PROJETO E-JOVEM

EDUCANDUS

APOSTILA BANCO DE DADOS

Índice

1. Introdução

Imagine uma situação onde alguém escreve em seu caderno apenas o numero 3, você com certeza iria perguntar, o que significa esse 3? Agora imagine que essa mesma pessoa escreve em se caderno 3 reais, ou 3 cadeiras ou 3 kilos de batatas, dessa vez você iria conseguir entender o que a pessoa escreveu essa é a diferença entre um DADO e uma INFORMAÇÃO.

Por definição temos que um dado é um valor que sozinho não tem sentido algum. Já uma informação é um DADO contextualizado.

Isso faz com que a informação se torne algo muito importante para todos nós, é com elas que podemos tomar as mais diversas decisões em nossas vidas e também é um grande alerta para o seguinte problema: Informação Errada = Decisão Errada.

Nas organizações a importância da informação para a tomada de, baseada nos seus dados gerados, tem impulsionado o desenvolvimento dos sistemas de processamento de informações.

Em tempos atrás o grande problema residia na limitação à quantidade e capacidade de processamento de informação que era possível tratar num computador, hoje em dia a tecnologia permite ultrapassar essas dificuldades.

A necessidade das organizações em possuir um sistema de gestão mais eficaz torna-as cada vez mais dependentes da informação existente e dos métodos para tratá-la. O tempo das enormes e fastidiosas listagens de computador já passou hoje a informação tem de ser compreensível, completa, fácil e rápida de se obter.

1.1. Banco de Dados e SGBD

Bancos de dados, (ou bases de dados), são conjuntos de dados com uma estrutura regular que organizam informação. Um banco de dados normalmente agrupa informações utilizadas para um mesmo fim. Um banco de dados é usualmente mantido e acessado por meio de um software conhecido como Sistema Gerenciador de Banco de Dados (SGBD).

Sistema de Gerenciamento de Banco de Dados

São um conjunto de programas que fazem uma gestão autônoma da informação, de acordo com um modelo preestabelecido e adaptado à empresa ou qualquer outro órgão que os utilizem.

Diferença entre Banco de Dados e Sistema Gerenciador de Banco de Dados?

Banco de Dados é uma coleção de dados inter-relacionados e logicamente coerente, representando informações sobre um domínio específico.

SGBD é um software com recursos específicos para facilitar o processo de definição, construção, manipulação e o gerenciamento das informações dos bancos de dados e o desenvolvimento de programas aplicativos.

Exemplos de Banco de Dados:
Agenda Telefônica
Atlas Geográfico
Cadastro de Fornecedores
Lista de Pedidos
Fichas do acervo de uma biblioteca

Exemplos de SGBD:
Oracle
PostgreSQL
SQL Server
MySQL
HSQLDB
Paradox
FireBird

2. Abstração

O SGBD deve fornecer ao usuário uma "representação conceitual" dos dados, sem fornecer muitos detalhes de como as informações são armazenadas. Um "modelo de dados" é uma abstração de dados que é utilizada para fornecer esta representação conceitual utilizando conceitos lógicos como objetos, suas propriedades e seus relacionamentos. Uma abstração depende mais do observador do que da realidade observada.

Existem 3 mecanismos de abstração:

- **Classificação/Instanciação**: Consiste na categorização dos objetos em grupos, com base em algum conjunto de propriedades comuns. A classificação estabelece um relacionamento (É INSTANCIA DE), entre cada elemento e a sua classe.

- **Generalização/Especialização**: Consiste em um relacionamento de subconjunto entre elementos de duas ou mais categorias/conjuntos. Todas as propriedades definidas para a categoria generalizada são herdadas pelas categorias especializadas. Todo elemento de um subconjunto especializado é também elemento do seu respectivo conjunto generalizado. Essa abstração estabelece um relacionamento(É UM) entre a categoria generalizada e as categorias especializadas.

- **Agregação/Desagregação**: Define uma nova categoria a partir de outras categorias. Essa abstração estabelece um relacionamento (É PARTE DE) entre os componentes e a classe.

  Abstração de Dados:

  Abstração de dados é o uso da abstração para selecionar as propriedades relevantes de um conjunto de dados. Deve-se analisar o contexto e verificar a pertinência de cada propriedade e quais propriedades são relevantes para armazenarmos em um banco de dados.

Existem 3 níveis de abstração:

- **Nível de Visão:** consiste no nível mais alto de abstração. Descreve partes do BD, de acordo com as necessidades de cada usuário. É a forma como os dados são vistos pelo usuário.

- **Nível Lógico:** consiste nos dados que estão armazenados e seus relacionamentos. Neste nível, o BD é descrito através de estruturas relativamente simples, que podem envolver estruturas complexas no nível físico.

- **Nível Físico:** consiste no nível mais baixo de abstração. Descreve como os dados estão realmente armazenados, englobando estruturas complexas de baixo nível.

  Cada nível de abstração tem interação e manipulação com o bando de dados de forma específica.

3. Modelo

Modelo é uma representação não ambígua que visa ajudar a entender a abstração de uma realidade.

Complexidade

Modelar BD envolve gerenciar Complexidades e Riscos. A complexidade e os riscos são proporcionais ao tamanho do BD. Modelos ajudam a minimizar a complexidade e planejar a solução.

Por que usar Modelos?

- **Gerenciamento da Complexidade:** Dividir para conquistar + princípio da abstração.
- **Comunicação entre as pessoas envolvidas:** Servem como ponto de referência comum e não ambíguo.
- **Redução dos custos no desenvolvimento:** É mais barato corrigir erros no modelo do que na implementação.
- **Predição do comportamento futuro do sistema:** Permite experimentar problemas/soluções antes do desenvolvimento.

Modelos de Dados

Modelo de Dados é um tipo de modelo para descrever dados, relações de dados, semântica de dados e restrições de consistência.

Instância de BD

Instância de BD é a coleção dos dados armazenados no BD em um determinado momento.

Esquema de BD

Esquema de BD é a descrição de um BD segundo um modelo de dados.

Modelagem de BD

Modelagem de BD é a atividade de especificação das estruturas de dados e regras de negócio para um esquema de BD.

Projeto de BD

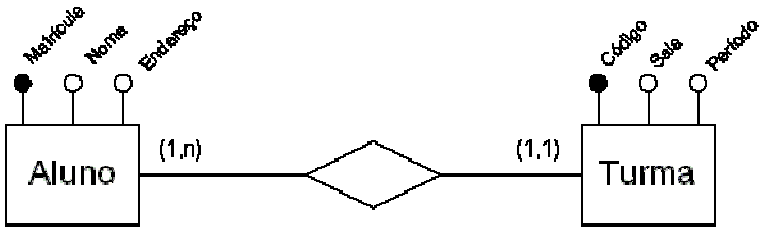
Projeto de BD é a atividade de modelagem de dados em diversos níveis de abstração de forma que o BD torne-se eficaz, eficiente e fácil de manter.

O projeto de Banco de Dados se dá em três fases:

- Projeto Conceitual
- Projeto Lógico
- Projeto Físico

Projeto Conceitual: É a descrição do BD de maneira independente ao SGBD, ou seja, define quais os dados que aparecerão no BD, mas sem se importar com a implementação que se dará ao BD. Desta forma, há uma abstração em nível de SGBD.

Uma das técnicas mais utilizadas dentre os profissionais da área é a abordagem entidade-relacionamento (ER), onde o modelo é representado graficamente através do diagrama entidade-relacionamento (DER).



Projeto Lógico:

Descreve o BD no nível do SGBD, ou seja, depende do tipo particular de SGBD que será usado. Não podemos confundir com o Software que será usado. O tipo de SGBD que o modelo lógico trata é se o mesmo é relacional, orientado a objetos, hierárquico, etc. Abordaremos o SGBD relacional, por serem os mais difundidos. Nele, os dados são organizados em tabelas.

Aluno(mat_aluno, nome, endereco)
Turma (cod_turma, sala, periodo)

Obs: É importante salientar que os detalhes internos de armazenamento, por exemplo, não são descritos no modelo lógico, pois estas informações fazem parte do modelo físico, que nada mais é que a tradução do modelo lógico para a linguagem do software escolhido para implementar o sistema.

Projeto Físico

Inicia com Esquema Lógico e resulta no Esquema Físico.É uma descrição da implementação do Esquema Lógico segundo as estruturas de armazenamento e métodos de acesso do SGBD, cujo o objetivo é otimizar a manipulação dos dados.

Esquema Físico: É uma descrição da estrutura do BD segundo a Linguagem de Definição de Dados (LDD) do SGBD alvo, que é especificado segundo um Modelo Físico

Ex: LDD do Oracle.

4. Ciclo de Vida de um Banco de Dados

Ciclo de Vida de BD é o conjunto de fases que compreende desde a concepção até a manutenção e evolução do BD.

1ª Fase: Estudo Inicial do Banco de Dados

Propósito Geral:

- **Analisar situação da companhia:**
  - Qual é o ambiente geral da organização e qual é sua missão dentro deste ambiente?
  - Qual é a estrutura da organização?
- **Definir problemas e restrições:**
  - Como funcionam os sistemas existentes?
  - O que o sistema requer como entrada?
  - O que o sistema gera como saída?
  - Quais são as relações operacionais entre as unidades de negócio?
  - Quais são os limites e restrições impostos sobre o sistema?

- **Definir objetivos**

    - Qual é o objetivo inicial do sistema proposto?
    - O sistema irá fazer interface com outro sistema?
    - O sistema deve compartilhar dados com outros sistemas?

- **Definir escopo e limites**

    - **Escopo** – Qual é a extensão do projeto?
    - **Limite** – Quais são as limitações?
        - De orçamento;
        - De hardware;
        - De software.

2ª Fase: Projeto do Banco de Dados

É a fase mais importante e apresenta as seguintes subfases.

- Projeto Conceitual;
- Seleção do SGBD;
- Projeto Lógico;
- Projeto Físico.

3ª Fase: Implementação e Carga

Criação e inserção de dados nas tabelas e outras questões importantes de implementação:

- Performance;
- Segurança;
- Backup e recovery;
- Integridade;

4ª Fase: Teste e Avaliação

O BD é testado e aproveita-se para fazer ajustes finos de performance, integridade, segurança etc. Esta fase pode acontecer em paralelo com a programação da aplicação.

Algumas ações são realizadas quando os testes falham:

- Ajustes Finos conforme manuais de referências;
- Modificação do projeto físico;
- Modificação do projeto lógico;
- Atualização ou mudança de SGBD, de Hardware ou de Software.

5ª Fase: Operação

Nesta fase o BD é considerado operacional. A partir desta fase, inicia-se o processo de evolução do BD. Onde alguns problemas inesperados podem acontecer. E também a demanda por mudanças é constante.

6ª Fase: Manutenção e Evolução

Nesta fase você deve realizar as seguintes ações:

- Manutenção preventiva;
- Manutenção corretiva;
- Manutenção adaptativa;
- Atribuição de novas permissões de acesso;
- Geração de estatísticas de acesso ao BD para monitorar a performance;
- Auditoria periódica da segurança do BD;
- Resumos periódicos sobre o uso do sistema;

Quando criamos uma base de dados temos os seguintes objetivos:

- Diminuir o espaço ocupado pela informação;
- Facilitar a atualização da informação;
- Aumentar a velocidade de pesquisa;
- Evitar a redundância de informação.

Quando não utilizar um SGBD:

Em algumas situações, o uso de um SGBD pode representar uma carga desnecessária aos custos quando comparado à abordagem processamento tradicional de arquivos.

Por exemplo:

- Alto investimento inicial na compra de software e hardware adicionais;
- Generalidade que um SGBD fornece na definição e processamento de dados;
- Sobrecarga na provisão de controle de segurança, controle de concorrência, recuperação e integração de funções.

Problemas adicionais podem surgir caso os projetistas de banco de dados ou os administradores de banco de dados não elaborem os projetos corretamente ou se as aplicações não são implementadas de forma apropriada. Se o DBA não administrar o banco de dados de forma apropriada, tanto a segurança quanto a integridade dos sistemas podem ser comprometidas.

A sobrecarga causada pelo uso de um SGBD e a má administração justificam a utilização da abordagem processamento tradicional de arquivos em casos como:

- O banco de dados e as aplicações são simples, bem definidas e não se espera mudanças no projeto;
- A necessidade de processamento em tempo real de certas aplicações, que são terrivelmente prejudicadas pela sobrecarga causada pelo uso de um SGBD;
- Não haverá múltiplo acesso ao banco de dados.

Vantagens na utilização de BD's

Os benefícios aqui abordados dividem-se em três categorias principais, nomeadamente:

Benefícios de centralização de dados:

- **Redução/Eliminação de redundância de dados**: evitar a repetição de informação desnecessária, reduzindo também o espaço ocupado pela base de dados;
- **Melhoria na concorrência de dados**: aumentar a eficiência no acesso aos dados;
- **Obtenção de informação antecipadamente**: Aceder e obter informação de forma mais rápida e eficaz;
- **Simplificação da infra-estrutura de informação**: permitir uma estruturação e organização da informação de forma mais simples permitindo, deste modo, alcançar os pontos referidos acima.

Benefícios resultantes de uma melhor gestão de dados:

- **Organização e controlo dos dados**: a simplificação da própria estrutura da base de dados implica benefícios na organização dos dados o que é uma mais valia para a gestão e controlo dos dados;
- **Recuperação, backup e rasteio de dados**: por vezes pode acontecer que se percam dados relevantes para a base de dados. Nestes casos é importante que haja mecanismos de recuperação dos mesmos, de forma a ser possível o seu restauro. É importante, também, que a base de dados permita a execução de backups, isto é, de cópias de segurança da informação armazenada;
- **Simplificação e aperfeiçoamento da segurança**: a segurança é crucial nos dias de hoje. Não só para prevenir ataques externos à base de dados, mas também como forma de restringir o acesso aos dados por parte de utilizadores sem privilégios para tal;
- **Melhor integridade dos dados**: apresentar os dados com rigor e qualidade.

Vantagens na utilização de BD's:

Melhoria de performance através de:

- **Chamadas (*calls*) de aplicações**: permitem que outras aplicações, que não aquela que gere a base de dados, aceda aos dados e os utilizem para obter determinados resultados;

- **Mecanismos de recuperação (*unit recovery mechanism*, URM)**: são mecanismos que permitem a qualquer momento restaurar informação perdida pela base de dados.

Componentes de um Banco de Dados

Um Banco de Dados é composto pelas seguintes partes:

Gerenciador de Acesso ao Disco: O SGBD utiliza o Sistema Operacional para acessar os dados armazenados em disco, controlando o acesso concorrente às tabelas do Banco de Dados. O Gerenciador controla todas as pesquisas *queries*) solicitadas pelos usuários no modo interativo, os acessos do compilador DML, os acessos feitos pelo Processador do Banco de Dados ao Dicionário de Dados e também aos próprios dados.

O **Dicionário de Dados** contém o esquema do Banco de Dados, suas tabelas, índices, forma de acesso e relacionamentos existentes.

O **Compilador DDL** (*Data Definition Language*) processa as definições do esquema do Banco de Dados, acessando quando necessário o Dicionário de Dados do Banco de Dados.

O **Processador do Banco de Dados** manipula requisições à própria Base de Dados em tempo de execução. É o responsável pelas atualizações e integridade da Base de Dados.

O **Processador de Pesquisas** (*queries*) dos usuários, analisa as solicitações, e se estas forem consistentes, aciona o Processador do Banco de Dados para acesso efetivo aos dados.

As aplicações fazem seus acessos ao pré-compilador DML da linguagem hospedeira, que os envia ao Compilador DML (*Data Manipulation Language*) onde são gerados os códigos de acesso ao Banco de Dados.

5. Tabelas

Tabelas em banco de dados são objetos criados para armazenar os dados fisicamente, onde os dados são armazenados em linhas (registros) e colunas (campos). Os dados de uma tabela normalmente descrevem um assunto tal como clientes, vendas etc.

Campos

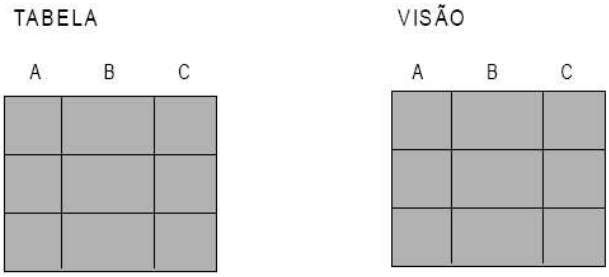
Registros	Nome	Número	Classe	Departamento
	Soares	17	1	DCC
	Botelho	8	2	DCC

6. Visões

Consistem em tabelas do banco de dados que não armazenam dados, sua principal está no aumento de segurança por propiciar uma visão limitada e controlada dos dados que podem ser obtidos da base (a depender do SGBD utilizado) e a performance por utilizar uma consulta previamente otimizada, tornando desnecessário este processo quando ela é realizada.

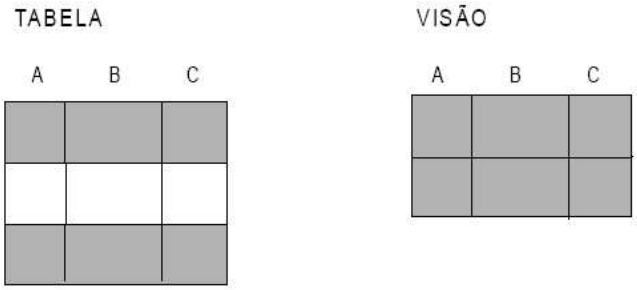
As visões apresentam os seguintes tipos:

Visão idêntica:

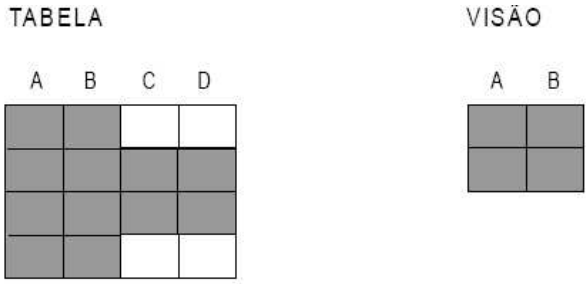


Visão por seleção de colunas:

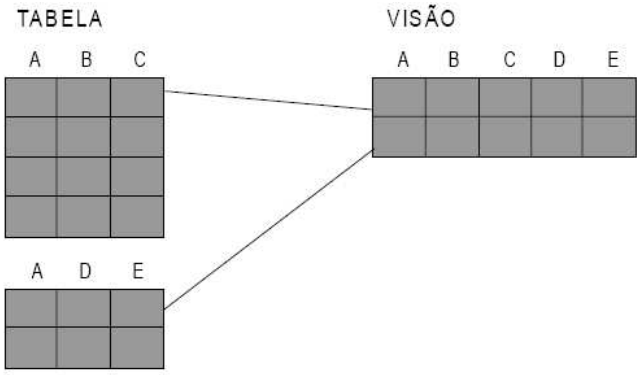
Visão por seleção de linhas:



Visão por seleção de colunas:



Visão por junção de tabelas:



7. Índice

É uma ferramenta usada pelo gerenciador de Banco de Dados para facilitar a busca de linhas dentro de uma tabela.

E apresenta os seguintes tipos:

Índice Único:

- Índice criado a partir da chave primária, não permite a inclusão de linhas duplicadas.

Índice de Performance:

- Facilita a busca de linhas na tabela

8. Linguagem de Banco de Dados

Para podermos inserir, apagar, atualizar dados em uma base de dados, precisamos de uma linguagem de banco de dados. A linguagem que veremos aqui será a de Consulta Estruturada (Structured Query Language) mais conhecida como SQL.

Constituem subconjunto da linguagem, a Linguagem de Definição de Dados (Data Definition Language - DDL) e a Linguagem de Manipulação de Dados (Data Manipulation Language - DML).

SQL: *Structured Query Language* que não é mais do que uma linguagem padrão de comunicação com base de dados. Falamos, portanto, de uma linguagem normalizada que nos permite trabalhar com qualquer tipo de linguagem (ASP ou PHP) em combinação com qualquer tipo de base de dados (MS Access, Oracle, MySQL etc).

DDL: é o conjunto de comandos SQL responsáveis pela definição dos dados, ou seja, pela criação de bancos, esquemas, tabelas, campos, tipos de dados, restrições etc. É utilizada pelo DBA e projetistas de banco de dados para definir seus esquemas.

Veja abaixo os exemplo de DDL:

Criar Banco de Dados

Sintaxe:
```
CREATE DATABASE <nome_do_banco_de_dados>;
```

Exemplo:
```
CREATE DATABASE locadora;
```

Exclui Banco de Dados

Sintaxe:
```
DROP DATABASE <nome_do_banco_de_dados>;
```

Exemplo:
```
DROP DATABASE locadora;
```

Criar Tabela

Sintaxe:
```
CREATE TABLE <nome_tabela> (
<nome_coluna_1>  <tipo_dado>  <restricao>,
<nome_coluna_2>  <tipo_dado>  <restricao>,
<nome_coluna_n>  <tipo_dado>  <restricao>,
PRIMARY KEY (nome(s)_atributo(s)),
FOREIGN KEY (nome_atributo) REFERENCES <nome_tabela>
);
```

Alterar Tabela

Para modificar a estrutura de uma tabela após esta ter sido criada use a instrução ALTER TABLE.
Sua sintaxe é:

```
ALTER TABLE <nome_tabela> alteração1, alteração2, ... , alteraçãoN;
```

Renomear Tabela

O seguinte commando também pode ser usado para renomear uma tabela:

```
RENAME TABLE nome_tabela TO novo_nome_tabela
        [,nome_tabela2 TO novo_nome_tabela2]
        [,nome_tabelaN TO novo_nome_tabelaN];
```

Excluir Tabela

Para excluir tabelas use o comando:

```
DROP TABLE <nome_tabela> [, <nome_tabela2>,...];
```

DROP TABLE remove uma ou mais tabelas. Todos os dados e definições de tabela são *removidos*, assim **tenha cuidado** com este comando!

Criar Índice

Sintaxe:
```
CREATE INDEX <nome_indice> ON <nome_tabela> (<index_col_name>,...)

<index_col_name>:
     <col_name> [ASC | DESC]
```

Uma especificação <index_col_name> pode finalizar com ASC ou DESC. Estas palavras chaves são permitidas para extensão futura para especificar o armazenamento do valor do índice em crescente ou decrescente. Atualmente elas são analisadas, mas ignoradas; valores de índice são sempre armazenados em ordem crescente.

Excluir Índice

Sintaxe:

```
DROP INDEX <nome_indice> ON <nome_tabela>;
```

DROP INDEX apaga o índice chamado nome_indice da tabela nome_tabela. DROP INDEX pode não fazer nada em algumas versões do MySQL. Dependendo da versão o DROP INDEX é mapeado em uma instrução ALTER TABLE para apagar o índice.

Criar View

Sintaxe:

```
CREATE [OR REPLACE] VIEW
<nome_view> AS <SELECT statement> [<check options>];
```

Excluir View

Sintaxe:

```
DROP VIEW [IF EXISTS] <nome_view> [, <nome_view2>,…];
```

DML: É o conjunto de comandos SQL responsáveis pela manipulação dos dados, como: inserir, consultar, atualizar e excluir.

Obs: Atente para que suas consultas sejam:

- simples e claras;
- contenham somente campos estritamente necessários;
- sejam otimizadas para o desempenho máximo.

Pesquisando no banco de dados

A pesquisa no banco de dados é feita com a instrução SELECT.

Sua sintaxe básica é:

SELECT <campos> FROM <nome_tabela> WHERE <condição> ORDER BY <campo>;

Especificando condições para a pesquisa: WHERE

A cláusula `WHERE` especifica a condição da pesquisa. Abaixo segue uma tabela com os operadores mais usados com `WHERE`.

OPERADOR	EXEMPLO	DESCRIÇÃO
=	nrcliente = 154	Testa se dois valores são iguais.
>	quantidade > 100	Testa se um valor é maior que outro.
<	quantidade < 100	Testa se um valor é menor que outro.
<=	quantidade <= 100	Testa se um valor é menor ou igual a outro.
!= OU < >	quantidade != 0	Testa se dois valores são diferentes.
IS NOT NULL	endereco IS NOT NULL	Testa se um campo não está vazio.
IS NULL	endereco IS NULL	Testa se um campo está vazio (nulo, ou seja, **nada** foi inserido nele, nem um espaço em branco).
BETWEEN	quantidade BETWEEN 0 AND 100	Testa se um valor está entre um valor mínimo (inclusive) e um valor máximo (inclusive).
IN	cidade IN ("Recife","Olinda","Caruaru")	Testa se um valor pertence a um conjunto.
NOT IN	cidade NOT IN ("Recife","Olinda","Caruaru")	Testa se um valor não pertence a um conjunto.
LIKE	nome LIKE ("Samuel%")	Testa se um valor corresponde a um padrão (**%** equivale ao conhecido *****, é o curinga).
NOT LIKE	nome NOT LIKE ("Samuel_")	Testa se um valor **não** corresponde a um padrão (_ equivale ao curinga que corresponde a um caracter).
REGEXP	nome REGEXP	Testa se um valor corresponde a uma expressão regular.

Especificando mais de uma condição para a pesquisa

Para especificar mais de uma condição para a pesquisa use as palavras and e or:

Se quiser selecionar registros que atendam a CONDIÇÃO1 e a CONDIÇÃO2, faça assim:

SELECT <campos>

FROM <tabela>

WHERE <CONDIÇÃO1> AND <CONDIÇÃO2>;

Se quiser selecionar registros que atendam a CONDIÇÃO1 ou a CONDIÇÃO2, faça assim:

SELECT <campos>

FROM <tabela>

WHERE <CONDIÇÃO1> OR <CONDIÇÃO2>;

Exibindo todos os registros de uma tabela

Caso queira exibir todos os registros, você pode usar o "*" (asterisco):

```
mysql> select * from cliente;
+--------+---------+
| codigo | nome    |
+--------+---------+
|      1 | João    |
|      2 | Maria   |
|      3 | José    |
|      4 | Manuel  |
|      5 | Adão    |
|      6 | Rodrigo |
|      7 | Davi    |
|      8 | Karla   |
|      9 | Samuel  |
|     10 | Ana     |
+--------+---------+
10 rows in set (0.00 sec)

mysql>
```

Inserindo dados I

Para inserir dados numa tabela use o comando:

INSERT INTO TABELA (campo1, campo2, ... , campon) VALUES

(valor1, valor2, ... , valorn);

Inserindo dados II

Outra sintaxe utilizada para inserir dados é:

INSERT INTO TABELA

SET campo1=valor1, campo2=valor2, ..., campon=valorn;

Atualizando registros

Para atualizar registros use a instrução UPDATE.

Sua sintaxe é:

UPDATE <tabela>

SET <coluna1 = valor1>,<coluna2 = valor2>,<...>

WHERE <condição>

LIMIT <nr1>, <nr2>;

9. A linguagem SQL

Fundamentada no modelo relacional, inclui comandos para:

- Definição de dados;
- Atualização;
- Consulta.

Vantagens de SQL

Independência de fabricante.
Portabilidade entre sistemas.
Redução de custos com treinamento.
Comandos em inglês.
Consulta interativa.
Múltiplas visões de dados.
Manipulação dinâmica dos dados.

Desvantagens de SQL

A padronização inibe a criatividade, fazendo com que a linguagem fique longe de ser a linguagem relacional ideal.

Algumas críticas:

- Falta de ortogonalidade nas expressões;
- Discordância com as linguagens hospedeiras não dá suporte a alguns aspectos do modelo relacional.

Enfoques de SQL

- Linguagem interativa de consulta (ad-hoc): usuários podem definir consultas independente de programas.

- Linguagem de programação para acesso a banco de dados: comandos SQL embutidos em programas de aplicação.

- Linguagem de administração de dados: o DBA pode utilizar SQL para realizar suas tarefas.

- Linguagem cliente/servidor: os programas clientes usam comandos SQL para se comunicarem e compartilharem dados com o servidor.

- Linguagem para banco de dados distribuídos: auxilia na distribuição de dados por vários nós e na comunicação de dados com outros sistemas.

- Caminho de acesso a outros bancos de dados em diferentes máquinas: auxilia na conversão entre diferentes produtos em diferentes máquinas.

9.1. Extração de Dados de uma Tabela

Comando SELECT: selecionando atributos (Projeção).

SELECT <lista de atributos> FROM <tabela>

Exemplos:

Listar nome e salário de todos os empregados:

SELECT Nome, Salário FROM Empregado;

Listar o conteúdo de Empregado:

SELECT * FROM Empregado;

O "*" selecionando todos os atributos.

9.1.1. Select e Condições

Selecionando tuplas da tabela:

SELECT <lista de atributos> FROM <tabela> WHERE <condição>

Onde condição é:

<nome atributo> <**operador**> <valor>

Operadores Relacionais: =, !=, <, <=, >, >=.

Operadores Lógicos: AND, OR e NOT.

Selecionando tuplas da tabela.

Exemplos:
Listar nome e sexo dos empregados do departamento 15:
SELECT Nome, Sexo FROM Empregado WHERE Num-Dep = 15;
Listar nome e sexo dos empregados do departamento 15 com salário > R$ 1.000,00:
SELECT Nome, Sexo FROM Empregado WHERE Num-Dep = 15 AND Salário > 1000;

9.1.2. Operadores BETWEEN e NOT BETWEEN

Esses operadores substituem o uso dos operadores <= e >=:
... WHERE <nome atributo> BETWEEN <valor1> and <valor2>

Exemplo:

Listar os empregados com salário **entre** R$ 1.000,00 e R$ 2.000,00

SELECT * FROM Empregado WHERE Salário BETWEEN 1000 and 2000;

Observação: A palavra between vem do inglês e significa entre.

9.1.3. Operadores LIKE e NOT LIKE

Esses operadores só se aplicam sobre atributos do tipo char.
Operam como = (igual) e != (diferente), utilizando os símbolos: % (substitui uma palavra) e _ (substitui um caracter):
...WHERE <nome atributo> LIKE <valor1>

Exemplo:

Listar os empregados que tem como primeiro nome 'José':

SELECT Nome FROM Empregado WHERE Nome LIKE 'José%';

9.1.4. Operadores IN e NOT IN

Os operadores IN e NOT IN procuram dados que estão ou não contidos em um dado conjunto de valores:
... WHERE <nome atributo> IN <valores>

Exemplo:

Listar o nome e data de nascimento dos dependentes com grau de parentesco 'M' ou 'P':

SELECT Nome, Data-Nasc FROM Dependentes WHERE Grau-P IN ('M', 'P');

9.1.5. Operadores IS NULL e IS NOT NULL

Os operadores IS NULL e IS NOT NULL identificam valores nulos dos atributos:
... WHERE <nome atributo> IS NULL

Exemplo:

Listar os projetos que não tenham local definido:

SELECT * FROM Projeto WHERE Local IS NULL;

9.1.6. Ordenando os Dados

Para ordenar (ORDER BY) os dados que foram resultados de "selects" feitos no banco utilizamos a seguinte sintaxe:

SELECT <lista atributos> FROM <tabela> [WHERE <condição>] ORDER BY <Nome atributo> {ASC | DESC}

Observação:

- ASC: Ordem ascendente.
- DESC: Ordem descendente.

Exemplos:

Listar todos os empregados ordenados ascendentemente por nome:

SELECT * FROM Empregado ORDER BY Nome;

Listar todos os empregados ordenados descendentemente por salário:

SELECT * FROM Empregado ORDER BY Salário DESC;

9.1.7. Realizando cálculo com informação selecionada

Pode-se criar um campo que não pertença à tabela a partir de cálculos sobre atributos da tabela.

Exemplo:

Mostrar o novo salário dos empregados calculado com base no reajuste de 60% para os que ganham abaixo de R$ 1.000,00:

SELECT Nome, Novo-salário = (Salário * 1.60) FROM Empregado WHERE Salário < 1000;

9.1.8. Funções sobre conjuntos

Para realizarmos operações sobre conjuntos temos os seguintes comandos: MAX, MIN, SUM, AVG, COUNT.

Exemplos:

Mostrar o valor do maior salário dos empregados:

SELECT MAX (Salário) FROM Empregado;

Mostrar qual o salário médio dos empregados:

SELECT AVG (Salário) FROM Empregado;

Quantos empregados ganham mais de R$1.000,00?
SELECT COUNT (*) FROM Empregado WHERE Salário > 1000;

9.1.9. Cláusula DISTINCT

A cláusula DISTINCT elimina tuplas duplicadas do resultado de uma consulta.

Exemplo:

Quais os diferentes salários dos empregados?

SELECT DISTINCT Salário FROM Empregado;

9.1.10. GROUP BY

Para agrupar informações selecionadas usamos: GROUP BY.

Ele organiza a seleção de dados em grupos.

Exemplo:

Listar os empregados por sexo:

SELECT Nome, Sexo FROM Empregado GROUP BY Sexo;

9.1.11. HAVING

HAVING agrupa informações de forma condicional. Seleciona entre as tuplas resultantes, as que satisfazem uma dada condição.

Exemplo:

Listar o número total de empregados que recebem salários superiores a R$1.000,00, agrupados por departamento, mas só daqueles com mais de 5 empregados:

SELECT Num-Dep, COUNT (*) FROM Empregado WHERE Salário > 1000 GROUP BY Num-Dep HAVING COUNT(*) > 5;

9.1.12. JOIN

JOIN recupera dados de várias tabelas. Cita as tabelas envolvidas na cláusula FROM. Utiliza qualificadores de nomes para referenciar o nome do empregado, Empregado.Nome.

Exemplos:

Listar o nome do empregado e do departamento onde está alocado:

SELECT Empregado.Nome, Departamento.Nome FROM Empregado, Departamento

WHERE Empregado.Num-Dep = Departamento.Numero;

Listar os nomes dos departamentos que têm projetos:

SELECT Departamento.Nome FROM Departamento, Projeto

WHERE Projeto.Num-Dep = Departamento.Numero;

Observação: Podemos utilizar as cláusulas (NOT) LIKE, (NOT) IN, IS (NOT) NULL misturadas aos operadores AND, OR e NOT nas equações de junção.

Exemplo:
Listar os departamentos que têm projetos com número superior a 99 e localizados em RJ ou SP ordenados por nome de departamento:

SELECT Departamento.Nome FROM Departamento, Projeto WHERE Projeto.Local IN ('RJ', 'SP') AND Projeto.Número > 99 AND Projeto.Num-Dep = Departamento.Codigo
ORDER BY Departamento.Nome;

9.1.13. Juntando mais de duas tabelas

Observe abaixo um exemplo onde fazemos a junção de mais de duas tabelas.

Exemplo:

Listar o nome dos empregados, com seu respectivo departamento que trabalhem mais de 20 horas em algum projeto:

SELECT Empregado.Nome, Departamento.Nome FROM Empregado, Departamento, Trabalha-em
WHERE Trabalha-em.Horas > 20 AND Trabalha-em.Cad-Emp = Empregado.Cad AND Empregado.Num-Dep = Departamento.Número;

9.1.14. Utilizando consultas encadeadas

O resultado de uma consulta é utilizado por outra consulta, de forma encadeada e no mesmo comando SQL.

Exemplo:

Listar os departamentos que tenham qualquer projeto em RJ:

SELECT Departamento.Nome FROM Departamento WHERE Departamento.Número IN (SELECT Projeto.Num-Dep FROM Projeto WHERE Projeto.Local = 'RJ');

9.1.15. Adicionando tupla à tabela

Como já vimos em aulas passadas, para adicionar tupla à tabela usamos o comando INSERT. Veja a sintaxe:
INSERT INTO <tabela> (<lista de atributos>) VALUES (<valores>);

Exemplo:

INSERT INTO Empregado(Cad, Nome, Sexo, Salário, Num_Dep, Cad_Supv) VALUES (015, 'José da Silva', 'M', 1000, 1, 020);

9.1.16. Adicionando tuplas usando SELECT

Para adicionar tuplas à tabela usando SELECT usamos a seguinte sintaxe:
INSERT INTO <tabela> (<lista de atributos>) SELECT <lista de atributos> FROM <tabela> WHERE <condição>;

Exemplo:

INSERT INTO depto-info (**nome-depto**, **numemp**, **total-sal**) SELECT **D.nome**, **COUNT(*)**, **SUM (E.salario)** FROM Departamento D, Empregado E WHERE D.numero = E.Num-Dep GROUP BY D.nome HAVING COUNT (*) > 50;

9.1.17. Criando Visões (VIEW)

São tabelas virtuais que não ocupam espaço físico.

Exemplo:
Criar uma visão dos empregados do departamento 10 que tenham mais de 20 horas de trabalho em projetos:
CREATE VIEW Dep-10 AS SELECT Empregado.Nome, Trabalha-em.Num-Proj FROM Empregado, Trabalha-em WHERE Trabalha-em.Horas > 20 AND Trabalha-em.Cad-Emp = Empregado.Cad AND Empregado.Num-Dep = 10;

9.1.18. Privilégios

Para garantir privilégios de acesso usamos GRANT e para revogar privilégios usamos REVOKE:

- GRANT <privilégios> ON <nome tabela/view> TO <usuário>
- REVOKE <privilégios> ON <nome tabela/view> FROM <usuário>

Onde <privilégios>: SELECT, INSERT, DELETE, UPDATE, ALL PRIVILEGES e

<usuário>: usuário cadastrado, PUBLIC

Exemplos:
Garante privilégios ao empregado Carlos para realizar "selects" no banco:

GRANT SELECT ON Empregado TO carlos;

Revoga os privilégios de "selects" na tabela Projeto para todos os usuários que não tem privilégios de um grupo específico:

REVOKE SELECT ON Projeto FROM PUBLIC;

10. A importância da Modelagem de Dados

Para desenvolvermos aplicações que usam banco de dados deveremos possuir os conceitos básicos sobre modelagem de dados. Não importa se sua aplicação é muito simples; a correta modelagem dos seus dados irá com certeza tornar sua aplicação mais robusta e mais fácil de manter.

Alguns motivos que tornam a modelagem de dados fundamental:

- É o projeto conceitual da base de dados;
- É o alicerce da construção do sistema de informação;
- Representa o ambiente observado;
- Fornece processos de validação;
- Observa processos de relacionamentos entre objetos;
- É a documentação do projeto das bases de dados.

Com isso evitamos:

- A redundância de dados;
- A baixa performance do acesso aos dados (índices e chaves mal elaborados);
- O mau funcionamento da programação x acesso aos dados.

E facilitaremos:

- A manutenção das bases (novas tabelas e campos);
- Inserção de novos negócios (customizações);

11. Modelo de Entidade-Relacionamento (MER)

O Modelo de Entidade-Relacionamento (MER) foi desenvolvido para auxiliar o **projeto de banco de dados**, através da especificação de um esquema que define a organização da base de dados.

Por definição temos, Modelo baseado na percepção do mundo real, que consiste em um conjunto de objetos básicos chamados **entidades** e nos **relacionamentos** entre esses objetos.

11.1. Entidade e Atributos

Entidade pode ser definida como um objeto do mundo real, concreto ou abstrato e que possui existência independente. Cada entidade possui um conjunto particular de propriedades que a descreve chamado "**atributos**". Um atributo pode ser dividido em diversas sub-partes com significado independente entre si, recebendo o nome de "atributo composto". Um atributo que não pode ser subdividido é chamado de "atributo simples" ou "atômico".

Os atributos que podem assumir apenas um determinado valor em uma determinada instância são denominados "atributo simplesmente valorado", enquanto que um atributo que pode assumir diversos valores em uma mesma instância é denominado "multi valorado".

Um atributo que é gerado a partir de outro atributo é chamado de "atributo derivado".

Entre os diversos atributos que definem uma entidade deve existir um ou mais campos que identifiquem inequivocamente cada registro. A este(s) atributo(s) dá-se o nome de Atributo *Identificador*.

Chave Primária

É um atributo identificador que representa univocamente cada ocorrência ou registro de uma tabela.

Existem dois tipos de chave primária:

- **Simples** - constituída apenas por um atributo;
- **Composta** - constituída por dois ou mais atributos.

Propriedades da Chave primária

Uma chave primária deve ser:

- **Unívoca** – O valor da chave primária deve ser único para todos os registros.
- **Não Redundante** – No caso de uma chave composta não devem ser incluídos mais campos do que os necessários.
- **Não Nula** – Nenhum dos valores que compõem a chave primária pode conter valores nulos.

Chave Estrangeira ou Externa

É um atributo que definido como chave primária de uma tabela é incluído na estrutura de uma outra tabela.

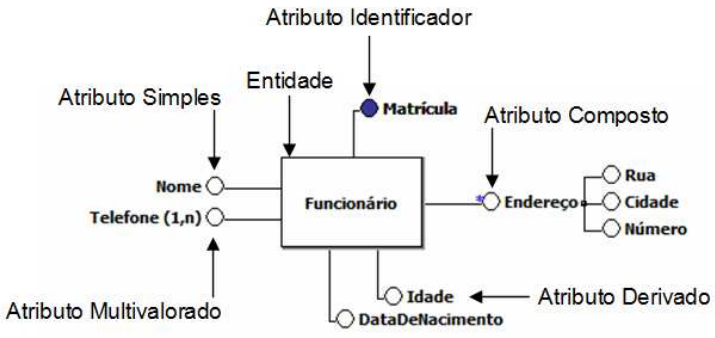
Exemplo: Consideremos as entidades Cd´s e Faixas, que identificam um Cd e as suas respectivas Faixas.

CD´s (CódigoCD, Título, Intérprete, Gravadora)
Faixas (CódigoCd, CódigoFaixa, Título, Duração)

O atributo CódigoCd da entidade Faixas faz parte da sua chave primária, no entanto como é chave primária da entidade CD´s é considerado uma chave estrangeira na entidade Faixas.

Observação: Os atributos chaves, nessa forma de representação da entidade Faixas, são os atributos sublinhados.

Exemplo:



Relacionamento

Conjunto de associações do mesmo tipo entre ocorrências de entidades.

Exemplo: Todos os clientes de um banco têm contas = Relacionamento **Tem**

Conjunto de associações entre ocorrências de entidades sobre as quais se deseja manter informações no BD.

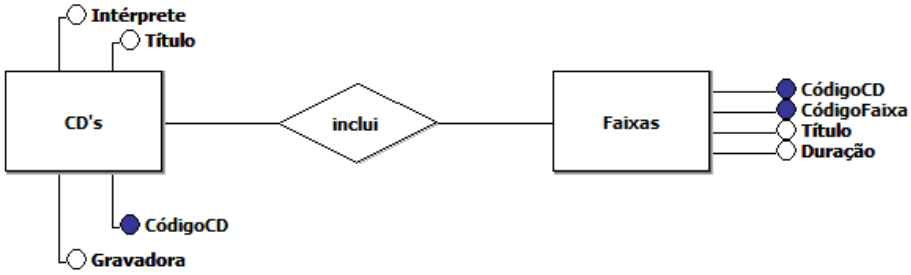
Relacionamento x Instância de Relacionamento:

- Para referir um relacionamento particular fala-se em instância ou ocorrência de relacionamento.
- Tem autores que consideram uma associação específica como um relacionamento e o conjunto de associações como conjunto de relacionamentos.

Relacionamento e a Chave

O relacionamento entre entidades é um dos propósitos das bases de dados relacionais, daí a importância dada à seleção da chave primária (vista na aula anterior), pois é através destas que são estabelecidas as associações entre as diferentes entidades.

Exemplos:

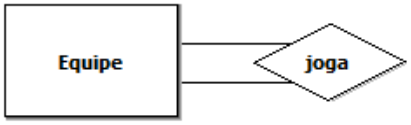


Tipos de Relacionamentos

São as formas como as entidades se relacionam num determinado modelo de informação. As associações podem classificar-se em: unárias, binárias e complexas.

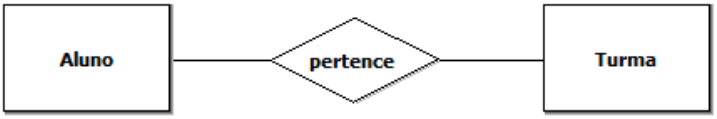
Unário e Binário

Relacionamento unário associa uma entidade com ela própria.



Neste caso, uma equipe joga com outra equipe.

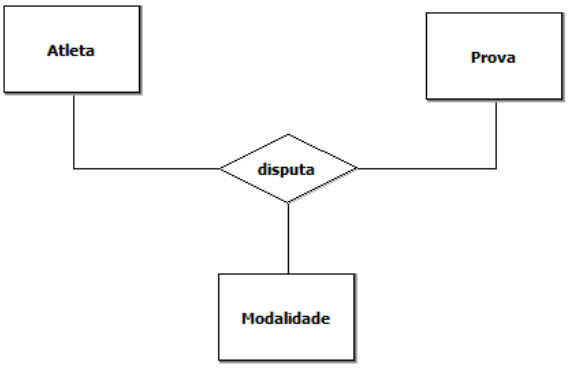
Relacionamento binário associa duas entidades.



Neste tipo de relacionamento, um aluno pertence a uma turma.

Complexos

Relacionamento complexo associa mais do que duas entidades.



Tem-se que um atleta que pratica uma determinada modalidade disputa uma prova dessa modalidade.

Observação: **Regra geral, uma associação do tipo complexa ternária é obrigatória entre as três entidades.**

Cardinalidade

Determina a quantidade (mínima e máxima) de ocorrências de relacionamentos que uma instância de entidade pode ter com outras instâncias de entidades.
Exemplo:

Cardinalidade Mínima de um Relacionamento

Determina a quantidade mínima de ocorrências de relacionamentos que uma instância de entidade pode ter com outras instâncias de entidades.
Indica se a participação das ocorrências de entidades no relacionamento é obrigatória ou opcional.

Valores válidos:

- Min = 0 → relacionamento opcional ou parcial
- Min > 0 → relacionamento obrigatório ou total
- Min ≤ Max

Para efeito prático apenas duas cardinalidades mínimas são relevantes, 0 (zero) e 1 (um).

Cardinalidade Máxima de um Relacionamento

Determina a quantidade máxima de ocorrências de relacionamentos que uma instância de entidade pode ter com outras instâncias de entidades.

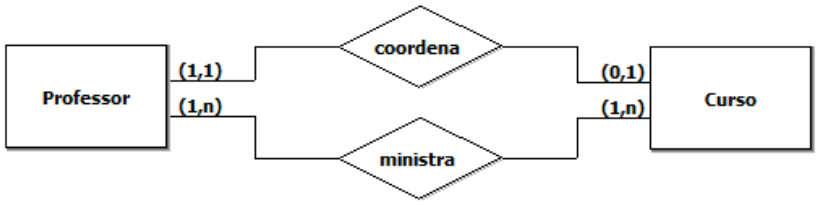
Valores válidos:

- Max > 0
- Max ≥ Min

Para efeito prático apenas duas cardinalidades máximas são relevantes, 1 e n.

Vários Relacionamentos

Às vezes, pode existir mais de um relacionamento entre as mesmas entidades:



Observação: Pode-se nomear relacionamentos a partir dos nomes das entidades envolvidas.

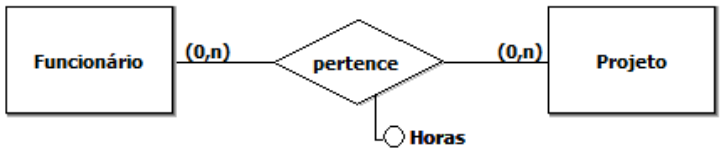
Exemplo: cliente_conta

Atributos do Relacionamento

Quando um determinado relacionamento possui atributos, também conhecido como relacionamento valorado. Esta situação ocorre apenas em relacionamento N:N.

Ex. Pedro trabalha no projeto Alfa 30 horas.

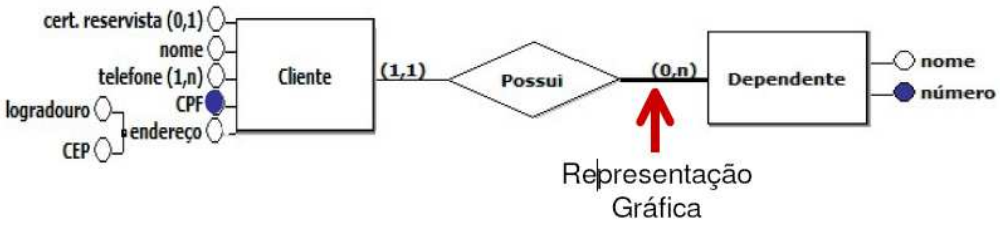
- Pedro - Elemento do conjunto de valores do atributo Nome da entidade Funcionário.
- Alfa - Elemento do conjunto de valores do atributo Nome do Projeto da entidade Projeto.
- trabalha - Ligação existente entre um funcionário e um projeto. Neste caso, este funcionário trabalha 30 horas neste projeto, porém este mesmo funcionário poderá trabalhar outro número de horas em outro projeto, assim como outro funcionário trabalha outro número de horas no mesmo projeto Alfa. Podemos concluir que 30 horas é o atributo que pertence ao Pedro no projeto Alfa.



Relacionamento Identificador (Entidade Fraca)

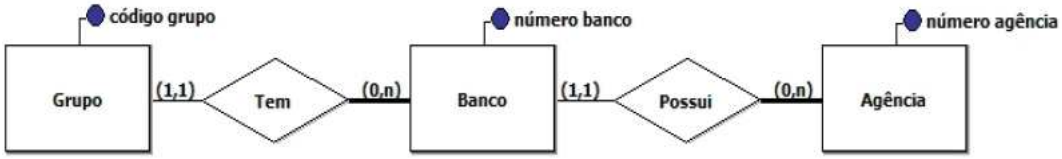
A entidade não tem atributos suficientes para formar seu identificador, daí usa o identificador da entidade Forte para formar o seu.

A entidade Dependente só existe quando está relacionada a outra entidade, ou seja, se a entidade Cliente for eliminada do banco a entidade também será eliminada.



Relacionamento Identificador (Recursão)

O termo Entidade Fraca deve ser usado com cautela, pois uma entidade fraca em um relacionamento não necessariamente é também fraca em outro relacionamento, observe:



Auto-Relacionamento

Representa uma associação entre ocorrências de uma mesma entidade.
Exige a identificação de papéis.

- Um Cliente pode ser patrocinador de vários de clientes.
- Um Cliente só pode ser patrocinado por um Cliente.

Grau de relacionamento

Corresponde ao o número de entidades, não necessariamente distintas, que participam de um relacionamento.

Uma entidade pode ser tratada como um relacionamento de grau zero para efeito de comparação com outros relacionamentos.

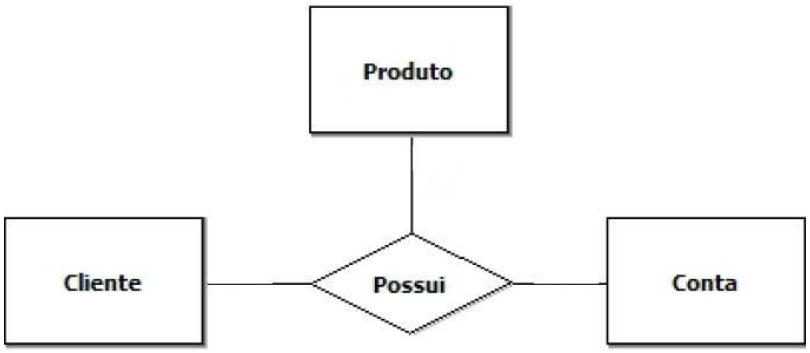
Tipos de Grau de Relacionamento

Binário: Uma ocorrência do relacionamento deve envolver simultaneamente duas instâncias de entidades

Ternário: Uma ocorrência do relacionamento deve envolver simultaneamente três instâncias de entidades. Não pode relacionar três entidades em um momento e duas em outro.

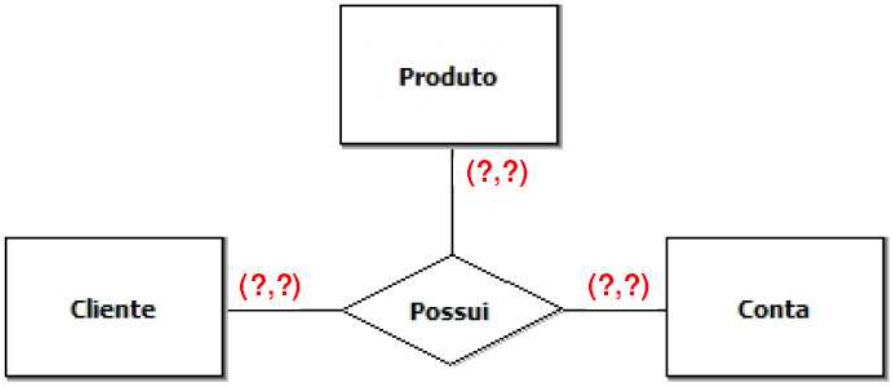
Relacionamento Ternário

Cada ocorrência de "Possui" relaciona 3 ocorrências de entidade: Cliente, Produto e Conta!



Relacionamento Ternário - Cardinalidade

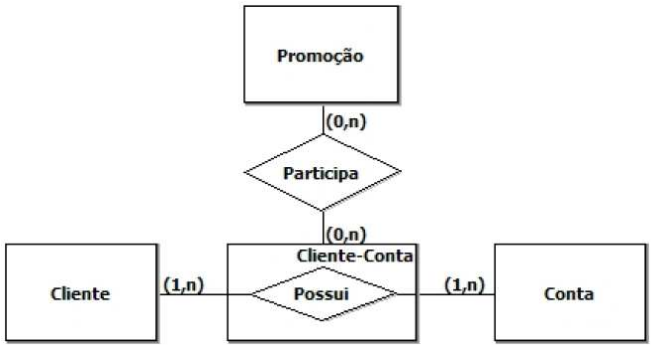
Pense na cardinalidade que prevista para a modelagem abaixo?



Atenção: A cardinalidade refere-se ao par das demais entidades.

Entidade Associativa (ou Agregação)

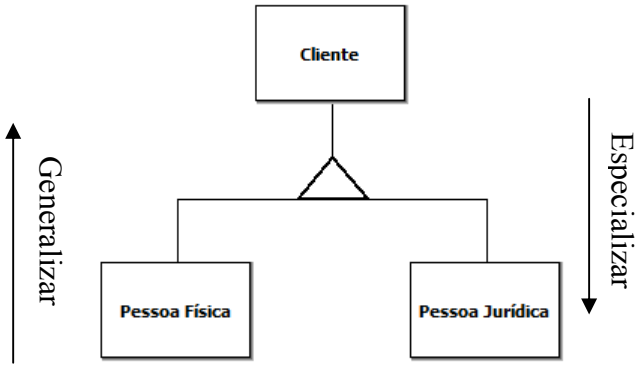
Substitui a associação entre relacionamentos, a qual não é prevista pelo Modelo ER. É um relacionamento que passa a ser tratado como entidade. Permite o uso de relacionamento opcional!



Generalização/Especialização

Permite atribuir propriedades particulares a um subconjunto de entidades especializadas. Permite a herança de propriedades (atributos):

- Agrega ao seu conjunto de propriedades as propriedades da entidade genérica.



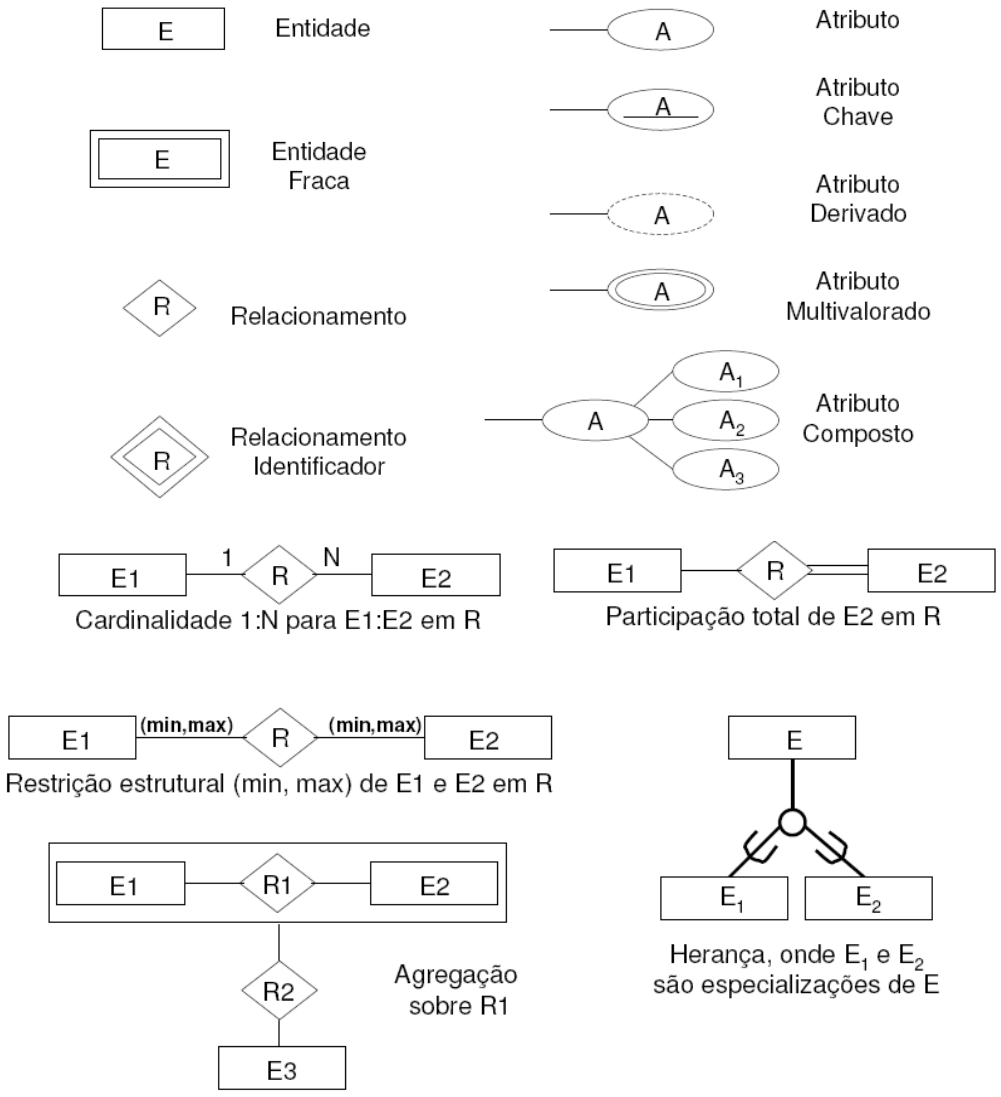
Tipos de Generalização/Especialização

Abaixo são descritas 4 tipos de generalização/especialização que podem ocorrer entre entidades:

1. **Total**: Todas as entidades especializadas têm que estar relacionada à entidade genérica.
2. **Parcial**: Pelo menos uma entidade genérica não está relacionada à entidade especializada.
3. **Exclusiva**: A entidade genérica está associada a uma única entidade especializada.
4. **Não exclusiva:** A entidade genérica está associada a duas ou mais entidades especializadas.

Outros tipos de notação gráfica

A notação gráfica mostrada abaixo também é bastante usada no dia-a-dia, mas não é a de Peter Chen que é a que vimos em todos os exemplos anteriores, mas sim a de R. Elmasri & S. Navathe.



Ferramentas para modelagem

Você pode utilizar algumas das ferramentas abaixo para realizar a modelagem dos dados:

- **PowerDesigner;**
- **Erwin;**
- **DBDesigner;**
- **Microsoft Visio;**
- **brModelo**

12. O Modelo Relacional

Criado por Codd, nos anos 70, começou a ser realmente utilizado nas empresas a partir de 1987, através do SGBDs. A abordagem relacional está baseada no princípio de que as informações em uma base de dados podem ser consideradas como relações matemáticas e que estão representadas de maneira uniforme, através do uso de tabelas, ou falando de uma forma mais direta, um arquivo. Porém, um arquivo é mais restrito que uma tabela. Toda tabela pode ser considerada um arquivo, porém, nem todo arquivo pode ser considerado uma tabela.

Este princípio coloca os dados (entidades e relacionamentos) dirigidos para estruturas mais simples de armazenar dados, que são as tabelas. O conceito principal vem da teoria dos conjuntos (álgebra relacional) atrelado à idéia de que não é relevante ao usuário saber onde os dados estão nem como os dados estão (transparência). Os usuários manipulam objetos dispostos em linhas e colunas das tabelas, como mostra a _gura a seguir. O usuário pode lidar com estes objetos, conta com um conjunto de operadores e funções de alto nível, constantes na álgebra relacional.

Terminologia do modelo relacional:

- **Tabela** é chamada de **RELAÇÃO**;
- **Linha** de uma tabela é chamada de **TUPLA**;
- **Coluna** é chamado de **ATRIBUTO**;
    - O tipo de dado que descreve cada coluna é chamado de **DOMÍNIO**.

12.1. Conceitos Gerais: Relação

Dados os conjuntos S1, S2 , ..., Sn, não necessariamente distintos, diz-se que R é uma relação sobre esses n conjuntos se R é um conjunto de m tuplas, nas quais o primeiro elemento assume valores em S1, o segundo em S2 , e assim por diante.

12.2. Conceitos Gerais: Atributos

Dada uma relação R, define-se como seus atributos os nomes das funções que mapeiam os valores de cada um dos elementos de cada tupla nos respectivos conjuntos S1, S2 , ..., Sn.

12.3. Conceitos Gerais: Domínios

Dada uma relação R, o domínio do atributo Aj, dom (Aj) , é o conjunto Sj no qual o atributo assume valores.

- Todo domínio possui uma descrição física e outra semântica.
- A descrição física serve para identificar o tipo e o formato dos valores que compõem o domínio.
    - exemplo: char(13), "(dd)dddd-dddd"
- A descrição semântica serve para ajudar na interpretação de seus valores.
    - exemplo: "Números de telefone do Estado do RJ"

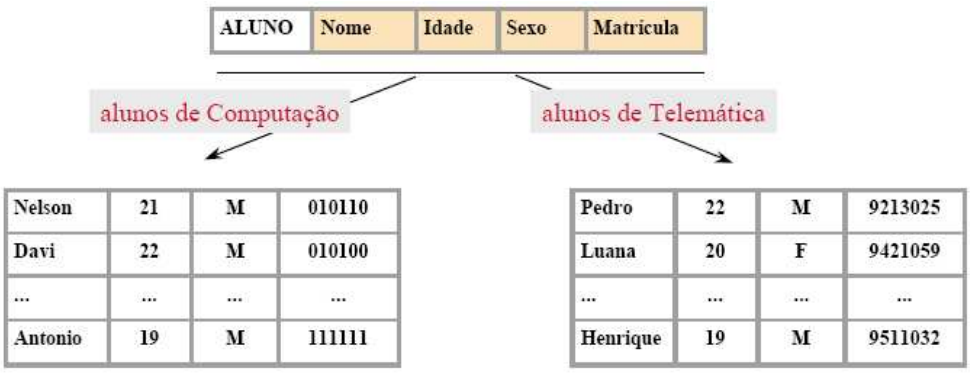
12.4. Conceitos Gerais: Esquemas de uma Relação

O esquema de uma relação R, denotado por R (A1:D1, A2:D2 , ..., Am:Dm) , é formado pelo nome da relação e pela lista de seus atributos e respectivos domínios.

Usado para descrever uma **relação** através da especificação de seus atributos e domínios. Formado pelo nome da relação e uma lista de atributos A**1**, A**2**, ..., A**n,** onde A**j** é o nome do papel desempenhado pelo domínio D**j** no esquema da relação R.
Também chamado de **intenção** da relação R.

12.5. Conceitos Gerais: Intenção X Extensão

Para uma mesma intenção pode existir mais de uma extensão



13. Banco de Dados Relacional

O universo de um banco de dados relacional é um conjunto finito, não vazio, de relações.
O esquema de um banco de dados relacional é o conjunto dos esquemas das relações que o formam, isto é:

$$\begin{array}{ll} R_1 & (A_{11}, A_{12}, \ldots\ldots, A_{1n}) \\ R_2 & (A_{21}, A_{22}, \ldots\ldots, A_{2n}) \\ \ldots & \\ R_m & (A_{m1}, A_{m2}, \ldots\ldots, A_{mn}) \end{array}$$

Uma instância de um banco de dados relacional é o conjunto das instâncias de suas relações.
O mesmo esquema pode se aplicar a diferentes instâncias de um banco de dados.

13.1. Características das Relações

- Não existe ordem entre as tuplas de uma relação;

- Desde que seja mantida a correspondência entre atributos e valores, não existe ordem entre os atributos de uma relação.

- Todos os atributos de uma relação devem ser atômicos, isto é, indivisíveis em termos de valores e componentes.
- O esquema de uma relação pode ser interpretado como uma declaração ou um tipo de assertiva.

- Cada tupla da relação pode ser interpretada como um fato ou uma instância particular dessa assertiva.

- Numa relação, todos os atributos devem ter nomes distintos**:** Conceitualmente, não existe nada que impeça dois atributos de terem o mesmo nome. A restrição é apenas de ordem prática, para facilitar a consulta ao banco de dados. Mesmo que pudessem ter nomes iguais poderiam assumir valores em domínios diferentes.

- Nenhuma relação possui atributos em duplicata. Essa restrição decorre não apenas do fato de uma relação ser um conjunto, nos entidomatemático do termo, como também do fato de suas tuplas representarem uma assertiva sobre o mundo real. Não faz sentido representar o mesmo valor duas vezes no mesmo fato. Ex: Existe um ALUNO de Nome “Antonio ”, com “19 ” anos de Idade, do Sexo “ M ”, cujo nome é “Antonio”.

13.2. Restrições de Integridade

- **Restrições de Domínio** – especificadas através do tipo de dados de cada atributo do esquema de uma relação.

- **Restrições de Chave e de Integridade de Entidade** – especificadas através da definição de uma chave de acesso em cada relação.

- **Restrições de Integridade Referencial** – especificadas através de regras de relacionamento entre os esquemas das relações que compõem o esquema do banco de dados.

- **Restrições Semânticas** – especificadas através de regras sobre os esquemas do banco de dados.

13.3. Restrições de Domínio

Especifica que o valor de cada atributo A deve ser um elemento atômico do domínio de A. Em geral, é especificado através de tipos primitivos de dados, tais como integer, float, char, date, time, money, etc. Também podem ser descritos através da definição de subconjuntos de tipos primitivos ou de listas enumeradas.

13.4. Restrições de Chave

O valor de um atributo-chave pode ser utilizado para identificar uma tupla específica numa relação.

Uma chave é uma propriedade do esquema de uma relação, isto é, uma propriedade que deve ser respeitada por todas as instâncias da relação.

Atributos cujos valores podem ser duplicados, não devem ser definidos como chaves de uma relação (Nome, por exemplo).

Em geral, uma relação pode ter mais de uma chave. Nesse caso, cada chave da relação é chamada de CHAVE CANDIDATA.

A chave candidata escolhida para identificar as tuplas de uma relação é chamada de CHAVE PRIMÁRIA. Em geral, entre todas as chaves candidatas, escolhe-se para chave primária aquela com o menor número de atributos.

13.5. Integridade de Entidade

Estabelece que nenhum d o s atributos pertencentes a chave-primária de uma relação pode ter valor nulo.

Isso é para garantir a identidade individual das tuplas de uma relação, uma vez que a chave - primária é utilizada para identificar cada uma de suas tuplas.

13.6. Integridade Referencial

Estabelece que qualquer tupla pertencente a uma relação R1 que referencie uma outra relação R2, tem de necessariamente referenciar uma tupla existente em R2.

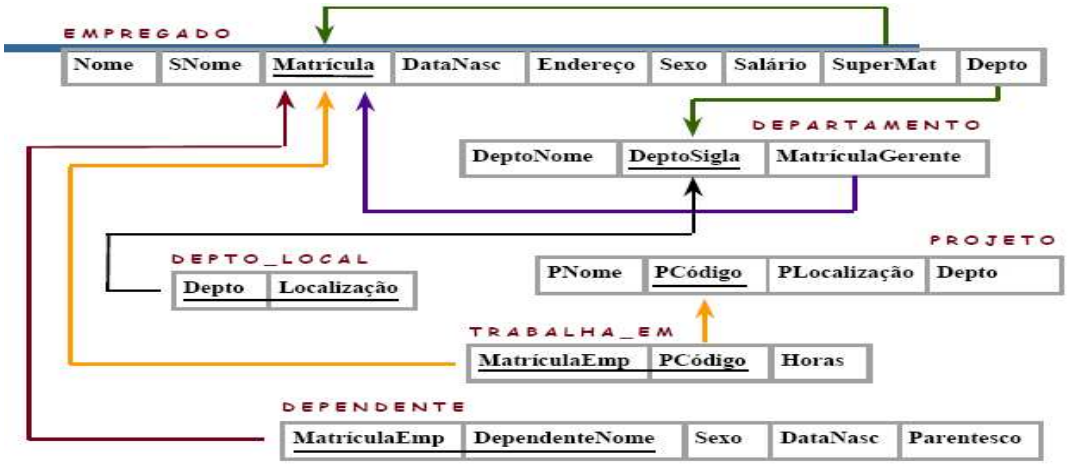
13.7. Chave Estrangeira

Seja FK um conjunto de atributos de um esquema de relação R 1 . Diz-se que FK é uma chave-estrangeira de R1s e FK satisfizer as seguintes condições:

- Os atributos pertencentes a FK assumirem valores dos mesmos domínios correspondentes aos atributos PK da chave-primária de um outro esquema de relação R2 ; e
- O v a l o r d e F K , em qualquer tupla de R 1 , for nulo ou igual ao valor de PK, em alguma tupla de R2 .

Utilizadas para manter a consistência entre as tuplas de duas relações distintas e relacionadas entre si.
Decorrem tipicamente dos relacionamentos entre entidades definidos no modelo conceitual do banco de dados.

Exemplo:



13.8. Restrições Semânticas

Especificadas através de regras sobre o esquema do banco de dados.

Exemplos:

- o salário de um empregado tem de ser menor ou igual ao salário de seu supervisor.
- o número de horas semanais que um empregado pode trabalharem projetos tem de ser menor ou igual a 56 .

13.9. Notação Resumida

Exemplos:

Departamento (deptonome, deptosigla, matriculagerente)
matriculagerente **referencia** empregado

Trabalha_em (matriculaemp, pcodigo, horas)
matriculaemp **referencia** empregado
pcodigo **referencia** projeto

14. Modelo ER x Modelo Relacional

O modelo ER é um Modelo Formal

Um Diagrama Entidade-Relacionamento (DER) é um modelo formal, preciso, não ambíguo. Isto significa que diferentes leitores de um mesmo DER devem sempre entender exatamente o mesmo. Tanto é assim, que um DER pode ser usado como entrada a uma ferramenta CASE na geração de um banco de dados relacional. Por isso, é de fundamental importância que todos os envolvidos na confecção e uso de diagramas ER estejam treinados na sua perfeita compreensão.

Sub-utilização do modelo ER

Observa-se em certas organizações, que modelos ER são sub-utilizados, servindo apenas como ferramenta para apresentação informal de idéias. Isso pode ser evitado com treinamento formal de todos envolvidos na modelagem e projeto do banco de dados.

A abordagem ER tem poder de expressão limitado

Em um modelo ER, são apresentadas apenas algumas propriedades de um banco de dados. Em realidade, a linguagem dos DER é uma linguagem muito pouco poderosa e muitas propriedades desejáveis do banco de dados necessitam ser anotadas adicionalmente ao DER.

Entendendo um pouco a transformação

A transformação de um relacionamento n:n em entidade segue o seguinte processo:

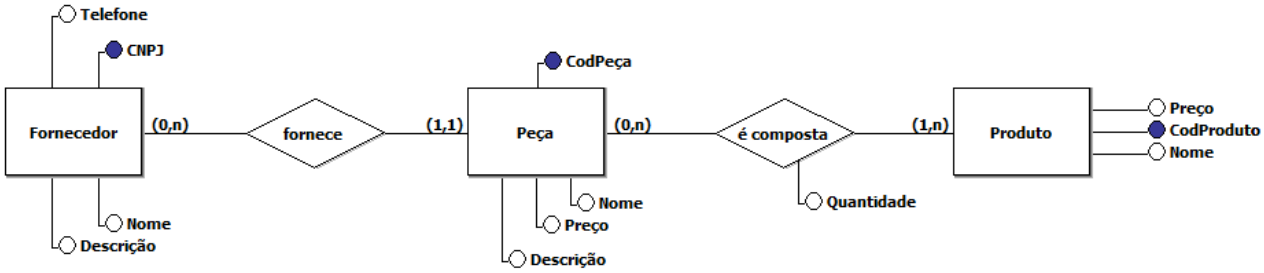
- O relacionamento n:n é representado como uma entidade.
- A entidade criada é relacionada às entidades que originalmente participavam do relacionamento.
- A entidade criada tem como identificador:
  - As entidades que originalmente participavam do relacionamento;
  - Os atributos que eram identificadores do relacionamento original (caso o relacionamento original tivesse atributos identificadores).
- As cardinalidades da entidade criada nos relacionamentos de que participa é sempre (1,1).
- As cardinalidades das entidades que eram originalmente associadas pelo relacionamento transformado em entidade são transcritas ao novo modelo conforme mostrado na figura da página anterior.

Modelo Relacional

No modelo Lógico Relacional as entidades e os relacionamentos são trabalhados na forma de tabelas ou relações, que são formadas por um conjunto de tuplas e atributos. A terminologia tabela é mais comum nos produtos comerciais e na prática. Já a terminologia relação foi utilizada na literatura original sobre a abordagem relacional (daí a denominação "relacional") e é mais comum na área acadêmica e nos livros texto.

Modelo ER e Modelo Relacional

Seja o Modelo ER de um indústria de automóveis:



Teríamos então as seguintes Tabelas no Modelo Relacional:

PEÇA(**CodPeça**, Descrição, Nome, Preço)
FORNECEDOR(**CNPJ**, Nome, Telefone, **CodPeça**)
PRODUTO(**CodProduto**, Nome, Preço)
COMPOSICAO(**CodProduto , CodPeça** , Quantidade)

15. Mapeamento do Modelo ER para o Modelo Relacional

Vamos mostrar o que observar e como fazer o mapeamento de um modelo Entidade-Relacionamentos (Modelo Conceitual) para um modelo Relacional (Modelo Lógico).

As regras foram definidas tendo em vista dois objetivos básicos:

- **Bom desempenho:**
  - Bom desempenho significa basicamente diminuir acesso ao disco
    - Acessos ao disco consomem o maior tempo na execução de uma instrução de banco de dados
- **Simplificação do desenvolvimento e da aplicação de BD.**

15.1. Princípios das regras de mapeamento

As regras foram definidas tendo por base, entre outros, os seguintes princípios:

- Evitar junções;
- Diminuir o número de chaves;
- Evitar campos opcionais.

15.1.1. Evitar junções

Junção é uma operação que busca dados de diversas linhas através da igualdade de campos.

SGBD Relacional geralmente armazenam dados continuamente no disco:

- Um único acesso ao disco traz todos os dados do "bloco" para a memória RAM.

Junções envolvem comparações entre diversas linhas:

- Junções requerem diversos acessos a disco (minimizam o desempenho).

Assim, quando possível, evite usar junções!
Dica para evitar Junções:

- Ter os dados necessários ao resultado da consulta em uma única linha da tabela.

15.1.2. Diminuir o número de chaves

Para implementar eficientemente o controle de chaves primárias o SGBD usa índices para cada chave primária.

- Índices tendem a ocupar espaço considerável em disco.
- A inserção ou remoção de entradas em um índice podem exigir diversos acessos a disco.

Assim, quando possível, diminua a quantidade de chaves primárias!

Dica para diminuir chaves:

- Ter os dados subordinados as chaves primárias em uma única tabela.

15.1.3. Evitar Campos Opcionais

Campo opcional = campo que pode ser VAZIO (NULL).

- SGBD Relacional não desperdiçam espaço pelo fato de campos de uma linha estarem vazios (usam técnicas de compressão de dados).
- Campo opcional não tem influência na performance.

Problema: Quando o SGBD Relacional não controla a obrigatoriedade do campo (quando o preenchimento de um campo depende de outro)

- Exemplo: Se a pessoa for casada deve preencher Campo X, Y e Z.
- Neste caso, o controle da obrigatoriedade deve ser feito por programação.

Assim, quando possível, evite campos opcionais!

Dica para evitar campos opcionais:

- Ter apenas campos obrigatórios Esta regra é mais "fraca" do que as precedentes

15.2. Regras para transformação do ER para o relacional

Passos da transformação ER para Relacional:

- Tradução inicial de entidades e respectivos atributos;
- Tradução de relacionamentos e respectivos atributos;
- Tradução de generalizações/especializações.

15.2.1. Implementação inicial de entidades

Cada **entidade** é traduzida para uma **tabela.**
Cada **atributo** da entidade define uma **coluna** desta tabela.
Atributos identificadores da entidade correspondem à **chave primária** da tabela.

Esta é uma tradução inicial:

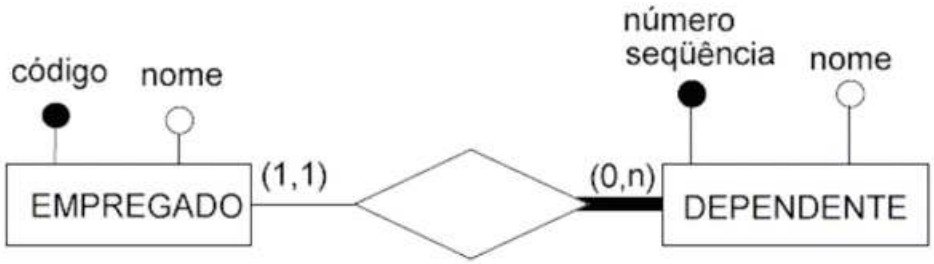
- As próximas regras podem fundir tabelas.

Exemplo:
Modelo ER:

Modelo Relacional:
Pessoa (Codigo,Nome,Endereço,DtNasc,DtAdm)

15.2.2. Tradução de Entidade Fraca

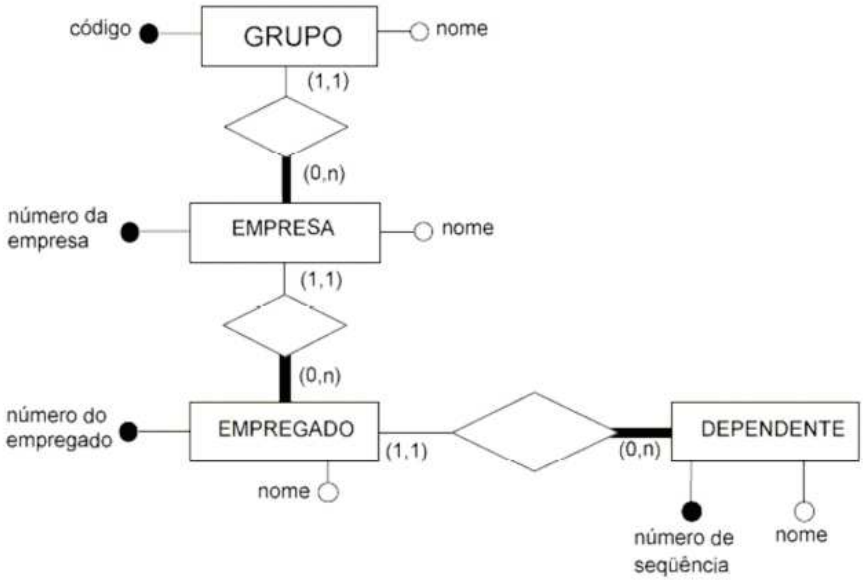
Faz-se a tradução das entidades fracas em tabelas, seus atributos em campos e adiciona-se a chave primária da entidade forte na chave primária da entidade fraca.



Dependente (Codigo, NoSeq, Nome)

Tradução de entidade fraca-recursão:

- Semelhante anterior.
- Entretanto faz-se sucessivamente!



Grupo (CodGrup, Nome)
Empresa (CodGrup, NoEmpresa, Nome)
Empregado (CodGrup, NoEmpresa, NoEmpreg, Nome)
Dependente (CodGrup, NoEmpresa, NoEmpreg, NoSeq, Nome)

15.2.3. Nomes de colunas

São freqüentemente referenciados em programas:

- Use nomes das colunas que sejam curtos e significativos. Isto facilita a vida do programador.

No SGBDR os nomes das colunas não podem conter brancos.

Não transcreva os nomes de atributos para nomes de colunas.

- Abrevie os nomes de atributos compostos de diversas palavras
- Exemplo: DtNasc, EstCivil, CartMotorista,... (use um padrão Data => Dt)

Nomes de colunas não necessitam conter o nome da tabela:

- Prefira usar Nome do que usar NomeCli, NomeFor, NomePro
- Em SQL pode-se fazer Tabela.Coluna (Exemplo: Cliente.Nome)

15.2.4. Nome da coluna chave primária

Chave primária:
Pode aparecer em outras tabelas na forma de chave estrangeira

Recomendável:
Nomes das colunas que compõem a chave primária sufixados ou prefixados com o nome ou sigla da tabela na qual aparecem como chave primária

Exemplo:
CodCli, CodFor, CodProd

15.2.5. Implementação de relacionamento

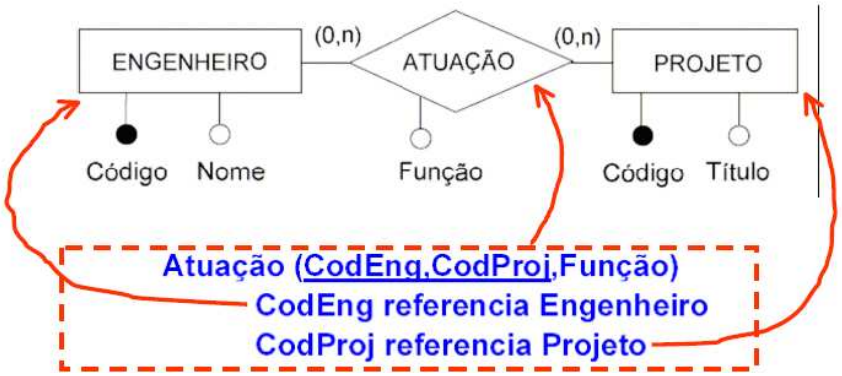
Alternativas básicas:

- Tabela própria
- Adição de colunas a uma das tabelas
- Fusão de tabelas

A escolha de umas dessas alternativas depende da cardinalidade (máxima e mínima do relacionamento).

15.2.6. Tabela própria

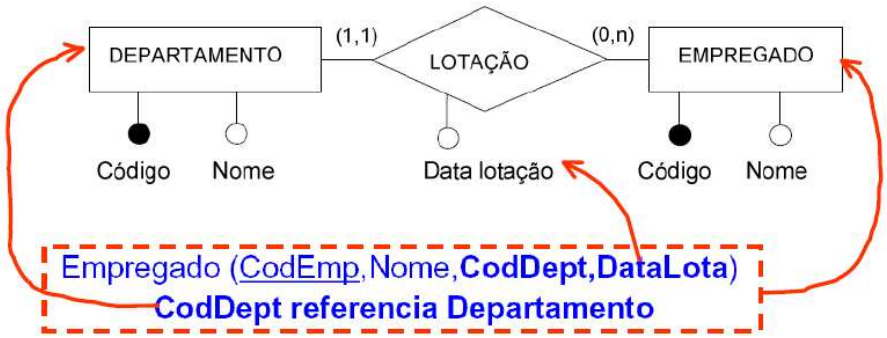
Nesta, o relacionamento é implementado através de uma tabela própria. Esta contém os atributos identificadores das entidades participantes mais os atributos do relacionamento.



15.2.7. Adição de colunas

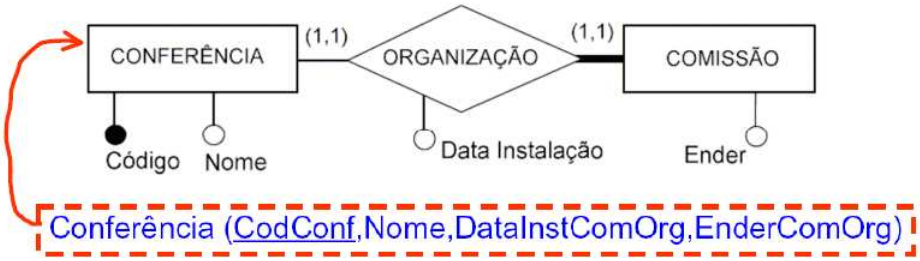
Nesta, o relacionamento é implementado através da inserção de colunas na tabela oposta a multiplicidade máxima 1.

Só é possível quando tem-se relacionamentos com pelo menos uma cardinalidade máxima 1.



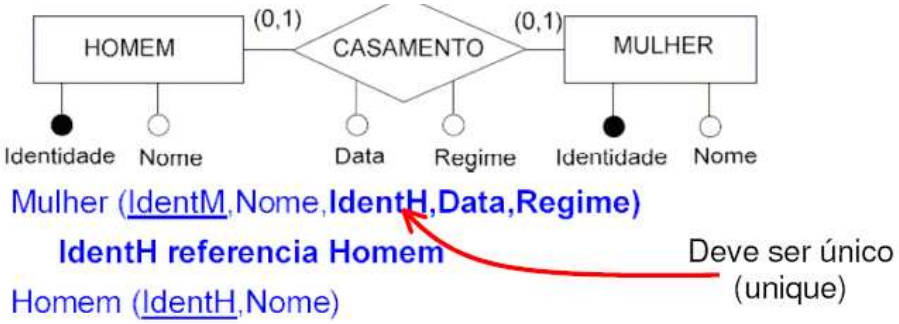
15.2.8. Fusão de tabelas

Nesta, o relacionamento é implementado através da união das tabelas participantes. Só é possível para relacionamentos 1:1.



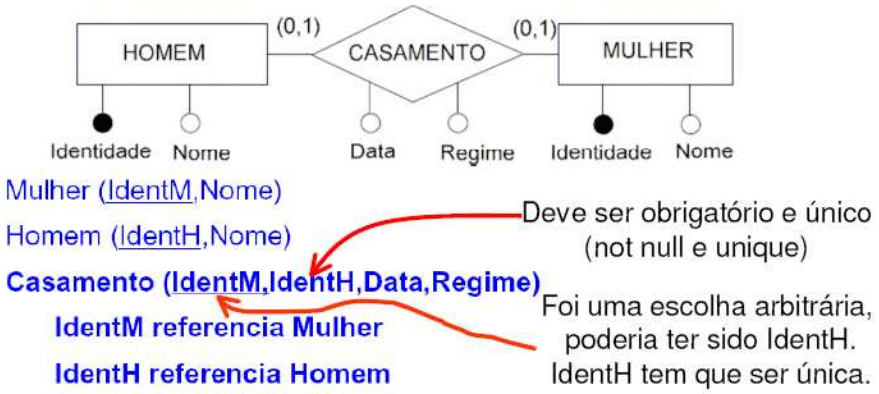
15.2.9. Implementação de relacionamento 1:1

1:1 – onde ambas entidades têm relacionamentos opcionais:



Melhor tradução:
Adição de Colunas - neste caso escolheu-se Mulher arbitrariamente e IdentH tem que ser única.

1:1 – onde ambas entidades têm relacionamentos opcionais:

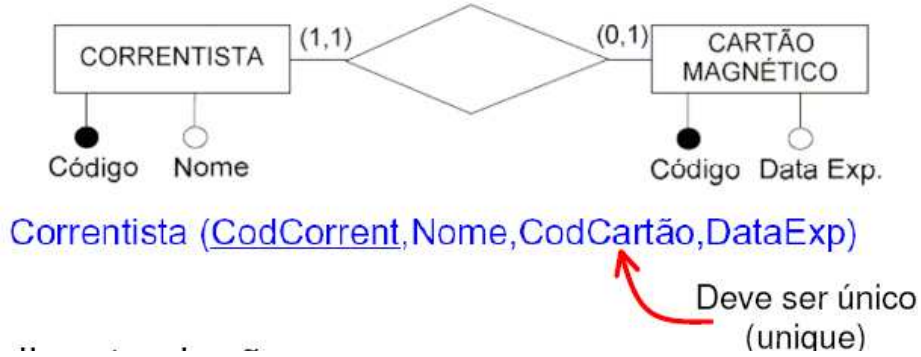


Tradução alternativa:
Tabela Própria - neste caso Casamento torna-se uma tabela.

1:1 – onde ambas entidades têm relacionamentos opcionais.

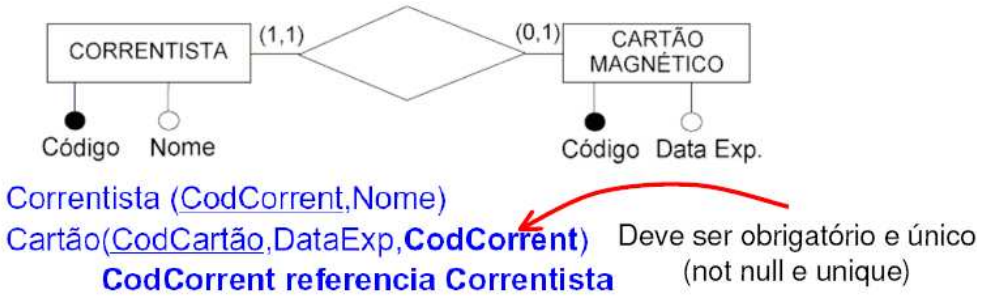
> ***Discussão***:
> *A solução por adição de colunas é a melhor, pois minimiza a quantidade de junções e chaves, entretanto, pode-se ter atributos opcionais, os quais devem ser tratados via programação.*
>
> *A solução de tabela própria é aceitável, mas a de fusão de tabela é semanticamente inviável para o contexto.*

1:1 – onde uma entidade tem relacionamento opcional e a outra tem relacionamento obrigatório:



Melhor tradução: Fusão de Tabela.

1:1 – onde uma entidade tem relacionamento opcional e a outra tem relacionamento obrigatório:



Tradução alternativa: Adição de Colunas.

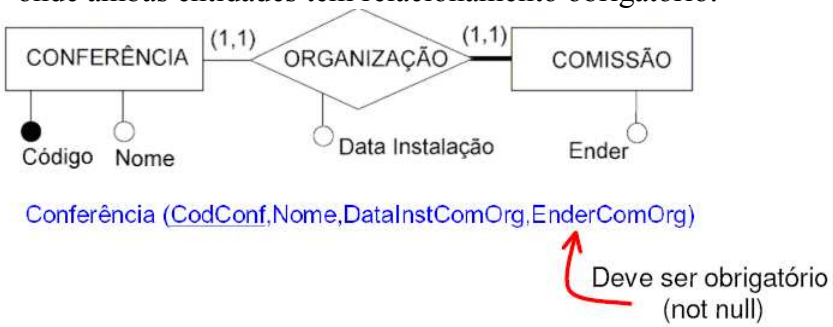
1:1 – onde uma entidade tem relacionamento opcional e a outra tem relacionamento obrigatório.

Discussão:
A solução por fusão de tabelas é a melhor, pois elimina a necessidade de junção e diminuindo a qtd de chave. Entretanto, pode-se ter atributos opcionais, os quais devem ser tratados por programação.

A solução por tabela própria é pior que a solução por adição de colunas, pois gera uma quantidade maior de junções e chaves.

- *Ambas não têm campos opcionais.*

1:1 – onde ambas entidades têm relacionamento obrigatório:



Melhor tradução: Fusão de Tabelas.

1:1 – onde ambas as entidades têm relacionamento obrigatório.

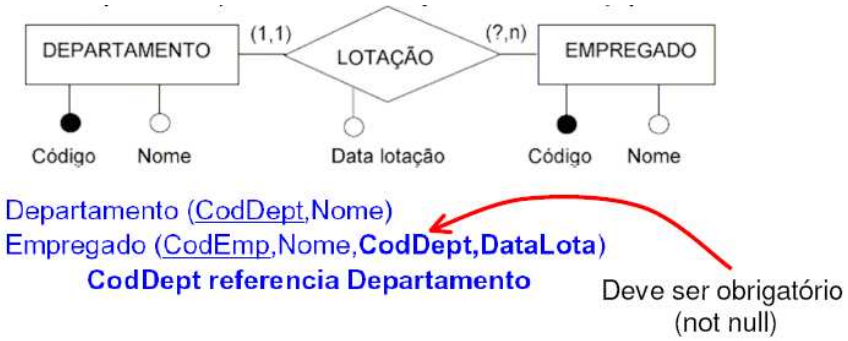
Discussão:
A solução por fusão de tabelas é a melhor, pois elimina a necessidade de junção, diminui a quantidade de chave e não tem atributos opcionais.
Nenhumas das duas outras abordagens são soluções adequadas, Pois:

- *As entidades que participam do relacionamento seriam representadas através de duas tabelas distintas, mas com a mesma chave primária e relação um-para-um entre suas linhas.*

- *Maior número de junções.*
- *Maior número de chaves.*

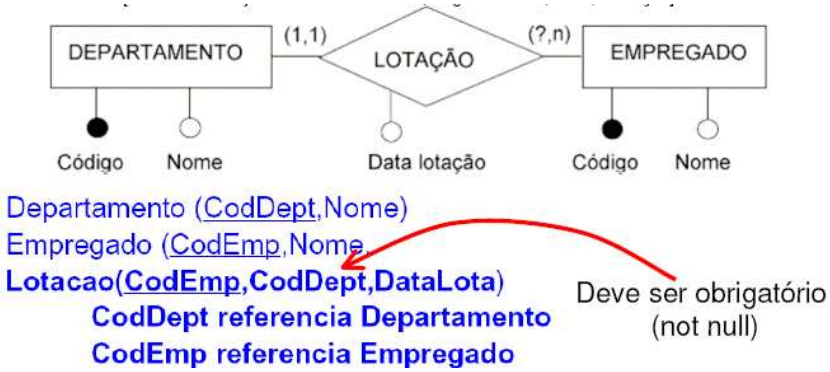
15.2.10. Implementação de relacionamento 1:N

Relacionamentos 1:N onde a entidade que tem multiplicidade máxima 1 é obrigatória:



Melhor tradução: Adição de Colunas.

1:N – onde a entidade que tem multiplicidade máxima 1 é obrigatória:

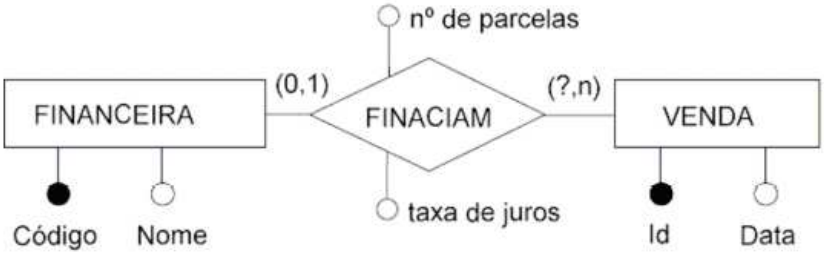


Tradução Alternativa: Tabela Própria.

1:N – onde a entidade que tem multiplicidade máxima 1 é obrigatória.

Discussão:
A solução por adição de colunas é a melhor, pois elimina a necessidade de junção extra, diminui a quantidade de chave e não tem atributos opcionais.
A solução de tabela própria é aceitável, mas evite usá-la, pois com esta, tem-se uma junção e uma chave a mais. Quanto a fusão de tabela, esta é inviável para o contexto, pois implicaria em uma redundância desnecessária de dados sobre departamento.

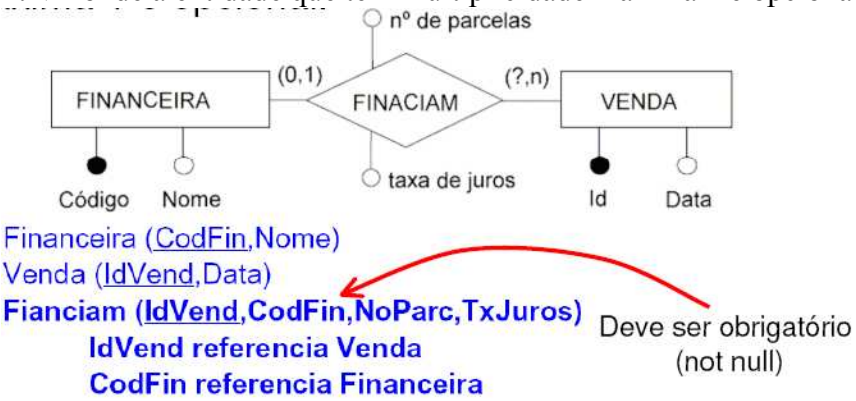
1:N – onde a entidade que tem multiplicidade máxima 1 é opcional:



Financeira (CodFin,Nome)
Venda (IdVend,Data,**CodFin,NoParc,TxJuros**)
CodFin referencia Financeira

Melhor tradução: Adição de Colunas.

1:N – onde a entidade que tem multiplicidade máxima 1 é opcional:



Tradução alternativa: Tabela Própria.

1:N – onde a entidade que tem multiplicidade máxima 1 é opcional.

Discussão:
A solução por adição de colunas é a melhor, pois elimina a necessidade de junção extra e diminui a quantidade de chave. Entretanto, tem-se atributos opcionais, os quais devem ser tratados por programação.
A solução de tabela própria é aceitável (tem junção a mais, mas não tem atributos opcionais). Contudo, a de fusão de tabela é inviável para o contexto, pois implicaria em uma redundância desnecessária de dados sobre financeira.

Note: Em relacionamentos 1:N, sempre a melhor tradução é a de Adição de Colunas.

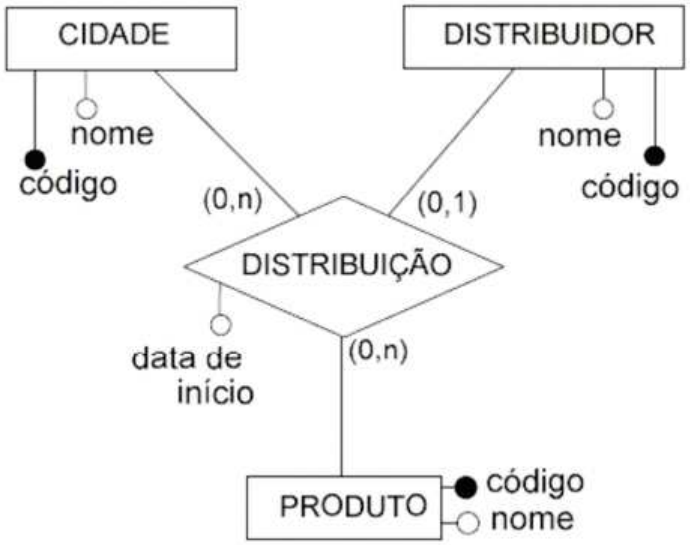
15.2.11. Implementação de relacionamento N:N

N:N – em todas as combinações de multiplicidade, á única tradução possível é de Tabela Própria (onde a Chave Primária é composta).

Engenheiro (CodEng,Nome)
Projeto (CodProj,Título)
Atuação (CodEng,CodProj,Função)
CodEng referencia Engenheiro
CodProj referencia Projeto

15.2.12. Relacionamentos com grau maior que 2

- As regras vistas até agora são aplicáveis apenas a relacionamentos binários.
- Para relacionamento com cardinalidade/grau maior que 2, não são definidas regras específicas. Mas no geral, o relacionamento é transformado em uma entidade, onde sua CP é normalmente composta de forma a atender a regra de negócio.
- Se não for afetar a regra de negócio transforme relacionamentos com cardinalidade/grau maior que 2 para relacionamentos binários.

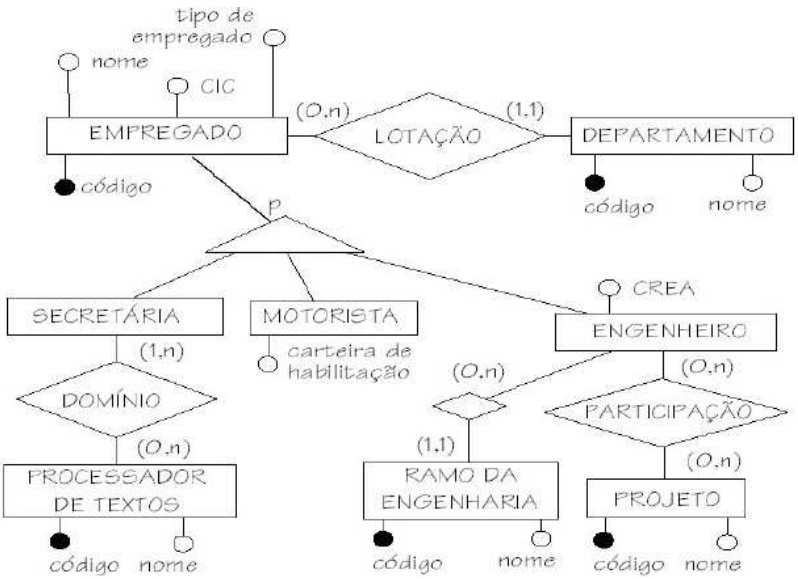


Produto (CodProd,Nome)
Cidade (CodCid,Nome)
Distribuidor (CodDistr,Nome)
Deve ser obrigatório (not null)
Distribuição (CodProd,CodDistr,CodCid,DataInicio)
CodProd referencia Produto
CodDistr referencia Distribuidor
CodCid referencia Cidade

15.2.13. Implementação de generalização/especialização

Duas alternativas básicas:

1. Uso de uma única tabela para toda hierarquia;
2. Uso de uma tabela para cada entidade.



Uso de uma única tabela para toda hierarquia:
Todas as tabelas referentes às especializações são fundidas em uma única tabela, a qual contém:

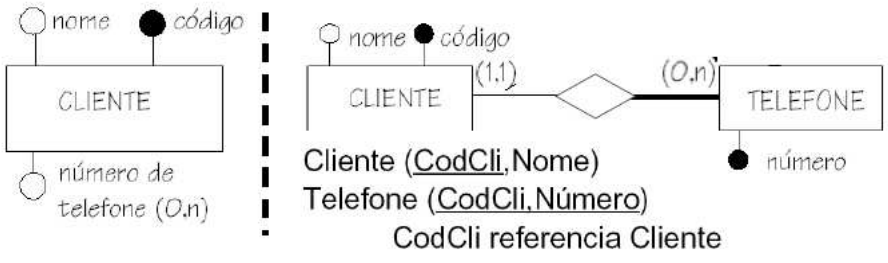
- Chave primária correspondente ao identificador da entidade mais genérica;
- Caso não exista uma coluna tipo, a mesma deve ser adicionada;
- Uma coluna para cada atributo da entidade genérica;
- Uma coluna para cada chave estrangeira da entidade genérica;
- Uma coluna para cada atributo de cada entidade especializada;
    - Estas não devem ser colunas obrigatórias.
- Uma coluna para cada chave estrangeira da entidade especializada.
    - Estas também não devem ser obrigatórias.

Emp (CódigoEmp,Tipo,Nome,CIC,CodigoDept,
CartHabil,CREA,CódigoRamo)
CódigoDept referencia Depto
CódigoRamo referencia Ramo
Depto (CódigoDept, Nome)
Ramo (CódigoRamo,Nome)
ProcessTexto (CódigoProc,Nome)
Domínio (CódigoEmp,CódigoProc)
CódigoEmp referencia Emp
CódigoProc referencia ProcessTexto
Projeto (CódigoProj,Nome)
Participação (CódigoEmp,CodigoProj)
CódigoEmp referencia Emp
CódigoProj referencia Projeto

Discussão:
O uso de uma única tabela para toda hierarquia minimiza junções e diminui a quantidade de chaves, entretanto, tem-se atributos opcionais, os quais devem ser tratados por programação.
A solução com uso de uma tabela para cada entidade não tem atributos opcionais, mas apresenta um número maior de junções e chaves.

15.2.14. Implementação de Atributos Multivalorados

Atributos Multivalorados:
Transformar o atributo multivalorado em uma nova tabela fraca, incluindo na sua CP, a CP da tabela origem.



ATENÇÃO: Deve-se avaliar a real necessidade de se criar uma nova tabela (evitar junção!) Um cliente pode ter N telefones, mas se dois são suficientes então:

Cliente (CodCli,Nome,NumTel1,NumTel2)

Inconveniente:
Consulta por telefone serão mais complicadas, pois devem referenciar todas as colunas referentes ao atributo multivalorado.

15.3. Normalização

Obtido o esquema relacional correspondente ao documento, passa-se ao processo de *normalização*. Este processo baseia-se no conceito de *forma normal*. Uma forma normal é uma regra que deve ser obedecida por uma tabela para que esta seja considerada "bem projetada". Há diversas formas normais, isto é, diversas regras, cada vez mais rígidas, para verificar tabelas relacionais.

O processo de normalização pode ser visto como o processo no qual são eliminados esquemas de relações (tabelas) não satisfatórios, decompondo-os, através da separação de seus atributos em esquemas de relações menos complexas, mas que satisfaçam as propriedades desejadas.

O processo de normalização como foi proposto inicialmente por Codd conduz um esquema de relação através de um bateria de testes para certificar se o mesmo está na 1a, 2a e 3ª Formas Normais. Estas três Formas Normais são baseadas em dependências funcionais dos atributos do esquema de relação.

O processo de normalização deve ser aplicado em uma relação por vez, pois durante o processo de normalização vamos obtendo quebras e, por conseguinte, novas relações. No momento em que o sistema estiver satisfatório, do ponto de vista do analista, este processo iterativo é interrompido. De fato existem literaturas indicando quarta, quinta formas normais, que não nos parece tão importante, nem mesmo academicamente.

A normalização para formas apoiadas em dependências funcionais evita inconsistências, usando para isso a própria construção da Base. Se a mesma consistência for passível de ser garantida pelo aplicativo, a normalização pode ser evitada com ganhos reais no desempenho das pesquisas. No caso da consistência não ser importante, também podemos não normalizar totalmente uma Base de Dados.

15.3.1. 1ª Forma Normal

A **1ª Forma Normal** prega que todos os atributos de uma tabela devem ser **atômicos** (indivisíveis), ou seja, não são permitidos atributos multivalorados, atributos compostos ou atributos multivalorados compostos, ou seja, uma tabela encontra-se na 1FN quando não contém tabelas aninhadas. Portanto, a passagem à 1FN consta da eliminação das tabelas aninhadas eventualmente existentes.

Alternativas para a 1ª Forma Normal

Para transformar um esquema de tabela não-normalizada em um esquema na 1FN há duas alternativas:

Construir uma única tabela com redundância de dados:

- Cria-se uma tabela na qual os dados das linhas externas à tabela aninhada são repetidos para cada linha da tabela aninhada. No caso da tabela da Figura 1, o esquema resultante seria o seguinte:

  ProjEmp (CodProj, Tipo, Descr, CodEmp, Nome, Cat, Sal, DataIni, TempAl)
- Nesta tabela, os dados do projeto - CodProj, Tipo e Descr - aparecem repetidos para cada linha da tabela de empregados - Emp.

Construir uma tabela para cada tabela aninhada:

- Cria-se uma tabela referente à própria tabela que está sendo normalizada e uma tabela para cada tabela aninhada. No caso da tabela da Figura 1, o esquema resultante seria o seguinte:

  Proj (CodProj, Tipo, Descr)
  ProjEmp (CodProj,CodEmp, Nome, Cat, Sal, DataIni, TempAl)

Dependência Funcional

Em uma tabela relacional, diz-se que uma coluna C2 *depende funcionalmente* de uma coluna C1 (ou que a coluna C1 *determina* a coluna C2) quando, em todas as linhas da tabela, para cada valor de C1 que aparece na tabela, aparece o mesmo valor de C2.

15.3.2. 2ª Forma Normal

Uma dependência funcional **X → Y** é **total** se removemos um atributo **A** qualquer do componente **X** e desta forma, a dependência funcional deixa de existir. A dependência funcional **X → Y** é uma **dependência funcional parcial** se existir um atributo **A** qualquer do componente **X** que pode ser removido e a dependência funcional **X → Y** não deixa de existir.

A passagem à segunda forma normal (2FN) objetiva eliminar certo tipo de redundância de dados. A passagem à segunda forma normal objetiva eliminar este tipo de redundância de dados. Uma tabela encontra-se na segunda forma normal (2FN) quando, além de encontrar-se na primeira forma normal, cada coluna não chave depende da chave primária *completa*.

Uma tabela que não se encontra na segunda formal contém *dependências funcionais parciais*, ou seja, contém colunas não chave que dependem apenas de uma parte da chave primária.

3ª Forma Normal
Uma tabela encontra-se na 3FN quando, além de estar na 2FN, toda coluna não chave depende *diretamente* de chave primária, isto é, quando não há dependências funcionais *transitivas* ou *indiretas*.

Uma dependência funcional transitiva ou indireta acontece quando uma coluna não chave primária depende funcionalmente de outra coluna ou combinação de colunas não chave primária. A passagem à 3FN consta em dividir tabelas de forma a eliminar as dependência transitivas.

16. SGBD

A Seguir traremos exemplos de SGBDS, mas antes vamos a conceitos a respeitos dos SGBDS:

Um Sistema de Gerenciamento de Banco de Dados é o conjunto de programas de computador (softwares) responsáveis pelo gerenciamento de uma base de dados. O principal objetivo é retirar da aplicação cliente a responsabilidade de gerenciar o acesso, manipulação e organização dos dados. O SGBD disponibiliza uma interface para que os seus clientes possam incluir, alterar ou consultar dados. Em bancos de dados relacionais a interface é constituída pelas APIs ou drivers do SGBD, que executam comandos na linguagem SQL.

Abaixo já trazemos alguns dos mais conhecidos SGBD's do mercado:

- **MySQL;**
- **Oracle;**
- **PostgreSQL;**
- **Firebird;**
- **Sybase;**
- HSQLDB;
- mSQL;
- SQL-Server;
- TinySQL;
- JADE;
- ZODB;
- IBM DB2.

16.1. MySQL

É um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (Structured Query Language - Linguagem de Consulta Estruturada) como interface. É atualmente um dos bancos de dados mais populares, com milhões de instalações pelo mundo.

O MySQL possui as seguintes características:

- Suporta diferentes plataformas: Win32, Linux, FreeBSD, Unix, etc...
- Alta compatibilidade com linguagens como PHP, Java, Python, C#, Ruby e C/C++;
  Vários sistemas de armazenamento de dados (batabase engine), como MyISAM, MySQL Cluster, CSV, Merge, InnoDB, entre outros;
  Recursos como transactions (transações), conectividade segura, indexação de campos de texto, replicação, etc;
- Baixa exigência de processamento (em comparação como outros SGBD);
  Excelente desempenho e estabilidade;
- Pouco exigente quanto a recursos de hardware;
- Facilidade de uso;
- É um Software Livre;
- Suporte a vários tipos de tabelas (como MyISAM, InnoDB e Maria), cada um específico para um fim;
- Aceita Triggers;
- Aceita Stored Procedures e Functions;
- Instruções em SQL, como indica o nome.
- Replicação facilmente configurável.

Abaixo são listadas algumas ferramentas de gerenciamento de banco de dados para o MySQL:

- **SQLyog**: É um programa desenvolvido pela WEByog Enterprise que possibilita a edição de bancos de dados Mysql, que baseados na linguagem SQL. Utilizado na criação, edição, sincronização de banco de dados internos e em servidores. É disponibilizado em duas versões, Enterprise, paga, e Community, gratuita e de código aberto.

- **DBDesigner**: Integra criação, modelagem, desenvolvimento e manutenção dos bancos em um ambiente simples e agradável. O DBDesigner é OpenSource distribuído sobre a licença GPL.

- **PhpMyAdmin**: é um programa de computador desenvolvido em PHP para administração do MySQL pela Internet. A partir deste sistema é possível criar e remover bases de dados, criar, remover e alterar tabelas, inserir, remover e editar campos, executar códigos SQL e manipular campos chaves.

16.2. Oracle

Apesar de ser proprietário o SGBD da Oracle é líder de mercado. O Oracle 9i foi pioneiro no suporte ao modelo web. O Oracle 10g, mais recente, se baseia na tecnologia de grid. Recentemente fora lançado o Oracle 11g que veio com melhorias em relação ao Oracle 10g.

Além da base de dados, a Oracle desenvolve uma suíte de desenvolvimento chamada de Oracle Developer Suite, utilizada na construção de programas de computador que interagem com a sua base de dados.

A Oracle também criou a linguagem de programação PL/SQL, utilizada no processamento de transações.

Abaixo são listadas algumas ferramentas de gerenciamento de banco de dados para o ORACLE:

- **DBDesigner**;
- **TOAD**;
- **SQLNavigator**.

16.3. PostgreSQL

É um dos SGBDs de código aberto mais avançados, contando com recursos como:

- Consultas complexas;
- Chaves estrangeiras;
- Integridade transacional;
- Controle de concorrência multi-versão;
- Suporte ao modelo híbrido objeto-relacional;
- Triggers;
- Visões;
- Procedimentos armazenados em várias linguagens.

Abaixo são listadas algumas ferramentas de gerenciamento de banco de dados para o PostgreeSQL:

- **phpPgAdmin**;
- **PGExplorer**;

17. Exercícios Resolvidos

1) O que é um SGBD? Cite 3 exemplos.

É um software com recursos específicos para facilitar a manipulação das informações de um BD e o desenvolvimento de programas aplicativos. Exemplos: Oracle, SQL Server, MySQL, Access, Interbase, Paradox, Sybase.

2) Quais as principais vantagens da utilização de um Sistema de Banco de Dados em relação aos sistemas tradicionais de gerenciamento de arquivos?

- Rapidez na manipulação e no acesso à informação;
- Redução do esforço humano no desenvolvimento e utilização das aplicações;
- Disponibilização da informação no tempo necessário;
- Controle integrado de informações distribuídas fisicamente;
- Redução da redundância e de inconsistência de informações;
- Compartilhamento de dados;
- Aplicação automática de restrições de segurança;
- Redução de problemas de integridade.

3) Indique alguns problemas que dificultariam o uso de banco de dados.

Problemas:

- Custos iniciais (hardware, software, treinamento);
- Sobrecarga de processamento.

Não usar se:

- Aplicações e dados simples e fixos;
- Urgência no tempo de resposta;
- Usuário único.

4) Explique o que são Visões em SGBD?

Um SGBD deve permitir que cada usuário visualize os dados de forma diferente daquela existente previamente no BD. Uma visão consiste de um subconjunto de dados do BD, mas não necessariamente estes deverão estar armazenados no BD. Portanto, uma replicação de uma estrutura, para fins de acesso de forma diferenciada por outros aplicativos, não caracteriza o uso de um SGBD.

5) O sistema de banco de dados deve prover uma visão abstrata de dados para os usuários, isolando, desta forma, detalhes mais internos do BD. A abstração se dá em três níveis. Explique estes níveis de abstração.

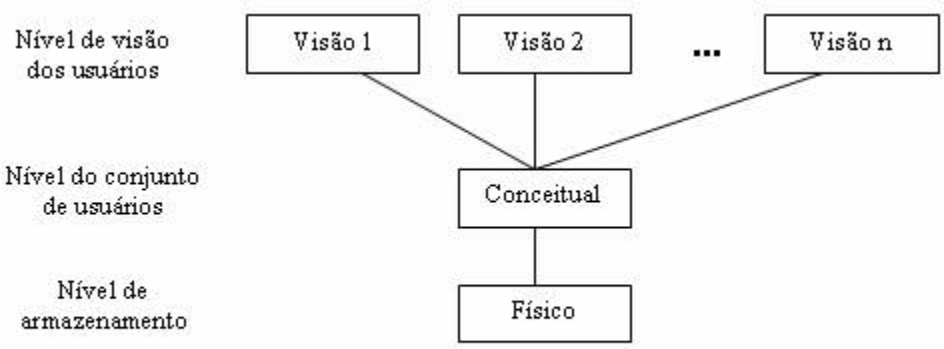


FIGURA 1.1 – NÍVEIS DE ABSTRAÇÃO

Nível físico: também chamado de "Esquema interno", é o nível mais baixo de abstração. Descreve como os dados estão realmente armazenados, englobando estruturas complexas de baixo nível.

Nível conceitual (ou lógico): conhecido também como "Esquema Conceitual", descreve quais os dados estão armazenados e seus relacionamentos. Neste nível, o BD é descrito através de estruturas relativamente simples, que podem envolver estruturas complexas no nível físico.

Nível de visões do usuário: é o nível externo, descrevendo partes do BD que serão visualizadas pelos usuários de acordo com suas necessidades. Uma visão é um subconjunto de dados do BD, sem que exista a necessidade de estarem armazenados no BD.

6) O que são modelos lógicos de dados?

É o conjunto de ferramentas conceituais para a descrição dos dados, dos relacionamentos entre os mesmos e das restrições de consistência e integridade. Em outras palavras, é o conjunto de conceitos que podem ser usados para descrever a estrutura de um banco de dados (tipos de dados, relacionamentos e restrições que devem ser mantidas sobre os dados).

7) Defina Esquema e Instância de um Banco de Dados.

Esquema:

- O esquema é a definição das estruturas que compõem o banco de dados;
- Espera-se que o esquema sofra nenhuma ou muito poucas alterações depois de implementado;

O esquema independe dos dados a serem armazenados.

Instância:

- Instância é a materialização do banco de dados composto pelas estruturas mais os dados armazenados.
- Uma instância é um "retrato" do banco de dados em um determinado momento.
- Podemos ter a mesma estrutura replicada em vários locais, cada uma com seu conjunto de dados.

8) Defina o Modelo Entidade-Relacionamento.

O Modelo Entidade-Relacionamento é uma forma de se definir o esquema do banco de dados, composta pelos seguintes elementos: Entidades, Relacionamentos, Atributos, Chaves Primárias, Estrangeiras, e outras informações descritas em forma textual.

É a principal ferramenta gráfica para representação do Modelo de Dados.

9) As pessoas envolvidas num sistema de banco de dados podem ser divididas em usuários e administradores. Descreva os principais tipos de usuários e administradores existentes.

Usuários

- Programador de aplicação - Desenvolve programas aplicativos que manipulam banco de dados
- Usuário sofisticado - Capaz de manipular dados num BD sem o uso de aplicativos utilizando as linguagens de consulta
- Usuário "ingênuo" - Manipula BD somente pelas interfaces definidas nos programas aplicativos

Administradores

- Administrador de dados - Define e atualiza do esquema do BD
- Administrador do SGBD
  - Define estrutura de armazenamento e estratégia de acesso;
  - Concede autorização de acesso a dados;
  - Define controles de integridade;
  - Define estratégias para backup;
  - Monitora desempenho.

10) Criar uma tabela chamada DEPARTAMENTO com as seguintes colunas: COD_DEP, NOME_DEP.

```
CREATE TABLE DEPARTAMENTO (
COD_DEP NUMBER(5),
NOME_DEP VARCHAR2(50));
```

11) Alterar a tabela anterior: criar uma chave primária chamada DEP_PK na coluna COD_DEP.

```
ALTER TABLE DEPARTAMENTO
ADD CONSTRAINT DEP_PK PRIMARY KEY (COD_DEP);
```

12) Renomear a tabela criada no item 1 para DEP.

```
RENAME DEPARTAMENTO TO DEP;
```

13) Criar uma tabela chamada FUNCIONARIO com as seguintes colunas: COD_FUNC, NOME_FUNC.

```
CREATE TABLE FUNCIONARIO (
COD_FUNC NUMBER(5),
NOME_FUNC VARCHAR2(80));
```

14) Alterar a tabela anterior (item 4) conforme segue:

a. Renomear a tabela para FUNC.

```
RENAME FUNCIONARIO TO FUNC;
```

b. Criar uma chave primária na coluna COD_FUNC.

```
ALTER TABLE FUNC
ADD CONSTRAINT FUNC_PK PRIMARY KEY (COD_FUNC);
```

c. Criar uma nova coluna chamada COD_DEP (mesmo tipo e tamanho da coluna com mesmo nome da tabela DEP).

```
ALTER TABLE FUNC
ADD COD_DEP NUMBER(5);
```

d. Criar uma chave estrangeira chamada FUNC_DEP_FK na coluna COD_DEP referenciando a tabela DEP.

```
ALTER TABLE FUNC
ADD CONSTRAINT FUNC_DEP_FK FOREIGN KEY (COD_DEP)
REFERENCES DEP (COD_DEP);
```

e. Alterar a coluna NOME_FUNC para NOT NULL.

```
ALTER TABLE FUNC
MODIFY NOME_FUNC NOT NULL;
```

15) Criar uma constraint unique na coluna NOME_DEP da tabela DEP.

```
ALTER TABLE DEP
ADD CONSTRAINT DEP_NOME_UN UNIQUE (NOME_DEP);
```

16) Criar mais uma coluna chamada TEMP na tabela DEP.

```
ALTER TABLE DEP
ADD TEMP VARCHAR2(10);
```

17) Alterar a largura da coluna TEMP.

```
ALTER TABLE DEP
MODIFY TEMP VARCHAR2(15);
```

18) Alterar o nome da coluna TEMP para TEMP2.

```
ALTER TABLE DEP
RENAME COLUMN TEMP TO TEMP2;
```

19) Excluir a coluna TEMP2.

```
ALTER TABLE DEP
```

```
DROP COLUMN TEMP2;
```

20) Criar uma tabela chamada DEP2 com base na tabela DEP (com todas as suas colunas e linhas).

```
CREATE TABLE DEP2 AS SELECT * FROM DEP;
```

21) Criar uma coluna chamada ESTADO_CIVIL na tabela FUNC.

```
ALTER TABLE FUNC
ADD ESTADO_CIVIL VARCHAR2(10);
```

22) Criar uma constraint check na coluna ESTADO_CIVIL da tabela FUNC (valores aceitos: 'solteiro','casado','viúvo', 'separado'.

```
ALTER TABLE FUNC
ADD CONSTRAINT FUNC_ESTADO_CIVIL_CK
CHECK (ESTADO_CIVIL IN ('solteiro','casado','viúvo','separado'));
```

23) Criar uma coluna chamada DATA_ADM na tabela FUNC (default: SYSDATE).

```
ALTER TABLE FUNC
ADD DATA_ADM DATE DEFAULT SYSDATE;
```

24) 15.Criar um índice chamado NOME_FUNC_IN na coluna NOME_FUNC da tabela FUNC.

```
CREATE INDEX NOME_FUNC_IN ON FUNC (NOME_FUNC);
```

25) Eliminar o índice criado no item anterior.

```
DROP INDEX NOME_FUNC_IN;
```

26) Criar uma view chamada DEP_VW com base nas colunas COD_DEP e NOME_DEP da tabela DEP.

```
CREATE VIEW DEP_VW AS SELECT COD_DEP, NOME_DEP FROM DEP;
```

27) Excluir a view criada no item anterior.

```
DROP VIEW DEP_VW;
```

28) Eliminar todas as linhas da tabela FUNC (utilizar o comando TRUNCATE).

```
TRUNCATE TABLE FUNC;
```

29) Eliminar a tabela FUNC.

```
DROP TABLE FUNC;
```

30) Eliminar a tabela DEP.

```
DROP TABLE DEP;
```

31) Modele o sistema abaixo no Dia Modeler (sugestão) ou manualmente, se preferir.

O sistema Educare é responsável pelo controle de alunos, professores, salas, cursos, disciplinas, recursos didáticos e coordenações do campus. Os dados armazenados neste sistema estão descritos abaixo.

O sistema permite realizar o cadastro da instituição e de seus campi. Para o cadastro da instituição é necessário entrar com nome, cidade, endereço e sigla. Cada campus é descrito por nome, cidade, endereço e data de implantação.

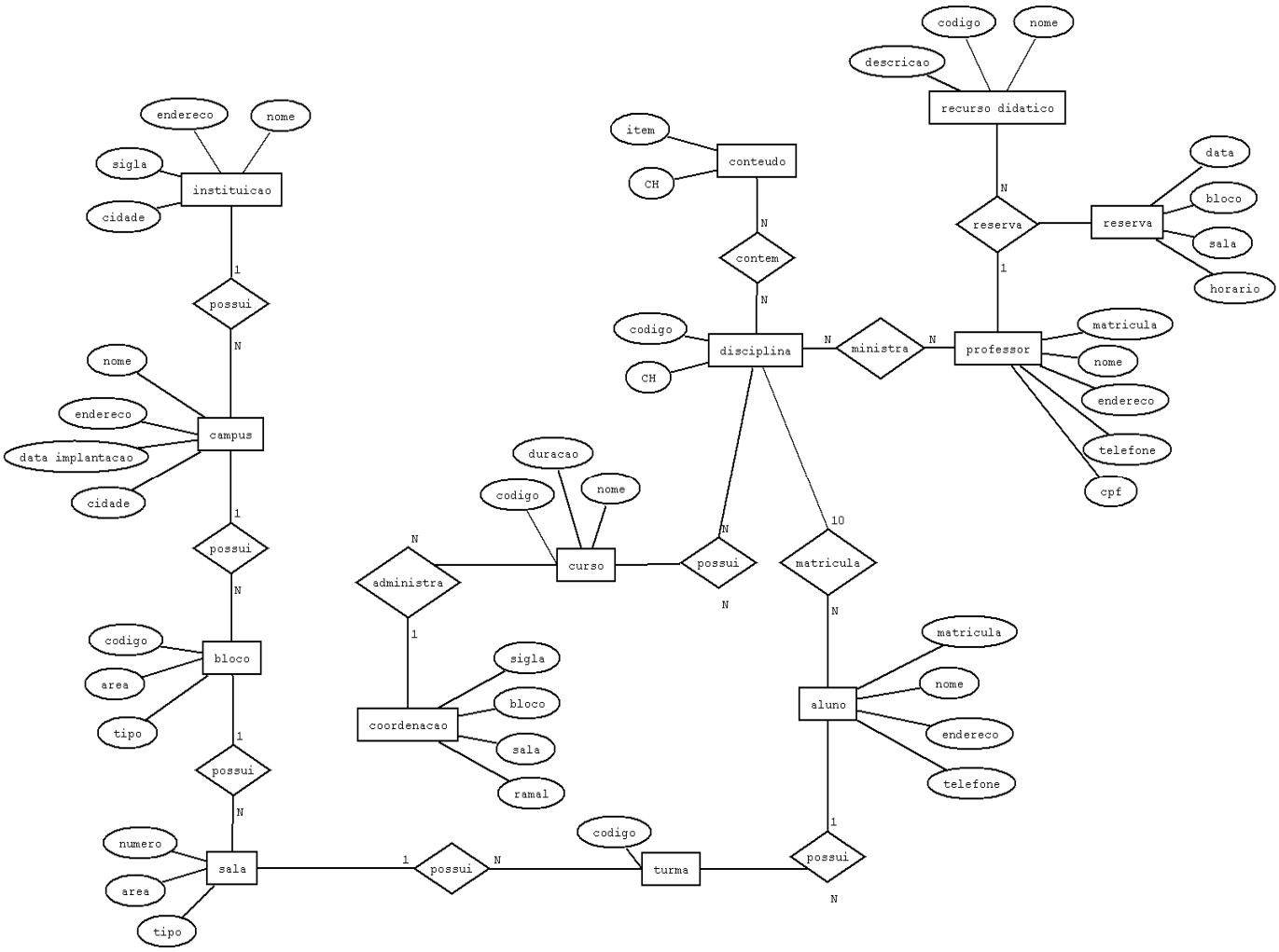
Para cada campus, é possível armazenar os blocos. No bloco, informamos código de identificação (campo alfanumérico), área (m2) e tipo (didático ou administrativo). Os blocos possuem salas. Em um bloco didático pode haver salas administrativas. Logo, é preciso que o cadastro da sala indique se ela á administrativa ou didática. Para a sala, também guarda-se o número e área (m2). Os alunos estão cadastrados no sistema por matrícula, nome, endereço e telefone. Cada aluno pode estar matriculado em até 10 disciplinas por vez. Uma disciplina é descrita por código e carga horária. O conteúdo da disciplina é guardado em outra tabela através dos campos item e carga horária. O mesmo conteúdo pode estar em disciplinas diferentes. Por exemplo: a disciplina de Introdução à Computação do curso de Engenharia em Alimentos possui o mesmo conteúdo da disciplina Informática Básica do curso de Ciências Contábeis.

As disciplinas pertencem aos cursos. Um curso possui várias disciplinas e a mesma disciplina pode estar em cursos diferentes. Sobre o curso, o sistema armazena o nome, código e duração (em anos).

O sistema também organiza os alunos em turmas, que são associadas aos cursos. Da turma, é possível conhecer seu código e sala.

Os professores ministram disciplinas. Pode ocorrer de mais de um professor ministrar a mesma disciplina. O professor é cadastrado por matrícula, nome, cpf, endereço e telefone.
Os cursos estão ligados a coordenações (Informática, Direito, Administração, etc). Uma só coordenação pode ser responsável por vários cursos. No entanto, um curso não pode ser administrado por mais de uma coordenação. Da coordenação, o sistema guarda sigla, bloco, sala e ramal.

O sistema também gerencia recursos didáticos, tais como projetores multimídia, notebooks e DVDs. O recurso é cadastrado por nome, código e descrição. Há o processo de reserva, sendo que cada recurso pode ser reservado por um professor para uma determinada data, bloco, sala e horário.



32) Listar todos os dados da tabela funcionários ordenados por matrícula:

```
Select *
From funcionario
Order by matricula
```

33) Forneça o número total de empregados da companhia:

```
Select count(*)
From funcionário
```

34) Liste todos os funcionários que tenham entre 10 e 12 anos de serviço (inclusive):

```
Select *
From funcionário
Where anos_servico between ‘10’and’12’
```

35) Liste o nome, a matrícula e o salário de todas as pessoas cujo salário não esteja entre R$ 1.000,00 e R$ 3.000,00:

```
Select nome_func,matricula,salário
From funcionário
Where salário <'1000,00' or salário>'3000,00'
```

36) Liste todos os funcionários com 1, 2 ou 3 anos de serviço, ou cujo valor de anos de serviço seja nulo:

```
Select *
From funcionário
Where anos_servico=1 or anos_servico=2 or anos_servico=3 or anos_servico is null
```

37) Liste a matrícula, nome e salário de todas as pessoas em ordem alfabética por nome:

```
Select matricula, nome_func, salário
From funcionário
Order by nome_func
```

38) Liste o departamento, a matrícula, o nome e o salário dos funcionários em ordem

```
decrescente de salário em seu departamento:
Select depto,matricula,nome_func,salário
From departamento,funcionário,lotado_depto
Where matric_func=matricula and cód_depto=depto
Order by salário desc
```

39) Liste as matrículas, os nomes e os departamentos de todos os empregados que gerencia algum departamento:

```
Select matricula, nome_func,nome_depto
From funcionário,departamento,gerencia_depto
Where matricula_ger=matricula and num_div=depto
```

40) Liste a média salarial de cada departamento:

```
Select avg(salário)
From departamento,funcionário,lotado_depto
Where matric_func=matricula and cod_depto=depto
```

41) Liste matrícula, nome, anos de serviço de todos os gerentes de departamentos que recebem salários maiores que R$ 2000,00 ou que possuam matrículas maiores que 30, e que tenham mais de 6 anos de serviço:

```
Select matricula,nome_func,anos-servico
From funcionario,gerencia_depto
Where matricula_ger=matricula and anos_servico>'6' and (salário>'2000,00' or matricula>'30')
```

42) Liste o departamento, matrícula, nome e salário de todos os gerentes em ordem decrescente de salário dentro de departamento:

```
Select nome_depto,matricula,nome_func,salário
From departamento,funcionário,gerencia_depto,lotado_depto
Where matricula_ger=matricula and cód_depto=depto and matric_func=matricula
Order by salário desc
```

43) Descubra qual a média de ganhos totais e quantos empregados são considerados para cálculo dessa média:

```
Select avg(salário), count(*)
From funcionário
```

44) Liste o nome, a matrícula e o salário de todas as pessoas que não ganhem salário inferior a R$ 1000,00:

```
Select nome_func,matricula,salário
From funcionário
Where salário> '1000,00'
```

45) Liste a matrícula, nome, anos de serviço e salário de todas as pessoas com mais de 6 anos de serviço e que, ou seja gerente de departamento ou ganhe salário superior a R$ 3000,00:

```
Select matricula,nome_func,anos_servico,salário
From funcionario,gerencia_depto,departamento,lotado_depto
Where cod_depto=depto and matric_func=matricula and anos_servico>'6' and
(matricula_ger=matricula or salário>'3000,00')
```

46) Liste a matrícula, o nome e o salário de todas as pessoas em ordem decrescente de salário:

```
Select matricula,nome_func,salário
From funcionário
Group by salário desc
```

47) Liste o nome, a matrícula e o departamento de todas as pessoas que não trabalhem nos departamentos 10 e 20

```
Select nome_func,matricula,nome_depto
```

```
From funcionario,departamento,lotado_depto
Where depto not in('10) and depto not in('20') and matricula=matric_func and
cod_depto=depto
```

48) Liste a localização, a matrícula e o nome de todos os empregados do departamento de compra e venda:

```
Select localização,matricula,nome_func
From departamento,funcionário,lotado_depto
Where nome_depto='compra' and nome_depto='venda' and cod_depto=depto and
matric_func=matricula
```

49) Liste o nome e a matrícula de todos os funcionários alocados no projeto "holografia":

```
Select nome_func,matricula
From funcionário,projeto,alocado
Where nome_proj='holografia' and mat_func=matricula and
cod_proj=codigo_proj
```

50) Liste todos os funcionários que trabalhem na zona sul:

```
Select nome_func
From departamento,funcionario,lotado_depto
Where localização='zona sul' and cód_depto=depto and matric_func=matricula
```

18. Exercícios Propostos

1. Uma Empresa possui funcionários. Um funcionário trabalha em uma Empresa.

2. Os Atletas participam de competições. Em uma competição participam vários atletas.

3. Deseja-se fazer um banco de dados para uma rede de hotelaria. Um hotel possui quartos. Cada quarto pertence a apenas um hotel.

4. Um soldado, que possui as características nome, Registro Militar (RM), data de nascimento, possui armas. Uma arma, que possui as características de série, registro e calibre, é de um soldado. Uma arma é limpa por vários soldados. Um soldado limpa várias armas.

5. Um funcionário é supervisionado por um gerente (o gerente também é um funcionário). Um gerente (que também é funcionário) supervisiona vários funcionários. Do funcionário deseja-se saber nome, cpf e endereço.

6. Um médico trata de pacientes. Do médico deseja-se saber CRM, nome e suas especialições. Um paciente, no qual há a necessidade de sabermos seu nome, endereço e idade, é tratado por vários médico. Um paciente realiza vários tipos de exames. Um tipo de exame, destes há a necessidade de guardar seu número, data e descrição, é feito por vários paciente.

7. O aluno cursa disciplinas lecionadas por um professor cada uma. Para cada aluno deve-se manter as informações de, nome e seus telefones. Uma disciplina é cursada por vários alunos e é lecionada por um professor. Das disciplinas deseja-se saber codigo, número de créditos e descrição. Os professores lecionam diversas disciplinas cada um e em cada disciplina possui diversos alunos. Dos professores deseja-se saber seu código, nome e telefone.

8. Um médico trata de pacientes. Do médico deseja-se saber CRM, nome e suas especialições. O médico pede exames para vários pacientes. Um paciente, no qual há a necessidade de sabermos seu nome, endereço e idade, é tratado por vários médico. Um paciente realiza vários tipos de exames pedidos pelos médicos. Um tipo de exame, destes há a necessidade de guardar seu número, data e descrição, é feito por vários paciente a pedido dos médicos.

9. Uma empresa possui funcionários. Os funcionários podem ser engenheiros, secretárias ou técnicos.
10. Uma empresa fabrica automóveis. Os automóveis podem ser carros de passeio, caminhões ou ônibus.

11. Dê o conjunto de dependências funcionais para o esquema relacional R(A,B,C,D) com chave primária nos atributos AB. Sabe-se que a relação está na 1FN mas não está na 2FN.

12. De_na a terceira forma normal. Dê um exemplo de uma relação em 2FN mas não em 3FN. Transforme a relação em relação em 3FN.

13. Modele a situação abaixo em um Diagrama Entidade-Relacionamento considerando:

    - O banco Dindin contratou a Hardsoft para construção de um sistema de gerenciamento da logística do banco. Abaixo estão transcritas as informações obtidas por entrevistas de levantamento de requisitos que ajudam no projeto do bando de dados do sistema.

        - O sistema deve gerenciar o cadastro de clientes, contas e transações efetuadas. A respeito do cadastro de clientes, devem ser armazenados os dados: nome, endereços (rua, no. e bairro), telefones, data de nascimento e idade.

        - O banco Dindin possui agências espalhadas em várias cidades de vários estados do país. A agência é cadastrada no sistema por nome, endereço (rua, no. e bairro) e telefones. As agências estão agrupadas em cidades. A cada cidade é atribuído um código e também é informado o estado a que pertence.

14. Considerando os conceitos de banco de dados abordados na disciplina descreva e relacione:

    A. Linguagens de banco de dados
    B. Usuários de banco de dados
    C. Arquitetura de SGBD

15. O que é uma operação Atômica?

Para as próximas questões utilize o enunciado abaixo para respostas:

CódigoCantor Nome País
Música
CódigoMúsica Nome Duração Categoria
Canta
Cantor Música DataGravação Gravadora
Gravadora
CódigoGravadora NomeGravadora País

16. Considerando as relações acima mostre:

    a) como criá-las em SQL.
    b) os comandos para inserção de pelo menos dois registros em cada relação.
    c) altere o país da gravadora Records para China.

d) supondo que existam músicas da Categoria MPB já cadastradas, altere a duração dessas músicas para 20 unidades.

17. Mostre como realizar as seguintes consultas sobre as tabelas da questão 1.

    a) Selecione o nome das músicas que não foram gravadas pelo cantor Tiririca.

    b) Selecione o nome das músicas que foram gravadas em 2000 pelo cantor Tiririca.

    c) Selecione quantas músicas foram gravadas pelo cantor Tiririca em novembro de 2000.

    d) Selecione o nome das gravadoras da França que possuem cantores da Índia.

    e) Selecione a duração total de todas as músicas do cantor Jessé para cada uma das gravadoras nas quais esse cantor trabalhou.

    f) Selecione o nome do cantor da música da gravadora Records que apresenta maior duração.

18. Uma transportadora aérea pretende implementar uma base de dados com a seguinte informação:

    A transportadora tem vários aviões. Cada avião tem, além da matrícula, um nome, o modelo do avião, o número de lugares, e a indicação da sua autonomia. Na transportadora trabalham vários pilotos. Sobre cada piloto pretendese guardar o nome e número de licença, assim como quais os modelos de aviões que podem pilotar. Pretende-se ainda, guardar a informação relativa ao nome, data de nascimento de cada um dos descendentes (caso existam) dos pilotos. Cada avião faz vários vôos. Cada vôo deve ter, pelo menos, a indicação da data e hora em que acontecerá, dos locais de partida e de destino. Cada vôo de um dado avião é pilotado por um piloto.

19. Para guardar a informação relevante à organização de uma conferência, vai ter de ser criada uma base de dados.

    Na conferência são apresentados vários artigos, cada um dos quais com um título e um número. Cada artigo tem um ou mais autores. De cada autor, pretendese armazenar além do nome, o endereço de email, e o nome e endereço da instituição a que estão associados.

    Há ainda a informação relativa aos participantes da conferência. De cada participante deverá ser retida a informação do seu nome, morada (endereço) e endereço de e-mail. Além disso, distingue-se entre os participantes os que são estudantes e os que não são. Cada participante não estudante tem de pagar antecipadamente a inscrição por transferência bancária, pelo que é necessário guardar o número da transação. Para não

pagar, o estudante tem de enviar antecipadamente um comprovante e na base de dados deve ser armazenado o nome da universidade que o passou.

20. Quais são as principais diferenças entre um Sistema de Banco de Dados e um sistema baseado em arquivos.

21. O que é o papel de uma entidade em um relacionamento? Quando é necessário especificar o papel das entidades em um relacionamento?

22. Considere o relacionamento POSSUI entre funcionário e dependente. Considere que um dependente de um funcionário possa também ser funcionário? Como o diagrama deveria ser modificado para evitar o armazenamento redundante das informações das pessoas que são tanto dependentes quanto empregados?

23. Tende em mente o modelos de dados relacional, o que você entende por relação, tupla e atributo?

24. A secretaria de uma determinada universidade precisa gerenciar informações sobre suas disciplinas, estudantes e professores. Um aluno pode estar cursando uma ou mais disciplinas, que por sua vez _e ministrada por um professor. Para cada disciplina guardam-se o seu nome, as salas onde podem ser ministradas e as disciplinas que lhe são pré-requisitos. Para cada alunos temos o nome, o RGA e o curso que está fazendo. Para cada professor, é de interesse da secretaria o seu nome, telefone e o departamento onde está alocado.

USE O ENUNCIADO ABAIXO PARA RESPONDER AS PRÓXIMAS 10 QUESTÕES:

Considere o esquema de uma base de dados sobre carros composta pelos seguintes esquemas de relação (a chave primaria de cada esquema encontra-se sublinhada). Suponha a existência de uma relação (tabela) para cada um dos esquemas abaixo e com o mesmo nome dos esquemas.

AUTOMOVEIS(C_odigo, Ano, Fabricante, Modelo, Pa__s, Prec_oTabela)
REVENDEDORAS(CGC, Nome, Propriet_ario, Cidade, Estado)
CONSUMIDORES(Identidade, Nome, Sobrenome)
NEGOCIOS(Comprador, Revenda, CodAuto, AnoAuto, Data, Prec_o)
GARAGENS(CGCRevenda, CodoAuto, AnoAuto, Quantidade)

A relação AUTOMOVEIS, cada automóvel é identificado por um código juntamente com o seu ano de fabricação. Apenas revendedoras autorizadas, ou seja, cadastradas na relação REVENDEDORAS, podem vender os carros no mercado. O CGC identifica unicamente uma revendedora. Os consumidores têm identidade única em território nacional e são cadastrados na relação CONSUMIDORES. Cada negócio efetuado é registrado na tabela NEGOCIOS, com detalhamento de data, preço pago, identidade do comprador (consumidor), revendedora, código e ano do automóvel. Por uma relação

GARAGEM determina quais automóveis as revendedoras têm a intenção de negociar e qual o seu potencial de venda a cada momento. Isto é, a quantidade de carros que podem ser negociados pelas revendedoras. Com base no esquema e descrição acima, expresse as seguintes consultas Usando SQL:

25. Listar os carros (código e ano) que custam menos do que 23.000,00.

26. Listar os nomes dos fabricantes dos automóveis na base de dados e os respectivos paises de fabricação.

27. Listar os estados onde se vende o modelo Xantia, cujo fabricante _e a Citroen.

28. Quais revendedoras não vendem automóveis de origem francesa?

29. Quais os nomes dos consumidores que compram apenas carros de 1996?

30. Listar os nomes das revendedoras, e de seus respectivos proprietários, que venderam em 1995 carros de 1996 por valor abaixo da tabela.

31. Quantos automóveis, independentes do ano de fabricação, podem ser negociados por cada revendedora?

32. Quais automóveis, de um mesmo fabricante e modelo, são colocados a venda por mais.de uma revendedora?

33. Qual o carro cujo pereço de tabela _e o mais caro?

34. Qual o valor total pago em negócios efetuados por automóvel e por revendedora?

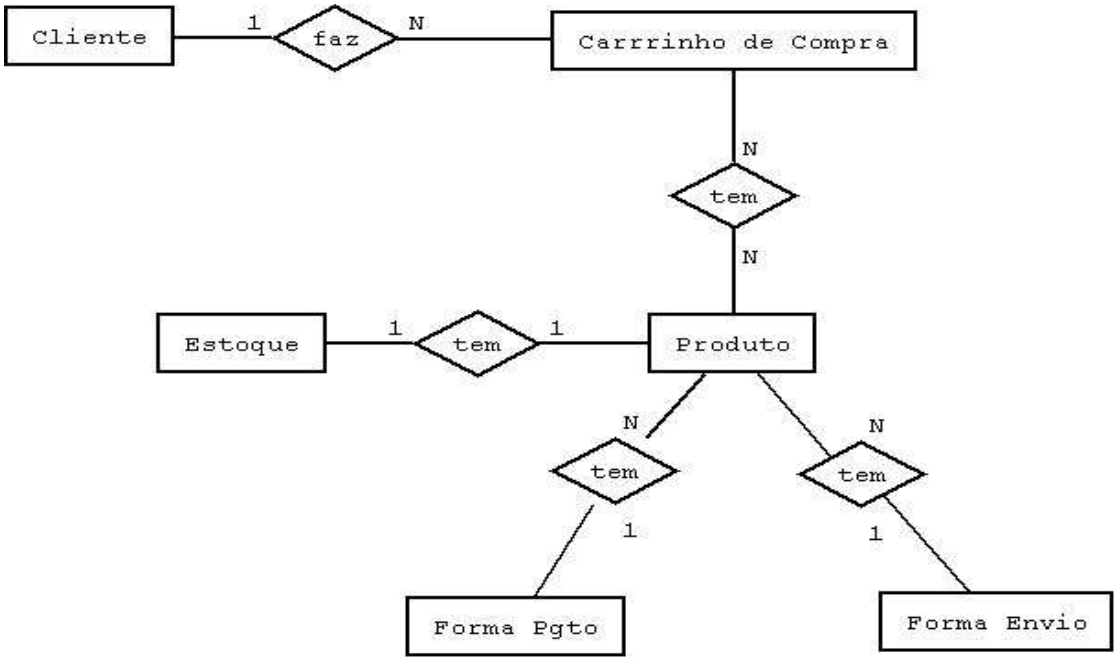
35. Crie um modelo Hierárquico para:
    a) Controlar as aulas de educação Física;
    b) Controlar a freqüência de um laboratório de Informática.

36. Escreva a SQL de criação do banco do sistema abaixo, seguinte roteiro:

    1. Criar base de dados;
    2. Usar base criada;
    3. Criar tabelas (...);
    4. Descrever tabelas (...);
    5. Destruir tabelas (...);
    6. Destruir base de dados;

SISTEMA DE COMÉRCIO ELETRÔNICO

Entidades e Atributos
· Cliente: CPF*, Nome*, Data de nascimento, Endereço*, Telefone
· Carrinho de Compra: Data*, Total*
· Produto: Nome*, Descrição*, Foto*, Preço*, Desconto*
· Estoque (do produto): Quantidade*, Quantidade Mínima*
· Formas de Envio: Tipo*, Prazo*, Preço*
· Formas de Pagamento: Data Limite*, Tipo*



37. Quais as perguntas que devem ser levantadas quanto a um projeto de BD?

38. Citar vantagens e desvantagens no uso da tecnologia de BD.

39. Citar problemas que podem ocorrer quanto ao processamento de consultas em um BD

40. Construa um modelo E-R com entidades, relacionamentos e atributos para o caso a seg

O banco de dados de uma empresa mantém informações sobre empregados, departamentos e projetos. Após a coleta de requisitos e a fase de análise, chegou-se à seguinte descrição para modelagem de dados:

- A empresa é organizada em departamentos. Cada departamento possui um nome e código únicos, além de um empregado que gerencia o departamento. O banco de dados deve armazenar a data em que o empregado passou a gerenciar o departamento.
- Cada departamento controla um certo número de projetos, cada qual com seu título e número únicos.
- Para cada empregado, armazena-se seu nome, CPF, salário, sexo e data de nascimento. Cada empregado é vinculado a um único departamento, mas pode trabalhar em vários projetos que não são necessariamente controlados pelo seu departamento. Deve-se registrar no BD a carga horária semanal do empregado em cada projeto. Também deve-se manter informação sobre o supervisor direto de cada empregado.
- Deseja-se igualmente que o BD armazene dados de dependentes de cada empregado. Para cada dependente, mantém-se seu nome, sexo, data de nascimento e relação com o empregado.

41. Defina os termos cardinalidade máxima e cardinalidade mínima.

42. Elabore um esquema de banco de dados para uma locadora de vídeo com pelo menos as seguintes entidades:

    1. Filme,
    2. Mídia (cópia de filme em fita(s) *VHS* ou *DVD*),
    3. Cliente,
    4. Reserva de filme,
    5. Movimentação de mídia (compra, venda ou locação).

USE O ENUNCIADO ABAIXO PARA RESPONDER AS PRÓXIMAS 10 QUESTÕES:

Banco de Dados de uma Revenda Internacional de Automóveis

Este banco de dados foi criado por uma revenda de automóveis com o objetivo de manter atualizado um cadastro com os modelos de carros de cada marca que cada uma das suas filiais tem disponíveis na própria loja. Para isto foram utilizadas as seguintes relações: **descrição dos modelos dos carros e suas respectivas marcas (*Modelo*)**, **descrição das marcas dos carros (*Marca*)** e **descrição das filiais da revenda (*Filial*).** A relação ***Disponíveis*** representa a disponibilidade dos carros de cada marca em cada filial da loja.

43. Obter todos os dados de todas as filiais;
44. Obter o código das marcas que estão disponíveis na filial “Filial_X”;
45. Obter os códigos dos modelos de marcas “brasileiras” que estão disponíveis nas lojas situadas no “Brasil”;
46. Obter as cores predominantes dos modelos da marca “Ford”;

47. Obter os códigos dos modelos e sua respectivas marcas, mas somente para modelos disponíveis em lojas que estejam situadas em “Curitiba”;
48. Obter os códigos das filiais que não expõem nenhum carro de cor predominante “azul” e de marcas da “Alemanha”;
49. Obter os códigos das filiais que possuem todos os carros que a marca “Ford” produz;
50. Obter os códigos daqueles modelos que encontram-se disponíveis em todas as filiais de “Porto Alegre”;
51. Obter os códigos das marcas que tem determinado modelo disponível em todas as filiais;
52. Obter os códigos das filiais que tenham disponível algum modelo de uma marca que fabrique modelos com cor predominante “vermelha”.
53. Liste todos os atributos de projeto em ordem crescente de código
54. Descubra qual a média de ganhos totais e quantos empregados são considerados para cálculo dessa média.
55. Liste o nome, a matrícula e o salário de todas as pessoas que não ganhem salário inferior a R$ 1.000,00.
56. Liste matrícula, nome, anos de serviço e salário de todas as pessoas com mais de 6 anos de serviço e que, ou seja gerente de departamento ou ganhe salário superior a R$ 3.000,00
57. Liste a matrícula, o nome e o salário de todas as pessoas em ordem decrescente de salário.
58. Liste o nome, a matrícula e o departamento de todas as pessoas que não trabalhem nos departamentos 10 e 20.
59. Liste a localização, a matrícula e o nome de todos os empregados do departamento de compra e venda.
60. Liste a matrícula, o nome e o salário de todos os coordenadores de projeto que não ganhem salário inferior a R$ 1.500,00.
61. Liste o nome e a localização de todos os departamentos subordinados à divisão “administração”. 4/4
62. Liste o nome e a matrícula de todos os funcionários alocados no projeto “holografia”.
63. Liste o nome e a matrícula de todos os gerentes de departamento e o respectivo nome do departamento gerenciado e da divisão a que são subordinados.

64. Liste o salário médio para os departamentos que possuam salário médio superior à média da companhia.
65. Liste o menor, o maior e a média de salários de cada departamento que possua salário médio superior a R$ 1.000,00.
66. Liste o salário médio dos empregados por departamento e local para grupos de mais de um empregado.
67. Liste o salário médio dos empregados por local para grupos de mais de um empregado.
68. Liste a matrícula e os nomes dos coordenadores, e matrícula dos funcionários que são supervisionados por eles.