TABELA HASHING

Conceito

- Como já vimos, em muitos casos, operações como inserção, remoção e consulta de dados são feitas em tempo proporcional ao tamanho da estrutura
- Através do uso de uma tabela hashing procura-se melhorar, em média, operações como estas.
- Os elementos a serem armazenados na tabela hashing possuem um valor-chave que é utilizado para calcular o endereço da tabela onde serão alocados.

Conceito

- Em uma tabela hashing existe uma função de espalhamento/dispersão ou simplesmente função de hashing que calcula o endereço (ou índice) onde um determinado dado será armazenado.

- A função de espalhamento tem o objetivo de reduzir o espaço de endereços utilizado para armazenamento dos elementos.

Conceito

- A maioria das funções de hashing assume que os elementos-chave são números naturais, sendo que é possível encontrar casos em que strings são os elementos-chave.

- Um dos métodos utilizados para criar as funções de hashing é o método da divisão.
  - Uma chave *x* é mapeada em um dos m endereços da tabela hashing calculando o resto da divisão de *x* por *m*
    - A função de hashing é dada por $h(x) = x \% m$.

Conceito

- Por exemplo, em uma tabela hashing com tamanho m = 8, a chave x = 100 seria armazenada no endereço 4.

- No entanto, ao aplicar qualquer função de hashing sobre um conjunto de chaves, duas ou mais chaves podem ser mapeadas para o mesmo endereço.
  - Esta situação e chamada de colisão.

Conceito

- Segundo Cormen (2002) e Szwarcfiter (1994), uma função de hashing deveria satisfazer as seguintes condições:
  - Gerar um número pequeno de colisões;
  - Ser facilmente computável ; e
  - Ser uniforme.

Conceito

- Como nem sempre é possível gerar um número pequeno de colisões, tenta-se, ao menos, minimizá-las, utilizando métodos de tratamento de colisão.
- A colisão em uma tabela hashing pode ser tratada utilizando o endereçamento aberto
  - Colisões são tratadas dentro da própria tabela.
- Outra alternativa para o tratamento de colisões é o encadeamento
  - Cria-se para cada endereço da tabela uma lista encadeada das chaves que são mapeadas nele.

Tabela Hashing implementada com endereçamento aberto

- Neste tipo de implementação, a tabela hashing é um vetor com m posições.
- Todas as chaves armazenadas na própria tabela sem a necessidade de espaços extras ou ponteiros.
- Este método normalmente é aplicado quando o número de chaves a serem armazenadas na tabela hashing é reduzido e as posições vazias da tabela são utilizadas para o tratamento de colisões.

Tabela Hashing implementada com endereçamento aberto

- Quando uma chave x é endereçada na posição h(x) e essa já está ocupada, outras posições vazias na tabela são procuradas para armazenar x.
- Caso nenhuma seja encontrada, a tabela está totalmente preenchida e x não pode ser armazenada.
- Na sequência vamos visualizar um exemplo de funcionamento da tabela hashing com *tentativa linear*.

Tabela Hashing implementada com endereçamento aberto

- Na tentativa linear, quando uma chave x deve ser inserida e ocorre uma colisão, a seguinte função é utilizada para obter um novo endereço:
  - *h´(x) = (h(x)+i) mod m*, para *1 <= i <= m-1*, sendo que *h(x) = x mod m.*

- O objetivo é armazenar a chave no endereço consecutivo h(x) + 1, h(x)+2,..., até encontrar uma posição vazia.

Tabela Hashing implementada com endereçamento aberto

- A operação de remoção é delicada, não se pode remover de fato uma chave do endereço, pois haveria perda na sequência.

- Com isso, cada endereço da tabela é marcado da seguinte maneira:
  - (L) Livre, quando a posição ainda não foi utilizada
  - (O) Ocupado, ocupado quando uma chave está armazenada
  - (R) Removido, quando armazena uma chave que já foi removida
    - Uma nova chave poderá ocupar a posição marcada como removido.

Exemplo

	livre	chave
0	L	
1	L	
2	L	
3	L	
4	L	
5	L	
6	L	
7	L	

Tabela hashing vazia

Exemplo

	livre	chave
0	L	
1	L	
2	L	
3	L	
4	L	
5	L	
6	L	
7	L	

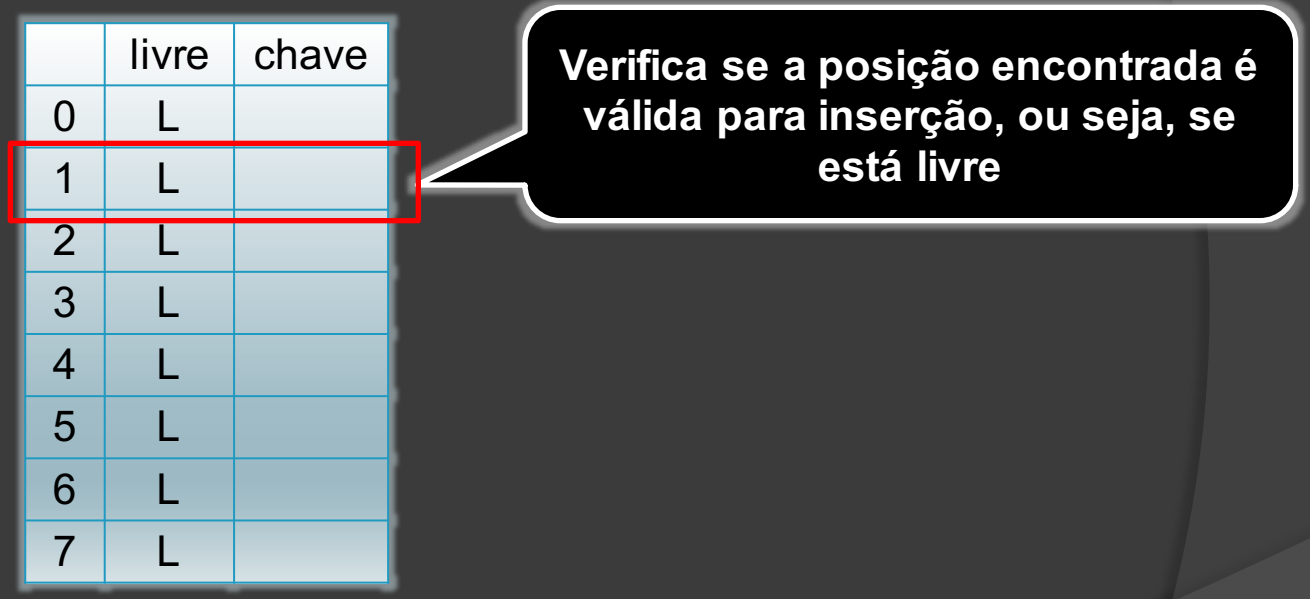
Deseja-se realizar a inserção do número 41 na tabela hashing

Exemplo

	livre	chave
0	L	
1	L	
2	L	
3	L	
4	L	
5	L	
6	L	
7	L	

Realiza-se o cálculo da posição
41 % 8 = 1

Exemplo


livre
chave
0 L
1 L
2 L
3 L
4 L
5 L
6 L
7 L
Verifica se a posição encontrada é válida para inserção, ou seja, se está livre

Exemplo

	livre	chave
0	L	
1	O	**41**
2	L	
3	L	
4	L	
5	L	
6	L	
7	L	

Como a posição está marcada como L (livre), insere o número 41 e muda o status na posição correspondente para O (ocupado)

Exemplo

	livre	chave
0	L	
1	O	**41**
2	L	
3	L	
4	L	
5	L	
6	L	
7	L	

Agora, vamos realizar a inserção do número 23 na tabela hashing

Exemplo

	livre	chave
0	L	
1	O	**41**
2	L	
3	L	
4	L	
5	L	
6	L	
7	L	

Realiza-se o cálculo da posição
23 % 8 = 7

Exemplo

	livre	chave
0	L	
1	O	**41**
2	L	
3	L	
4	L	
5	L	
6	L	
7	L	

Verifica se a posição encontrada é válida para inserção, ou seja, se está livre

Exemplo

	livre	chave
0	L	
1	O	**41**
2	L	
3	L	
4	L	
5	L	
6	L	
7	O	**23**

Como a posição está marcada como L (livre), insere o número 23 e muda o status na posição correspondente para O (ocupado)

Exemplo

	livre	chave
0	L	
1	O	**41**
2	L	
3	L	
4	L	
5	L	
6	L	
7	O	**23**

Agora, vamos realizar a inserção do número 25 na tabela hashing

Exemplo

	livre	chave
0	L	
1	O	**41**
2	L	
3	L	
4	L	
5	L	
6	L	
7	O	**23**

Realiza-se o cálculo da posição
25 % 8 = 1

Exemplo

	livre	chave
0	L	
1	O	**41**
2	L	
3	L	
4	L	
5	L	
6	L	
7	O	**23**

Verifica se a posição encontrada é válida para inserção, ou seja, se está livre

Exemplo

	livre	chave
0	L	
1	O	**41**
2	L	
3	L	
4	L	
5	L	
6	L	
7	O	**23**

Posição não está livre.
Tenta próxima posição
(25 + 1) % 8 = 2

Exemplo

	livre	chave
0	L	
1	O	**41**
2	O	**25**
3	L	
4	L	
5	L	
6	L	
7	O	**23**

Como a posição está marcada como L (livre), insere o elemento 25 e muda o status na posição correspondente para O (ocupado)

Exemplo

	livre	chave
0	L	
1	O	**41**
2	O	**25**
3	L	
4	L	
5	L	
6	L	
7	O	**23**

Agora, vamos realizar a inserção do número 39 na tabela hashing

Exemplo

	livre	chave
0	L	
1	O	**41**
2	O	**25**
3	L	
4	L	
5	L	
6	L	
7	O	**23**

Realiza-se o cálculo da posição
39 % 8 = 7

Exemplo

	livre	chave
0	L	
1	O	**41**
2	O	**25**
3	L	
4	L	
5	L	
6	L	
7	O	**23**

Verifica se a posição encontrada é válida para inserção, ou seja, se está livre

Exemplo

	livre	chave
0	L	
1	O	**41**
2	O	**25**
3	L	
4	L	
5	L	
6	L	
7	O	**23**

Posição não está livre, então tenta próxima posição...

Mas estamos na última posição, e agora?

Exemplo

	livre	chave
0	L	
1	O	**41**
2	O	**25**
3	L	
4	L	
5	L	
6	L	
7	O	**23**

Procuramos a próxima posição livre a partir do início
(39 + 1) % 8 = 0

Exemplo

	livre	chave
0	O	**39**
1	O	**41**
2	O	**25**
3	L	
4	L	
5	L	
6	L	
7	O	**23**

Como a posição está marcada como L (livre), insere o elemento 39 e muda o status na posição correspondente para O (ocupado)

Exemplo

	livre	chave
0	O	**39**
1	O	**41**
2	O	**25**
3	L	
4	L	
5	L	
6	L	
7	O	**23**

Agora, deseja-se realizar a remoção do número 25 na tabela hashing

Exemplo

	livre	chave
0	O	**39**
1	O	**41**
2	O	**25**
3	L	
4	L	
5	L	
6	L	
7	O	**23**

Realiza-se o cálculo da posição
25 % 8 = 1

Exemplo

	livre	chave
0	O	**39**
1	O	**41**
2	O	**25**
3	L	
4	L	
5	L	
6	L	
7	O	**23**

Verifica se a posição encontrada é válida para remoção, ou seja, se a posição está marcada como ocupada e o elemento da posição é igual ao que se deseja remover

Exemplo

	livre	chave
0	O	**39**
1	O	**41**
2	O	**25**
3	L	
4	L	
5	L	
6	L	
7	O	**23**

Posição não está livre, mas elementos são diferentes
41 != 25

Exemplo

	livre	chave
0	O	**39**
1	O	**41**
2	O	**25**
3	L	
4	L	
5	L	
6	L	
7	O	**23**

Tenta próxima posição
(25 + 1) % 8 = 2

Exemplo

	livre	chave
0	O	**39**
1	O	**41**
2	R	**25**
3	L	
4	L	
5	L	
6	L	
7	O	**23**

Como a posição está marcada como O (ocupado) e o elemento procurado é igual ao encontrado, remove o elemento 25 mudando o status na posição correspondente para R (removido)

Tabela Hashing implementada com lista encadeada

- Na implementação com lista, a tabela Hashing é um vetor com m posições.
- Cada posição possui um ponteiro para uma lista encadeada.
- Nesta lista ficam armazenados todos os elementos que possuem o mesmo endereço mapeado.

Exemplo

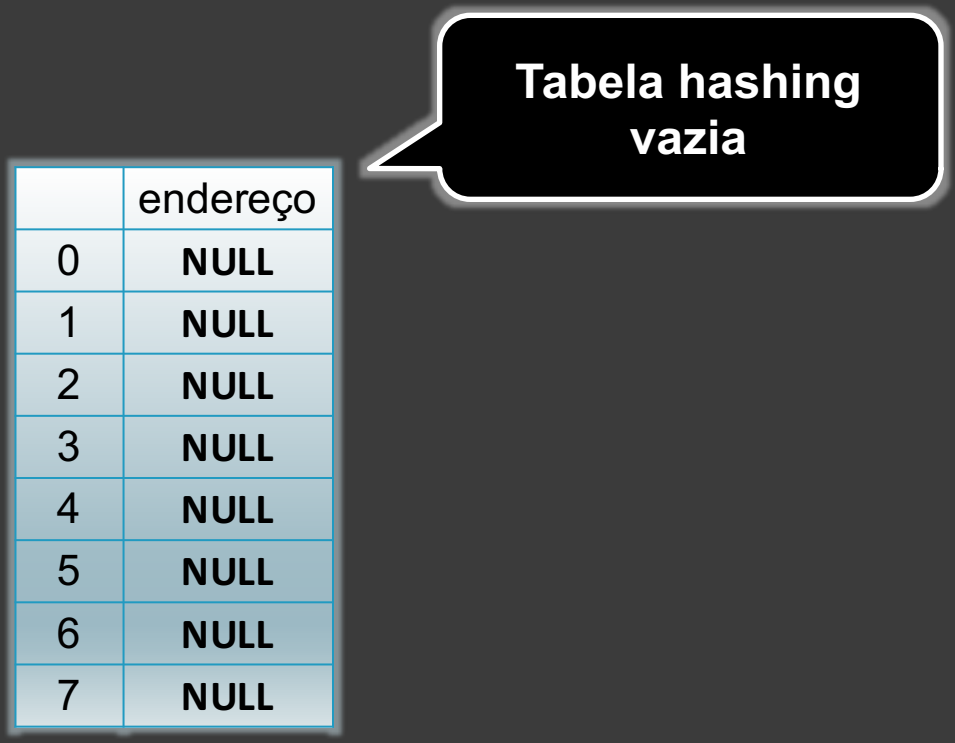


Tabela hashing vazia
endereço
0 NULL
1 NULL
2 NULL
3 NULL
4 NULL
5 NULL
6 NULL
7 NULL

Exemplo

	endereço
0	**NULL**
1	**NULL**
2	**NULL**
3	**NULL**
4	**NULL**
5	**NULL**
6	**NULL**
7	**NULL**

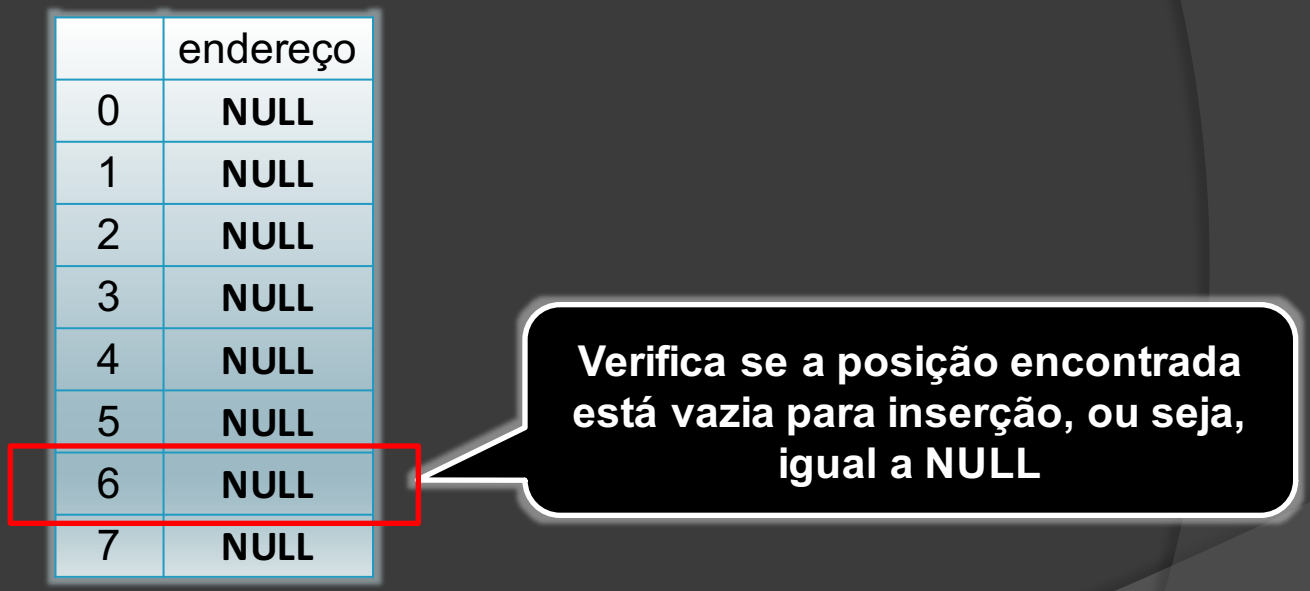
Deseja-se realizar a inserção do número 14 na tabela hashing

Exemplo

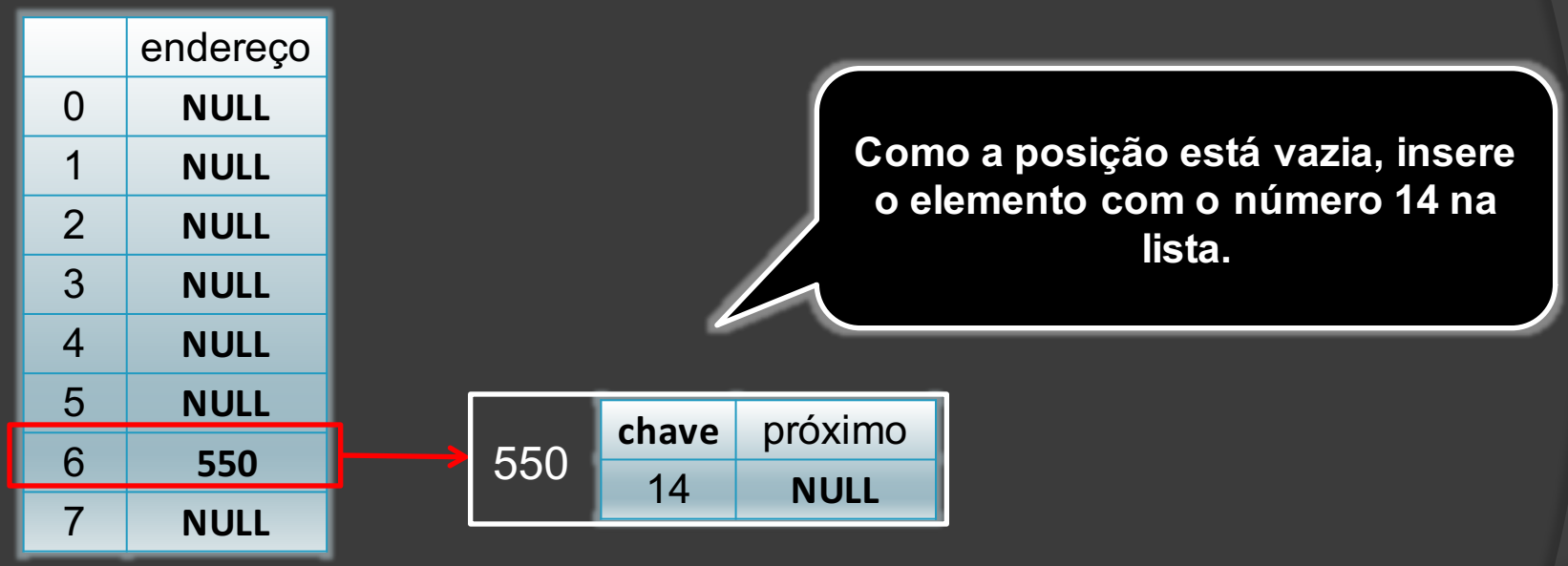
	endereço
0	**NULL**
1	**NULL**
2	**NULL**
3	**NULL**
4	**NULL**
5	**NULL**
6	**NULL**
7	**NULL**

Realiza-se o cálculo da posição
14 % 8 = 6

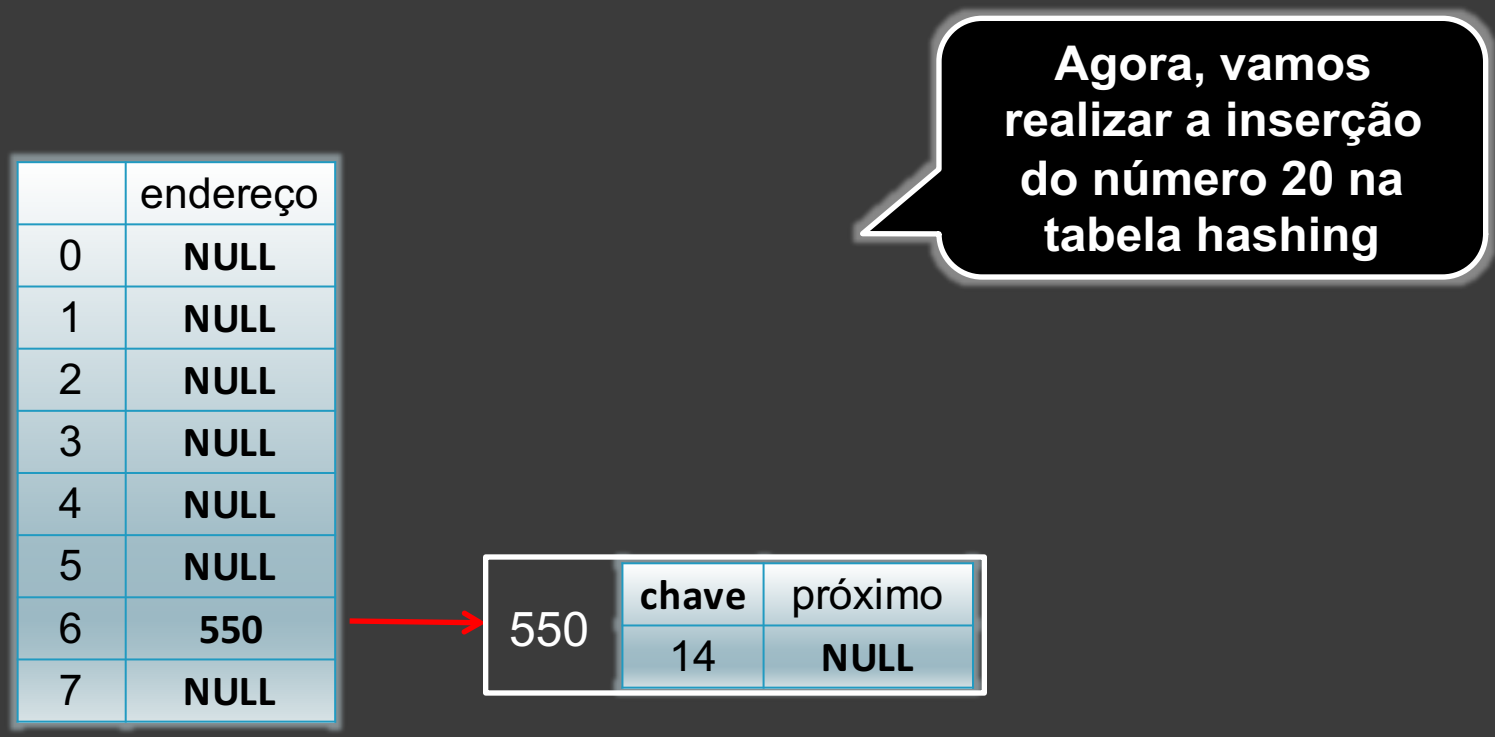
Exemplo


endereço
0 NULL
1 NULL
2 NULL
3 NULL
4 NULL
5 NULL
6 NULL
7 NULL
Verifica se a posição encontrada está vazia para inserção, ou seja, igual a NULL

Exemplo


endereço
0 NULL
1 NULL
2 NULL
3 NULL
4 NULL
5 NULL
6 550
7 NULL
550
chave
próximo
14
NULL
Como a posição está vazia, insere o elemento com o número 14 na lista.

Exemplo

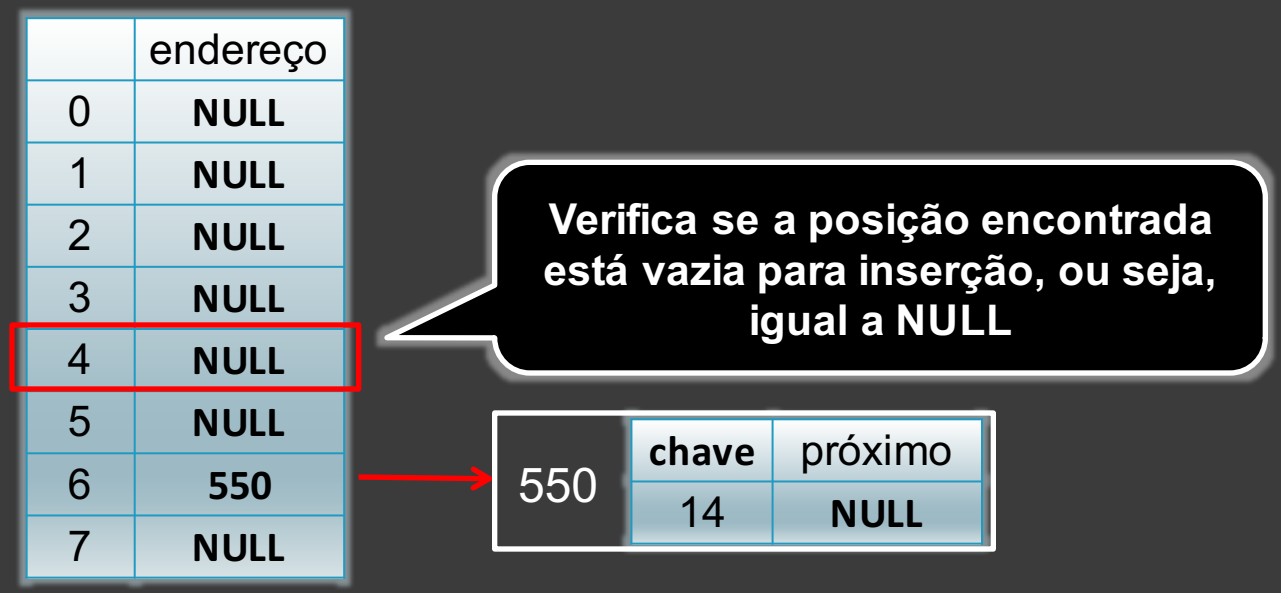

Agora, vamos realizar a inserção do número 20 na tabela hashing
endereço
0 NULL
1 NULL
2 NULL
3 NULL
4 NULL
5 NULL
6 550
7 NULL
550
chave
próximo
14
NULL

Exemplo

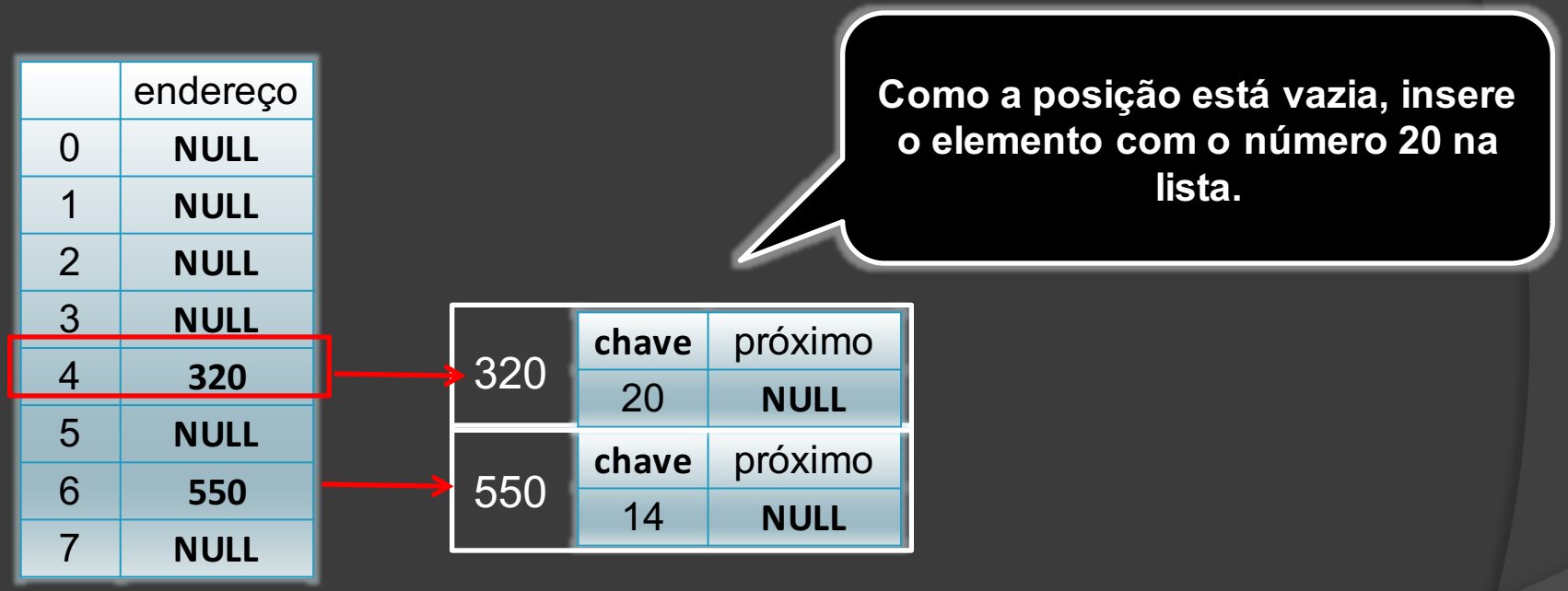
Realiza-se o cálculo da posição
20 % 8 = 4

	endereço
0	**NULL**
1	**NULL**
2	**NULL**
3	**NULL**
4	**NULL**
5	**NULL**
6	**550**
7	**NULL**

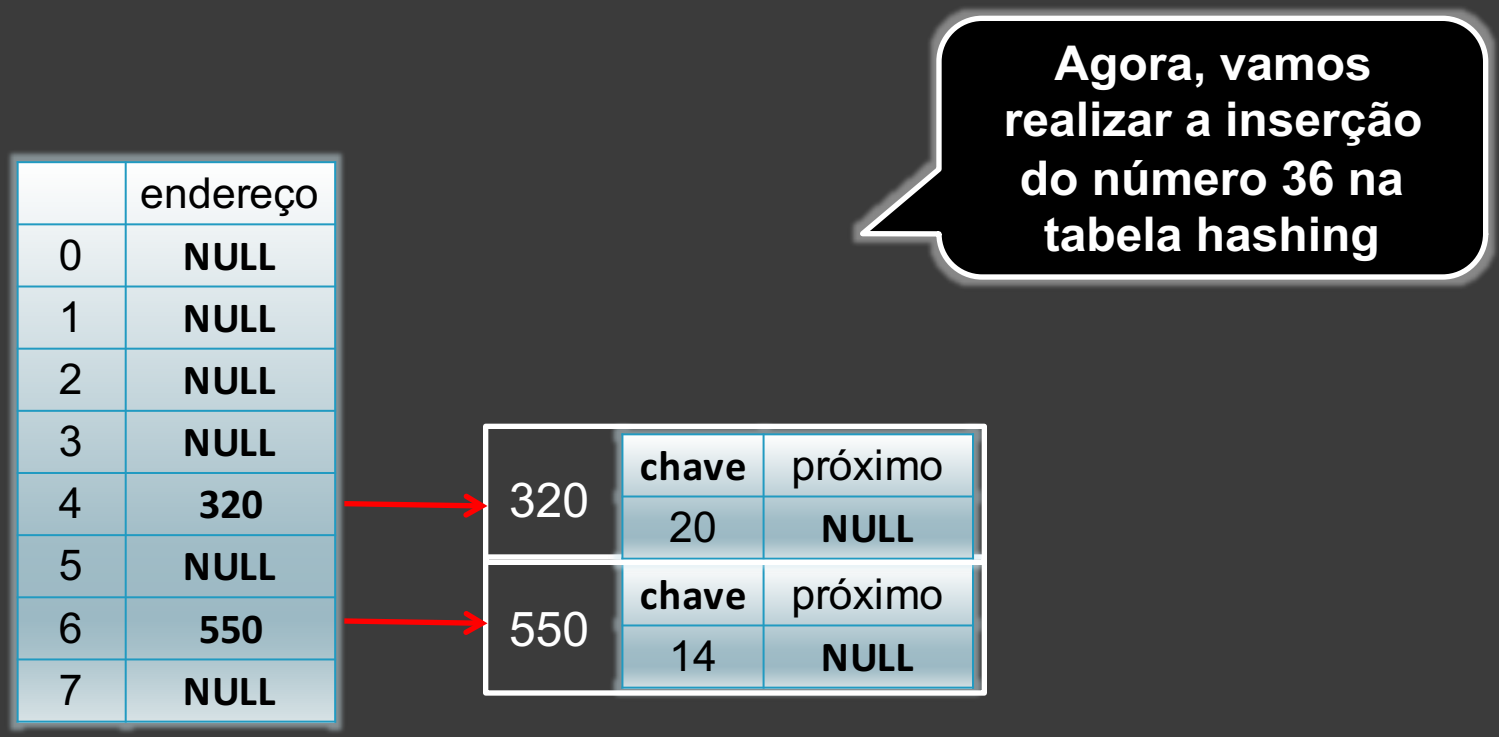
Exemplo


endereço
0 NULL
1 NULL
2 NULL
3 NULL
4 NULL
5 NULL
6 550
7 NULL
Verifica se a posição encontrada está vazia para inserção, ou seja, igual a NULL
550
chave
próximo
14
NULL

Exemplo


endereço
0 NULL
1 NULL
2 NULL
3 NULL
4 320
5 NULL
6 550
7 NULL
320
chave
próximo
20
NULL
550
chave
próximo
14
NULL
Como a posição está vazia, insere o elemento com o número 20 na lista.

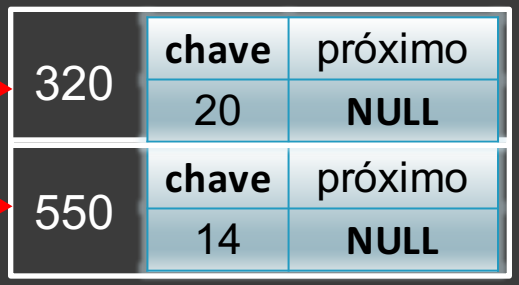
Exemplo


Agora, vamos realizar a inserção do número 36 na tabela hashing
endereço
0 NULL
1 NULL
2 NULL
3 NULL
4 320
5 NULL
6 550
7 NULL
320
chave
próximo
20
NULL
550
chave
próximo
14
NULL

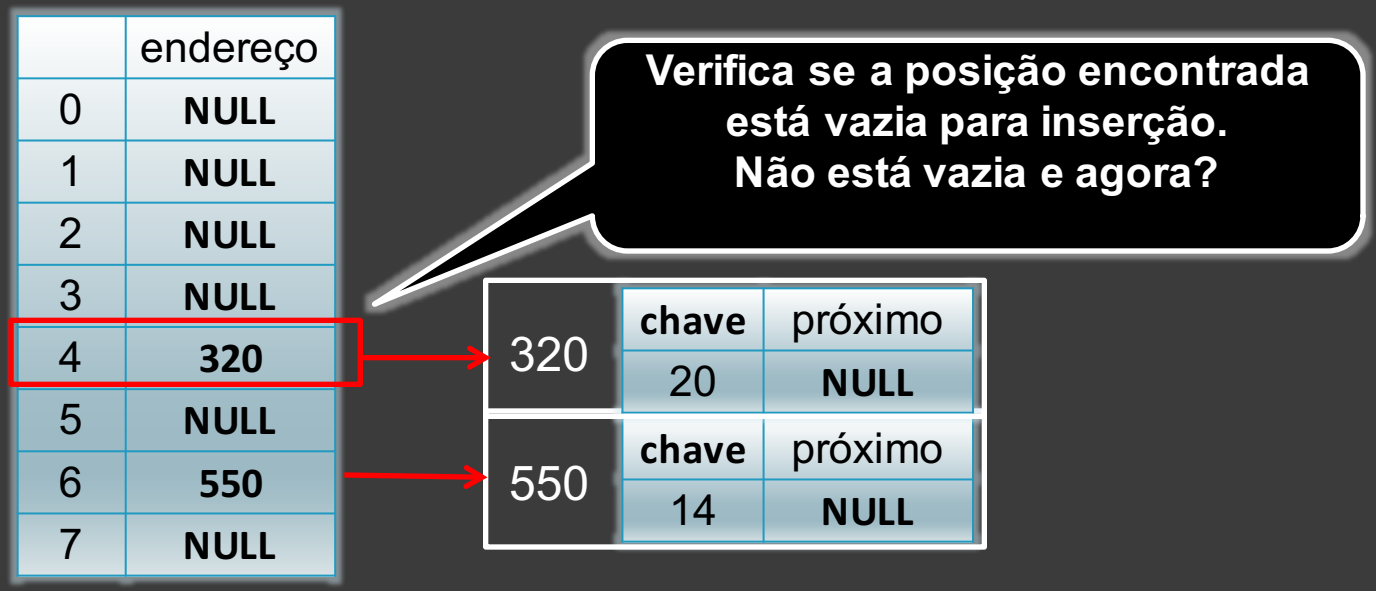
Exemplo

Realiza-se o cálculo da posição
36 % 8 = 4

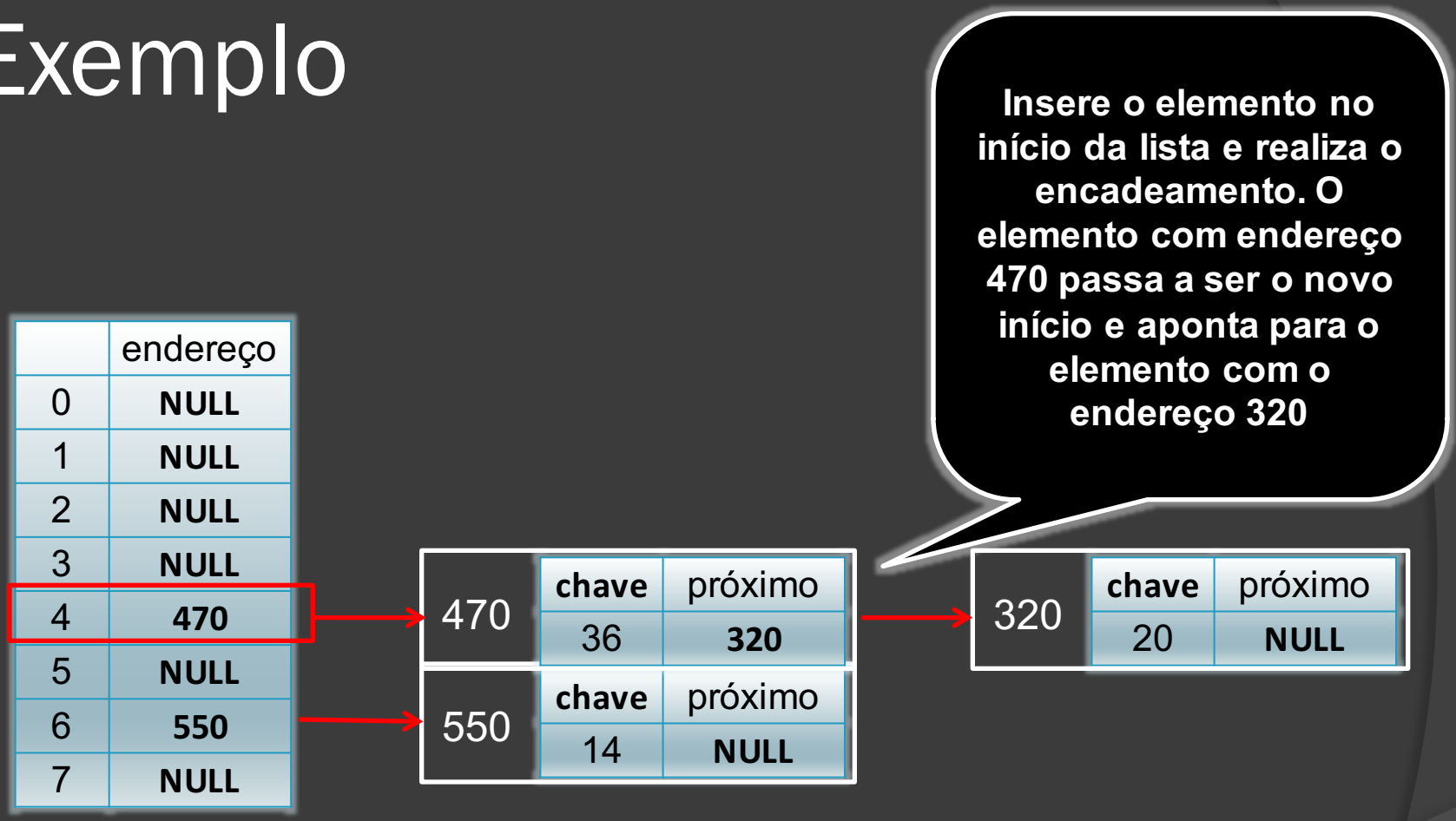
	endereço
0	**NULL**
1	**NULL**
2	**NULL**
3	**NULL**
4	**320**
5	**NULL**
6	**550**
7	**NULL**



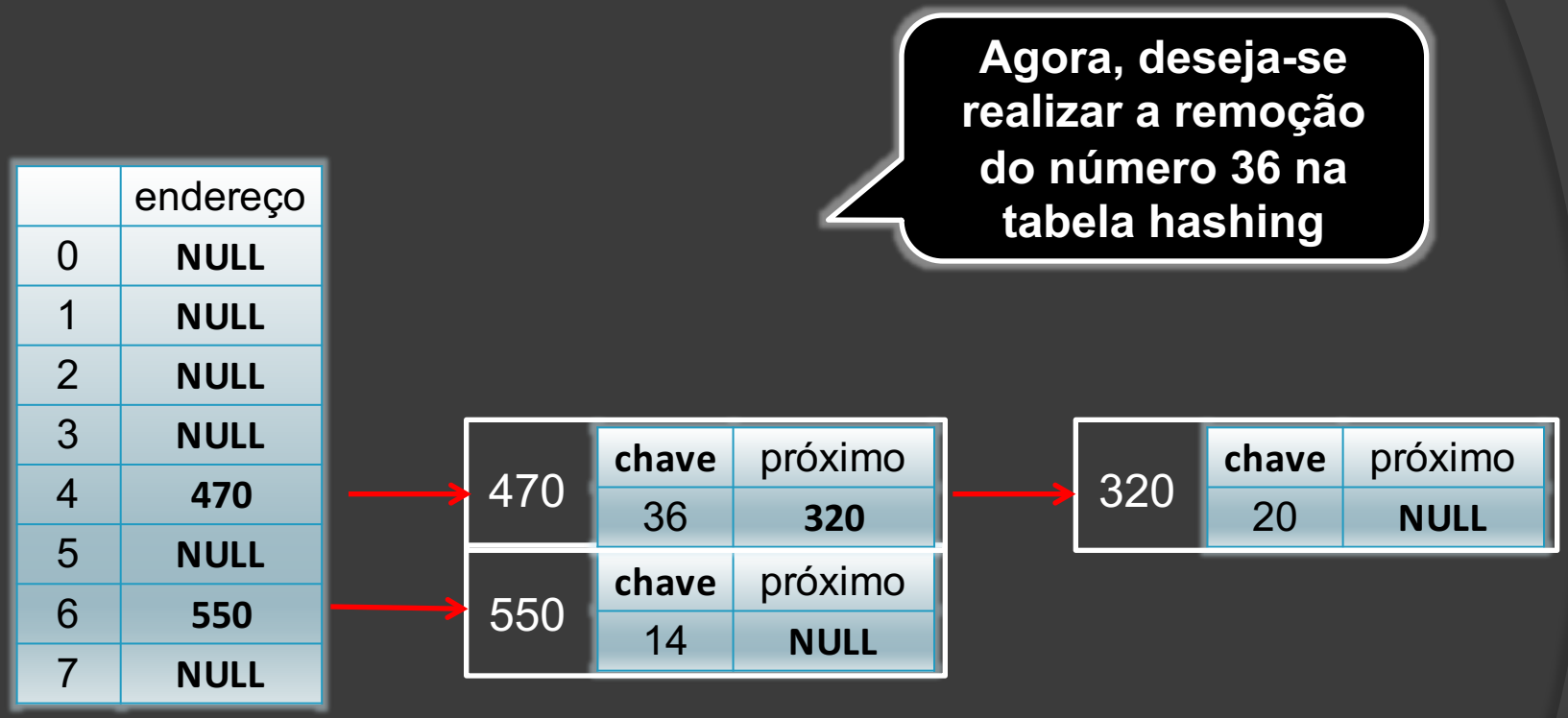
Exemplo


endereço
0 NULL
1 NULL
2 NULL
3 NULL
4 320
5 NULL
6 550
7 NULL
Verifica se a posição encontrada está vazia para inserção.
Não está vazia e agora?
320
chave
próximo
20
NULL
550
chave
próximo
14
NULL

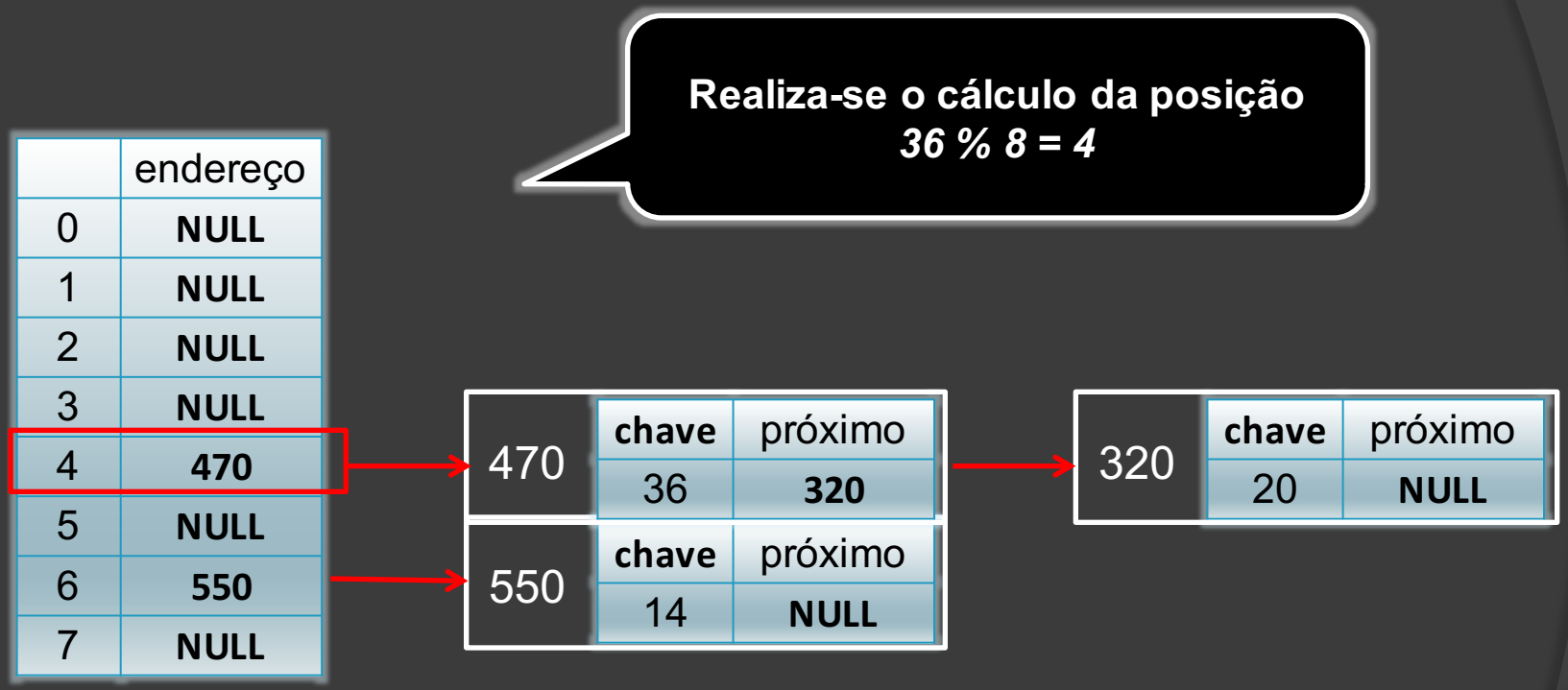
Exemplo


Insere o elemento no início da lista e realiza o encadeamento. O elemento com endereço 470 passa a ser o novo início e aponta para o elemento com o endereço 320
endereço
0 NULL
1 NULL
2 NULL
3 NULL
4 470
5 NULL
6 550
7 NULL
470
chave
36
próximo
320
550
chave
14
próximo
NULL
320
chave
20
próximo
NULL

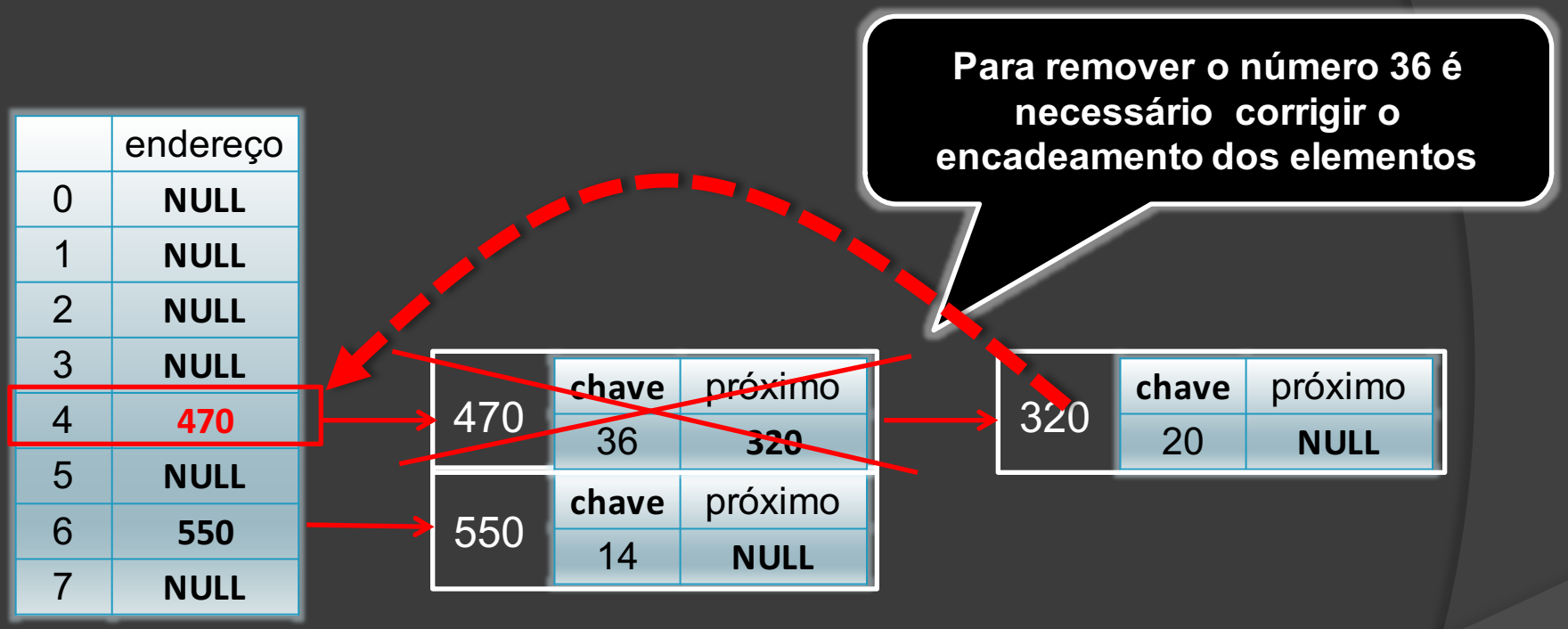
Exemplo


Agora, deseja-se realizar a remoção do número 36 na tabela hashing
endereço
0 NULL
1 NULL
2 NULL
3 NULL
4 470
5 NULL
6 550
7 NULL
470
chave 36
próximo 320
550
chave 14
próximo NULL
320
chave 20
próximo NULL

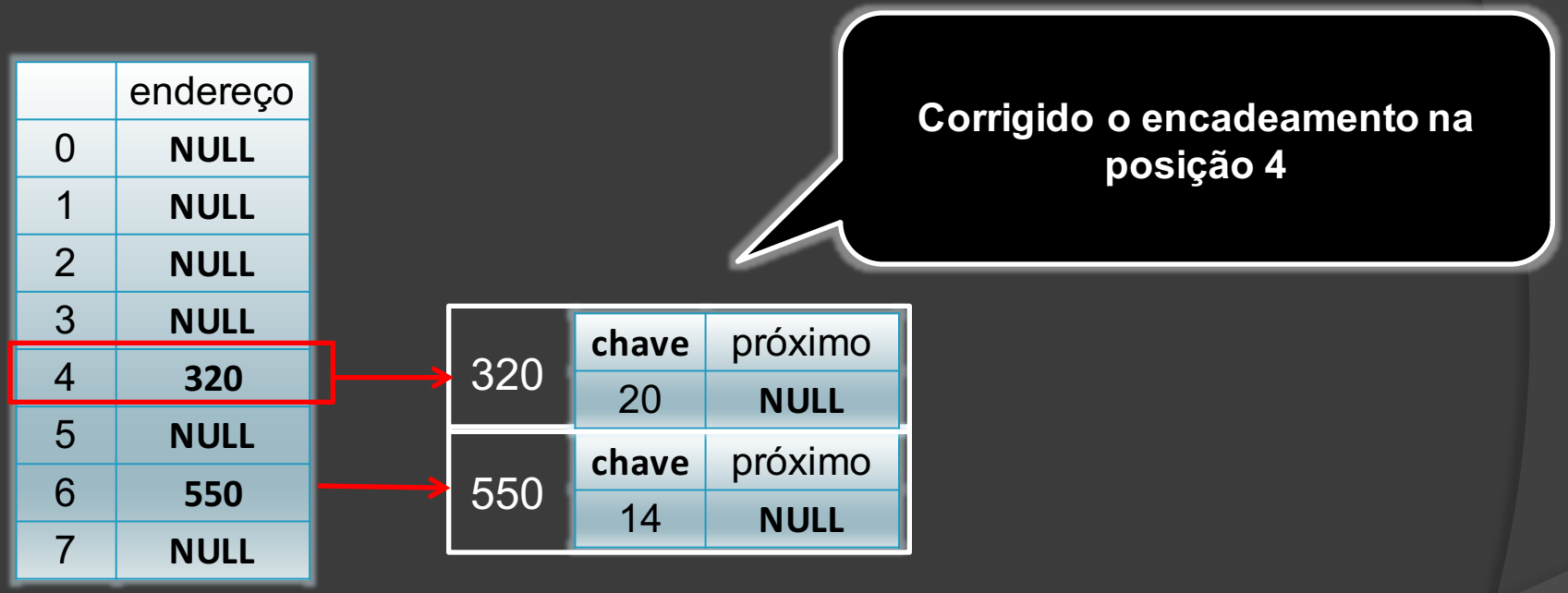
Exemplo


Realiza-se o cálculo da posição
36 % 8 = 4
endereço
0 NULL
1 NULL
2 NULL
3 NULL
4 470
5 NULL
6 550
7 NULL
470
chave
36
próximo
320
550
chave
14
próximo
NULL
320
chave
20
próximo
NULL

Exemplo


Para remover o número 36 é necessário corrigir o encadeamento dos elementos
endereço
0 NULL
1 NULL
2 NULL
3 NULL
4 470
5 NULL
6 550
7 NULL
470
chave 36
próximo 320
550
chave 14
próximo NULL
320
chave 20
próximo NULL

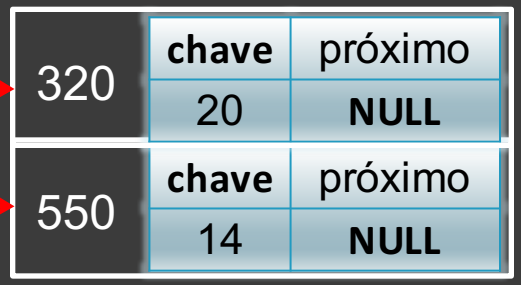
Exemplo


endereço
0 NULL
1 NULL
2 NULL
3 NULL
4 320
5 NULL
6 550
7 NULL
320
chave
próximo
20
NULL
550
chave
próximo
14
NULL
Corrigido o encadeamento na posição 4

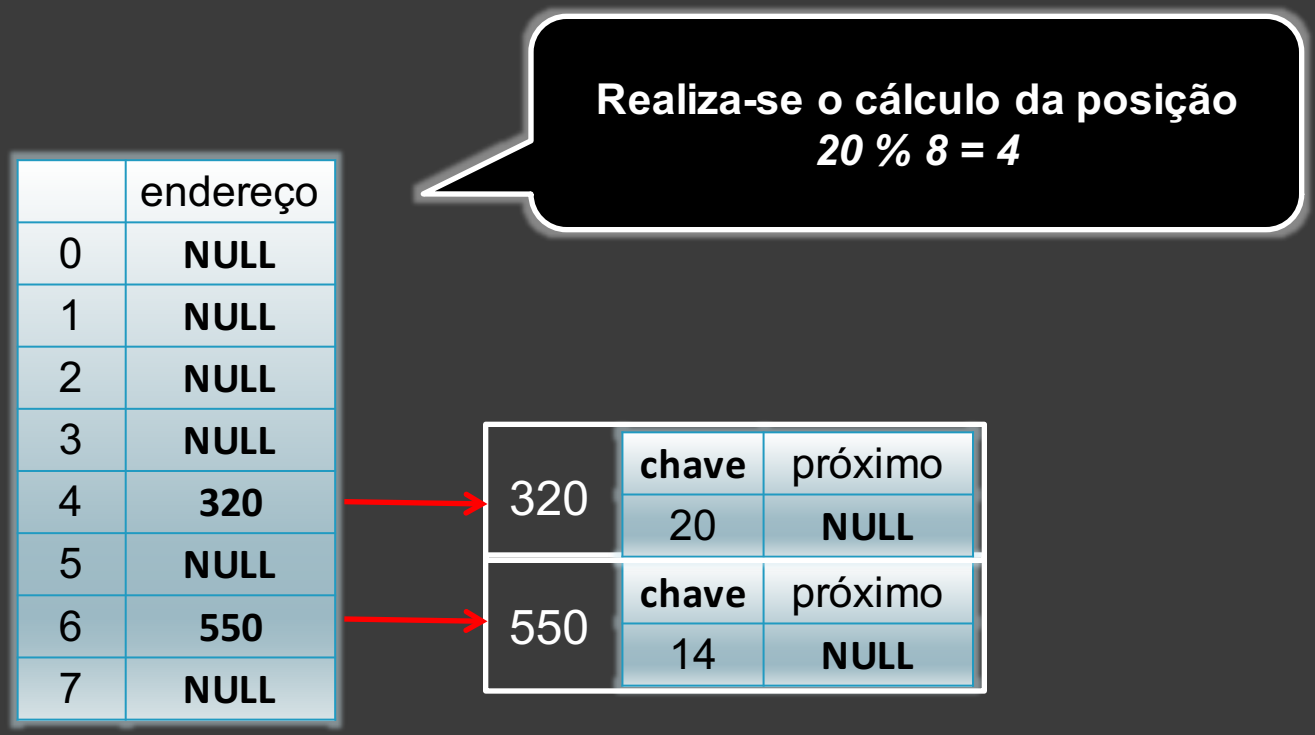
Exemplo

Para finalizar, vamos realizar a remoção do número 20 na tabela hashing

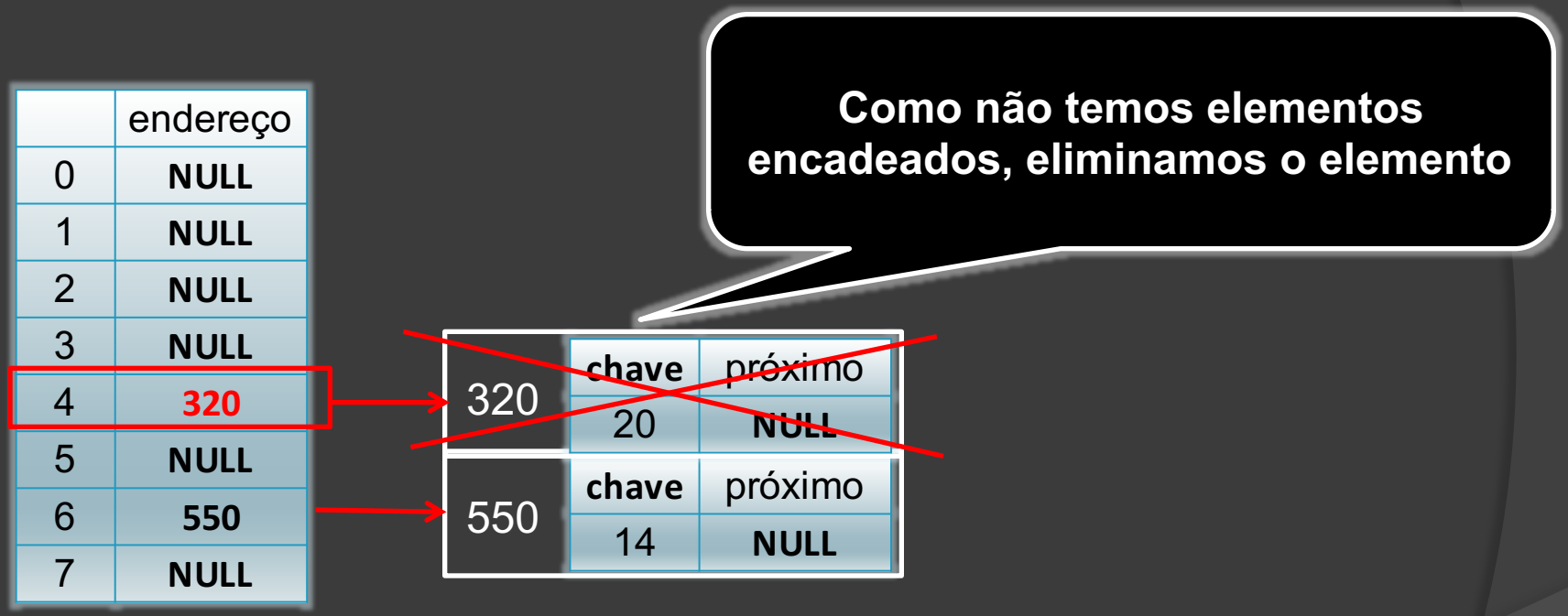
	endereço
0	**NULL**
1	**NULL**
2	**NULL**
3	**NULL**
4	**320**
5	**NULL**
6	**550**
7	**NULL**



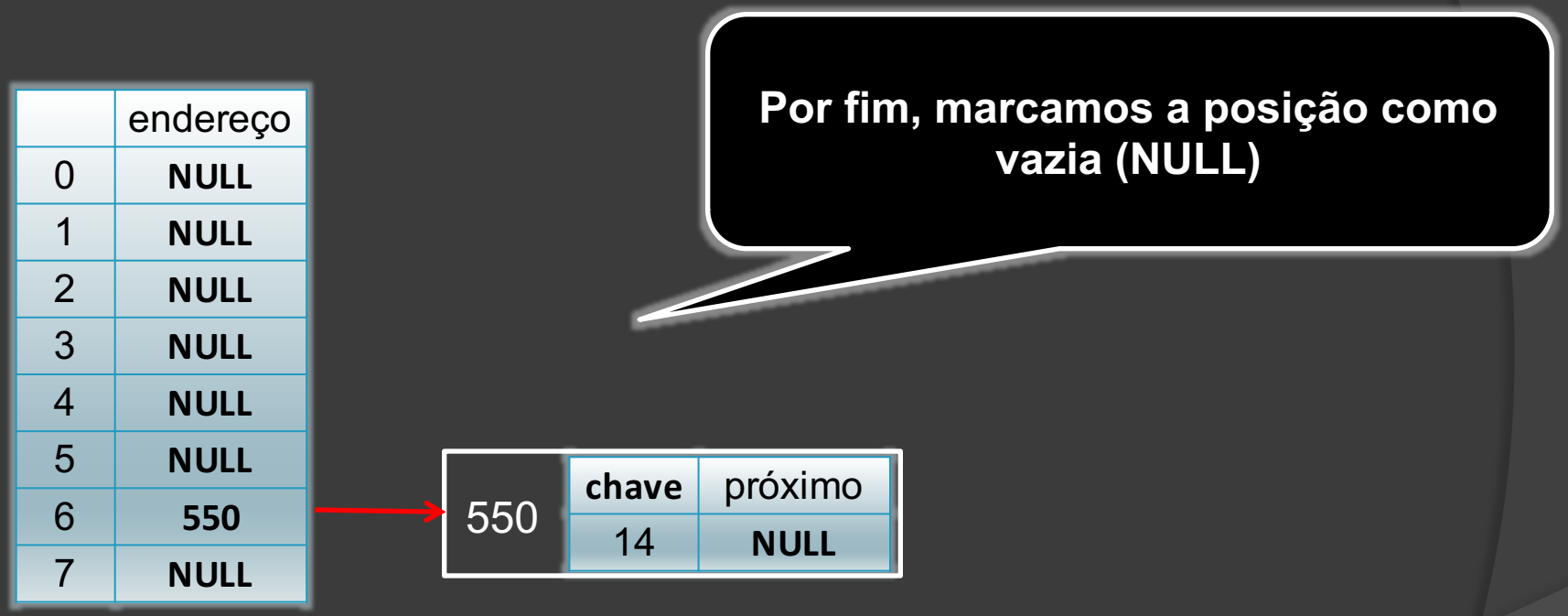
Exemplo


Realiza-se o cálculo da posição
20 % 8 = 4
endereço
0 NULL
1 NULL
2 NULL
3 NULL
4 320
5 NULL
6 550
7 NULL
320
chave
próximo
20
NULL
550
chave
próximo
14
NULL

Exemplo


Como não temos elementos encadeados, eliminamos o elemento
endereço
0 NULL
1 NULL
2 NULL
3 NULL
4 320
5 NULL
6 550
7 NULL
320
chave
20
próximo
NULL
550
chave
14
próximo
NULL

Exemplo


Por fim, marcamos a posição como vazia (NULL)
endereço
0 NULL
1 NULL
2 NULL
3 NULL
4 NULL
5 NULL
6 550
7 NULL
550
chave
próximo
14
NULL