

# Banco de Dados

Gerando o Esquema Físico  
SQL

Professor: Paulo Sérgio Ruiz Del Aguila

## Projeto do Banco de Dados

- Projeto Conceitual
  - Modelo Entidade Relacionamento
  - Entrada: Minimundo
  - Saída: Esquema Conceitual
- Projeto Lógico
  - Modelo Relacional
  - Entrada: Esquema Conceitual
  - Saída: Esquema Lógico
- Projeto Físico
  - Modelo SQL DDL do SGBD a ser usado
  - Entrada: Esquema Lógico
  - Saída: Esquema Físico SQL DDL do SGBD usado

## BRModelo

- Ferramenta freeware voltada para o ensino de modelagem de banco de dados relacional.
- Criada como TCC de pós-graduação em banco de dados pela UFSC.

3

## Instalação

- Para instalar o BRModelo basta fazer o download do arquivo “brModelo.exe” em: <https://sourceforge.net/projects/brmodelo/>
- Extrair o arquivo em uma pasta qualquer.
- Executar o arquivo brModelo.exe na pasta dist.

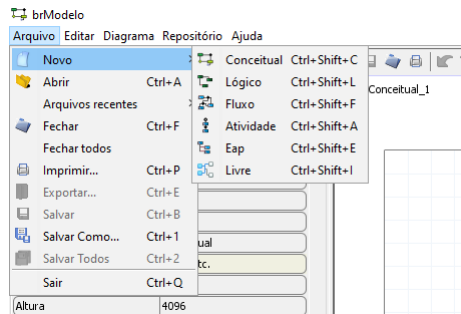


brModelo.exe

4

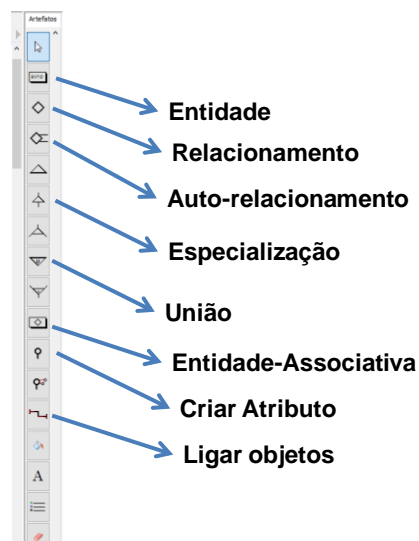
## Criando um novo modelo

- Para criar um novo modelo conceitual vá em:  
Arquivo → Novo → Conceitual




5

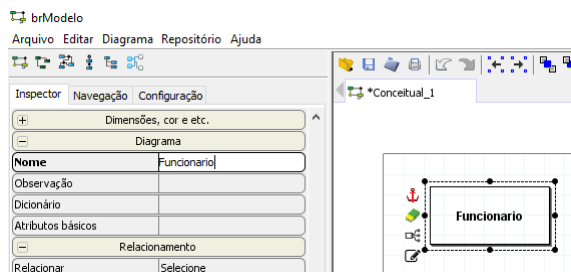
## Barra de ferramentas



6


## Criando uma entidade

- Clique no botão de Entidade  na barra lateral e clique na área de trabalho.
- Para alterar o nome da entidade, selecione-a e mude o campo “Nome” na guia da esquerda.





7

## Adicionando atributos

- Clique no botão de atributo  na barra lateral e clique na entidade que irá receber o atributo.
- Para alterar o nome do atributo, selecione-o e mude o campo “Nome” na guia da esquerda.

8

## Adicionando relacionamentos

- Clique no botão Criar Relacionamento  na barra lateral e depois na área de trabalho.
- Para alterar o nome do relacionamento, selecione-o e mude o campo “Nome” na guia da esquerda.
- Em seguida clique no botão Ligar Objetos  e ligue o relacionamento criado às entidades desejadas.


9

## Cardinalidades

- Para definir a cardinalidade de um relacionamento, selecione a cardinalidade do lado do relacionamento e modifique o campo “Cardinalidade”.
  - (0,1) → relacionamento não obrigatório com cardinalidade 1.
  - (0,n) → relacionamento não obrigatório com cardinalidade n.
  - (1,1) → relacionamento obrigatório com cardinalidade 1.
  - (1,n) → relacionamento obrigatório com cardinalidade n.

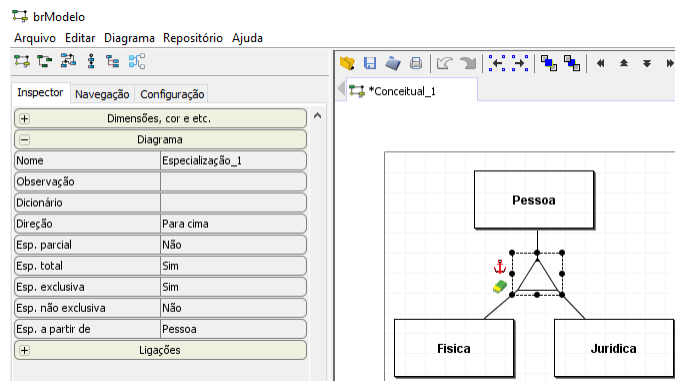
10

## Herança

- Para criar um especialização de uma entidade, clique no botão Especialização  na barra lateral.
- Em seguida clique na entidade que desejar criar a especialização.
- Por padrão, já serão criadas duas entidades (subclasses).

11

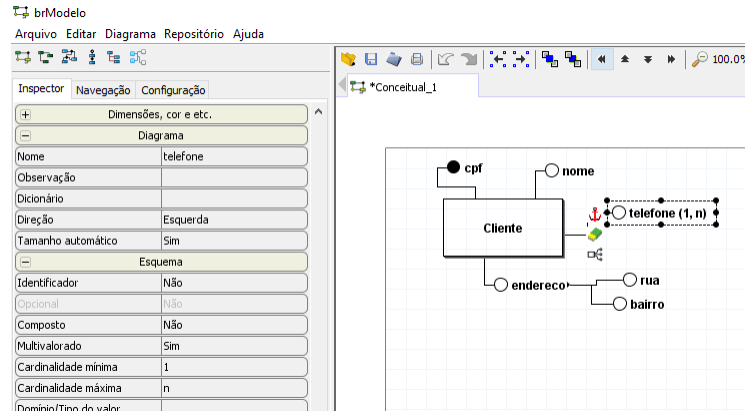
## Herança



12

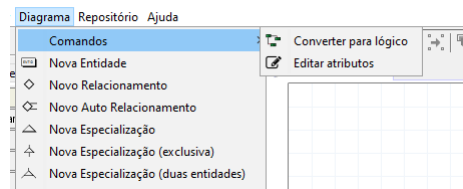
# Atributos

- Compostos e Multivalorados



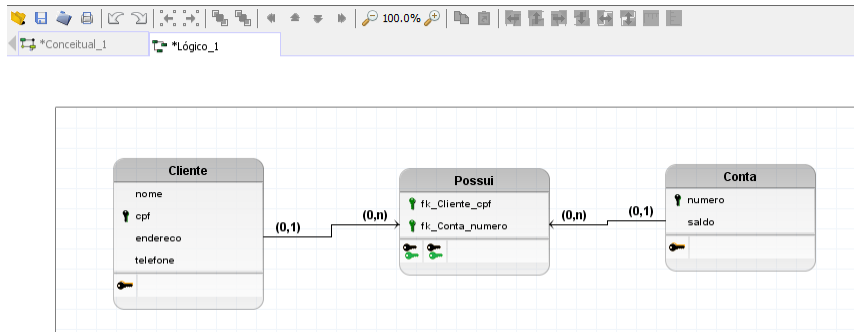
13

# Gerando Esquema Lógico



14

## Esquema Lógico



15

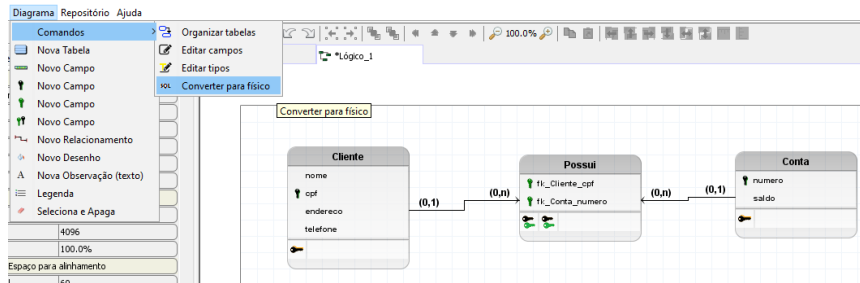
## Esquema Lógico

- O esquema lógico gerado deve ser revisado, para ser corrigido segundo as regras de mapeamento E-R/Relacional, como visto em aula.

16



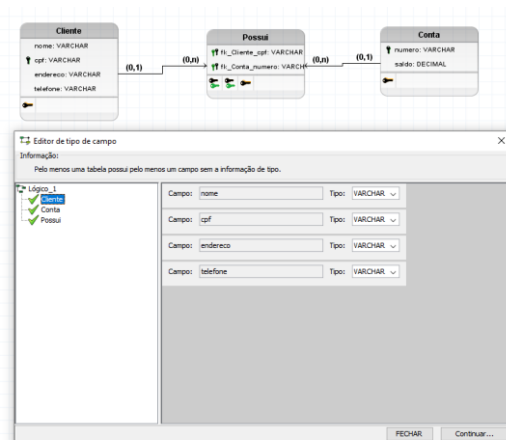
## Gerando Esquema Físico



17

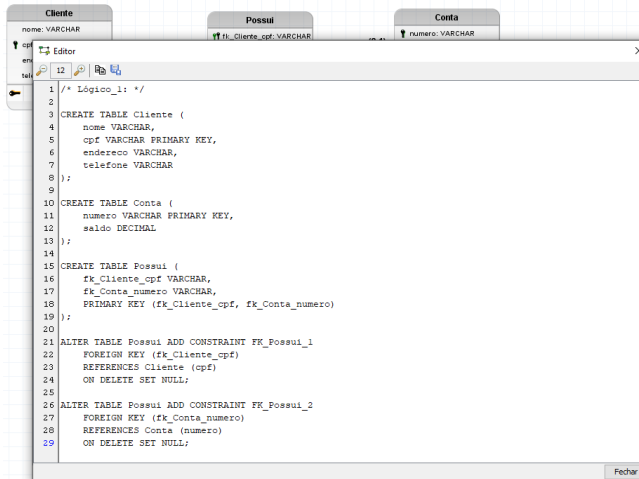
## Gerando Esquema Físico

- Setando os tipos dos campos



18

## Gerando Esquema Físico



```

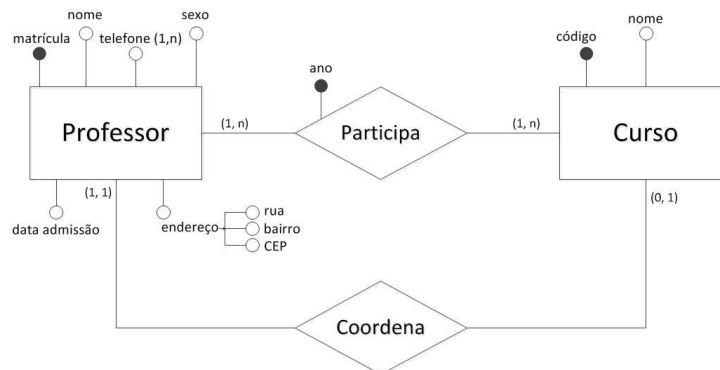
1  /* Lógico_1: */
2
3  CREATE TABLE Cliente (
4      nome VARCHAR,
5      cpf VARCHAR PRIMARY KEY,
6      endereco VARCHAR,
7      telefone VARCHAR
8  );
9
10 CREATE TABLE Conta (
11     numero VARCHAR PRIMARY KEY,
12     saldo DECIMAL
13 );
14
15 CREATE TABLE Possui (
16     fk_Cliente_cpf VARCHAR,
17     fk_Conta_numero VARCHAR,
18     PRIMARY KEY (fk_Cliente_cpf, fk_Conta_numero)
19 );
20
21 ALTER TABLE Possui ADD CONSTRAINT FK_Possui_1
22 FOREIGN KEY (fk_Cliente_cpf)
23 REFERENCES Cliente (cpf)
24 ON DELETE SET NULL;
25
26 ALTER TABLE Possui ADD CONSTRAINT FK_Possui_2
27 FOREIGN KEY (fk_Conta_numero)
28 REFERENCES Conta (numero)
29 ON DELETE SET NULL;

```

19

## Exemplo

- Gerar o esquema físico SQL DDL do esquema conceitual abaixo:



20

## SQL

- SQL – (Structured Query Language) Linguagem de Consulta Estruturada.
- Linguagem que permite não apenas consultar, como também definir (DDL) e manipular (DML) dados.
- DDL – Data Definition Language.
- DML – Data Manipulation Language.

21

## Uso de SQL

- DDL
  - Criar (CREATE)
  - Destruir (DROP)
  - Modificar (ALTER)
- DML
  - Consultar (SELECT)
  - Inserir (INSERT)
  - Remover (DELETE)
  - Atualizar (UPDATE)

22

## Criação de Tabelas

- ```
CREATE TABLE <nome da tabela>
    (<descrição dos atributos>
    <descrição das chaves>
    <descrição das restrições>);
```
- Descrição dos atributos → <nome> <tipo>
  - Tipos de dados: integer, decimal, numeric, real, varchar, char, text, timestamp, date, time, etc...

23

## Criação de Tabelas

- Descrição das chaves
  - A chave primária deve ser declarada como  
 CONSTRAINT tabela\_pkey  
 PRIMARY KEY (<atributos>)
  - Lista das chaves estrangeiras na forma  
 CONSTRAINT tabela\_fkey  
 FOREIGN KEY (<atributo>)  
 REFERENCES <tabela> (<atributo>)

24

## Criação de Tabelas

- Descrição de restrições
    - Só admite valor único
- CONSTRAINT tabela\_const  
UNIQUE (nome)

Ex. CPF varchar(10) UNIQUE;

25

## Criação de Tabelas

- Descrição de restrições
    - Não admite valores nulos
- CONSTRAINT tabela\_const  
nome NOT NULL

Ex. NOME varchar(30) NOT NULL;

26

## Remoção de Tabelas

- Elimina a tabela que foi previamente criada
- **TEM QUE SER USADO COM CUIDADO!**

`DROP TABLE <tabela>;`

- Exemplo:

`DROP TABLE Professor;`

27

## Comando SELECT

- Selecionando atributos  
`SELECT <lista de atributos> FROM <tabela>;`
- Ex: Listar matricula e nome de todos os professores  
`SELECT matricula, nome FROM Professor;`

28

## Comando SELECT

- Selecionando todos os atributos  
SELECT <lista de atributos> FROM <tabela>;
- Ex. Listar todo o conteúdo de Professor  
SELECT \* FROM Professor

29

## Comando SELECT

- Selecionando tuplas da tabela com condição  
SELECT <lista de atributos> FROM <tabela> WHERE  
<condição>;
  - Onde condição  
<nome atributo> <operador> <valor>
  - Operadores relacionais  
=, <>, <, <=, >, >=
  - Operadores lógicos  
AND, OR e NOT

30

## Comando SELECT

- Exemplo:

```
SELECT nome, telefone, rua, bairro, cep FROM  
Professor WHERE matricula = "123";
```

```
SELECT * from Professor WHERE bairro = "Alvorada"
```

31

## Operadores SQL

- BETWEEN e NOT BETWEEN: substituem o uso dos operadores  $\leq$  e  $\geq$

```
... 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 Salario BETWEEN 1000 AND 2000;
```

32



## Operadores SQL

- LIKE e NOT LIKE: só se aplicam sobre atributos do tipo char. Operam como = e < >, utilizando o símbolo % (substitui uma palavra)  
...WHERE <nome atributo> LIKE <valor1>;
- Exemplo: Listar os empregados que têm como primeiro nome José  
SELECT Nome FROM Empregado  
WHERE Nome LIKE 'José%';

33

## Operadores SQL

- IN e NOT IN: procuram dados que estão ou não contidos em um dado conjunto de valores (Interseção e Diferença de conjuntos)  
... WHERE <nome atributo> IN <valores>;
- Exemplo: Listar todos os atributos de Carro com as cores 'branco' ou 'preto'  
SELECT \* FROM Carro WHERE cor IN ('branco', 'preto');

34

## Operadores SQL

- IS NULL e IS NOT NULL: identificam valores nulos dos atributos  
... WHERE <nome atributo> IS NULL;
- Exemplo: Listar os dados dos projetos que não tenham local definido  
SELECT \* FROM Projeto WHERE Local IS NULL;

35

## Ordenação

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

36

## Ordenação

- Exemplos:
  - Listar todos os empregados ordenados ascendentemente por nome  
`SELECT * FROM Empregado ORDER BY Nome;`
  - Listar todos os empregados ordenados decendentemente por salário  
`SELECT * FROM Empregado ORDER BY Salario DESC;`

37

## Funções agregadas

- Utilização de funções sobre conjuntos MAX, MIN, SUM, AVG, COUNT disparadas a partir do SELECT
- Exemplos:
  - Mostrar o valor do maior salário dos empregados e o nome do empregado  
`SELECT Nome, Salario FROM Empregado  
WHERE Salario IN (SELECT MAX (Salario)  
FROM EMPREGADO);`

38

## Funções agregadas

- Mostrar qual o salário médio dos empregados

```
SELECT AVG (Salario) FROM Empregado;
```

- Quantos empregados ganham mais de R\$1.000,00?

```
SELECT COUNT (*) FROM Empregado  
WHERE Salario > 1000;
```

39

## Cláusula DISTINCT

- Elimina tuplas duplicadas do resultado de uma consulta
- Exemplo: Quais os diferentes salários dos empregados?

```
SELECT DISTINCT Salario FROM Empregado;
```

40

## Cláusula GROUP BY

- Organiza a seleção de dados em grupos
- Exemplo: Listar os empregados agrupados por sexo, informando as quantidades

```
SELECT Sexo, Nome, Count(*) FROM Empregado GROUP  
BY Sexo, Nome;
```

- **Obs.** Todos os atributos do SELECT devem aparecer no GROUP BY

41

## Uso de Alias

- Alias são utilizados para substituir nomes de tabelas em comandos SQL
- São definidos na cláusula FROM
- Ex:

```
SELECT A.nome FROM Aluno A  
WHERE A.matricula = 15;
```

42

## Junção de Tabelas

- Junção de Tabelas
  - Citar as tabelas envolvidas na cláusula FROM
  - Alias para tabelas: utilizados para evitar ambiguidades
    - Referenciar os nomes de Empregado e de Departamento
    - ...e.nome...FROM Empregado e
    - ...d.nome...FROM Departamento d

43

## Junção de Tabelas

- Junção de Tabelas
    - Exemplo: Listar o nome do empregado e do departamento onde está alocado
- ```
SELECT E.Nome, D.Nome
FROM Empregado E, Departamento D
WHERE E.Num_Dep = D.Numero;
```

44

## Adicionando tuplas

- Comando INSERT  
INSERT INTO <tabela> (<lista de atributos>)  
VALUES (<valores>);
- Ex: Inserir dados de um professor  
INSERT INTO Professor(matricula, nome, telefone,  
salario, rua, bairro, cep) VALUES ('123', 'João', '9999-  
6969', 1500, 'rua A', 'bairro B', '6900000')

45

## Atualizando tuplas

- Comando UPDATE  
UPDATE <tabela>  
SET <nome atributo> = <valor>  
WHERE <condição>;
- Ex: Atualizar salario do professor de matricula  
123 para R\$ 4000  
UPDATE Professor SET salario = 4000 WHERE  
matricula = '123';

46

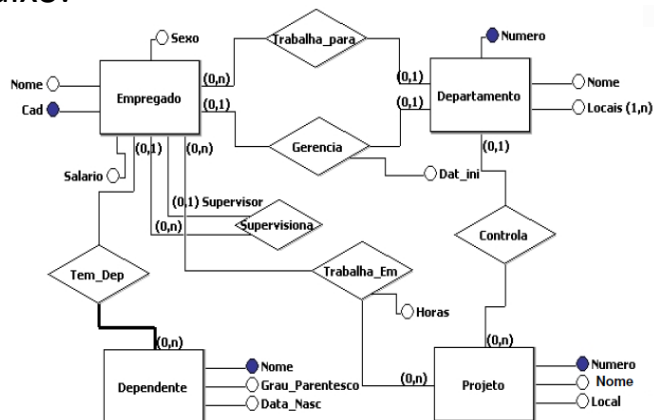
## Apagando tuplas

- Comando DELETE  
DELETE FROM <tabela> WHERE <condição>;
- Ex: Remover professor de matricula 123  
DELETE FROM Professor WHERE matricula = '123';

47

## Exercício

- Gerar o SQL DDL do esquema conceitual abaixo:



48



# Obrigado

E-mail: paulo.aguila@ifam.edu.br

Telefone: (92) 98189-8899