COLUNAS OPCIONAIS EM SQL

Sérgio Mergen

Introdução

- A pergunta:
  - Uma coluna deve ser especificada como null ou not null?
- Presunção inicial básica
  - Colunas nulas tornam o modelo mais flexível
  - No entanto, aumentam a complexidade de uma tabela
- Se uma coluna sempre terá valor definido para seus registros
  - Use NOT NULL
  - A flexibilidade dos campos nulos não será nunca usada.
  - A estrutura interna da tabela é simplificada.

Introdução

- Mas existem situações em que uma coluna pode conter valores nulos
- Exemplos
  1. O usuário não quer informar o valor
  2. A informação não estava disponível no momento do cadastro

- Nesse caso, pode-se
  - Definir a coluna como opcional
  - Definir a coluna como obrigatória

- Qual alternativa é melhor?

Sumário

- Organização física dos registros
- Uso de placeholders
- Semântica dos valores nulos em consultas
- Valores nulos em subconsultas
- Colunas opcionais e especialização
- Uso de divisão de tabela

Organização física dos registros

- Como vimos na aula anterior
  - o uso de colunas opcionais pode levar à economia de espaço
- No entanto
  - elas fazem os registros terem tamanho variável.
- Sendo assim
  - vale a pena investigar a diferença entre o uso de registros de tamanho fixo e tamanho variável

Organização física dos registros

- Registros de tamanho fixo (200 bytes cada)

reg1	reg2	reg3	reg4	reg5	reg6	reg7
0	200	400	600	800	1000	1200

- Registros de tamanho variável (máx. 200 bytes cada)

reg1	reg2	reg3	reg4	reg5	reg6	reg7
0	150	320	500	700	860	1010

Organização física dos registros

- Registros de tamanho fixo (200 bytes cada)

reg1	reg2	reg3	reg4	reg5	reg6	reg7
0	200	400	600	800	1000	1200

- Registros de tamanho variável (máx. 200 bytes cada)

reg1	reg2	reg3	reg4	reg5	reg6	reg7
0	150	320	500	700	860	1010

Organização física dos registros

- Vantagem dos registros de tamanho variável
  - A tabela ocupa menos espaço

- Vantagem dos registros de tamanho fixo
  - Permitem a busca eficiente através de offsets
    - Ex. Dentro da página, para encontrar o terceiro registro, considerando que os dois anteriores têm tamanho fixo (200 bytes)
      - Offset = 200 * 2
    - Ex. dentro do registro, para encontrar a posição da quarta coluna, se todas as anteriores forem do tipo INT (4 bytes)
      - Offset = 3 * 4

Organização física dos registros

- Quando um registro terá tamanho variável?

- Quando qualquer uma das colunas da tabela
  - Tiver tamanho variável
    - VARCHAR
    - TEXT
    - ...
  - Aceitar NULO

Organização física dos registros

- No MySQL (engine MyISAM), o tamanho fixo é usado quando a tabela não possui nenhuma coluna de tamanho variável
  - VARCHAR, VARBINARY, BLOB, TEXT

- Caso contrário
  - é usado o formato DYNAMIC

Organização física dos registros

- No MyISAM, pode-se forçar o tamanho fixo quando uma coluna possui tipo variável
  - CREATE TABLE PESSOA (...) ROW FORMAT = STATIC

- Nesse caso
  - VARCHAR se comporta como CHAR
  - VARBINARY se comporta como BINARY
  - BLOB e TEXT levam a um erro de criação de tabela

Organização física dos registros

- No MySQL (engine InnoDB), não existe um formato dedicado à registros de tamanho fixo
- Ou seja, não existe endereçamento direto através de offsets
  - Os registros são ligados através de ponteiros
- Dessa forma, para essa engine pouco importa se as colunas têm tamanho variável ou não

Organização física dos registros

- Exemplos de formatos de registro no InnoDB
  - COMPACT
  - DYNAMIC
  - COMPRESSED

- Ex. de uso
  - CREATE TABLE PESSOA (...) ROW FORMAT = DYNAMIC

Organização física dos registros

- COMPACT
  - Os primeiros 768 bytes de cada coluna são salvos na página onde está o registro
    - O restante é salvo em um bloco de estouro
- DYNAMIC
  - O SGBD pode armazenar todo o conteúdo de uma coluna em outra página, se julgar conveniente
- COMPRESSED
  - A página inteira é comprimida usando um algoritmo baseado em LZ77

Organização física dos registros

- Alguns SGBDs impõe limites ao tamanho dos registros, colunas e número de colunas por tabela
  - Isso está diretamente relacionado ao espaço que o registro ocupa na página.

- Exemplos de possíveis contornos quando um limite for atingido
  - Mudança no formato de armazenamento (ex. row-format = '**COMPRESSED**' no mySQL (**InnoDB**)
  - Uso do particionamento vertical

- É importante que o DBA conheça os recursos e configurações avançadas que cada fabricante de SGBD disponibiliza

Sumário

- Organização física dos registros
- Uso de placeholders
- Semântica dos valores nulos em consultas
- Valores nulos em subconsultas
- Colunas opcionais e especialização
- Uso de divisão de tabela

Uso de placeholders

- Caso uma coluna seja definida como obrigatória, e não tenha sido fornecido algum valor para ela
  - É necessário usar algum valor artificial (placeholder)
- Deve-se ter cuidado para evitar que os placeholders tragam resultados indesejados em consultas

Uso de placeholders

- Ex. Func (id, nome, email, sexo, salario, comissao)
  - Se **comissão** for uma coluna obrigatória
    - que valor usar se a comissão for desconhecida?

- Poderia ser usado um placeholder (ex. -1 ou 0)

Uso de placeholders

- Ex. Func (id, nome, email, sexo, salario, comissao)
  - Se **comissão** for uma coluna obrigatória
    - que valor usar se a comissão for desconhecida?

- Poderia ser usado um placeholder (ex. -1 ou 0)

- E se alguém quisesse filtrar pelo valor de comissão anual?
  - WHERE comissão * 12 < 3.000

  - **Os registros com valor artificial de comissão seriam erroneamente retornados**

Uso de placeholders

- Ex. Func (id, nome, email, sexo, salario, comissao)
  - Se **email** for uma coluna obrigatória
  - Que valor usar se o e-mail for desconhecido?

- Poderia ser usado um valor vazio(‘’)

Uso de placeholders

- Ex. Func (id, nome, email, sexo, salario, comissao)
  - Se **email** for uma coluna obrigatória
  - Que valor usar se o e-mail for desconhecido?
- Poderia ser usado um valor vazio(‘’)
- E se alguém quisesse os e-mails que não comecem com A?
  - WHERE email NOT LIKE ’A%’
  - **Os registros com valor vazio de email seriam erroneamente retornados**

Uso de placeholders

- Para evitar que resultados errados sejam retornados, é necessário incluir os placeholders na consulta
  - WHERE comissão * 12 < 3.000 **AND comissão <> -1**
  - WHERE email NOT LIKE 'A%' **AND email <> ''**

- Desvantagens;
  - Torna as consultas menos intuitivas
  - Necessário disciplina para acrescentar os placeholders
  - Pode afetar o desempenho

Uso de placeholders

- Já o uso de NULL deixa claro que o valor é desconhecido
  - E não simplesmente vazio ou preenchido com um valor qualquer

- **Observação:** O resultado de uma consulta SQL onde o valor do atributo é nulo pode ser diferente do resultado se o valor for artificial
  - Como veremos a seguir

Sumário

- Organização física dos registros
- Uso de placeholders
- Semântica dos valores nulos em consultas
- Valores nulos em subconsultas
- Colunas opcionais e especialização
- Uso de divisão de tabela

Valores Nulos e comparações

- Qualquer comparação com *null* retorna *unknown*
  - *Ex.*
    - *5 < null = unknown*
    - *null <> null = unknown*
    - *null = null = unknown*
- Qualquer expressão aritmética com *null* retorna *unknown*
  - *Ex.*
    - *5 * null = unknown*
    - *5 + null = unknown*
    - *null - null = unknown*

Valores Nulos e comparações

- Quando o valor da comparação for unknown
  - a condição da cláusula **where** é tratada como false
  - Ou seja, o registro não é retornado
- Geralmente não se quer que registros incertos sejam retornados
  - O SGBD não se compromete com o que ele desconhece
- Caso o objetivo seja retorná-los, deve-se incluir a comparação com desconhecido na consulta
  - where expr is unknown

Valores Nulos e comparações

Listar os projetos com tempo estimado igual ou inferior a 24 meses

Projeto				
idProj	**nome**	**duracao**	**custo**	**idDepto**
1	ABC	2	12.000	1
2	Lucrei	3	30.000	1
3	Show	null	5.000	2

select *nome*
from *projeto p*
where *duracao * 12 <= 24*

Resposta
nome
ABC

O SGBD não retorna 'Show', pois desconhece a sua duração

Valores Nulos e comparações

- Caso o objetivo seja retornar registros referentes a valores desconhecidos, deve-se recorrer à filtros especiais

- Exemplos
  - Adicionar um critério is null
  - Adicionar o desconhecido (is unknown) na consulta
  - Usar a função coalesce, que devolve o primeiro valor não nulo

Valores Nulos e comparações

Projetos com status indefinidos ou nulo (que implica indefinição)

Projeto		
idProj	**nome**	**status**
1	ABC	Ativo
2	Lucrei	Indefinido
3	Show	null

select *nome*
from *projeto p*
where *status = 'Indefinido' or*
status ***is null***

Resposta
nome
ABC
Show

Com is null, registros sem status são retornados

Valores Nulos e comparações

Projetos com tempo estimado igual ou inferior a 24 meses

Projeto		
idProj	**nome**	**status**
1	ABC	Ativo
2	Lucrei	Indefinido
3	Show	null

select *nome*
from *projeto p*
Where *status = 'Indefinido' or*
status = 'Indefinido' ***is unknown***

Resposta
nome
ABC
Show

Com is unknown, o filtro é satisfeito quando seu resultado for desconhecido

Valores Nulos e comparações

Projetos com tempo estimado igual ou inferior a 24 meses

Projeto		
idProj	**nome**	**status**
1	ABC	Ativo
2	Lucrei	Indefinido
3	Show	null

Simplifica expressões onde pode haver várias colunas que aceitem nulo
(ex. col1 * col2 > 10).

Nesse caso, usa-se apenas uma comparação is unknown em vez de várias comparações is null

select *nome*
from *projeto p*
Where *status = 'Indefinido' or*
status = 'Indefinido' ***is unknown***

Resposta
nome
ABC
Show

Com is unknown, o filtro é satisfeito quando seu resultado for desconhecido

Valores Nulos e comparações

Projetos com tempo estimado igual ou inferior a 24 meses

Projeto		
idProj	**nome**	**status**
1	ABC	Ativo
2	Lucrei	Indefinido
3	Show	null

select *nome*
from *projeto p*
where **coalesce**(*status,'Indefinido'*) = *'Indefinido'*

Resposta
nome
ABC
Show

Com coalesce, os nulos são tratados como 'Indefinido', o que satisfaria o filtro

Valores Nulos e comparações

Projetos com tempo estimado igual ou inferior a 24 meses

Projeto		
idProj	**nome**	**status**
1	ABC	Ativo
2	Lucrei	Indefinido
3	Show	null

Cuidado. Índice sobres colunas indexadas não são usados caso o filtro use a coluna como atributo de alguma função.

Se existir um índice sobre status, talvez seja melhor evitar a função coalesce.

```
select    nome
from      projeto p
where     coalesce(status,'Indefinido') = 'Indefinido'
```

Resposta
nome
ABC
Show

Com coalesce, os nulos são tratados como 'Indefinido', o que satisfaria o filtro

Valores Nulos e Lógica Ternária

- Lógica trivalente (ternária) usando o valor lógico *unknown*:
  - OR:
    - (*unknown* **or** *true*) = *true*
    - (*unknown* **or** *false*) = *unknown*
    - (*unknown* **or** *unknown)* = *unknown*
  - AND:
    - *(true* **and** *unknown)* = *unknown*
    - *(false* **and** *unknown)* = *false*
    - *(unknown* **and** *unknown)* = *unknown*

Valores Nulos e Lógica Ternária

Listar os projetos com tempo estimado inferior a 3 anos e custo estimado diferente de 15.000

Projeto				
idProj	**nome**	**duracao**	**custo**	**idDepto**
1	ABC	2	12.000	1
2	Lucrei	3	30.000	1
3	Show	2	null	2

```
select   nome
from     projeto p
where    duracao < 3  or custo <> 15.000
```

Resposta
nome
ABC
Show

O SGBD retorna 'Show', pois basta que um dos critérios seja satisfeito.

No caso, o custo é menor do que 3

Valores Nulos e Lógica Ternária

Listar os projetos com tempo estimado inferior a 3 anos e custo estimado diferente de 15.000

Projeto				
idProj	**nome**	**duracao**	**custo**	**idDepto**
1	ABC	2	12.000	1
2	Lucrei	3	30.000	1
3	Show	2	null	2

```
select   nome
from     projeto p
where    duracao < 3  and custo <> 15.000
```

Resposta
nome
ABC

O SGBD não retorna 'Show', pois os dois critérios precisariam ser satisfeito, e em um deles o valor é desconhecido (custo)

Valores Nulos e Lógica Ternária

Listar os projetos com tempo estimado inferior a 3 anos e custo estimado diferente de 15.000

Projeto				
idProj	**nome**	**duracao**	**custo**	**idDepto**
1	ABC	2	12.000	1
2	Lucrei	3	30.000	1
3	Show	null	null	2

select *nome*
from *projeto p*
where *duracao* < *3* **and** *custo* <> 15.000

Resposta
nome
ABC

O SGBD não retorna 'Show', pois desconhece o seu custo e a sua duração

Exercício

Projetos com tempo inferior a 3 anos e custo diferente de 15.000

Projeto				
idProj	**nome**	**duracao**	**custo**	**idDepto**
1	ABC	2	12.000	1
2	Lucrei	3	30.000	1
3	Show	null	null	2

Caso se deseje retornar os registros incertos (Projeto Show) ?

```
select   nome
from     projeto p
where    ...
```

Resposta
nome
ABC
Show

Exercício

Projetos com tempo inferior a 3 anos e custo diferente de 15.000

Projeto				
idProj	**nome**	**duracao**	**custo**	**idDepto**
1	ABC	2	12.000	1
2	Lucrei	3	30.000	1
3	Show	null	null	2

Caso se deseje retornar os registros incertos (Projeto Show) ?

```
select    nome
from      projeto p
where     (coalesce(duracao,0) < 3  and
           coalesce(custo,0) <> 15.000)
```

Resposta
nome
ABC
Show

Com coalesce, o valor resultante da coluna foi ajustado.
Índices sobre as colunas filtradas não ficam disponíveis nesse caso

Exercício

Projetos com tempo inferior a 3 anos e custo diferente de 15.000

Projeto				
idProj	**nome**	**duracao**	**custo**	**idDepto**
1	ABC	2	12.000	1
2	Lucrei	3	30.000	1
3	Show	null	null	2

Caso se deseje retornar os registros incertos (Projeto Show) ?

```
select   nome
from     projeto p
where    ((duracao < 3  and  custo <> 15.000) or
          (duracao < 3  and  custo <> 15.000) is unknown)
```

Resposta
nome
ABC
Show

Com is unknown, foi necessário repetir toda a expressão.

Exercício

Projetos com tempo inferior a 3 anos e custo diferente de 15.000

Projeto				
idProj	**nome**	**duracao**	**custo**	**idDepto**
1	ABC	2	12.000	1
2	Lucrei	3	30.000	1
3	Show	null	null	2

Caso se deseje retornar os registros incertos (Projeto Show) ?

```
select    nome
from      projeto p
where (  select (duracao < 3 and custo <> 15000)
       ) is not false;
```

Resposta
nome
ABC
Show

Com uma subconsulta escalar, o retorno pode ser true, false ou unknown.

Isso evite a duplicação da expressão

Exercício

Projetos com tempo inferior a 3 anos e custo diferente de 15.000

Projeto				
idProj	**nome**	**duracao**	**custo**	**idDepto**
1	ABC	2	12.000	1
2	Lucrei	3	30.000	1
3	Show	null	null	2

Caso se deseje retornar os registros incertos (Projeto Show) ?

```
select   nome
from     projeto p
where    ((duracao < 3 or duracao is null)  and
           (custo <> 15.000 or custo is null) )
```

Resposta
nome
ABC
Show

Com is null, foi necessário acrescentar dois critérios extras.

Exercício

Projetos com tempo inferior a 3 anos e custo diferente de 15.000

Projeto				
idProj	**nome**	**duracao**	**custo**	**idDepto**
1	ABC	2	12.000	1
2	Lucrei	3	30.000	1
3	Show	null	null	2

Caso se deseje retornar os registros incertos (Projeto Show) ?

```
select   nome
from     projeto p
where    ((duracao < 3 or duracao is null)  and
          (custo <> 15.000 or custo is null) )
```

Resposta
nome
ABC
Show

Para algumas das soluções, é possível que índices não sejam devidamente explorados

Por isso é importante conhecer as alternativas e testá-las

Valores Nulos e Funções de agregação

- As seguintes funções de agregação ignoram valores nulos
  - AVG
  - SUM
  - MIN
  - MAX
  - COUNT(COL)

- **Cuidado:** Ao adicionar um placeholder a uma coluna que deveria ser nula, ela passa a ser usada por essas funções
  - Adulterando o resultado da consulta

Valores Nulos e Funções de agregação

Listar o custo médio de todos os projetos

Projeto				
idProj	**nome**	**duracao**	**custo**	**idDepto**
1	ABC	2	6.000	1
2	Lucrei	3	30.000	1
3	Show	1	null	2

select **avg** (*custo)*
from *projeto p*

Resposta
avg(custo)
18.000

Valores Nulos e Funções de agregação

Listar o custo médio de todos os projetos

Projeto				
idProj	**nome**	**duracao**	**custo**	**idDepto**
1	ABC	2	6.000	1
2	Lucrei	3	30.000	1
3	Show	1	0	2

select **avg** (*custo*)
from *projeto p*

Resposta
avg(custo)
12.000

Com custo artificial o resultado não reflete a realidade.

Sumário

- Organização física dos registros
- Uso de placeholders
- Semântica dos valores nulos em consultas
- Valores nulos em subconsultas
- Colunas opcionais e especialização
- Uso de divisão de tabela

Valores nulos em subconsultas

- Como vimos, o SGBD exclui da resposta os registros que possuem nulos em colunas filtradas
  - Ou seja, não retorna o que desconhece
- Mas e se a comparação ocorre em uma subconsulta?
- Quando a subconsulta devolve uma lista de valores para ser comparada
  - Se
    - a condição deve ser satisfeita para todos os registros (ALL)
    - e a subconsulta possui algum valor nulo
  - o resultado sempre será falso

Valores nulos em subconsultas

- O valor 50 é maior do que a lista de valores abaixo?

$$50 > \textbf{all}\ (1,7, 9, 15, 23, \text{null})$$

Valores nulos em subconsultas

- O valor 50 é maior do que a lista de valores abaixo?

  50 > **all** (1,7, 9, 15, 23, null) (false)

- Apesar de 50 ser maior do que todos os valores, não se sabe se é maior do que nulo

- Qualquer valor comparado com essa sublista retornaria o mesmo resultado

Valores nulos em subconsultas

- O valor 50 é diferente do que toda a lista de valores abaixo?

  50 <> **all** (1,7, 9, 15, 23, null)

Valores nulos em subconsultas

- O valor 50 é diferente do que toda a lista de valores abaixo?

  50 <> **all** (1,7, 9, 15, 23, null) (false)

- Apesar de 50 ser diferente do que todos os valores, não se sabe se é diferente do que nulo

- Obs.
  - (<> all) é o mesmo que (NOT IN)
  - Se expressarmos uma consulta com NOT IN, e a subconsulta devolver NULO, a resposta será sempre falsa

Valores nulos em subconsultas

Retornar projetos cujo custo seja superior a todas as comissoes de qualquer funcionário

Projeto				
idProj	**nome**	**duracao**	**custo**	**idDepto**
1	ABC	2	6.000	1
2	Lucrei	3	30.000	1
3	Show	1	10.000	2

Aloc		
idProj	**idFunc**	**comissão**
1	1	1.000
1	3	1.300
2	1	10.000
2	2	0

select nome
from *projeto p*
Where *custo > all (**select** comissao **from** aloc a)*

nome
Lucrei

Sem comissões nulas, o projeto é retornado

Valores nulos em subconsultas

Retornar projetos cujo custo seja superior a todas as comissoes de qualquer funcionário

Projeto				
idProj	**nome**	**duracao**	**custo**	**idDepto**
1	ABC	2	6.000	1
2	Lucrei	3	30.000	1
3	Show	1	10.000	2

Aloc		
idProj	**idFunc**	**comissão**
1	1	1.000
1	3	1.300
2	1	10.000
2	2	null

select nome
from *projeto p*
Where *custo > all (**select** comissao **from** aloc a)*

nome

Havendo pelo menos uma comissão nula, nenhum projeto é retornado

Valores nulos em subconsultas

Retornar projetos cujo custo seja diferente de todas as comissoes de funcionários alocados

Projeto				
idProj	**nome**	**duracao**	**custo**	**idDepto**
1	ABC	2	6.000	1
2	Lucrei	3	30.000	1
3	Show	1	10.000	2

Aloc		
idProj	**idFunc**	**comissão**
1	1	1.000
1	3	1.300
2	1	10.000
2	2	null

select nome
from *projeto p*
Where *custo NOT IN (**select** comissao **from** aloc a)*

nome

O mesmo acontece quando se usa NOT IN

Valores nulos em subconsultas

- Exemplos de formas para lidar com o problema dos nulos em subconsultas
  - Acrescentar o filtro IS NOT NULL
  - Usar a função COALESCE
  - Usar placeholders
  - Usar NOT EXISTS em vez de NOT IN

Valores nulos em subconsultas

Retornar projetos cujo custo seja diferente de todas as comissões de funcionários alocados

Projeto				
idProj	**nome**	**duracao**	**custo**	**idDepto**
1	ABC	2	6.000	1
2	Lucrei	3	30.000	1
3	Show	1	10.000	2

Aloc		
idProj	**idFunc**	**comissão**
1	1	1.000
1	3	1.300
2	1	10.000
2	2	null

select nome
from *projeto p*
***Where** custo **NOT IN** (**select** comissao **from** aloc a where comissao **is not null**)*

nome
ABC
Lucrei

Usando o filtro IS NOT NULL

Valores nulos em subconsultas

Retornar projetos cujo custo seja diferente de todas as comissões de funcionários alocados

Projeto				
idProj	**nome**	**duracao**	**custo**	**idDepto**
1	ABC	2	6.000	1
2	Lucrei	3	30.000	1
3	Show	1	10.000	2

Aloc		
idProj	**idFunc**	**comissão**
1	1	1.000
1	3	1.300
2	1	10.000
2	2	null

select nome
from *projeto p*
***Where** custo **NOT IN** (**select coalesce**(comissao,-1) **from** aloc a)*

nome
ABC
Lucrei

Usando a função coalesce

Valores nulos em subconsultas

Retornar projetos cujo custo seja diferente de todas as comissões de funcionários alocados

Projeto				
idProj	**nome**	**duracao**	**custo**	**idDepto**
1	ABC	2	6.000	1
2	Lucrei	3	30.000	1
3	Show	1	10.000	2

Aloc		
idProj	**idFunc**	**comissão**
1	1	1.000
1	3	1.300
2	1	10.000
2	2	-1

select nome
from *projeto p*
***Where** custo **NOT IN** (**select** comissao **from** aloc a)*

nome
ABC
Lucrei

Usando placeholder

Valores nulos em subconsultas

Retornar projetos cujo custo seja diferente de todas as comissões de funcionários alocados

Projeto				
idProj	**nome**	**duracao**	**custo**	**idDepto**
1	ABC	2	6.000	1
2	Lucrei	3	30.000	1
3	Show	1	10.000	2

Aloc		
idProj	**idFunc**	**comissão**
1	1	1.000
1	3	1.300
2	1	10.000
2	2	null

```
select   nome
from     projeto p
Where NOT EXISTS (select 1 from aloc a WHERE p.custo = a.comissao)
```

nome
ABC
Lucrei

Usando NOT EXISTS

Valores nulos em subconsultas

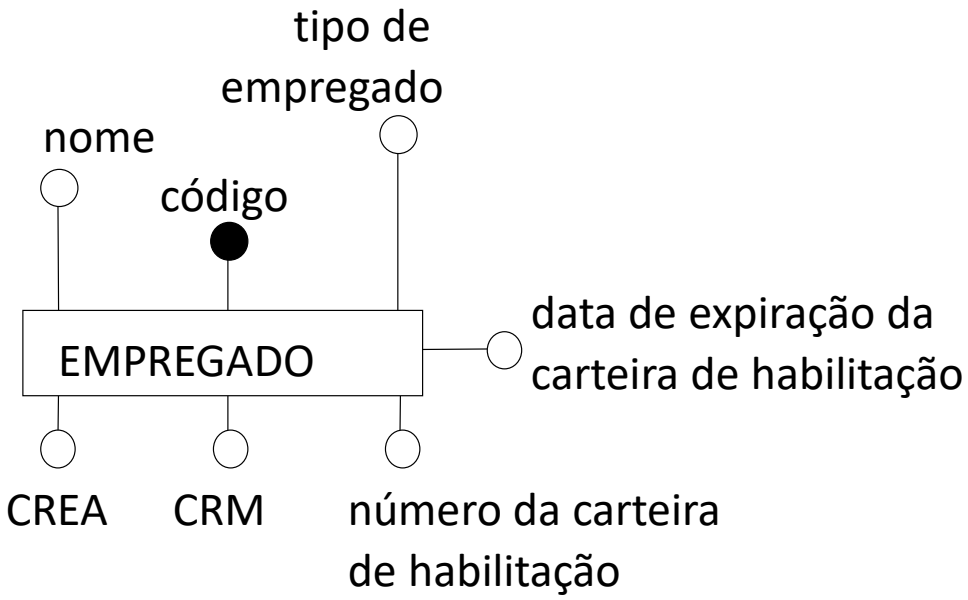
- NOT IN é **semanticamente** diferente de NOT EXISTS
- O NOT EXISTS verifica se a subconsulta devolve algum resultado
- O NOT IN realiza comparações contra uma lista de valores calculada pela subconsulta
  - Caso essa lista contenha algum valor nulo
    - O NOT IN retornará um resultado vazio

Sumário

- Organização física dos registros
- Uso de placeholders
- Semântica dos valores nulos em consultas
- Valores nulos em subconsultas
- Colunas opcionais e especialização
- Uso de divisão de tabela

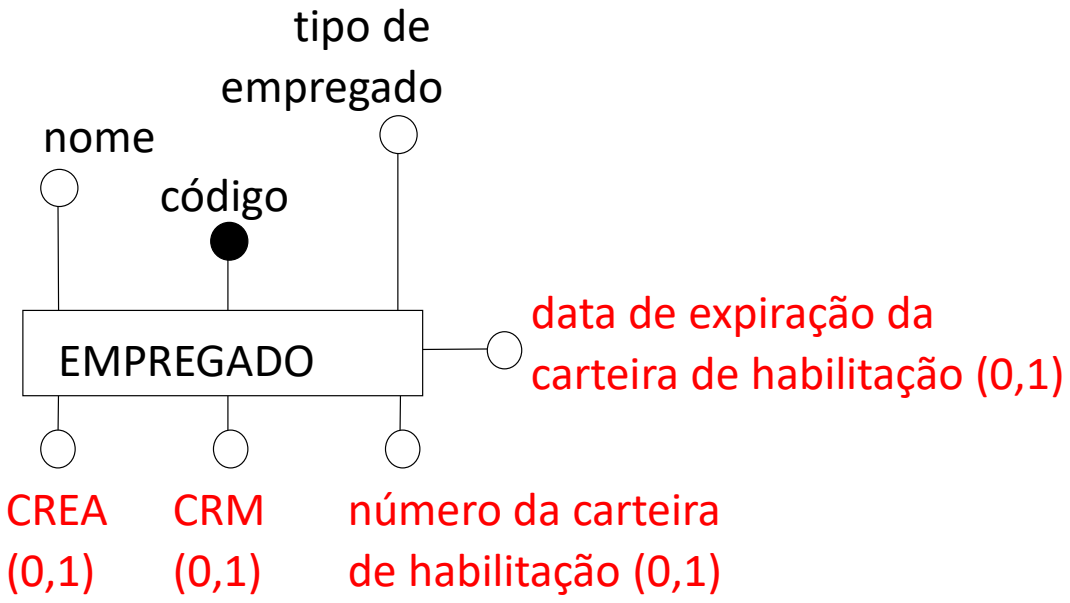
Colunas opcionais e especialização

- Quais colunas você definiria como nulas?
  - Considere que as opções a seguir são algumas das possíveis ocupações de um empregado: engenheiro, médico, motorista.



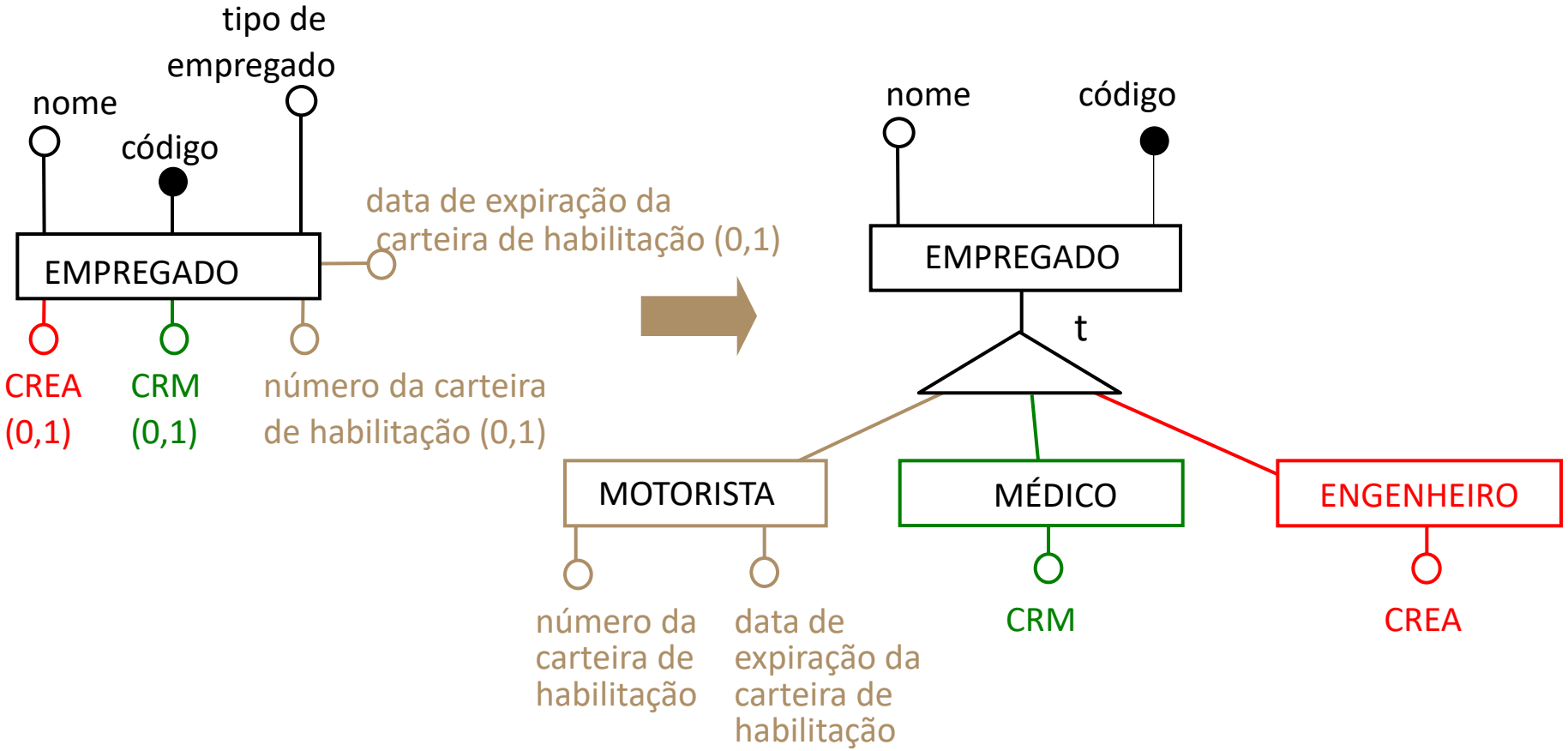
Colunas opcionais e especialização

- Quais colunas você definiria como nulas?
  - Considere que as opções a seguir são algumas das possíveis ocupações de um empregado: engenheiro, médico, motorista.



Colunas opcionais e especialização

- Em alguns casos, colunas nulas mascaram a modelagem de entidades especializadas



Colunas opcionais e especialização

- O que fazer quando o modelo possui essas entidades especializadas

- Duas estratégias básicas
  - Criar uma única tabela
  - Criar uma tabela por entidade

Colunas opcionais e especialização

- Uma única tabela

  Empregado (cod, nome, tipo, numCNH, expCNH, crm, crea)

- Uma tabela por entidade

  Empregado (cod, nome)

  Motorista (cod, numCNH, expCNH)

  cod referencia empregado(cod)

  Medico (cod, crm)

  cod referencia empregado(cod)

  Engenheiro (cod, crea)

  cod referencia empregado(cod)

Colunas opcionais e especialização

- Uso de uma tabela por entidade
- Prós
  - Tabelas com menos registros
  - Consultas eficientes por atributos específicos
  - Pode eliminar colunas nulas
- Contras
  - Consultas por todas colunas envolve junção
  - Mais tabelas para manter
    - Mais índices
  - Atualizações devem repercutir em mais do que uma tabela

Sumário

- Organização física dos registros
- Uso de placeholders
- Semântica dos valores nulos em consultas
- Valores nulos em subconsultas
- Colunas opcionais e especialização
- Uso de divisão de tabela

Uso de divisão de tabela

- A tabela **projeto** possui colunas opcionais
- Percebeu-se que as colunas **dataIni** e **Status** só recebem valor para projetos que já estejam em execução

Projeto		
*numProj	int	NOT NULL
descProj	char(30)	NOT NULL
dataIni	int	NULL
Status	char(10)	NULL

Uso de divisão de tabela

- A tabela **projeto** possui colunas opcionais
- Percebeu-se que as colunas **dataIni** e **Status** só recebem valor para projetos que já estejam em execução
  - Desse modo, decidiu-se dividir a tabela em duas
  - Com isso, foram eliminadas as colunas opcionais

Projeto		
*numProj	int	NOT NULL
descProj	char(30)	NOT NULL

1 1

ExecucaoProjeto		
*numProj	int	NOT NULL
dataIni	int	NOT NULL
Status	char(10)	NOT NULL

Uso de divisão de tabela

- Vantagens da aplicação dessa divisão de tabela
  - Tabelas ficam menores
    - A busca pode ser mais eficiente em consultas envolvendo uma dessas tabelas
  - Tabelas passam a ter registros de tamanho fixo
    - Com isso, a busca por offset pode ser usada

Projeto		
*numProj	int	NOT NULL
descProj	char(30)	NOT NULL

1 1

ExecucaoProjeto		
*numProj	int	NOT NULL
dataIni	int	NOT NULL
Status	char(10)	NOT NULL

Uso de divisão de tabela

- Desvantagens da aplicação dessa divisão de tabela
  - A busca por colunas das duas tabelas requer uma junção
  - Custo de atualização dos índices da nova tabela
  - Custo de espaço dos índices da nova tabela

Projeto		
*numProj	int	NOT NULL
descProj	char(30)	NOT NULL

1 1

ExecucaoProjeto		
*numProj	int	NOT NULL
dataIni	int	NOT NULL
Status	char(10)	NOT NULL

Uso de divisão de tabela

- Bons critérios para decidir pela separação das colunas de uma tabela
  - A coluna é pouco acessada e ocupa muito espaço
    - Seu deslocamento diminuirá o tamanho da tabela original
  - As colunas a serem movidas são as únicas que aceitam nulos ou têm tamanho variável
    - Seu deslocamento deixará a tabela original com registros de tamanho fixo
- Alguns cuidados
  - Se colunas costumam ser acessadas juntas
    - Não convém separá-las
    - Isso implica em junções extras

Uso de divisão de tabela

- Alguns especialistas consideram a divisão de tabela para fins de melhora no desempenho um problema de micro-otimização
  - Ou seja, surte pouco efeito na prática
- De qualquer forma, é bom saber que essa técnica existe
  - Pode ser que essa micro-otimização venha a ser útil

Considerações gerais

- De modo geral
  - Usar colunas nulas
    - Somente para colunas que podem não ter conteúdo definido

- Além disso, considere os seguintes fatores
  - a coluna ocupa bastante espaço?
  - a semântica dos nulos é importante na execução de consultas?
    - Geralmente é
  - o registro já terá tamanho variável de qualquer forma?
    - Alguma outra coluna for nula
    - Alguma outra coluna tiver tamanho variável (varchar, text, ...)

Exercício

- Refatore o modelo de contas bancárias de modo a resolver a consulta abaixo sem junções e agrupamento.

```
SELECT numConta, aniversario, saldo, SUM(juros)
FROM conta NATURAL JOIN contapoupanca
                NATURAL JOIN rendimento
GROUP BY numConta
```

A solução passa pela inclusão de colunas opcionais

Modelo de contas bancárias

Conta (numConta, saldo, nomeCliente, emailCliente)

ContaCorrente (numConta, limite)

ContaPoupanca (numConta, aniversario)

Rendimento (numConta, data, juros)

Exercício

- Refatore o modelo de contas bancárias de modo a resolver a consulta abaixo sem junções e agrupamento.

```
SELECT numConta, aniversario, saldo, SUM(juros)
FROM conta NATURAL JOIN contapoupanca
                NATURAL JOIN rendimento
GROUP BY numConta
```

Modelo de contas bancárias

Conta (numConta, saldo, nomeCliente, emailCliente)

ContaCorrente (numConta, limite)

ContaPoupanca (numConta, aniversario)

Rendimento (numConta, data, juros)

Atividade Individual

- Na empresa ACME
  - funcionários podem ser líderes de um projeto.
  - Um funcionário pode liderar no máximo um projeto.

- No modelo abaixo, isso foi resolvido colocando uma chave estrangeira na tabela funcionário
  - No entanto, a coluna que possui essa chave estrangeira é opcional

Projeto (idProj, nome, dataIni, prazo)

Funcionário (idFunc, nome, cargo, salario, idProj)
idProj referencia Projeto

Atividade Individual

- Refatore o banco de modo a não haver mais colunas opcionais
- A solução deve garantir que cada funcionário possa liderar no máximo um projeto

Projeto (idProj, nome, dataIni, prazo)

Funcionário (idFunc, nome, cargo, salario, idProj)
    idProj referencia Projeto