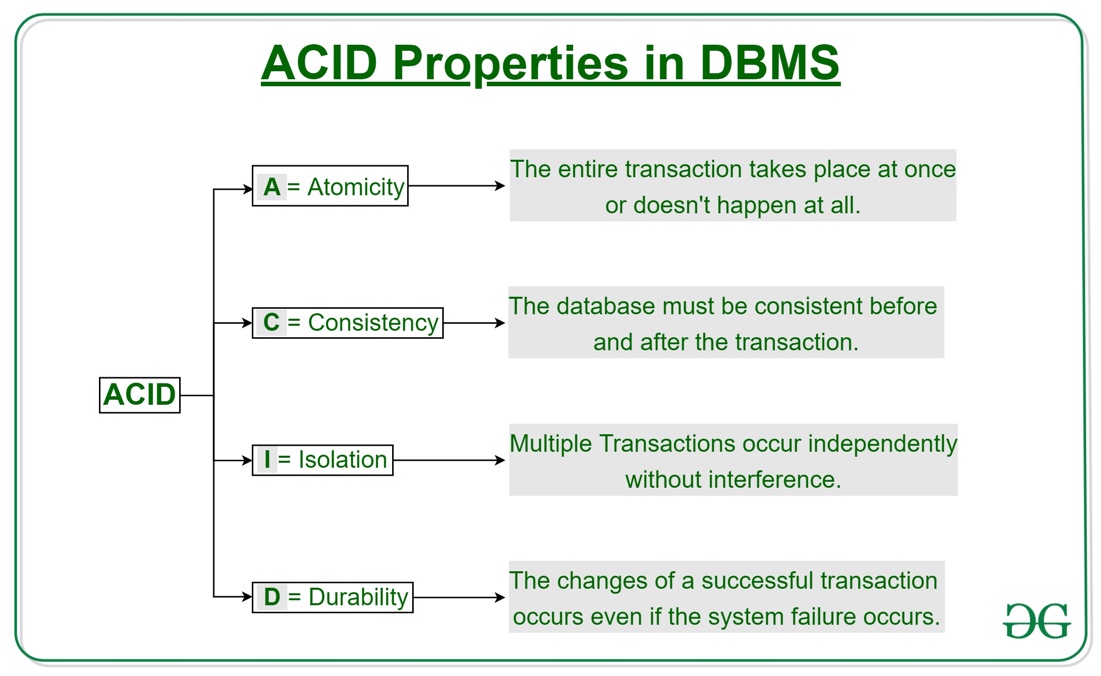
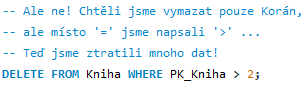
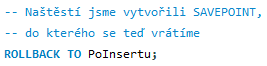
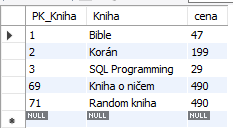
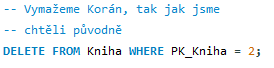
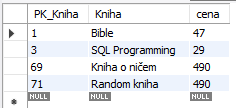
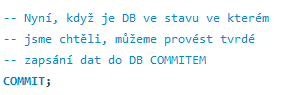
**15. Transakce a transakční zpracování (ACID)**

**ACID**

* **A**tomicity, **C**onsistency, **I**solation, **D**urability
* Transakce musí splňovat všechny ACID vlastnosti, jinak je narušený běh a správnost databáze.
* Atomicity *(Atomicita)*
  + Celá transakce se buďto provede celá, nebo vůbec.
  + Představme si, že se provádí převod peněz:
    - Účet A zašle převod
    - Z Účtu A se odeberou peníze
    - *Něco špatného se stane v DB*
    - Na Účet B peníze nepřišly
  + Řekněme, že takto by se databáze zachovala.
  + Transakce se neprovedla celá a ani se neprovedla vůbec
  + Tudíž nesplňuje ATOMICITU.
* Consistency *(Konzistence)*
  + Integritní omezení musí být zanechány.
  + Představme si, že se vytváří nový žánr knihy:
    - Vytvoří se nový žánr a k tomu i její PK
    - Vytvoří se nová kniha a přiřadí se k ní FK na PK nového žánru
    - *Něco špatného se stane v DB*
    - Nový žánr zmizí a kniha s FK odkazující na jeho PK zůstane
  + Řekněme, že takto by se databáze zachovala.
  + FK na žánr odkazuje na PK, který neexistuje.
  + Tudíž nesplňuje KONZISTENCI.
* Isolation *(Izolace)*
  + Každá transakce je od jiné izolována, aby mezi sebou nenarušovali data. *(Spuštěny sériově za sebou, pokud zasahují do stejných hodnot)*
  + Představme si, že vkládáme peníze a převádíme je:
    - Peníze převádíme -> (100Kč-50Kč) Již se zapsali příjemci, ale ještě se neodebrali od nás
    - V tomto mezi-stavu peníze vkládáme -> (100Kč+100Kč)
    - Peníze se od nás odebrali a **zapsali** do DB (50Kč)
    - Peníze byly vloženy a **zapsány** do DB, ale se špatnými daty, jelikož nebyli nejdříve odebrány peníze od nás, tudíž máme na účtu teď 200Kč. (100Kč+100Kč)
  + Řekněme, že takto by se databáze zachovala.
  + Přes to, že nyní máme mít na účtu 150 Kč, máme na účtu 200 Kč
  + Tudíž nesplňuje IZOLACI.
* Durability *(Odolnost)*
  + Jakmile je transakce dokončena, jakékoliv modifikace databáze a jejich dat jsou zapsány na disk a zůstávají tam tak i přes veškeré systémové a jiné chyby.
    - Představme si, že převádíme peníze:
      * Účet A převádí peníze na Účet B
      * Vše proběhne hladce a transakce je správně ukončena.
      * *Po jednom dni se stane něco špatného v DB*
      * Na disku jsou data před převodem peněz, či dokonce nejsou data žádná
    - Řekněme, že takto by se databáze zachovala.
    - Peníze se nepřevedly či se dokonce o veškeré peníze přišlo
    - Tudíž nesplňuje ODOLNOST.
* Pro databázi a transakce je důležité, aby se spouštěla a fungovala podle všech bodů **ACID**.



**Transakce - TCL**

* *Transaction Control Language*
* Zajišťuje transakce v databázi - při více DML příkazech se provedou buďto všechny, nebo žádná! *(A když se nepovede žádná, databáze se vrátí do původního stavu)*
  + Výhodné např. u bankovních převodů - Účet A chce převést peníze na Účet B.
  + Zašle platbu a databáze úspěšně odebere peníze z Účtu A.
  + Něco se nepovede a databáze před přidáním peněz na Účet B nečekaně spadne.
  + Nepoužívala se transakce - tudíž není databáze vrácena do původního stavu, a Účet A pouze ztratil své peníze, aniž by se připsali na Účet B.
* Používá se pro zajištění vrácení chyb uživatelů, či nečekaných neprovedených důležitých DML příkazů
* Příkazy:
  + **START TRANSACTION / BEGIN** – Zapne transakci
  + **COMMIT** - Pevně zapíše data do databáze *(Již nejde transakčně vrátit zpět, pouze manuálně!)*
  + **ROLLBACK [TO ‘savepoint’]** - Vrátí zpět databázi do původního stavu k poslednímu commitu či danému SAVEPOINTU
  + **SAVEPOINT [jmeno]** - Vytvoří SAVEPOINT, ke kterému se ROLLBACK může vrátit.
* Většina databází má transakce DEFAULTNĚ vypnuté a COMMIT se použije ihned po každém DML příkazu.
* 
* 
* 
*   
  
*   
  
*   
  
* 
* 