

Modelo Relacional

BCD29008 – Engenharia de Telecomunicações

Prof. Emerson Ribeiro de Mello

mello@ifsc.edu.br

Licenciamento



Slides licenciados sob [Creative Commons "Atribuição 4.0 Internacional"](https://creativecommons.org/licenses/by/4.0/)

Esquema e instância de banco de dados

■ Esquema de banco de dados

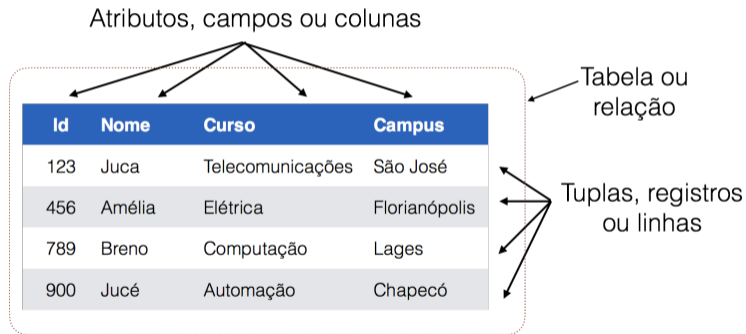
- Projeto lógico do banco de dados
- Fazendo analogia com a linguagem Java, o esquema seria equivalente a declaração de uma classe

■ Instância de banco de dados

- Situação dos dados em um banco de dados em um determinado instante no tempo
- Fazendo analogia com a linguagem Java, a instância seria equivalente a um objeto, que nada mais que é uma instância da classe

Tabela ou Relação

Em um banco de dados relacional os dados estão organizados na forma de **tabelas**, também chamadas de **relações**



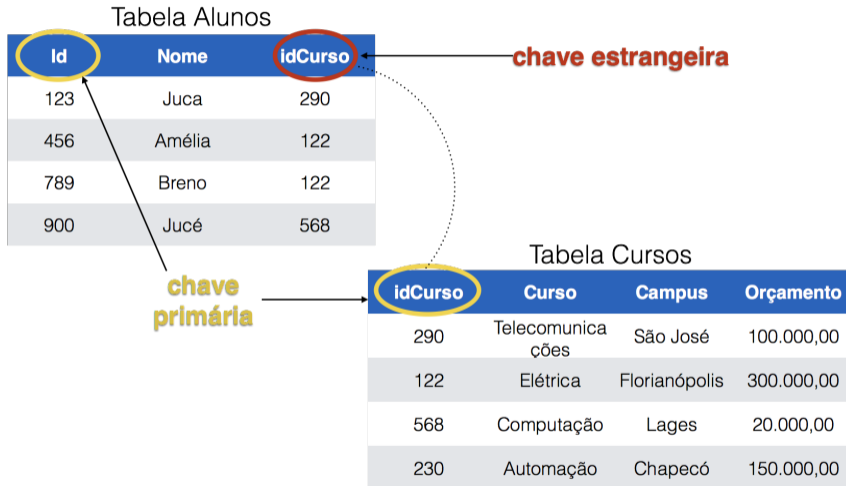
- **Tabela** é um **conjunto não ordenado de linhas** (tuplas). Cada **linha é composta** por uma série de campos (**colunas** ou atributos)

Chave em um banco de dados relacional

tem por objetivo identificar linhas e estabelecer relações entre linhas de diferentes tabelas

- Não trata-se de um índice para tornar o acesso mais rápido. Trata-se apenas de uma restrição de integridade
- **Chave primária** (*primary key* – PK)
 - Coluna ou combinação de colunas cujos valores distinguem uma linha das demais dentro de uma relação
- **Chave estrangeira** (*foreign key* – FK)
 - Coluna ou combinação de colunas cujo valores aparecerem necessariamente na chave primária de uma outra tabela
 - O **mecanismo que permite** a implementação de **relacionamentos** em banco de dados relacionais

Chaves



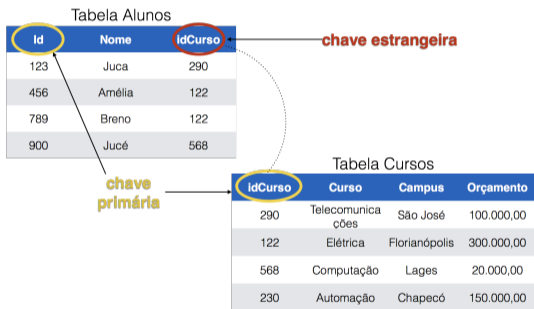
Chave estrangeira

- Uma relação r_1 pode incluir entre seus atributos a chave primária de uma outra relação, por exemplo, r_2
- Esse atributo é então chamado de **chave estrangeira** de r_1 , referenciando r_2
- r_1 é chamada de **relação referenciadora** da dependência da chave estrangeira
- r_2 é chamada de **relação referenciada** da chave estrangeira

Restrição de integridade referencial

Em qualquer instância de banco de dados, dada qualquer tupla t_a de r_1 , deverá haver alguma tupla t_b em r_2 , tal que o valor do atributo da chave estrangeira de t_a seja o mesmo valor da chave primária de t_b

Restrições impostas por chave estrangeira



- **Inclusão** de linha na tabela que contém chave estrangeira
- **Alteração** do valor **da chave estrangeira**
- **Exclusão** de linha na relação referenciada da chave estrangeira
- **Alteração** do valor **da chave primária** referenciada pela chave estrangeira

Restrições impostas por chave estrangeira

- **Inclusão de linha na tabela que contém chave estrangeira**
 - O valor a ser colocado na chave estrangeira deve obrigatoriamente aparecer na coluna chave primária da tabela referenciada
- **Alteração do valor da chave estrangeira**
 - O novo valor deve obrigatoriamente aparecer na coluna chave primária da tabela referenciada
- **Exclusão de linha na relação referenciada da chave estrangeira**
 - Deve ser garantido que na coluna chave estrangeira da relação referenciadora não apareça o valor que está sendo excluído da chave primária da tabela referenciada
- **Alteração do valor da chave primária referenciada pela chave estrangeira**
 - Na chave estrangeira da relação referenciadora não pode aparecer o valor antigo da chave primária que está sendo alterada

Domínios e valores vazios

- **Domínio do campo** é o conjunto de valores que são permitidos na referida coluna em uma tabela
 - Cadeia de caracteres, inteiro, data, ...
- Ao criar um campo em uma tabela deve-se especificar seu domínio e se a mesma poderá aceitar valores nulos (NULL)
 - Colunas obrigatórias não permitem valores nulos
 - Colunas opcionais permitem valores nulos
- Colunas que compõem **chaves primárias são colunas obrigatórias**, porém tal exigência não é necessária para chave estrangeira

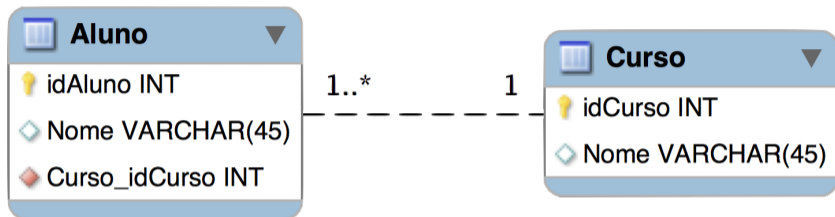
Domínios e valores vazios

- **Domínio do campo** é o conjunto de valores que são permitidos na referida coluna em uma tabela
 - Cadeia de caracteres, inteiro, data, ...
- Ao criar um campo em uma tabela deve-se especificar seu domínio e se a mesma poderá aceitar valores nulos (NULL)
 - Colunas obrigatórias não permitem valores nulos
 - Colunas opcionais permitem valores nulos
- Colunas que compõem **chaves primárias são colunas obrigatórias**, porém tal exigência não é necessária para chave estrangeira

Restrições de integridade de domínio

SGBD garante automaticamente a integridade de domínio, de valores nulos, integridade de chave e integridade referencial . O desenvolvedor de aplicação não precisa se preocupar em fazer tais verificações

Representação do esquema de banco de dados relacional



- Existem diferentes tipos de notações para representação gráfica de um esquema de banco de dados (depende da ferramenta)
- Representação acima foi feita com o MySQL Workbench
 - Notação UML para relacionamento
 - Notação MySQL Workbench simplificada para tabelas

Representação do esquema de banco de dados relacional

Representação textual

- Linguagem SQL é a linguagem padrão

```
1 CREATE TABLE Aluno (idAluno INT NOT NULL, Nome VARCHAR(45) NULL,  
2 Curso_idCurso INT NOT NULL,  
3 PRIMARY KEY (idAluno),  
4 CONSTRAINT fk_Aluno_Curso  
5 FOREIGN KEY (Curso_idCurso)  
6 REFERENCES Curso (idCurso));
```

- Representação resumida

```
1 Aluno (idAluno, Nome, idCurso)  
2 idCurso referencia Curso  
3  
4 Curso (idCurso, Nome)
```

Transformação do modelo ER para o modelo relacional

ER e Relacional

- **Modelagem ER**

- Modelo conceitual independente do SGBD

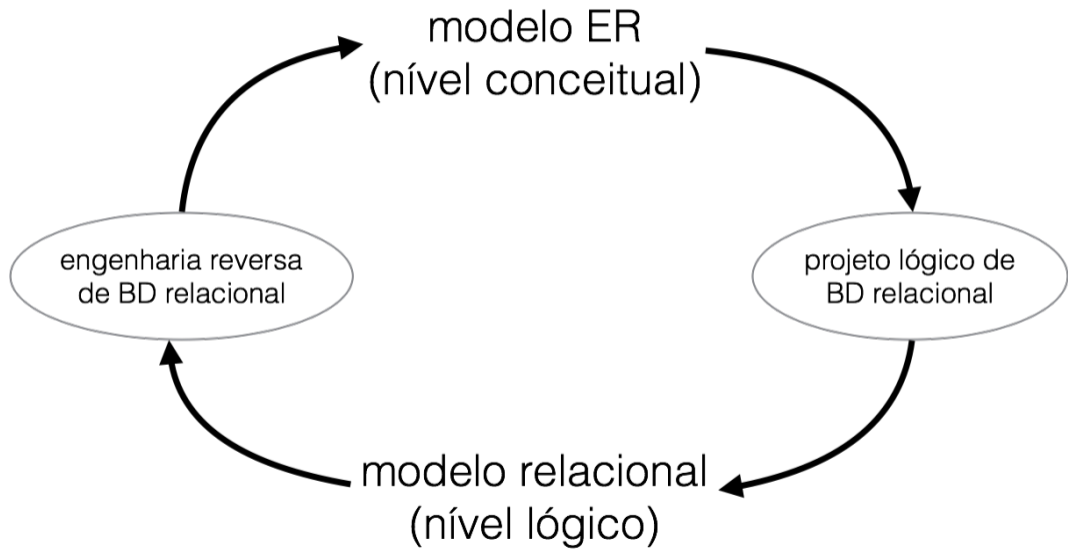
- **Modelagem Relacional**

- Modelo lógico – modela os dados no nível de SGBD

Modelo ER pode ser implementado por diferentes modelos relacionais

Diferentes modelos relacionais podem gerar desempenho, facilidades de uso e manutenção diferentes

Transformação entre modelo ER e relacional



Objetivos do projeto de BD e regras de tradução

- **Objetivos básicos de um projeto de BD**

- Bom desempenho nas operações de consulta e alteração
- Simplificar o desenvolvimento e manutenção de aplicações

Objetivos do projeto de BD e regras de tradução

■ **Objetivos básicos de um projeto de BD**

- Bom desempenho nas operações de consulta e alteração
- Simplificar o desenvolvimento e manutenção de aplicações

Regras de tradução que serão usadas nessa aula

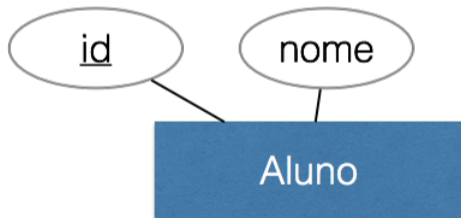
- Evitar junções (JOIN)
 - ter os dados necessários a uma consulta em uma única tabela
- Minimizar número de chaves
 - evitar a criação de índices pelo BD
- Evitar campos opcionais
 - campos com valores NULL

Processo de um projeto lógico

- 1 Implementar **entidades** e respectivos atributos
- 2 Implementar **relacionamentos** e respectivos atributos
- 3 Implementar **generalizações/especializações**

Implementar entidades

- Nome da Entidade pode ser usado como nome da tabela
- Nome dos atributos são mapeados para nome de colunas
- É recomendável que o nome da chave primária tenha como sufixo o nome da tabela



■ `Aluno(idAluno, nome)`

Implementação de relacionamentos

A cardinalidade mínima e máxima são fatores determinantes para indicar qual a tradução adequada

- Tabela própria
- Adição de colunas em uma das entidades participantes
- Fusão de tabelas de entidades

Relacionamento: Tabela própria

Obrigatório quando cardinalidade n..n

Chave primária formada pelas chaves primárias das entidades relacionadas + atributos identificadores do relacionamento

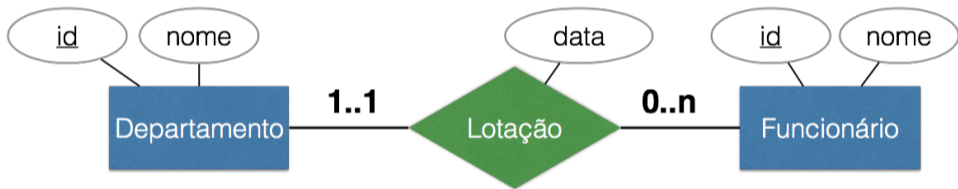


```
1 Engenheiro(idEng, nome)
2 Projeto(idProj, nome)
3 Atuacao(idEng, idProj, funcao)
4   idEng referencia Engenheiro
5   idProj referencia Projeto
```

Relacionamento: Adição de coluna

Possível quando uma das entidades possuir **cardinalidade máxima 1**

Inserir além dos atributos do relacionamento, as colunas identificadoras da entidade relacionada, definidas como chave estrangeira

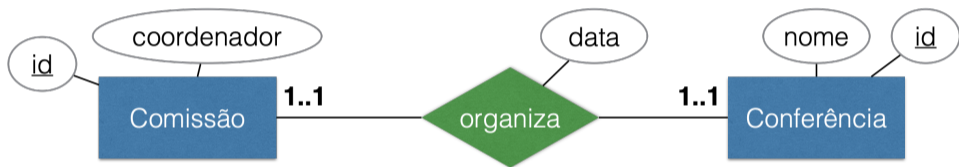


```
1 Departamento(idDepto, nome)
2 Funcionario(idFuncionario, nome, idDepto, dataLotacao)
3   idDepto referencia Departamento
```

Relacionamento: Fusão de tabelas

Possível somente quando o relacionamento é **um-para-um**

Uma única tabela combina atributos das entidades e do relacionamento



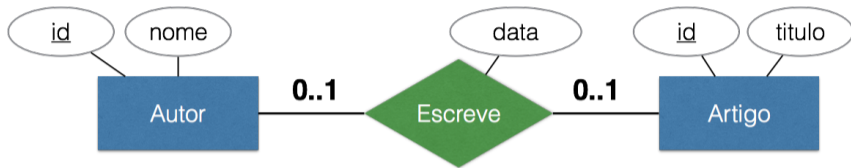
```
1 Conferencia(idConferencia, nome, data, coordenador)
```


Implementação de relacionamentos

Tipo	Tabela própria	Adição de Coluna	Fusão
Um-para-um			
0..1 \diamond 0..1	+	✓	✗
0..1 \diamond 1..1	-	+	✓
1..1 \diamond 1..1	-	-	✓
Um-para-muitos			
0..1 \diamond 0..n	+	✓	✗
0..1 \diamond 1..n	+	✓	✗
1..1 \diamond 0..n	-	✓	✗
1..1 \diamond 1..n	-	✓	✗
Muitos-para-muitos			
0..n \diamond 0..n	✓	✗	✗
0..n \diamond 1..n	✓	✗	✗
1..n \diamond 1..n	✓	✗	✗

Relacionamentos um-para-um: 0..1 – 0..1

Ambas entidades com participação opcional

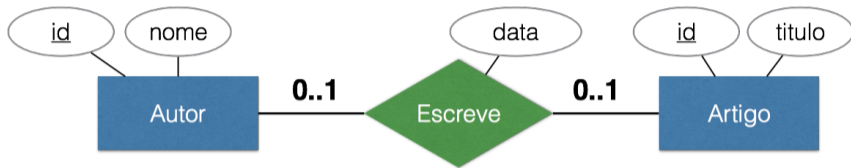


✓ **Adição de colunas** na tabela referente a qualquer uma das entidades participantes

```
1 Artigo(idArtigo, titulo, data, idAutor)
2   idAutor referencia Autor
3 Autor(idAutor, nome)
```

Relacionamentos um-para-um: 0..1 – 0..1

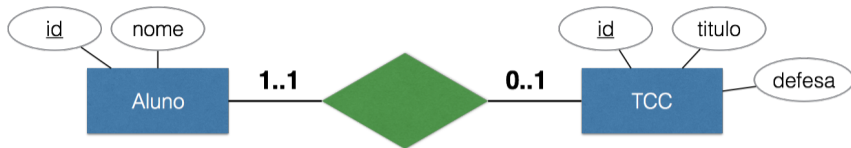
Ambas entidades com participação opcional



+ Tabela própria

```
1 Artigo(idArtigo, titulo)
2 Autor(idAutor, nome)
3 Escrita(idArtigo, idAutor, data)
4   idArtigo referencia Artigo
5   idAutor referencia Autor
```

Relacionamentos um-para-um: 0..1 – 1..1



✓ Fusão de tabelas

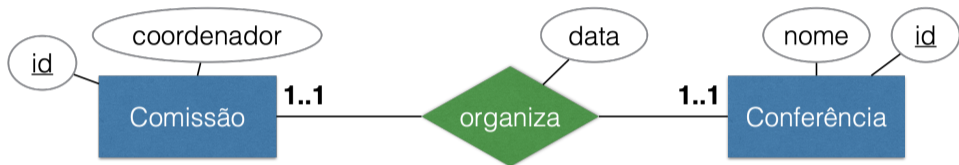
```
1 Aluno(idAluno, nome, idTCC, titulo, defesa)
```

+ Adição de colunas

```
1 Aluno(idAluno, nome)
2 TCC(idTCC, titulo, defesa, idAluno)
3   idAluno referencia Aluno
```

Relacionamentos um-para-um: 1..1 – 1..1

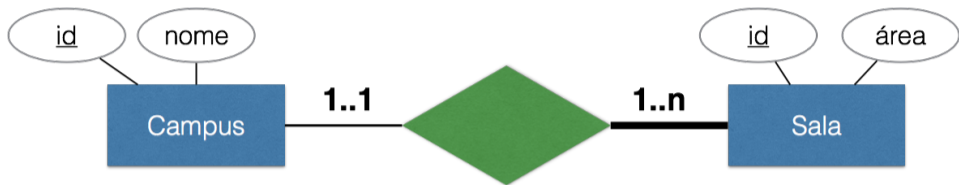
Ambas entidades tem participação obrigatória



✓ Fusão de tabelas

```
1 Conferencia(idConferencia, nome, data, coordenador)
```

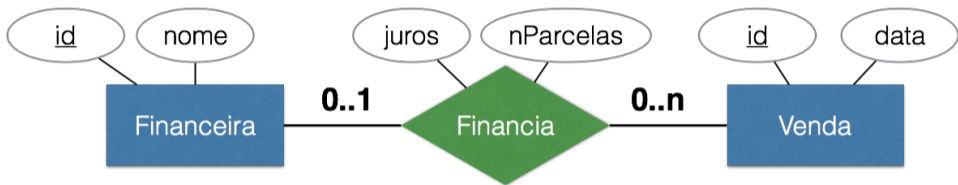
Relacionamentos um-para-muitos: 1..1 – 1..n



✓ Adição de colunas

```
1 Campus(idCampus, nome)
2 Sala(idCampus, idSala, area)
3   idCampus referencia Campus
```

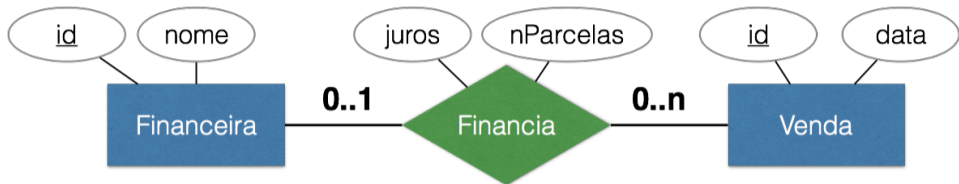
Relacionamentos um-para-muitos: 0..1 – 0..n



✓ Adição de colunas

```
1 Financa(idFinanca, nome)
2 Venda(idVenda, data, idFinanca, nParcelas, juros)
3     idFinanca referencia Financa
```

Relacionamentos um-para-muitos: 0..1 – 0..n



+ Tabela própria

```
1 Financaira(idFinancaira, nome)
2 Venda(idVenda, data)
3 Financa(idVenda, idFinancaira, nParcelas, juros)
4   idVenda referencia Venda
5   idFinancaira referencia Financaira
```


Relacionamentos muitos-para-muitos: $n - n$

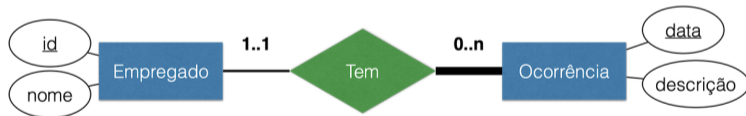
Sempre será necessário fazer com **tabela própria**



```
1 Engenheiro(idEng, nome)
2 Projeto(idProj, nome)
3 Atuação(idEng, idProj, funcao)
4   idEng referencia Engenheiro
5   idProj referencia Projeto
```

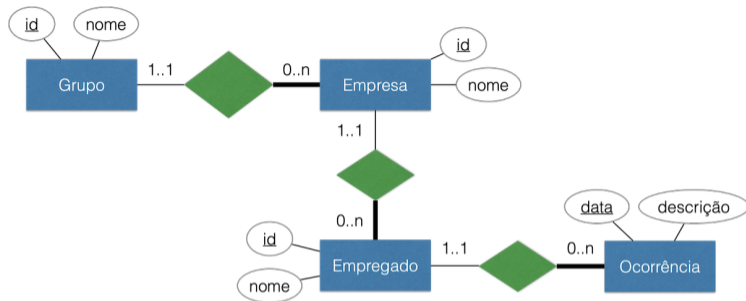
Relacionamento identificador

- Cria-se uma chave estrangeira na tabela que implementa a entidade identificada pelo relacionamento
- Na entidade fraca, a **chave primária deve ser** formada por
 - **atributos identificadores** da entidade
 - **chaves estrangeiras** que implementam os relacionamentos identificadores



```
1 Empregado(idEmpregado, nome)
2 Ocorrencia(data, idEmpregado, descricao)
3     idEmpregado referencia Empregado
```

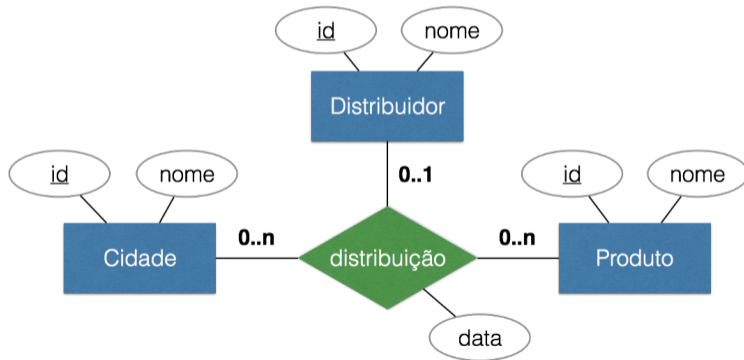
Relacionamento identificador



```
1 Grupo(idGrupo, nome)
2 Empresa(idEmpresa, idGrupo, nome)
3     idGrupo referencia Grupo
4 Empregado(idEmpregado, idEmpresa, idGrupo, nome)
5     idEmpresa, idGrupo referencia Empresa
6 Ocorrencia(data, idEmpregado, idEmpresa, idGrupo, descricao)
7     idEmpregado, idEmpresa, idGrupo referencia Empregado
```

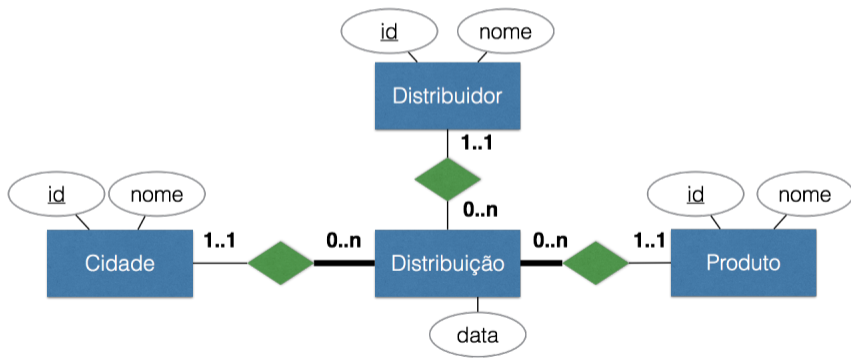
Relacionamentos de grau maior que 2

- 1 Relacionamento é transformado em entidade
- 2 Essa nova entidade é ligada por meio de relacionamento binário com cada uma das demais entidades que participavam do relacionamento

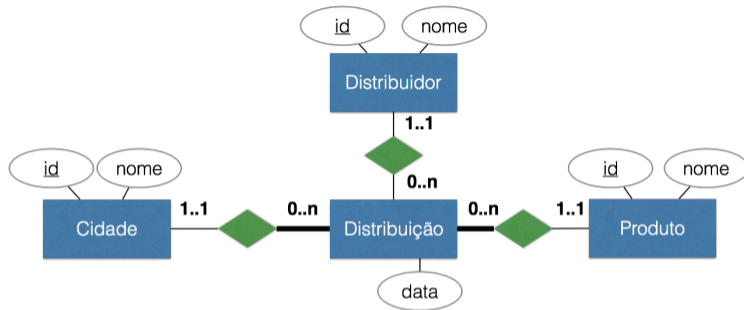


Relacionamentos de grau maior que 2

- 1 Relacionamento é transformado em entidade
- 2 Essa nova entidade é ligada por meio de relacionamento binário com cada uma das demais entidades que participavam do relacionamento

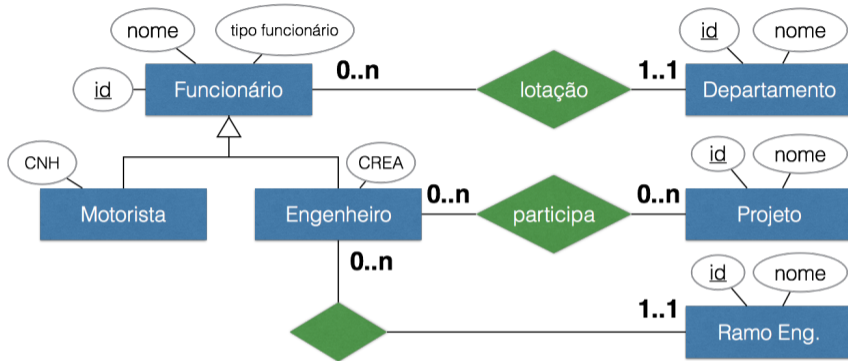


Relacionamentos de grau maior que 2



```
1 Distribuidor(idDist, nome)
2 Cidade(idCid, nome)
3 Produto(idProd, nome)
4 Distribuicao(idProd, idCid, idDist, data)
5     idProd referencia Produto
6     idCid referencia Cidade
7     idDist referencia Distribuidor
```

Generalização/Especialização



Uma tabela para toda hierarquia

Uma tabela por entidade especializada

Generalização/Especialização

Uma tabela para toda hierarquia

```
1 Funcionario(idFuncionario, nome, tipo, idDepto, CNH, CREA, idRamo)
2   idDepto referencia Departamento
3   idRamo referencia Ramo
4
5 Participa(idFuncionario, idProjeto)
6   idFuncionario referencia Funcionario
7   idProjeto referencia Projeto
```

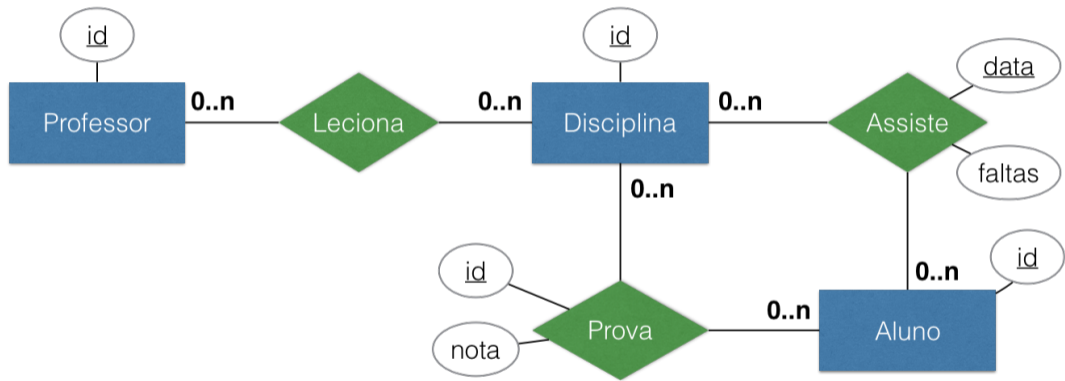

Generalização/Especialização

Uma tabela por entidade especializada

```
1 Funcionario(idFuncionario, nome, tipo, idDepto)
2     idDepto referencia Departamento
3
4 Motorista(idFuncionario, CNH)
5     idFuncionario referencia Funcionario
6
7 Engenheiro(idFuncionario, CREA, idRamo)
8     idFuncionario referencia Funcionario
9     idRamo referencia Ramo
10
11 Participa(idFuncionario, idProjeto)
12     idFuncionario referencia Engenheiro
13     idProjeto referencia Projeto
```

Exercícios

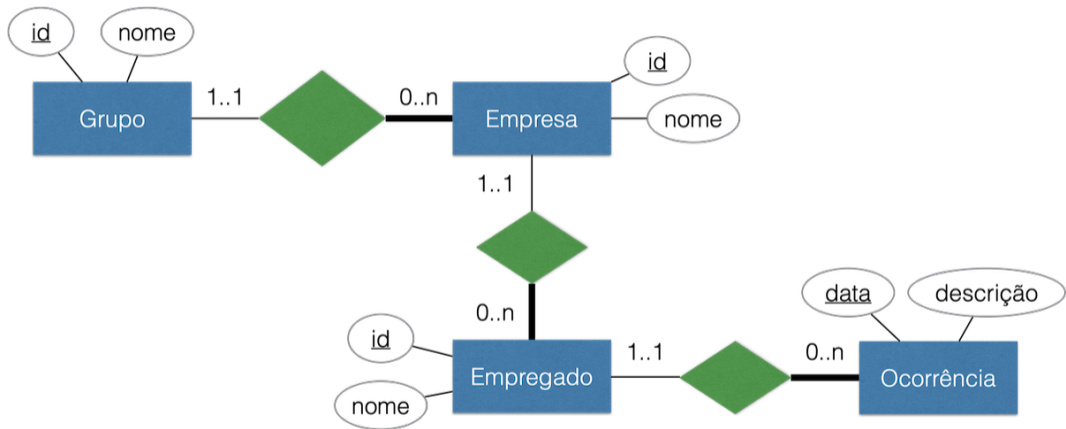
Exercício 1



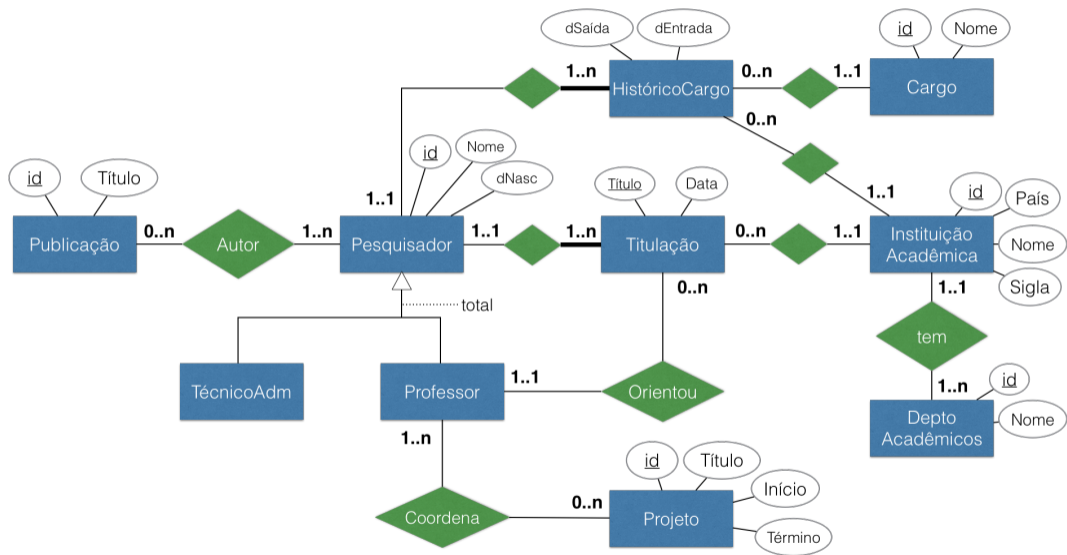
Exercício 2






Exercício 3



Exercício 4



Aulas baseadas em

-  Henry F.; Sudarshan Silberschatz, Abraham; Korth.
Sistemas de banco de dados.
6a. Edição - Editora Campus, 2012
-  Heuser, C. A.
Projeto de banco de dados
6a. Edição - Editora Bookman, 2009
-  Sullivan, D. G.
Computer Science – Harvard University