SQL – tříhodnotová logika

Jmeno	**Prijmeni**	**Student**
Jaroslav	Novák	true
Josef	Novotný	false
Jiří	Brabenec	

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

Jaký bude výsledek?

SQL – tříhodnotová logika

Jmeno	Prijmeni	Student
Jaroslav	Novák	true
Josef	Novotný	false
Jiří	Brabenec	

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

Jaký bude výsledek?

Jmeno	Prijmeni	Student
Josef	Novotný	false

SQL – tříhodnotová logika

	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

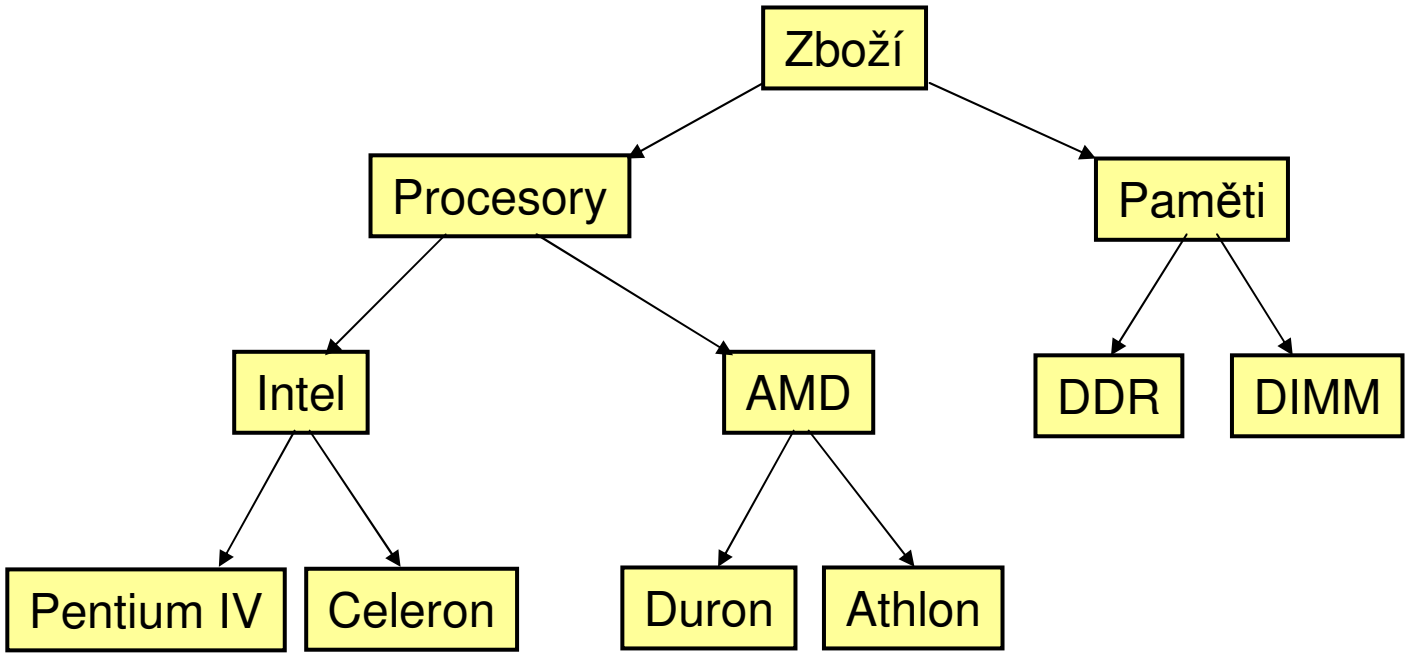
	A and B		
	B == true	**B == false**	**B == null**
A == true	true	false	null
A == false	false	false	false
A == null	null	false	null

	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

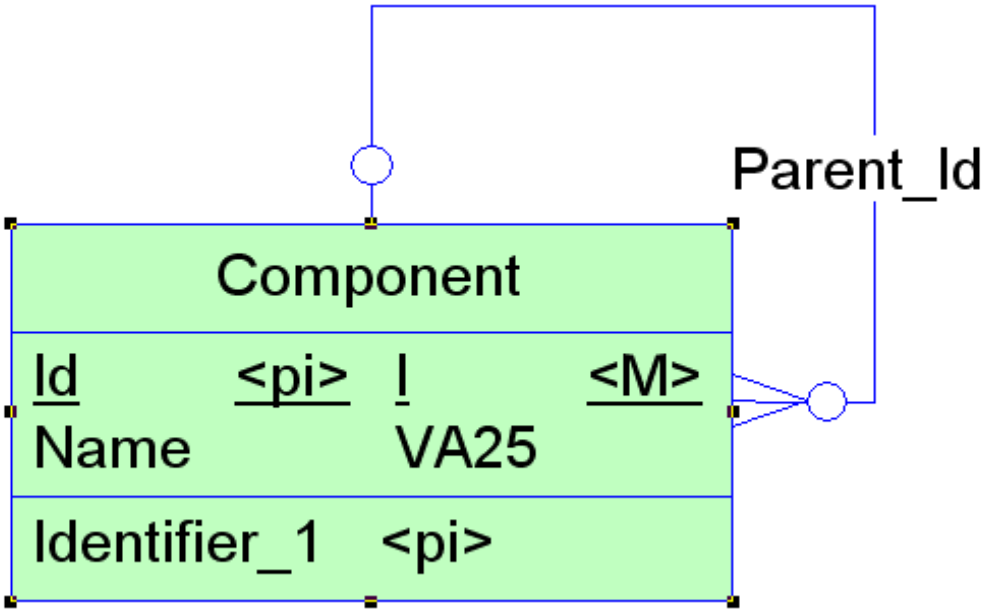
Stromové struktury v relační databázi

Stromové struktury a relační databáze



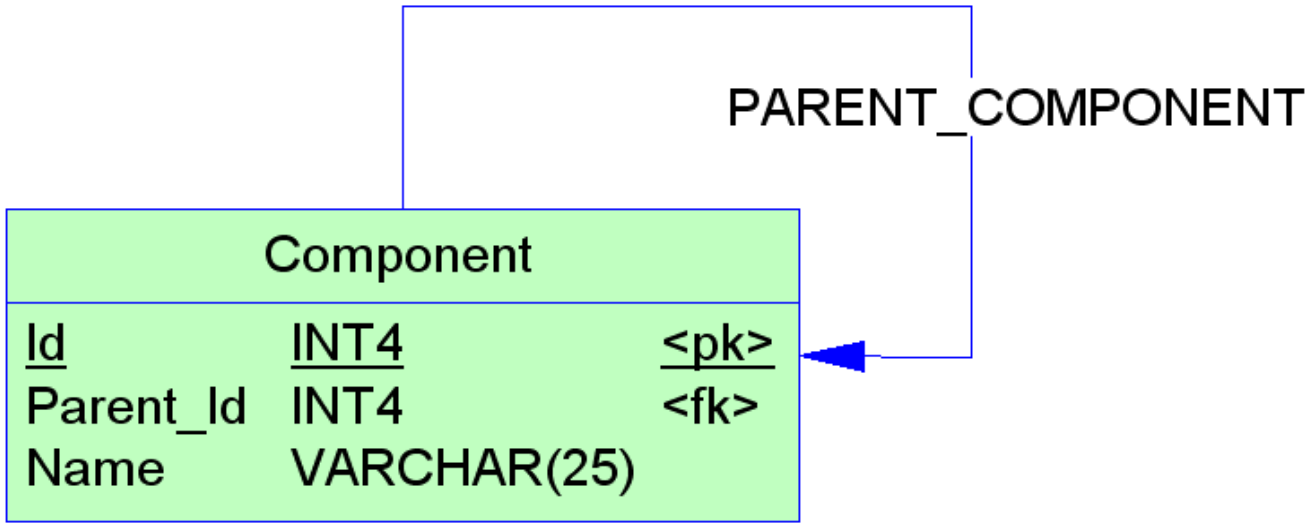
http://interval.cz/clanky/metody-ukladani-stromovych-dat-v-relacnich-databazich/

Stromové struktury a relační databáze



Konceptuální model

Stromové struktury a relační databáze



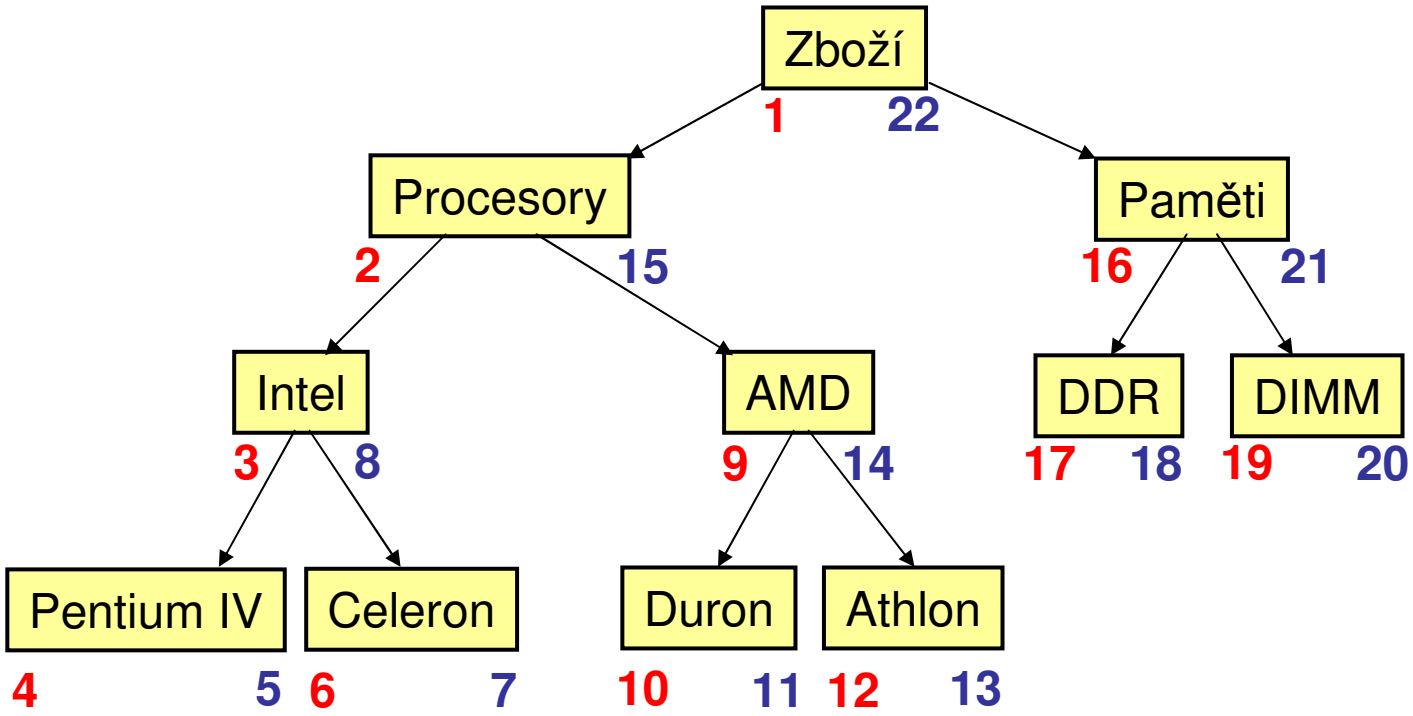
Logický model

Stromové struktury a relační databáze

Hledání všech uzlů z podstromu daného uzlu - rekurze

```
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);
   }
}
```

Stromové struktury a relační databáze



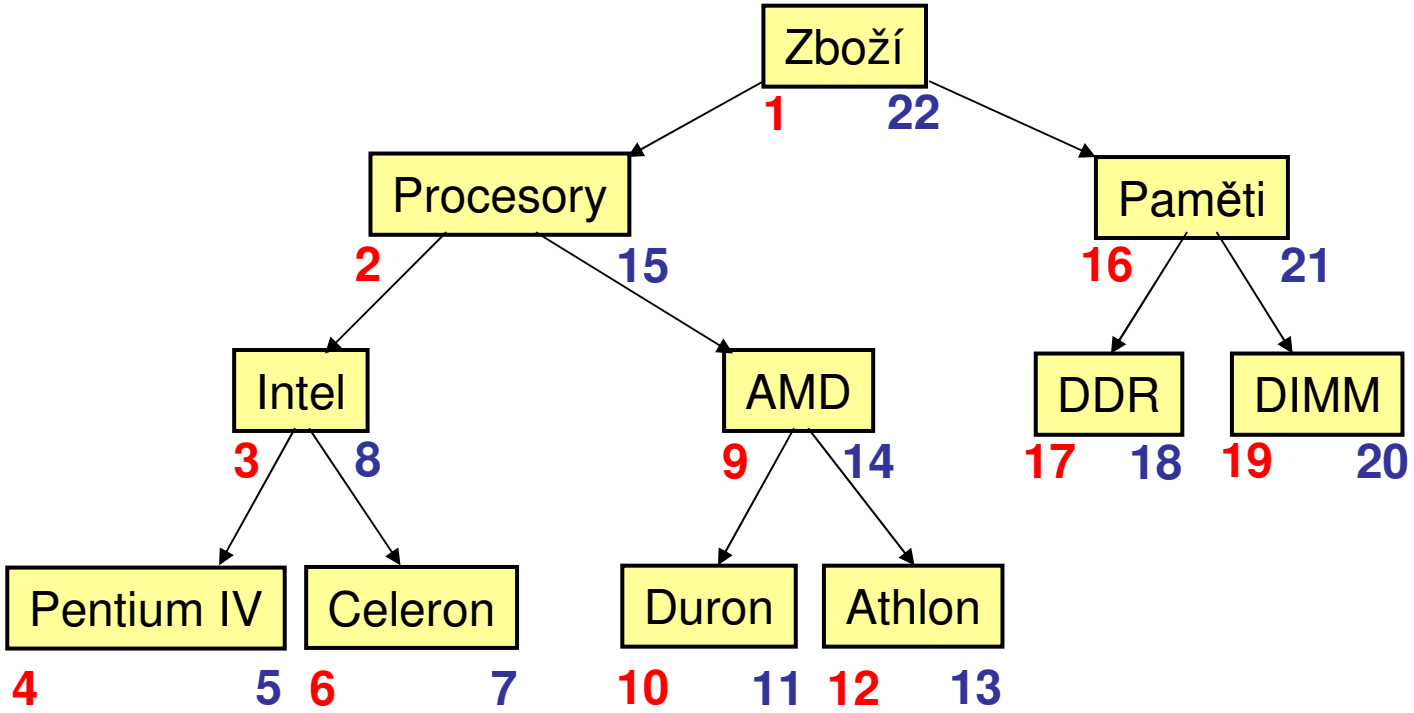
http://interval.cz/clanky/metody-ukladani-stromovych-dat-v-relacnich-databazich/

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

http://interval.cz/clanky/metody-ukladani-stromovych-dat-v-relacnich-databazich/

Stromové struktury a relační databáze



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

Indexace pomocí B-stromů

Indexace jako prostředek optimalizace výkonu

SELECT *
    FROM *PERSON*
    WHERE *(GENDER=FEMALE)* AND *(AGE < 32)*

Dotaz tohoto typu bude vykonán mnohem rychleji, bude-li k dispozici index, jehož indexačním výrazem bude {*GENDER*, *AGE*}.

Jednou z hodnot tohoto indexačního výrazu může být například dvojice <*FEMALE, 27*>.

Teorie idexačních technik (vyhledávacích datových struktur) používá pojem **klíč** namísto pojmu **indexační výraz**. Nezaměňovat s **klíčem tabulky** !

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

Indexace jako prostředek optimalizace výkonu

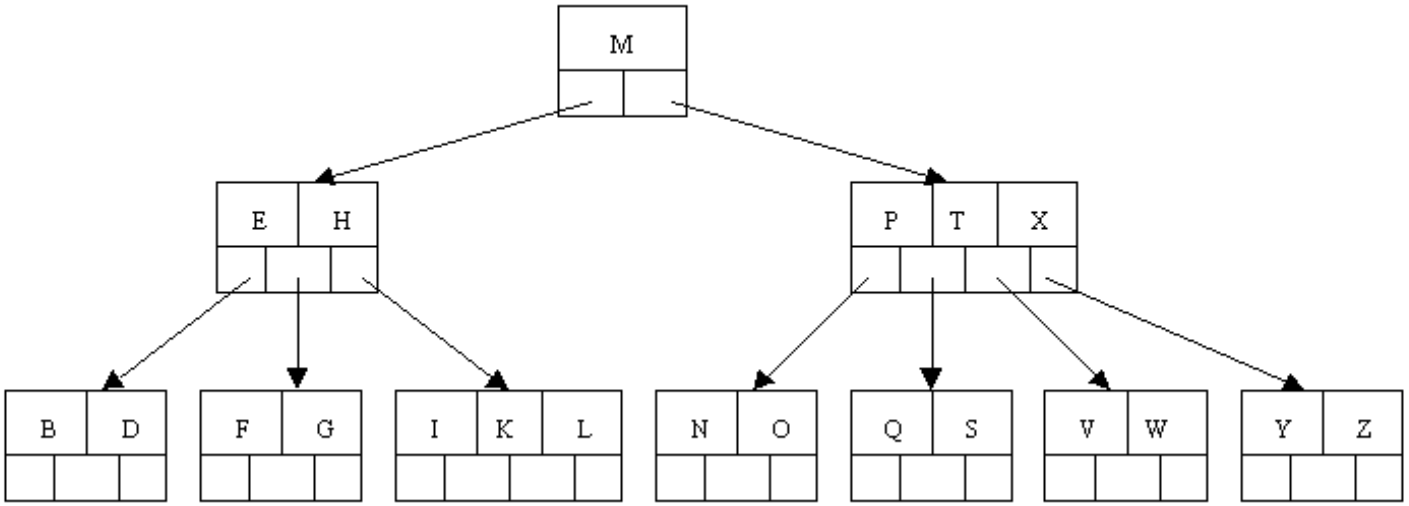
Opatrně s disjunkcemi v podmínkách.

SELECT *
    FROM *PERSON*
    WHERE *(GENDER=FEMALE)* **OR** *(AGE < 32)*

DBMS by měl využít dvou klíčů – a to {*GENDER*} a {*AGE*}

Některé DBMS (např. PostgreSQL) nemusejí pro takovéto dotazy využívat efektivně existující indexy.

Princip B-stromu



B-strom má definovánu:

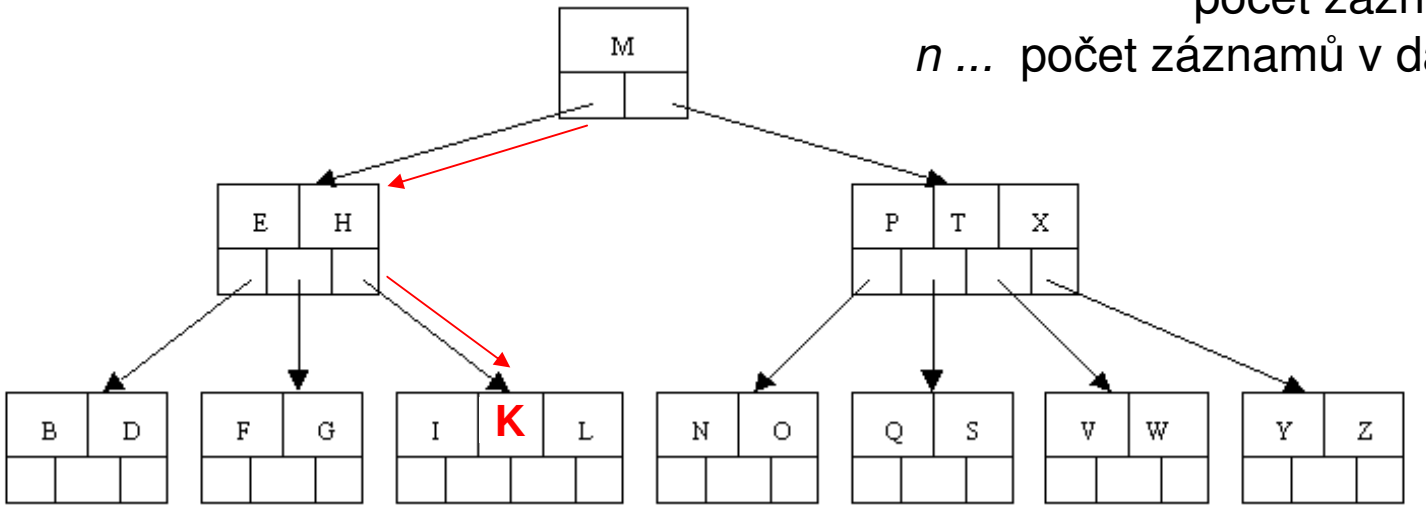
- maximální kapacitu uzlu (max. počet záznamů v uzlu)
- minimální kapacitu uzlu (min. počet záznamů v uzlu)

Záznamy uvnitř uzlu jsou setříděné podle hodnoty klíče.

Princip B-stromu

max(min) … maximální (minimální) počet záznamů v uzlu

n ... počet záznamů v databázi

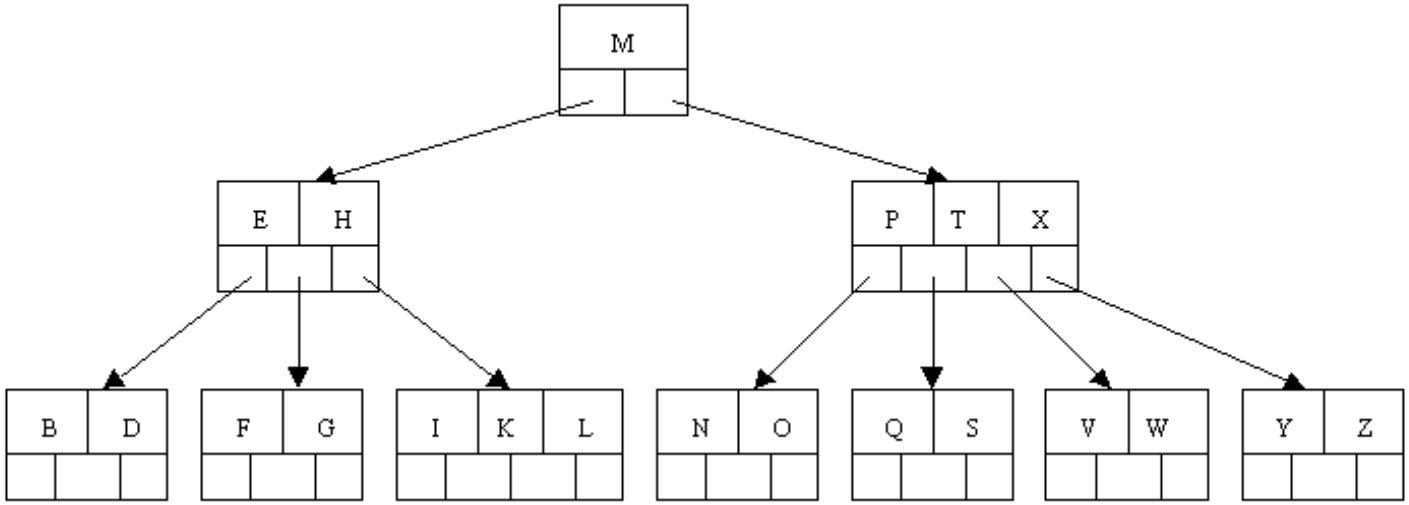


- Každý uzel – stránka v databázi (typicky stránka = sektor)
- Smysl – minimalizace počtu přístupů do databáze
- Hloubka B-stromu

v nejlepším případě (všechny uzly naplněny na 100%) ... $\log_{max} n$

v nejhorším případě (všechny uzly naplněny na *min*) ... $\log_{min} n$

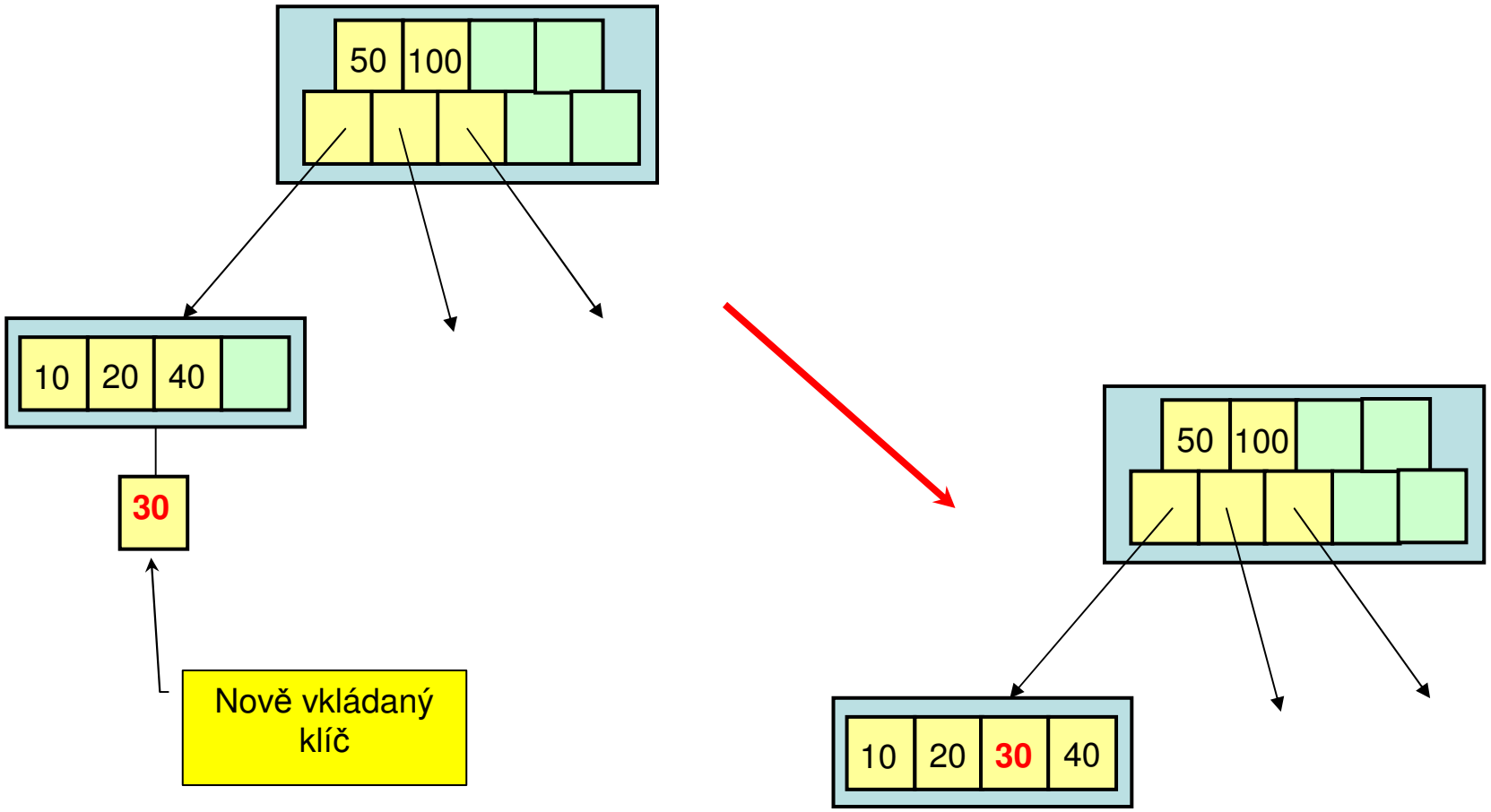
Vkládání do B-stromu



- Každý uzel – stránka v databázi (typicky stránka = sektor)
- Při prvotní konstrukci stromu se uzly naplňují pouze částečně, 25% - 30% kapacity uzly se nechávají volné jako rezerva pro nově vkládané uzly
- Je-li uzel zcela zaplněn a je třeba do něj přidat další záznam, uzel se rozštěpí na 2 zaplněné z 50%. V takovém případě se musí přidat záznam i do příslušného uzlu o patro výš

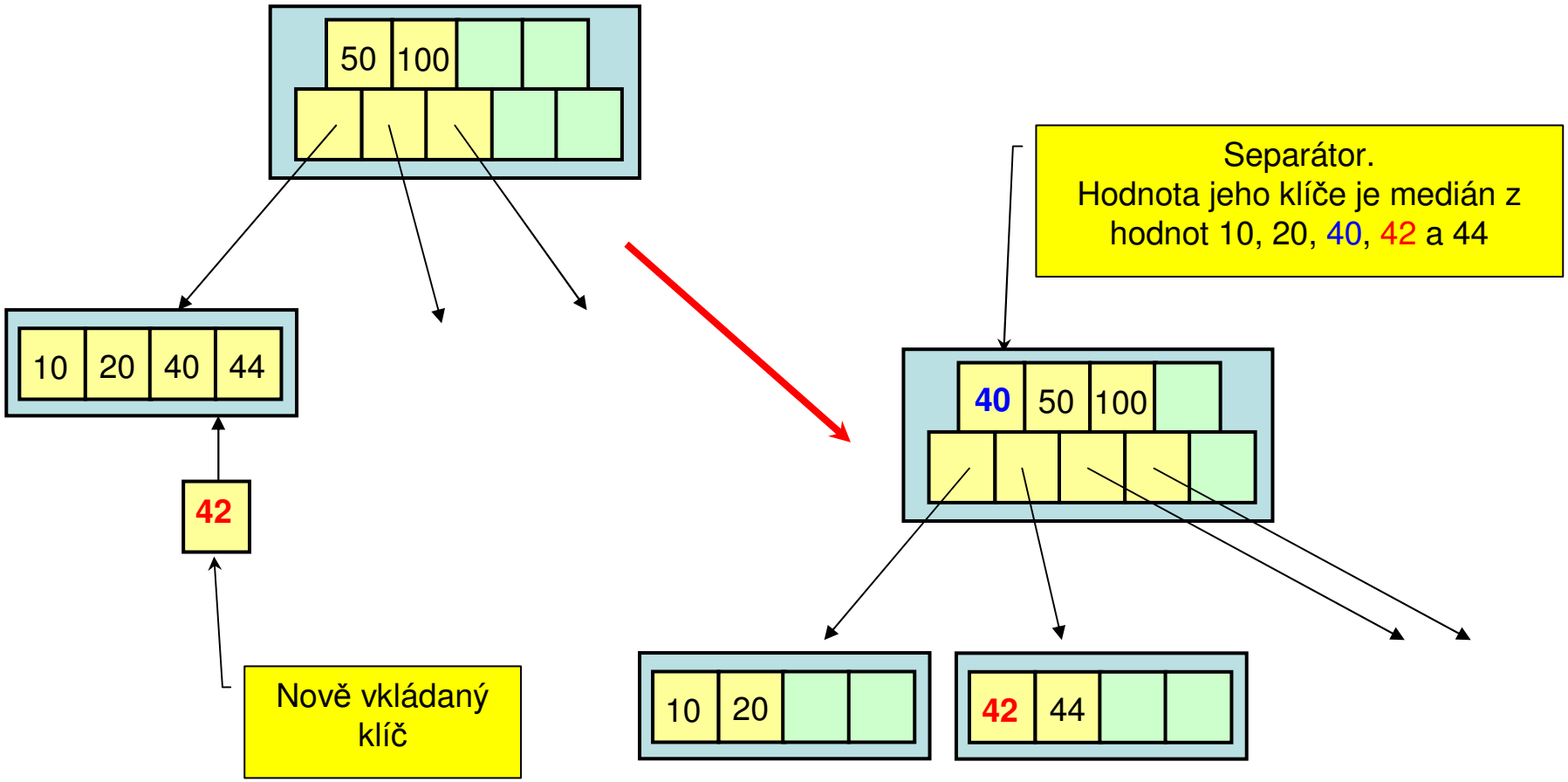
Vkládání záznamu do B-stromu

Triviální, není-li kapacita daného uzlu naplněna



Vkládání záznamu do B-stromu

Je-li kapacita daného uzlu naplněna, musí dojít k rozdělení uzlů:



Vkládání záznamu do B-stromu

Algoritmus:

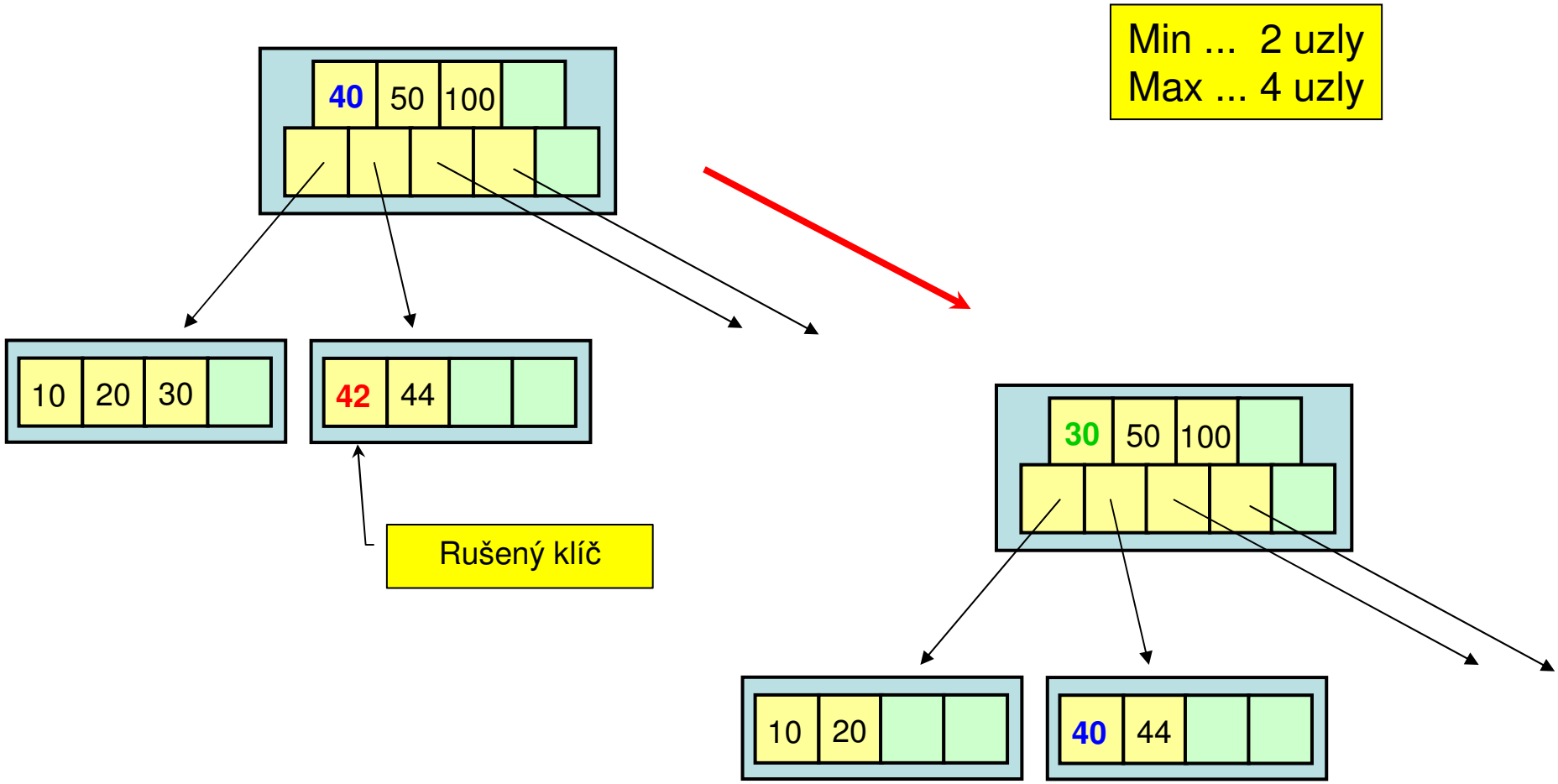
1. Pokud uzel, do něhož máme přidat nový záznam, obsahuje méně záznamů, než je jeho kapacita (maximální počet záznamů, který se „vejde“ do jednoho uzlu), vložíme nový záznam do tohoto uzlu
   při zachování uspořádání záznamů.
2. V opačném případě (uzel je naplněn na svou kapacitu), rozdělíme stávající uzel na dva uzly.
   a. Nalezneme separační záznam jako medián množiny klíčů, která je tvořena klíči záznamů existujících v děleném uzlu plus klíče vkládaného uzlu.
   b. Záznamy s klíči menšími než klíč separačního záznamu necháme v původním uzlu, záznamy s klíčem větším než klíč separačního záznamu uložíme do nového uzlu (zařazeného napravo od původního uzlu).
   c. Separační záznam vložíme do rodičovského uzlu, který se zase může rozlomit, pokud jeho kapacita již byla naplněna. Pokud uzel nemá rodiče (t.j. byl kořenem B-stromu), vytvoříme nový kořen nad tímto uzlem (dojde ke zvětšení hloubky stromu).

Rušení záznamu v listu B-stromu

- Rušení záznamu v **listu** B-stromu je jednodušší než rušení záznamu v nelistovém uzlu.
- Pokud po zrušení záznamu klesne počet záznamů pod minimální kapacitu a sourozenecký uzel má více záznamů, než je minimální kapacita uzlu, **přesuneme** záznam ze sourozeneckého uzlu.
- Klesne-li počet záznamů v obou sourozeneckých uzlech pod minimální kapacitu uzlu, uzly **spojíme**.

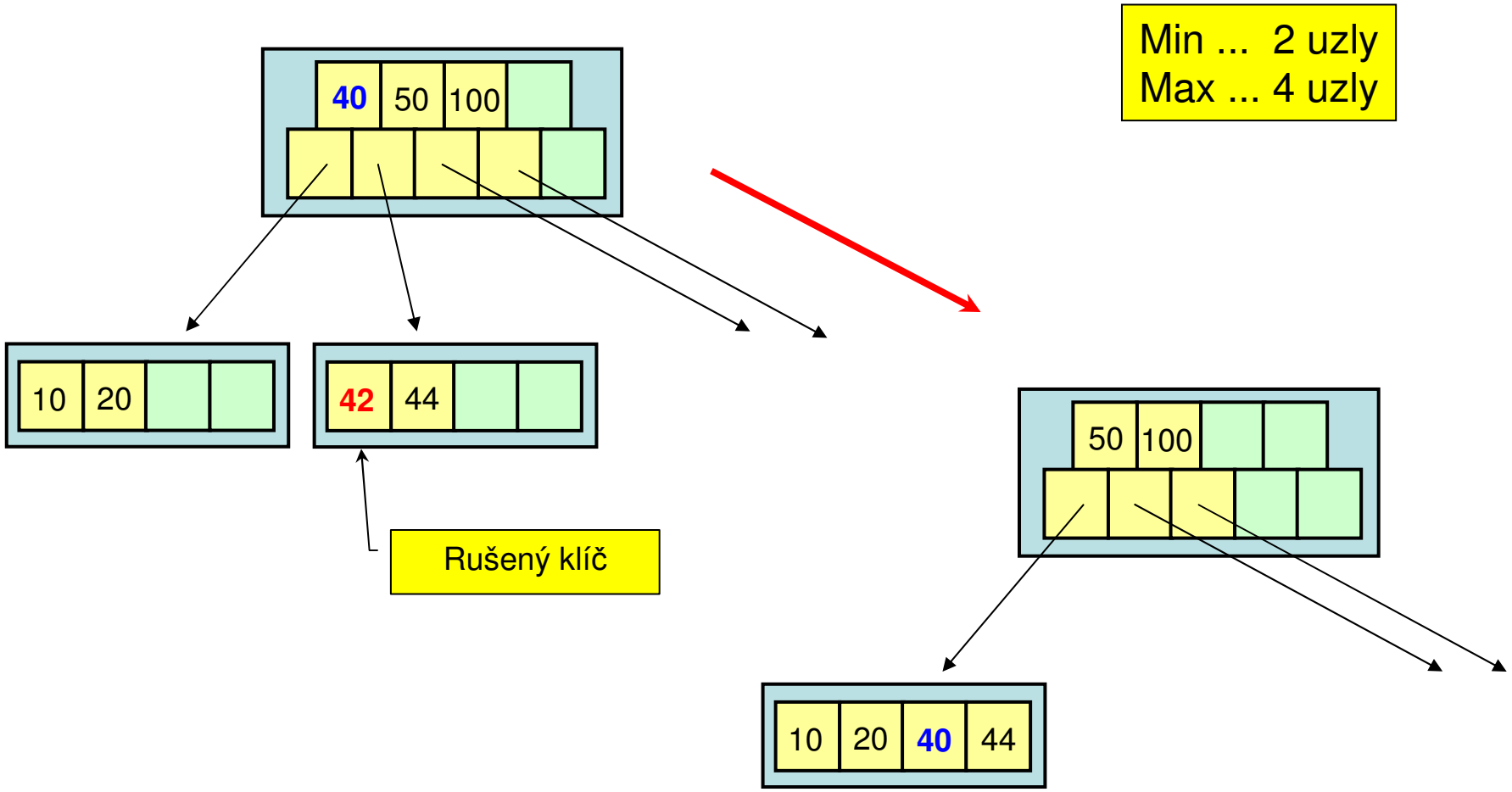
Rušení záznamu v listu B-stromu

Přesun uzlu

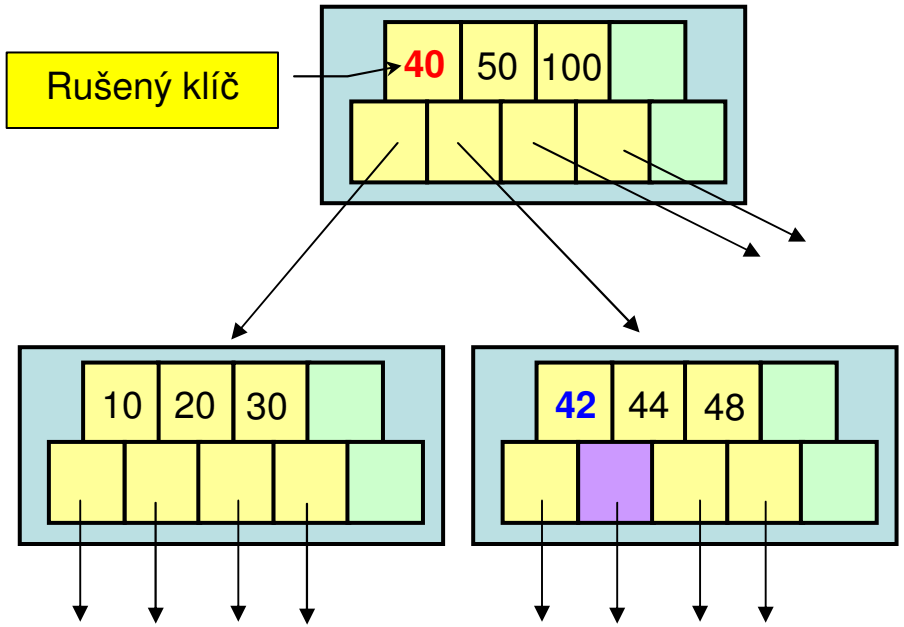


Rušení záznamu v listu B-stromu

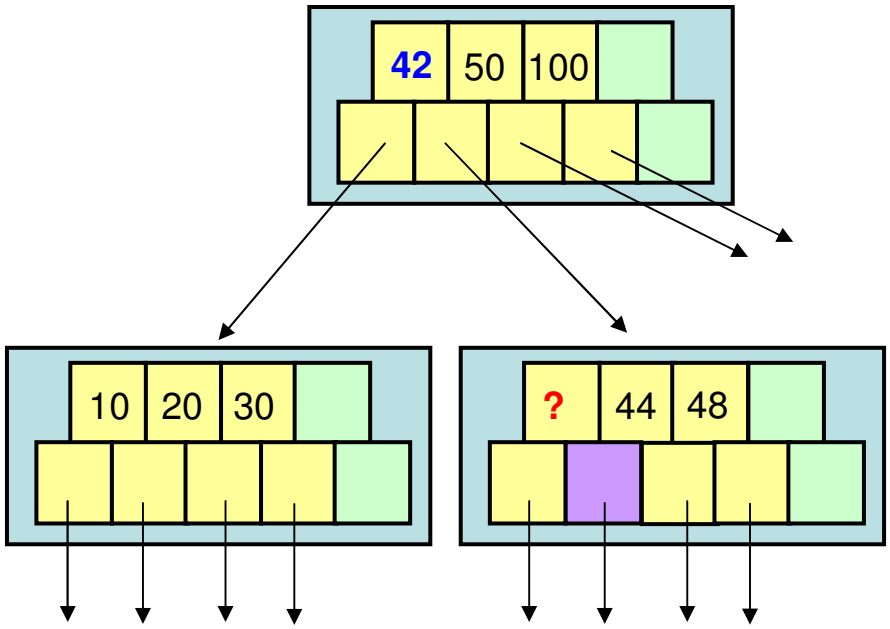
Spojení uzlů



Rušení záznamu v nelistovém uzlu B-stromu



Min ... 2 uzly
Max ... 4 uzly



Klíč označený **?** bude nejmenší klíč **fialového** podstromu.

Může následovat:
- spojení uzlů
- přesun od sourozence

Rušení záznamu v nelistovém uzlu B-stromu

- Tento přístup znamená, že při rušení uzlu smažeme rušený klíč a poté uvádíme strom znovu do vyváženého stavu.
- Není to jediný možný algoritmus, existují i jiné.