EXEMPLOS DE MAPEAMENTOS

ATENÇÃO - VERSÃO: *MYSQL 8.0.18*

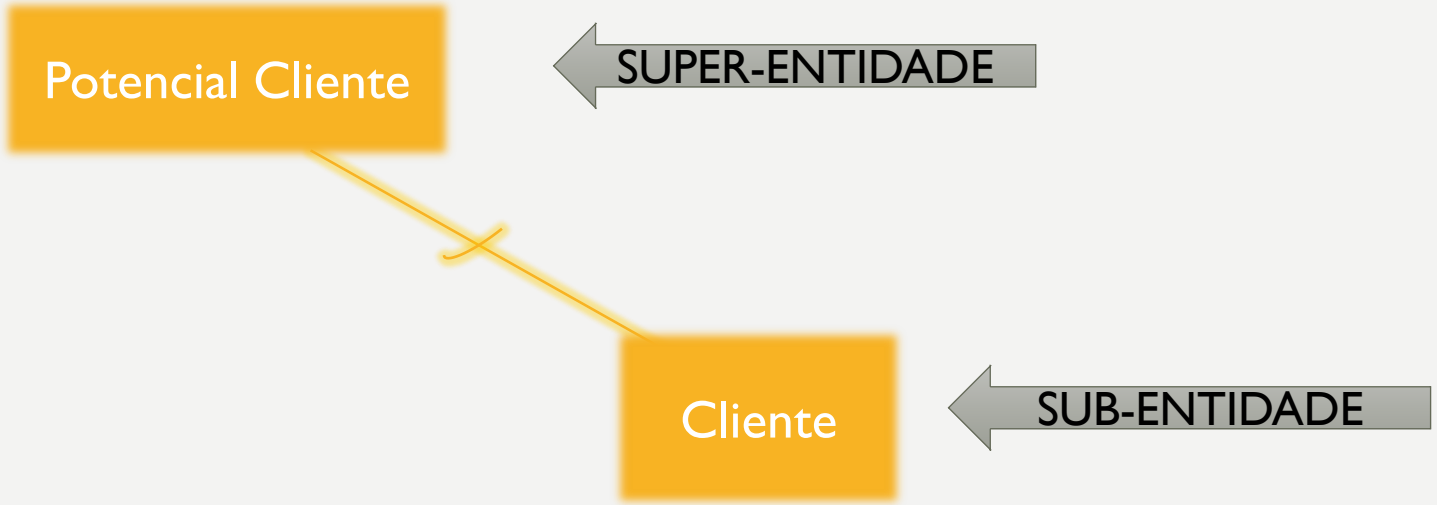
CDBD – TESP(PSI)

HIERARQUIA SIMPLES
DESCRIÇÃO

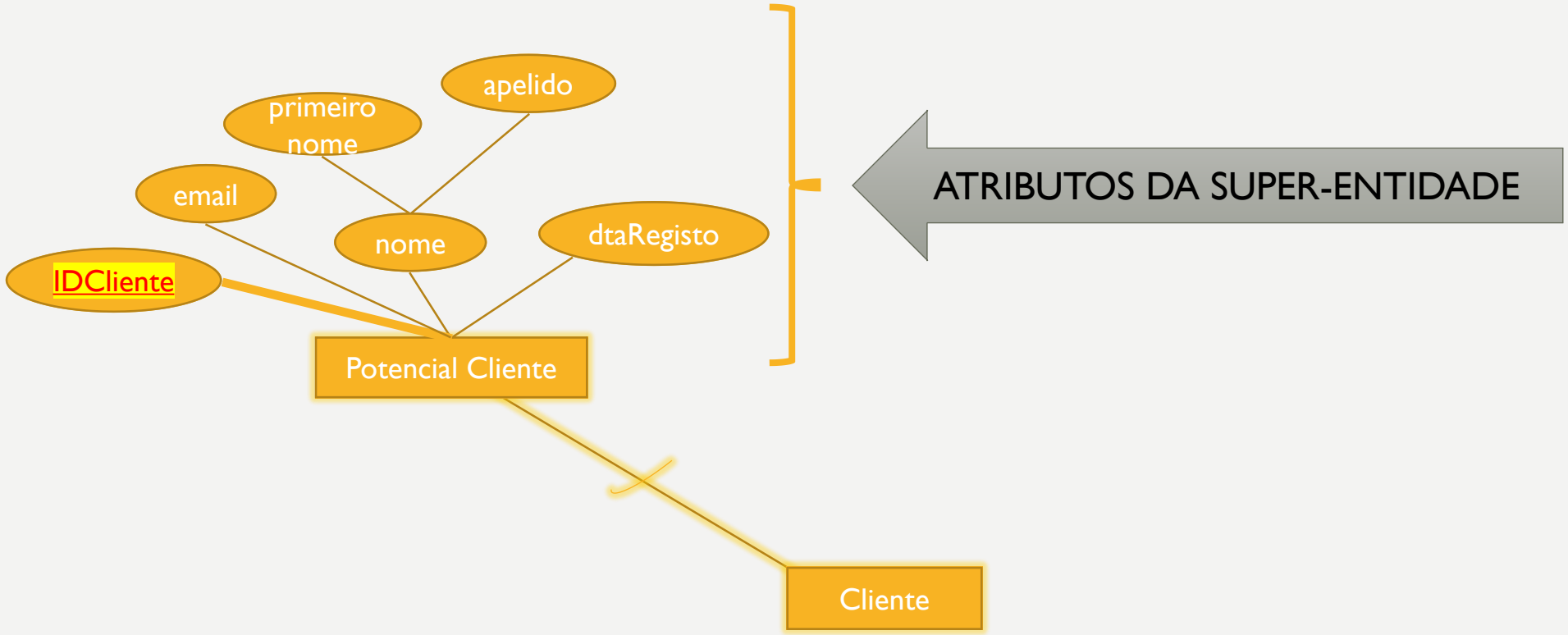
BREVE DESCRIÇÃO DO PROBLEMA

- Uma loja online recolhe dados sobre potenciais clientes permitindo o registo de dados através de um formulário. Por exemplo: o email, o primeiro nome e o apelido. No entanto os clientes que desejem realizar compras têm de preencher dados adicionais, tais como, *nickname*, password, telefone, nif e morada completa (rua, localidade e código postal).

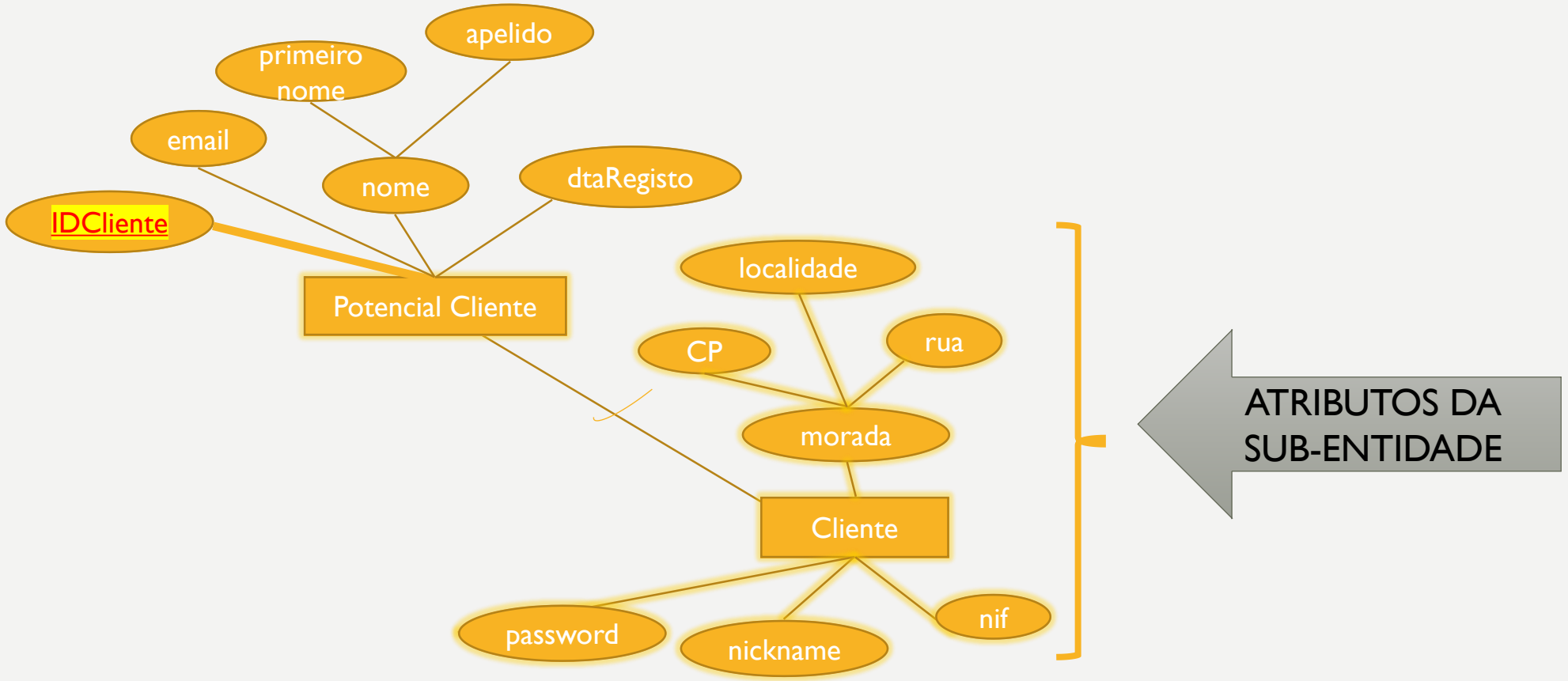
HIERARQUIA SIMPLES
DIAGRAMA ENTIDADE-RELACIONAMENTO



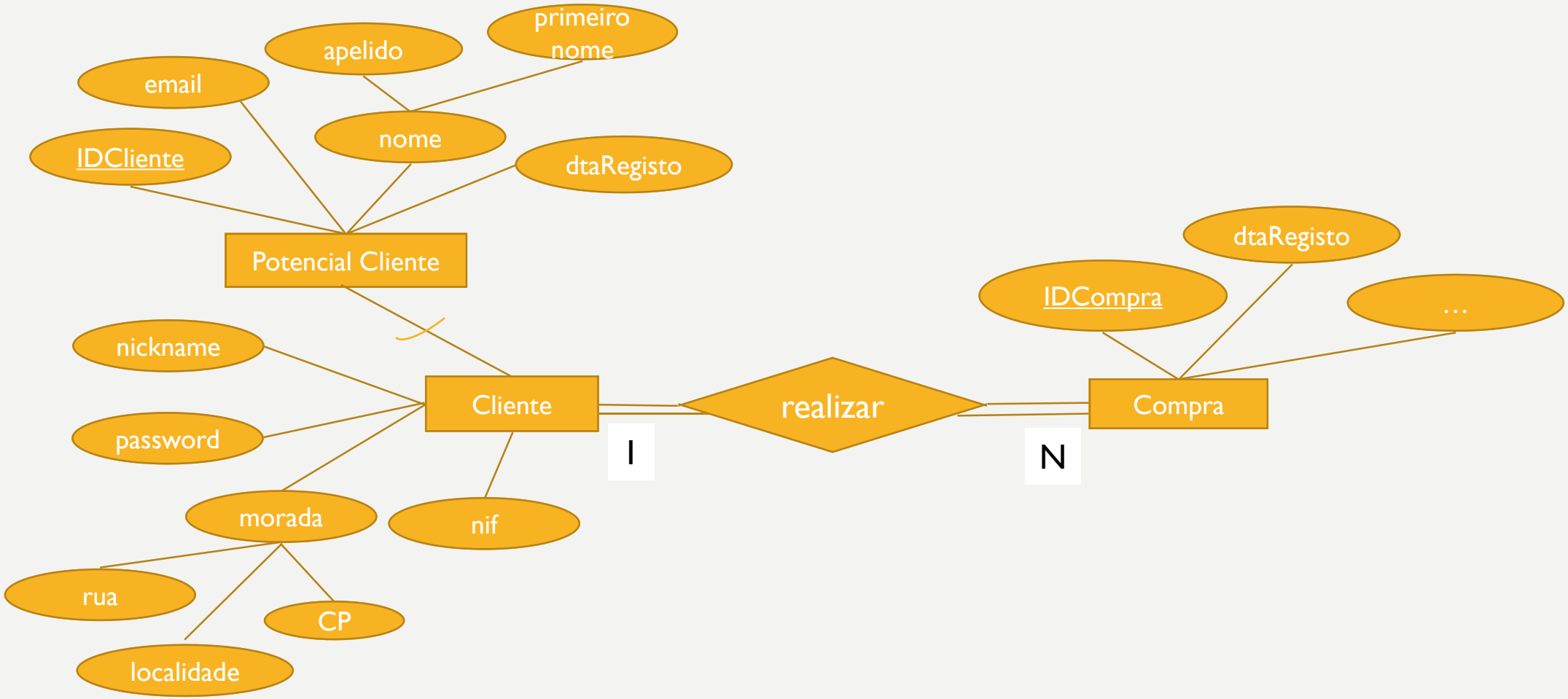
HIERARQUIA SIMPLES
DIAGRAMA ENTIDADE-RELACIONAMENTO



HIERARQUIA SIMPLES
DIAGRAMA ENTIDADE-RELACIONAMENTO



HIERARQUIA SIMPLES
DIAGRAMA ENTIDADE-RELACIONAMENTO



HIERARQUIA SIMPLES MODELO LÓGICO

PotencialCliente (**IDCliente**, email, primeiroNome, apelido, dtaRegisto)

Cliente(***IDCliente***, nif, localidade, rua, CP, nickname, password)FK (IDCliente) REFERENCES PotencialCliente(IDCliente)

Chaves Primárias a **BOLD** e SUPLINHADO

Chaves Estrangeira a *ITÁLICO*

HIERARQUIA SIMPLES MODELO LÓGICO

PotenciaisCliente (**IDCliente**, email, primeiroNome, apelido, dtaRegisto)

Clientes(***IDCliente***, localidade, rua, CP, nickname, password)FK (IDCliente) REFERENCES PotenciaisCliente(IDCliente)

Compras(**IDCompra**, …, *IDCliente*) FK (IDCliente) REFERENCES Cliente(IDCliente)

Chaves Primárias a **BOLD** e SUPLINHADO

Chaves Estrangeira a *ITÁLICO*

```
CREATE TABLE potenciaisClientes(
IDCliente       INT UNSIGNED    AUTO_INCREMENT,
email           VARCHAR(128)    NOT NULL,
primeiroNome    VARCHAR(64)     NOT NULL,
apelido         VARCHAR(64)     NOT NULL,
dtaRegisto      DATETIME        DEFAULT current_timestamp NOT NULL,
CONSTRAINT pk_pc_IDCliente      PRIMARY KEY(IDCliente),
CONSTRAINT uk_pc_email          UNIQUE(email))engine=innoDB;
```

HIERARQUIA SIMPLES
IMPLEMENTAÇÃO

```
CREATE TABLE Clientes(
IDCliente   INT UNSIGNED,
localidade  VARCHAR(128)    NOT NULL,
nif         VARCHAR(12)     NOT NULL,
rua         VARCHAR(128)    NOT NULL,
CP          VARCHAR(8)      NOT NULL,
nickname    VARCHAR(15)     NOT NULL,
password    VARCHAR(254)    NOT NULL,
CONSTRAINT  pk_clientes_IDCliente   PRIMARY KEY(IDCliente),
CONSTRAINT  uk_clientes_nif         UNIQUE(nif),
CONSTRAINT  uk_clientes_nickname    UNIQUE(nickname),
CONSTRAINT  fk_clientes_pc_idcliente FOREIGN KEY(IDCliente) REFERENCES potenciaisClientes(IDCliente)
)engine=innoDB;
```

HIERARQUIA SIMPLES
IMPLEMENTAÇÃO

```
CREATE TABLE Compras(
IDCompra     INT UNSIGNED                          NOT NULL AUTO_INCREMENT,
dtaCompra    DATETIME DEFAULT current_timestamp    NOT NULL,
IDCliente    INT UNSIGNED                          NOT NULL,
CONSTRAINT   pk_compras_compras PRIMARY KEY(IDCompra),
CONSTRAINT   fk_compras_clientes_idcliente FOREIGN KEY(IDCliente) REFERENCES Clientes(IDCliente))engine=InnoDB;
```

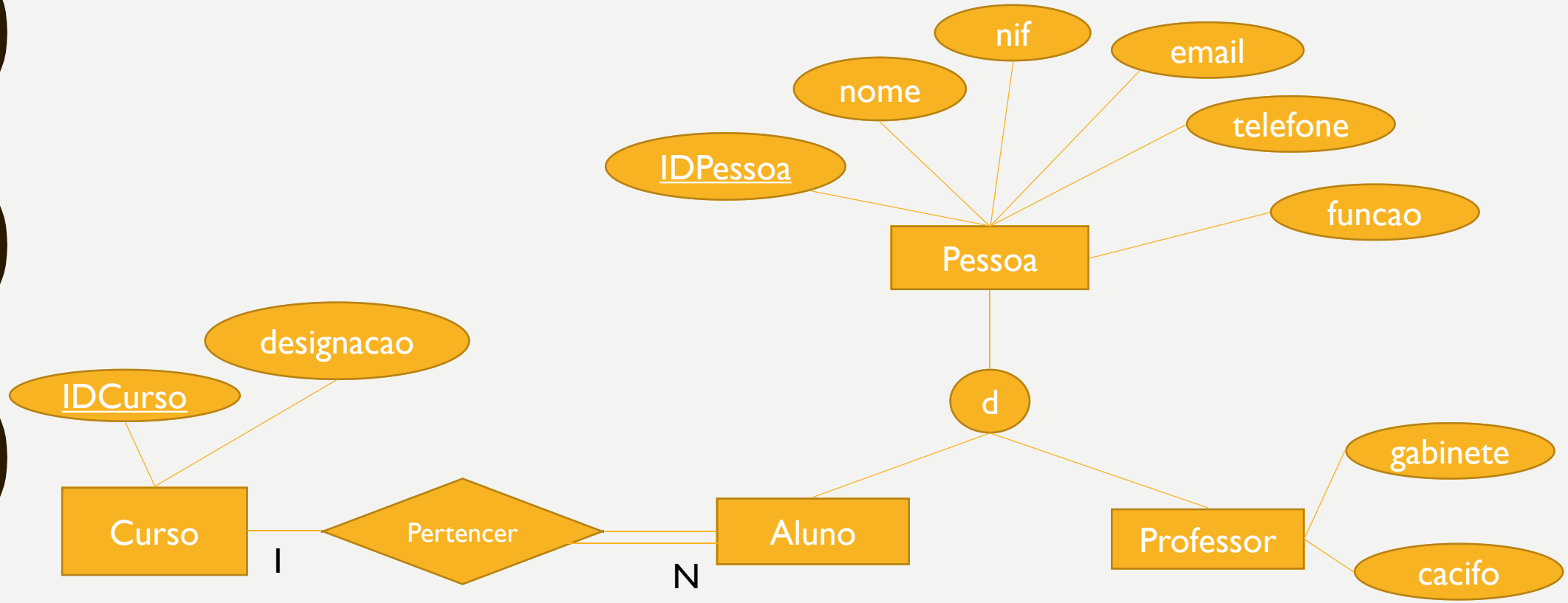
HIERARQUIA SIMPLES
IMPLEMENTAÇÃO

HIERARQUIA DISJUNÇÃO
DESCRIÇÃO DO PROBLEMA

DESCRIÇÃO DO PROBLEMA

- Uma escola pretende um sistema de informação para guardar dados de pessoas em geral. Sobre pessoas em geral é necessário guardar o nome, a data de nascimento, o número de identificação fiscal, o email e o número de telefone. As pessoas podem ser alunos, professores, assistentes operacionais, seguranças, entre outros. Sobre alunos é necessário saber o seu curso. Sobre professores é preciso conhecer o seu gabinete e o número do cacifo. **Os cursos têm uma designação e o número total de ECTS.**

HIERARQUIA DISJUNÇÃO
DIAGRAMA ENTIDADE RELACIONAMENTO



HIERARQUIA DISJUNÇÃO
MODELO LÓGICO

- Cursos(**IDCurso**, designacao)
- Pessoas(**IDPessoa**, nome, email, telefone, funcao)
- Professores(***IDProfessor***, cacifo, gabinete) FK (IDProfessor) REFERENCES Pessoas(IDPessoa)
- Alunos(***IDAluno***, *IDCurso*) FK (IDAluno) REFERENCES Pessoas(IDPessoa)
  FK (IDCurso) REFERENCES Cursos(IDCurso)

Chaves Primárias a **BOLD** e SUPLINHADO

Chaves Estrangeira a *ITÁLICO*

```
CREATE TABLE cursos(
IDCurso      INT UNSIGNED AUTO_INCREMENT,
designacao   VARCHAR(128) NOT NULL,
CONSTRAINT pk_cursos_idcurso PRIMARY KEY(IDCurso))engine=InnoDB;
```

HIERARQUIA DISJUNÇÃO
IMPLEMENTAÇÃO

```sql
CREATE TABLE pessoas(
IDPessoa INT UNSIGNED AUTO_INCREMENT,
nome     VARCHAR(128) NOT NULL,
email    VARCHAR(128) NOT NULL,
telefone VARCHAR(15)  NOT NULL,
funcao   ENUM('aluno','professor','assistente','seguranca', 'outra') DEFAULT 'aluno' NOT NULL,
CONSTRAINT pk_pessoa_idPessoa PRIMARY KEY(IDPessoa),
CONSTRAINT uk_pessoa_email UNIQUE(email)) engine=innoDB
;
```

HIERARQUIA DISJUNÇÃO IMPLEMENTAÇÃO

```sql
CREATE TABLE professores(
IDProfessor INT UNSIGNED,
cacifo      VARCHAR(5) NOT NULL,
gabinete    VARCHAR(10) NOT NULL,
CONSTRAINT pk_professores_idProfessor PRIMARY KEY(IDProfessor),
CONSTRAINT fk_professores_idProfessor FOREIGN KEY(IDProfessor) REFERENCES Pessoas(IDPessoa))engine=InnoDB;
```

HIERARQUIA DISJUNÇÃO IMPLEMENTAÇÃO

```
CREATE TABLE alunos(
IDAluno INT UNSIGNED,
IDCurso INT UNSIGNED NOT NULL,
CONSTRAINT  pk_aluno_idAluno PRIMARY KEY(IDAluno),
CONSTRAINT  fk_aluno_idAluno FOREIGN KEY(IDAluno) REFERENCES pessoas(IDPessoa),
CONSTRAINT  fk_cursos_idCurso FOREIGN KEY(IDCurso) REFERENCES cursos(IDCurso))engine=innoDB;
```

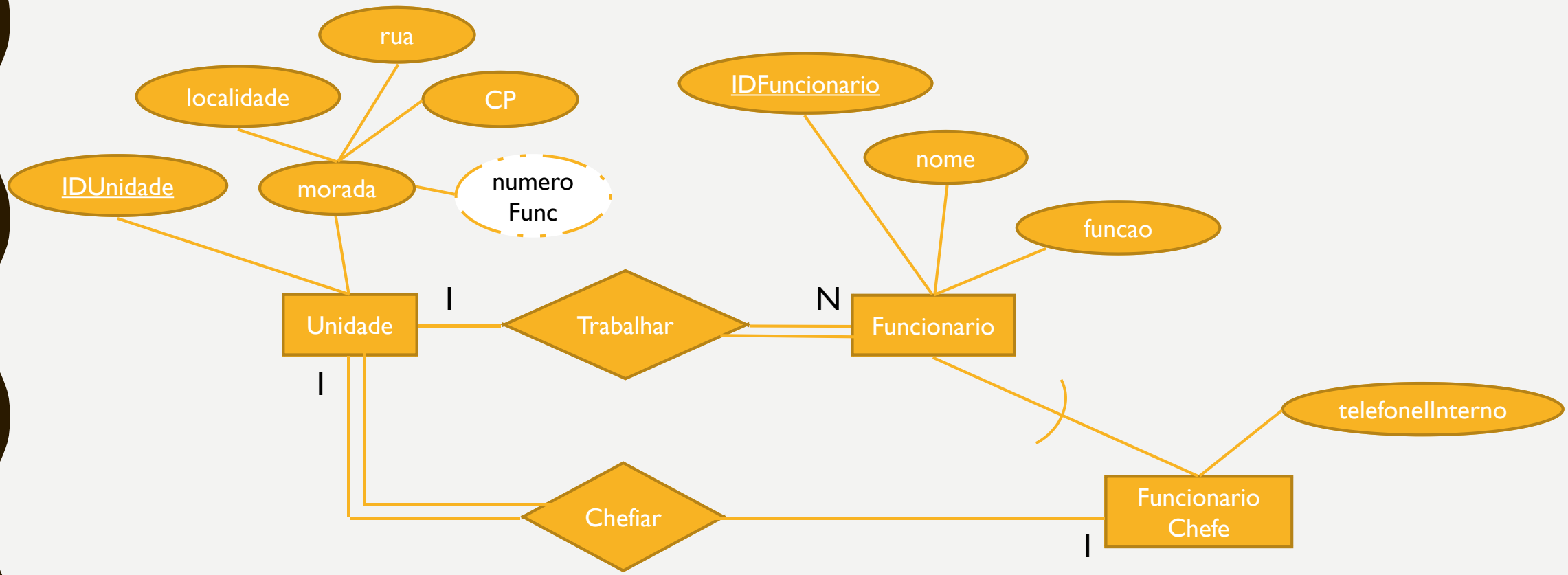
HIERARQUIA DISJUNÇÃO IMPLEMENTAÇÃO

RELACIONAMENTO DE 1:1 S/PO DE UM LADO

DESCRIÇÃO DO PROBLEMA

- Uma empresa de produção de moldes e plásticos, é composta por vários unidades fabris. Nas unidades fabris trabalham vários funcionários. Considere que um funcionário só trabalha numa unidade fabris. Há alguns empregados que são chefes. Desses alguns são responsáveis por unidades fabris. Considere que um empregado chefe não pode ser responsável por mais que uma unidade fabril e que todas têm um responsável.

RELACIONAMENTO 1:N S/ PO DE UM LADO



RELACIONAMENTO 1:1 S/ PO DE UM LADO MODELO LOGICO

- MODELO LÓGICO
- Funcionarios(**IDFuncionario**, nome, função, *IDUnidade*) FK (IDUnidade) REFERENCES Unidades(IDUnidade)
- Chefe(***IDChefe***, telefoneInterno) FK (IDChefe) REFERENCES Funcionarios(IDFuncionario)
- Unidades(**IDUnidade**, localidade, rua, CP, *IDChefe*) FK (IDChefe) REFERENCES Chefes (IDChefe)
- **Uma Vista para Calcular o número de Funcionários por Unidade Fabril**

```sql
CREATE TABLE funcionarios(
IDFuncionario    INT UNSIGNED AUTO_INCREMENT,
nome             VARCHAR(128) NOT NULL,
funcao           SET('Operario','Administrador','Secretario','Director','Outro'),
CONSTRAINT pk_funcionarios_idfuncionario PRIMARY KEY(IDFuncionario))ENGINE=INNODB;
```

RELACIONAMENTO 1:1 S/ PO DO LADO 1
IMPLEMENTAÇÃO

```sql
CREATE TABLE chefes(
IDChefe       INT UNSIGNED ,
telefoneInterno VARCHAR(10) NOT NULL,
CONSTRAINT pk_chefes_idchefe PRIMARY KEY(IDChefe),
CONSTRAINT fk_funcionarios_chefes_idchefe FOREIGN KEY(IDChefe) References funcionarios(IDFuncionario))ENGINE=INNODB;
```

RELACIONAMENTO 1:1 S/ PO DO LADO 1
IMPLEMENTAÇÃO

```sql
CREATE TABLE Unidades(
IDUnidade INT  UNSIGNED AUTO_INCREMENT,
localidade VARCHAR(64) NOT NULL,
rua        VARCHAR(128) NOT NULL,     .
CP          VARCHAR(8) NOT NULL,
IDChefe    INT UNSIGNED  NOT NULL,
CONSTRAINT PK_unidades_idunidade PRIMARY KEY(IDUnidade),
CONSTRAINT FK_unidades_idchefe FOREIGN KEY(IDChefe) REFERENCES chefes(IDChefe),
CONSTRAINT UK_unidades_idchefe UNIQUE(IDCHEFE))ENGINE=INNODB;
```

RELACIONAMENTO 1:1 S/ PO DO LADO 1
IMPLEMENTAÇÃO

```sql
ALTER TABLE funcionarios ADD IDUnidade INT UNSIGNED;
ALTER TABLE funcionarios ADD CONSTRAINT fk_funcionarios_unidades FOREIGN KEY(IDUnidade) REFERENCES Unidades(IDUnidade);
```

RELACIONAMENTO 1:1 S/ PO DO LADO 1
IMPLEMENTAÇÃO