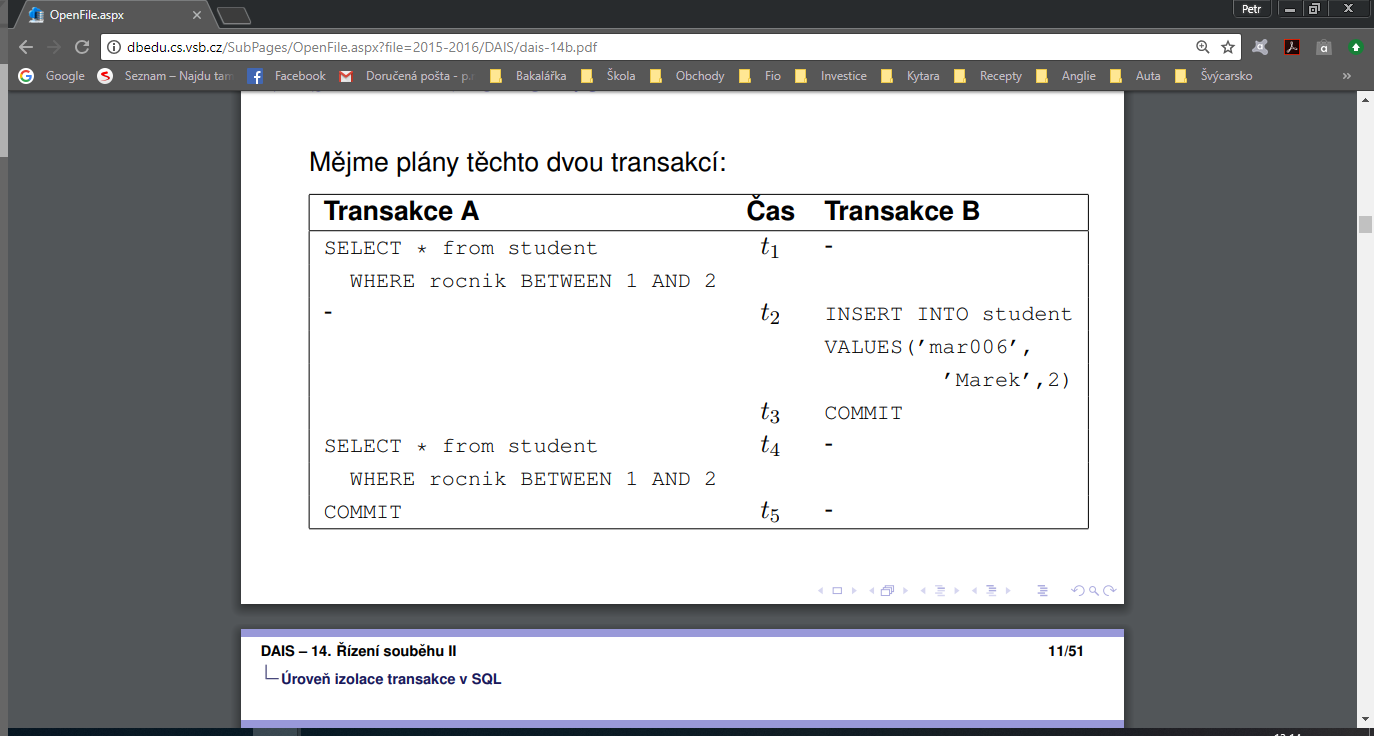
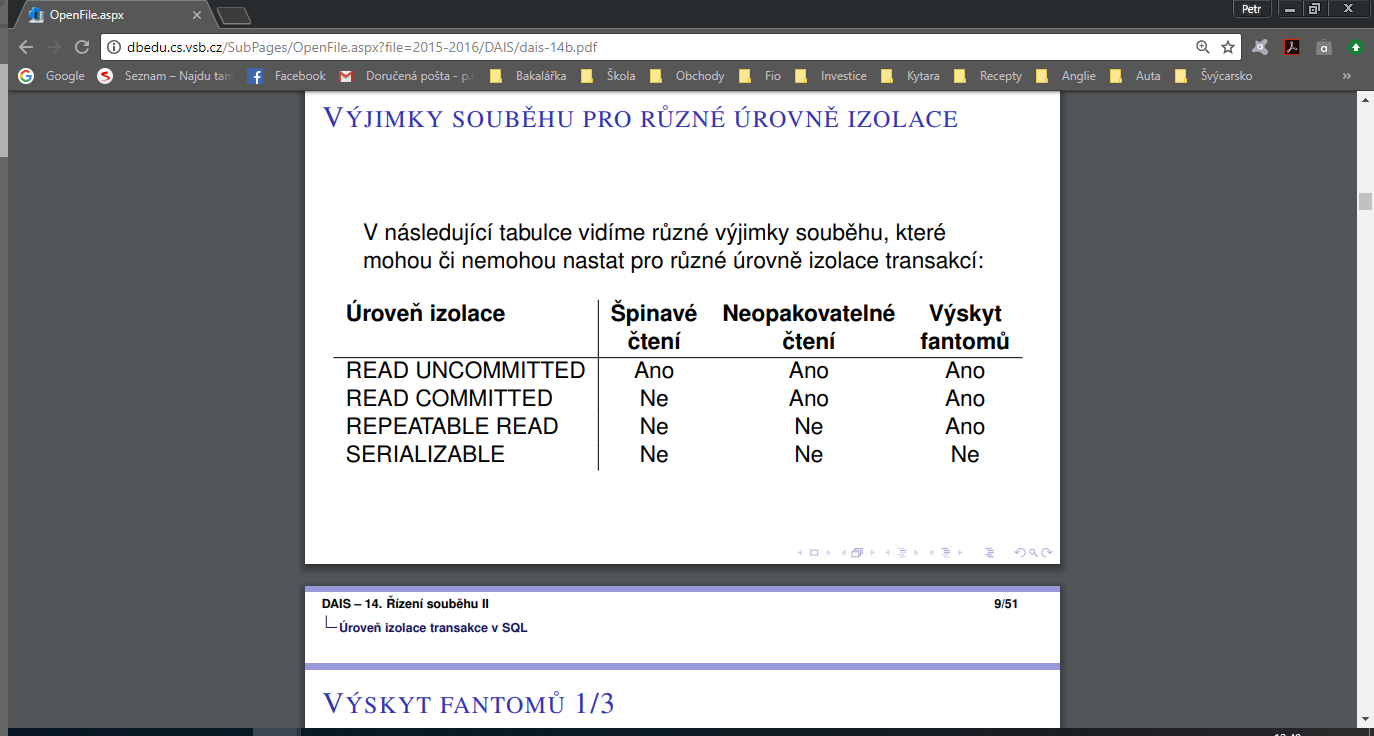
**15. Řízení souběhu II**

* Co se může stát u uncommited read a ne u commited read izolace transakce? Popište na příkladu
* Popište problém souběhu zvaný výskyt fantomů, uveďte i příklad
  + Jedná se o problém kdy dvě transakce běžící v jednu chvíli se ovlivňují, transakce A přečte počet řádků, transakce B provede INSERT/DELETE, později přečte transakce A jiný počet řádků, než dříve
  + Může nastat, pokud je povolena úroveň REPEATABLE READ a nižší
  + T1 – A: Select count(\*) from user -> 2 záznamy
  + T2 – B: Insert into user values(2, 2);
  + T3 – B: COMMIT
  + T4 – A: Select count(\*) from user -> 3 záznamy
  + 

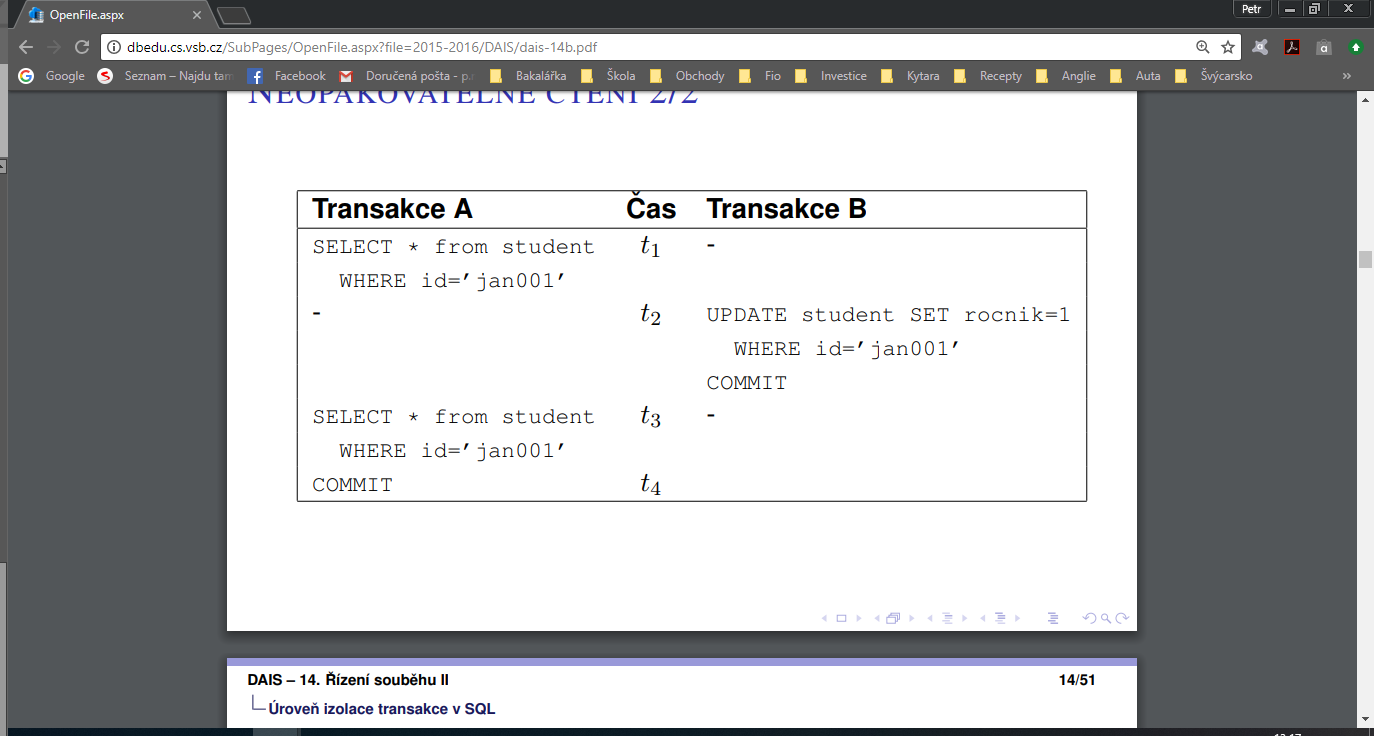
**Úroveň izolace transakce**

* Serializovatelnost garantuje izolaci transakcí ve smyslu podmínky ACID
* Pokud je plán transakcí serializovatelný, pak se neprojevují negativní vlivy souběhu
* Izolovanost ale znamená menší výkon souběhu
* SŘBD umožňují nastavit úroveň izolace:
  + READ UNCOMMITED
  + READ COMMITED
  + REPEATABLE READ
  + SERIALIZABLE (nejvyšší úroveň)
* Vyšší úroveň značí vyšší míru izolace, ale nižší propustnost
* 

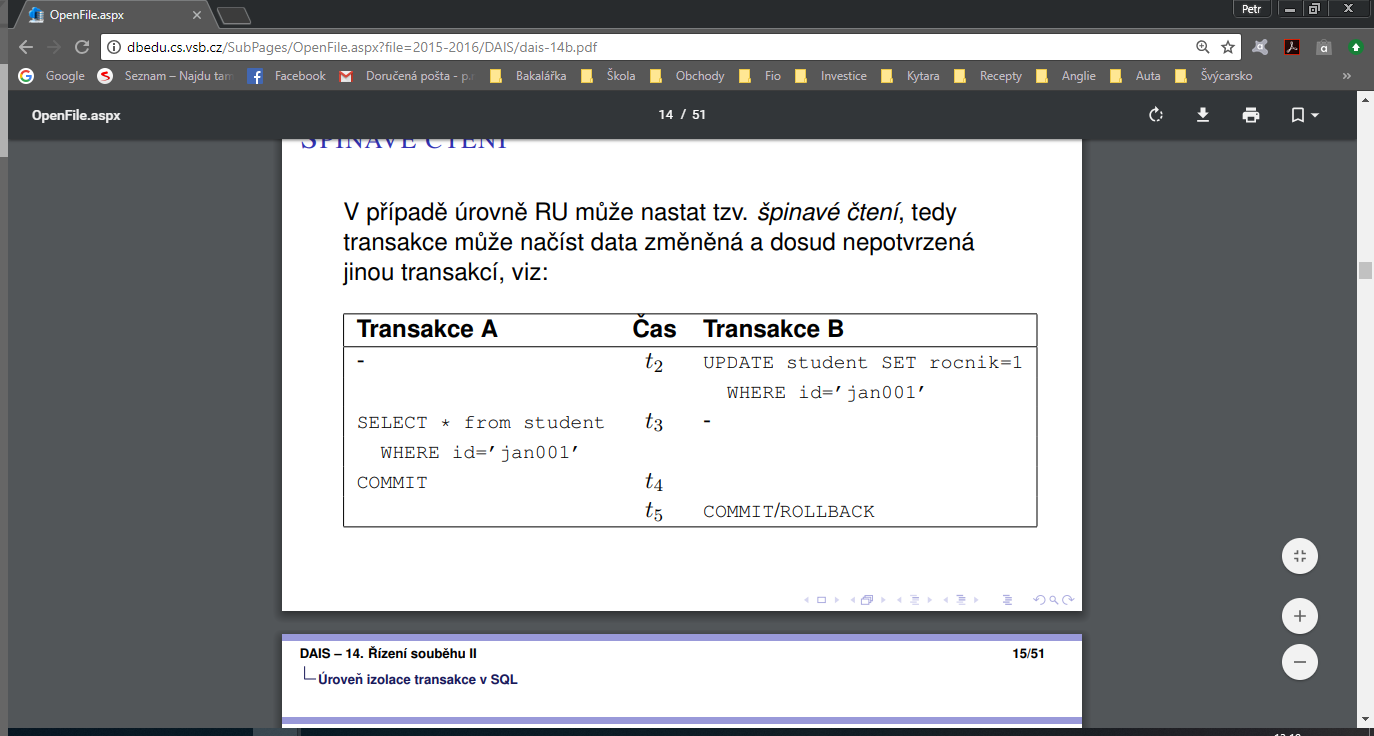
**Výskyt fantomů**

* Viz. Otázka nahoře

**Neopakovatelné čtení**

* Úroveň READ COMMITED a nižší
* Sdílené zámky můžou být uvolněny před ukončením transakce
* Výlučné zámky jsou uvolněny až na konci transakce
* 

**Špinavé čtení**

* Úroveň READ UNCOMMITED
* Transakce může načíst data změněná v dosud nepotvrzené jiné transakci
* 

**Syntaxe transakcí**

START TRANSACTION ISOLATION LEVEL <izolace> (READ UNCOMMITED, READ COMMITED, RR., SR.)

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

**Granularita zámků**

* Můžeme uzamykat např. hodnoty atributů, záznamů, tabulek..
* Zvýšení propustnosti DB systému
* Zamknutím menších objektů zlepšíme míru paralelizace (jemná granularita)
* Zamknutí větších objektů znamená menší počet zámků a teda menší režii zamykání (hrubá granularita)
* Např. zamknutím celé DB zamezíme paralelnímu přístupu
* Tohle téma jsem přeskočil…

**Správa verzí**

* Optimistický přístup
* Předpokládáme, že paralelní transakce se nebudou vzájemně ovlivňovat
* Nevýhoda: velká interní režie – zvýšený požadavek na paměť při správě kopií dat
* Výhoda: efektivnější (než zamykání) pokud převažují READ operace

**Úroveň READ COMMITED**

* Vhodná v případě prostředí s málo konflikty souběhu
* Např. pokud dotaz prochází celou tabulku s milionem záznamů a další transakce aktualizuje záznam 150 000, pak původní transakce nevidí tuto změnu

**Úroveň SERIALIZABLE**

* Každý dotaz spuštěný transakcí vidí data potvrzená před začátkem transakce a změny provedené samotnou transakcí
* Vhodná pro:
  + velké DB a krátké transakce
  + dlouhé transakce používající čtení

**Úroveň READ ONLY**

* vhodná pro generování sestav, u kterých požadujeme konzistentní obsah
* Transakce nesmí aktualizovat data

**Zámky tabulek**

* ROW SHARE, ROW EXCLUSIVE, SHARE, SHARE ROW EXCLUSIVE, EXCLUSIVE

**Explicitní zamykání**

* Programátor žádá o přidělení zámku explicitně
* Např. LOCK TABLE, SELECT FOR UPDATE