**Modelagem conceitual/modelo domínio**: utilizado para o entendimento dos requisitos do sistema, pois explora as estruturas e conceitos do negócio. Ele se concentra nos requisitos e necessidades do usuário, fornecendo uma visão de alto nível do que o banco de dados deve conter e como os dados estão relacionados. Ex.: Diagramas Entidade Relacionamento que mostram entidades, atributos e relacionamentos entre eles, sem preocupações com detalhes de implementação.

O **modelo lógico** traduz o modelo conceitual em um formato que pode ser implementado por um SGBD específico. Ele define as tabelas, relacionamentos, chaves primárias, chaves estrangeiras e restrições de integridade necessários para representar os dados de acordo com os requisitos conceituais. É utilizada linguagem de modelagem de dados como o Modelo Relacional (SQL). Ex.: Tabelas, campos, tipos de dados e relacionamentos definidos em um sistema de gerenciamento de banco de dados relacional.

**Modelo físico** trata dos detalhes de como os dados são armazenados e acessados em nível de hardware. Isso inclui decisões sobre a estrutura de armazenamento, índices, partições, otimização de consultas e configuração do hardware. Ex.: Esquemas de armazenamento, layout de disco, otimização de índices e configurações de cache.

**Ferramentas CASE (Computer-Aided Software Engineering, ou Engenharia de Software Assistida por Computador):** permitem a modelagem de dados, geração de código SQL, otimização de consultas, documentação, gerencimaneto de mudanças, segurança e controle de acesso, integração com SGBD, colaboração...

Exemplos:

OracleDesigner ™ (Oracle ®); PowerDesigner ™ (Sybase ®); ERWin (CA ®); Freeware DBDesigner;

PyDesigner; VISIO ™ (Microsoft ®).

**O Banco de Dados representa um aspecto/recorte do mundo real que pode ser chamado de Mini-mundo.**

**A abstração, essencial na criação de modelos, consiste em simplificar elementos do mundo real, representando apenas características essenciais para resolver problemas específicos.**

Para simplificar o entendimento, é fundamental adotar práticas eficazes:

**Definição Clara das Regras de Negócio**: É fundamental garantir que as regras de negócio sejam devidamente documentadas e compreendidas por todos os envolvidos no processo de modelagem. Dessa forma, as decisões tomadas ficam mais alinhadas com as necessidades e objetivos do negócio, promovendo uma abordagem mais objetiva.

**Comunicação Aberta:** O ideal é sempre promover uma comunicação simples e aberta entre os membros da equipe de modelagem. Sempre que possível discuta as decisões e troque ideias para garantir que todos tenham uma visão compartilhada do que está sendo modelado.

**Validação com os Stakeholders:** Envolver os stakeholders (as partes interessadas) no processo de validação do modelo é crucial. Eles podem fornecer feedback valioso sobre a precisão do modelo e se ele atende aos requisitos do negócio.

**Conhecimento Multidisciplinar**: Tente contar com uma equipe de modelagem que possua conhecimentos diversos. Pessoas com diferentes perspectivas e especialidades podem enriquecer o processo de abstração e tomar decisões mais abrangentes.

**Documentação Detalhada:** Manter uma documentação detalhada das decisões de modelagem é fundamental. Esse registro permitirá que qualquer pessoa envolvida no projeto compreenda as escolhas feitas e suas justificativas, facilitando a colaboração e evitando ambiguidades.

**Iteração e Melhoria Contínua:** A modelagem é um processo contínuo e iterativo. À medida que o projeto avança, novas informações surgem, tornando ajustes e refinamentos no modelo necessários. Adotar uma postura aberta a mudanças é essencial para garantir que o modelo permaneça relevante e eficiente diante das evoluções do projeto e das demandas do negócio.

O **MER** (modelo entidade relacionamento) é um modelo conceitual usado para descrever os objetos (entidades) com suas características (atributos) e como elas se relacionam entre si (relacionamentos). Já o **DER** (diagrama entidade relacionamento) é uma representação gráfica que ajuda a visualizar as informações em situações práticas, para sua construção vamos utilizar o **BrModelo**.

**Entidades:** é um objeto único no mundo real. Ex.: clientes de uma empresa, carros que são vendidos, departamento de vendas. Pode ser **Abstrata** ou **Concreta.** No diagrama é representada por um retângulo.

**Entidade Forte:** é aquela que existe independentemente de outra entidade. No diagrama é representada por um retângulo.

**Entidade Fraca:** depende da existência de outra entidade. No diagrama é representada por um retângulo duplo, um dentro do outro.

**Graus de relacionamento**

Relacionamento **binário** tem apenas duas entidades, **ternário** seriam 3 entidades e suas relações e assim por diante, **n-ário.**

Desenho de personagem de desenho animado

Descrição gerada automaticamente com confiança média

**Relacionamento reflexivo:** o auto relacionamento é a forma utilizada para representar associações entre objetos da mesma classe, ou seja, quando temos a ocorrência de uma entidade que está associada com outras ocorrências da mesma entidade. Podemos dizer também que os relacionamentos reflexivos são uma forma de demonstrar uma relação hierárquica contida dentro de uma entidade.

De acordo com a imagem abaixo, a entidade Colaborador(a) terá um relacionamento com ela mesma, já que o gerente, que faz parte dessa entidade, gerencia outros funcionários. Dessa forma, conseguimos elucidade que um(a) colaborador(a), no caso a pessoa gerente, gerencia outros colaboradores(as) sem a necessidade de criar uma nova entidade que a represente, já que esse cargo também faz parte da classe colaboradores(as).

Uma imagem contendo Diagrama

Descrição gerada automaticamente

**Cardinalidade**

A cardinalidade descreve o número de ocorrências em um relacionamento entre entidades.

**(1, 1)** um-para-um: Uma entidade A está associada a no máximo uma entidade B, e uma entidade em B está associada a no máximo uma entidade em A. Ex.: aluno/ID -> um aluno possui apenas um ID, e um ID tem atribuído somente um aluno a ele.

**(0, 1)** zero-para-um: Uma entidade A pode ou não estar associada a uma entidade em B, e uma entidade em B, se existir, está associada a no máximo uma entidade em A.

**(1, n)** um-para-muitos: Uma entidade em A está associada a uma ou mais entidades em B, mas uma entidade em B está associada a no máximo uma entidade em A.

**(0, n)** zero-para-muitos: Uma entidade em A pode ou não estar associada a uma ou mais entidades em B, e uma entidade em B, se existir, está associada a no máximo uma entidade em A.

**(n, m)** muitos-para-muitos: uma entidade em A está associada a uma ou mais entidades em B, e uma entidade em B está associada a uma ou mais entidades em A. Ex.: cliente/produto -> um produto pode ser comprado por vários clientes, e um cliente pode comprar vários produtos.

Além desses, existem também conceitos de cardinalidade mínima e máxima, que ditam as regras de como as entidades irão se comportar diante de um relacionamento. Ex.: cada ocorrência da entidade Curso poderá se relacionar com no mínimo nenhuma e com no máximo várias ocorrências de Aluno. Cada ocorrência de Aluno irá se relacionar com no mínimo e no máximo uma ocorrência da entidade Curso.

Para determinar a cardinalidade do relacionamento entre entidades em um modelo conceitual, você pode fazer as seguintes perguntas:

1. \*\*Cardinalidade Máxima\*\* (também conhecida como tipo de relacionamento):

- Uma instância da entidade A pode estar relacionada a quantas instâncias da entidade B?

- Uma instância da entidade B pode estar relacionada a quantas instâncias da entidade A?

2. \*\*Cardinalidade Mínima\*\* (também conhecida como opção de existência):

- Uma instância da entidade A deve estar necessariamente relacionada a pelo menos uma instância da entidade B?

- Uma instância da entidade B deve estar necessariamente relacionada a pelo menos uma instância da entidade A?

As respostas a essas perguntas ajudarão a determinar se o relacionamento é um-para-um (1:1), um-para-muitos (1:N), muitos-para-um (N:1) ou muitos-para-muitos (N:M).

Por exemplo, considere duas entidades: `Estudante` e `Curso`. Se perguntarmos "Quantos cursos um estudante pode fazer?" e a resposta for "muitos", e se perguntarmos "Quantos estudantes podem fazer um curso?" e a resposta for "muitos", então o relacionamento entre `Estudante` e `Curso` é muitos-para-muitos (N:M).

**Restrição de participação ou Dependência de existência**

De acordo com as referências utilizadas para realizar os seus estudos, você pode encontrar uma outra forma de representar a cardinalidade mínima, conhecida como restrição de participação ou dependência de existência.

A restrição de participação é utilizada para especificar se a existência da entidade depende da associação a uma outra entidade, ou seja, depende do relacionamento. Existem dois tipos de restrição de participação: restrição total e restrição parcial.

A restrição total ocorre quando todas as instâncias de uma entidade X precisam estar obrigatoriamente relacionadas a alguma instância da entidade Y. Por exemplo: se nas regras de negócio do projeto foi levantado que todo(a) colaborador(a) precisa estar associado a um departamento para que a pessoa possa trabalhar na empresa, consideramos que a restrição de participação entre colaborador(a) e departamento é total, pois, ele/ela precisa estar trabalhando em, no mínimo, um departamento.

Já a restrição parcial ocorre quando todas as instâncias de uma entidade X não precisam estar obrigatoriamente relacionadas a alguma instância da entidade Y. Por exemplo: todo departamento precisa ser gerenciado por um(a) colaborador(a), mas nem todo(a) colaborador(a) precisa ser gerente de um departamento. Então, consideramos que a restrição de participação entre a relação de colaborador(a) gerenciar um departamento é uma restrição parcial, pois nem todo(a) colaborador(a) irá gerenciar um departamento.

**Entidades Associativas**

Utilizadas para simplificar relacionamentos mais complexos, ela associa duas entidades que possuem relação de muitos-para-muitos. Funciona de forma intermediária entre duas tabelas, onde existe um relacionamento complexo entre seus atributos. Ou seja, um atributo de uma tabela A se relaciona com vários atributos de uma tabela B. Ao construir o banco de dados, a ausência, de uma entidade associativa, que relaciona duas entidades, pode causar redundância de dados ou simples repetição desnecessária. Contudo, as entidades associativas ajudam na redução de dados que se repetem, melhorando o desempenho do banco de dados...

Bing responde:   
As entidades associativas, além de auxiliar no desempenho do banco de dados através da redução da repetição de dados, também trazem outros benefícios importantes:

* Organização de informações complexas: As entidades associativas ajudam a organizar informações complexas nos bancos de dados;
* Facilitação de consultas complexas: Elas facilitam a realização de consultas complexas nos bancos de dados.
* Modelagem de relacionamentos abstratos ou intrincados: As entidades associativas fornecem uma maneira eficaz de modelar relacionamentos mais abstratos ou intrincados entre entidades.
* Questionamento de regras da organização: Elas permitem questionar as regras da organização.
* Revelação de incoerências atuais: Elas ajudam a revelar incoerências atuais.
* Construção faseada: Elas permitem uma construção faseada do banco de dados.
* Facilitação da comunicação: Elas facilitam a comunicação entre os usuários e os informáticos.

No BrModelo uma entidade associativa é representada por um retângulo com um losango dentro.

Os relacionamentos podem ser vistos como a forma que uma ou mais entidades se conectam entre si. O grau de um relacionamento é determinado pelo número de entidades envolvidas na mesma relação. Geralmente, vamos identificar os relacionamentos através de verbos como: compra de, depende de, etc.

**Generalizações e Especializações**

Quando uma entidade possui uma variação/graduação em sua estrutura, precisamos criar sub-entidades. Por exemplo, uma entidade Cliente, que pode ser pessoa física ou jurídica, essa variação nos leva a criar duas novas entidades ou subclasses da entidade ou superclasse Cliente.

Na modelagem conceitual de banco de dados, a \*\*especialização\*\* é um conceito usado para representar objetos do mundo real que podem ser categorizados e representados em uma hierarquia que mostra as dependências entre entidades de uma mesma categoria¹.

Existem diferentes tipos de especializações:

1. \*\*Especialização Total\*\*: Toda a entidade na superclasse deve ser um membro de pelo menos uma das subclasses na especialização¹. Por exemplo, todo cliente OU é uma pessoa física OU uma pessoa jurídica⁴.

2. \*\*Especialização Parcial\*\*: Uma entidade na superclasse não precisa necessariamente ser membro de pelo menos uma das subclasses na especialização¹. Por exemplo, nem todo funcionário é Motorista ou Secretária⁴.

3. \*\*Especialização Exclusiva\*\*: Especifica que as subclasses da especialização devem ser mutuamente exclusivas. Isto significa que uma entidade pode ser membro de, no máximo, uma das subclasses da especialização¹².

4. \*\*Especialização Não-exclusiva\*\*: Especifica que as subclasses da especialização podem se sobrepor. Uma entidade pode ser membro de mais de uma subclasses da especialização¹².

**Atributos**

Os atributos descrevem as propriedades das entidades.

Atributos de valor único ou atômico, quando há um único valor no atributo. Ex.: nome, idade.

Atributos multivalorados, pode ter mais de um valor. Ex.: telefone, e-mail...

Atributos armazenados, são os atributos que receberão um armazenamento por interação. Ex.: o nome, o usuário irá preencher o nome, assim como telefone...

Atributos derivados, são derivados de informações de outros atributos. Por exemplo: suponha que exista um campo para inserir a data de nascimento e um campo com a data de hoje. Pode ser o aniversário ou a idade atual, por exemplo.

Atributo null, pode ser interpretado como não aplicável ou desconhecido. Suponha que um campo específico do banco de dados solicite uma informação que, para determinada pessoa, não faz sentido, não é aplicável. Nesse caso, ele ficará como Null (nulo). A partir disso, temos dois desdobramentos a considerar: Um campo que pode ser obrigatório, ou seja, não aceita nulos. A pessoa precisa preencher com alguma informação.

Um campo opcional, que aceita o Null. Ele aceitará uma resposta que não é aplicável

**Atributo simples:** É um tipo de atributo indivisível, ou seja, é um atributo atômico. Um exemplo deste tipo é o atributo CPF, pois ele não pode ser dividido em partes menores para formar outros atributos.

**Atributo composto:** Pode ser dividido em partes menores que representam outros atributos, como endereço. Ele pode ser subdividido em atributos menores, como: cidade,estado, rua, CEP.

**Atributo multivalorado:** É aquele que pode ter um ou N (vários) valores associados a ele. Por exemplo: o atributo telefone de um cliente. Este pode ter um ou vários telefones.

**Atributo derivado e armazenado:** Atributos derivados dependem de outro atributo ou até mesmo outra entidade para existir, como, idade e data de nascimento. Para descobrirmos a idade de uma pessoa, precisamos da sua data de nascimento. Então, consideramos o atributo idade como derivado do atributo data de nascimento, chamado, por sua vez, de atributo armazenado.

**Atributo chave:** É utilizado para identificar de forma única uma entidade, ou seja, os valores associados a esse atributo são distintos entre o conjunto de entidades. Como exemplo, podemos utilizar o Código do Produto. Ele é único e pode ser utilizado como atributo chave, uma vez que cada produto recebe apenas um Código distinto.

**Chave primária:** é um campo ou conjunto de campos que identifica de forma única cada registro em uma tabela de banco de dados. A chave primária não pode conter valores nulos e deve ser única em toda a tabela, além de só entidades fortes as possuem. Ex.: ID de uma tabela ‘Pessoa’.

CREATE TABLE Pessoa (

ID int NOT NULL,

Nome varchar(255) NOT NULL,

Idade int,

PRIMARY KEY (ID)

);

**Chave parcial:** também conhecida como chave candidata, é um conjunto de atributos que poderia ser uma chave primária, mas não foi escolhida como tal. Em outras palavras, é um conjunto de campos que também identifica de forma única um registro, mas não é a chave primária principal da tabela. Ex.: CPF.

**Chave Estrangeira:** é um campo ou conjunto de campos em uma tabela que faz referência à chave primária de outra tabela. A chave estrangeira estabelece uma relação entre as tabelas e permite manter a integridade referencial entre elas. Ex.: PessoaID, fazendo referencia ao ID da tabela ‘Pessoa’.

CREATE TABLE Pedido (

PedidoID int NOT NULL,

PedidoNumero int NOT NULL,

PessoaID int,

PRIMARY KEY (PedidoID),

FOREIGN KEY (PessoaID) REFERENCES Pessoa(ID)

);

**Revisão**

**Entidades e Atributos**

Em um modelo Entidade-Relacionamento, as entidades representam objetos ou conceitos do mundo real e são descritas por seus atributos. Por exemplo, em um sistema de biblioteca, podemos ter a entidade ‘Livro’ com atributos como “Titulo”, “Autor”, “Ano de Publicação” etc.

**Chave Primária e Identificadores**

Cada entidade possui uma chave primária, que é um atributo (ou conjunto de atributos) que identifica exclusivamente cada instância dessa entidade. Por exemplo, a entidade “Editora” pode ter a chave primária “cod\_editora” para identificar cada editora de forma única.

**Entidades Fortes e Fracas**

Uma entidade forte é aquela que possui sua própria chave primária e não depende de nenhuma outra entidade para ser identificada de forma única. Por outro lado, uma entidade fraca é aquela que não possui uma chave primária única por si só e depende da relação com uma entidade forte para ser identificada.

**Relacionamento e Entidades Fracas**

Os relacionamentos são conexões significativas entre entdades. Podemos ter relacionamentos entre entidades fortes ou entre entidades fortes e fracas. No exemplo anterior, temos a entidade forte “Editora” e a entidade fraca “Livro”. A entidade “Livro” é fraca porque precisa de uma referência à entidade “Editora” para ser identificada de forma única. Portanto, além de seus próprios atributos, a entidade “Livro” inclui a chave primária da entidade “Editora” (“cod\_editora”) como parte de sua própria chave primária.

**Entidade Estoque como Exemplo**

Agora, vamos considerar a entidade "Estoque". Ela está relacionada com a entidade "Livro" e, portanto, precisa receber as chaves identificadoras da entidade "Livro" para manter a integridade da relação entre elas. Nesse caso, a entidade "Estoque" receberá tanto a chave "cod\_livro" como a chave da entidade "Editora" ("cod\_editora") como parte de sua própria chave primária. As entidades fortes têm suas próprias chaves primárias, enquanto as entidades fracas herdam as chaves primárias das entidades fortes com as quais estão relacionadas.

**BrModelo**

O aplicativo é bem intuitivo, possui funcionalidade para suporte de vários diagramas além do Conceitual (usado nesse material), conta com 2 menus laterais como consta na imagem abaixo:

Diagrama

Descrição gerada automaticamente

No menu da direita, ficam os artefatos utilizados para construir o diagrama, conta com cursor, entidade, relacionamento, auto relacionamento, entidade associativa, dentre outros...