

Fyzická implementácia DB, B-stromy a indexy

- 1) Súbory
- 2) Stránka, jej typy. Dátový a indexový súbor.
- 3) Klastrový a neklastrový index
- 4) B-strom
- 5) Návrh indexov
- 6) Príklady vytvorenia indexov

1) Súbory

```
SELECT name AS LogicalFileName, physical_name AS PhysicalLocation, state_desc AS Status FROM sys.master_files;
```

SQL Server mapuje DB na systémové súbory.

SQL Server rozlišuje 3 typy súborov:

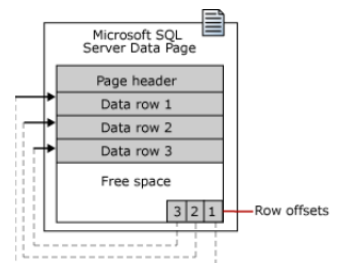
- **primárny dátový súbor** - *.mdf obsahuje napr. tabuľky, indexy
 - každá DB ma iba jeden
 - môže ukázať aj na ďalšie súbory + master DB
- **sekundárny dátový súbor** - *.ndf - používa sa, keď jeden súbor na tab už nestačí
- **Log súbor** - *.ldf
 - slúži na obnovenie DB po transakcii

2) Stránka, jej typy. Dátový a indexový súbor.

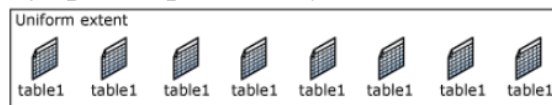
Tabuľky a indexy sú uložené ako kolekcia 8-KB stránok.

Stránka - Page

- je **základná jednotka** ukladania dát (tabuliek a indexov)
- operácie I/O sú vykonané pomocou stránok
- 1 page = 8kb.
- stránka začína hlavičkou za ktorou nasledujú seriové riadky
- 8 typov strán - **dáta**, **indexy**, image/text, info o voľných miestach + 4 ďalšie



Extent je skupina ôsmich logicky susediacich stránok (1 extent obsahuje 8*8 KB). Ak stránka je plná a príde nový záznam, *riadok*, preňho sa vytvorí nová stránka.



a) Indexový súbor

Index v knihách je zoznam dvojíc - dôležitých pojmov a čísiel strán umiestnený na konci knihy v abecednom poradí. Index zrýchli vyhľadávanie pojmov. Indexy v DB tiež slúžia na rýchle vyhľadávanie riadkov tabuľky.

Indexový súbor / indexová tabuľka je v podstate **zoznam s dvomi poliami**: prvé pole je **index** ⇔ **kľúč** a druhé pole môže byť **dvojaké**:

- buď obsahuje samotný záznam
- alebo smerník na blok dátového súboru, kde sa nachádzajú riadky s príslušnou hodnotou indexového poľa.

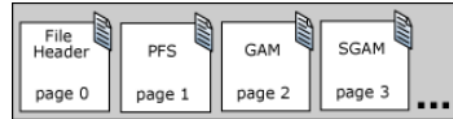
Indexový súbor je oveľa **menší** ako dátový súbor.

Rozlišujeme klastrový a neklastrový index. Tabuľka s **klastrovým** indexom je preorganizovaná do štruktúry B-stromu, pozri nižšie. Ak tabuľka má **neklastrový** index, ten má **indexová tabuľka** ktorá je **aj fyzicky** zotriedená.

b) Dátové súbory majú prísnu štruktúru a obsahujú: header, PFS (page free space), GAM (global allocation map), ...

Ukázkové T-SQL príkazy:

```
DBCC fileheader (Poliklinika)
DBCC IND('Poliklinika',Navstevy,-1) -- _____
DBCC TRACEON(3604); DBCC PAGE('Poliklinika',1,143,3) WITH TABLERESULTS --vykonat spolu
```



Dátové tabuľky

Dátové stránky tabuľky môžu byť organizované ako

- klastrové tabuľky s klastrovým indexom (B-stromy)

alebo

- HEAP tabuľky (halda) bez klastrového indexu

Kým klastrové tabuľky sú zoradené, halda je bez zoradenia.

3) Klastrový a neklastrový index

Ak tabuľka obsahuje

- **klastrový index**, potom

- o **dáta** (presnejšie **údajové/ dátové stránky**) sú **v listových uzloch** alebo na **listovými smerníkmi** presmerovaných miestach. Tabuľka môže mať iba **jeden** klastrový index.

- **neklastrový index**, ten je tiež organizovaný ako B-strom, ale

- o **v listových uzloch** sú **indexové stránky** namiesto údajových stránok. Tabuľka môže mať až **249** neklastrových indexov.

Indexové stránky obsahujú indexové riadky, každý z ktorých sa skladá z kľúča a z **lokátora** riadku.

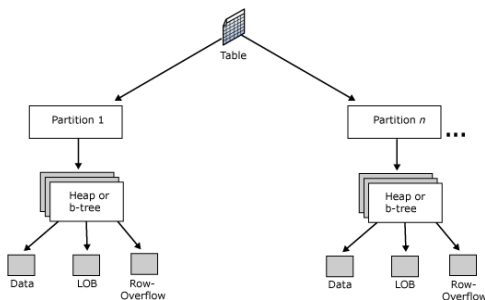
Neklastrový index môže byť definovaný na

- o tabuľke alebo view s klastrovým indexom, potom **lokátor** je **kľúč** klastrového indexu na dátový riadok
- o halde, potom **lokátor** je **ukazovateľ**/pointer na dátový riadok.

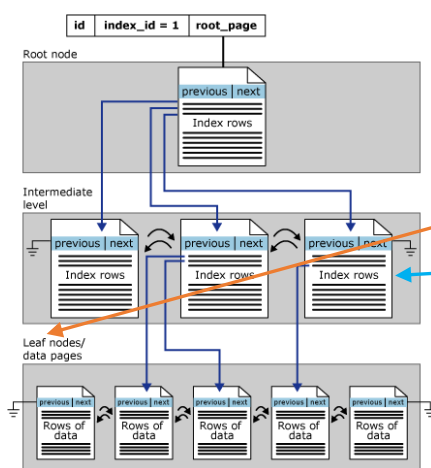
Partícia je horizontálne rozdelenie tabuľky alebo indexu (ich riadkov) do menších jednotiek, definovaných užívateľom! **Extent** je fyzická jednotka úložiska na disku, zatiaľ čo **partícia** je logické rozdelenie veľkej tabuľky na menšie kusy.

Vďaka partícii veľké tabuľky a indexy sa chovajú optimálnejšie.

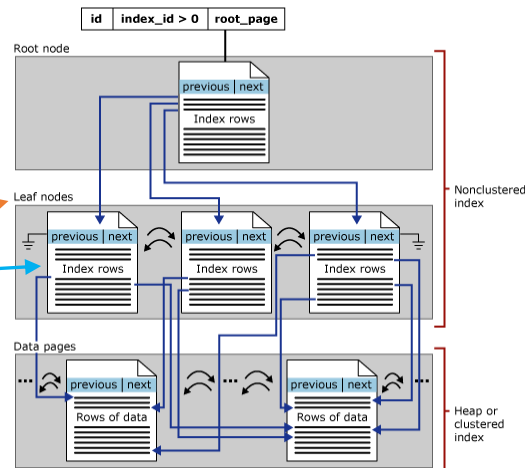
- tabuľka => partície [http://msdn.microsoft.com/en-us/library/ms188706\(SQL.90\).aspx](http://msdn.microsoft.com/en-us/library/ms188706(SQL.90).aspx)



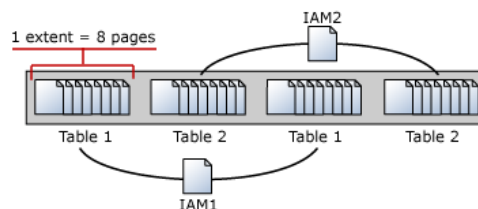
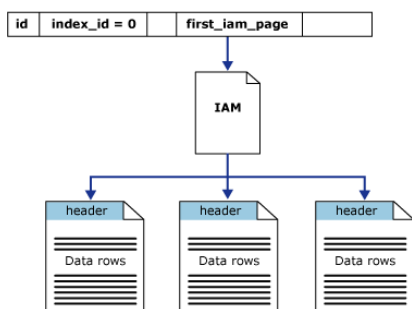
- partícia: riadky => pages – **klastrová tabuľka s klastrovým indexom**
- **heap (halda) tabuľka bez kl. indexu**



Štruktúra klastrového indexu v jednej partícii.
[http://msdn.microsoft.com/en-US/library/ms177443\(v=sql.90\).aspx](http://msdn.microsoft.com/en-US/library/ms177443(v=sql.90).aspx)



Štruktúra neklastrového indexu v jednej partícii.
[http://msdn.microsoft.com/en-us/library/ms177484\(v=sql.90\).aspx](http://msdn.microsoft.com/en-us/library/ms177484(v=sql.90).aspx)



SQL Server používa IAM (Index Allocation Map) stránky na získanie dátových riadkov v jednej **heap** partícii. [http://msdn.microsoft.com/en-us/library/ms188270\(v=sql.90\).aspx](http://msdn.microsoft.com/en-us/library/ms188270(v=sql.90).aspx) [http://msdn.microsoft.com/en-us/library/ms187501\(SQL.90\).aspx](http://msdn.microsoft.com/en-us/library/ms187501(SQL.90).aspx)

4) B-stromy

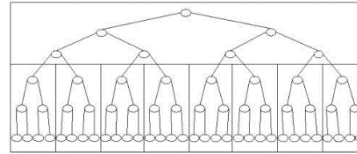
- **Strom**
- **B-strom**
- **Výška B-stromu**
- **Operácie na B-stromoch**

Terminológia stromu

Strom je tvorený **uzlami**/vrcholmi a **hranami**. Rozlišujeme 3 typy **uzlov**:

- koreňový
- vnútorný, nelistový
- vonkajší, **listový**

Každý uzel, okrem koreňa má jeden **rodičovský** uzel a niekoľko – nula a viac – **potomkov**.



Binárny strom rozdelený na "stránky" (po 7 uzlov)

Výška, hĺbka uzla je vždy o jeden viac ako úroveň rodičovského uzla, pričom úroveň koreňa je nula.

Podstrom uzla sa skladá z uzla a všetkých jeho potomkov.

- **Koreň (Root):** Úroveň 0

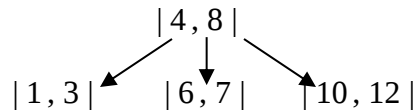
- **Deti koreňa:** Úroveň 1

- **Vnuci koreňa:** Úroveň 2

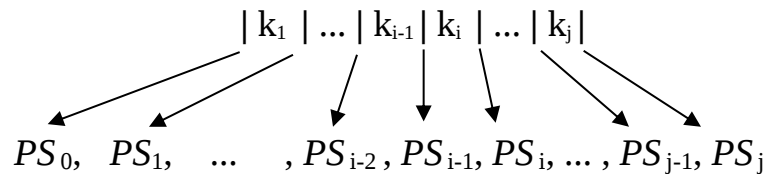
B-strom

B-strom je stromová dátová štruktúra, vhodná na ukladanie a čítanie veľkých celkov dát a zabezpečuje vyhľadanie, vkladanie a vymazanie za **logaritmický čas**. B-strom zaviedol Bayer (1972) a McCreight.

Schematicky B-strom vyzerá nasledovne (rádu 3, pozri nižšie ***)



Každý interný uzel v **B-strome** má tvar



alebo

$$\langle p_1, k_1, \dots, p_{i-1}, k_{i-1}, p_i, \dots, p_i, k_i, p_{i+1} \rangle$$

kde p_i je **pointer**/smerník (šípka) na iný uzel v strome / podstrom / PS_i , k_i je hodnota kľúča, i je počet **kľúčov/prvkov** k_i v uzle, $i=1,2,\dots,I$ a $I=1,2,\dots,m-1$, kde m je **rád** stromu.

Definícia B-stromu

B-strom **rádu m** (B_m) je strom s vlastnosťami:

- 1) Koreň buď má aspoň **dvoch potomkov** alebo je **list**.
- 2) Každý vnútorný uzel má **rovnaký počet** potomkov - **minimálne** $\lceil m/2 \rceil$ a **maximálne m potomkov**.
- 3) **Vnútorný uzel s $j+1$ potomkami obsahuje j kľúčov/prvkov** ***.
- 4) Kľúče v uzle sú **usporiadané**

$$k_1 < \dots < k_{i-1} < k_i < \dots < k_j.$$

- 5) Pre kľúče **podstromu** v_{j-1} a v_j platí

$$k_{j-1} < k < k_j.$$

- 6) Každý **list** je v **tej istej hĺbke** od koreňa.

B-stromy rádu 4 sú známe ako 2-3-4 stromy (pozri obr. v dôkaze nižšie).

Výška/Hĺbka B-stromu

Počet operácií v B-strome je proporcionálny k hĺbke stromu, ktorá rastie pomaly.

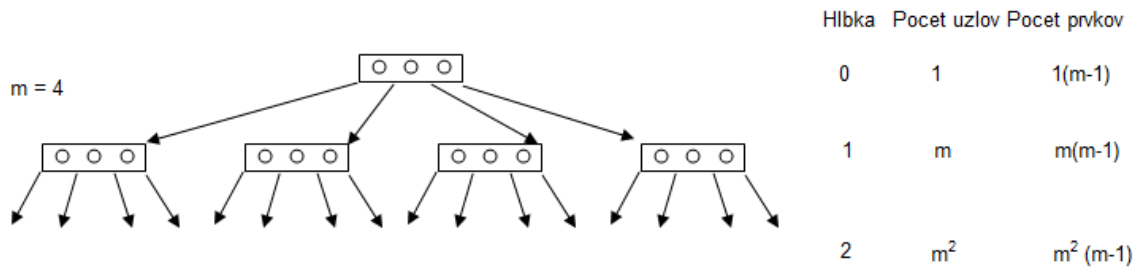
Veta

Úplny B-strom rádu $m \geq 2$ s $n \geq 1$ prvkami má hĺbku

$$h = \log_m(n+1) - 1.$$

Dôkaz:

Vieme, že m zadá maximálny počet potomkov a $m-1$ zadá max. počet prvkov. Nech $m=4$.



Teda pre maximálny počet **prvkov** n až do hĺbky h platí:

$$n = (m-1) \sum_{i=0}^h m^i = (m-1)(m^{h+1}-1)/(m-1) = m^{h+1}-1$$

z čoho po algebrických úpravách a logaritmovaní oboch strán plynie: $h = \log_m(n+1) - 1$.

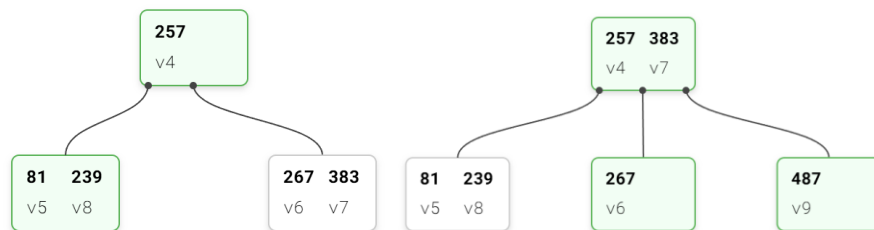
Základné operácie na B-stromoch

- **Vyhľadávanie**
- **Vkladanie** – po zlúčení (1) stredný hore a zároveň štiepime (2,3), pozri obr. 1, 2, 3
V kroku stredný hore sa počet prvkov horného uzla zväčší o jeden, preto musí nasledovať buď ďalší hore alebo štiepenie
- **Vymazávanie** – štiepime alebo zlučujeme

Vkladanie – B₂:

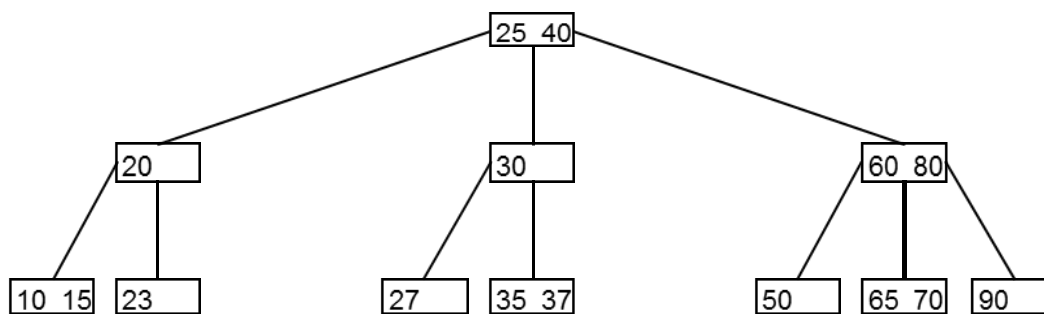
<https://btree.app/>

257 81 267 383 239 487

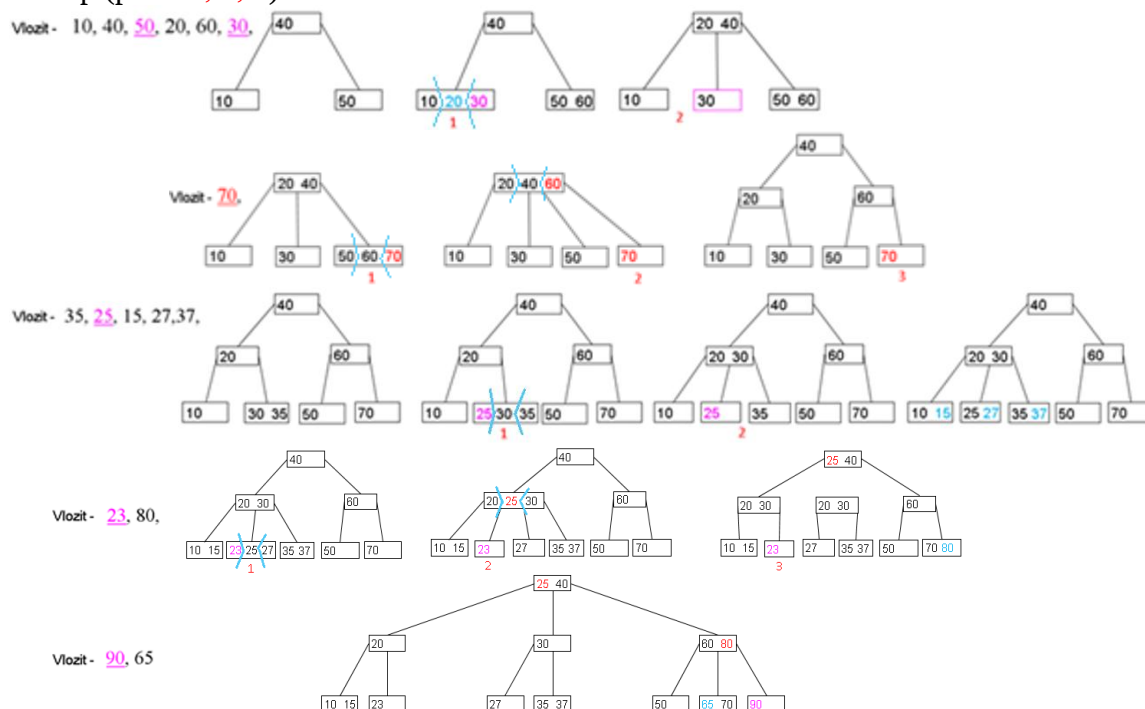


Vkladanie - B₃:

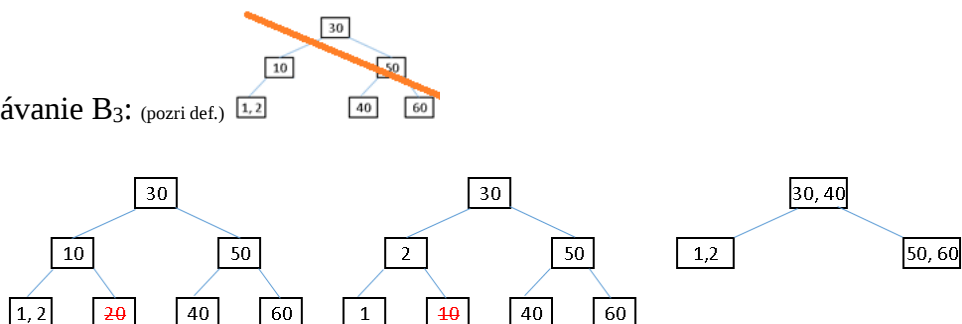
10, 40, 50, 20, 60, 30, 70, 35, 25, 15, 27, 37, 23, 80, 90, 65



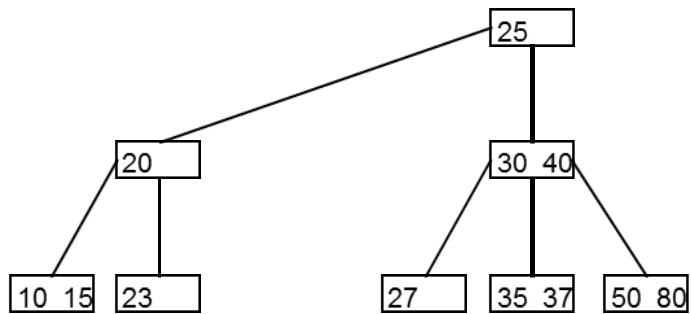
Postup (pozri 1, 2, 3)



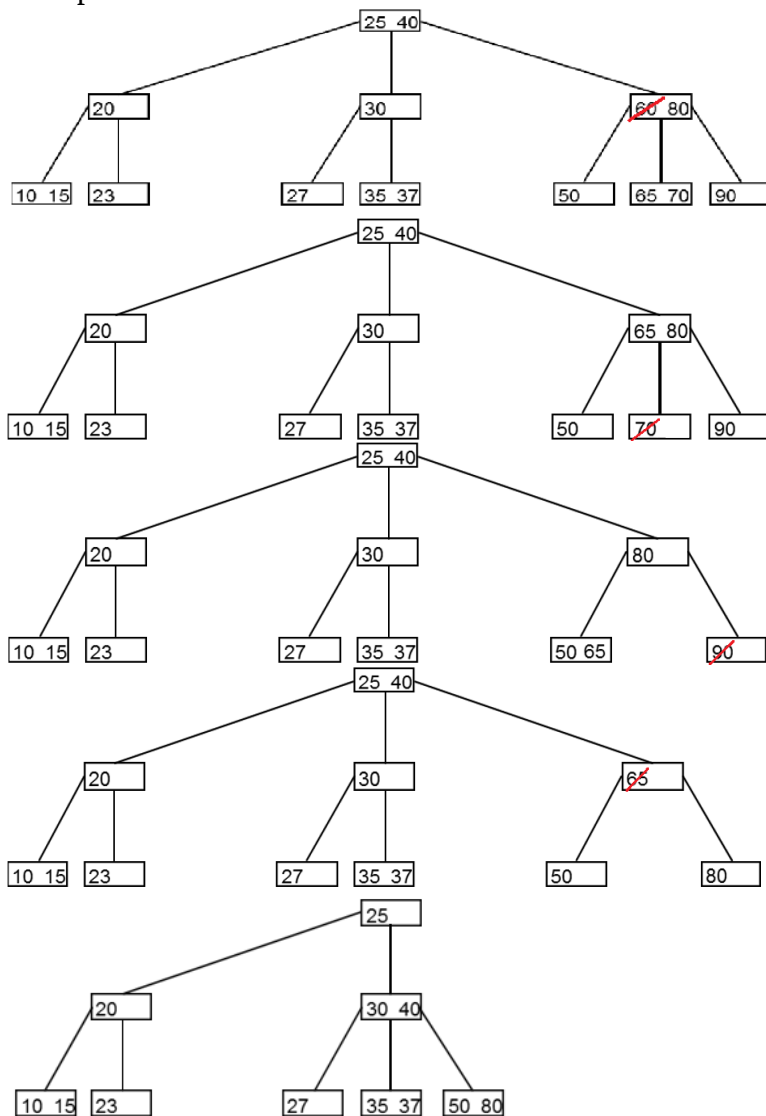
Vymazávanie B₃: (pozri def.)



Vymazávanie 60, 70, 90, 65



Postup



5) Návrh indexov

Indexovať?

Význam indexu

- výhody
- nevýhody (pozri príklady)

Návrh indexu

Výber indexu a jeho typu treba vždy dôkladne premyslieť. Existujú všeobecné odporúčania, kedy (ne)použiť index a aký typ. SQL Server má index-tuning system: Database Tuning Advisor (DTA).

Dobrý index

- ak tabuľka je viac dopytovaná ako modifikovaná
- stĺpce, ktoré nie sú často modifikované
- stĺpec/ce s cudzím kľúčom
- stĺpec, ktorý má veľa odlišných hodnôt
- stĺpce vo WHERE klauzule
- dopyty s order by / join / agregácie

Zlý index

- ak tabuľka je viac modifikovaná ako dopytovaná
- tabuľka s malým počtom riadkov
- stĺpec, ktorý má málo odlišných hodnôt
- stĺpec, ktorý sa v dopyte nepoužíva
- stĺpce s dátovými typmi: **text, ntext, image, a bit**

Treba mať na pamäti, že ak index sa skladá z viacerých atribútov, potom pri vkladaní nových riadkov alebo modifikovaní daných stĺpcov prebieha viac dodatočných úkonov, súvisiacich s údržbou indexu.

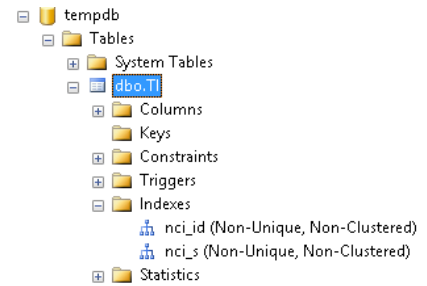
6) Príklady vytvorenia indexov

Prezerat' napr. tempdb - **Tables** - T1 - Keys & **Indexes**:

<https://msdn.microsoft.com/en-us/library/ms188783.aspx>

Názvy indexov:

- pki - PRIMARY KEY index - nedá sa drop-núť
- ci - CLUSTERED (klastrový) index
- nci - NONCLUSTERED (neklastrový) index



Príklady - Typy a spôsoby vytvorenia indexov

Nižšie uvedieme rôzne možnosti definície klastrových či neklastrových indexov s generovanými či vlastnými názvami.

Rôzne scenáre

a) Žiadny index (TI0)

b1) PRIMARY KEY => CLUSTERED index (TI1), (TI2)

Nedá sa DROP-núť pki, iba ci a nci:

TYPY objektov a indexov: sys.objects, sys.indexes

Dátum vytvorenia:

b2) CLUSTERED - sys.indexes <=> sysindexes (TI3), (TI4)

c1) NONCLUSTERED (TI5)

c2) Default - NONCLUSTERED (TI5)

```
USE tempdb
```

```
GO
```

```
IF OBJECT_ID('TI0','U') IS NOT NULL DROP TABLE TI0;
```

```
GO
```

```
-- a) Tab. bez indexu
```

```
CREATE TABLE TI0(id int, s CHAR(10))
```

```
-- b1) primary key => clustered index
```

```
-- Bez nazvu
```

```
IF OBJECT_ID('TI1','U') IS NOT NULL DROP TABLE TI1;
```

```
GO
```

```
CREATE TABLE TI1(
```

```
    id int NOT NULL primary key,  
    s CHAR(10)) -- generovaný index je dlhý: PK_TI1_3213E...
```

```
-- vs
```

```
-- S kratším názvom pki_TI2:
```

```
IF OBJECT_ID('TI2','U') IS NOT NULL DROP TABLE TI2;
```

```
GO
```

```
CREATE TABLE TI2(
```

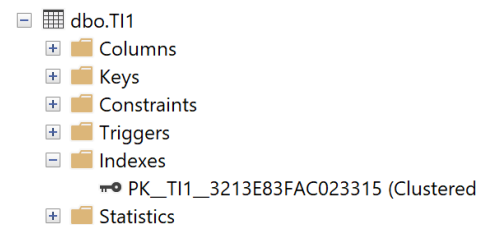
```
    id int NOT NULL,  
    s CHAR(10),  
    CONSTRAINT pki_TI2 primary key (id) ) -- s názvom
```

```
-- Neda sa DROP-núť, pozri o 1 riadok vyššie:
```

```
IF EXISTS (SELECT name FROM sys.indexes WHERE OBJECT_ID=OBJECT_ID('TI2')  
          AND name = 'pki_TI2' )
```

```
    DROP INDEX TI2.pki_TI2;
```

```
GO
```



```

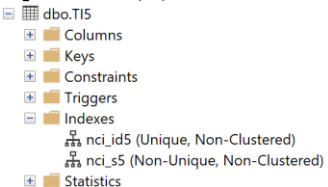
--b2) CLUSTERED cez NOT EXISTS: sys.indexes <=> sysindexes
---- CREATE CLUSTERED INDEX ind_id ON TI(id) <=>
---- CREATE INDEX ind_s ON TI(s) -- default NONCLUSTERED

-- PK nie, CI ano
IF OBJECT_ID('TI3','U') IS NOT NULL DROP TABLE TI3;
CREATE TABLE TI3(id int, s CHAR(10))
GO
IF NOT EXISTS (SELECT name FROM sys.indexes WHERE name = 'ci_TI3')
    CREATE CLUSTERED INDEX [ci_TI3] ON TI3(id)

-- Pomocou EXISTS a DROP:
IF OBJECT_ID('TI4','U') IS NOT NULL DROP TABLE TI4;
CREATE TABLE TI4(id int, s CHAR(10))
GO
IF EXISTS (SELECT name FROM sys.indexes WHERE OBJECT_ID=OBJECT_ID('TI4')
            AND name = 'ci_TI4' )
DROP INDEX TI4.ci_TI4;
CREATE CLUSTERED INDEX [ci_TI4] ON TI4(id)
-----

-- c1-c2) Dva NONCLUSTERED na dva stlpce - raz unique:
IF OBJECT_ID('TI5','U') IS NOT NULL DROP TABLE TI5;
CREATE TABLE TI5(id int, s CHAR(10))
GO
-- Na oba stlpce nonclustered index:
IF NOT EXISTS (SELECT name FROM sys.indexes WHERE name = 'nci_id5')
CREATE UNIQUE NONCLUSTERED INDEX [nci_id5] ON TI5(id)
IF NOT EXISTS (SELECT name FROM sys.indexes WHERE name = 'nci_s5')
CREATE INDEX [nci_s5] ON TI5(s) -- default je NONCLUSTERED

```



Výhody jedinečného/unique indexu

- Viacstĺpcové jedinečné indexy zaručujú, že žiadne dva riadky v tabuľke nemôžu mať rovnakú kombináciu hodnôt pre tieto stĺpce.
- Jedinečné indexy zabezpečujú integritu údajov definovaných stĺpcov.
- Jedinečné indexy poskytujú dodatočné informácie užitočné pre **optimalizátor** dotazov, ktorý môže vytvárať efektívnejšie plány vykonávania.

Cvičenie

Na cvičení ukážeme z rôznych hľadísk, ako vplýva index na čas vykonania ...

Definícia:

0_index_Create.sql

Nevýhoda:

1_CrInd_Insert__Insert_CrInd.sql

Uloženie:

2_ClIndex_Ukladanie.sql

Výhoda – hľadanie:

3a RAND_SEED_KERNEL.sql

3b Search_RandTab.sql

3c Search.sql

<https://github.com/0x00000000/0x00000000/blob/master/3c/Search.sql>