SQL – three-valued logics

Name	Surname	Student
Jaroslav	Novák	true
Josef	Novotný	false
Jiří	Brabenec	

```
SELECT * FROM OSOBA WHERE Student != true
```

What will be the result?

SQL – three-valued logics

Name	Surname	Student
Jaroslav	Novák	true
Josef	Novotný	false
Jiří	Brabenec	

```
SELECT * FROM OSOBA WHERE Student != true
```

What will be the result?

Name	Surname	Student
Josef	Novotný	false

SQL – three-valued logics

	A = true	**A = false**	**A = null**
A == true	true	false	null
A != true	false	true	null
A == false	false	true	null
A != false	true	false	null

- is null
- is true
- is false

	A = true	**A = false**	**A = null**
A is true	true	false	false
A is not true	false	true	true
A is false	false	true	false
A is not false	true	false	true
A is null	false	false	true
A is not null	true	true	false

SQL – tříhodnotová logika

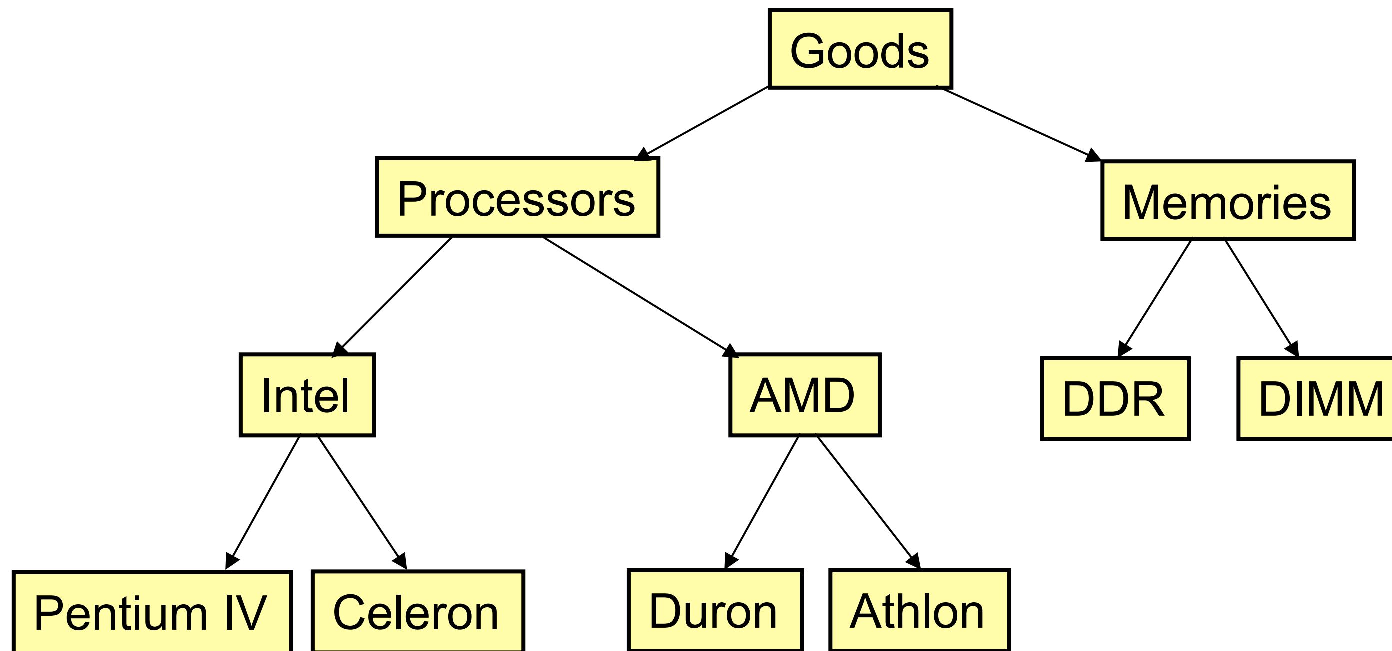
	A and B		
	B == true	**B == false**	**B == null**
A == true	true	false	null
A == false	false	false	false
A == null	null	false	null

	A or B		
	B == true	**B == false**	**B == null**
A == true	true	true	true
A == false	true	False	null
A == null	true	null	null

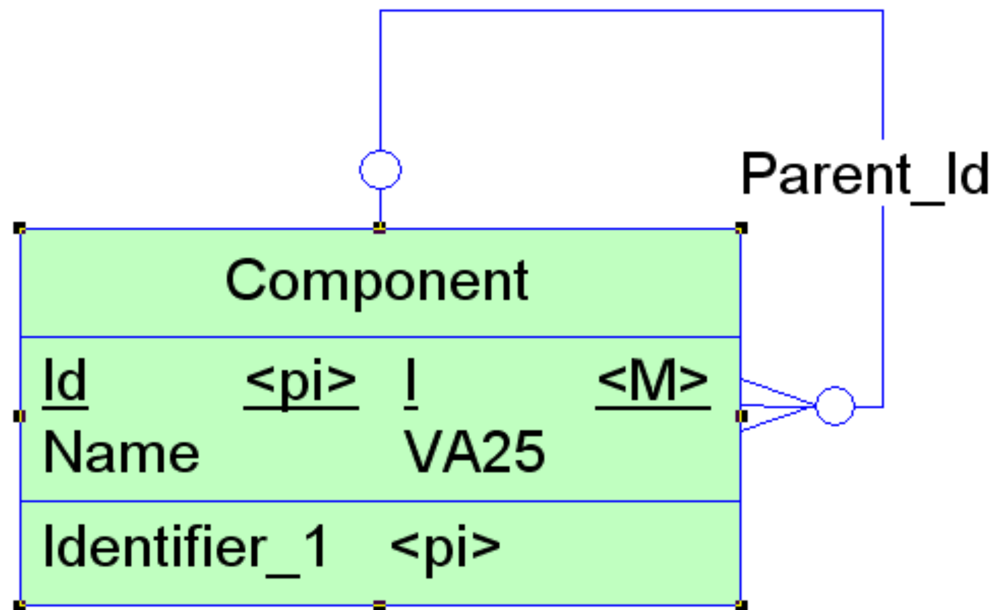
	Not A
A == true	false
A == false	true
A == null	null

Tree structures in relational database

Representing a tree structure in a relational database

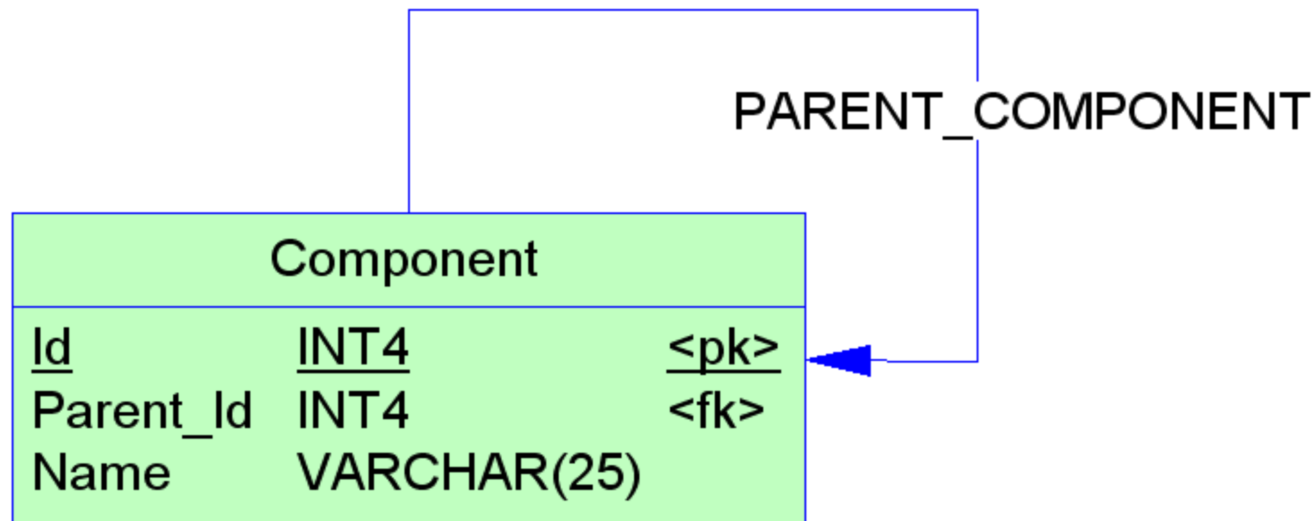


Representing a tree structure in a relational database



Conceptual model

Representing a tree structure in a relational database



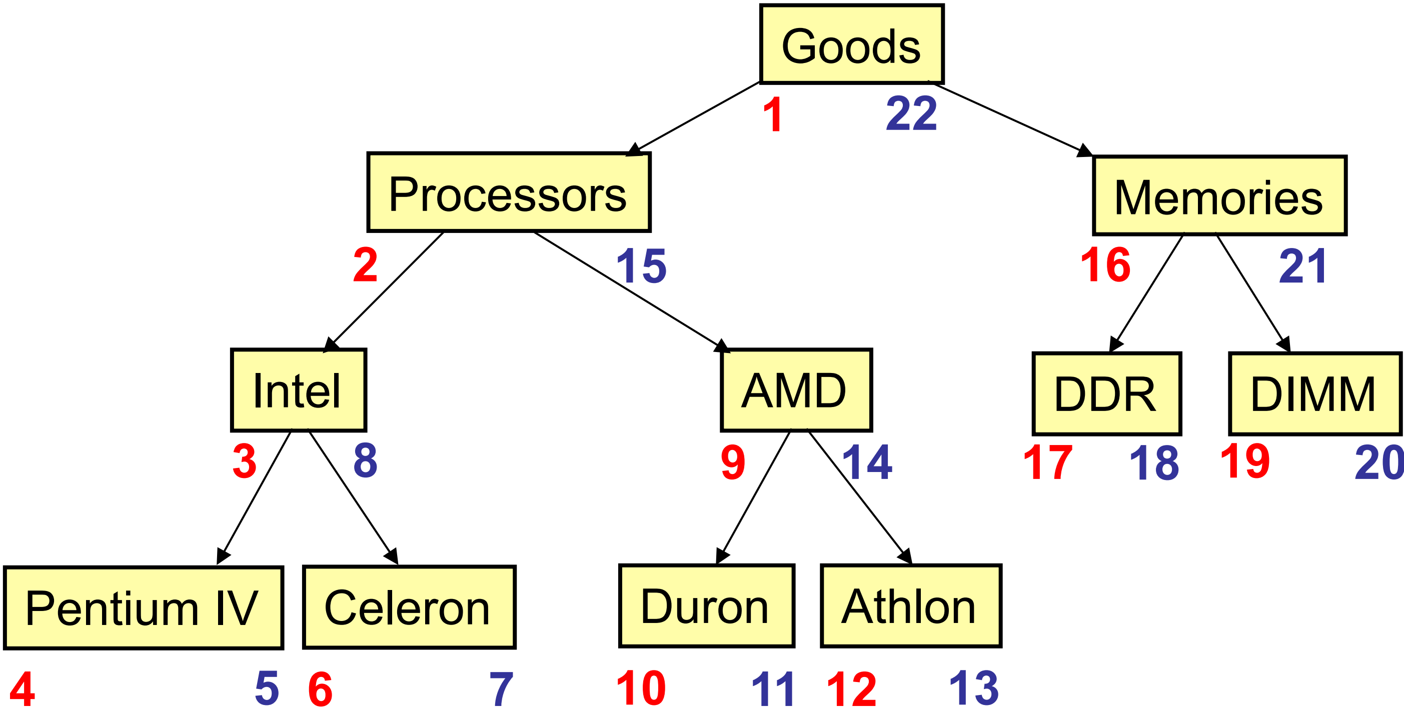
Logical model

Representing a tree structure in a relational database

Search for all nodes of given subtree - recursione

```
void getTree(int parent) {
  ResultSet rsData = statement.executeQuery(
                "SELECT * FROM TREE WHERE Parent_Id=" + parent);
  while (rsData.next()) {
     System.out.println();
     System.out.println(rsData.getString("Name"));
     parent = rsData.getString("Parent_Id");
     getTree(parent);
  }
}
```

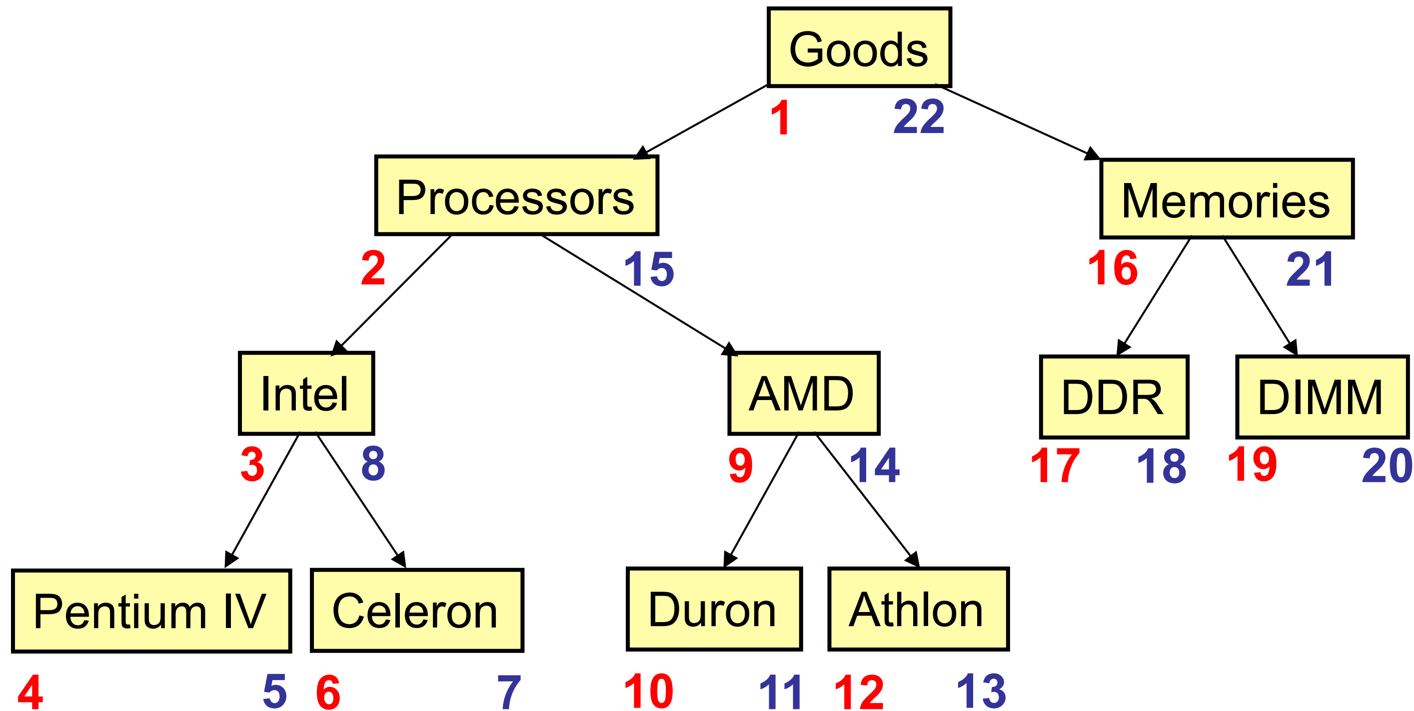
Representing a tree structure in a relational database



Stromové struktury a relační databáze

COMPONENTS				
ID	**NAME**	**PARENT_ID**	**LEFT**	**RIGHT**
1	Kategorie zboží	0	1	22
2	Procesory	1	2	15
3	Intel	2	3	8
4	Pentium IV	3	4	5
5	Celeron	3	6	7
6	AMD	2	9	14

Representing a tree structure in a relational database



SELECT *
FROM *COMPONETS C1, COMPONENTS C2*
WHERE *C1.NAME = "INTEL"* AND
C2.LEFT > C1.LEFT AND
C2.RIGHT < C1.RIGHT

Indices using B-trees

Indicing – a means for perfomance optimization

SELECT *

    FROM *PERSON*

    WHERE *(GENDER=FEMALE)* AND *(AGE < 32)*

The response will be much fatser if there is an index with the index expression {*GENDER*, *AGE*}.

One of the values of this index expressions may be (for example) the pair <*FEMALE, 27*>.

Indicing techniques theory (search data structures) is using the term **key** instead of **index expression**. Different from **table key** !

CREATE INDEX *PERSON_GENDER_AGE* **ON** *PERSON* (*GENDER*, *AGE*)

Indicing – a means for perfomance optimization

Disjunction in a where condition – be carefull.

SELECT *

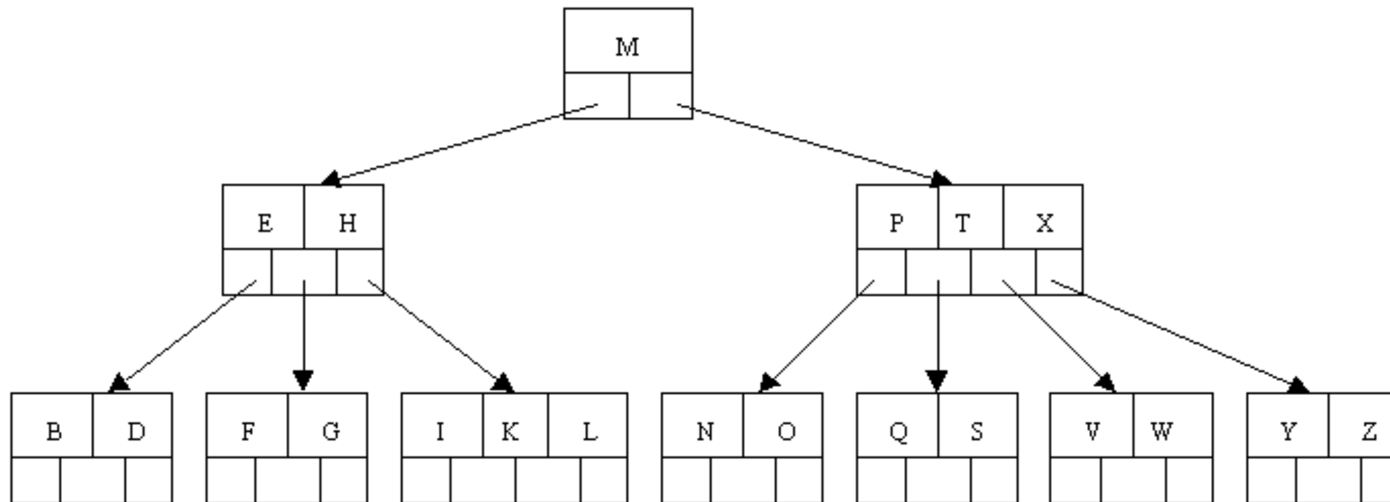
FROM *PERSON*

WHERE *(GENDER=FEMALE)* **OR** *(AGE < 32)*

DBMS should make use of two indices – a to {*GENDER*} a {*AGE*}

Some DBMSs (e.g. PostgreSQL) may not use existing indices eficiently – response to disjunctive queries is slow.

B-tree



B-tree has edfiend:

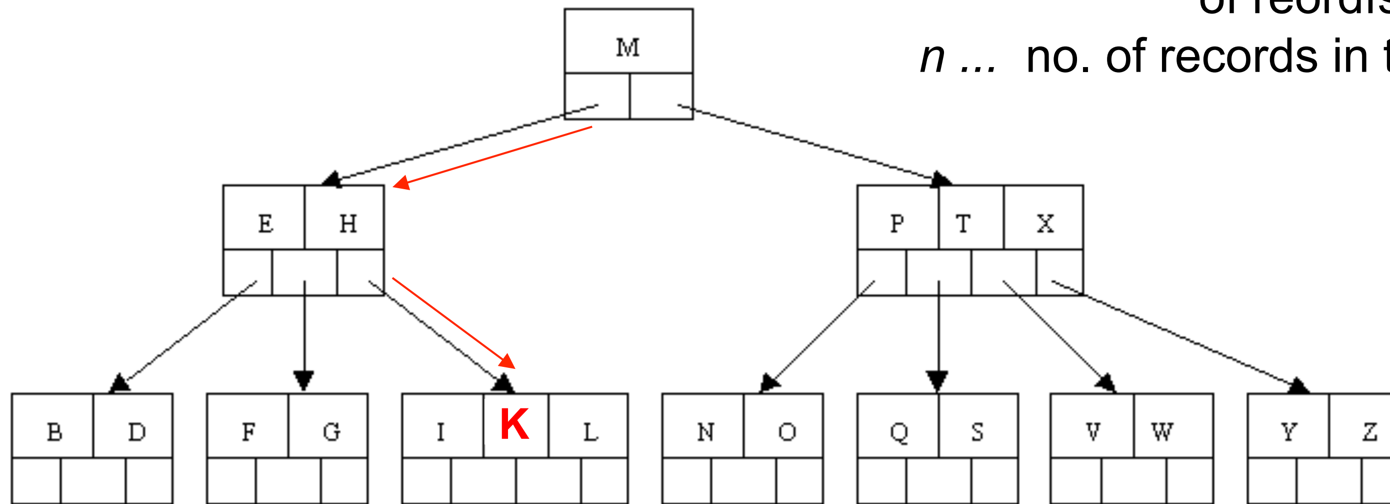
- maximum node capacity (max. number of records in a node)
- minimum node capacity (min. number of records in a tree)

Records inside of a node sorted by the value of the key.

B-tree

max(min) … max (min) number of reordfs in a node

n … no. of records in the DB

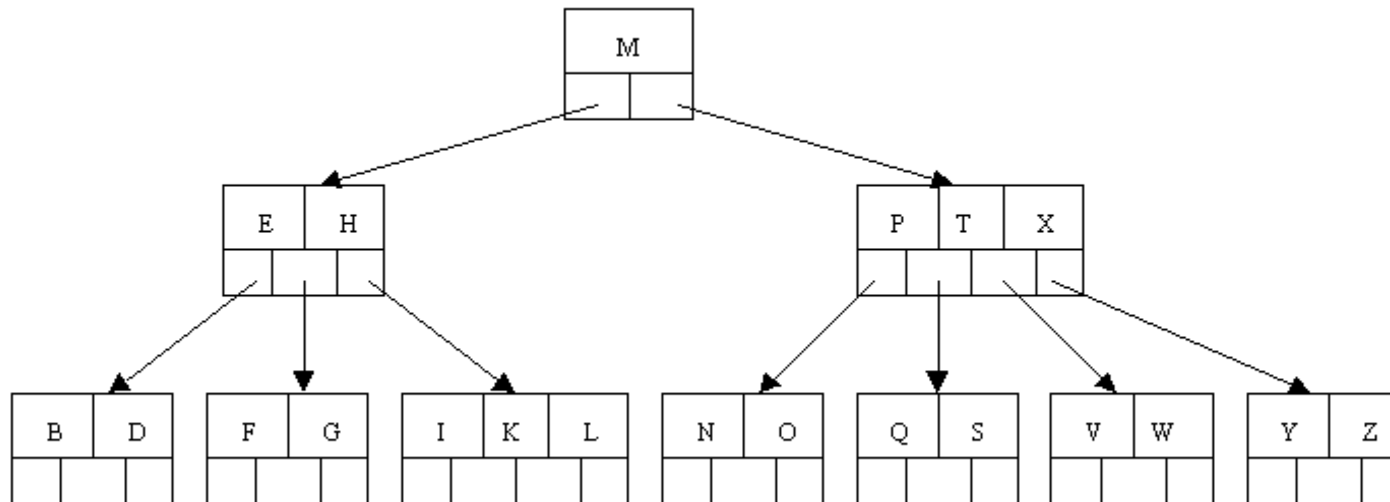


- Each node – 1 database page (typically 1 page = 1 sector)
- Aim – minimization of the number of databas accesses
- Depth B-stromu

best case (all nodes 100% full) ... $\log_{max} n$

worst case (all nodes have min rocords ... $\log_{min} n$

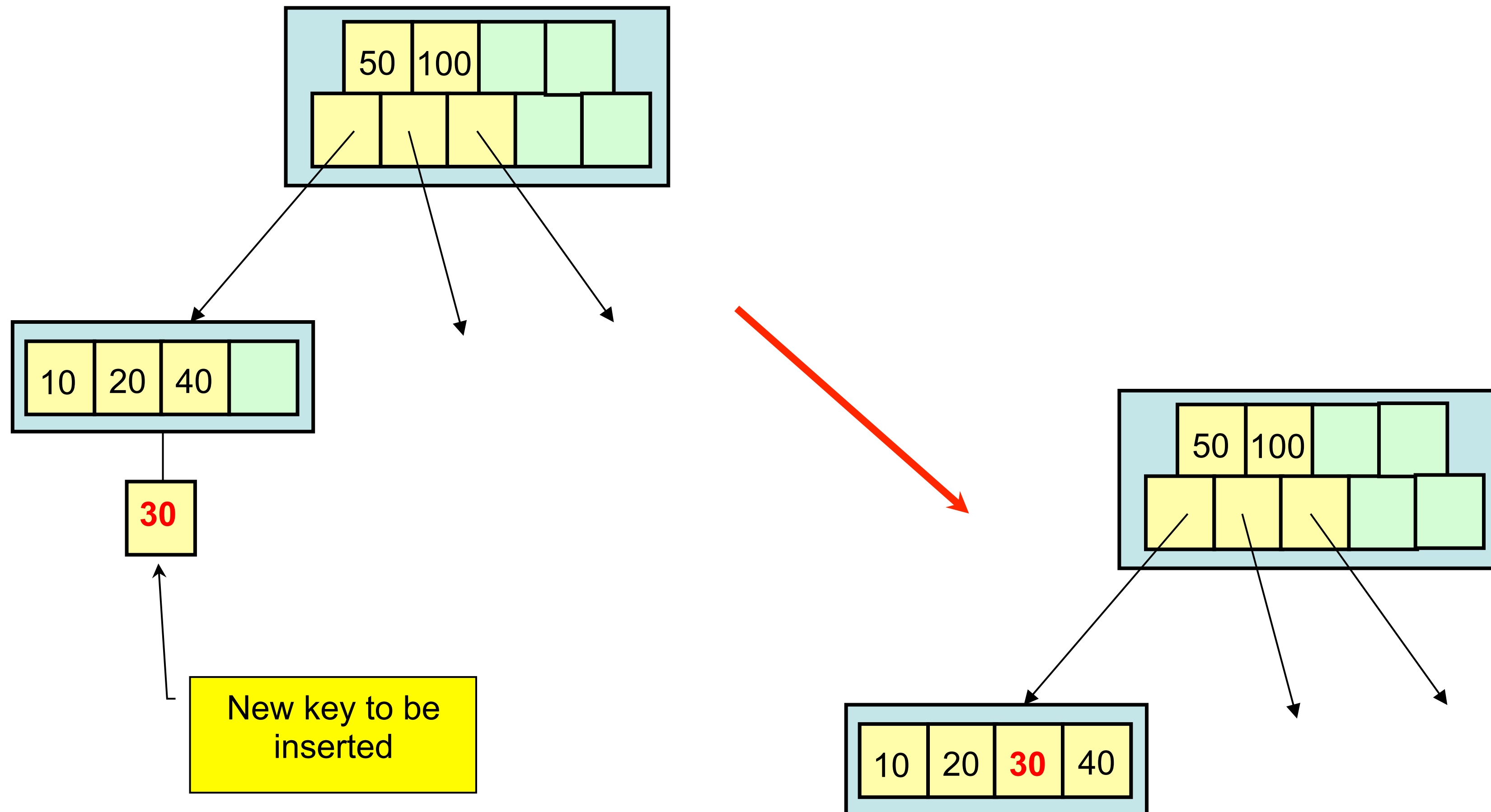
Insert into a B-tree



- Each tree – 1 database page (typically 1 page = 1 disk sector)
- Initial tree construction – do not fill nodes fully, leave 25% - 30% of capacity free as a space for records that will be inserted in the future
- If a node is full and a new record should be inserted into it, the node needs to be split. In such a case also the predecestor node needs to be modified.

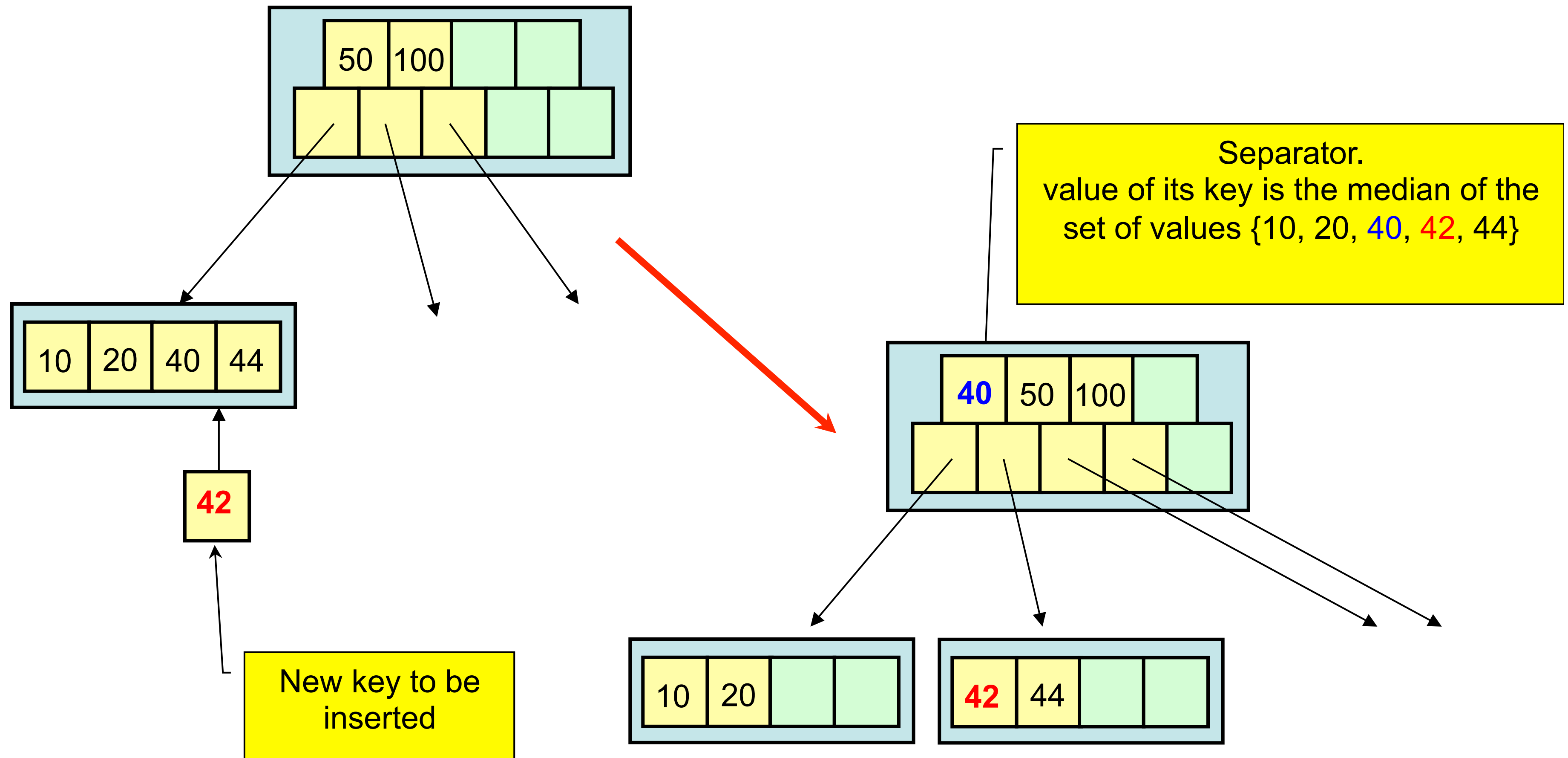
Inserting a record into B-tree

Trivial, if the node capacity is not exhausetd yet:



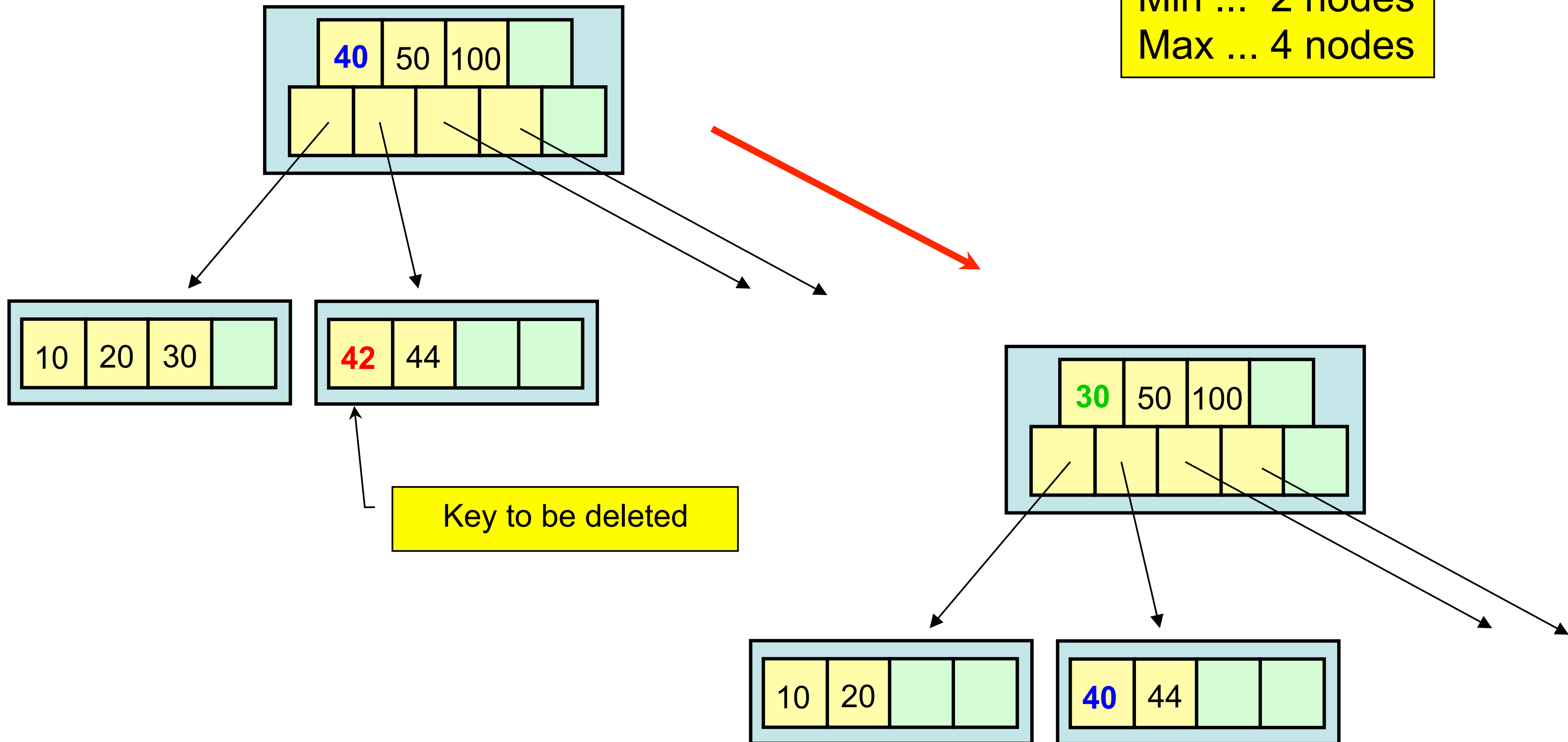
Inserting a record into B-tree

If the node is full, it must be splitted:



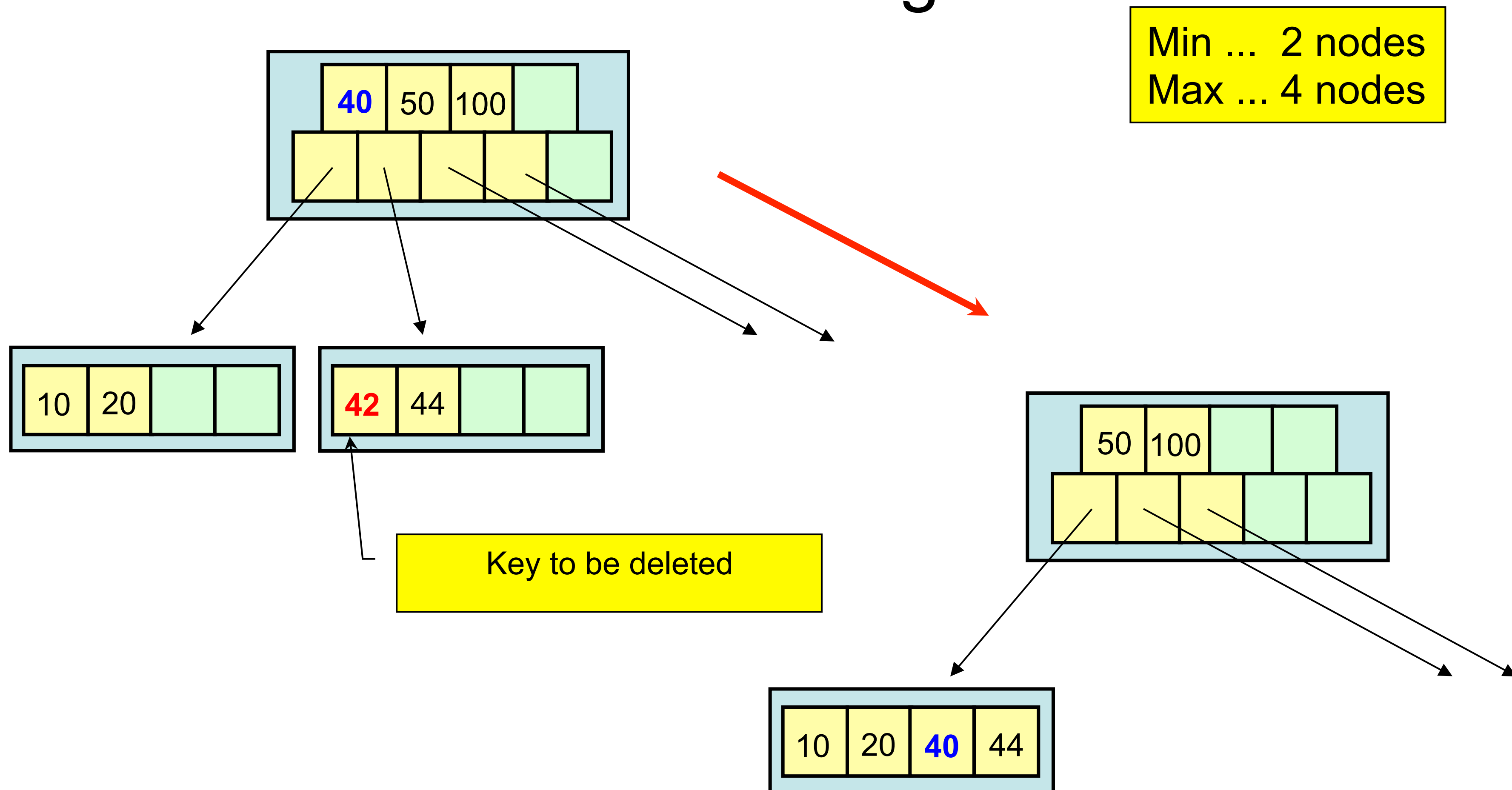
Deletion of a record in the leaf of the B-tree

move a key

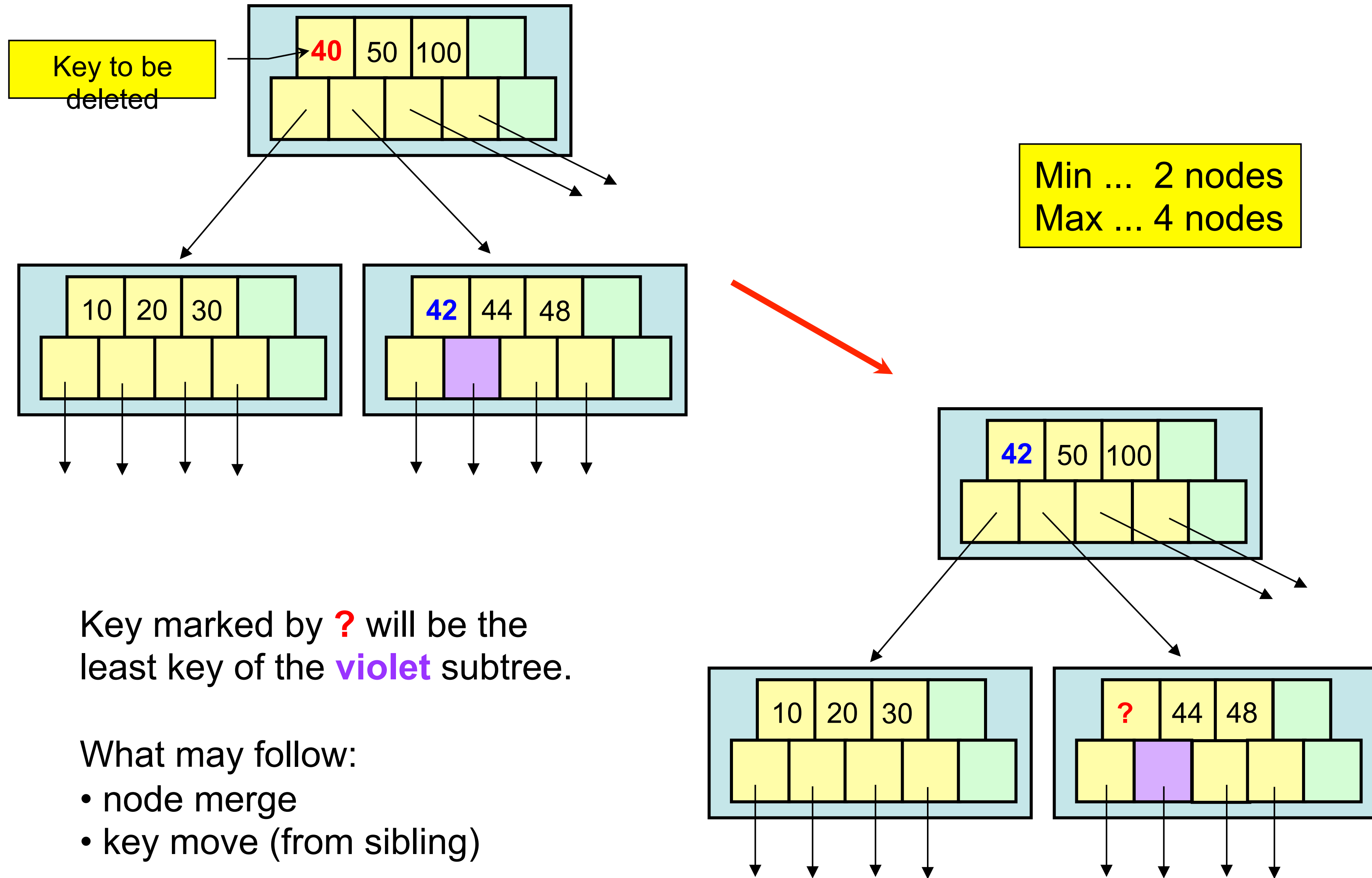


Deletion of a record in the leaf of the B-tree

Node merge



Deletion of a record in a non-leaf node of B-tree



Key marked by **?** will be the least key of the **violet** subtree.

What may follow:

- node merge
- key move (from sibling)

Deletion of a record in a non-leaf node of B-tree

- This approach means, that we remove the key to be deleted and afterwards bring the tree into balanced status again.
- Not the only one algorithm, other approaches exist.