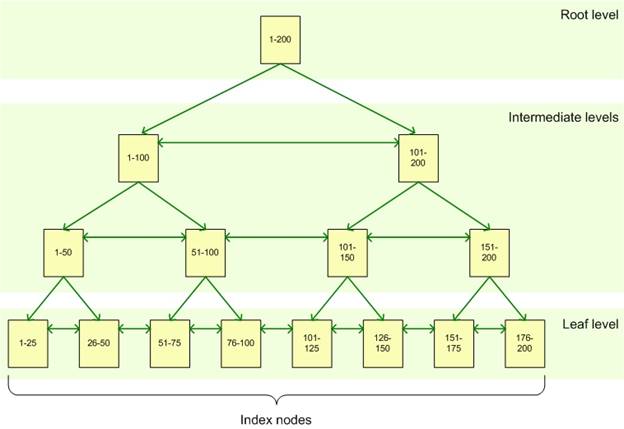
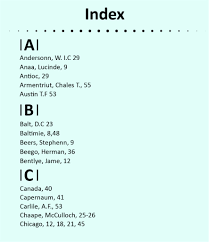
**Maturitní otázka**  
**B**áleš Antonín  
C4a  
16. 03. 2023

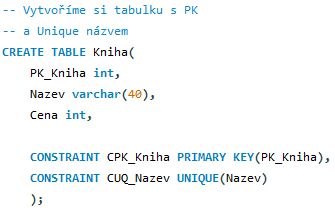
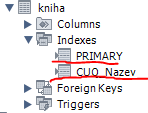
**Databázové systémy**

**16. Indexy a indexace dat v databázi (UNIQUE, INDEX)**

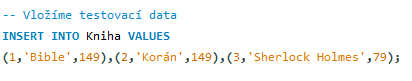
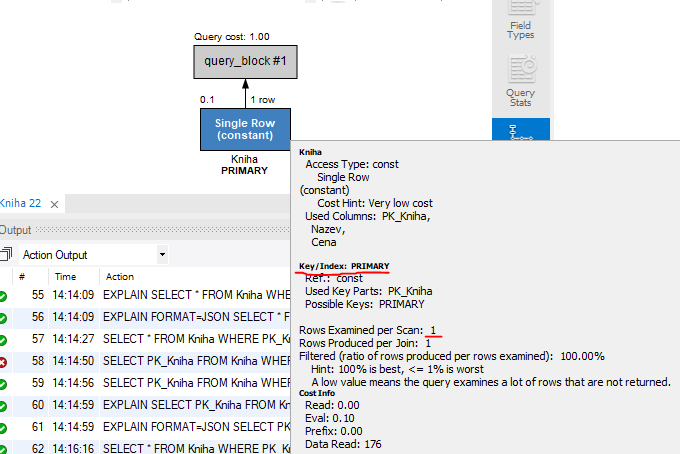
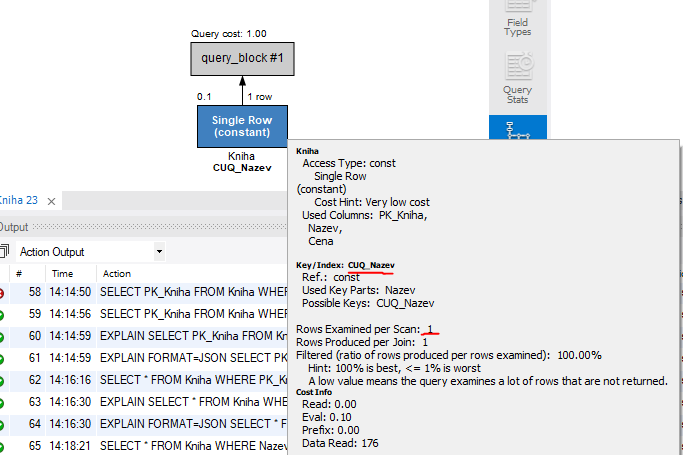
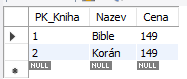
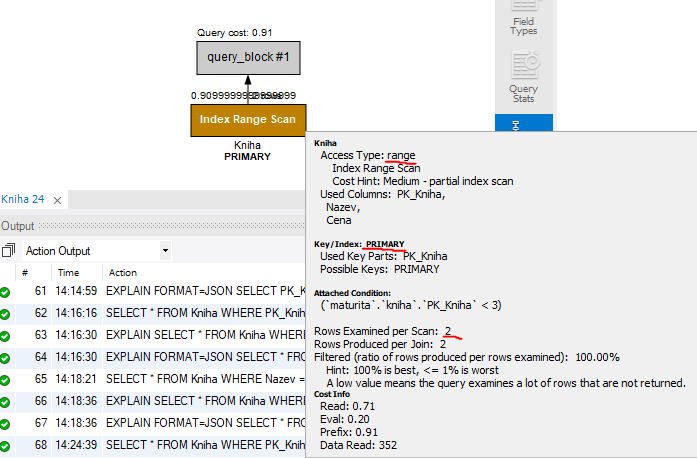
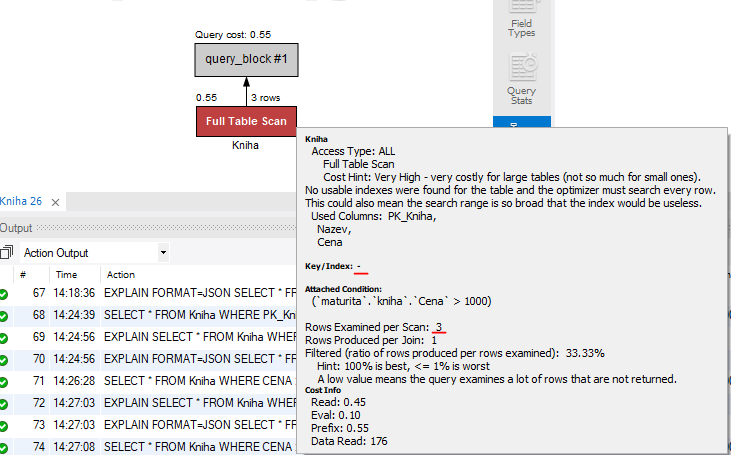
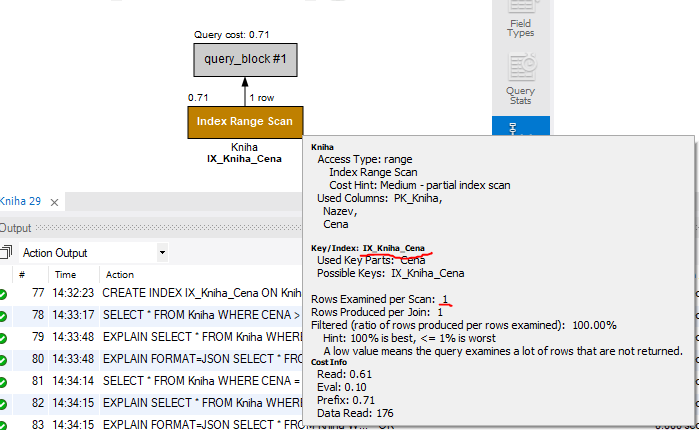
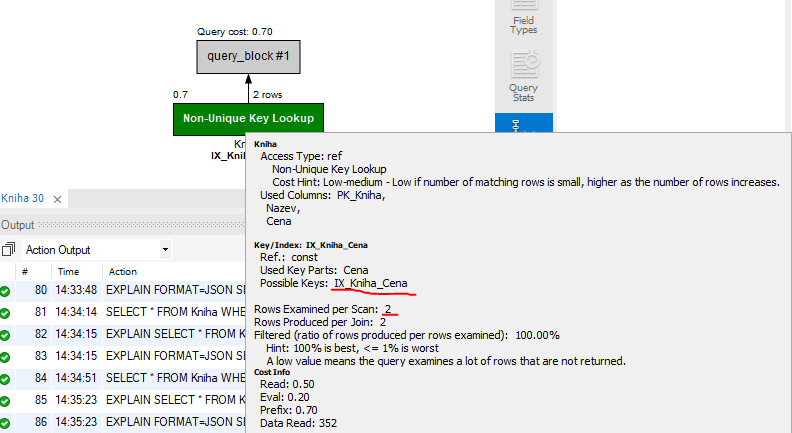
**Indexace**

* Indexy slouží pro urychlené hledání dat v databázi
* Jsou to speciální tabulky, do kterých se nahlíží
* Jestliže neexistují INDEXY, SELECT musí projít a zkontrolovat každý řádek zvlášť.
  + Představme si tabulku o 2.000.000.000 záznamů *(Nemají INDEX a proto nejsou data nijak zorganizována.. Musíme zkontrolovat každý řádek = také nazýváno jakožto FULL TABLE* ***SCAN****)*
  + SELECT s WHERE klauzulí tuto tabulku musí zkontrolovat každý řádek zvlášť, zda splňuje a má data, které požadujeme ve WHERE klauzuli...
* Existují dva typy Indexů:
  + **Clustered**
    - Může být použit pouze jednou, protože není speciální tabulkou, ale doopravdy organizuje reálné data v tabulce *(Nemůžeme např. Jednu tabulku seřadit jak abecedně dle jednoho sloupce, tak číselně dle sloupce druhého..)*
    - DEFAULTNĚ nastaven vždy na Primary KEY
    - Rozdělí data do stromu a několik středních levelů a listů, na které odkazují pouze daným rozmezím sloupce, který hledáme
    - SELECT tudíž nemusí projíždět celou tabulku, jen se pouze přesouvá ve stromu dokud nenalezne LEAF level posledního rozmezí, kde se sloupec může nacházet a poté najde náš výsledek
    - Clustered INDEX si můžeme představit jako seznam našich Kontaktů:  
      
      * Seznamy kontaktů jsou jakožto samotné již zorganizované, logicky většinou abecedně dle jmen
      * Jelikož víme, jak funguje abeceda, rychle najdeme například **“Učitelka na matiku”**
      * Urychleně jsme našli náš kontakt a všechny jiné data k němu. *(telefonní číslo, adresa … )*
      * Nemusíme zkontrolovat každý kontakt v kontaktech zvlášť, abychom našli Učitelku na matiku!
  + **Non-Clustered**
    - Vytvoří se nová speciální tabulka, která obsahuje organizovaná data požadovaného sloupce, a pouze POINTER kde se nachází požadovaná reálná instance v reálné tabulce *(tudíž neukládá celou tabulku znova a poté ji zorganizuje, zorganizuje pouze požadovaný sloupec/sloupce k zorganizování a jen odkáže, kde se v reálné tabulce nachází instance se všemi jejími daty)*
    - Principem rozdělení do stromu jinak funguje stejně, jako CLUSTERED Index
    - Non-Clustered INDEX si můžeme představit jako indexaci knížek:  
      
      * Na první/zadní straně je vždy nějak (většinou abecedně) seřazený celý obsah knížky *(Všimněme si, že se tam většinou nachází jen dané “téma,” ne celý obsah! Můžeme si tím tedy představit “téma” jako sloupec, na který jsme použili Non-Clustered INDEX)*
      * Díky tomu že je indexace seřazená, a jelikož víme jak funguje abeceda, se jednoduše zorientujeme a rychle zjistíme, kde je v knížce například téma na **“INDEXY”**
      * Tím zjistíme, kde se dané téma nachází a můžeme začít číst *(např. Odkazuje na stranu 63 = To si můžeme v databázích představit právě jako POINTER!)*
      * Nemusíme PROLISTOVAT CELOU KNIHU, abychom našli téma které potřebujeme!
* Jak databáze pozná, jestli má tabulka INDEX?
  + Při QUERY zkontroluje jestli na vámi požadované sloupce ve WHERE klauzuli z tabulky neexistuje INDEX, a pokud ano, použije jej. Pokud ne, musí databázi/tabulky celé projet.
* Jsou nějaké nevýhody INDEXU?
  + Jelikož Clustered Indexy organizují samotné data, nezabírají místo navíc. Na druhé stráně, Non-Clustered indexy vytváří nové speciální hledací zorganizované tabulky pro každý INDEX - databáze se tím může značně velikostně zvětšit.
  + Aby Non-Clustered indexy byli konzistentní, při každém novém INSERTU, UPDATU, DELETU, ... tabulky, na který je INDEX použit, se musí se také aktualizovat právě každý INDEX pro danou tabulku.
    - Představme si tabulku s 5 sloupci a na 3 z těchto sloupců máme Non-Clustered INDEX
    - Provedeme INSERT do tabulky
      * Musí se provést INSERT jak do tabulky, tak do všech třech příslušných indexů..
      * Kdyby byl INSERT jakkoliv složitější, například díky proceduře, a ovlivňoval by tak více tabulek naráz, které má své vlastní INDEXY, můžeme si představit, jaké kvanta tabulek navíc musíme aktualizovat..
  + Proto i přes to, že nám INDEXY pomáhají tabulky prohledávat a dělají nám život jednodušší, ne vždy jsou na místě a ne vždy by se měli použít
  + Do tabulek které jsou značně častěji aktualizovány (Ať insert, update, či delete), než přistupovány, není žádný důvod proč vytvářet INDEX
* MySQL používá B-Tree strukturu k indexaci dat

**MySQL UNIQUE + PRIMARY KEY**

* **UNIQUE**
  + Zajišťuje, že data v tabulce budou unikátní
  + Každý UNIQUE má také automaticky vytvořený INDEX
* **PRIMARY KEY**
  + Ohledně indexace v podstatě funguje na stejném principu jako UNIQUE
  + DEFAULTNĚ nastaven jako Clustered INDEX *(Nejvíce se totiž používá a přistupuje)*
* 
* Poté můžeme vidět automaticky vytvořené INDEXY  
  

**INDEXY**

* Před vytvořením vlastního INDEXU, pojďme se podívat, jak fungují doopravdy v MySQL Workbench Indexy PRIMARY KEY a UNIQUE
  + 
  +   
      
      
    *Všimněme si, že i přes to že tabulka má 3 řádky, díky PRIMARY indexu tento SELECT scanoval pouze řádek jeden!*
  +   
      
      
    *Všimněme si podobných výsledků i na UNIQUE indexu!*
  +   
      
      
    *Všimněme si, že INDEX nyní musel hledat v rozmezí*  
    *I přes to každopádně SCANOVAL pouze 2 vrácené řádky, ne všechny 3!*
  +   
      
      
    *Ne že v tabulce nemáme žádnou knihu která by měla větší cenu jak 1.000, takže nemáme žádný výsledek, ještě za to že jsme nezískali žádné informace jsme museli prohledat KOMPLETNĚ CELOU TABULKU... (protože nemáme INDEX na ceně)*
* **Vytváření a manipulace INDEXŮ**
  + Představme si, že tedy chceme mít index na ceně.
  + Není to ani PK, ani nemusí být každá cena unikátní.
  + Pojďme si to názorně ukázat:
    - 
    -   
        
      *INDEX uvážil zkontrolovat jeden řádek, i přes to, že nemáme žádný výsledek*
    -   
        
      *Všimněme si, že INDEX použil jakýsi Non-Unique Key Lookup*  
      *A i přes to, že se v tabulce nachází 3 řádky, SCANoval pouze 2!*
  + Mazání INDEXŮ
    - INDEX je sice objekt, ale spadá pod tabulku
    - Tudíž při vytváření používáme CREATE ale přiřazujeme ho k tabulce
    - Jestli INDEX chceme zničit, musíme ho z dané tabulky odebrat
    - 
    - 