Banco de Dados

Aula 05: Relacionamento entre Tabelas - Primary Key e Foreign Key

Roteiro da Aula

- Relacionamentos entre tabelas através de chaves:
  - Primary Key
  - Foreign Key
- Comando ALTER TABLE

Modelo Relacional - Chaves

- No modelo relacional a única forma de relacionar dados que existem em uma tabela com dados que existem em outra tabela é através de atributos comuns.
- Assim, vão existir atributos especiais(chaves estrangeiras) que servem para fazer a ligação com outras tabelas, onde esses mesmos atributos são usados para identificar, unicamente, cada uma das linhas(chaves primárias)

Modelo Relacional - Chaves

- Tipos de Chaves
  - Chave Primária: chave que identifica cada tupla(linha)
  - Chave Estrangeira: atributo ou conjunto de atributos de uma relação, que é chave primária em outra relação

Modelo Relacional - Chave Primaria

- Uma chave primária(atributo identificador) é **uma coluna ou um grupo de colunas que assegura a unicidade das linhas dentro de uma tabela**;
- Uma chave primária que tenha mais de uma coluna é chamada de chave primária composta;
- Chaves primárias são geralmente indicadas pela sigla **PK** (**primary key**)
- Exemplos de Chaves Primárias:
  - Produto (Codigo_Produto, Descrição, ...)
  - Veiculo (Chassi, Cor, Tipo, ...)
  - Usuario (Login,Senha, ...)
  - Conta_Bancaria(Agencia,Conta_Corrente, Titular)

Modelo Relacional - Chave Primaria

- **Regras:**
  1. Valores de chave primária não podem ser nulos (***not null***):
     - Valores de chaves primarias não podem ser nulos porque uma linha sem chave primaria não se distingue de outras linhas da mesma tabela;
  2. Colunas com chaves primarias não podem ter valores duplicados
     - Um valor duplicado é um valor exatamente igual a outro de uma ou mais linhas da mesma coluna na mesma tabela;
     - Se a coluna possui valores duplicados esta não pode servir de identificador da linha;

Modelo Relacional - Chave Primaria

- **Recomendações:**
  1. Selecione chaves primarias absolutamente disciplinadas e que permaneçam únicas;
  2. Selecione chaves primarias que não tenham qualquer tendência a alterações;
  3. Se possível seleciona chaves primarias simples;
  4. De preferência a colunas numéricas para chaves primarias;
  5. Selecione chaves primarias que sejam familiares;
  6. Se não houver nenhuma coluna com chave primaria candidata, utilize chaves primarias atribuídas pelo sistema (autonumeração).

Modelo Relacional - Chave Estrangeira

- Uma chave estrangeira é uma **coluna ou grupo de colunas que pode ou não ser chave primária da tabela em questão, mas, com certeza é chave primária de outra tabela**;
- Uma chave estrangeira formada por mais de uma coluna é chamada de chave estrangeira composta. Chaves estrangeiras são indicadas pela sigla **FK (Foreign Key)**
- Exemplos de Chave Estrangeira:
  - Dependentes (Codigo_Dependente, **Codigo_Socio**)
  - Pedido (Nota_Fiscal, **Codigo_Cliente**)

Modelo Relacional - Chave Estrangeira

- Em todo relacionamento entre entidades (MER) e entre tabelas (MR), deverá existir uma chave estrangeira que identifique a qual tupla ela está relacionada na tabela onde ela é a chave primária.
- No exemplo abaixo, na implementação para o modelo relacional, a tabela FUNCIONÁRIO deverá possuir uma chave estrangeira que identifique a qual DEPARTAMENTO ele está lotado. A chave estrangeira de FUNCIONÁRIO é uma chave primária na tabela DEPARTAMENTO.

Constraint PRIMARY KEY

- Como estudamos, toda entidade(tabela) deverá possuir um atributo único e não nulo que identifique um registro. Esse atributo é chamado de chave primária ou **primary key**.
- A *constraint* **primary key** é equivalente às cláusulas **NOT NULL** + **UNIQUE** juntas
- Em uma tabela deve-se existir apenas uma *constraint* **primarykey**, enquanto podem existir variadas ocorrências da cláusula **UNIQUE**

Constraint PRIMARY KEY

```
CREATE TABLE Dados_Pessoais
(
Nome CHAR(60) UNIQUE,
CPF CHAR(15),
Tempo_Servico INTEGER CHECK(Tempo_Servico >= 0)
Primary Key(CPF)
)
```

- Exemplos:

```
CREATE TABLE Comissao
(
Matricula_Funcionario INTEGER,
Codigo_Compra INTEGER,
Valor_Compra NUMERIC(10,2) CHECK(Valor_Compra >= 1000)
Primary Key(Matricula_Funcionario,Codigo_Compra)
)
```

Constraint REFERENCES

- A constraint REFERENCES permite fazer a validação das chaves estrangeiras. Isto é, não se podem inserir nos campos referenciados como chaves estrangeira valores que não existam na tabela onde os campos são chave primária.
- Veja Exemplo

```
CREATE TABLE PESSOA
(
Nome CHAR(60) UNIQUE,
CPF CHAR(15),
Idade INTERGER,
Telefone CHAR(15),
Codigo_Postal NUMERIC(10) REFERENCES Postal(Codigo),
Primary Key(CPF)
)
```

Cardinalidade dos Relacionamentos

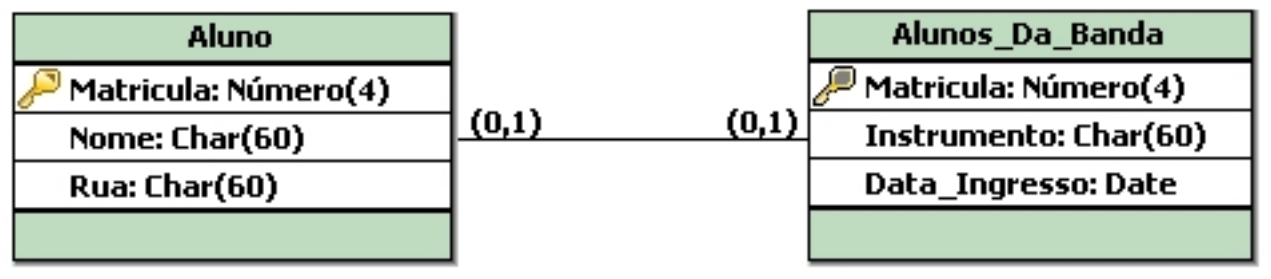
- Estudamos que em um banco de dados, precisamos de alguma maneira para representar estes relacionamentos da vida Real, em termos das tabelas e de seus atributos.
- Isto é possível com a utilização de "Relacionamentos entre tabelas", os quais podem ser de três tipos:
  - Um para Um
  - Um para Vários
  - Vários para Vários

Relacionamento 1:1

- Relacionamento Um para UM
  - Esta relação existe quando os campos que se relacionam são ambos do tipo Chave Primária, em suas respectivas tabelas.
  - Cada um dos campos não apresenta valores repetidos.
  - Na prática existem poucas situações onde utilizaremos um relacionamento deste tipo.

Relacionamento 1:1

- Imagine uma escola com um Cadastro de Alunos na tabela Alunos, destes apenas uma pequena parte participa da Banda da Escola. Por questões de projeto do Banco de Dados, podemos criar uma Segunda Tabela "Alunos da Banda", a qual se relaciona com a tabela Alunos através de um relacionamento do tipo Um para Um.
- Cada aluno somente é cadastrado uma vez na Tabela Alunos e uma única vez na tabela Alunos da Banda. Poderíamos utilizar o Campo Matrícula do Aluno como o Campo que relaciona as duas Tabelas.



Relacionamento 1:1

- Codificação em SQL

```
CREATE TABLE Alunos
(
Matricula NUMERIC(4),
Nome CHAR(60),
Rua CHAR(60),
Primary Key(Matricula)
)
```

```
CREATE TABLE Alunos_Da_Banda
(
Matricula NUMERIC(4) REFERENCES Alunos(Matricula),
Instrumento CHAR(60),
Data_Ingresso DATE,
Primary Key(Matricula)
)
```

Relacionamento 1:N

- Relacionamento Um para Vários:
  - É o tipo de relacionamento mais comum entre duas tabelas.
  - Uma das tabelas (o lado um do relacionamento) possui um campo que é a Chave Primária e a outra tabela (o lado vários) se relaciona através de um campo cujos valores relacionados podem se repetir várias vezes.

Relacionamento 1:N

- Considere o exemplo entre a tabela Clientes e Pedidos. Cada Cliente somente é cadastrado uma única vez na tabela de Clientes (por isso o campo Código do Cliente, na tabela Clientes, é uma chave primária, indicando que não podem ser cadastrados dois clientes com o mesmo código), portanto a tabela Clientes será o lado um do relacionamento.

Relacionamento 1:N

- Ao mesmo tempo cada cliente pode fazer diversos pedidos, por isso que o mesmo Código de Cliente poderá aparecer várias vezes na tabela Pedidos: **tantas vezes quantos forem os pedidos que o Cliente tiver feito**. Por isso que temos um relacionamento do tipo Um para Vários entre a tabela Clientes e Pedidos, através do campo Código do Cliente, indicando que **um mesmo Cliente pode realizar diversos (vários) pedidos**.

Relacionamento 1:N

- Representação Relacional de CLIENTE E PEDIDO

Relacionamento 1:N

- Codificação em SQL

```
CREATE TABLE Cliente
(
CPF NUMERIC(4),
Nome CHAR(60),
Telefone CHAR(15),
Email CHAR(60),
Primary Key(CPF)
)
```

```
CREATE TABLE Pedido
(
Numero_Pedido NUMERIC(4),
CPF NUMERIC(4) REFERENCES Cliente(CPF),
Data DATE,
Valor NUMERIC(4,
Primary Key(Numero_Pedido)
)
```

Relacionamento N:N

- Relacionamento Vários para Vários:
  - Este tipo de relacionamento **"aconteceria"** em uma situação onde os dois lados do relacionamento, os valores poderiam se repetir.
  - Vamos considerar o caso entre Produtos e Pedidos. Posso ter **Vários Pedidos** nos quais aparece um determinado produto, além disso **vários Produtos** podem aparecer no mesmo Pedido.
  - **Esta é uma situação em que temos um Relacionamento do Tipo Vários para Vários.**

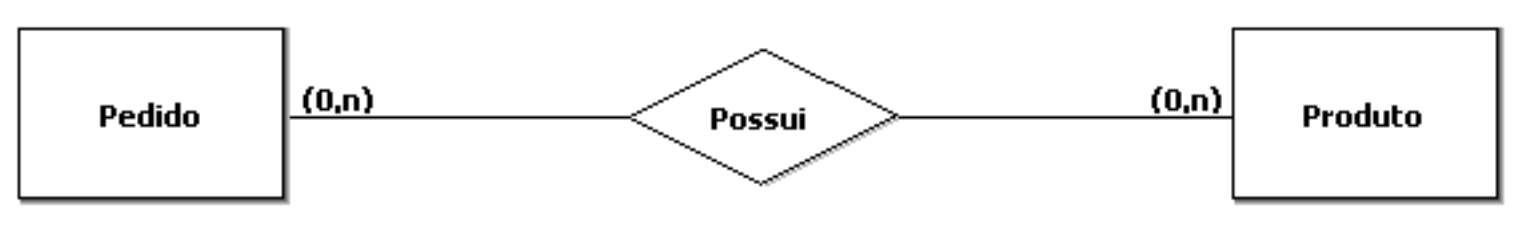
Relacionamento N:N

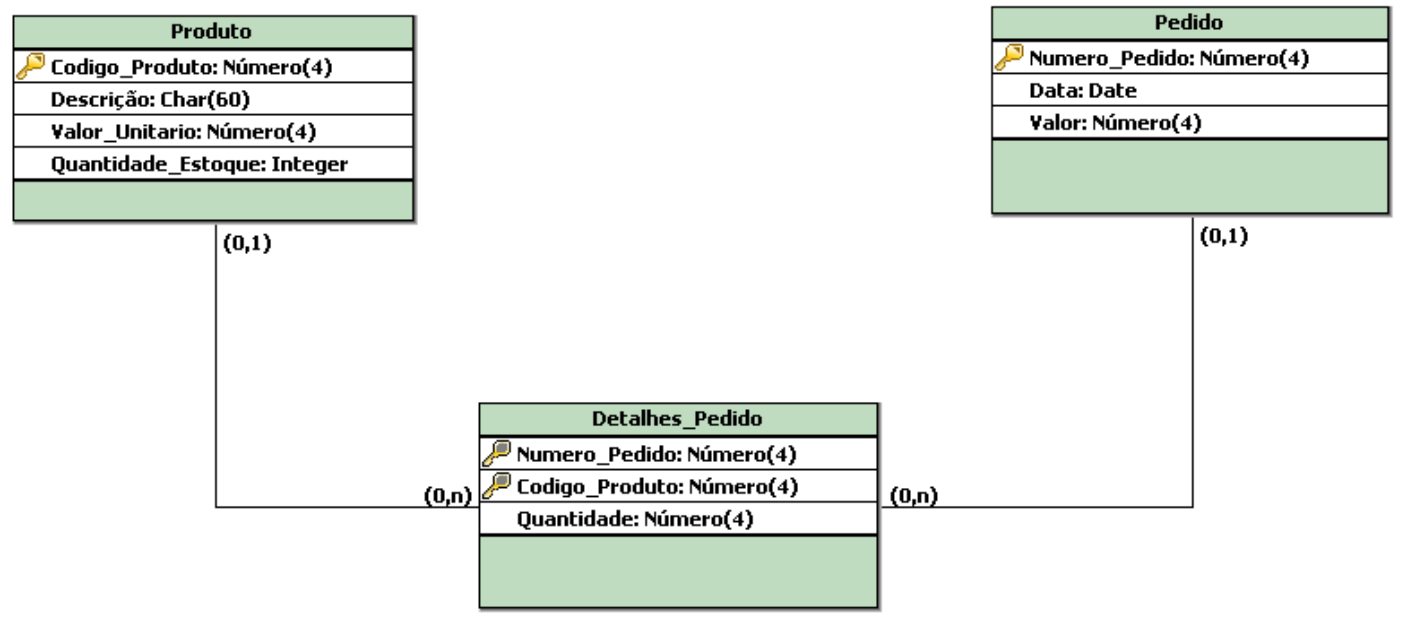
- Na prática não é possível implementar um relacionamento deste tipo, devido a uma série de problemas que seriam introduzidos no modelo do banco de dados.
- Por exemplo, na tabela Pedidos teríamos que repetir o Número do Pedido, Nome do Cliente, Nome do Funcionário, Data do Pedido, etc para cada item do Pedido.

Relacionamento N:N

- Para evitar este tipo de problema um relacionamento do tipo Vários para Vários é quebrado em dois relacionamento do tipo Um para Vários.
- Isso é feito através da criação de uma nova tabela, a qual fica com o lado Vários dos relacionamentos.
- No nosso exemplo vamos criar a tabela Detalhes do Pedido, onde ficam armazenadas as informações sobre os diversos itens de cada pedido, aí ao invés de termos um relacionamento do tipo Vários para Vários, teremos dois relacionamentos do tipo um para vários, conforme descrito pela próxima tabela.

Relacionamento N:N





Relacionamento N:N

Codificação em SQL

```
CREATE TABLE Produto
(
Codigo_Produto NUMERIC(4),
Descricao CHAR(60),
Valor_Unitario NUMERIC(4),
Quantidade_Estoque NUMERIC(4),
Primary Key(Codigo_Produto)
)
```

```
CREATE TABLE Pedido
(
Numero_Pedido NUMERIC(4),
Data DATE,
Valor NUMERIC(4,
Primary Key(Numero_Pedido)
)
```

```
CREATE TABLE Detalhes Pedido
(
Codigo_Produto NUMERIC(4) REFERENCES Produto(Codigo_Produto),
Numero_Pedido NUMERIC(4) REFERENCES Pedido(Codigo_Pedido),
Quantidade NUMERIC(4),
Primary Key(Codigo_Produto,Numero_Pedido)
)
```

Comando ALTER TABLE

- O comando ALTER TABLE permite alterar a estrutura de uma tablea.
- Com o comando é possível adicionar uma nova coluna, modificar uma coluna já existente ou eliminar uma coluna.
- Veja a sintaxe dos comandos:

```
ALTER TABLE Nome_Da_Tabela ADD Nome_Coluna Tipo_Coluna
```

```
ALTER TABLE Nome_Da_Tabela MODIFY Nome_Coluna Tipo_Coluna
```

```
ALTER TABLE Nome_Da_Tabela DROP Nome_Coluna Tipo_Coluna
```

Comando ALTER TABLE

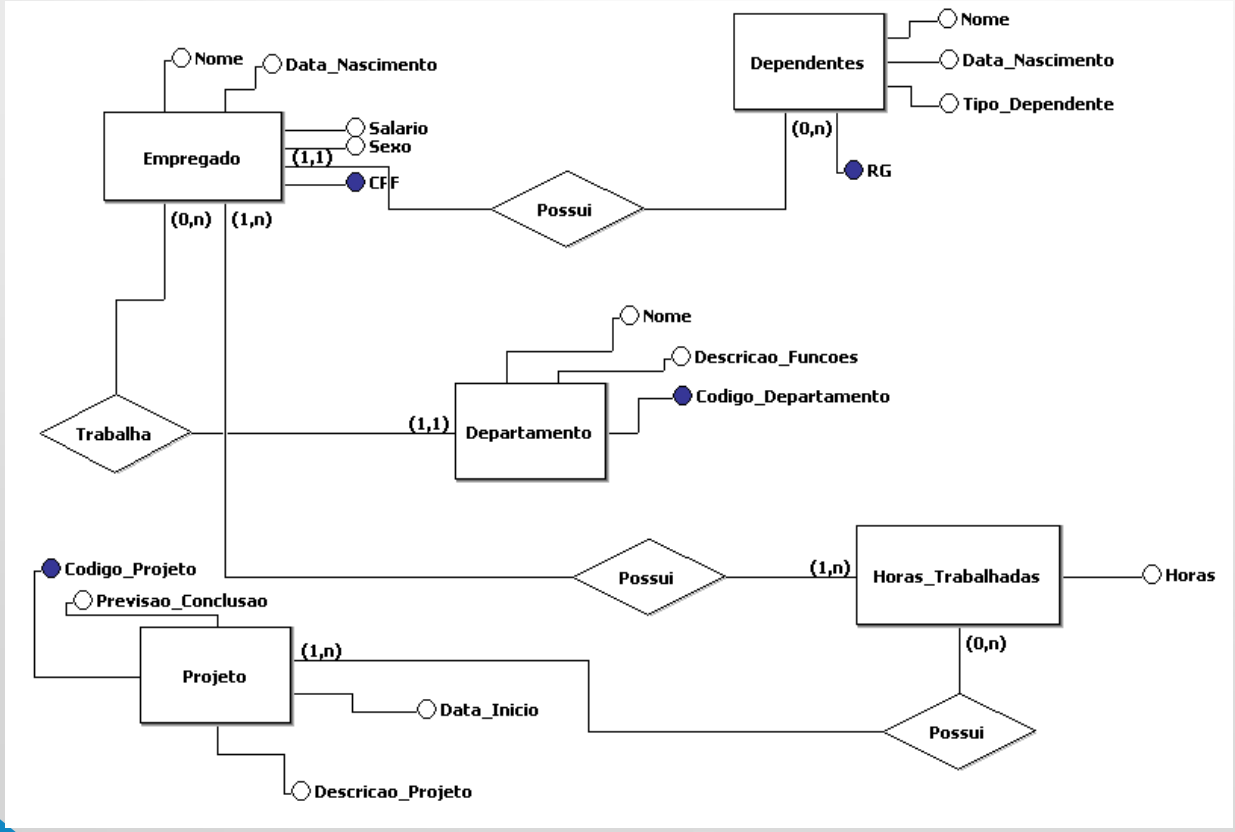
- Exemplos:

```
ALTER TABLE Fornecedor ADD Fax CHAR(15)

ALTER TABLE Fornecedor MODIFY Logradouro CHAR(60)

ALTER TABLE Funcionario MODIFY Salario NOT NULL

ALTER TABLE Funcionario MODIFY Matricula UNIQUE

ALTER TABLE Fornecedor DROP COLUMN Caixa_Postal
```

OBSERVAÇÃO:
Alguns SGBDs podem não disponibilizar a opção MODIFY. Em caso de dúvida consulte o Manual de Referência.

Exemplo Prático 01

- A partir do MER, construir o modelo relacional.



Exemplo Prático 02

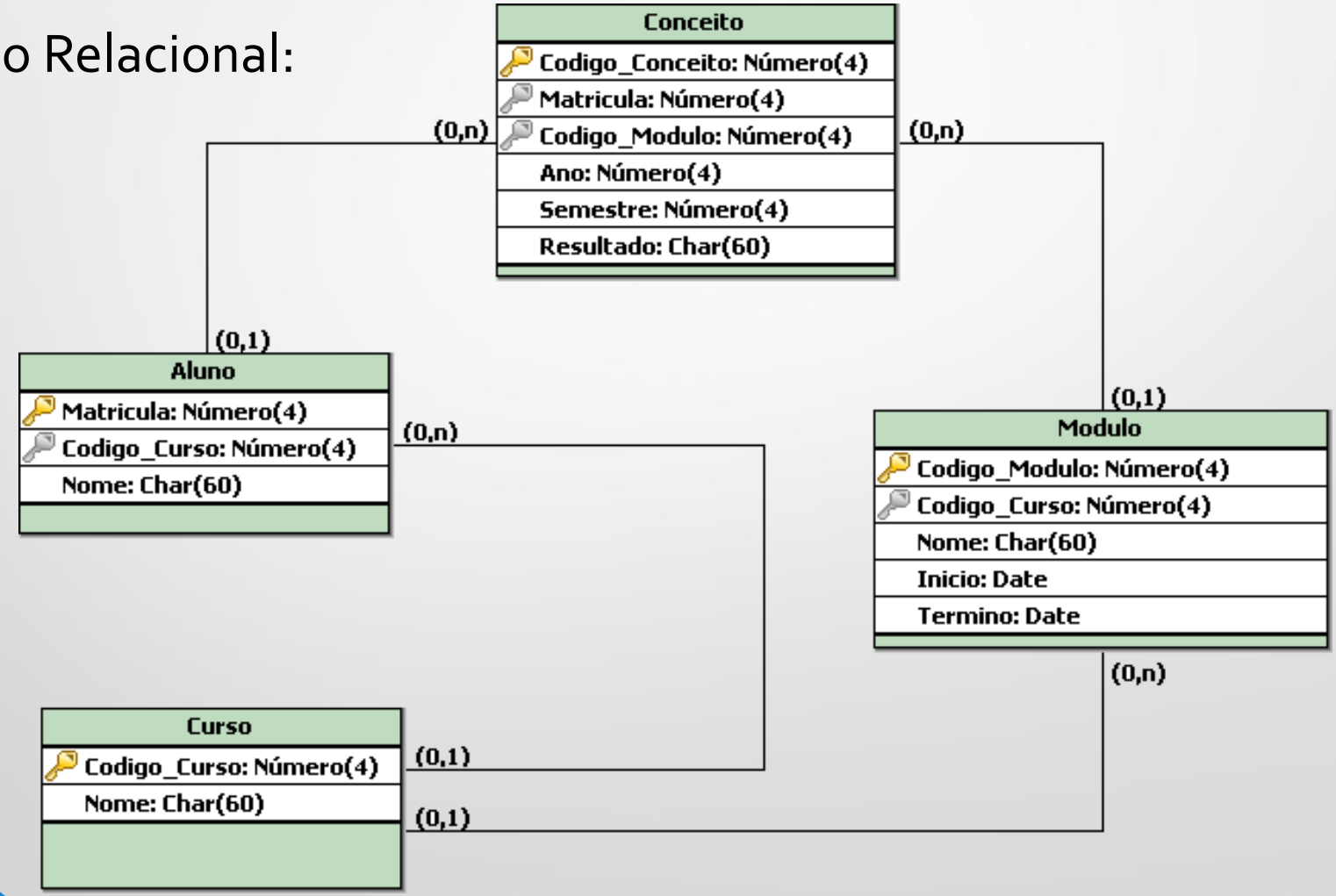
- A partir da implementação do banco de dados para o slide anterior, realizar as seguintes alterações:
  - Adicionar a Tabela DEPENDENTES o campo SEXO
  - Remover da Tabela PROJETO o campo Previsao_Conclusao e adicionar o campo Data_Termino
  - Remover da tabela DEPENDENTES o campo Tipo_Dependente e adicionar o campo Parentesco
  - Modificar o campo Nome, da tabela DEPARTAMENTO para que aceita apenas nomes únicos.

Exemplo Prático 03

- Implementar através de comandos SQL o banco de dados para a situação abaixo:
  - Uma escola contém vários cursos, onde cada aluno possui uma matricula num determinado curso.
  - Estes cursos, por sua vez, possuem módulos, aos quais serão atribuídos resultados finais a cada aluno.
- A seguir será apresentado uma modelagem física para a situação proposta.

Exemplo Prático 03

- Modelo Relacional:



Exemplo Prático 04

- A partir da base de dados criada anteriormente, realizar as seguintes configurações:
  - Adicione na Tabela Cursos, os campos Carga_Horaria e Objetivos
  - Altere o campo Nome, na tabela Cursos para que não aceite valores vazios e que seja único

Próxima Aula

- Lista de Exercícios Criação e Manutenção de Tabelas no SQL SERVER e MYSQL