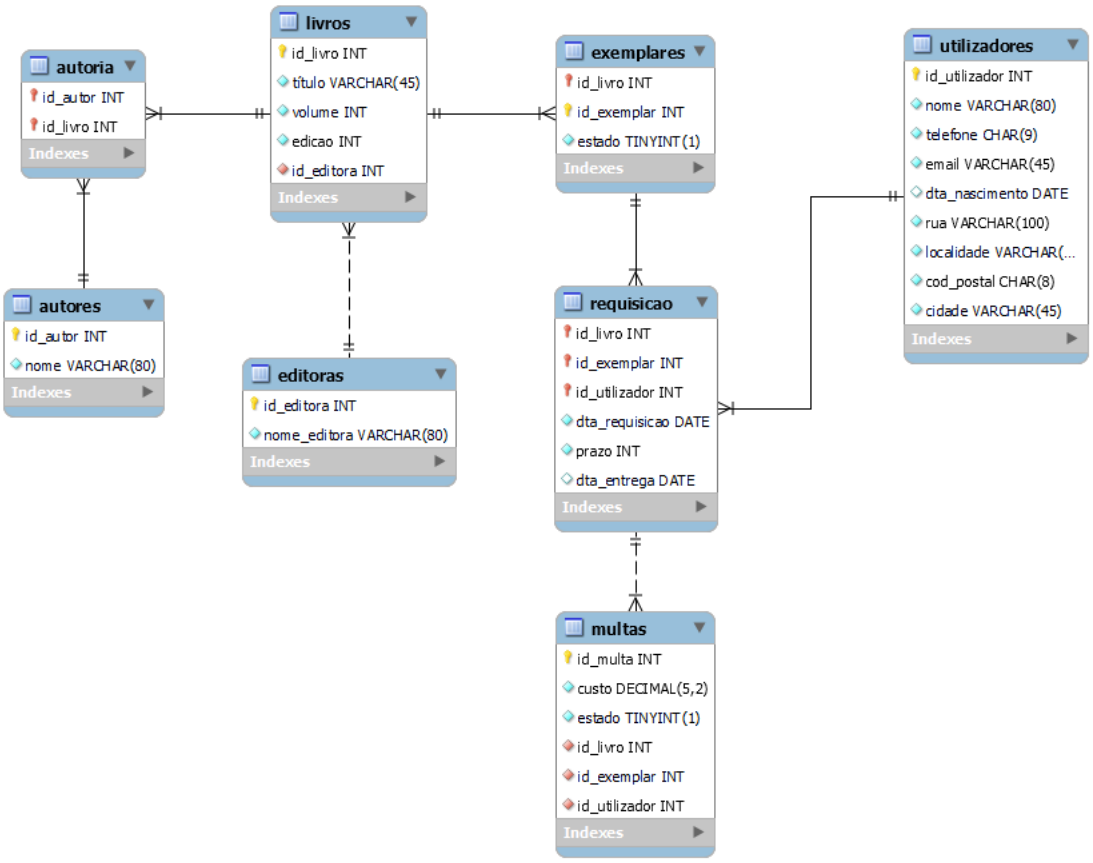
Universidade do Minho
Departamento de Informática

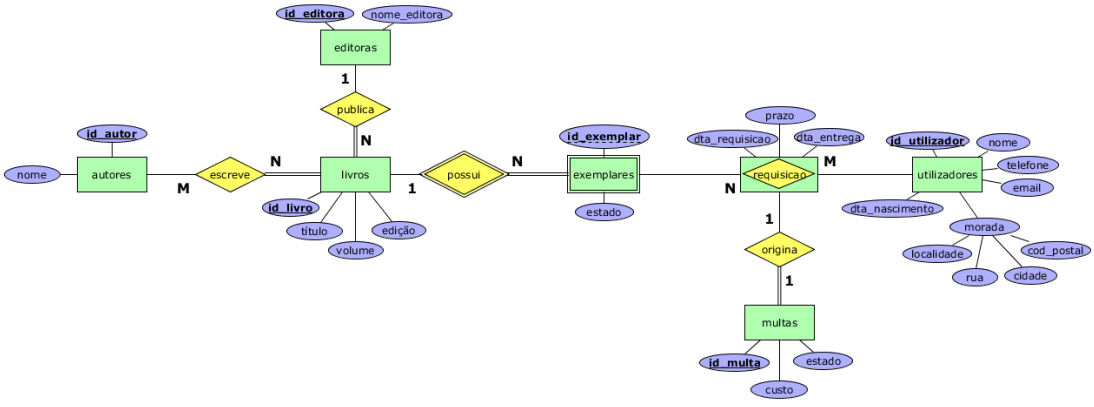
Curso: Licenciatura em Engenharia Informática (LEI);
Mestrado Integrado em Engenharia Informática (MIEI)
Unidade Curricular: Bases de Dados
Ano Letivo: 2021/2022 – 2º Semestre

FICHA DE EXERCÍCIOS – PL11

Considere o seguinte modelo de dados lógico:



Questão 1. Converta o modelo de dados lógico anterior no respetivo modelo E-R usando a Notação de Chen.



Questão 2. Identifique as dependências funcionais das relações do modelo de dados lógico acima descrito. Indique em que forma normal se encontram as relações (apenas até a 3FN), justificando, e, caso não estejam normalizadas, normalize-as.

-- tabela autores

id_autor → nome

-- tabela editoras

id_editora → nome_editora

-- tabela livros

id_livro → título, volume, edicao, id_editora

-- tabela exemplares

id_livro, id_exemplar → estado

-- tabela requisicao

id_livro, id_exemplar, id_utilizador → dta_requisicao, prazo, dta_entrega

-- tabela multas

id_multa → custo, estado, id_livro, id_exemplar, id_utilizador

IMPORTANTE: Esta foi a proposta de solução definida na aula com base na ideia de que um código postal completo (7 dígitos) corresponde apenas a uma rua, localidade e cidade. No entanto, isto não é sempre verdade, já quem existem zonas menos habitadas onde um código postal completo está associado a muitas ruas.

-- tabela utilizadores

id_utilizador → nome, telefone, email, dta_nascimento, cod_postal

cod_postal → rua, localidade, cidade

A tabela utilizadores é a única que não se encontra normalizada. Atualmente, a tabela encontra-se na 2FN uma vez que não existem dependências parciais, possui chave primária, todos os atributos são atómicos e não existem grupos de dados repetidos. No entanto, existem dependências transitivas, nomeadamente, os atributos rua, localidade, e cidade são determinados funcionalmente pelo código postal. Sendo assim, de forma a normalizar o esquema lógico apresentado para a 3FN, seria necessário criar duas tabelas:

utilizadores(id_utilizador, nome, telefone, email, dta_nascimento, cod_postal)

moradas(cod_postal, rua, localidade, cidade)

-- tabela utilizadores

id_utilizador → nome, telefone, email, dta_nascimento, cod_postal, rua, localidade, cidade

A base de dados encontra-se normalizada até à 3FN já que não existem dependências transitivas nem parciais, todas as tabelas possuem uma chave primária, todos os atributos são atómicos e não existem grupos de dados repetidos.

Questão 3. Desenvolva a instrução SQL DDL necessária para a criação das tabelas livros e exemplares. Tenha em consideração que o estado dos exemplares pode ser 0, 1 ou 2. No qual, 0 indica exemplares disponíveis, 1 indica exemplares indisponíveis e 2 indica exemplares descontinuados.

```
CREATE SCHEMA IF NOT EXISTS `ficha_11` DEFAULT CHARACTER SET utf8;


CREATE TABLE IF NOT EXISTS `ficha_11`.`livros` (
  `id_livro` INT NOT NULL AUTO_INCREMENT,
  `título` VARCHAR(45) NOT NULL,
  `volume` INT NOT NULL,
  `edicao` INT NOT NULL,
  `id_editora` INT NOT NULL,
  PRIMARY KEY (`id_livro`),
  CONSTRAINT `fk_livro_editora`
    FOREIGN KEY (`id_editora`)
    REFERENCES `ficha_11`.`editoras` (`id_editora`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


CREATE TABLE IF NOT EXISTS `ficha_11`.`exemplares` (
  `id_livro` INT NOT NULL,
  `id_exemplar` INT NOT NULL AUTO_INCREMENT,
  `estado` TINYINT(1) NOT NULL,
  PRIMARY KEY (`id_livro`, `id_exemplar`),
  CONSTRAINT `chk_estado`
       CHECK(`estado` IN (0,1,2)),
  CONSTRAINT `fk_exemplar_livro`
    FOREIGN KEY (`id_livro`)
    REFERENCES `ficha_11`.`livros` (`id_livro`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Questão 4. Desenvolva a instruções SQL que permita alterar a coluna *prazo* da tabela *requisicao* para definir 30 como valor padrão**.**

ALTER TABLE requisicao ALTER prazo SET DEFAULT 30;

Questão 5. Desenvolva a instrução SQL necessária para efetuar o povoamento da tabela livros. Considere pelo menos três instâncias.

-- como tem uma foreign key - id_editora – primeiro seria necessário povoar a tabela editoras

INSERT INTO editoras(id_editora, nome_editora) VALUES(1, 'Porto Editora'),(2,'Editorial Presença');

-- como definimos o id_livro com auto_increment podemos ocultar o id_livro da instrução

INSERT INTO livros (título, volume, edicao, id_editora) VALUES ('A Paciente Silênciosa',2, 1, 1), ('O Alquimista', 1, 1, 2), ('O Monte dos Vendavais',4,2,1);

-- caso contrário, seria necessário introduzir o id_livro na instrução

INSERT INTO livros (id_livro, título, volume, edicao, id_editora) VALUES (123, 'A Paciente Silênciosa',2, 1, 1), (122, 'O Alquimista', 1, 1, 2), (126, 'O Monte dos Vendavais',4,2,1);

Questão 6. Desenvolva as instruções SQL DML capazes de responder às seguintes questões, acompanhadas pela respetiva expressão em Álgebra Relacional:

Questão 6.1. Liste o estado das multas com um custo superior a 5€.

SELECT estado FROM multas WHERE custo>5;

$$\pi_{estado}(\sigma_{custo>5}(multas))$$

Questão 6.2. Liste os livros cujo título contém "al".

SELECT * FROM livros WHERE título LIKE'%al%';

$$\sigma_{título\ LIKE\ "\%al\%"}(livros)$$

Questão 6.3. Liste o id_exemplar, o título, o volume e a edição dos livros cujos exemplares ainda não foram entregues.

SELECT e.id_exemplar, l.titulo, l.volume, l.edicao FROM livros l NATURAL JOIN exemplares e WHERE estado IN (2,3);

$$\pi_{id\_exemplar,título,volume,edicao}(livros \bowtie \sigma_{estado=2\ \vee\ estado=3}(exemplares))$$

SELECT e.id_exemplar, l.titulo, l.volume, l.edicao FROM livros l NATURAL JOIN exemplares e NATURAL JOIN requisicao r WHERE r.dta_entrega IS NULL;

$$\pi_{id\_exemplar,título,volume,edicao}(\sigma_{dta\_entrega\ =\ NULL}(livros \bowtie exemplares \bowtie requisicao))$$

Questão 6.4. Liste os utilizadores de Braga para os quais nunca foi emitida uma multa.

SELECT * FROM utilizadores u NATURAL JOIN moradas m WHERE m.cidade='Braga' AND u.id_utilizador NOT IN(SELECT id_utilizador FROM multas);

$$R_{utilizadores\_braga} \leftarrow utilizadores \ltimes (\sigma_{cidade="Braga"}(moradas))$$

$$R_{utilizadores\_multas} \leftarrow utilizadores \ltimes multas$$

$$R_{utilizadores\_braga} - R_{utilizadores\_multas}$$

Questão 6.5. Indique os títulos dos livros que foram requisitados por todos os utilizadores.

A **SELECT título FROM livros l WHERE NOT EXISTS (SELECT * FROM utilizadores u WHERE id_utilizador NOT IN (SELECT id_utilizador FROM requisicao r WHERE l.id_livro=r.id_livro));** B

$$A \leftarrow livros \bowtie requisicoes$$

-- para efetuarmos a operação de divisão A/B, os atributos da tabela B têm que ser um *subset* dos atributos da tabela A por isso temos que realizar uma operação de projeção na tabela utilizadores

$$B \leftarrow \pi_{id\_utilizador}(utilizadores)$$

-- com a divisão A:B obtemos os livros requisitados por todos os utilizadores c/ as colunas id_livro, titulo, volume, edição, id_editora, id_exemplar, dta_requisicao, prazo, dta_entrega (ou seja as colunas resultantes do natural join com a exceção do atributo que liga as tabelas A e B – id_utilizador – que é removido após a operação de divisão

$$R \leftarrow A \div B$$

-- como só queremos o título do livro, fazemos uma operação de projeção sobre a divisão

$$\pi_{titulo}(R_2)$$