



Insper

Preparação ENADE Megadados

ENADE 2014

Engenharia da Computação

Questão 34

- SQL
- Modelo relacional

ENADE 2014 – Engenharia de Computação

QUESTÃO 34

Suponha que um banco de investimentos possua um sistema que controla, para cada cliente, os tipos de investimentos que eles mesmos realizam ao longo do tempo. Cada cliente pode ter apenas uma aplicação de cada tipo de investimento oferecido pelo banco.

Considere as tabelas *Cliente*, *TipoInvestimento* e *Investimento* pertencentes a um modelo relacional do sistema citado (as chaves primárias estão sublinhadas).

Cliente (codCliente, nomeCliente, enderCliente, cidadeCliente, anoIngressoCliente)

TipoInvestimento (codInvestimento, descricaoInvestimento, taxaRemuneracao)

Investimento (codCliente, codInvestimento, valor, dataDeposito)

A partir do modelo relacional apresentado, avalie as afirmações a seguir.

I. O comando SQL que lista todos os pares de clientes que residem na mesma cidade é:

```
select c1.NomeCliente, c2.NomeCliente
from   Cliente c1, Cliente c2
where  c1.CidadeCliente = c2.CidadeCliente
and    c1.CodCliente < c2.CodCliente;
```

II. O comando SQL que retorna, para cada cidade, os clientes mais antigos é:

```
select codCliente, nomeCliente, cidadeCliente, anoIngressoCliente
from   Cliente
where  anoIngressoCliente in (select min(anoIngressoCliente)
from   Cliente
group by cidadeCliente);
```

III. O comando SQL que retorna, para cada cidade (de um cliente), o ano de ingresso mais antigo, porém apenas para as cidades com mais de um cliente, é:

```
select cidadeCliente, min(AnoIngressoCliente)
from   Cliente
group by cidadeCliente
having count(*) > 1;
```

IV. O comando SQL que retorna o maior valor de cada investimento de cada cliente é:

```
select codCliente, max(valor)
from   Cliente c, Investimento i
where  c.codCliente = i.codCliente
group by codCliente, codInvestimento
```

É correto apenas o que se afirma em

- A** I.
- B** II.
- C** I e III.
- D** II e IV.
- E** III e IV.

Questão típica de ENADE: requer atenção às alternativas, e leitura cuidadosa.

Leitura cuidadosa que na verdade faltou aos organizadores do exame: esta questão foi anulada... esqueceram de sublinhar as chaves primárias!

Mas a questão é boa e resolvível se completarmos o que falta

ENADE 2014 – Engenharia de Computação

QUESTÃO 34

Suponha que um banco de investimentos possua um sistema que controla, para cada cliente, os tipos de investimentos que eles mesmos realizam ao longo do tempo. Cada cliente pode ter apenas uma aplicação de cada tipo de investimento oferecido pelo banco.

Considere as tabelas Cliente, TipoInvestimento e Investimento pertencentes a um modelo relacional do sistema citado (as chaves primárias estão sublinhadas).

Cliente (codCliente, nomeCliente, enderCliente, cidadeCliente, anoIngressoCliente)

TipoInvestimento (codInvestimento, descricaoInvestimento, taxaRemuneracao)

Investimento (codCliente, codInvestimento, valor, dataDeposito)

A partir do modelo relacional apresentado, avalie as afirmações a seguir.

Ainda faltam as restrições de chave estrangeira:

- Investimento (codCliente) referencia Cliente (codCliente)
- Investimento (codInvestimento) referencia TipoInvestimento (codInvestimento)

Diagrama do modelo relacional

Cliente	
PK	<u>codCliente</u>
	nomeCliente
	enderCliente
	cidadeCliente
	anoIngressoCliente



Chave primária
simples

Investimento	
PK,FK	<u>codCliente</u>
PK,FK	<u>codInvestimento</u>
	valor
	dataDeposito



Chave primária
composta

TipoInvestimento	
PK	<u>codInvestimento</u>
	descricaoInvestimento
	taxaRemuneracao



Chave primária
simples

Diagrama do modelo relacional



Cada linha da tabela **Cliente**
se relaciona com **zero ou mais linhas**
da tabela **Investimento**

Diagrama do modelo relacional



Cada linha da tabela **Investimento**
se **relaciona necessariamente com**
uma linha da tabela **Cliente**

I. O comando SQL que lista todos os pares de clientes que residem na mesma cidade é:

```
select c1.NomeCliente, c2.NomeCliente
from   Cliente c1, Cliente c2
where  c1.CidadeCliente = c2.CidadeCliente
and    c1.CodCliente < c2.CodCliente;
```



1. **FROM** `<source_tables>` : indica as tabelas que serão usadas nesta query e, conceitualmente, combina estas tabelas através de *produto cartesiano* em uma grande tabela. (Note o termo "*conceitualmente*" que usei: em termos de implementação da query este produto cartesiano raramente é construído.)
2. **WHERE** `<filter_expression>` : filtra linhas.
3. **GROUP BY** `<grouping_expressions>` : agrupa conjuntos de linhas.
4. **SELECT** `<select_heading>` : escolha de colunas e de agregados.
5. **HAVING** `<filter_expression>` : outra filtragem, esta aplicada apenas **depois** da agregação. Pode usar resultados do processo de agregação. Obriga o uso de **GROUP BY**.
6. **DISTINCT** : Elimina linhas duplicadas.
7. **ORDER BY** : ordena as linhas do resultado.
8. **OFFSET** `<count>` : Pula linhas do resultado. Requer **LIMIT**.
9. **LIMIT** `<count>` : Mantém apenas um número máximo de linhas.

I. O comando SQL que lista todos os pares de clientes que residem na mesma cidade é:

```
3 select c1.NomeCliente, c2.NomeCliente  
1 from Cliente c1, Cliente c2  
2 where c1.CidadeCliente = c2.CidadeCliente  
and c1.CodCliente < c2.CodCliente;
```

I. O comando SQL que lista todos os pares de clientes que residem na mesma cidade é:

```
select c1.NomeCliente, c2.NomeCliente
1 from Cliente c1, Cliente c2
where c1.CidadeCliente = c2.CidadeCliente
and c1.CodCliente < c2.CodCliente;
```

ProdutoCartesiano	
c1.codCliente	
c1.nomeCliente	
c1.enderCliente	
c1.cidadeCliente	
c1.anoIngressoCliente	
c2.codCliente	
c2.nomeCliente	
c2.enderCliente	
c2.cidadeCliente	
c2.anoIngressoCliente	

CONCEITUALMENTE, o resultado do from é o produto cartesiano entre c1 (uma cópia da tabela Cliente) e c2 (outra cópia da tabela Cliente).

A IMPLEMENTAÇÃO REAL É MUITO MAIS OTIMIZADA

Se a tabela Cliente tem N linhas, o produto cartesiano tem N^2 linhas.

I. O comando SQL que lista todos os pares de clientes que residem na mesma cidade é:

```
select c1.NomeCliente, c2.NomeCliente  
from Cliente c1, Cliente c2
```

2

```
where c1.CidadeCliente = c2.CidadeCliente  
and c1.CodCliente < c2.CodCliente;
```

ProdutoCartesiano	
c1.codCliente	←
c1.nomeCliente	
c1.enderCliente	
c1.cidadeCliente	
c1.anoIngressoCliente	
c2.codCliente	←
c2.nomeCliente	
c2.enderCliente	
c2.cidadeCliente	
c2.anoIngressoCliente	

... e estes estão em
ordem segundo a
clausula where

Mantem apenas
aqueles em que estes
são iguais...

I. O comando SQL que lista todos os pares de clientes que residem na mesma cidade é:

```
3 select c1.NomeCliente, c2.NomeCliente  
   from Cliente c1, Cliente c2  
   where c1.CidadeCliente = c2.CidadeCliente  
   and   c1.CodCliente < c2.CodCliente;
```

Resultado
c1.nomeCliente
c2.nomeCliente

A clausula **where** manteve apenas as linhas onde as cidades eram as mesmas e os codigos de cliente estavam ordenados.

Ou seja, cada linha tinha as informações de dois clientes que moravam na mesma cidade, sem repetir os pares!

Conclusão: a afirmação
é VERDADEIRA

Agora estamos apenas selecionando os campos de nomes destes clientes.

- I é verdadeira

É correto apenas o que se afirma em

A I.

~~**B** II.~~

C I e III.

~~**D** II e IV.~~

~~**E** III e IV.~~

II. O comando SQL que retorna, para cada cidade, os clientes mais antigos é:

```
select codCliente, nomeCliente, cidadeCliente, anoIngressoCliente  
from Cliente  
where anoIngressoCliente in (select min(anoIngressoCliente)  
from Cliente  
group by cidadeCliente);
```

SUBQUERY

II. O comando SQL que retorna, para cada cidade, os clientes mais antigos é:

```

select codCliente, nomeCliente, cidadeCliente, anoIngressoCliente
from Cliente
where anoIngressoCliente in (select min(anoIngressoCliente)
1 from Cliente
2 group by cidadeCliente);
3

```



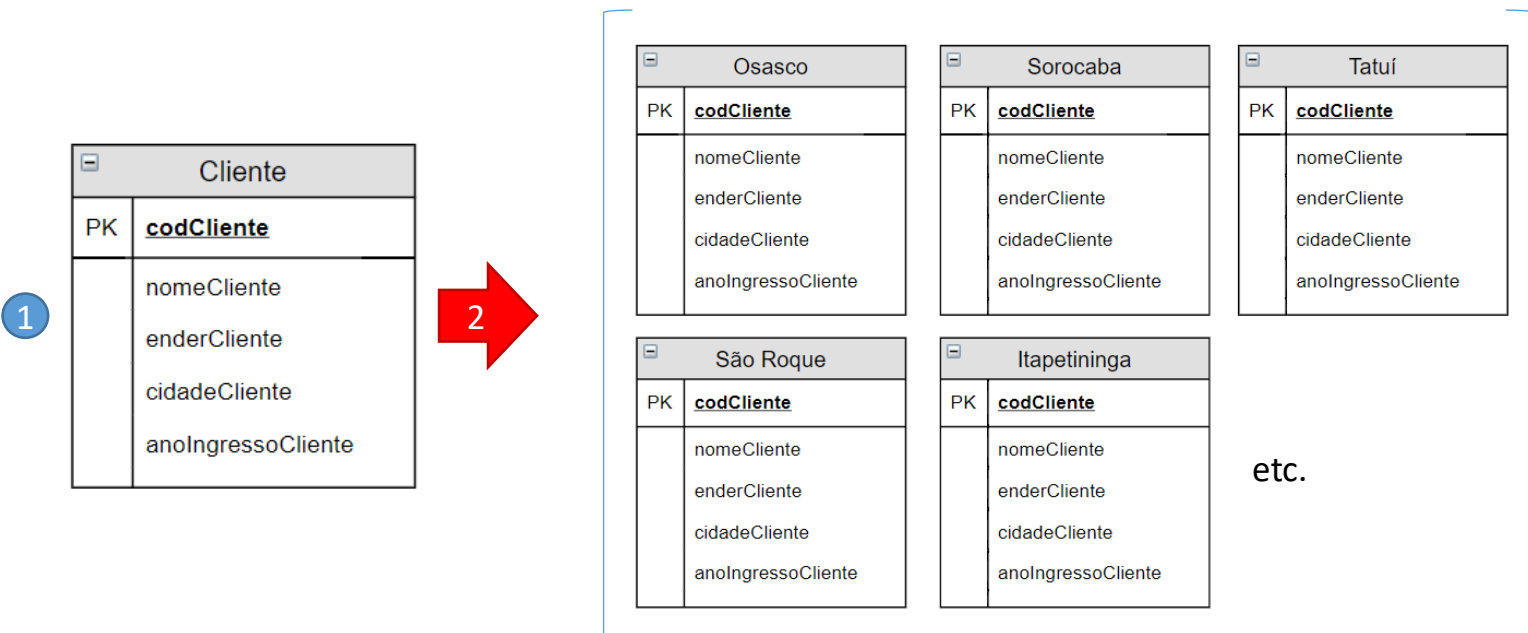
II. O comando SQL que retorna, para cada cidade, os clientes mais antigos é:

```
select codCliente, nomeCliente, cidadeCliente, anoIngressoCliente  
from Cliente  
where anoIngressoCliente in (select min(anoIngressoCliente)  
1 from Cliente  
group by cidadeCliente);
```

Cliente	
PK	<u>codCliente</u>
1	nomeCliente enderCliente cidadeCliente anoIngressoCliente

II. O comando SQL que retorna, para cada cidade, os clientes mais antigos é:

```
select codCliente, nomeCliente, cidadeCliente, anoIngressoCliente
from Cliente
where anoIngressoCliente in (select min(anoIngressoCliente)
from Cliente
group by cidadeCliente);
```



II. O comando SQL que retorna, para cada cidade, os clientes mais antigos é:

```
select codCliente, nomeCliente, cidadeCliente, anoIngressoCliente
from Cliente
where anoIngressoCliente in (select min(anoIngressoCliente)
from Cliente
group by cidadeCliente);
```

Osasco	
PK	<u>codCliente</u>
	nomeCliente
	enderCliente
	cidadeCliente
	anoIngressoCliente

Sorocaba	
PK	<u>codCliente</u>
	nomeCliente
	enderCliente
	cidadeCliente
	anoIngressoCliente

Tatuí	
PK	<u>codCliente</u>
	nomeCliente
	enderCliente
	cidadeCliente
	anoIngressoCliente



Resultado	
	min(anoIngressoCliente)

São Roque	
PK	<u>codCliente</u>
	nomeCliente
	enderCliente
	cidadeCliente
	anoIngressoCliente

Itapetininga	
PK	<u>codCliente</u>
	nomeCliente
	enderCliente
	cidadeCliente
	anoIngressoCliente

etc.

Uma tabela com o menor ano de ingresso para cada cidade, **MAS SEM A IDENTIFICAÇÃO DA CIDADE EM SI**

II. O comando SQL que retorna, para cada cidade, os clientes mais antigos é:

```
select codCliente, nomeCliente, cidadeCliente, anoIngressoCliente
from Cliente
where anoIngressoCliente in (select min(anoIngressoCliente)
from Cliente
group by cidadeCliente);
```

Osasco	
PK	<u>codCliente</u>
	nomeCliente
	enderCliente
	cidadeCliente
	anoIngressoCliente

Sorocaba	
PK	<u>codCliente</u>
	nomeCliente
	enderCliente
	cidadeCliente
	anoIngressoCliente

Tatuí	
PK	<u>codCliente</u>
	nomeCliente
	enderCliente
	cidadeCliente
	anoIngressoCliente



Resultado	
	min(anoIngressoCliente)

São Roque	
PK	<u>codCliente</u>
	nomeCliente
	enderCliente
	cidadeCliente
	anoIngressoCliente

Itapetininga	
PK	<u>codCliente</u>
	nomeCliente
	enderCliente
	cidadeCliente
	anoIngressoCliente

etc.

Uma tabela com o menor ano de ingresso para cada cidade, **MAS SEM A IDENTIFICAÇÃO DA CIDADE EM SI**

II. O comando SQL que retorna, para cada cidade, os clientes mais antigos é:

```

3 select codCliente, nomeCliente, cidadeCliente, anoIngressoCliente
1 from Cliente
2 where anoIngressoCliente in (select min(anoIngressoCliente)
from Cliente
group by cidadeCliente);

```

1

Cliente	
PK	codCliente
	nomeCliente
	enderCliente
	cidadeCliente
	anoIngressoCliente



Cliente	
PK	codCliente
	nomeCliente
	enderCliente
	cidadeCliente
	anoIngressoCliente

Retem apenas os clientes cujo ano de ingresso **É IGUAL A ALGUM ANO MÍNIMO DE INGRESSO, DE ALGUMA CIDADE QUALQUER!**

Não quer dizer que aquele cliente seja o mais antigo de sua cidade, mas sim que ele poderia ser o mais antigo de alguma cidade

Podemos parar por aqui. A consulta não permite obter apenas os clientes mais antigos de cada cidade. A afirmação é **FALSA**

II. O comando SQL que retorna, para cada cidade, os clientes mais antigos é:

```
select codCliente, nomeCliente, cidadeCliente, anoIngressoCliente
from Cliente
where anoIngressoCliente in (select min(anoIngressoCliente)
from Cliente
group by cidadeCliente);
```

Cliente	
PK	<u>codCliente</u>
	nomeCliente enderCliente cidadeCliente anoIngressoCliente

- I é verdadeira
- II é falsa

É correto apenas o que se afirma em

A I.

~~**B** II.~~

C I e III.

~~**D** II e IV.~~

~~**E** III e IV.~~

III. O comando SQL que retorna, para cada cidade (de um cliente), o ano de ingresso mais antigo, porém apenas para as cidades com mais de um cliente, é:

```
4 select cidadeCliente, min(AnoIngressoCliente)
1 from Cliente
2 group by cidadeCliente
3 having count(*) > 1;
```

III. O comando SQL que retorna, para cada cidade (de um cliente), o ano de ingresso mais antigo, porém apenas para as cidades com mais de um cliente, é:

```
select cidadeCliente, min(AnoIngressoCliente)
1 from Cliente
2 group by cidadeCliente
having count(*) > 1;
```

1

Cliente	
PK	<u>codCliente</u>
	nomeCliente
	enderCliente
	cidadeCliente
	anoIngressoCliente



2

Osasco	
PK	<u>codCliente</u>
	nomeCliente
	enderCliente
	cidadeCliente
	anoIngressoCliente

Sorocaba	
PK	<u>codCliente</u>
	nomeCliente
	enderCliente
	cidadeCliente
	anoIngressoCliente

Tatuí	
PK	<u>codCliente</u>
	nomeCliente
	enderCliente
	cidadeCliente
	anoIngressoCliente

São Roque	
PK	<u>codCliente</u>
	nomeCliente
	enderCliente
	cidadeCliente
	anoIngressoCliente

Itapetininga	
PK	<u>codCliente</u>
	nomeCliente
	enderCliente
	cidadeCliente
	anoIngressoCliente

etc.

III. O comando SQL que retorna, para cada cidade (de um cliente), o ano de ingresso mais antigo, porém apenas para as cidades com mais de um cliente, é:

```

1  select cidadeCliente, min(AnoIngressoCliente)
2  from Cliente
3  group by cidadeCliente
   having count(*) > 1;

```



Osasco	
PK	<u>codCliente</u>
	OK!
	anoIngressoCliente

Sorocaba	
PK	<u>codCliente</u>
	OK!
	anoIngressoCliente

Tatuí	
PK	<u>codCliente</u>
	OK!
	anoIngressoCliente

São Roque	
PK	<u>codCliente</u>
	OK!
	anoIngressoCliente

Itapetininga	
PK	<u>codCliente</u>
	OK!
	anoIngressoCliente

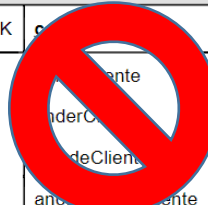
etc.

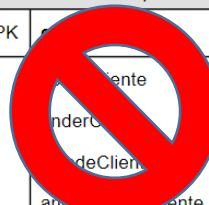
III. O comando SQL que retorna, para cada cidade (de um cliente), o ano de ingresso mais antigo, porém apenas para as cidades com mais de um cliente, é:

4 `select cidadeCliente, min(AnoIngressoCliente)`
 1 `from Cliente`
 2 `group by cidadeCliente`
 3 `having count(*) > 1;`

Osasco	
PK	<u>codCliente</u>
	OK!
	anoIngressoCliente

Sorocaba	
PK	<u>codCliente</u>
	OK!
	anoIngressoCliente

Tatuí	
PK	<u>codCliente</u>
	
	anoIngressoCliente

São Roque	
PK	<u>codCliente</u>
	
	anoIngressoCliente

Itapetininga	
PK	<u>codCliente</u>
	OK!
	anoIngressoCliente

etc.

Para cada cidade, o ano de ingresso mais antigo



Resultado	
	cidadeCliente
	min(anoIngressoCliente)

Portanto, a afirmação é
VERDADEIRA

- I é verdadeira
- II é falsa
- III é verdadeira

É correto apenas o que se afirma em

~~A I.~~

~~B II.~~

C I e III.

~~D II e IV.~~

~~E III e IV.~~

Resposta correta: C

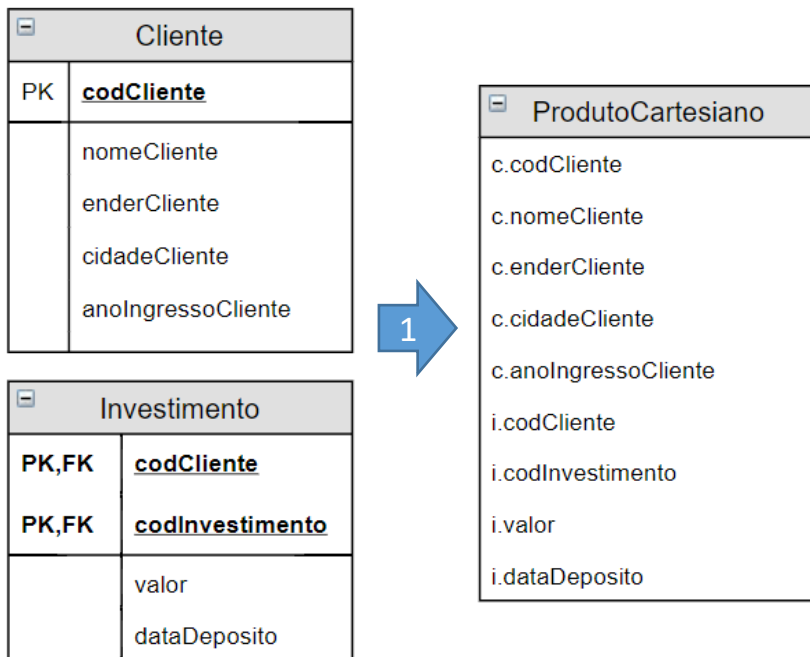
IV. O comando SQL que retorna o maior valor de cada investimento de cada cliente é:

```
4 select codCliente, max(valor)
1 from Cliente c, Investimento i
2 where c.codCliente = i.codCliente
3 group by codCliente, codInvestimento
```

IV. O comando SQL que retorna o maior valor de cada investimento de cada cliente é:

32

```
1 select codCliente, max(valor)
   from Cliente c, Investimento i
  where c.codCliente = i.codCliente
 group by codCliente, codInvestimento
```



IV. O comando SQL que retorna o maior valor de cada investimento de cada cliente é:

33

```
select codCliente, max(valor)
1 from Cliente c, Investimento i
2 where c.codCliente = i.codCliente
group by codCliente, codInvestimento
```

ProdutoCartesiano
c.codCliente
c.nomeCliente
c.enderCliente
c.cidadeCliente
c.anoIngressoCliente
i.codCliente
i.codInvestimento
i.valor
i.dataDeposito



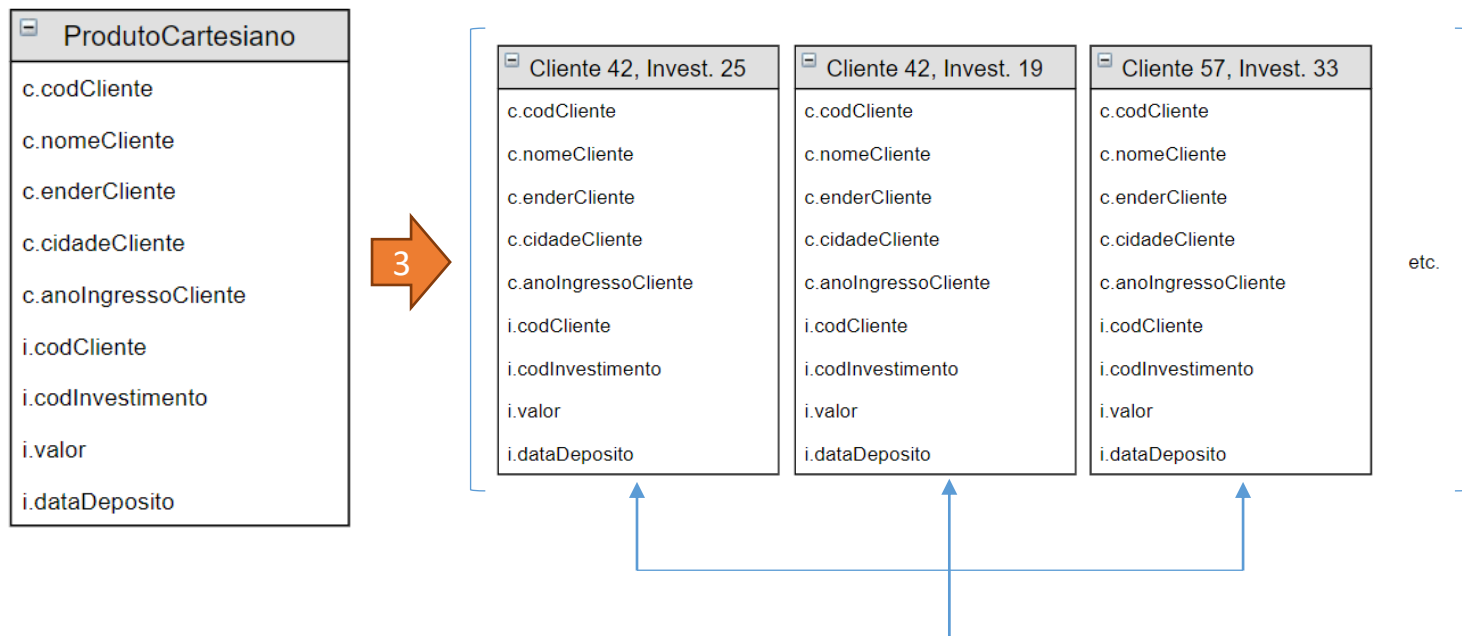
ProdutoCartesiano
c.codCliente
c.nomeCliente
c.enderCliente
c.cidadeCliente
c.anoIngressoCliente
i.codCliente
i.codInvestimento
i.valor
i.dataDeposito

Mantem apenas
linhas onde estes
coincidem

IV. O comando SQL que retorna o maior valor de cada investimento de cada cliente é:

34

```
select codCliente, max(valor)
1 from Cliente c, Investimento i
2 where c.codCliente = i.codCliente
3 group by codCliente, codInvestimento
```



Tabelas de uma linha só!

IV. O comando SQL que retorna o maior valor de cada investimento de cada cliente é:

35

```
4 select codCliente, max(valor)
1 from Cliente c, Investimento i
2 where c.codCliente = i.codCliente
3 group by codCliente, codInvestimento
```

Cliente 42, Invest. 25	Cliente 42, Invest. 19	Cliente 57, Invest. 33
c.codCliente	c.codCliente	c.codCliente
c.nomeCliente	c.nomeCliente	c.nomeCliente
c.enderCliente	c.enderCliente	c.enderCliente
c.cidadeCliente	c.cidadeCliente	c.cidadeCliente
c.anoIngressoCliente	c.anoIngressoCliente	c.anoIngressoCliente
i.codCliente	i.codCliente	i.codCliente
i.codInvestimento	i.codInvestimento	i.codInvestimento
i.valor	i.valor	i.valor
i.dataDeposito	i.dataDeposito	i.dataDeposito

etc.



Resultado
codCliente
max(valor)

Parece correto mas...

codCliente	max(valor)
42	R\$ 15.623,09
42	R\$ 3.354,12
57	R\$ 45.034,92
etc	

← Valor do investimento 25 do cliente 42
← Valor do investimento 19 do cliente 42
← Valor do investimento 33 do cliente 57
...

Não faz o que
é pedido.
Afirmação
FALSA



Insper

Preparação ENADE Megadados

ENADE 2014

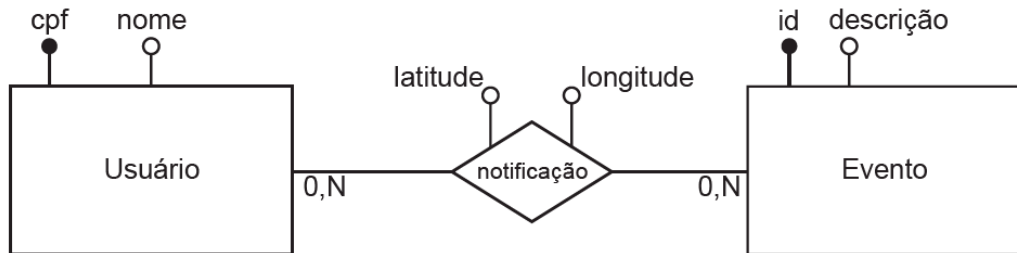
Ciência da Computação

Questão 4

- Modelo entidade-relacionamento

QUESTÃO DISCURSIVA 4

Muitas aplicações utilizam o sistema de localização (GPS) do dispositivo móvel do usuário para descobrir qual o melhor caminho a seguir. Algumas aplicações também permitem que o usuário notifique a ocorrência de eventos que ele presencia durante seu percurso, tais como acidentes ou trânsito lento. Em um possível cenário, esta notificação é enviada para um servidor centralizado, o qual é responsável por disseminar a notificação para os demais usuários do aplicativo. Uma equipe de desenvolvimento criou uma aplicação desse tipo utilizando uma base de dados relacional para o armazenamento de dados referentes aos usuários, eventos e notificações enviadas. A modelagem conceitual foi feita utilizando o diagrama entidade-relacionamento conforme apresentado na figura a seguir.

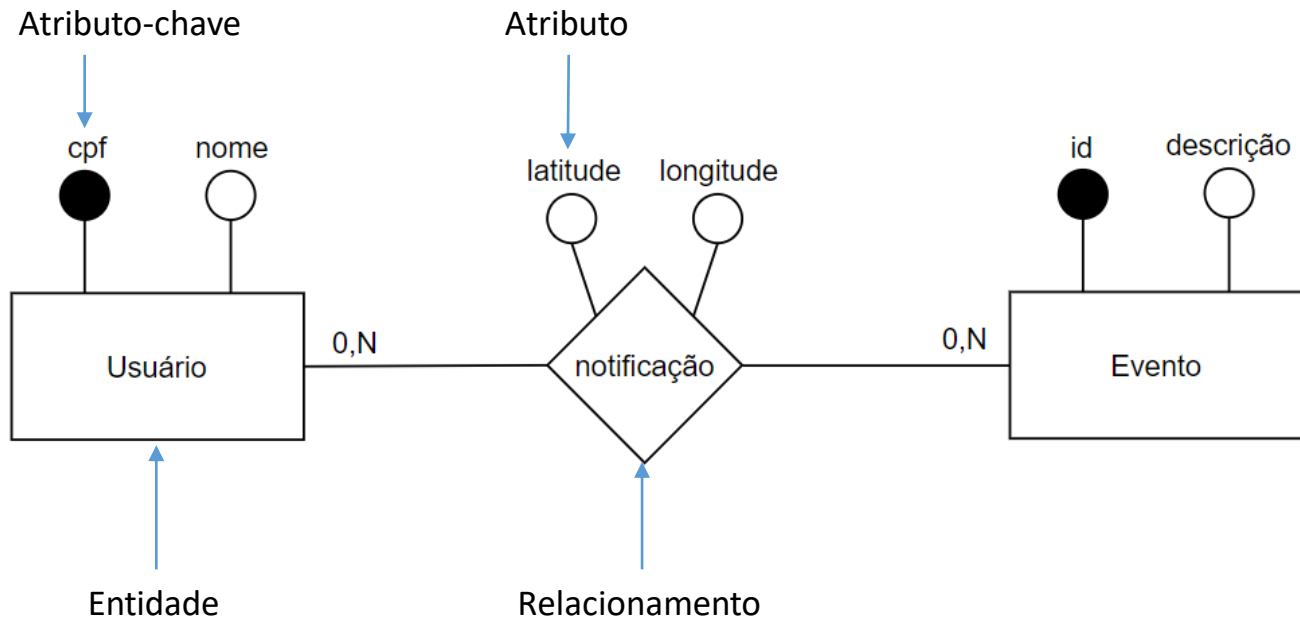


A equipe de desenvolvimento deseja adicionar as seguintes características ao modelo:

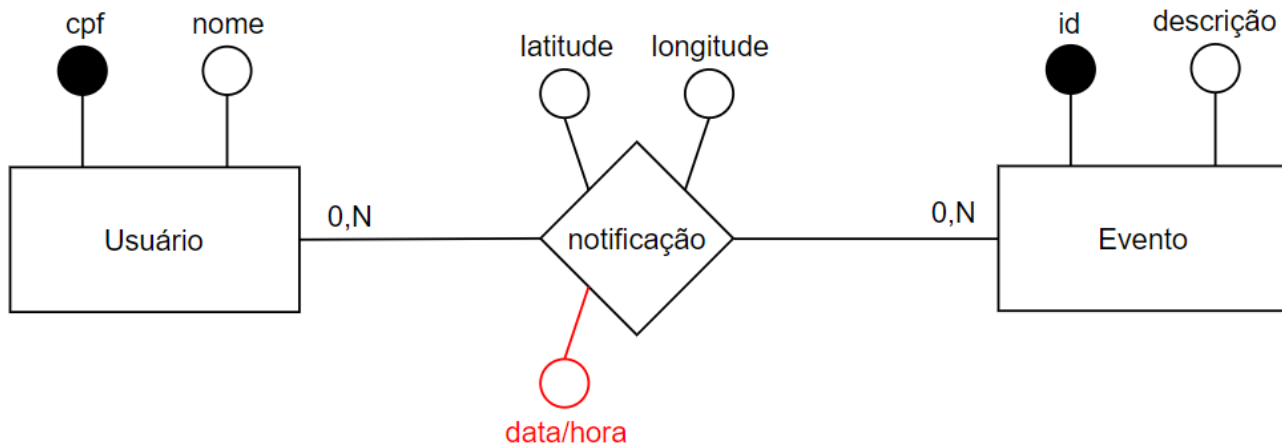
- Cada notificação deve ter data e hora;
- O grupo de usuários para o qual uma notificação é enviada deve ser restrito. Cada usuário deve ter um grupo com um número arbitrário de amigos, que também são usuários da aplicação, e as notificações enviadas por um usuário devem ser enviadas somente a seus amigos. Também se deseja armazenar informações sobre quais notificações foram enviadas para quais usuários.

Nessa situação, adapte o diagrama ER da figura para atender os novos requisitos.(valor: 10,0 pontos)

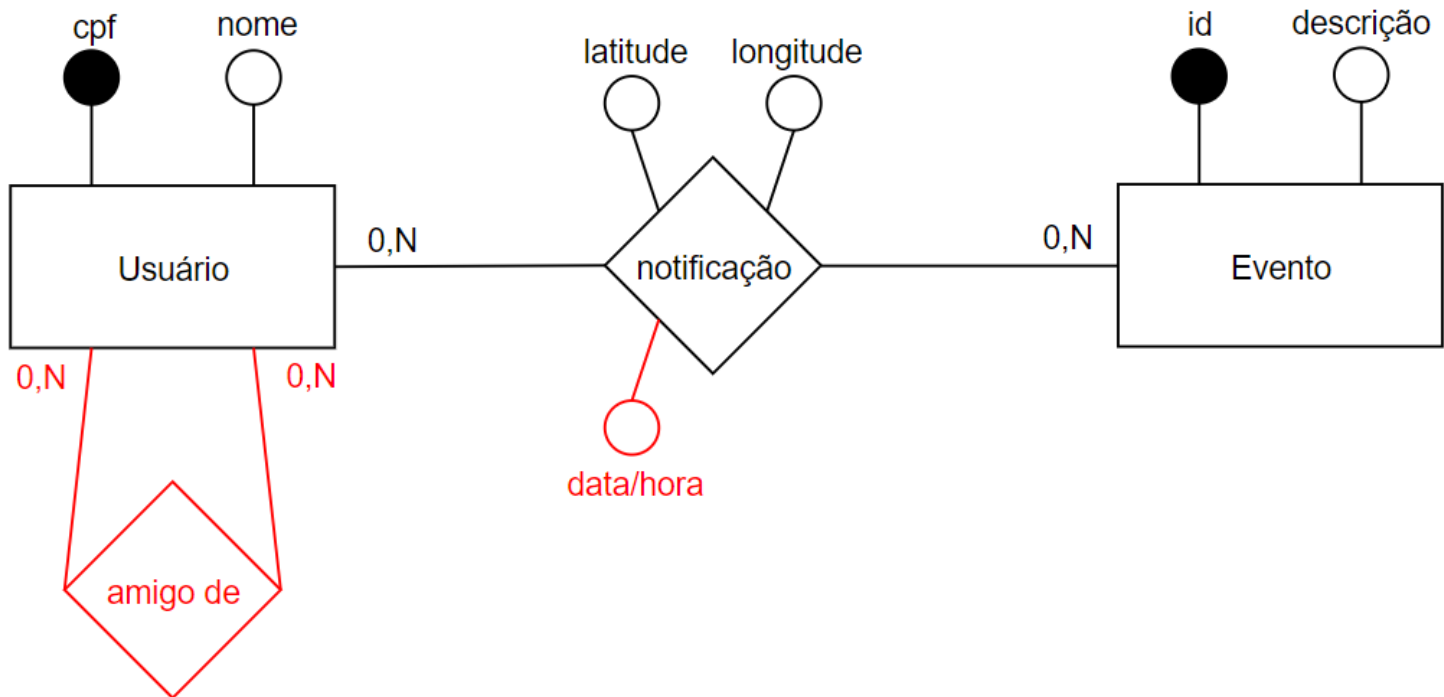
Diagrama Entidade-Relacionamento



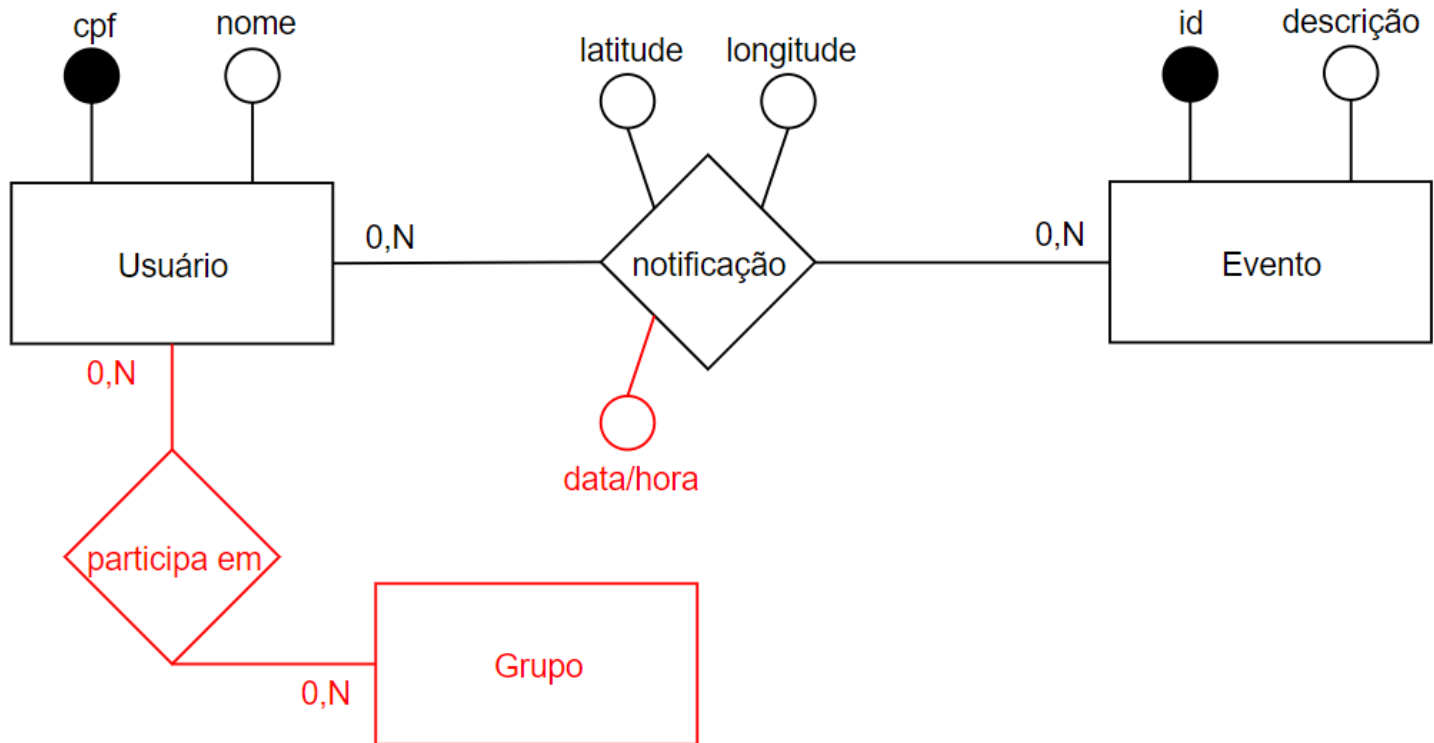
“Cada notificação deve ter data e hora”



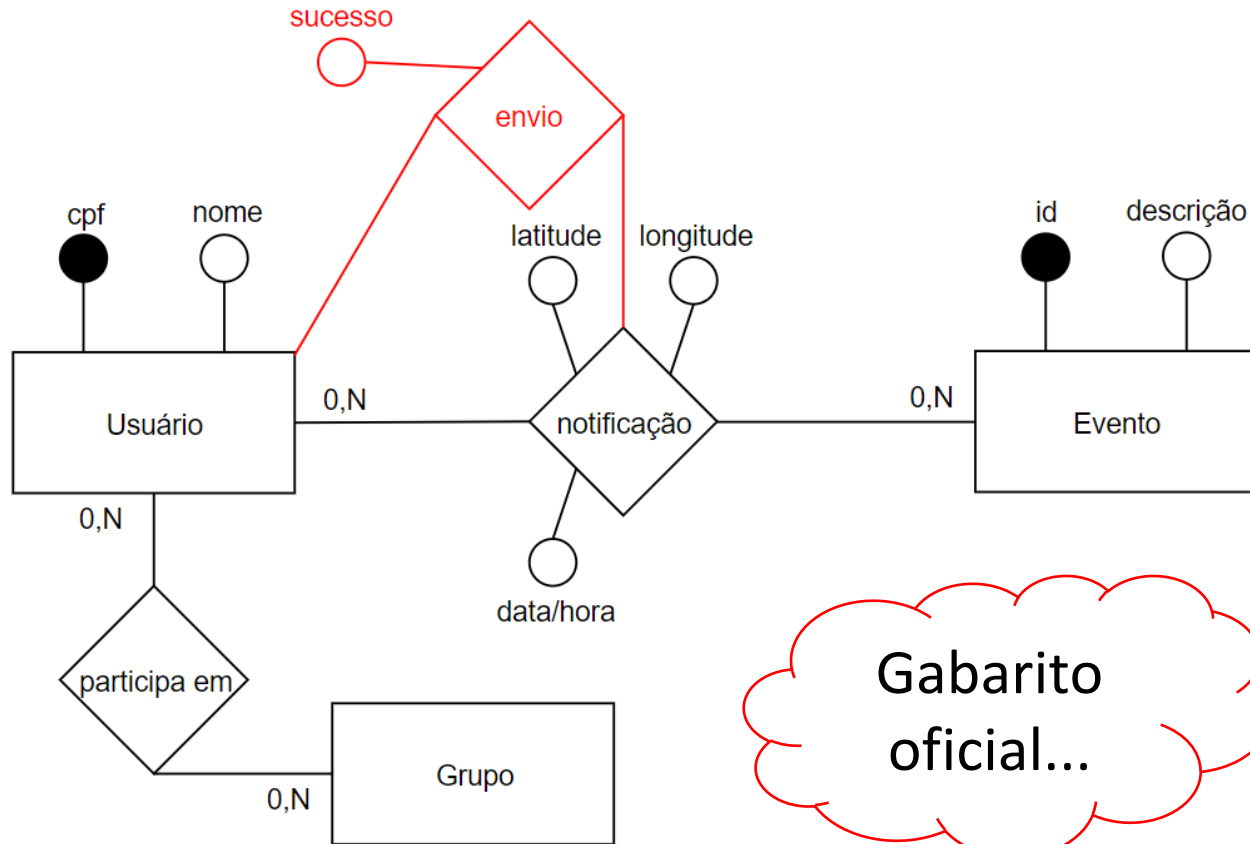
- O grupo de usuários para o qual uma notificação é enviada deve ser restrito.
- Cada usuário deve ter um grupo com um número arbitrário de amigos, que também são usuários da aplicação, e
- As notificações enviadas por um usuário devem ser enviadas somente a seus amigos.



- O grupo de usuários para o qual uma notificação é enviada deve ser restrito.
- Cada usuário deve ter um grupo com um número arbitrário de amigos, que também são usuários da aplicação, e
- As notificações enviadas por um usuário devem ser enviadas somente a seus amigos.



- Também se deseja armazenar informações sobre quais notificações foram enviadas para quais usuários.

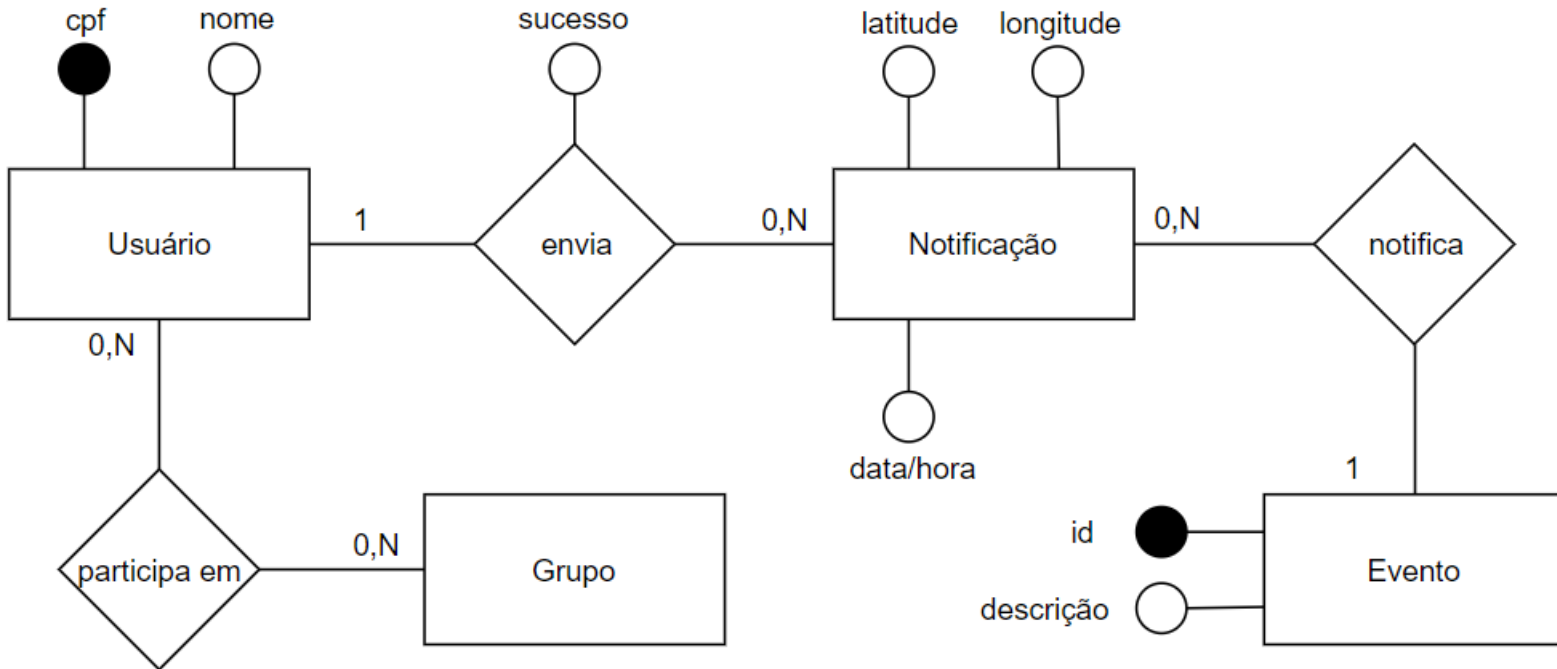


Como definir quem é quem?

Gramática	Função	Diagrama E-R
Substantivo	Nomeia um objeto	Entidade
Verbo transitivo	Nomeia uma ação	Relacionamento
Adjetivo	Caracteriza um objeto	Atributo de entidade
Advérbio	Caracteriza uma ação (quando, onde, como, quanto, etc.)	Atributo de relacionamento

Chen, Peter P.-S. *English, Chinese, and ER Diagrams*.
Data & Knowledge Engineering, 23 (1997) 5-16.

- Também se deseja armazenar informações sobre quais notificações foram enviadas para quais usuários.





Insper

Preparação ENADE Megadados

ENADE 2005

Engenharia da Computação

Questão 72

- Transações

A execução de duas transações, T_i e T_j , em um banco de dados, é serializável se produz o mesmo resultado para a execução serial de qualquer intercalação de operações dessas transações (T_i seguida de T_j ou T_j seguida de T_i). O uso de bloqueios (*locks*) é uma maneira de se garantir que transações concorrentes sejam serializáveis. A tabela acima mostra informações relativas a três transações, T_1 , T_2 e T_3 , que operam sobre dois dados compartilhados, A e B, e utilizam bloqueios para controle de concorrência. Com relação às transações T_1 , T_2 e T_3 , julgue os itens seguintes.

QUESTÃO 72

	T₁	T₂	T₃
1	bloqueia A	bloqueia B	bloqueia B
2	recupera A	recupera B	recupera B
3	atualiza A	atualiza B	atualiza B
4	desbloqueia A	bloqueia A	bloqueia A
5	bloqueia B	recupera A	recupera A
6	recupera B	atualiza A	desbloqueia A
7	atualiza B	desbloqueia A	desbloqueia A
8	desbloqueia B	desbloqueia B	desbloqueia B

- I O conjunto (T_1, T_2) não é serializável, e há o perigo de ocorrer *deadlock* durante a execução concorrente dessas transações.
- II O conjunto (T_1, T_3) não é serializável, mas não há o perigo de ocorrer *deadlock* durante a execução concorrente dessas transações.
- III O conjunto (T_2, T_3) é serializável, e não há o perigo de ocorrer *deadlock* durante a execução concorrente dessas transações.

Assinale a opção correta.

- A** Apenas um item está certo.
- B** Apenas os itens I e II estão certos.
- C** Apenas os itens I e III estão certos.
- D** Apenas os itens II e III estão certos.
- E** Todos os itens estão certos.

QUESTÃO 72

	T ₁	T ₂	T ₃
1	bloqueia A	bloqueia B	bloqueia B
2	recupera A	recupera B	recupera B
3	atualiza A	atualiza B	atualiza B
4	desbloqueia A	bloqueia A	bloqueia A
5	bloqueia B	recupera A	recupera A
6	recupera B	atualiza A	desbloqueia A
7	atualiza B	desbloqueia A	desbloqueia A
8	desbloqueia B	desbloqueia B	desbloqueia B

Será que teremos deadlock?

Como detectar um deadlock?

Um deadlock acontece quando existe uma sequência na qual duas transações tentam adquirir recursos que estão mutuamente bloqueados

Transação T_1	Transação T_2
Bloqueia A	Bloqueia B
Bloqueia B	Bloqueia A
Libera B	Libera A
Libera A	Libera B

Transação T_1	Transação T_2
Bloqueia A	Bloqueia B
Bloqueia B	Bloqueia A
Libera B	Libera A
Libera A	Libera B

A: Liberado

B: Liberado

Transação T_1	Transação T_2
Bloqueia A 1	Bloqueia B
Bloqueia B	Bloqueia A
Libera B	Libera A
Libera A	Libera B


A: Bloqueado

B: Liberado

Transação T_1	Transação T_2
Bloqueia A 1	Bloqueia B
Bloqueia B 2	Bloqueia A
Libera B	Libera A
Libera A	Libera B


A: Bloqueado

B: Bloqueado

Transação T_1	Transação T_2
Bloqueia A ①	Bloqueia B ③ 
Bloqueia B ②	Bloqueia A
Libera B	Libera A
Libera A	Libera B

A: Bloqueado

B: Bloqueado

Transação T_1	Transação T_2
Bloqueia A ①	Bloqueia B ③ 
Bloqueia B ②	Bloqueia A
Libera B ④	Libera A
Libera A	Libera B


A: Bloqueado

B: Liberado

Transação T_1	Transação T_2
Bloqueia A ①	Bloqueia B ⑤
Bloqueia B ②	Bloqueia A
Libera B ④	Libera A
Libera A	Libera B

A: Bloqueado

B: Liberado

Transação T_1	Transação T_2
Bloqueia A ①	Bloqueia B ⑤
Bloqueia B ②	Bloqueia A ⑥ 
Libera B ④	Libera A
Libera A	Libera B

A: Bloqueado

B: Liberado

Transação T_1	Transação T_2
Bloqueia A ①	Bloqueia B ⑤
Bloqueia B ②	Bloqueia A ⑧
Libera B ④	Libera A ⑨
Libera A ⑦	Libera B ⑩

A: Liberado

B: Liberado

Sem deadlock

Transação T_1	Transação T_2
Bloqueia A	Bloqueia B
Bloqueia B	Bloqueia A
Libera B	Libera A
Libera A	Libera B


A: Liberado

B: Liberado

Transação T_1	Transação T_2
Bloqueia A 1	Bloqueia B 2
Bloqueia B	Bloqueia A
Libera B	Libera A
Libera A	Libera B



A: Bloqueado

B: Bloqueado

Transação T_1	Transação T_2
Bloqueia A ①	Bloqueia B ②
Bloqueia B ③ 	Bloqueia A
Libera B	Libera A
Libera A	Libera B

A: Bloqueado

B: Bloqueado

Transação T_1	Transação T_2
Bloqueia A ①	Bloqueia B ②
Bloqueia B ③ 	Bloqueia A ④ 
Libera B	Libera A
Libera A	Libera B

A: Bloqueado

B: Bloqueado

Deadlock

Transação T_1	Transação T_2
Bloqueia A	Bloqueia A
Bloqueia B	Bloqueia B
Libera B	Libera B
Libera A	Libera A

- Transação T_2 só consegue travar B depois de travar A
- Portanto ninguém trava B depois que T_1 começa
- T_1 libera B normalmente, depois libera A

Não temos deadlock

QUESTÃO 72

	T_1	T_2	T_3
1	bloqueia A	bloqueia B	bloqueia B
2	recupera A	recupera B	recupera B
3	atualiza A	atualiza B	atualiza B
4	desbloqueia A	bloqueia A	bloqueia A
5	bloqueia B	recupera A	recupera A
6	recupera B	atualiza A	desbloqueia A
7	atualiza B	desbloqueia A	desbloqueia A
8	desbloqueia B	desbloqueia B	desbloqueia B

- Entre T_1 e T_2 ou T_3 : sem deadlock
 - T_1 não tem regiões críticas aninhadas
- Entre T_2 e T_3 : sem deadlock
 - Travamento e destravamento na mesma ordem

- I ~~O conjunto (T_1, T_2) não é serializável, e há o perigo de ocorrer *deadlock* durante a execução concorrente dessas transações.~~
- II O conjunto (T_1, T_3) não é serializável, mas não há o perigo de ocorrer *deadlock* durante a execução concorrente dessas transações.
- III O conjunto (T_2, T_3) é serializável, e não há o perigo de ocorrer *deadlock* durante a execução concorrente dessas transações.

QUESTÃO 72

	T ₁	T ₂	T ₃
1	bloqueia A	bloqueia B	bloqueia B
2	recupera A	recupera B	recupera B
3	atualiza A	atualiza B	atualiza B
4	desbloqueia A	bloqueia A	bloqueia A
5	bloqueia B	recupera A	recupera A
6	recupera B	atualiza A	desbloqueia A
7	atualiza B	desbloqueia A	desbloqueia A
8	desbloqueia B	desbloqueia B	desbloqueia B

QUESTÃO 72

	T_1	T_2	T_3
1	bloqueia A	bloqueia B	bloqueia B
2	recupera A	recupera B	recupera B
3	atualiza A	atualiza B	atualiza B
4	desbloqueia A	bloqueia A	bloqueia A
5	bloqueia B	recupera A	recupera A
6	recupera B	atualiza A	desbloqueia A
7	atualiza B	desbloqueia A	desbloqueia B
8	desbloqueia B	desbloqueia B	

T_1 antes de T_3

QUESTÃO 72

	T_1	T_2	T_3
1	bloqueia A	bloqueia B	bloqueia B
2	recupera A	recupera B	recupera B
3	atualiza A	atualiza B	atualiza B
4	desbloqueia A	bloqueia A	bloqueia A
5	bloqueia B	recupera A	recupera A
6	recupera B	atualiza A	desbloqueia A
7	atualiza B	desbloqueia A	desbloqueia A
8	desbloqueia B	desbloqueia B	desbloqueia B

- A: mudanças de T_1
- B: mudanças de T_3

- T_1 : leu A antigo, B antigo
- T_3 : leu A novo, B novo

T_3 antes de T_1

QUESTÃO 72

	T_1	T_2	T_3
1	bloqueia A	bloqueia B	bloqueia B
2	recupera A	recupera B	recupera B
3	atualiza A	atualiza B	atualiza B
4	desbloqueia A	bloqueia A	bloqueia A
5	bloqueia B	recupera A	recupera A
6	recupera B	atualiza A	desbloqueia A
7	atualiza B	desbloqueia A	desbloqueia A
8	desbloqueia B	desbloqueia B	desbloqueia B

- A: mudanças de T_1
- B: mudanças de T_1

- T_1 : leu A antigo, B novo
- T_3 : leu A antigo, B antigo

T_1 intercalado com T_3

QUESTÃO 72

	T_1	T_2	T_3
1	bloqueia A	bloqueia B	bloqueia B
2	recupera A	recupera B	recupera B
3	atualiza A	atualiza B	atualiza B
4	desbloqueia A	bloqueia A	bloqueia A
5	bloqueia B	recupera A	recupera A
6	recupera B	atualiza A	desbloqueia A
7	atualiza B	desbloqueia A	desbloqueia B
8	desbloqueia B	desbloqueia B	

- A: mudanças de T_1
- B: mudanças de T_1

- T_1 : leu A antigo, B novo
- T_3 : leu A novo, B novo

Serializa?

T_1 antes de T_3

- A: mudanças de T_1
- B: mudanças de T_3

- T_1 : leu A antigo, B antigo
- T_3 : leu A novo, B novo

T_3 antes de T_1

- A: mudanças de T_1
- B: mudanças de T_1


- T_1 : leu A antigo, B novo
- T_3 : leu A antigo, B antigo

Não serializa

T_1 intercalado
com T_3

- A: mudanças de T_1
- B: mudanças de T_1

- T_1 : leu A antigo, B novo
- T_3 : leu A novo, B novo



- I ~~O conjunto (T_1, T_2) não é serializável, e há o perigo de ocorrer *deadlock* durante a execução concorrente dessas transações.~~
- II O conjunto (T_1, T_3) não é  serializável, mas não há o perigo de ocorrer *deadlock* durante a execução concorrente dessas transações.
- III O conjunto (T_2, T_3) é serializável, e não há o perigo de ocorrer *deadlock* durante a execução concorrente dessas transações.

QUESTÃO 72

	T_1	T_2	T_3
1	bloqueia A	bloqueia B	bloqueia B
2	recupera A	recupera B	recupera B
3	atualiza A	atualiza B	atualiza B
4	desbloqueia A	bloqueia A	bloqueia A
5	bloqueia B	recupera A	recupera A
6	recupera B	atualiza A	desbloqueia A
7	atualiza B	desbloqueia A	desbloqueia A
8	desbloqueia B	desbloqueia B	desbloqueia B

Quando T_2 ou T_3 começam, elas devem terminar completamente antes que a outra transação comece.

⇒ Serializa

- I ~~O conjunto (T_1, T_2) não é serializável, e há o perigo de ocorrer *deadlock* durante a execução concorrente dessas transações.~~
- II O conjunto (T_1, T_3) não é  serializável, mas não há o perigo de ocorrer *deadlock* durante a execução concorrente dessas transações.
- III O conjunto (T_2, T_3) é  serializável, e não há o perigo de ocorrer *deadlock* durante a execução concorrente dessas transações.

Assinale a opção correta.

- A** Apenas um item está certo.
- B** Apenas os itens I e II estão certos.
- C** Apenas os itens I e III estão certos.
- D** Apenas os itens II e III estão certos.
- E** Todos os itens estão certos.