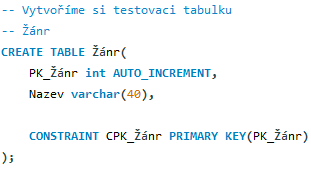
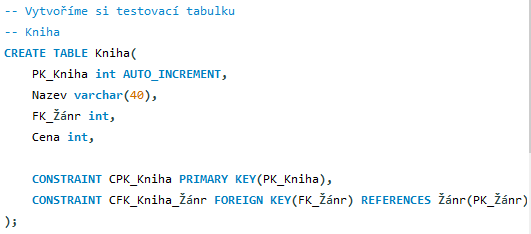
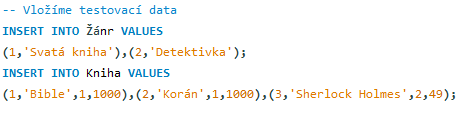
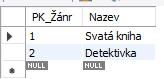
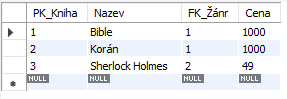
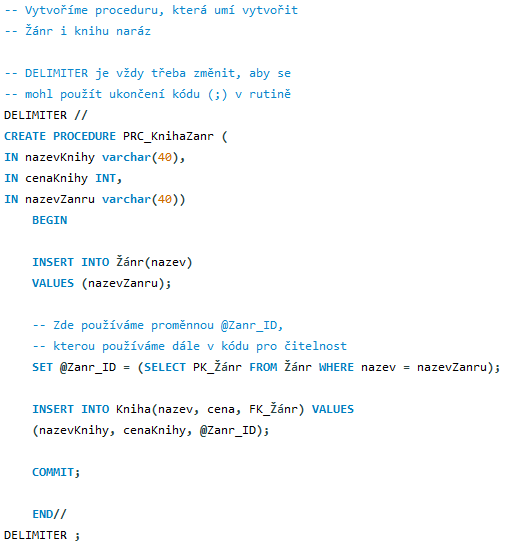
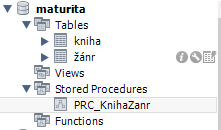
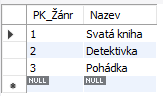
**17. Uložené procedury a funkce**

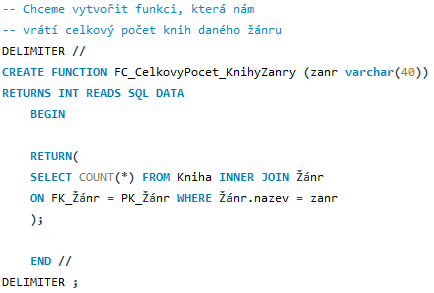
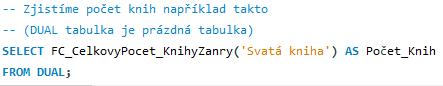
**Základní pojmy**

* Když chceme **znovu-použít blok kódu**, máme na výběr ze tří věcí:
  + **Pamatovat si ho a vždy ho vypsat**
    - Neefektivní způsob znovu-používání bloku kódu
    - Ne vždy musíme napsat kód tak, jak jsme doopravdy chtěli
    - Time-consuming
    - Redundance
  + **Spustit samotný .SQL soubor**
    - Více efektivní způsob znovu-použití bloku kódu
    - Musíme mít někde lokálně uložený .SQL soubor
    - Ten poté musíme vždy najít a spustit kód uvnitř
  + **Uložené rutiny**
    - Pod rutiny spadají jak **procedury**, tak **funkce**
    - **Nejefektivnější** způsob znovu-použití bloku kódu
    - Kód jednou napíšeme a vložíme ho do rutiny
    - Umožňují nám uchovat blok kódu (Jeden, či více příkazů), který je pak **spustitelný kdykoliv a kdekoliv** je třeba, pouhým voláním rutiny
    - I přes to že proměnné můžeme nastavit i jinde v .SQL kódu, nejvíce se používají právě v rutinách
      * Když potřebujeme udělat např. nějakou kalkulaci, jejíž výsledek je potřebný na více místech v rutině, místo volání kalkulace dvakrát, uložíme výsledek do proměnné a poté danou proměnou použijeme na místech, kde je třeba
* Než začneme ukazovat přímo příklady:
* 
* 
*   
    
  
* 

**Uložené procedury**

* Mají parametry – **INPUT** a **OUTPUT**.
  + Při volání procedury můžeme tedy nastavit, s jakými parametry má pracovat
* Vždy proceduru před použitím musíme volat příkazem **CALL**
*   
  *Můžeme si všimnout, že ji máme uloženou v ”Stored Procedures”*  
  
*   
    
    
    
  *Tímto jednoduchým zavoláním jsme ušetřili čas a vytvořili tak knihu a zároveň žánr k ní*

**Uložené User-Defined funkce**

* Mají pouze vstupní parametry – INPUT
* Místo OUTPUT parametrů má povinnost mít v sobě RETURN
* Vždy tedy musí funkce vracet jednu hodnotu
* Nevolá se příkazem **CALL**, ale pouze jejím vypsáním v DML Příkazu
* Funkce musí obsahovat datový typ RETURNU
* Funkce musí být označena také jedním z:
  + **DETERMINISTIC** - ?
  + **NOT DETERMINISTIC** - ?
  + **READS SQL DATA** – Pouze čte věci z databáze
  + **MODIFIES SQL DATA** – Modifikuje věci v databázi
  + **NO SQL** - Nemá žádné SQL příkazy
  + **CONTAINS SQL** - Má SQL příkazy, ale žádné které by zasahovali do DB
* 
*   
  

**Systémové funkce**

* Většina užitečných funkcí je již před-vytvořena
* Funkce jsou buďto **SINGLE-ROW** *(funkce se provede na každý řádek, vrací více hodnot)*, či **MULTI-ROW** *(zpracovává více hodnot naráz, vrací jednu hodnotu)*
* Nejdůležitější jsou:
  + Agregátní
    - **MIN()** - Nalezne minimální hodnotu sloupce
    - **MAX()** - Nalezne maximální hodnotu sloupce
    - **AVG()** - Vypočítá průměrnou hodnotu funkce
    - …
  + Porovnávací
    - **COALESCE()** - Postupně kontroluje LIST hodnot, zda nejsou NULL, dokud nenarazí na neNULLovou hodnotu
    - **ISNULL()** - Zkontroluje, zda je hodnota NULL
    - …
  + Control-flow
    - **CASE** - “Hrábě” možností, které postupně kontroluje kondice, a až nalezne kondici která platí, provede část kódu
    - **IF()** - Zkontroluje kondici a pokud platí, provede část kódu
    - **IFNULL(expr1, expr2)** - Zkontroluje zda je expr1 NULL, pokud ne, vrací expr1, pokud ano, vrací expr2
    - …
  + Datumové
    - **CURDATE()** - Vrátí dnešní datum v YYYY-MM-DD formátu
    - **DATEDIFF()** - Porovná dva datumy a vrátí jejich rozdíl
    - …
  + Stringové
    - **LENGTH()** - Vrátí velikost řetězce
    - **CONCAT()** - Překonvertuje hodnoty na řetězce a spojí je
    - …
  + Matematické
    - **ABS()** - Absolutní hodnota
    - **ROUND()** - Zaokrouhluje
    - ...