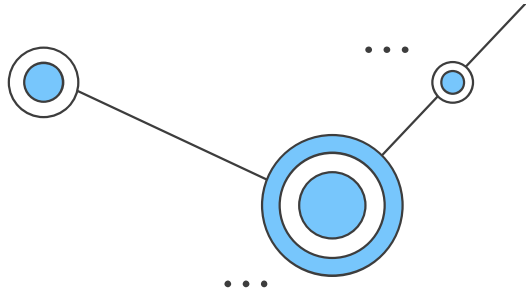
- Os atributos podem ser classificados quanto à sua função da seguinte forma:
  - **Descritivos**: representam característica intrínsecas de uma entidade, tais como nome ou cor.
  - **Nominativos**: além de serem também descritivos, estes têm a função de definir e identificar um objeto. Nome, código, número são exemplos de atributos nominativos.
  - **Referenciais**: representam a ligação de uma entidade com outra em um relacionamento. Por exemplo, uma venda possui o CPF do cliente, que a relaciona com a entidade cliente.

  Quanto à sua estrutura, podemos ainda classificá-los como:

  - **Simples**: um único atributo define uma característica da entidade. Exemplos: nome, peso.
  - **Compostos**: para definir uma informação da entidade, são usados vários atributos. Por exemplo, o endereço pode ser composto por rua, número, bairro, etc.



Atributos



- Alguns atributos representam valores únicos que identificam a entidade dentro do domínio e não podem se repetir. Em um cadastro de clientes, por exemplo, esse atributo poderia ser o CPF. A estes chamamos de Chave Primária.
- Já os atributos referenciais são chamados de Chave Estrangeira e geralmente estão ligados à chave primária da outra entidade. Estes termos são bastante comuns no contexto de bancos de dados. Mantendo o exemplo anterior, a entidade cliente tem como chave primária seu CPF, assim, a venda possui também um campo “CPF do cliente” que se relaciona com o campo CPF da entidade cliente.

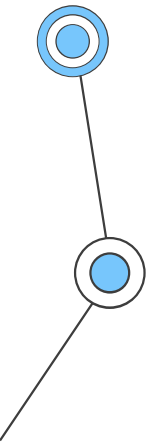
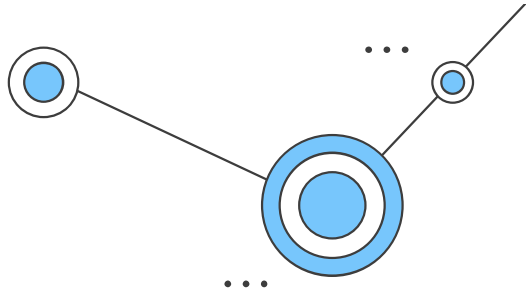
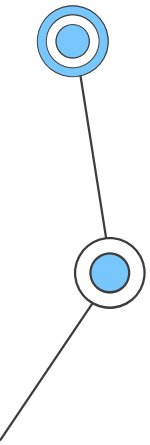


Diagrama Entidade Relacionamento



- Diagrama Entidade Relacionamento (Diagrama ER ou ainda DER) é a sua representação gráfica e principal ferramenta. Em situações práticas, o diagrama é tido muitas vezes como sinônimo de modelo, uma vez que sem uma forma de visualizar as informações, o modelo pode ficar abstrato demais para auxiliar no desenvolvimento do sistema. Dessa forma, quando se está modelando um domínio, o mais comum é já criar sua representação gráfica, seguindo algumas regras.



- O diagrama facilita ainda a comunicação entre os integrantes da equipe, pois oferece uma linguagem comum utilizada tanto pelo analista, responsável por levantar os requisitos, e os desenvolvedores, responsáveis por implementar aquilo que foi modelado.

Diagrama Entidade Relacionamento

- Em sua notação original, proposta por Peter Chen (idealizador do modelo e do diagrama), as entidades deveriam ser representadas por retângulos, seus atributos por elipses e os relacionamentos por losangos, ligados às entidades por linhas, contendo também sua cardinalidade (1..1, 1..n ou n..n).
- Porém, notações mais modernas abandonaram o uso de elipses para atributos e passaram a utilizar o formato mais utilizado na UML, em que os atributos já aparecem listados na própria entidade. Essa forma torna o diagrama mais limpo e fácil de ser lido.
- Observe na figura a seguir um exemplo simples de um diagrama para um sistema de imobiliárias.

Diagrama Entidade Relacionamento