Bases de Dados, 2018/19
Departamento de Ciência de Computadores, Faculdade de Ciências da Universidade do Porto
Exame de época de recurso — 04/07/2019 — Duração: 2:30 ★
Número: _____
Nome: _____

GRUPO A (25 %) — Responda na folha do enunciado.

1. Complete a criação de uma tabela `MOVIE_RATING`, em associação ao esquema da BD MovieLens, representando uma valorização ("rating") de um filme por parte de um cliente com os seguintes atributos: um identificador inteiro `MovieId` para o filme, chave externa para `MOVIE.MovieId`; um identificador inteiro `CustomerId` para o cliente, chave externa para `CUSTOMER.CustomerId`; um valor inteiro `RValue` para a valorização pelo cliente para o filme; `RDate`, a data da valorização; e uma string de tamanho 32 `Comment`, um comentário opcional ao filme por parte do utilizador. Considere que a chave primária da tabela é formada pelo par de atributos `MovieId` e `CustomerId`, e que o atributo `Comment` é o único que poderá não estar definido (i.e., pode tomar valor `NULL`),

```
CREATE TABLE MOVIE_RATING
(
  MovieId    INT NOT NULL,
  CustomerId INT NOT NULL,
  RValue     INT NOT NULL,
  RDate      DATE NOT NULL,
  Comment    VARCHAR(32),
  PRIMARY KEY(MovieId, CustomerId),
  FOREIGN KEY(MovieId) REFERENCES MOVIE(MovieId),
  FOREIGN KEY(CustomerId) REFERENCES CUSTOMER(CustomerId)
);
```

2. (Escolha múltipla) Tenha em conta os requisitos para a tabela `MOVIE_RATING`, em particular as definições de chave primária e chaves externas. Assuma que não existem por sua vez referências via chaves externas a `MOVIE_RATING`. Seja `r` um registo em `MOVIE_RATING`, `m = r[MovieId]` e `c = r[CustomerId]` (i.e., `m` e `c` são resp. os identificadores de filme em `MOVIE` e cliente em `CUSTOMER` definidos para `r`).

☐ O registo r só pode ser removido se os registos em `MOVIE` e `CUSTOMER` correspondentes a `m` e `c` forem previamente removidos.
☐ Em uma instrução `UPDATE` sobre `r`, os valores de `m` e `c` podem ser modificados arbitrariamente.
■ O registo `r` pode ser removido sem restrições usando uma instrução `DELETE`.
■ Pode existir um registo `r'` na tabela tal que `r'[MovieId] = m` e `r'[CustomerId] = c + 1`, desde que `c + 1` corresponda a um valor de `CustomerId` definido para um registo em `CUSTOMER`.
☐ Usando uma instrução `INSERT`, podemos definir um novo registo `r'` na tabela tal que `r'[MovieId] = m` e `r'[CustomerId] = c`.

3. Assinale com **F**, **P**, **T**, cada uma das características consoante estas se relacionem respectivamente com **(F)** "stored functions", **(P)** "stored procedures" ou **(T)** "triggers". Indique apenas **uma** das opções para cada uma das características abaixo.

P Requer o uso de `CALL` para ser invocado.
F Tem um tipo de retorno definido.
T Pode ter as variáveis `OLD` e/ou `NEW` implicitamente definidas.
P Os parâmetros podem ser de entrada (`IN`), saída (`OUT`), ou entrada-saída (`INOUT`).
T Executa automaticamente em reação a uma operação de manipulação de dados.

4. (Escolha múltipla) Considere a seguinte consulta SQL, onde, além da caracterização da tabela `MOVIE_RATING` na questão 1, deve assumir que `Title` é um atributo de `MOVIE` para o título de um filme e que `Name` é um atributo de `CUSTOMER` para o nome de um cliente.

```
SELECT Title, Name, RValue
FROM MOVIE JOIN MOVIE_RATING USING(MovieId)
           JOIN CUSTOMER USING(CustomerId)
WHERE CustomerId <= 100 AND RValue >= 4;
```

Assinale as consultas expressas em álgebra relacional abaixo que são equivalentes à consulta SQL.

■ $R_1 \longleftarrow \sigma_{\text{RValue} \geq 4}(\texttt{MOVIE\_RATING})$
$R_2 \longleftarrow \sigma_{\text{CustomerId} \leq 100}(\texttt{CUSTOMER})$
$R_3 \longleftarrow R_2 \bowtie_{\text{CustomerId}} R_1$
$R_4 \longleftarrow R_3 \bowtie_{\text{MovieId}} \texttt{MOVIE}$
$R_5 \longleftarrow \pi_{\text{Title,Name,RValue}}(R_4)$

□ $R_1 \longleftarrow \texttt{CUSTOMER} \bowtie_{\text{MovieId}} \texttt{MOVIE}$
$R_2 \longleftarrow R_1 \bowtie_{\text{CustomerId}} \texttt{MOVIE\_RATING}$
$R_3 \longleftarrow \sigma_{\text{CustomerId} \leq 100}(R_2)$
$R_4 \longleftarrow \sigma_{\text{RValue} \geq 4}(R_3)$
$R_5 \longleftarrow \pi_{\text{Title,Name,RValue}}(R_4)$

□ $R_1 \longleftarrow \texttt{MOVIE\_RATING} \bowtie_{\text{MovieId}} \texttt{MOVIE}$
$R_2 \longleftarrow R_1 \bowtie_{\text{CustomerId}} \texttt{CUSTOMER}$
$R_3 \longleftarrow \pi_{\text{Title,Name,RValue}}(R_2)$
$R_4 \longleftarrow \sigma_{\text{CustomerId} \leq 100}(R_3)$
$R_5 \longleftarrow \sigma_{\text{RValue} \geq 4}(R_4)$

■ $R_1 \longleftarrow \texttt{MOVIE\_RATING} \bowtie_{\text{MovieId}} \texttt{MOVIE}$
$R_2 \longleftarrow R_1 \bowtie_{\text{CustomerId}} \texttt{CUSTOMER}$
$R_3 \longleftarrow \sigma_{\text{CustomerId} \leq 100}(R_2)$
$R_4 \longleftarrow \pi_{\text{Title,Name,RValue}}(R_3)$
$R_5 \longleftarrow \sigma_{\text{RValue} \geq 4}(R_4)$

5. (Escolha múltipla) Considere o seguinte código de uma aplicação Python que acede a uma base de dados com a tabela `MOVIE_RATING`:

```
data = db.execute(
 'SELECT * FROM MOVIE_RATING WHERE Comment LIKE \'%' + expr + '%\''
 ).fetchall()
```

Suponha que `expr` é um valor obtido de uma fonte externa à aplicação (por ex. através da rede), e que esse valor não é alvo de qualquer validação.

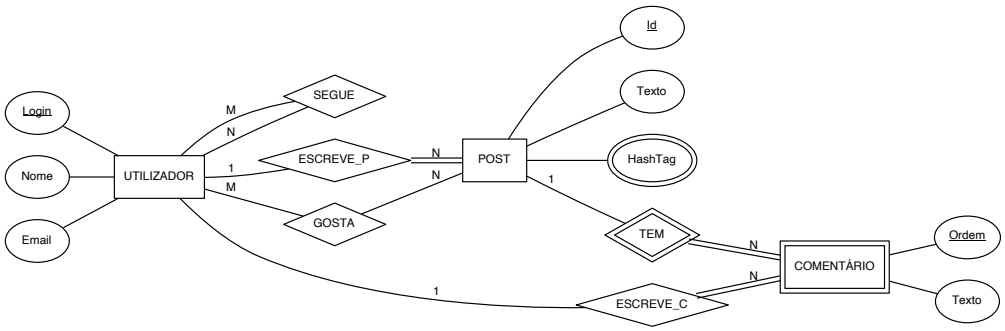
■ O código é passível de injeção SQL, podendo portanto levar à execução de código SQL indevido e potencialmente malicioso.
■ Para evitar injeção SQL, o código acima deveria ser corrigido por forma a termos uma instrução SQL parameterizada.
■ Um SGBD não consegue no caso geral detectar situações de injeção no código SQL gerado por uma aplicação.
□ Não há possibilidade de injeção SQL neste caso.
□ A possibilidade de injeção SQL não é neste caso problemática, já que está em causa apenas a execução de uma instrução `SELECT` para leitura de dados.

GRUPO B (25 %) — Use folhas de exame.

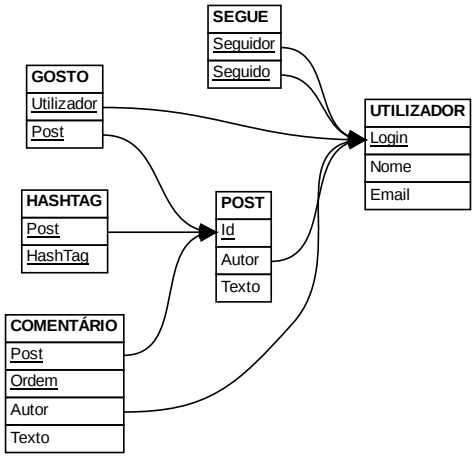
Considere os seguintes requisitos para uma base de dados de uma rede social simples em que:

— Um utilizador é caracterizado por um identificador de "login" único, nome, e email.
— Cada utilizador pode ser seguidor de outros em que tenha interesse de acompanhar actividades, tais como "posts" (descritos a seguir).
— Um utilizador escreve "posts", em que cada "post" é caracterizado por um identificador único, um texto, e um conjunto de "hashtags" (etiquetas).
— Os utilizadores pode reagir a "posts" através da escrita de comentários. Um comentário tem um número de ordem e um texto associado, sendo o número de ordem de um comentário (apenas) único por "post" (i.e., pode-se repetir para "posts" diferentes).
— Além de comentários, utilizadores podem assinalar que gostaram de "posts" com "likes". Comentários e "likes" são feitos de forma independente.
— Podem haver utilizadores sem qualquer participação em termos de seguimento de outros / seguidores / "posts" / comentários / "likes". Um "post" / comentário / "like" tem sempre um utilizador associado.

1. Apresente um modelo ER na forma de um diagrama para a base de dados.



2. Apresente uma tradução do modelo ER para um esquema relacional.



GRUPO C (15 %) — Use folhas de exame.

Considere uma BD para uma "play list" de canções em que:

— um artista é caracterizado por um identificador único (`ArtId`) e um nome (`ArtNome`);
— um álbum de um artista tem um identificador único (`AlbId`), um título (`AlbTítulo`), e ano de edição (`AlbAno`);
— uma canção num álbum tem associado o número de ordem no álbum (`CNum`), um título (`CTítulo`), e um tempo de duração (`CTempo`).

Assuma o seguinte esquema 1NF para a BD e exemplos de entradas correspondentes na BD:

`PLAYLIST(ArtId,ArtNome,AlbId,AlbTítulo,AlbAno,CNum,CTítulo,CTempo)`

ArtId	ArtNome	AlbId	AlbTítulo	AlbAno	CNum	CTítulo	CTempo
1	Caetano Veloso	1	Abraçaco	2012	1	Um Abraçaco	3:50
1	Caetano Veloso	1	Abraçaco	2012	2	Estou Triste	5:13
1	Caetano Veloso	1	Abraçaco	2012	8	Vinco	4:38
1	Caetano Veloso	2	Caetanear	1984	8	Sampa	3:17
2	Sonic Youth	3	Goo	1990	8	Mildred Pierce	2:12
2	Sonic Youth	4	Sister	1987	8	Hot Wire My Heart	3:47
3	Lou Reed	5	Best of	1984	1	Perfect Day	3:47
3	Lou Reed	5	Best of	1984	2	Walk On The Wild Side	4:15
4	Leonard Cohen	6	Best of	1975	4	Bird On The Wire	3:27

NOTA: Um exemplo bastante similar é discutido nos slides das aulas teóricas (slides 31-34).

1. Identifique as dependências funcionais entre atributos em `PLAYLIST` e o conjunto de atributos que formam a chave primária.

`ArtId` $\longrightarrow$ `ArtNome`

`AlbId` $\longrightarrow$ { `AlbTítulo, AlbAno, ArtId` }

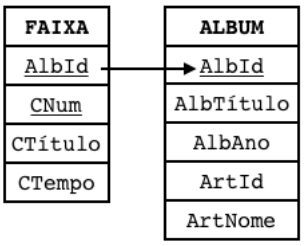
{ `AlbId, CNum` } $\longrightarrow$ { `CTítulo, CTempo` }

Chave primária: { `AlbId, CNum` }

2. Explique porque é que o esquema não é 2NF, e normalize-o para 2NF.

Não está na segunda forma normal porque temos uma dependência acima que envolve uma chave parcial: `AlbId` $\longrightarrow$ { `AlbTítulo, AlbAno, ArtId` } .

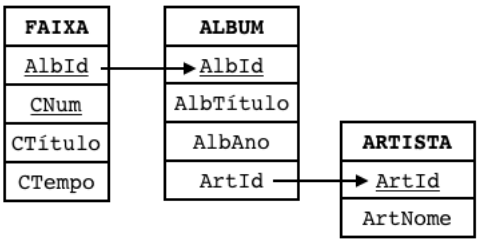
Normalização para 2NF:



3. Explique se o esquema que obteve na questão anterior é ou não também 3NF. Se não for o caso, normalize-o para 3NF.

O esquema não está na 3ª forma normal, já que temos a dependência transitiva (envolvendo atributo não-chave `ArtId` em `ALBUM`) `AlbId` $\longrightarrow$ `ArtId` $\longrightarrow$ `ArtNome`.

Normalização para 3NF:



GRUPO D (15 %) — Use folhas de exame.

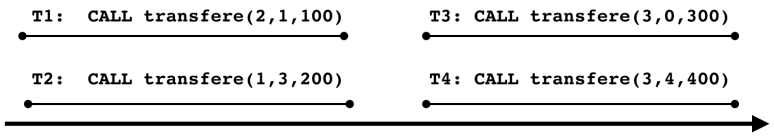
Considere uma tabela `CONTA` para uma conta bancária com atributos `Num` para o número da conta e `Saldo` para o saldo da conta, ambos inteiros. Suponha ainda que está definido um "stored procedure" `transfere(IN c1 INT, IN c2 INT, IN v INT)` de tal forma que `transfere(c1,c2,v)` executa a seguinte sequência de passos:

— Inicia uma transação com `START TRANSACTION`.
— Se `c1` e `c2` forem números de contas válidas e o saldo da conta `c1` for igual ou superior a `v` (código omitido) então é executado:

```
UPDATE CONTA SET Saldo = Saldo - v WHERE Num = c1;
UPDATE CONTA SET Saldo = Saldo + v WHERE Num = c2;
COMMIT;
```

isto é, o saldo de `c1` é decrementado (debitado) no montante de `v`, o de `c2` é incrementado (creditado) no montante de `v`, e a transação toma efeito permanente com `COMMIT`.
— Caso contrário, a transação iniciada é abortada com `ROLLBACK`.

1. Considere que as contas 1, 2, 3, 4 estão definidas na BD com saldos de respectivamente, 100, 200, 300, e 400 e que 0 **não** é um número válido de conta. Na execução de 4 transações da forma ilustrada abaixo, temos que `T1` e `T2` executam em simultâneo, seguidas de `T3` em simultâneo com `T4`. Assumindo que o SGDB executa as transações de forma serializável, justifique quais são os estados possíveis da BD no final para a tabela `CONTA`, e que transações completam com sucesso ou são abortadas para cada possibilidade.



A transação `T3` aborta sempre já que 0 não é uma conta válida, sendo assim irrelevante se `T4` tem efeito lógico antes de `T3` ou não. Só temos de considerar então os casos de serialização de transações obedecendo a uma das seguintes duas ordens:

- `T1` $\longrightarrow$ `T2` $\longrightarrow$ `T4`
  - `T1` sucede actualizando o saldo da conta 2 com o valor de $100 = 200 - 100$ e o saldo da conta 1 com o valor de $200 = 100 + 100$.
  - `T2` sucede actualizando o saldo da conta 1 com o valor de $0 = 200 - 200$ e o saldo da conta 3 com o valor $500 = 300 + 200$.
  - `T4` sucede actualizando o saldo da conta 3 com o valor de $100 = 500 - 400$ e o saldo da conta 4 com o valor de $800 = 400 + 400$.
- `T2` $\longrightarrow$ `T1` $\longrightarrow$ `T4`
  - `T2` é abortada porque o valor do saldo da conta 1 será 100, menor que o valor de 200 necessário para a transferência.
  - `T1` sucede actualizando o saldo da conta 2 com o valor de $100 = 200 - 100$ e o saldo da conta 1 com o valor de $200 = 100 + 100$.
  - `T4` é abortada porque o valor do saldo da conta 3 é 300 (já que `T1` foi abortada), menor que o valor de 400 necessário para a transferência.

2. Ilustre um escalonamento possível para uma execução serializável de `T1` e `T2`. Explique o escalonamento passo-a-passo em termos de leituras e escritas de registos, uso de "locks" segundo uma estratégia de "two-phase locking", e de tal forma que uma das transações `T1` ou `T2` bloqueie momentaneamente.

Veja slides das aulas teóricas, em particular o slide 30 que esquematiza um escalonamento "two-phase locking" para um exemplo muito parecido com este.

GRUPO E (20 %) — Use uma folha de exame SEPARADA para este grupo.

x	2	$]2,4]$	$]4,8]$	$]8,16]$	$]16,32]$	$]32,64]$	$]64,128]$	$]128,256]$	$]256,512]$	$]512,1024]$	$]1024,2048]$	$]2048,4096]$
$\lceil \log_2(x) \rceil$	1	2	3	4	5	6	7	8	9	10	11	12

Neste grupo justifique as suas respostas apresentando todos os cálculos relevantes que efectuou. Assuma em todos os casos que um registo está inteiramente guardado num único bloco de disco (i.e., um registo não "atravessa" blocos), que cada bloco de disco tem um tamanho de 4096 (2^{12}) bytes, e que uma referência a um bloco de disco tem um tamanho de 8 bytes.

1. Considere uma tabela `T` com 160,000 registos de tamanho fixo igual a 100 bytes, armazenada num ficheiro ordenado pela chave primária `K` de `T`, onde o tamanho de `K` é de 8 bytes. Indique o número de blocos em disco necessários para o armazenamento de `T`, e o número máximo de acessos a blocos de disco para localizar um registo com base num valor para `K`.

Sabemos que: o tamanho de um bloco em disco é de $B = 4096$ bytes; o tamanho fixo de cada registo é de $R = 100$ bytes. e o número total de registos é de $N = 160,000$.

Então o "blocking factor" (nº de registos máximo por bloco) é dado por

$$bfr = \left\lfloor \frac{4096}{100} \right\rfloor = 40,$$

e o número de blocos necessários para armazenar `T` é de

$$n = \left\lceil \frac{160,000}{40} \right\rceil = 4,000.$$

O número máximo de acessos a blocos a disco é então de:

$$\lceil \log_2(4,000) \rceil = 12$$

2. Caracterize um índice multi-nível sobre o ficheiro da questão anterior tendo `K` como atributo de indexação, de tal forma que o índice de último nível ocupe apenas 1 bloco em disco. Para cada nível i, identifique o número de blocos em disco necessários, e o número máximo de acessos a disco para localizar um registo a partir do nível i.

$$\begin{aligned} \text{(todos os níveis)} \quad & R_I = 8 + 8 = 16 \quad B = 4096 \\ \text{blocking factor} \quad & bfr_I = \left\lfloor \frac{4096}{16} \right\rfloor = 256 \end{aligned}$$

$$\begin{aligned} \text{Nível 1} \quad & N_I^1 = n = 4,000 \\ \text{nº blocos} \quad & n_I^1 = \left\lceil \frac{4,000}{256} \right\rceil = 16 \\ \text{acessos} \quad & \lceil \log_2(16) \rceil + 1 = 5 \end{aligned}$$

$$\begin{aligned} \text{Nível 2} \quad & N_I^2 = n_I^1 = 16 \\ \text{nº blocos} \quad & n_I^2 = \left\lceil \frac{16}{256} \right\rceil = 1 \\ \text{acessos} \quad & 1 + 1 + 1 = 3 \end{aligned}$$

3. Considere agora que `T` tem também um atributo `U` que é uma chave secundária (i.e., `UNIQUE`) com um tamanho de 24 bytes. Indique quantos blocos em disco teremos de aceder sem recorrer a

qualquer indexação para localizar um registo com base num valor para `U`. Caracterize de seguida um índice simples (com um 1 só nível) para o ficheiro usando `U` como atributo de indexação, identificando o número de blocos em disco necessários, e o número máximo de acessos a disco para localizar um registo usando o índice.

$$N_I = N = 160,000 \quad R_I = 24 + 8 = 32 \quad B = 4096$$

$$\text{blocking factor} \quad bfr_I = \left\lfloor \frac{4096}{32} \right\rfloor = 128$$

$$\text{nº blocos} \quad n_I = \left\lceil \frac{160,000}{128} \right\rceil = 1250$$

$$\text{acessos} \quad \lceil \log_2(1,250) \rceil + 1 = 12$$