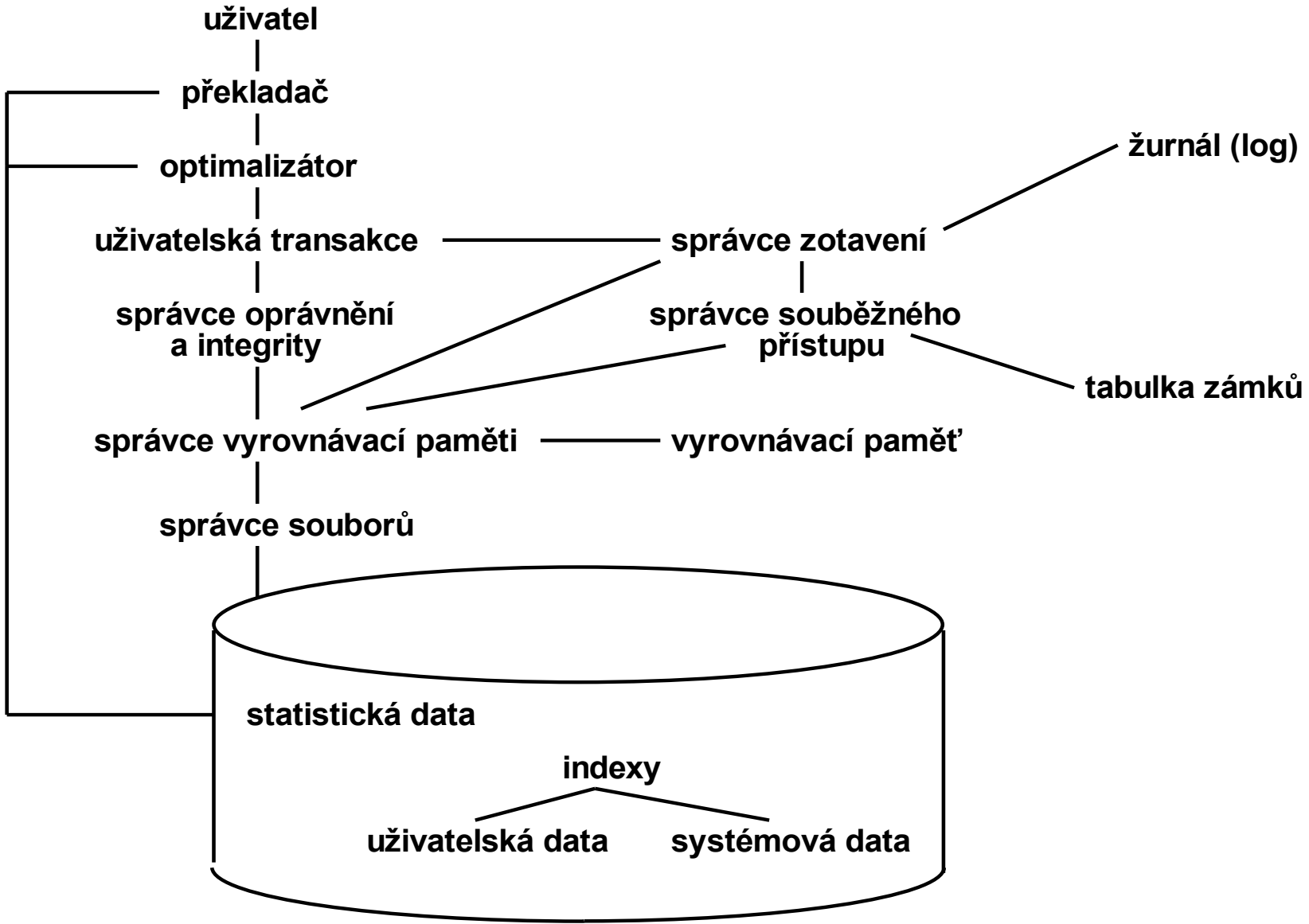
9. Fyzická (interní) úroveň databázového systému

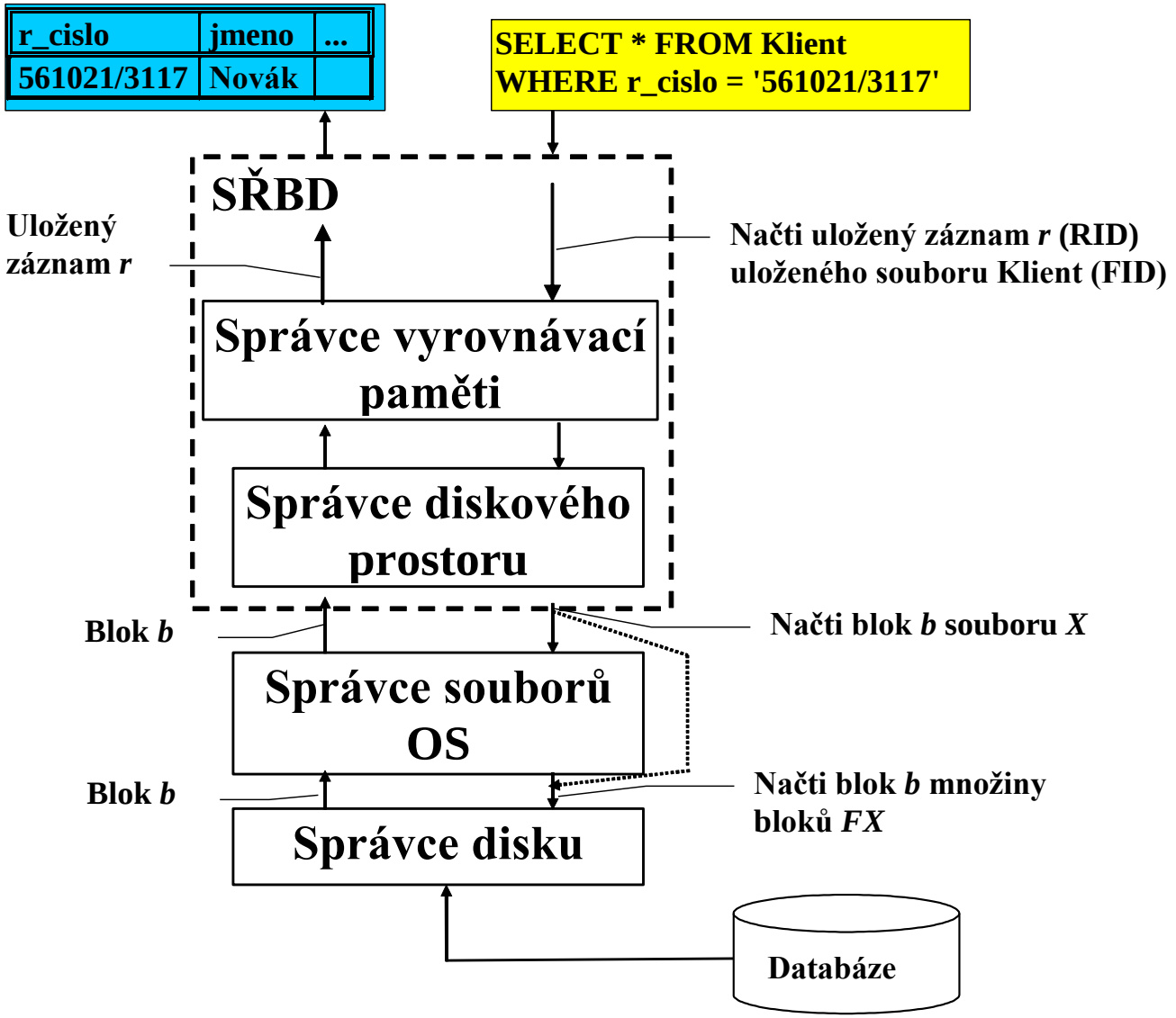
Ing. Vladimír Bartík, Ph.D.
RNDr. Marek Rychlý, Ph.D.

Osnova

9.1. Struktura databázového systému



9.2. Přístup k datům v databázi

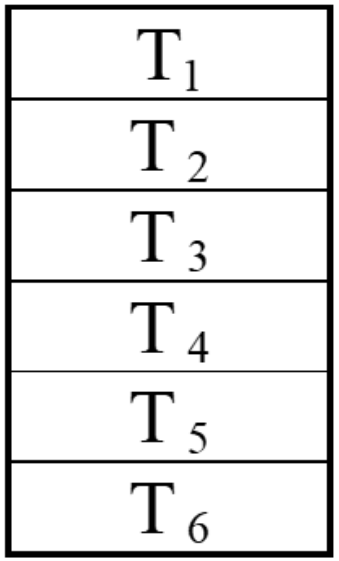


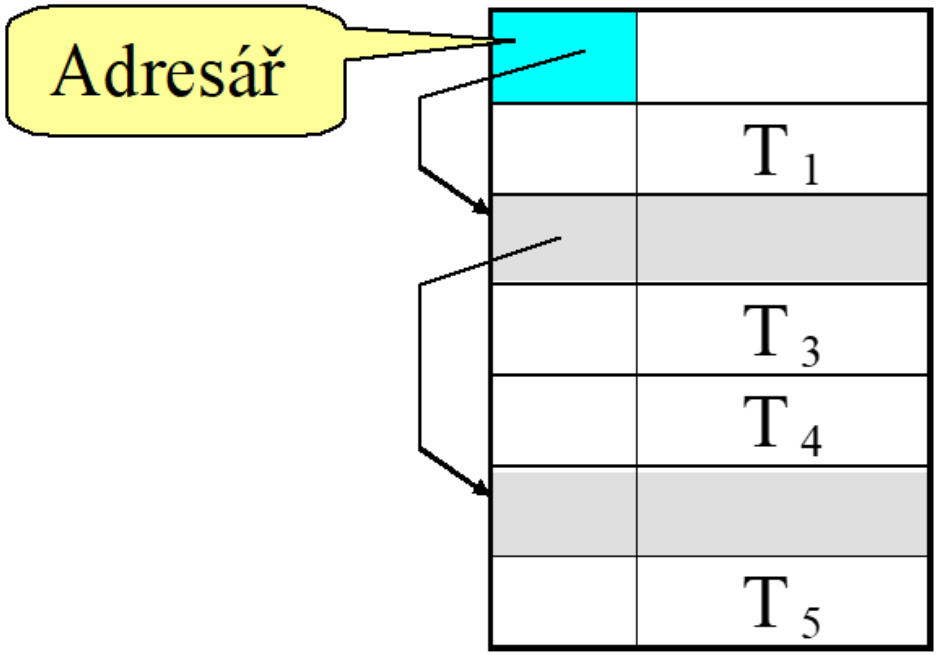
- hierarchie pamětí:
  - vnitřní
  - sekundární (disk)
  - terciální (mag. páska)

Cíl: **minimalizace přístupů k disku.**

9.3. Struktura souborů

- ***Soubor*** **je posloupnost záznamů seskupovaných do bloků tvořících logický celek.**
- **Organizace záznamů**
  - **záznamy pevné délky**





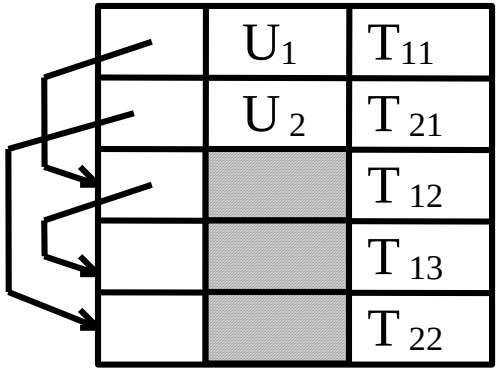
9.3. Struktura souborů

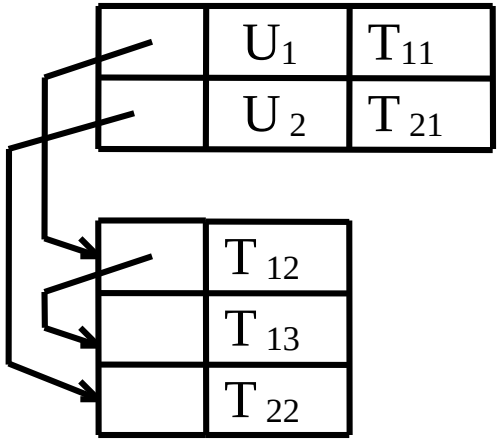
- **Organizace záznamů (pokračování)**
  - **záznamy proměnné délky**

Př.) Shlukované záznamy tabulek Ucet a Transakce

U_1	T_{11}	T_{12}	T_{13}	⊥
U_2	T_{21}	T_{22}	⊥	

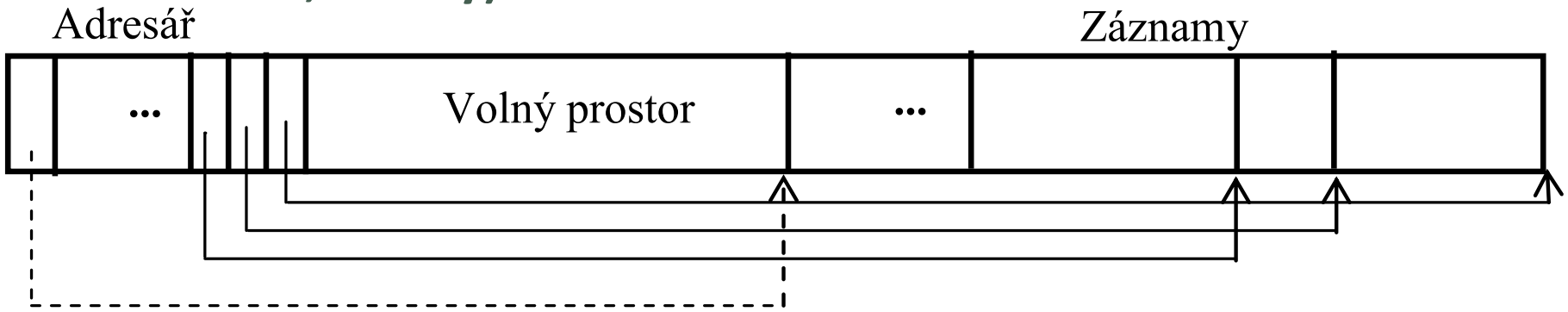
U_1	T_{11}	T_{12}	T_{13}	⊥
U_2	T_{21}	T_{22}	⊥	⊥



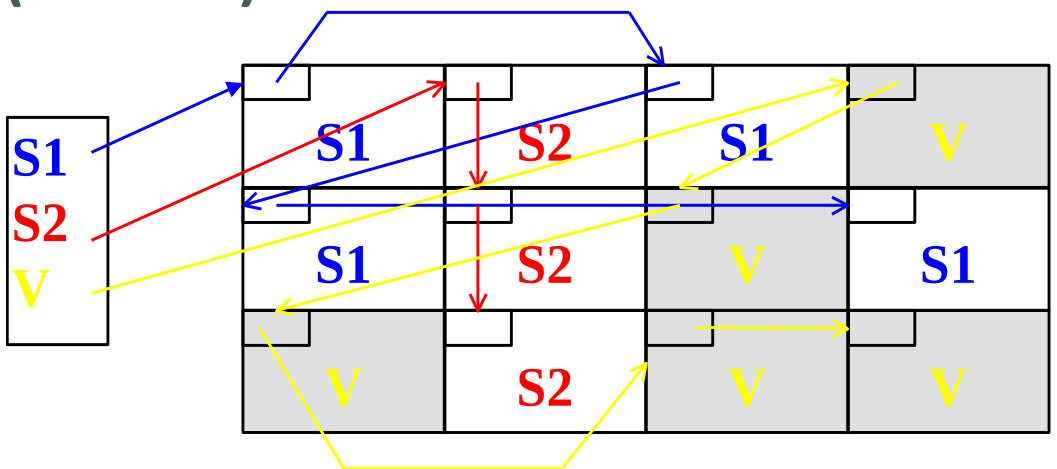


9.3. Struktura souborů

- **Organizace záznamů (pokračování)**
  - **struktura s adresářem bloku – je žádoucí mít pevnou adresu záznamu (viz identifikace RID, indexy)**



- **Organizace bloků (stránek)**

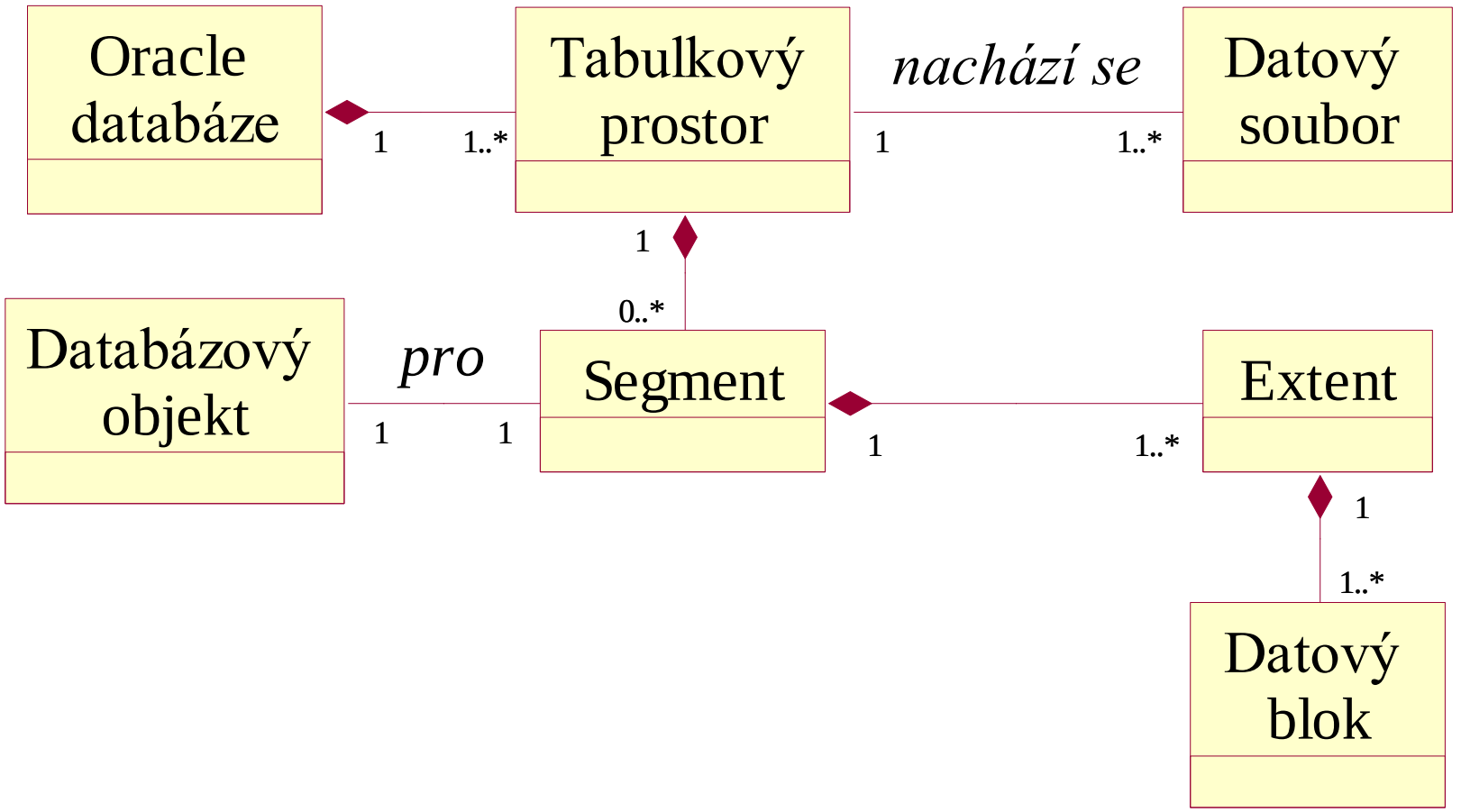


9.3. Struktura souborů

- **Zařazování záznamů do bloků souboru**
  - **Neuspořádaný soubor (heap organization)** – záznam je umístěn tam, kde je místo, neexistuje žádné uspořádání záznamů.
  - **Sekvenční soubor** – záznamy jsou uspořádány podle hodnoty tzv. ***vyhledávacího klíče*** (search key). V případě takového uspořádání, kdy jsou logicky svázané záznamy umístěny fyzicky blízko u sebe, hovoříme o ***shlukování*** (clustering). Klíč pro shlukování se pak označuje jako ***shlukovací klíč*** (cluster key).
  - **Hašovaný soubor** – záznamy jsou umísťovány do bloků na základě hodnoty ***hašovací funkce***.
- **Vztah databáze a souboru operačního systému**
  - Každá tabulka (příp. indexy) v samostatném souboru OS. Typické pro databáze systémů s architekturou PC file-server (např. dBase), ale používají i některé současné systémy, např. lze u MySQL.
  - Celá databáze (typicky obsahujících řadu schémat) v jednom nebo několika souborech OS (např. Oracle).

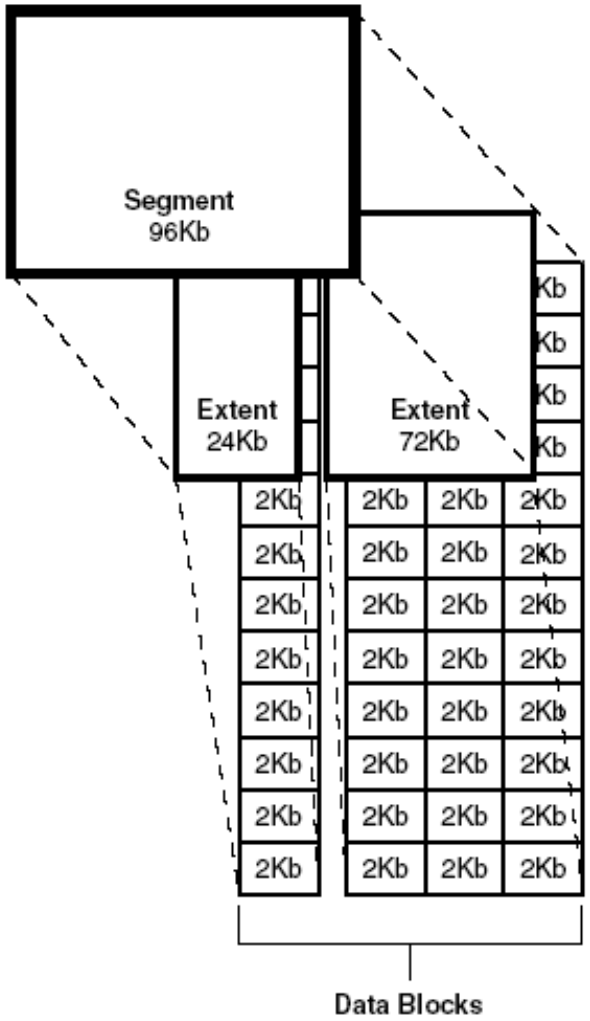
9.3. Struktura souborů

- **Databázové soubory serveru Oracle 10g**



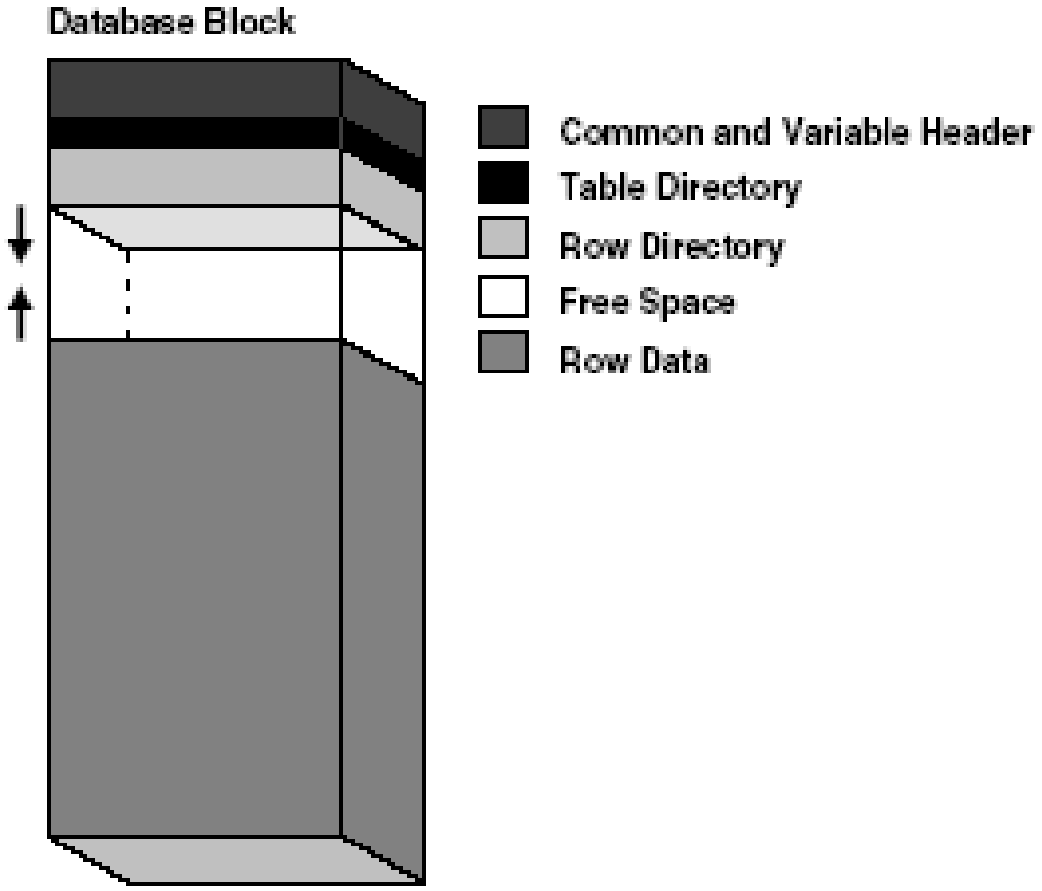
9.3. Struktura souborů

- **Databázové soubory serveru Oracle 10g – segmenty, extenty a datové bloky**



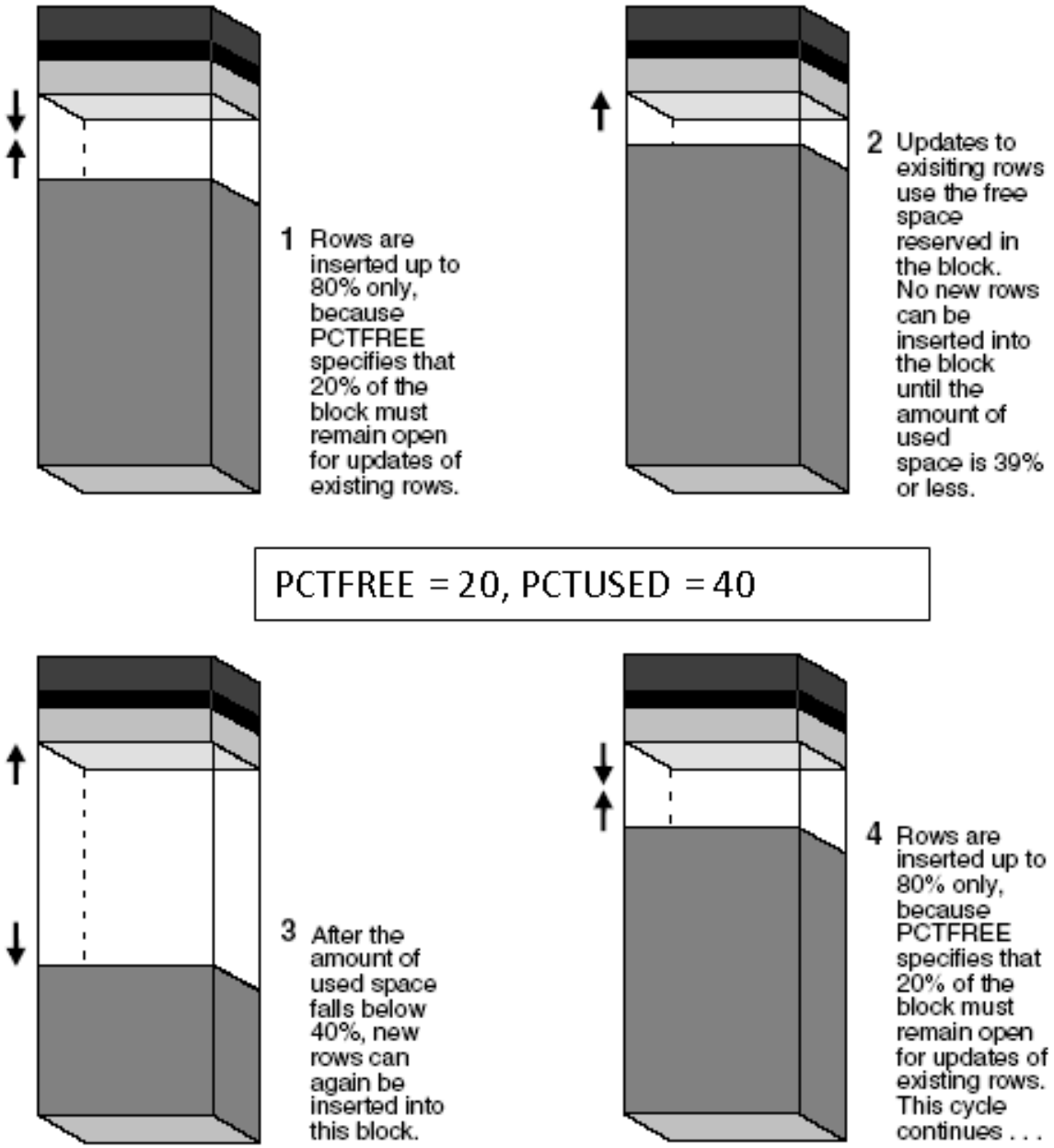
9.3. Struktura souborů

- **Databázové soubory serveru Oracle 10g – struktura datového bloku**



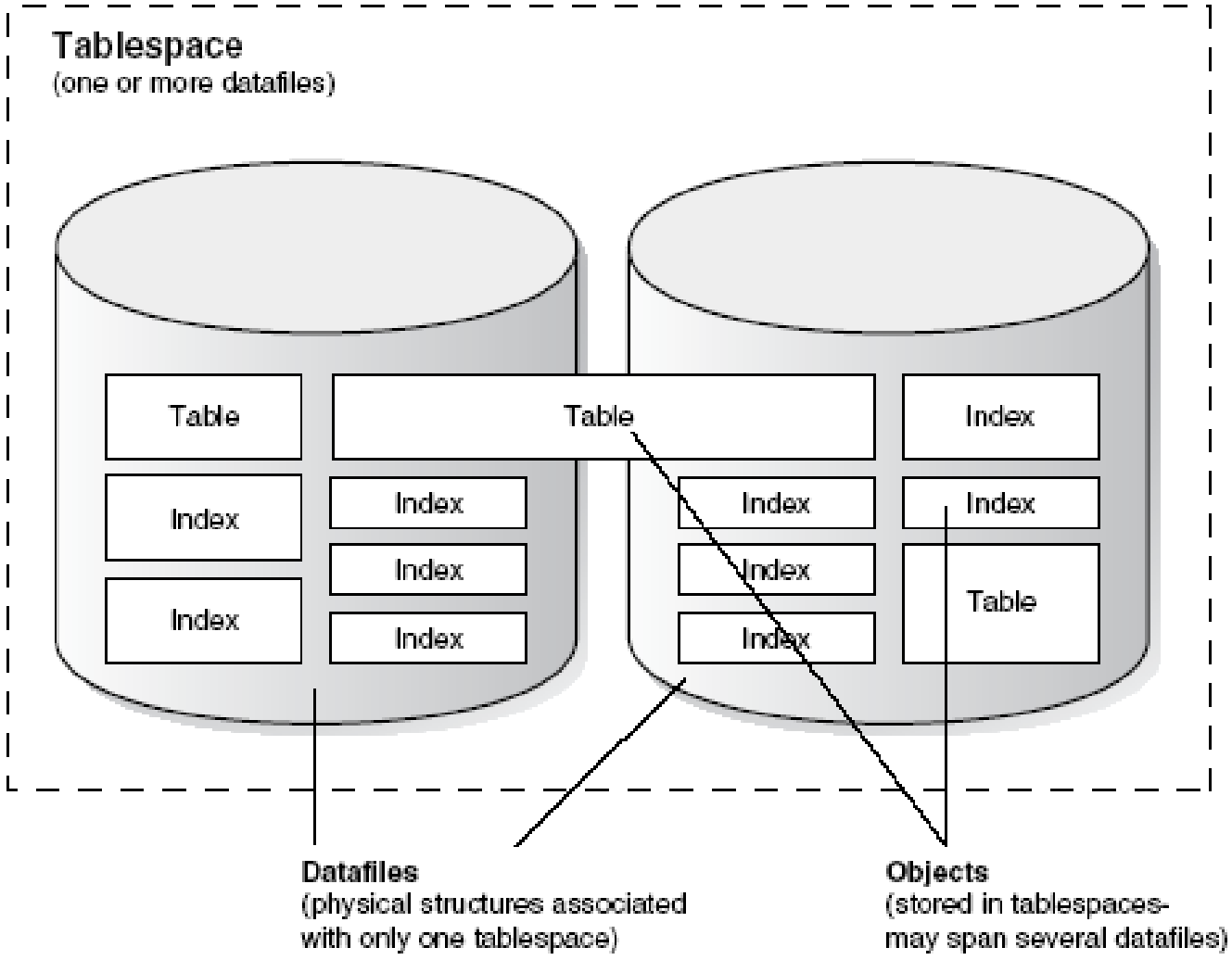
9.3. Struktura souborů

- **Databázové soubory serveru Oracle 10g – zaplňování a uvolňování datového bloku, parametry PCTFREE a PCTUSED**



9.3. Struktura souborů

- **Databázové soubory serveru Oracle 10g – tabulkové prostory**



9.3. Struktura souborů

- **Databázové soubory serveru Oracle 10g**

  Př.) Vytvoření tabulky s parametry fyzického uložení

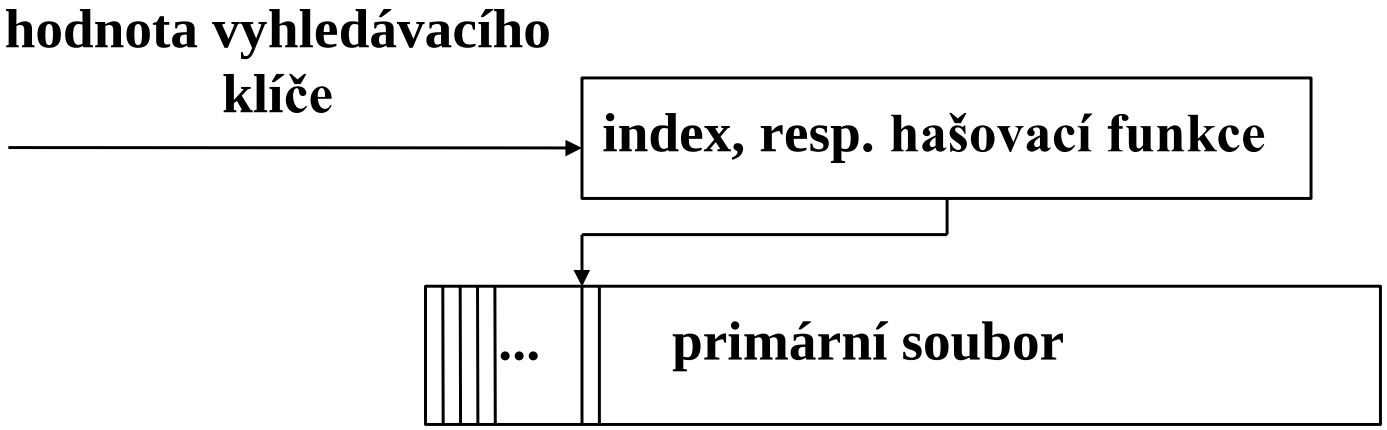
```
CREATE TABLE Klient (...)
   TABLESPACE users
   STORAGE (INITIAL 200K
      NEXT 300K
      MINEXTENTS 2
      MAXEXTENTS 20
      PCTINCREASE 33);
```

9.4. Správa vyrovnávací paměti

- **Požadavky**
  - **co nejlepší strategie výměny bloků (zpravidla LRU, ale mohla by být výhodnější MRU)**
  - **omezení času, kdy nelze blok zapsat na disk (princip WAL – viz transakční zpracování)**
  - **možnost vynuceného zápisu modifikovaných bloků na disk (kontrolní body – viz transakční zpracování)**
  - **preference často používaných bloků (katalogu, indexů).**
- **Možnosti řešení**
  - **vyrovnávací paměť spravovaná operačním systémem (OS),**
  - **vyrovnávací paměť spravovaná SŘBD.**
- **Problémy vyrovnávací paměti ve správě OS**

  ⇒ vyrovnávací paměť spravovaná SŘBD

9.5. Podstata indexování a hašování

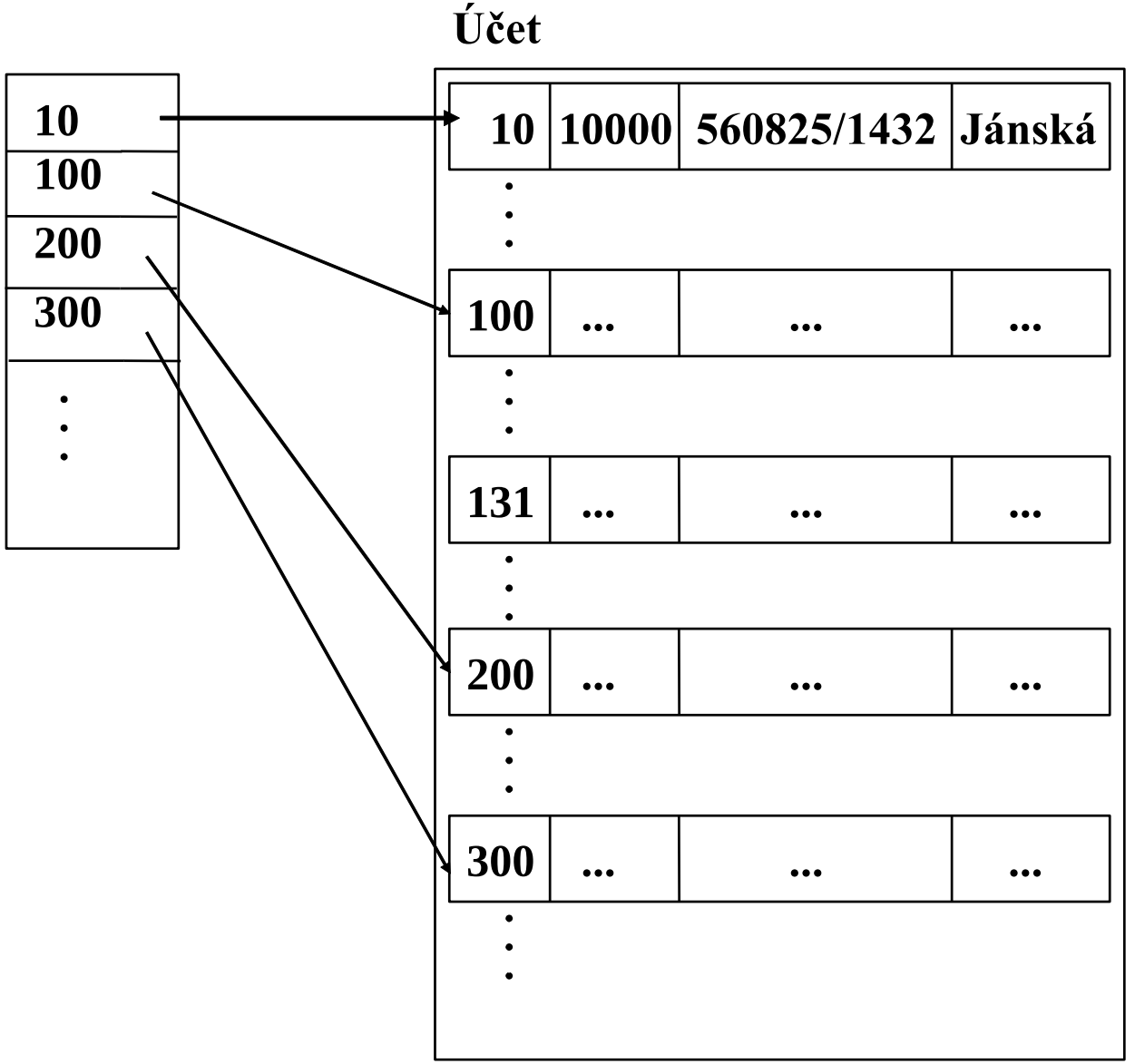


- **Primární soubor** – soubor se záznamy tabulky.
- **Vyhledávací klíč** – sloupec (případně složený) tabulky, prostřednictvím jehož hodnoty budeme přistupovat k požadovaným záznamům.
- **Primární vyhledávací klíč** – vyhledávací klíč, podle jehož hodnot je setříděn primární soubor.
- **Sekundární vyhledávací klíč** – vyhledávací klíč, který není primární.

9.5. Podstata indexování a hašování

- **Uspořádaný index** – založený na seřazení podle hodnot vyhledávacího klíče.
- **Hašovaný index** – založený na hašování. Záznamy primárního souboru (přímé hašování) nebo ukazatele na záznam (nepřímé hašování) se nachází v sektorech (bucket) (typicky o velikosti násobku diskového bloku). Sektor pro záznam s danou hodnotou vyhledávacího klíče nebo pro ukazatele na takový záznam je určen hašovací funkcí.

9.6. Uspořádané indexy

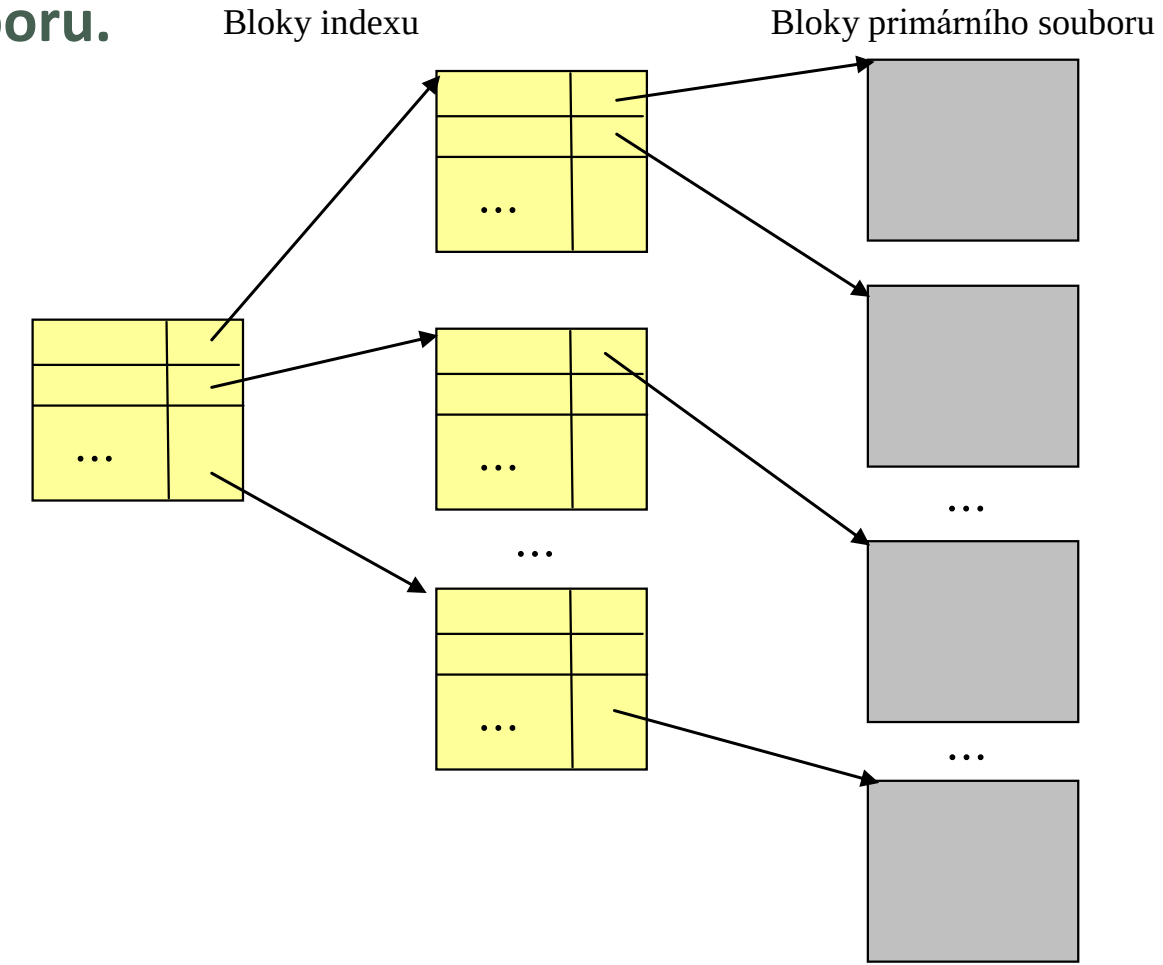


9.6. Uspořádané indexy

- **Primární index** (také shlukující) – index pro prim. vyhledávací klíč.
- **Sekundární index** – index pro sekundární vyhledávací klíč.
- **Indexsekvenční soubor** - uspořádaný primární soubor + primární index.
- **Položka indexu** – záznam obsahující hodnotu vyhledávacího klíče a ukazatele na jeden nebo několik záznamů primárního souboru s touto hodnotou primárního klíče, případně ukazatel na sektor ukazatelů.
- **Hustý index** – index s položkami obsahujícími všechny hodnoty vyhledávacího klíče vyskytující se v primárním souboru.
- **Řídký index** – index s položkami obsahujícími jen některé hodnoty vyhledávacího klíče vyskytující se v primárním souboru. Primární řídký index typicky obsahuje jednu položku pro každý blok primárního souboru.
- **Index s unikátními hodnotami (unique index)** – index pro vyhledávací klíč, jehož hodnoty jsou v primárním souboru unikátní.

9.6. Uspořádané indexy

- **Víceúrovňový index – index, u kterého je každá úroveň, s výjimkou poslední, řídkým primárním indexem úrovně následující. Poslední úroveň je jednoúrovňovým indexem primárního souboru.**



9.7. B+ strom (Bayer)

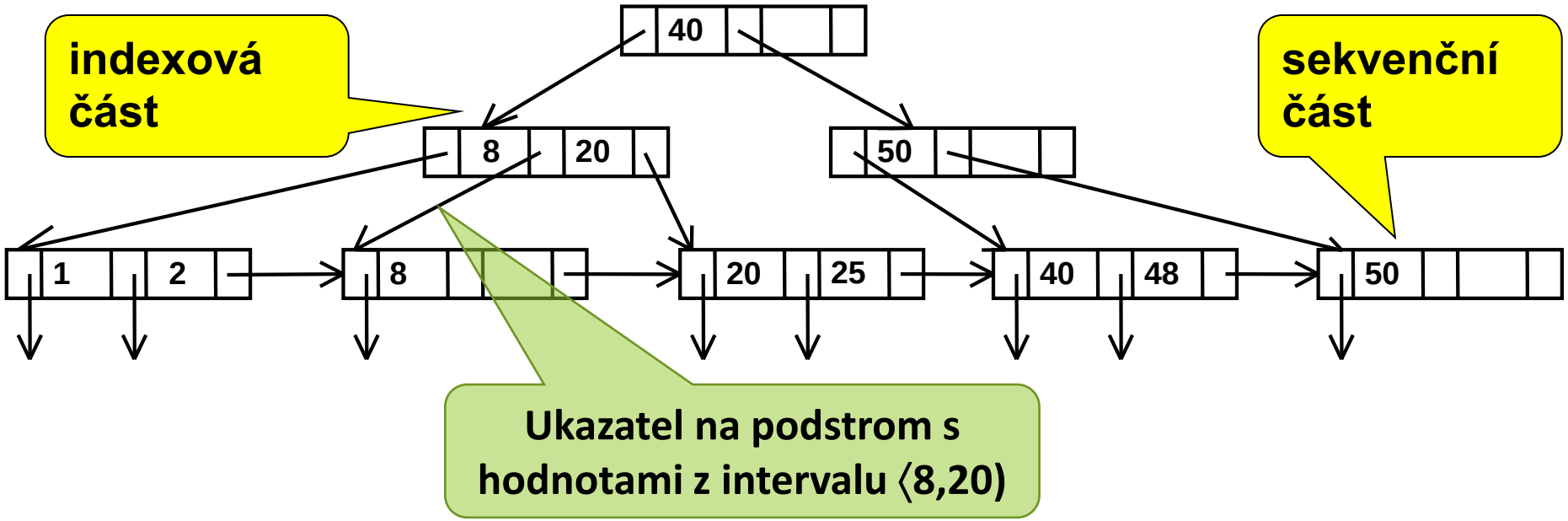
- **Nejpoužívanější indexová struktura v relačních databázových systémech**
- **Struktura B+ stromu**
  - **víceúrovňový index ve tvaru vyváženého n-nárního stromu, kde n je maximální počet následníků**
  - **každý uzel s výjimkou kořene a listu má $\lceil n/2 \rceil$ až n následníků,**
  - **kořen, není-li současně listem, má nejméně 2 následníky**
  - **list obsahuje $\lceil (n-1)/2 \rceil$ až n-1 hodnot vyhledávacího klíče**
  - **listové uzly tvoří jednosměrný uspořádaný seznam**
  - **uzel stromu má typicky velikost jednoho diskového bloku**
  - **pro K hodnot vyhledávacího klíče není cesta od kořene k listu delší než $\left\lceil \log_{\lceil n/2 \rceil} (K) \right\rceil$**

Pozn.: Výraz $\lceil x \rceil$ značí nejmenší celé číslo větší nebo rovno x a $\lfloor x \rfloor$ značí největší celé číslo menší nebo rovno x.

9.7. B+ strom (Bayer)

- **Typický tvar uzlu**

Př.) Struktura B+ stromu n=3



9.7. B+ strom (Bayer)

- **Operace na B+ stromu (uvažujeme pro jednoduchost unikátní index) - vyhledání**

```
Nastav kořen jako aktuální uzel;
WHILE aktuální uzel není listem DO
BEGIN
   Najdi nejmenší hodnotu vyhledávacího klíče Ki v uzlu, která je větší
   než hledaná hodnota;
   IF existuje THEN
      Nastav jako aktuální uzel, na který ukazuje Pi
   ELSE
      Nastav jako aktuální uzel, na který ukazuje poslední ukazatel v
      aktuálním uzlu
END
IF existuje v aktuálním uzlu hodnota Ki rovná hledané hodnotě THEN
   Ukazatel Pi ukazuje na hledaný záznam
ELSE
   Záznam s hledanou hodnotou neexistuje
```

9.7. B+ strom (Bayer)

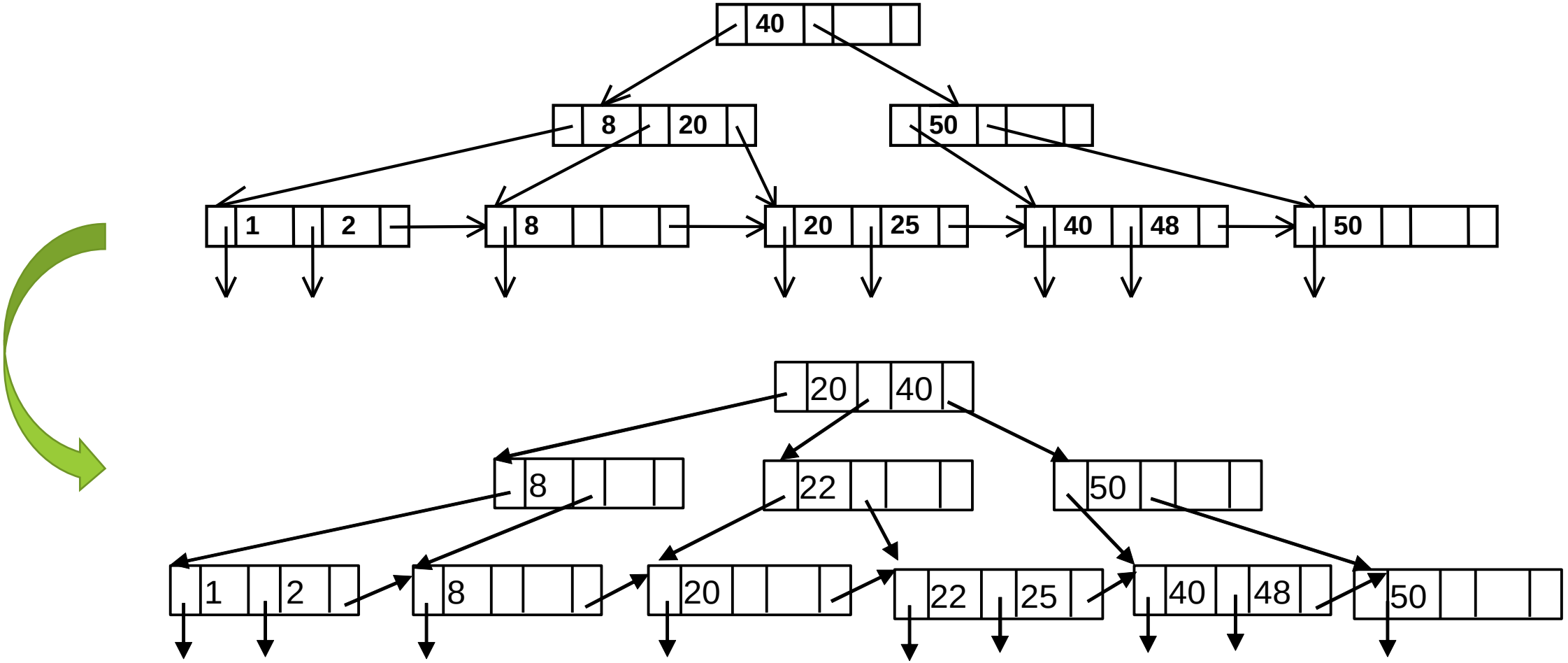
- **Operace na B+ stromu – vkládání**

```
Najdi listový uzel U, který by měl obsahovat novou položku indexu;
IF je v uzlu U místo pro vložení položky THEN
  Vlož položku indexu do uzlu U
ELSE
BEGIN /* Rozštěpení uzlu */
  Alokuj nový uzel U' indexu;
  Rozděl hodnoty štěpeného uzlu U včetně vkládané hodnoty na dvě „stejně
  velké" skupiny. Jedna zůstane v uzlu U a druhá se přesune do uzlu U';
  REPEAT
     Rekurzivně pokračuj s vkládáním nové položky indexu do uzlu
     předchůdce ve stromu s první hodnotou vyhledávacího klíče uzlu U' a
     ukazatelem na tento uzel
  UNTIL nedošlo k dalšímu štěpení nebo byl rozštěpen kořen
END;
IF byl rozštěpen kořen THEN
  Vytvoř nový kořen s ukazateli na dva uzly vzniklé rozštěpením původního
  kořene;
```

9.7. B+ strom (Bayer)

- **Operace na B+ stromu – vkládání**

  Př.) Vložení hodnoty 22



9.7. B+ strom (Bayer)

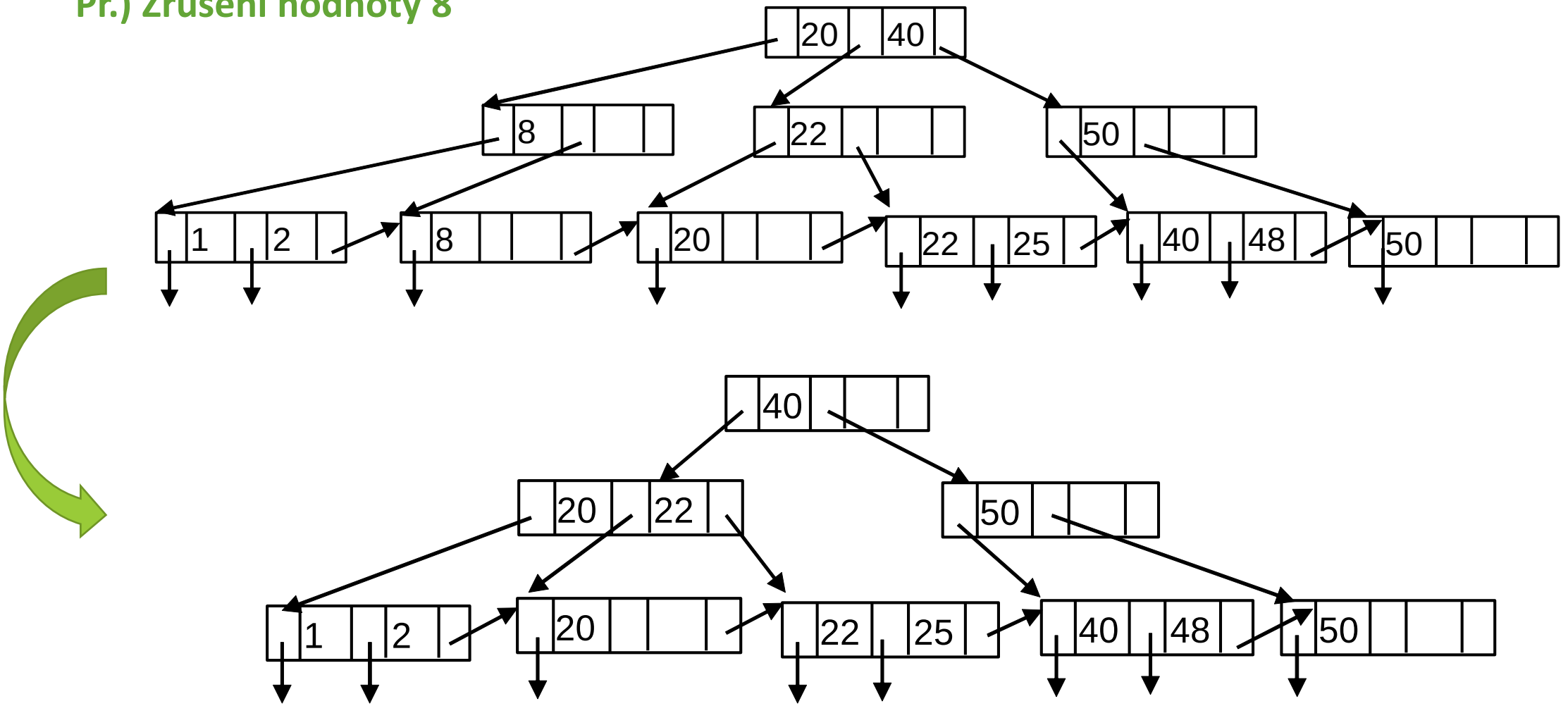
- **Operace na B+ stromu – rušení**

```
Najdi listový uzel U, který obsahuje rušenou položku indexu;
Zruš položku indexu v uzlu U;
IF kleslo zaplnění uzlu U pod ⌈(n-1)/2⌉ THEN
  IF lze přesunout obsah do levého nebo pravého souseda uzlu U THEN
  BEGIN /* slévání uzlů */
    Přesuň obsah uzlu U a zruš prázdný uzel U;
    REPEAT
      Rekurzivně pokračuj s rušením položky indexu s ukazatelem na uzel U v
      uzlu předchůdce ve stromu
    UNTIL nedošlo k dalšímu slévání nebo bylo dosaženo kořene;
    IF kořen má jen jednoho následníka THEN
      Zruš kořen, novým kořenem se stane následník
  END
  ELSE
  BEGIN /* redistribuce hodnot */
    Redistribuuj hodnoty a ukazatele sousedních uzlů tak, aby oba splňovaly
    podmínku minimálního zaplnění;
    Aktualizuj hodnotu vyhledávacího klíče v uzlu předchůdce ve stromu
  END
```

9.7. B+ strom (Bayer)

- **Operace na B+ stromu – rušení**

  Př.) Zrušení hodnoty 8



9.7. B+ strom (Bayer)

Př.) Uvažujme index pro primární klíč tabulky, která má 15000 řádků, ve tvaru B+ stromu. Blok primárního souboru obsahuje až 20 záznamů tabulky.

Jaký bude počet bloků a úrovní indexu, obsahuje-li blok indexu maximálně 50 hodnot primárního klíče?

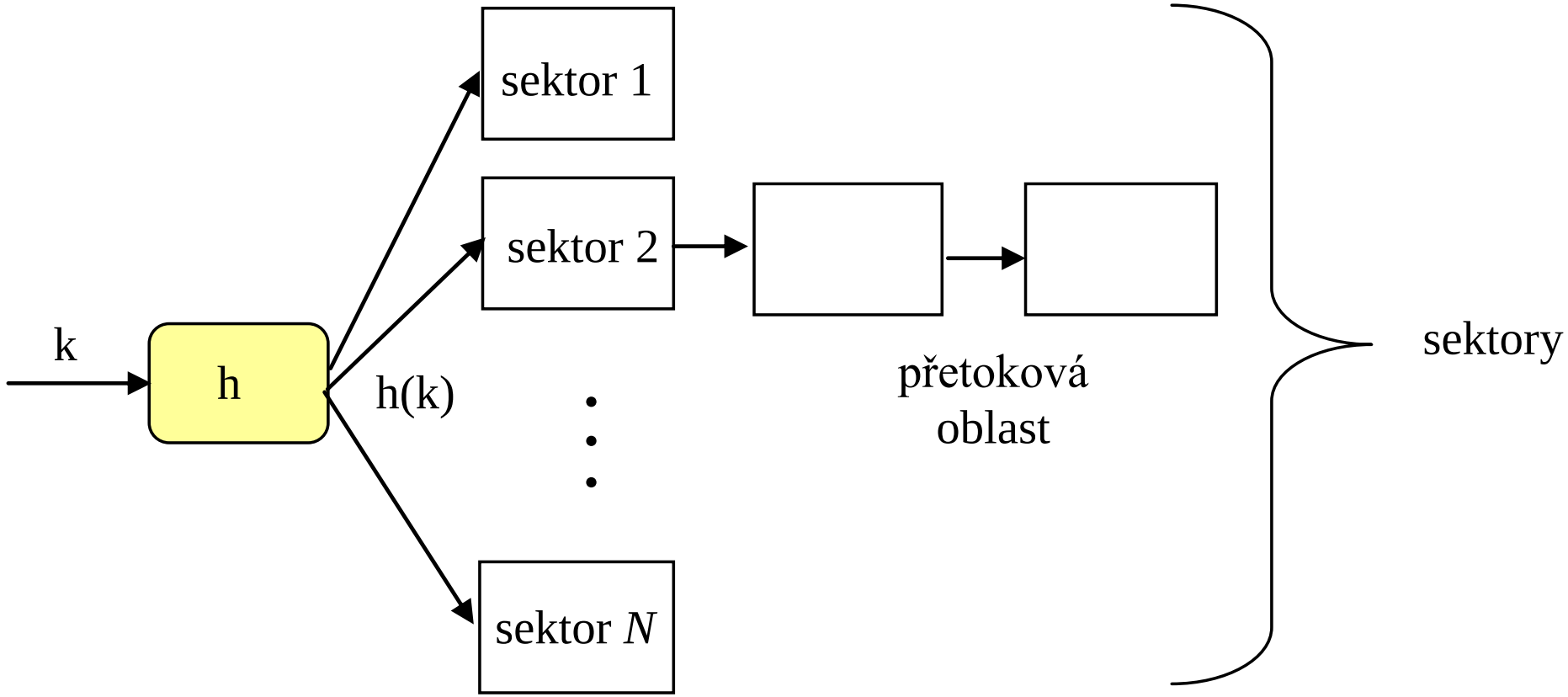
	Min. počet ukazatelů	Max. počet ukazatelů
Indexová část	$\lceil 51/2 \rceil = 26$	51
Sekvenční část	$\lceil (51-1)/2 \rceil = 25$	50

Úroveň	Minimální zaplnění (maximum bloků)	Maximální zaplnění (minimum bloků)
Sekvenční	$\lfloor 15000/25 \rfloor = 600$ bloků	$\lceil 15000/50 \rceil = 300$ bloků
Indexová X	$\lfloor 600/26 \rfloor = 23$ bloků	$\lceil 300/51 \rceil = 6$ bloků
Indexová X-1	1	1

Výsledek: Index bude mít 3 úrovně. Sekvenční část indexu bude mít 300 až 600 bloků a indexová bude tvořena kořenem a jednou úrovní s 6 až 23 bloky.

9.8. Hašování

- **Hašované soubory (přímé hašování)**
  - **Hašovací funkce převádí hodnotu vyhledávacího klíče na adresu sektoru záznamů (typicky blok).**



9.8. Hašování

- **Požadavky na hašovací funkci**
  - **rozložení hodnot je rovnoměrné**
  - **výskyt hodnot je náhodný**
- **Typické hašovací funkce**
  - **součet_bin_reprezentace_znaků_řetězce MOD počet_sektorů**
- **Přetečení sektorů**
  - **nedostatečný počet sektorů**
    - **musí platit: $N > n_z/z_s$, kde N je počet sektorů, n_z je celkový počet záznamů a z_s je počet záznamů sektoru**
  - **přetížení některých sektorů (skew)**
    - **nerovnoměrné rozdělení hodnot vyhledávacího klíče**
    - **počet sektorů se volí: $(n_z/z_s)*(1+d)$, kde d je faktor korekce (fudge), typicky 0.2**
  - **řešení řetězením přetokových sektorů**

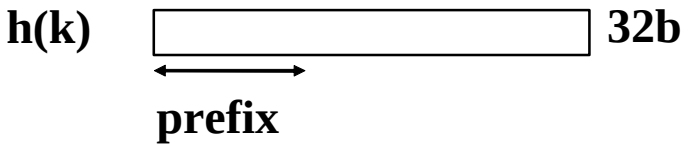
Př) Oracle: viz shlukování (dále)

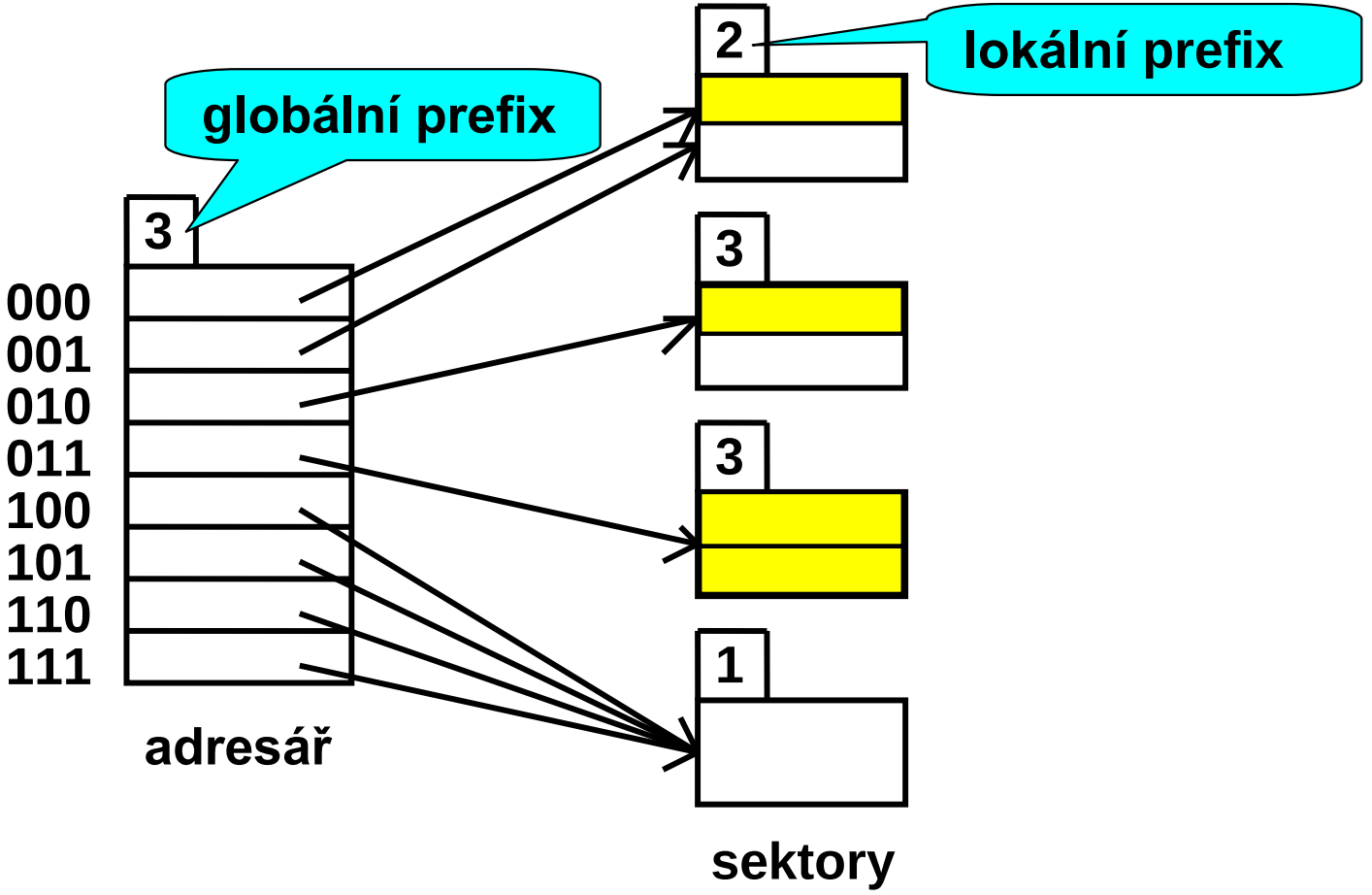
9.8. Hašování

- **Hašované indexy (nepřímé hašování)**
  - **soubor může být hašovaný pouze podle jednoho vyhledávacího klíče**
  - **u hašovaného indexu jsou v sektorech položky s hodnotou vyhledávacího klíče a ukazatelem do souboru (obdoba sekundárních indexů)**
- **Nevýhoda statického hašování**
  - **hašovací funkce se musí zvolit a paměť alokovat při definici hašovaného souboru nebo indexu a nelze je snadno změnit se změnou velikosti tabulky**

9.8. Hašování

- **Dynamické** hašování
  - dynamická modifikace hašovací funkce tak, aby odrážela změny velikosti tabulky
  - Rozšiřitelné hašování





9.9. Shlukování (clustering)

- **Podstata**
  - umístění záznamů se stejnými vlastnostmi (hodnotou tzv. klíče pro shlukování (cluster key)) fyzicky blízko u sebe
  - přístup k záznamům prostřednictvím indexu (v listové části mohou být přímo záznamy (tabulka organizovaná jako index)) nebo hašováním (obyčejné přímé hašování, tj. hašovaný soubor)
  - v souboru mohou být záznamy jedné nebo několika tabulek

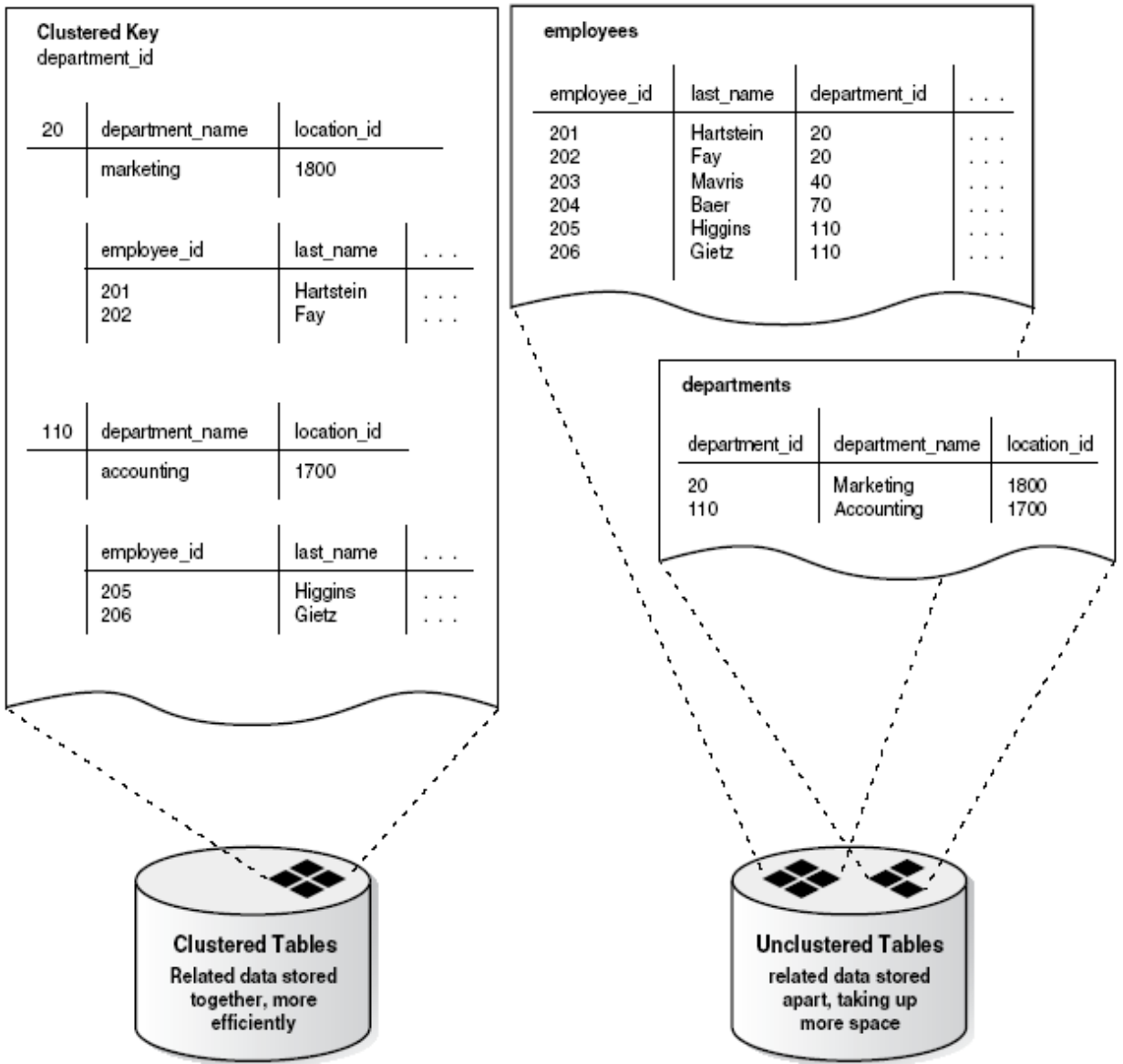
Př) shlukování v Oracle

```
CREATE CLUSTER employees_departments_cluster
     (department_id NUMBER(4)) SIZE 512;
CREATE INDEX idx_emp_dept_cluster ON CLUSTER
     employees_departments_cluster;
```

nebo

```
CREATE CLUSTER employees_departments_cluster
     (department_id NUMBER(4)) SIZE 512 HASHKEYS 10;
CREATE TABLE employees ( ... )
     CLUSTER employees_departments_cluster (department_id); ...
```

9.9. Shlukování (clustering)



9.10. Bitmapové indexy

- **Podstata**
  - uložení informace o tom, ve kterých záznamech tabulky se vyskytuje daná hodnota vyhledávacího klíče, ve formě bitových vektorů → velikost bitmapového souboru je výrazně menší
- **Předpoklad**
  - očíslování jednotlivých řádků tabulky (RID/ROWID)
  - ve sloupci vyhledávacího klíče je relativně málo různých hodnot

Př.) Tabulka Klient

číslo záznamu	r_cislo	jmeno	prijmova skupina	mesto
0	440726/0672	Jan Novák	S2	Brno
1	530610/4532	Petr Veselý	S3	Brno
2	601001/2218	Ivan Zeman	S1	Brno
3	510230/048	Pavel Tomek	S2	Brno
4	580807/9638	Josef Mádr	S5	Brno
5	625622/6249	Jana Malá	S2	Vyškov

9.10. Bitmapové indexy

Př.) Bitmapa pro příjmovou skupinu

S1	001000
S2	100101
S3	010000
S5	000010

Bitmapa pro město

Brno	111110
Vyškov	000001

- **Použití**
  - dotazy na hodnoty několika vyhledávacích klíčů spojené logickými spojkami
    Př) Najdi klienty z Brna z příjmové skupiny S2.
    $111110 \wedge 100101 = 100100 \rightarrow$ vyberou se odpovídající záznamy
  - agregační funkce COUNT(*) pro tabulku reprezentovanou bitovou mapou vzniklou operací nad bitovými indexy.
    Př) Kolik klientů z Brna patří do příjmové skupiny S2?
    počet_jedniček$(111110 \wedge 100101)$ = počet_jedniček $(100100) = 2$

9.11. Fyzický návrh databáze

- **Zdroje pro návrhová rozhodnutí**
  - **nefunkční (výkonnostní požadavky)**
  - **pochopení zátěže**
- **Popis zátěže**
  - **Seznam dotazů a jejich četnosti**
    - **které tabulky se čtou**
    - **které atributy jsou vybírány (klauzule SELECT)**
    - **atributy se vyskytují v podmínkách selekce nebo spojení (klauzule WHERE) a jak *selektivní* tyto podmínky jsou**
  - **Seznam aktualizací a jejich četnosti**
    - **atributy se vyskytují v podmínkách selekce (klauzule WHERE) a jak selektivní tyto podmínky jsou**
    - **typ aktualizace (INSERT, DELETE, UPDATE) a aktualizovaná tabulka**
    - **pro UPDATE sloupce, které jsou modifikovány**
  - **Výkonnostní požadavky pro každý typ dotazu a aktualizace, resp. kritické operace**

9.11. Fyzický návrh databáze

- **Základní rozhodování fyzického návrhu**
  - **kdy a jaké přístupové metody použít**
    - **pro které tabulky a které sloupce, resp. kombinace sloupců**
    - **typ přístupové metody**
- **Základní zásady rozhodování**
  1. **Zda vůbec použít nějakou speciální přístupovou metodu**

     Nevytvářej index, pokud nepřispěje ke zrychlení dotazů, resp. přístupu při aktualizaci. Důležité kritérium – selektivita dotazu.
  2. **Volba vyhledávacího klíče**

     Sloupce v klauzulích WHERE a výrazech s JOIN jsou kandidáty na vyhledávací klíč. Pro porovnání na rovnost je efektivnější hašování, pro rozsahové dotazy je nutné použít B+ stromy.

9.11. Fyzický návrh databáze

- **Základní zásady rozhodování (pokračování)**
  3. **Složené vyhledávací klíče**

     Složené vyhledávací klíče zvažuj v situaci, kdy klauzule WHERE obsahuje podmínky pro více než jeden sloupec tabulky.
  4. **Hašování vs. B+ strom**

```
SELECT a
FROM R
WHERE a = c;
```

```
SELECT a
FROM R
WHERE a BETWEEN c1 AND c2;
```

B+ stromy jsou obecnější (rozsahové dotazy). Hašování je výhodnější v těchto situacích:

a) **Pro podporu přirozeného spojení a spojení na rovnost metodou „hash join“.**
b) **Existuje-li velmi významný dotaz na rovnost a nejsou rozsahové dotazy se sloupci vyhledávacího klíče.**
c) **Lze-li určit potřebný prostor pro statické hašování.**

9.11. Fyzický návrh databáze

- **Základní zásady rozhodování (pokračování)**
  5. **Shlukování může přispět ke zvýšení výkonnosti při častém spojování řádků shlukovaných tabulek.**

     Musíme ale zvažovat případné přetečení bloků apod.
  6. **Vyvážení režie údržby indexu**

     Po vytvoření seznamu žádoucích indexů zvažuj dopad na aktualizace.

     a) **Pokud údržba indexu zpomalí významné aktualizační operace, vyřaď ho ze seznamu.**

     b) **Pamatuj ale, že přidání indexu může významně urychlit i aktualizační operace (prohledávací UPDATE a DELETE).**

9.11. Fyzický návrh databáze

- **Další rozhodování fyzického návrhu**
  - **Zda modifikovat návrh tabulek z důvodu vyšší výkonnosti**
    - **Alternativní normalizované schéma, je-li více možností**
    - **Denormalizace – omezení spojování tabulek vs. nebezpečí nekonzistence a prostor.**
    - **Vertikální rozčlenění tabulky – jsou-li významné dotazy jen na některé sloupce rozsáhlé tabulky.**
    - **Replikace dat**
  - **Zda přepsat často prováděné dotazy a transakce, aby byly rychlejší**

Literatura

1. Silberschatz, A., Korth H.F., Sudarshan, S.: Database System Concepts. Fifth Edition. McGRAW-HILL. 2006, str. 441-529.
2. Lemahieu, W., Broucke, S., Baesens, B.: Principles of Database Management. The Practical Guide to Storing, Managing and Analyzing Big and Small Data. Cambridge University Press 2018, str. 349-394.
3. Zendulka, J., Rudolfová, I.: Databázové systémy. IDS. Studijní opora. FIT VUT v Brně. 2006, str. 110-157.