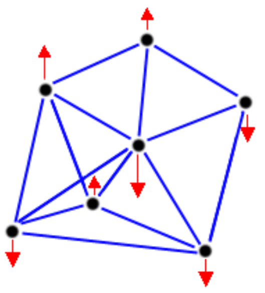
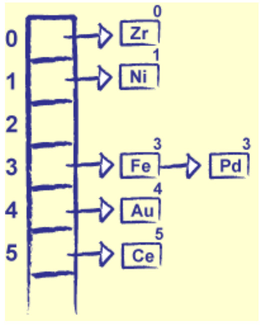


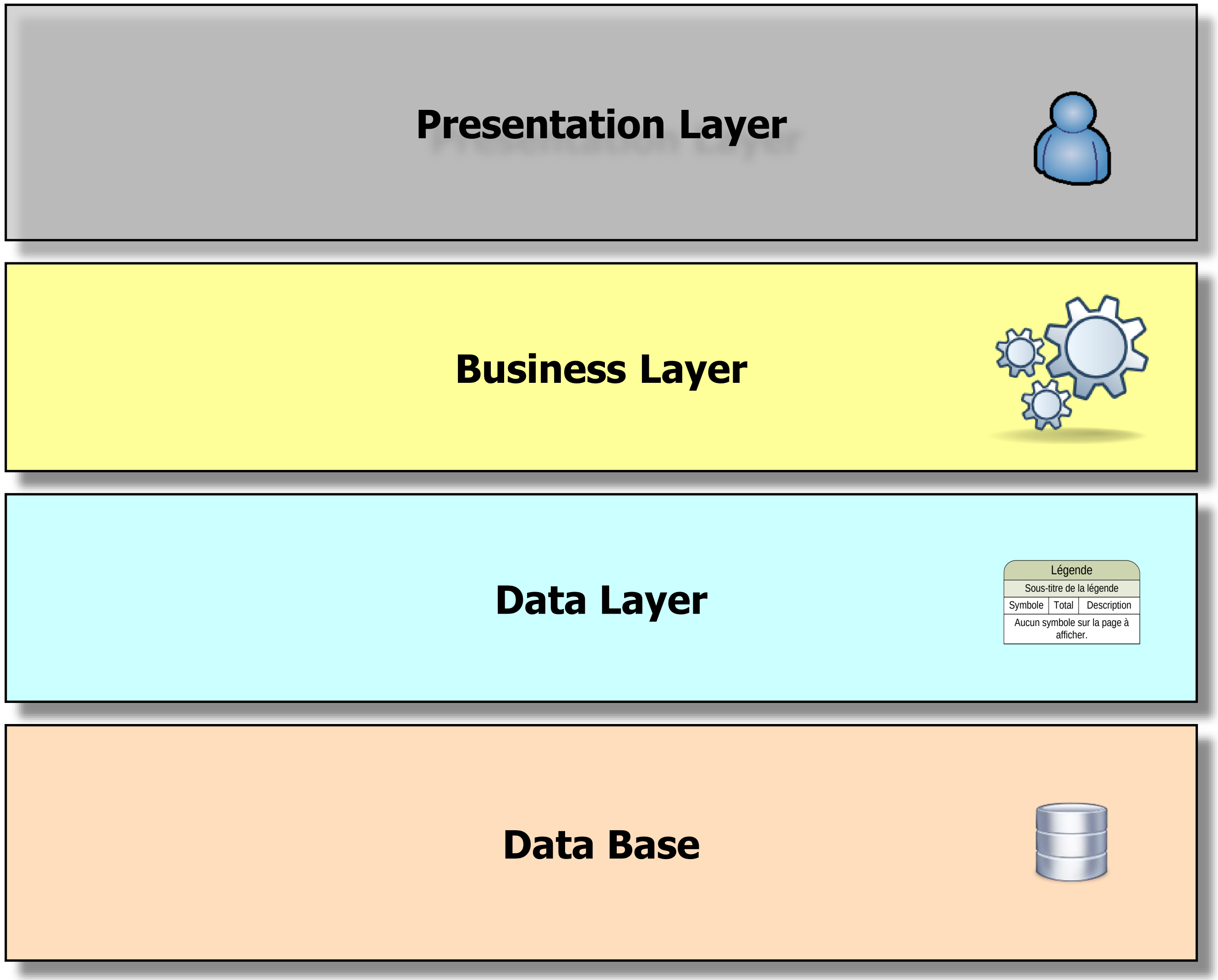
Conception Avancée de Bases de Données

Hash Table
Linear Probing
exemple

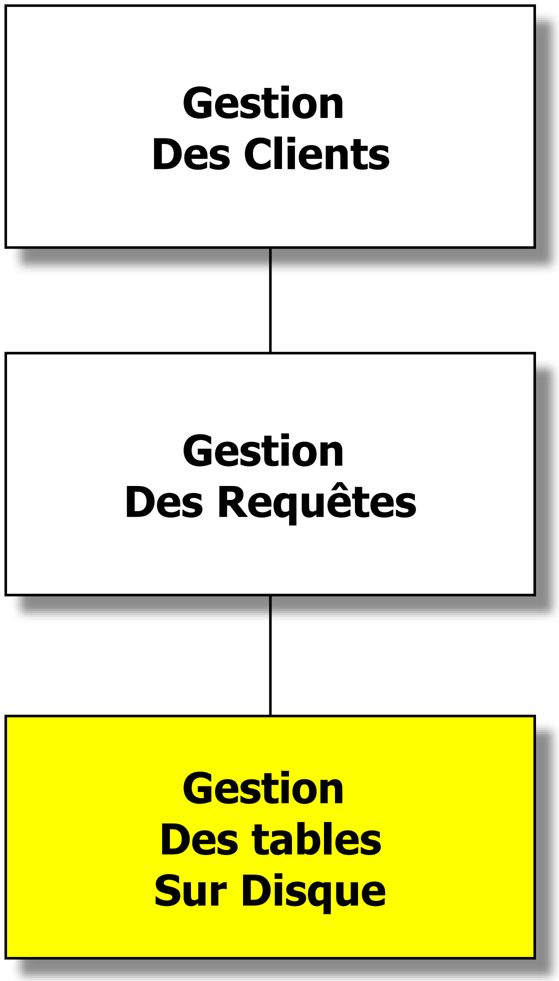


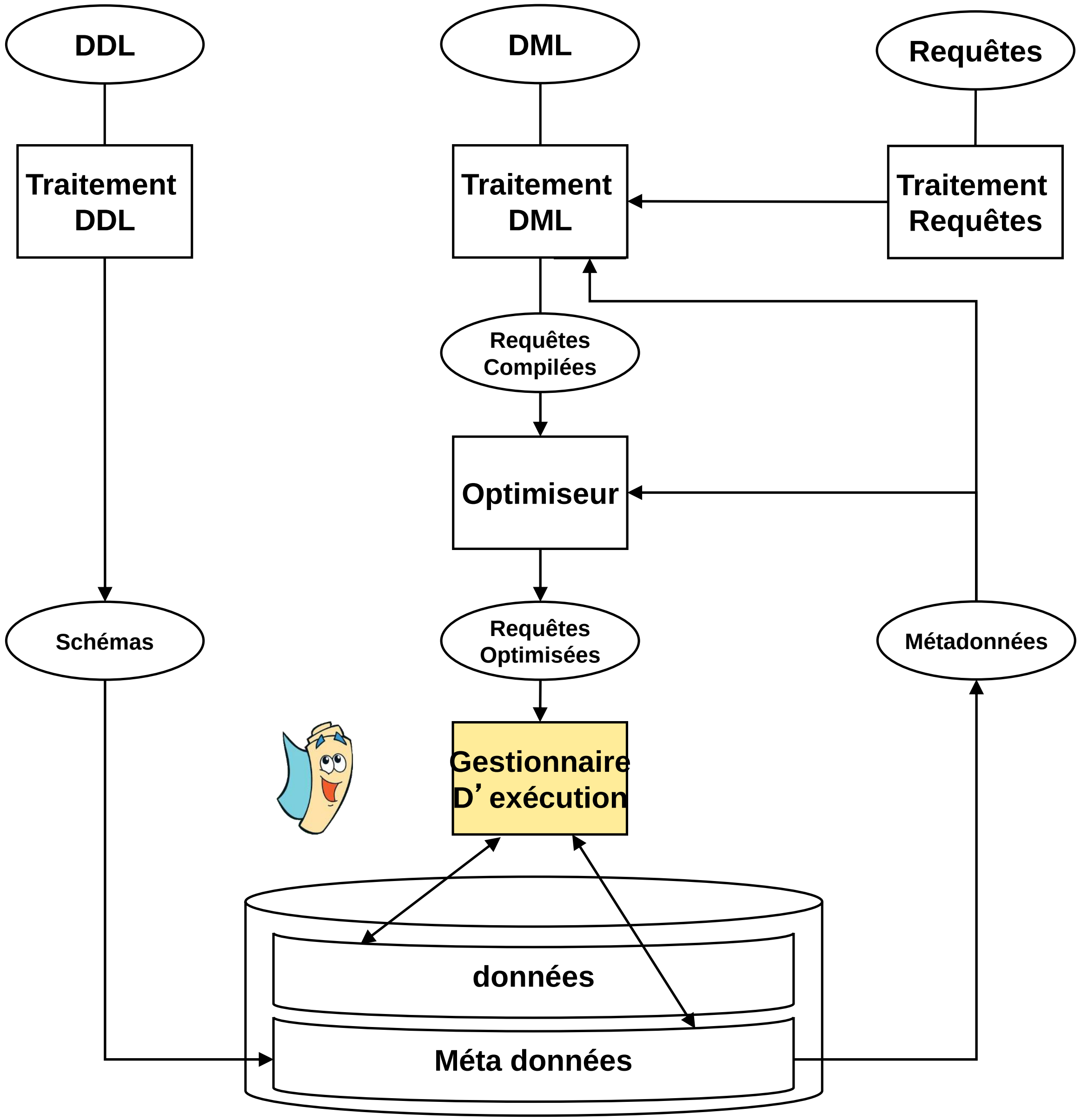
Traduction en cours

Layered Architecture



Big Picture

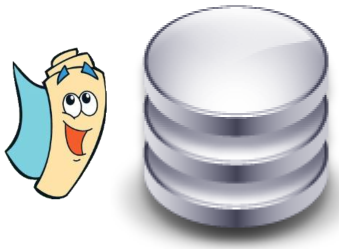

Gestion Des Clients
Gestion Des Requêtes
Gestion Des tables Sur Disque

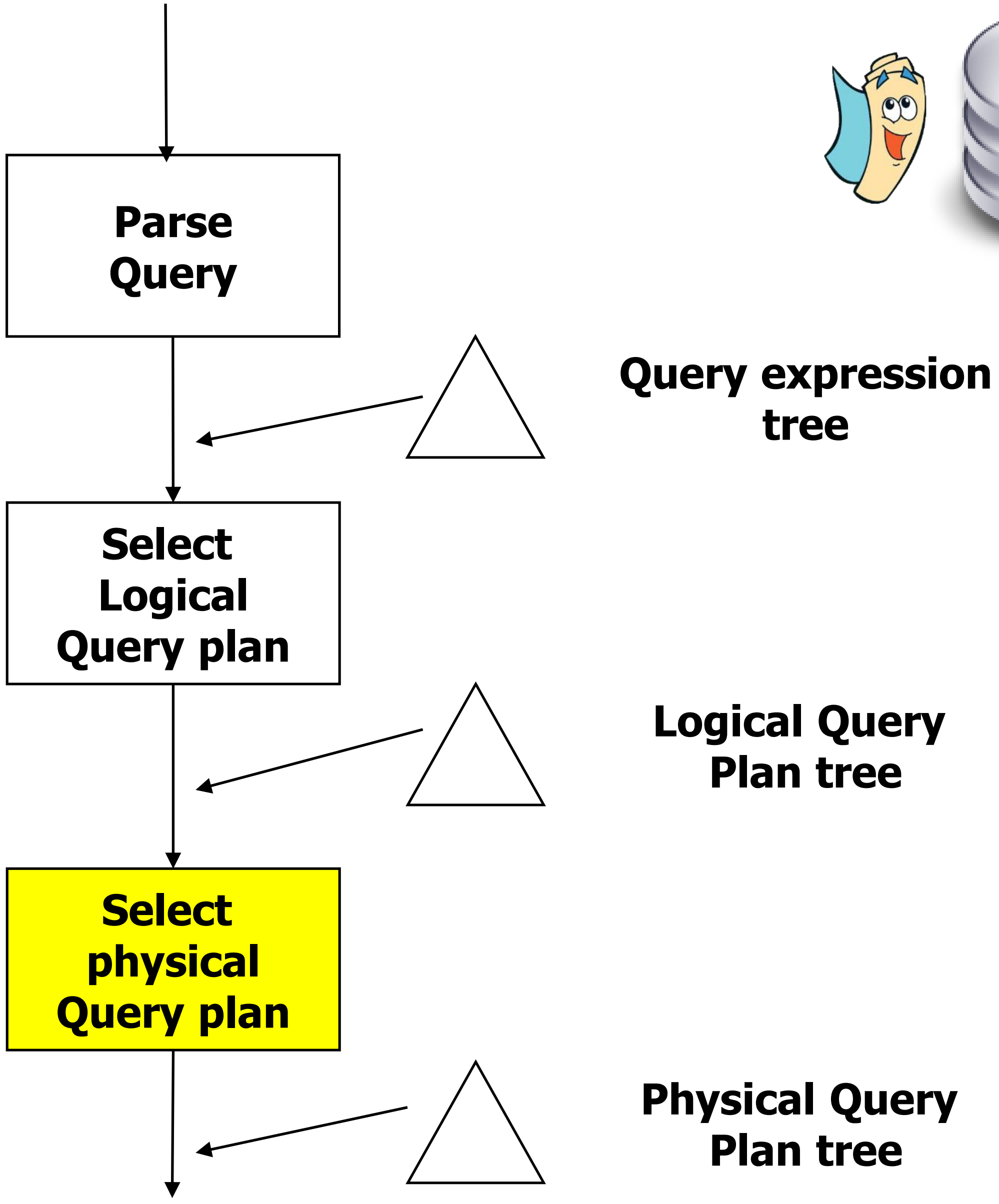


D’après C.J DATE

DDL : langage de définition des données; DML : langage de manipulation des données

From Ullman




Parse Query
Query expression tree
Select Logical Query plan
Logical Query Plan tree
Select physical Query plan
Physical Query Plan tree

Du modèle au code

Modèle

name1	owner1	name2	owner2
Fluffy	Harold	Buffy	Harold
Fluffy	Harold	Fang	Benny
Fluffy	Harold	Bowser	Diane
Claws	Gwen	Buffy	Harold
Claws	Gwen	Fang	Benny
Claws	Gwen	Bowser	Diane

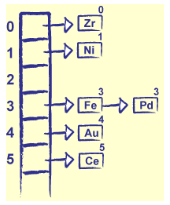
Algèbre

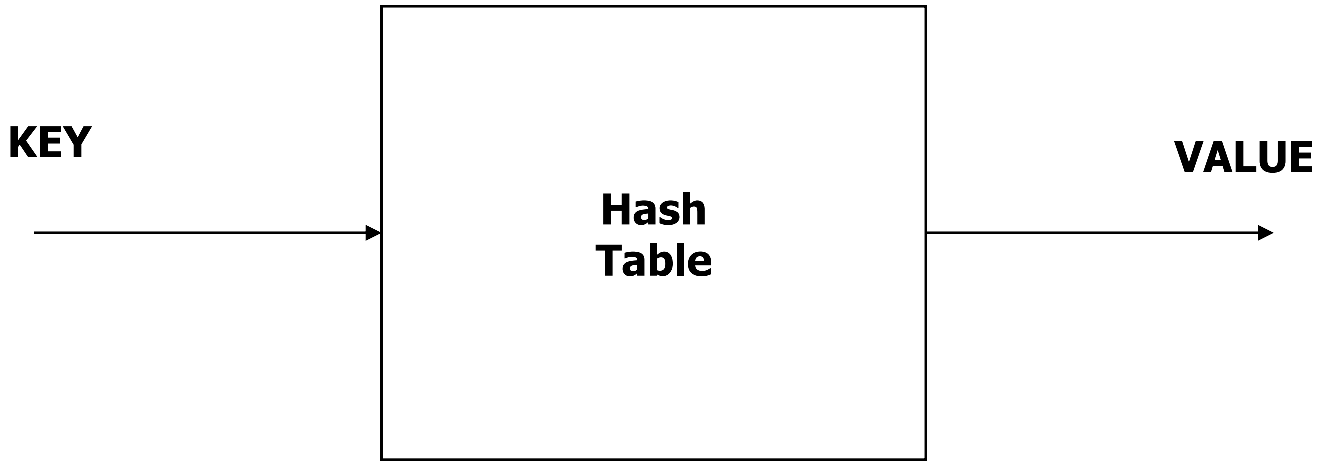
σ **owner1=owner2 (Cats $\otimes$ Dogs) = Cat $\bowtie$ Dogs**

Logiciel

Java, C++,..

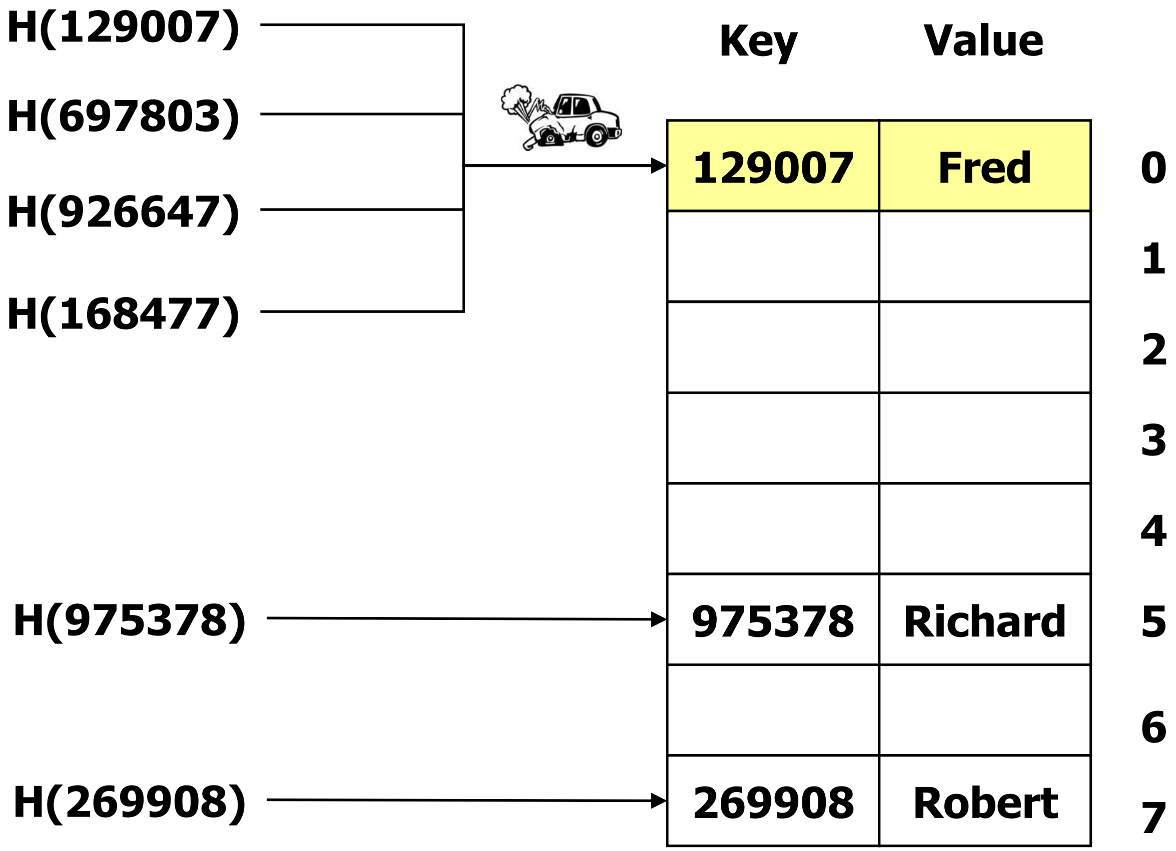
Key Value Pair

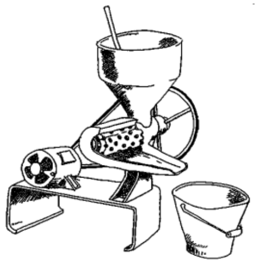




Collision handling strategies

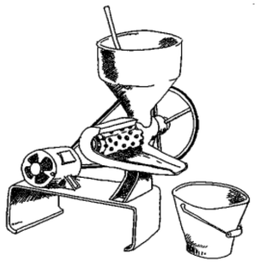
- Closed addressing (open hashing).
- Open addressing (closed hashing).

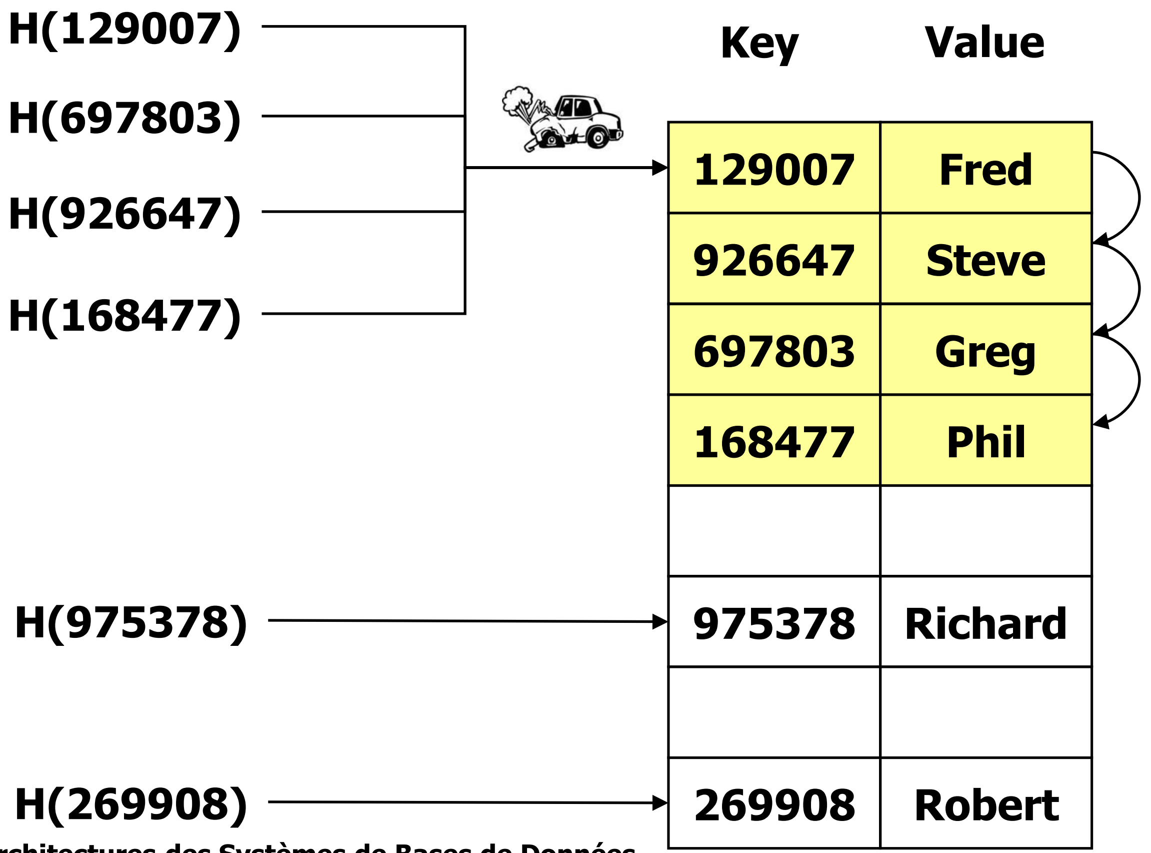




Open addressing (closed hashing).

- When there is a collision, "Probe" the array to find an empty slot after the occupied slot.

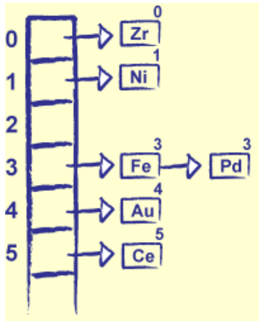


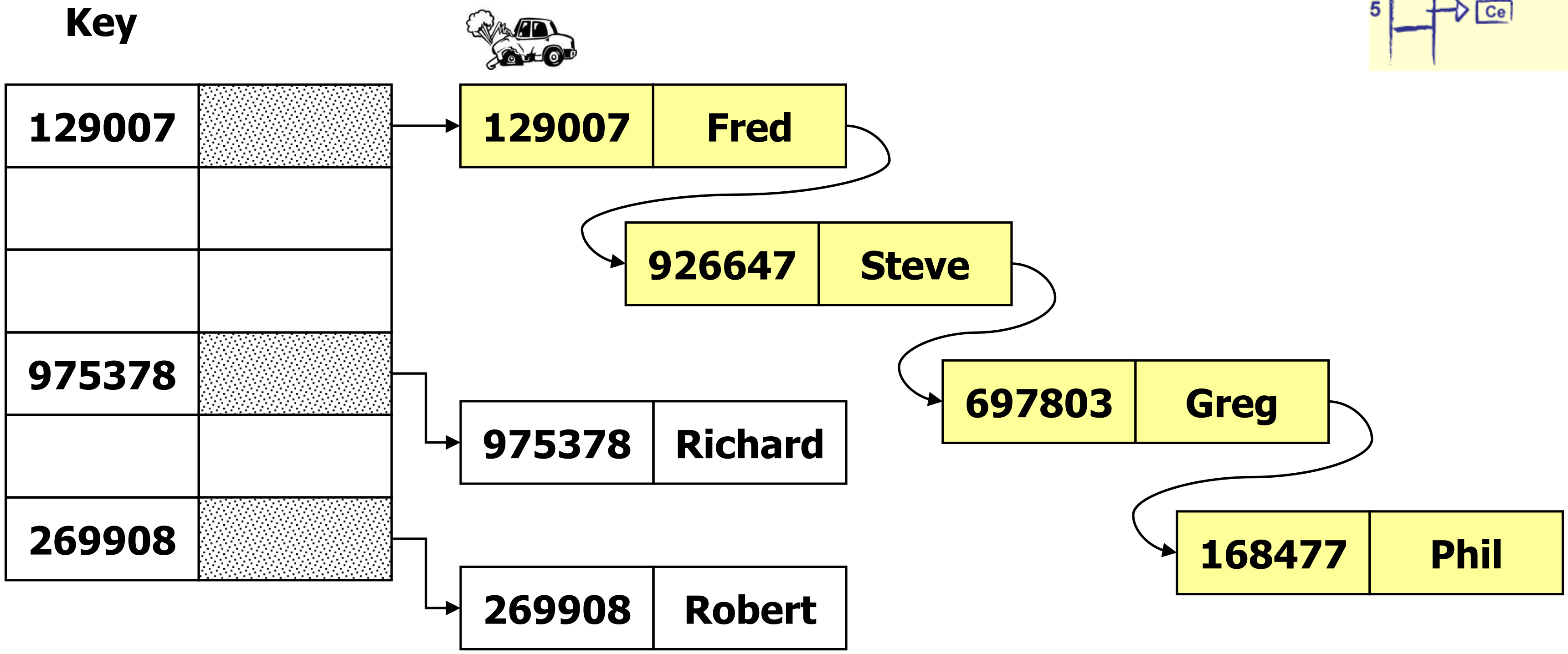


Key	Value
129007	Fred
926647	Steve
697803	Greg
168477	Phil
975378	Richard
269908	Robert

Closed addressing (open hashing).

- Each slot of the hash table contains a link to another data structure.



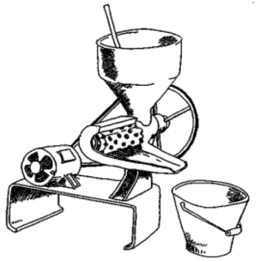


Exemple sur R

Relation R

Attribut A

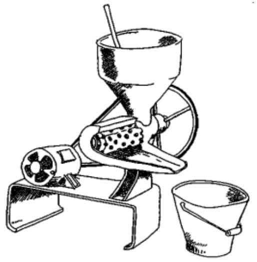
RID	R
0	B
1	O
2	E
3	P
4	C
5	L
6	X
7	N
8	D
9	M



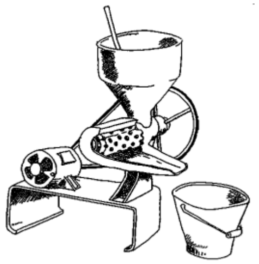
Utilisation du "modulo"

Code Ascii du caractère

Value	Key		
RID	R	CAR(R)	mod (11)
0	B	66	0
1	O	79	2
2	E	69	3
3	P	80	3
4	C	67	1
5	L	76	10
6	X	88	0
7	N	78	1
8	D	68	2
9	M	77	0



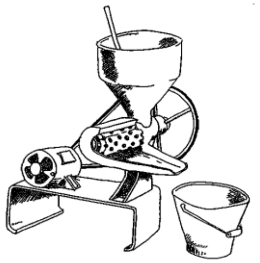
Linear Hashing



- Linear Hashing
  - Re-hachage : hi (x) = (h(x) + i) mod B
    - Stepsize : i
    - i = 1,2,3, …
- Hachage quadratique
  - Re-hachage : hi (x) = (h (x) + i ^{2}) mod B
    - Stepsize : i^2
    - i^2 = 1,4,9,…
- Hachage double
  - Re-hachage : hi (x) = (h(x) + i g (x)) mod B
    - Stepsize : g(x)

Class or Library HashLinearProbing

- Hash(key) → returns hash
- Put (key, value) → inserts key value pair
- Get (key) → gets key value
- Remove (key) → removes key preserving bucket structure.



Class HashMap JSE 1.4

- Object **get**(Object key)
  - Returns the value to which the specified key is mapped in this identity hash map, or null if the map contains no mapping for this key.
- Object **put**(Object key, Object value)
  - Associates the specified value with the specified key in this map.
- Object **remove**(Object key)
  - Removes the mapping for this key from this map if present

Linear Probing implementation

Relation

RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

N

Logical

Data Structure Implementation

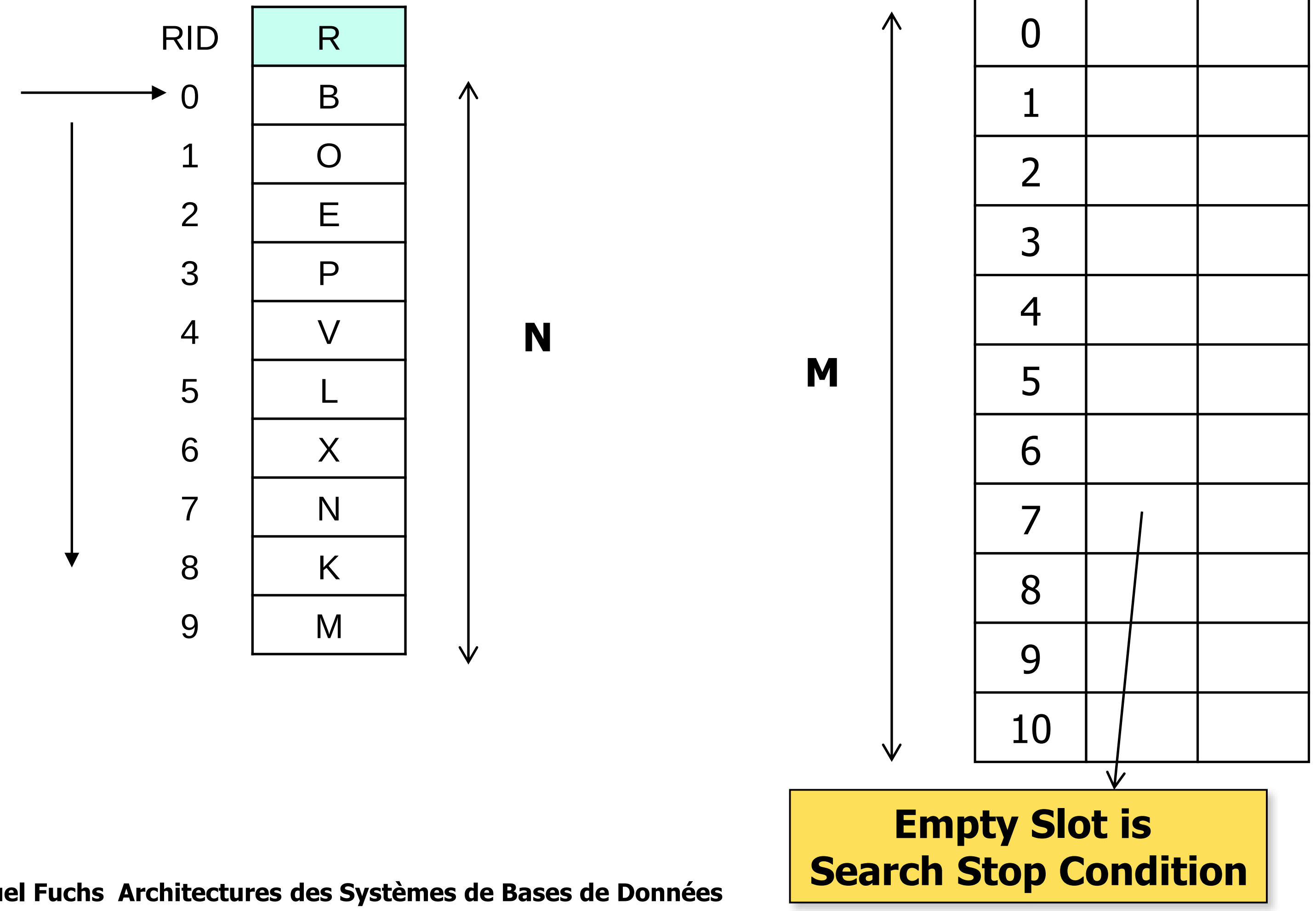
	KEY	VAL
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

M

Physical

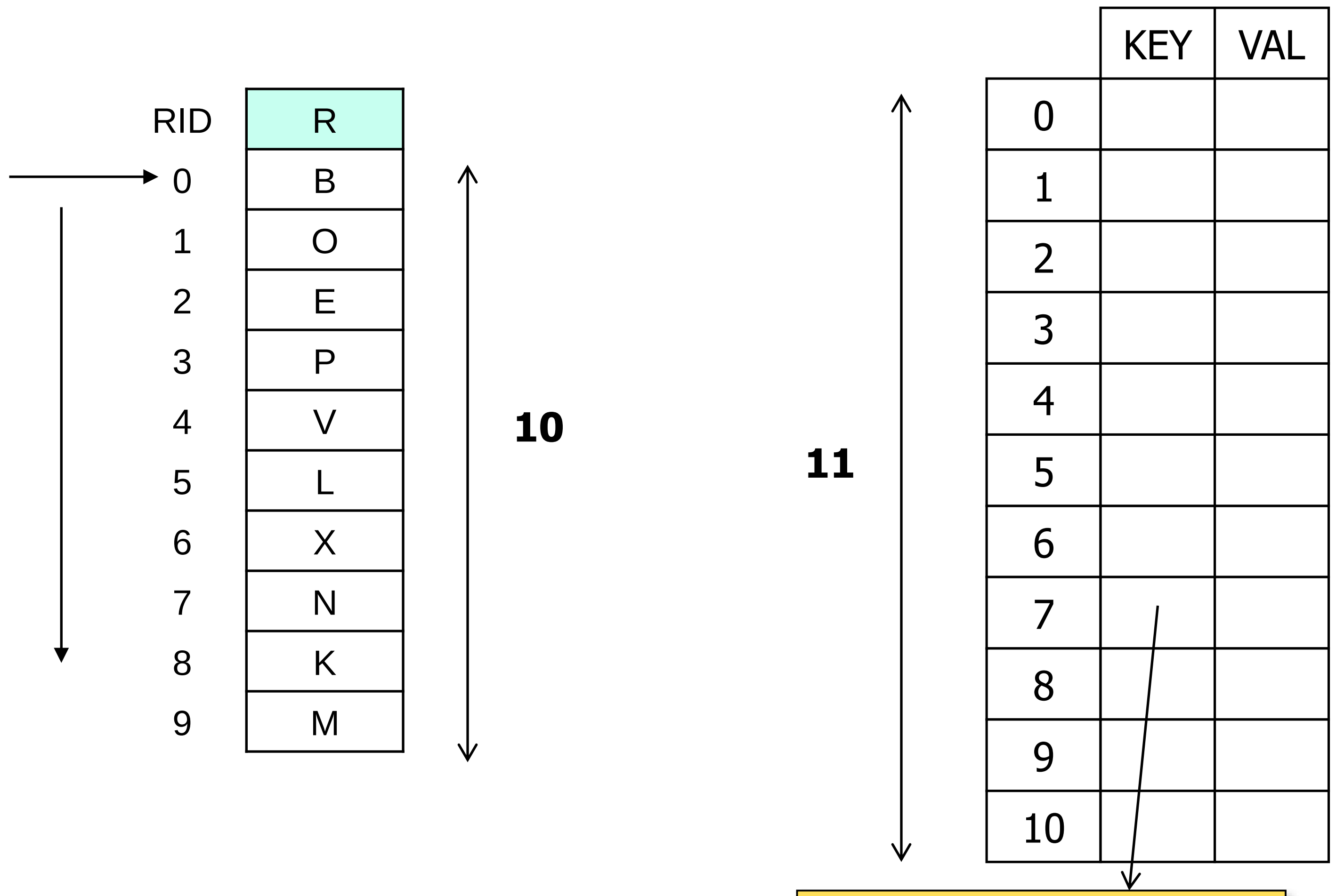
Linear Probing implementation

M > N



Linear Probing implementation

M > N



RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

	KEY	VAL
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

STOP

Empty Slot is
Search Stop Condition

Linear Probing

Put(N,7)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

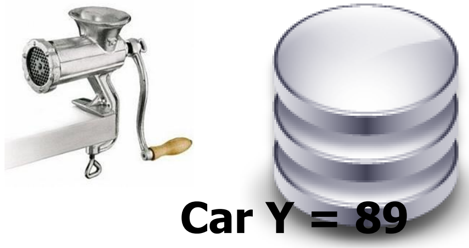
First Empty Slot ?

	KEY	VAL
0	B	0
1	X 🟥	6 🟥
2	O	1
3	E	2
4	P	3
5	N 🟩	7 🟩
6		
7		
8		
9	V	4
10	L	5

%M

Empty Slot is Search Place Stop Condition

Linear Probing



Car Y = 89
Y mod 11 = 1

Get(Y)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

Y ?

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	**8**
7	M	9
8		
9	V	4
10	L	5

Empty

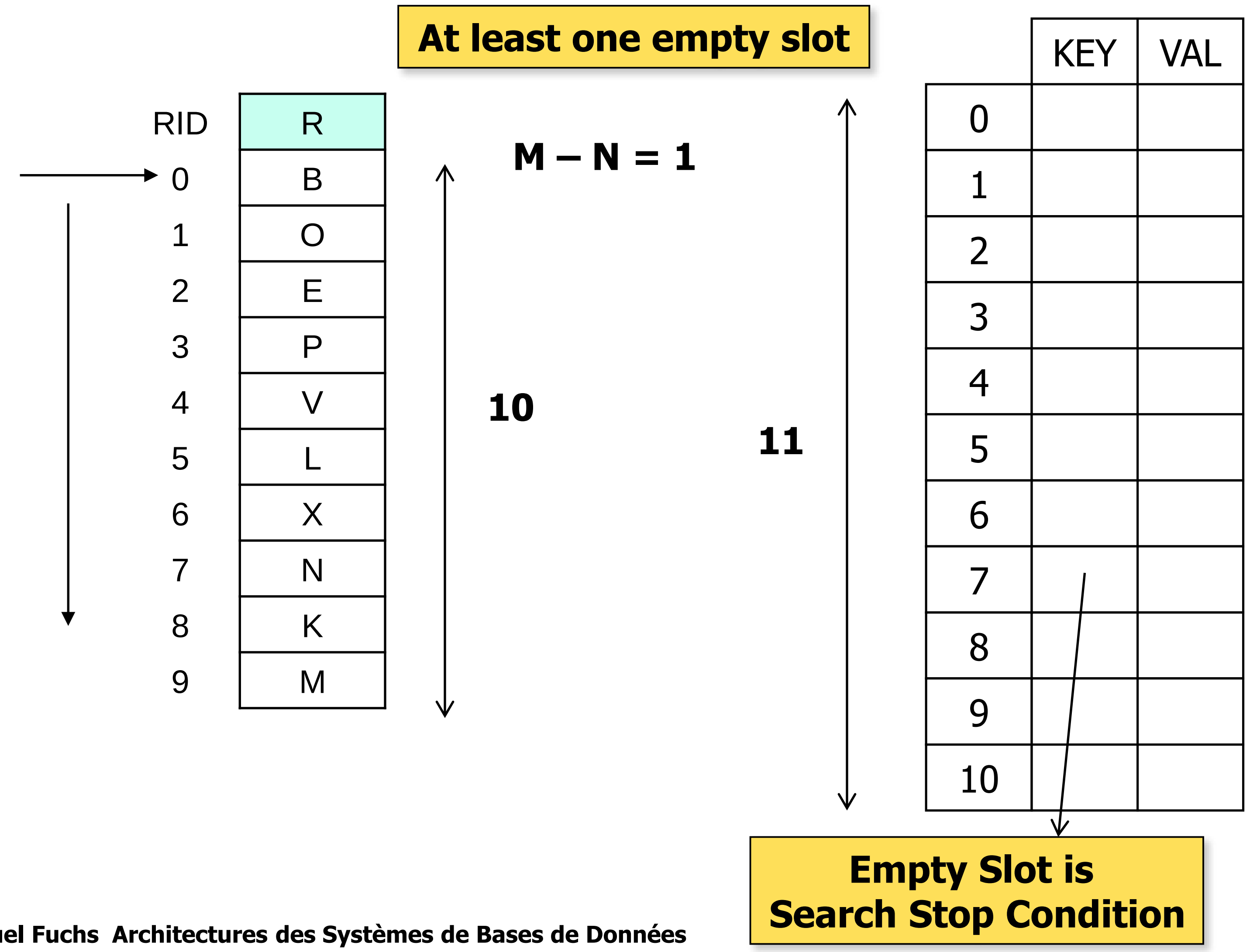
Stop

Return (-1)

Empty Slot is Search Stop Condition

Linear Probing implementation

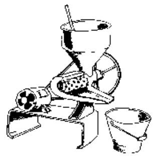
M > N

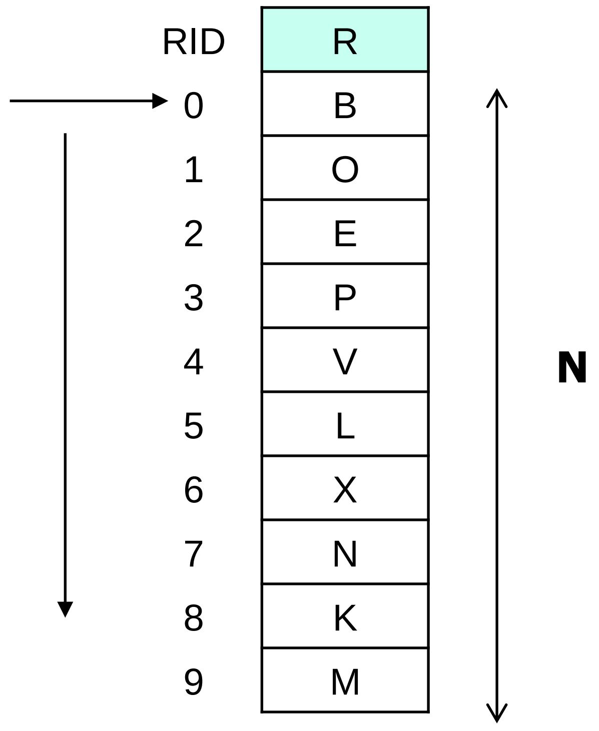


RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

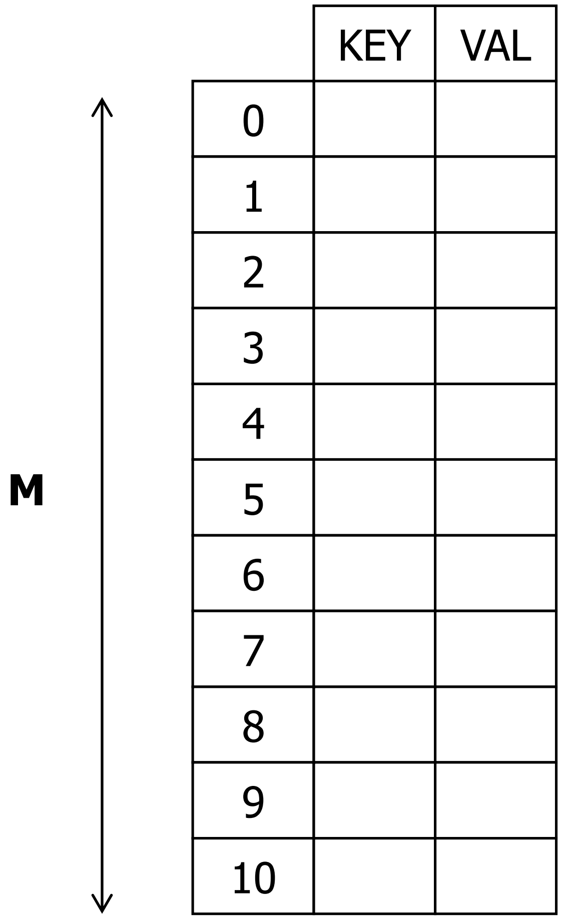
	KEY	VAL
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

Linear Probing





RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M



	KEY	VAL
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

Linear Probing

RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Example Relation Implementation

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

	KEY	VAL
0	B	0
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

Linear Probing

Hash(key) = CAR(R) Mod (M)

Value	Key	
RID	R	
0	B	0
1	O	2
2	E	3
3	P	3
4	V	9
5	L	10
6	X	0
7	N	1
8	K	9
9	M	0

M = 11

	KEY	VAL
0	B	0
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

Linear Probing

Put(B,0)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

Linear Probing

Put(O,1)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1		
2	O	1
3		
4		
5		
6		
7		
8		
9		
10		

Linear Probing

Put(E,2)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1		
2	O	1
3	E	2
4		
5		
6		
7		
8		
9		
10		

Linear Probing

Put(P,3)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

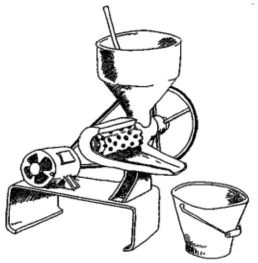
Empty Slot

?

	KEY	VAL
0	B	0
1		
2	O	1
3	E	2
4		
5		
6		
7		
8		
9		
10		

Linear Hashing

- Linear Hashing
  - Re-hachage : hi (x) = (h(x) + i) mod B
    - Stepsize : i
    - i = 1,2,3, …



Linear Probing

Put(P,3)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

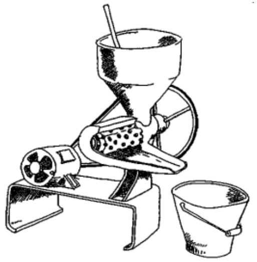
0
2
3
3
9
10
0
1
9
0

First Empty Slot ?

	KEY	VAL
0	B	0
1		
2	O	1
3	🟥 E	🟥 2
4	🟩 P	🟩 3
5		
6		
7		
8		
9		
10		

Linear Hashing

- Hachage quadratique
  - Re-hachage : hi (x) = (h (x) + i ²) mod B
    - Stepsize : i^2
    - $i^2 = 1,4,9,\ldots$



Linear Probing

Put(P,3)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

First Empty Slot ?

+ 4

	KEY	VAL
0	B	0
1		
2	O	1
3	E	2
4		
5		
6		
7	P	3
8		
9		
10		

Assume it is Occupied

Quadratic Hash

Linear Probing

Put(V,4)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1		
2	O	1
3	E	2
4	P	3
5		
6		
7		
8		
9	V	4
10		

Linear Probing

Put(L,5)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1		
2	O	1
3	E	2
4	P	3
5		
6		
7		
8		
9	V	4
10	L	5

Linear Probing

Put(X,6)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

First Empty Slot ?

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5		
6		
7		
8		
9	V	4
10	L	5

%M

Linear Probing

Put(N,7)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

First Empty Slot ?

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6		
7		
8		
9	V	4
10	L	5

%M

Linear Probing

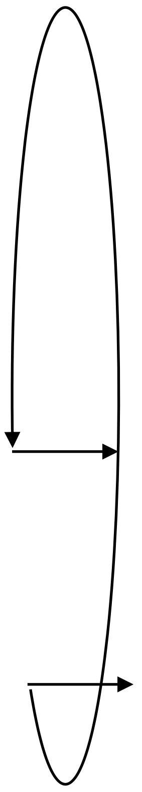
Put(K,8)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

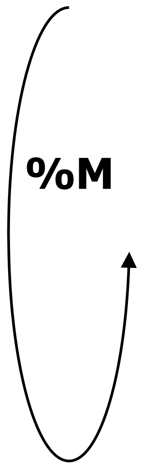
Hash(key)

0
2
3
3
9
10
0
1
9
0

First Empty Slot ?

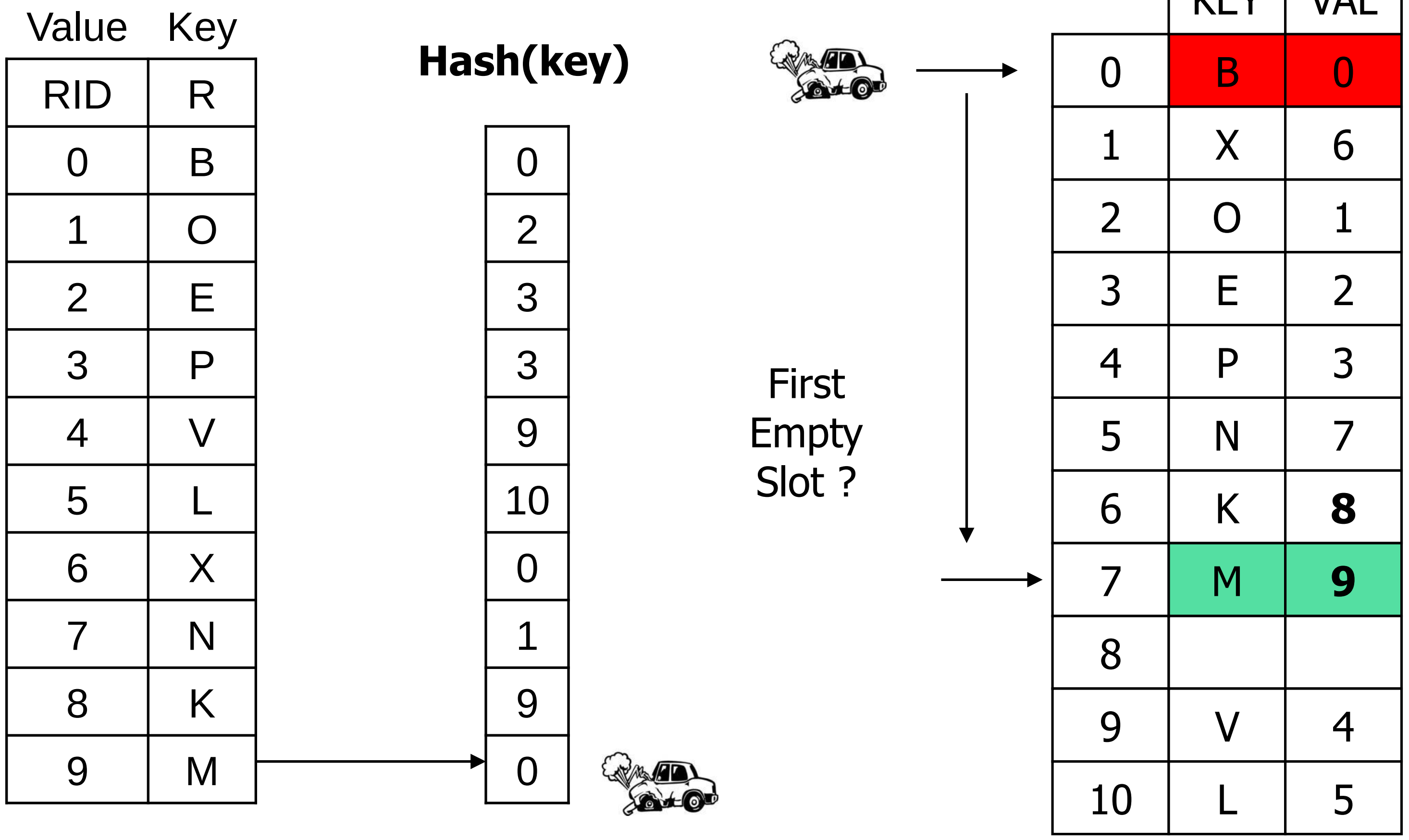


	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	**8**
7		
8		
9	V	4
10	L	5



Linear Probing

Put(M,9)



Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

First Empty Slot ?

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	**8**
7	M	**9**
8		
9	V	4
10	L	5

Linear Probing

Put(M,9)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	**8**
7	M	**9**
8		
9	V	4
10	L	5

Linear Probing

Put(M,9)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	**8**
7	M	**9**
8		
9	V	4
10	L	5

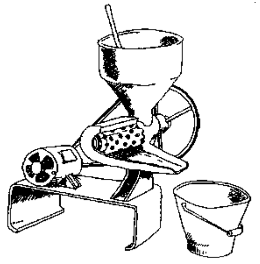
Implementation of function Hash(Key)

```
Hash (Key) { Return Key Modulo M }
```

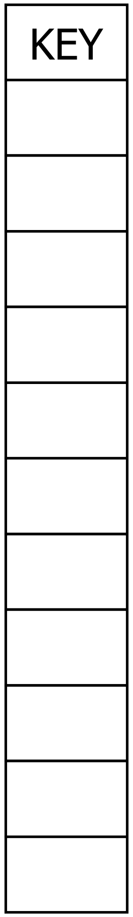
- In Java : Key % M
- Specific case of Java char : in Java char are integer (Byte).
- **char**: The char data type is a single 16-bit Unicode character. It has a minimum value of '\u0000' (or 0) and a maximum value of '\uffff' (or 65,535 inclusive).

http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

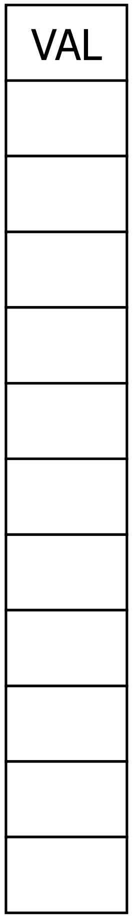
- Null char is 0 (zero).
- Default value for char is 0, or u\0000.



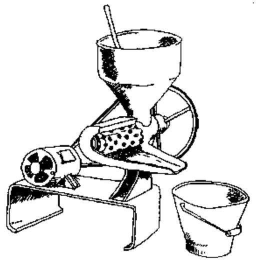
Bucket Table Implementation



char keys Array [M]



int values Array [M]



Put(key, value) simplified algo

```
M = # bucket entries

index = hash (key)

While Key [index) != empty

            index = (index + 1) % M

End while

Key [index] = key

Values [index] = value
```

Get(Key)

- Get existing key
- Get non inserted key

Linear Probing

Get(B)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

$$\xrightarrow{}$$

Hash(key)

0
2
3
3
9
10
0
1
9
0

B ?

$$\xleftarrow{}$$

B,0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	9
8		
9	V	4
10	L	5

Linear Probing

Get(M)

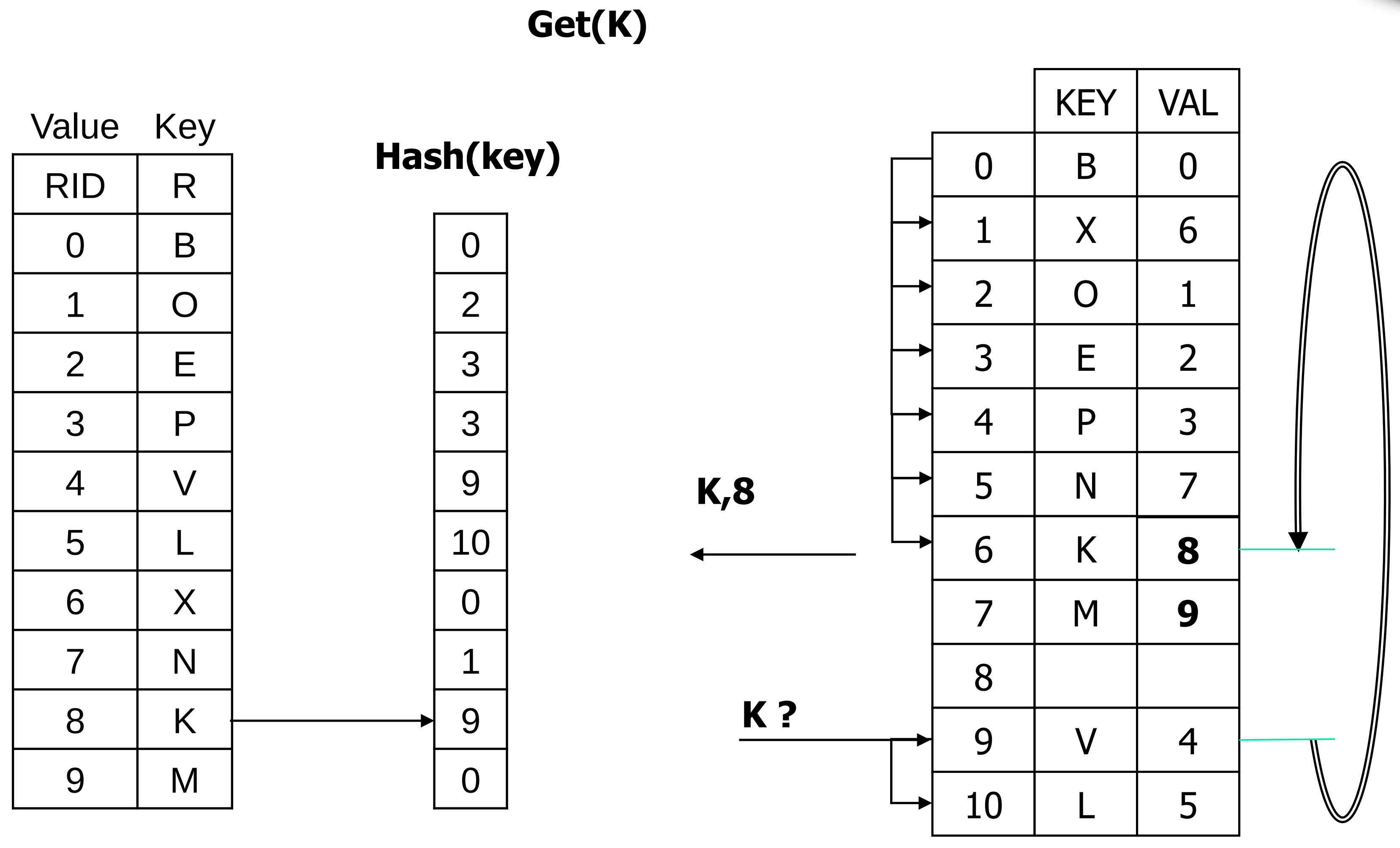
Value	Key	Hash(key)
RID	R	
0	B	0
1	O	2
2	E	3
3	P	3
4	V	9
5	L	10
6	X	0
7	N	1
8	K	9
9	M	0

M ?

M, 9

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	9
8		
9	V	4
10	L	5

Linear Probing



Value	Key	Hash(key)
RID	R	
0	B	0
1	O	2
2	E	3
3	P	3
4	V	9
5	L	10
6	X	0
7	N	1
8	K	9
9	M	0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	**8**
7	M	**9**
8		
9	V	4
10	L	5

Linear Probing

Get(Y)

Car Y = 89
Y mod 11 = 1

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

Y ?

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	**8**
7	M	9
8	🟩	🟩
9	V	4
10	L	5

Empty

Stop

Return (-1)

Get(key)

```
M = # buckets

index = hash (key)

valueToReturn = -1  // value to return if the key is not in the map

While Key [index] != key and  Key [index] != empty

            index = (index + 1) % M

End while

If (Key [index] = key) valueToReturn = Values [index]

Return valueToReturn
```

Remove (Key)

- Remove (M)
- Remove (N) : rehash end of the cluster.
- Remove (L) : rehash end of the cluster.

Linear Probing

Remove (M)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

K(0) != M

Scan for M

Blank

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	**9**
8		
9	V	4
10	L	5

Linear Probing

Remove (M)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

K(0) != M

Scan for M

Blank

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7		
8		
9	V	4
10	L	5

Linear Probing

Remove (N)

Value	Key	**Hash(key)**
RID	R	
0	B	0
1	O	2
2	E	3
3	P	3
4	V	9
5	L	10
6	X	0
7	N	1
8	K	9
9	M	0

K(1) != N

Scan for N

Blank and rehash
End of cluster

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	**9**
8		
9	V	4
10	L	5

Linear Probing

Remove (N)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

K(1) != N

Scan for N

Blank and rehash
End of cluster

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	**9**
8		
9	V	4
10	L	5

$\updownarrow$ **Cluster**

Linear Probing

Remove (N)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

K(1) != N

Scan for N

Blank and rehash
End of cluster

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5		
6	K	8
7	M	**9**
8		
9	V	4
10	L	5

$\updownarrow$ **EOf cluster**

Linear Probing

Remove (N)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

K(1) != N

Scan for N

Blank and rehash
End of cluster

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5		
6	K	8
7	M	**9**
8		
9	V	4
10	L	5

$\updownarrow$ **EOf cluster**

Blank Key(6) , Val(6)
put(K,8)

Linear Probing

Remove (N)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

K(1) != N

Scan for N

Blank and rehash
End of cluster

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5		
6		
7	M	**9**
8		
9	V	4
10	L	5

$\updownarrow$ **EOf cluster**

$\rightarrow$ **Blank Key(6) , Val(6)**
put(K,8)

Linear Probing

Remove (N)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

K(1) != N

Scan for N

Blank and rehash
End of cluster

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5		
6		
7	M	**9**
8		
9	V	4
10	L	5

Blank Key(6) , Val(6)
put(K,8)

Linear Probing

Remove (N)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

K(1) != N

Scan for N

Blank and rehash
End of cluster

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5		
6		
7	M	**9**
8		
9	🟥 V	🟥 4
10	L	5

Blank Key(6), Val(6)
put(K,8)

Linear Probing

Remove (N)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

K(1) != N

Scan for N

Blank and rehash
End of cluster

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	K	8
6		
7	M	**9**
8		
9	V	4
10	L	5

Blank Key(7) ,Val(7)
put(K,8)

Linear Probing

Remove (N)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

K(1) != N

Scan for N

Blank and rehash
End of cluster

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	K	8
6		
7	M	**9**
8		
9	V	4
10	L	5

$\longrightarrow$ **Blank Key(7) ,Val(7)**
put(K,8)

Linear Probing

Remove (N)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

K(1) != N

Scan for N

Blank and rehash End of cluster

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	K	8
6		
7		
8		
9	V	4
10	L	5

$\longrightarrow$ **Blank Key(7) , Val(7)**
put(M,9)

Linear Probing

Remove (N)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

K(1) != N

Scan for N

Blank and rehash End of cluster

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	K	8
6	🟩	🟩
7		
8		
9	V	4
10	L	5

First Empty Slot ?

Blank Key(7) , Val(7)
put(M,9)

Linear Probing

Remove (N)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

K(1) != N

Scan for N

Blank and rehash End of cluster

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	K	8
6	M	9
7		
8		
9	V	4
10	L	5

First Empty Slot ?

Blank Key(7) , Val(7)
put(M,9)

Linear Probing

Remove (L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	**9**
8		
9	V	4
10	L	5

Linear Probing

Remove (L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	**8**
7	M	**9**
8		
9	V	4
10	L	5

Blank and rehash
End of cluster

K(10) = L $\longrightarrow$ **blank**

Linear Probing

Remove (L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	**8**
7	M	**9**
8		
9	V	4
10		

Blank and rehash
End of cluster

K(10) = L $\longrightarrow$ blank

Linear Probing

Remove (L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	**8**
7	M	**9**
8		
9	V	4
10		

Blank and rehash
End of cluster

EOf cluster

K(10) = L $\longrightarrow$ **blank**

Linear Probing

Remove (L)

Value	Key	Hash(key)
RID	R	
0	B	0
1	O	2
2	E	3
3	P	3
4	V	9
5	L	10
6	X	0
7	N	1
8	K	9
9	M	0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	**8**
7	M	**9**
8		
9	V	4
10		

EOf cluster

Blank and rehash End of cluster

Blank Key(0) , Val(0) put(B,0)

Linear Probing

Remove (L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	**8**
7	M	9
8		
9	V	4
10		

EOf cluster

Blank and rehash End of cluster

Blank Key(1) , Val(1) put(X,6)

Linear Probing

Remove (L)

Value	Key	Hash(key)
RID	R	
0	B	0
1	O	2
2	E	3
3	P	3
4	V	9
5	L	10
6	X	0
7	N	1
8	K	9
9	M	0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	9
8		
9	V	4
10		

EOf cluster

Blank and rehash End of cluster

Blank Key(2) , Val(2) put(O,1)

Linear Probing

Remove (L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	9
8		
9	V	4
10		

EOf cluster

Blank and rehash End of cluster

Blank Key(3) , Val(3) put(E,2)

Linear Probing

Remove (L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	9
8		
9	V	4
10		

EOf cluster

Blank and rehash End of cluster

Blank Key(4), Val(4) put(P,3)

Linear Probing

Remove (L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	9
8		
9	V	4
10		

EOf cluster

Blank and rehash End of cluster

Blank Key(5), Val(5) put(N,7)

Linear Probing

Remove (L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	**8**
7	M	9
8		
9	V	4
10		

EOf cluster

Blank and rehash End of cluster

Blank Key(6) , Val(6) put(K,8)

Linear Probing

Remove (L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	🟩	🟩
7	M	**9**
8		
9	V	4
10		

EOf cluster

Blank and rehash End of cluster

Blank Key(6) , Val(6) put(K,8)

Linear Probing

Remove (L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	🟩	🟩
7	M	**9**
8		
9	V	4
10	K	**8**

EOf cluster

Blank and rehash End of cluster

Blank Key(6) , Val(6) put(K,8)

Linear Probing

Remove (L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6		
7	M	9
8		
9	V	4
10	K	**8**

EOf cluster

Blank and rehash End of cluster

Blank Key(7) , Val(7) put(M,9)

Linear Probing

Remove (L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	M	**9**
7	🟩	🟩
8		
9	V	4
10	K	**8**

EOf cluster

Blank and rehash End of cluster

Blank Key(7) , Val(7) put(M,9)

Linear Probing

Remove (L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	M	**9**
7		
8		
9	V	4
10	K	**8**

EOf cluster

Blank and rehash End of cluster

Blank Key(7) , Val(7) put(M,9)

Linear Probing

Remove (L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	M	**9**
7		
8		
9	V	4
10	K	**8**

EOf cluster

Blank and rehash End of cluster

Blank Key(7) , Val(7) put(M,9)

Remove (key) simplified algo

```
M = # buckets

index = hash (key)
While Key [index) != key and Key [index) != empty
        index = (index + 1) % M
End while

Key [index] = 0,  Value [index] = 0

// rehash

index = (index + 1) % M
While Key [index) != empty

        savedKey = Key [index],  savedValue = Value [index]
        Key [index] = 0  Value [index] = 0
        Put ( savedKey , savedValue )
        index = (index + 1) % M

End while
```