

Aula 6

Professor:

Claudio Esperança

Criando um banco de dados

Conteúdo:

- 6.1 Relações, relacionamentos e esquemas
- 6.2 Modelagem de banco de dados
- 6.3 Modelo e Diagrama Entidade-Relacionamento
- 6.4 Projetando um modelo físico
- 6.5 Criação de tabelas
- 6.6 O programa mysql
- 6.7 Autenticação e Privilégios

6.1 Relações

- SQL foi criado para permitir a manipulação de dados em Sistemas de Gerenciamento de Bancos de Dados Relacionais
- Nesses sistemas, um banco de dados consiste de um conjunto de tabelas ou *relações*
- Uma relação nada mais é do que uma coleção de registros (ou *linhas* da tabela) cada qual num mesmo formato
 - ▶ Isto quer dizer que todas as linhas têm o mesmo número e tipo de colunas
 - ▶ Uma coluna de uma tabela também é chamado de *atributo*
 - ▶ Cada atributo tem um nome e um *tipo*
 - Alguns tipos comuns são cadeias de caracteres, números inteiros, números decimais, datas

Exemplos de Relações

- Um banco comercial pode ter em seu banco de dados duas relações: uma para os dados cadastrais de seus clientes e outras com dados das contas

► Cliente

Nome	Endereço
Pedro da Silva	R Padre Miguel 100, ap 201
Julia de Medeiros	Av dos Independentes 91, casa 2
Carlos Gonçalves	R Rocha Castro 210, ap 903

► Conta

Numero	Saldo
101	310.45
2102	9934.05
1234	1201.91

6.1 Esquemas e Relacionamentos

- O esquema de um banco de dados consiste de:
 - ▶ Nomes das relações, seus atributos e seus tipos
 - ▶ Relacionamentos entre as relações
 - Quais atributos de uma relação se referem a outras relações
- No exemplo anterior, seria importante dizer que conta pertence a qual cliente
 - ▶ Para expressar esse relacionamento, podemos modificar o esquema de tal forma que:
 - Clientes sejam identificados por um número (`Id`) armazenado numa coluna adicional
 - Contas tenham mais uma coluna (`IdCliente`) indicando a que cliente pertencem

Exemplo com relacionamentos

- Cliente

Id	Nome	Endereco
21	Pedro da Silva	R Padre Miguel 100, ap 201
90	Julia de Medeiros	Av dos Independentes 91, casa 2
23	Carlos Gonçalves	R Rocha Castro 210, ap 903

- Conta

Numero	Saldo	IdCliente
101	310.45	90
2102	9934.05	21
1234	1201.91	23

6.2 Modelagem de Banco de Dados

- Um banco de dados pressupõe seu uso por uma ou mais aplicações
- O desenho de um banco de dados
 - ▶ Requer conhecer bem o domínio das aplicações
 - ▶ Permite especificar o esquema do banco de dados
- O projeto de um banco de dados pode ser auxiliado por diversos tipos de ferramentas de modelagem
 - ▶ Modelo Entidade-Relacionamento (ER)
 - ▶ Linguagem Unificada de Modelagem (UML)
- Uma vez projetado o esquema de um banco de dados
 - ▶ Implementa-se o esquema criando-se as tabelas
 - ▶ Aplicações podem ser escritas

6.3 Modelo Entidade-Relacionamento

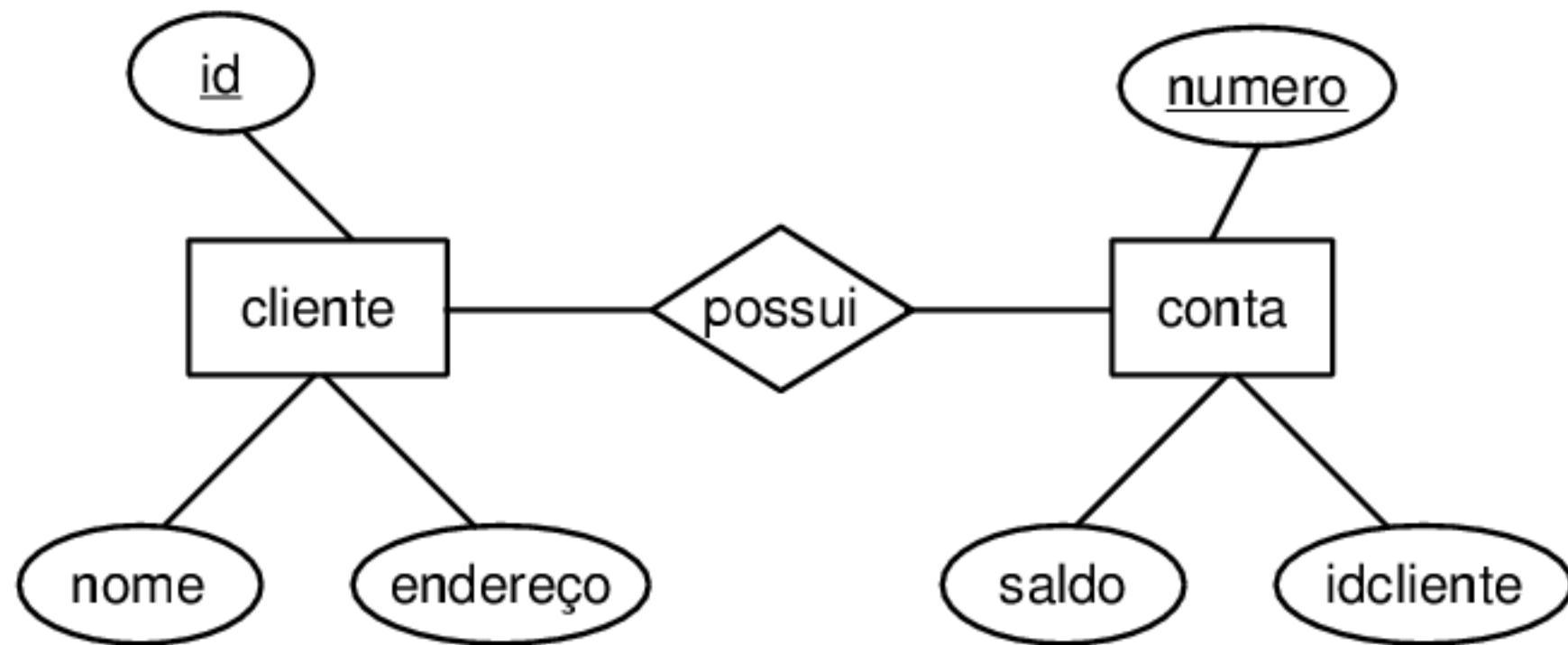
(Abordado com profundidade na disciplina de banco de dados)

- Ferramenta para modelagem *conceitual* de dados
- Permite representar graficamente esses modelos através de um *diagrama*
- Informação a ser representada é mapeada, por exemplo, por uma *análise de requisitos*
- Modelo conceitual mais tarde mapeado num modelo físico
 - ▶ Modelo físico é frequentemente *relacional* (esquema)
 - ▶ Mapeamento realizado usando técnicas tais como *normalização*

6.3 Diagrama Entidade-Relacionamento

- Entidades são objetos do domínio de aplicação
 - ▶ Normalmente pensamos em entidades como *substantivos*
 - ▶ São desenhadas nos diagramas como retângulos
- Relacionamentos capturam como entidades se relacionam
 - ▶ São associadas a *verbos*
 - ▶ Desenhadas como losangos
- Atributos são propriedades que podem estar associadas tanto a entidades como a relacionamentos
 - ▶ São desenhadas como elipses
- Toda entidade precisa possuir uma *chave primária*
 - ▶ Atributo ou conjunto de atributos identifica unicamente a entidade
 - ▶ Uma entidade que não possui uma chave primária é dita *fraca*

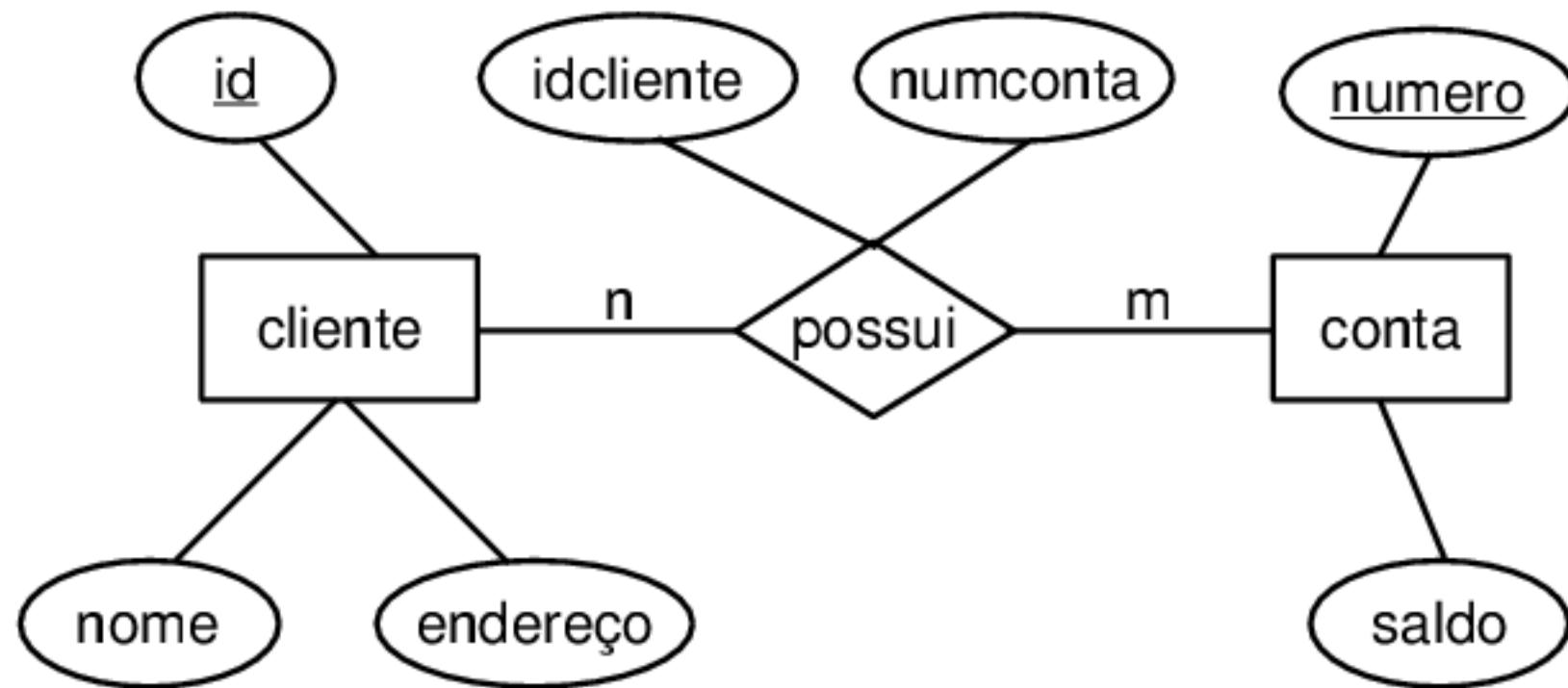
Exemplo de Diagrama



Cardinalidades num Relacionamento

- Designam quantos elementos de cada entidade participam de um relacionamento
 - ▶ É desenhado como um número ou uma letra próxima à linha que une o relacionamento com a entidade
- No nosso exemplo, podemos considerar que:
 - ▶ Cada cliente tem exatamente uma conta (1 para 1)
 - ▶ Cada cliente pode ter várias contas (1 para n)
 - ▶ Cada conta pode pertencer a vários clientes mas um cliente não pode ter mais que uma conta (n para 1)
 - ▶ Cada conta pode pertencer a vários clientes e cada cliente pode ter várias contas (n para m)
- A última hipótese é a mais geral, mas como implementá-la?
- Uma idéia é acrescentar a possui dois atributos:
 - ▶ `idcliente` indica um cliente que participa de uma conta
 - ▶ `numeroconta` indica a conta na qual ele participa

Exemplo com cardinalidades



6.4 Projetando um Modelo Físico

- Pudemos ver como uma modelagem conceitual pode revelar detalhes importantes sobre o domínio
- Após a modelagem conceitual, devemos criar um modelo físico que, para bancos de dados relacionais, consiste de um *esquema*
- No nosso exemplo, o modelo conceitual sugere que precisamos de 3 relações com os seguintes atributos
 - ▶ cliente: id, nome, endereço
 - ▶ possui: idcliente, numconta
 - ▶ conta: numero, saldo
- Em geral, a transformação de modelo conceitual em físico pode não ser tão simples

Tipos de atributos

- Antes de criar as tabelas, precisamos definir os tipos dos atributos (colunas)
- Alguns tipos comuns suportados pelo MySql são:

Tipo	Descrição
int(<i>compr</i>)	Um número inteiro escrito com <i>compr</i> dígitos
float(<i>compr, dec</i>)	Um número decimal escrito com <i>compr</i> dígitos e <i>dec</i> casas decimais
timestamp(<i>compr</i>)	Atributo para marcação de tempo. Pode ser alterado explicitamente ou automaticamente cada vez que a linha da tabela (registro) é modificada. Neste caso, guarda o momento em que a modificação ocorreu. <i>Compr</i> varia a forma em que o atributo é impresso.
char(<i>compr</i>)	Uma string com <i>compr</i> caracteres preenchida com brancos à direita
varchar(<i>compr</i>)	Uma string com comprimento variável contendo até <i>compr</i> caracteres

Tipos Variantes

- Tipos inteiros suportam variantes
 - ▶ ints são inteiros armazenados em 4 bytes pelo MySQL, mas é possível ter inteiros armazenados em 8 bytes usando atributos do tipo bigint ou em 2 bytes usando atributos do tipo smallint

Nome	Bytes	Mínimo	Máximo
TINYINT	1	-128	127
SMALLINT	2	-32768	32767
MEDIUMINT	3	-8388608	8388607
INT	4	-2147483648	2147483647

- O SQL define também o tipo decimal (*precisão, dec*) que deve ser usado para quantidades fracionárias que devem ser armazenadas com um número fixo de casas de precisão

6.5 O comando **create table**

- A parte do SQL que permite criar e modificar a estrutura de um banco de dados é chamada de linguagem de definição de dados
- Um dos comandos fundamentais dessa linguagem é o comando **create table**
 - ▶ Permite definir uma nova tabela no banco de dados corrente
- Ele tem três partes:
 - ▶ Após `create table` escreve-se o nome da tabela a ser criada
 - Não deve conter ponto (.) ou barra inclinada (/)
 - ▶ Segue-se entre parênteses uma lista de nomes de atributos, seus tipos e *modificadores*
 - ▶ Finalmente, após os atributos, uma lista de especificações de *chaves*

Exemplo

```
create table cliente (
    id int default 0 not null auto_increment,
    nome varchar(50) not null,
    endereco varchar(80),
    primary key (id),
    key nomecliente(nome)
);
create table conta (
    numero int default 0 not null auto_increment,
    saldo decimal(16,2) default 0.0,
    primary key (numero)
);
create table possui (
    idcliente int not null,
    numconta int,
    primary key (idcliente,numconta)
);
```

Modificadores de atributos

- **not null** significa que dados não podem ser adicionados à tabela sem que esse atributo receba um valor explícito
- **default** vem seguido de um valor que é assumido sempre que um atributo não recebe um valor explícito
- **auto_increment** significa que o valor é gerado automaticamente de forma a ser único na tabela
 - ▶ É usado para atributos como o `id` usado na tabela `cliente`
 - ▶ Apenas um atributo de cada tabela pode ter esse modificador
- **zerofill** preenche com zeros à esquerda tipos numéricos
- **unsigned** indica que o número não pode ser negativo

Especificações de chaves

- Uma chave é um atributo ou combinação de atributos que unicamente identifica uma linha (registro) da tabela
- A especificação `primary key` permite designar a chave de uma tabela
 - ▶ Em geral, é aconselhável que toda tabela possua uma chave
 - ▶ No exemplo, a tabela `cliente` tem como chave o atributo `id`, enquanto que a tabela `conta` possui requer a combinação de `idcliente` e `numconta` para identificar um relacionamento entre conta e cliente
- A especificação `key` indica a criação de um índice com o nome dado para o conjunto de atributos especificado entre parênteses
 - ▶ Na tabela `cliente`, a especificação `key nomecliente(nome)` significa que um índice chamado `nomecliente` vai ser usado para pesquisar cliente por nome

Manipulando Bancos de Dados

- Um banco de dados é projetado para ser manipulado por uma ou mais aplicações
 - ▶ Normalmente, aplicações utilizam uma biblioteca apropriada que transmite comandos SQL para o servidor
 - ▶ Em PHP, a interface com SGBDs MySQL é feita através de funções como `mysql_connect`, `mysql_query`, etc
- Um banco de dados pode também ser manipulado por uma aplicação genérica
 - ▶ Usuário digita consultas em SQL e examina resultados
 - ▶ Interface via linha de comando, gráfica ou web
 - ▶ Ex.: MySql possui os programas chamado `mysql` (para consultas genéricas) e `mysqladmin` (para administração)

6.6 O programa mysql

- É invocado usando a sintaxe

`mysql opções banco`

... onde *banco* é o nome (opcional) do banco de dados a ser usado
e *opções* incluem (entre outras):

► `-h host`

- Indica que o programa se comunicará com o servidor MySql da máquina *host* ao invés da máquina local default, isto é, localhost

6.6 O programa mysql

► -u *username*

- Indica que o usuário *username* será o autor das consultas
- Por default, o mysql assume *usuario@localhost* onde *usuario* é o login do usuário digitando o comando
- Caso *username* esteja protegido por senha, esta deverá ser especificada usando a opção -p

► -psenha

- Indica a *senha* correspondente ao *username* especificado com -u
- Se o parâmetro opcional *senha* não for especificado, o programa perguntará pela senha interativamente

6.7 Autenticação

- MySQL usa um sistema de autenticação diferente daquele do seu sistema operacional
- Quando o MySQL é instalado numa máquina, é definido um usuário chamado `root` que tem plenos privilégios para modificar qualquer banco de dados
 - ▶ O usuário `root` não está inicialmente protegido por senha e portanto o comando `mysql -u root` permite acesso ilimitado sem senha
 - ▶ Normalmente, é recomendado que se especifique imediatamente uma senha para esse usuário:
`mysqladmin -u root password 'suasenha'`
- Os dados dos usuários registrados para usar o servidor MySQL assim como todas as informações sobre os privilégios de cada um estão armazenados em tabelas de um banco de dados especial chamado `mysql`

6.7 Privilégios

- MySQL possui um esquema de privilégios em vários níveis
 - ▶ Global : consiste em realizar operações de administração do servidor e operações globais em bancos de dados
 - Ex: Criar um novo banco de dados ou desligar o servidor
 - ▶ Banco de dados: consiste em realizar operações sobre um banco de dados específico
 - ▶ Tabela: operações sobre uma tabela específica
 - ▶ Coluna: operações sobre colunas específicas de uma tabela
- Para cada nível, é possível especificar quais operações são permitidas
 - ▶ Por exemplo: `create` (criar), `alter` (alterar), `select` (consultar)
 - ▶ Algumas operações só fazem sentido para alguns níveis
 - Por exemplo, a operação `shutdown` (desligar o servidor) só faz sentido no nível global

Preparando o ambiente

- Os exemplos práticos que iremos adotar neste curso pressupõem a existência de um usuário chamado `aluno` com senha igual a `aluno` e que possui todos os privilégios para manusear um banco de dados chamado `prog2`
- Para criar este usuário e banco de dados chame o programa `mysql` como usuário `root` (com senha `polocederj`) e digite os comandos:

```
create database prog2;
grant all privileges on prog2.* to
    'aluno'@'localhost' identified by 'aluno';
grant all privileges on prog2.* to
    'aluno'@'%' identified by 'aluno';
```

- Uma vez criado o ambiente, é possível agora criar as tabelas do exemplo dentro do banco de dados `prog2` e utilizando o usuário `aluno`

Exemplo de criação do ambiente

```
esperanc@claudio:~$ mysql -u root -ppolocederj
Welcome to the MySQL monitor.  Commands end with ; or \
Your MySQL connection id is 54 to server version:
      4.0.24_Debian-10ubuntu2.3-log
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the
buffer.
```

```
mysql> create database prog2;
Query OK, 1 row affected (0.00 sec)
```

```
mysql> grant all privileges on prog2.* to
      'aluno'@'localhost' identified by 'aluno';
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> grant all privileges on prog2.* to 'aluno'@'%'
      identified by 'aluno';
```

```
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> quit
```

```
Bye
```

Abrir mysql

Exemplo de criação das tabelas

```
esperanc@claudio:~$ mysql -u aluno -paluno prog2
[ ... ]
```

```
mysql> create table cliente (
    ->     id int not null auto_increment,
    ->     nome varchar(50) not null,
    ->     endereco varchar(80),
    ->     primary key (id),
    ->     key nomecliente(nome)
    -> );
```

```
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> create table conta (
    ->     numero int not null auto_increment,
    ->     saldo decimal(16,2) default 0.0,
    ->     primary key (numero)
    -> );
```

```
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> create table possui (
    ->     idcliente int not null,
    ->     numconta int,
    ->     primary key (idcliente,numconta)
    -> );
```

```
Query OK, 0 rows affected (0.02 sec)
```

[Abrir mysql](#)