**14. Řízení souběhu I**

* Popsat řešení souběhu pomocí zamykání u ztráty aktualizace
* Na dvou souběžných plánech popište, jak se pomocí uzamykání řeší problém ztráty aktualizace

**Sériový a serializovatelný plán**

* **Sériový plán**
  + transakce jsou provedeny za sebou (sériově)
  + Zapisujeme jako entici uspořádanou dle vykonávání transakcí. (A, B) – A je vykonána před B
* **Serializovatelný plán**
  + Plán vykonávání dvou transakcí je korektní tehdy a jen tehdy pokud je serializovatelný: plán je ekvivalentní s výsledkem libovolného sériového plánu
  + přísné dvoufázové zamykání zaručuje, že plán bude vždy serializovatelný

**14. Řízení souběhu I**

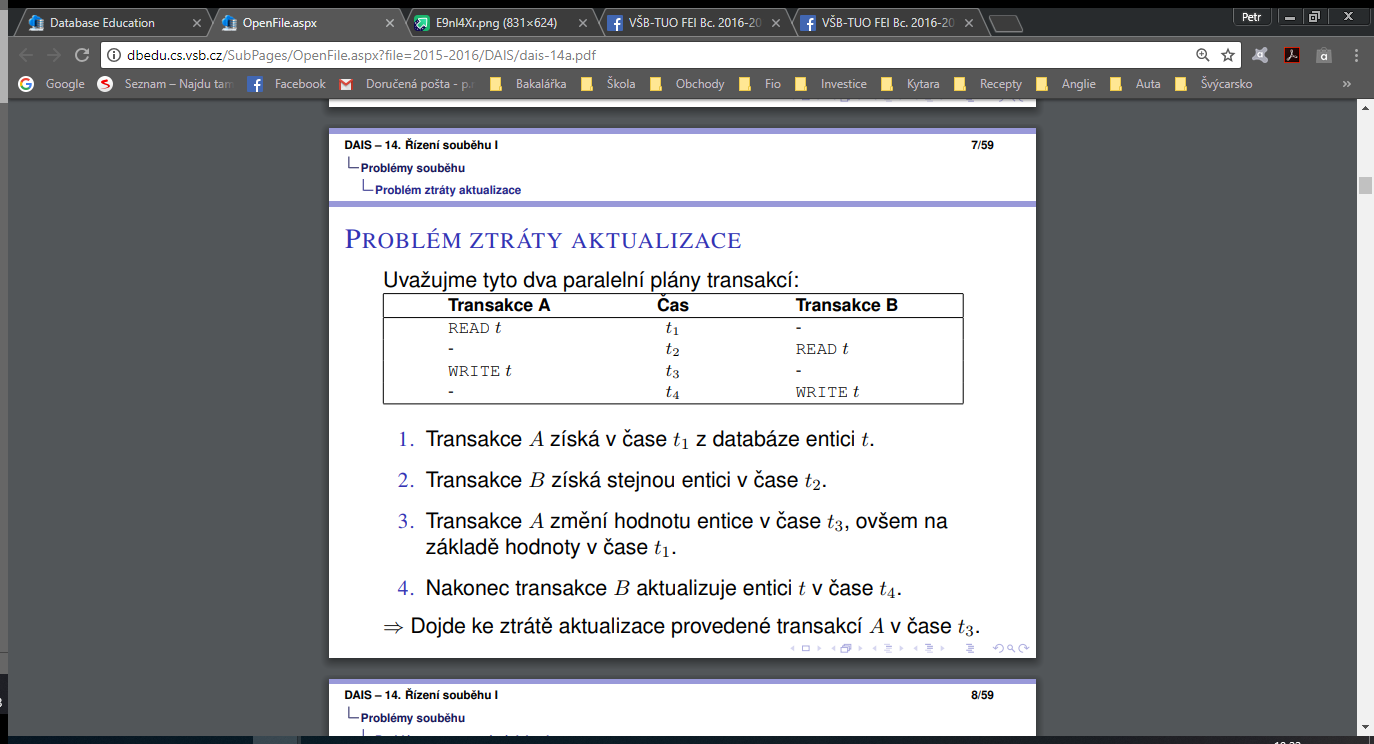
**Klasifikace DB systémů**

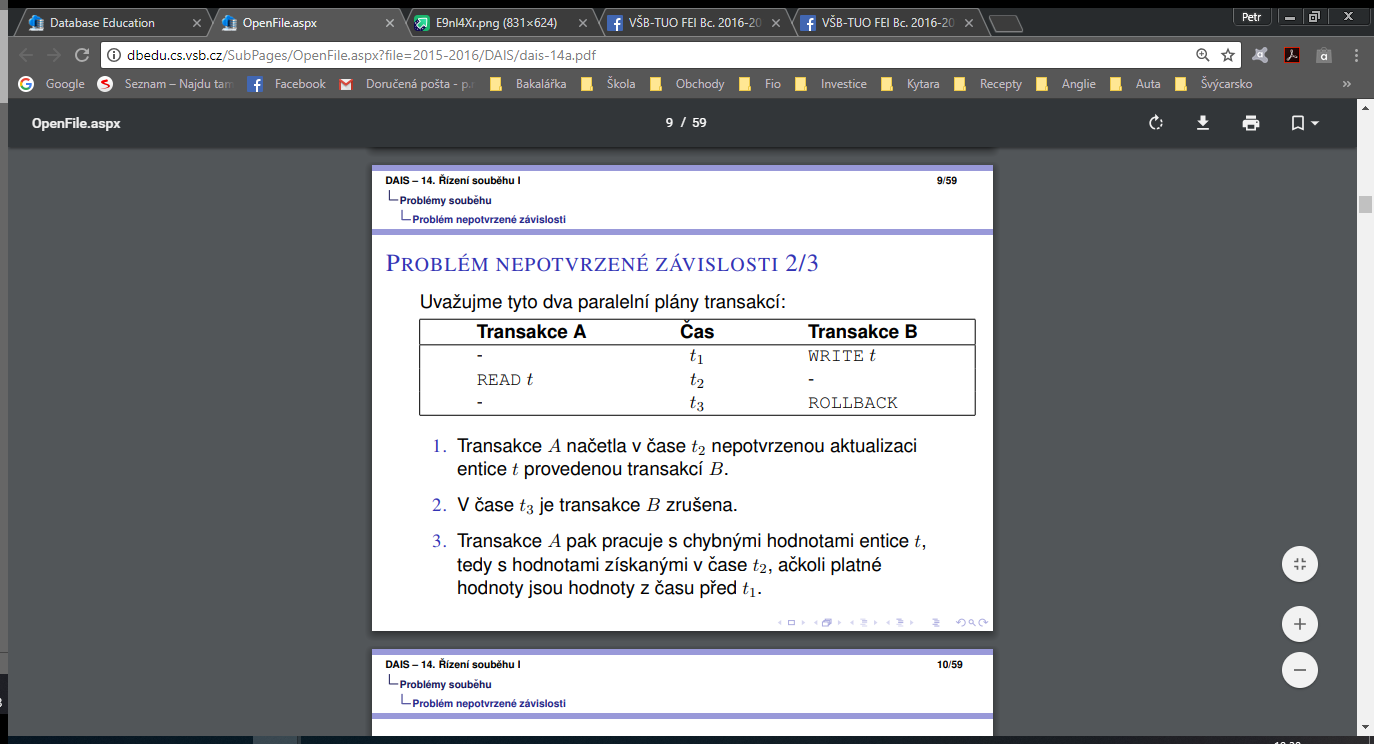
* Jednouživatelský
* Víceuživatelský – v tomto případě mluvíme o souběhu
  + Souběh umožňuje SŘBD zpřístupnit databázi mnoha transakcím ve stejném čase

**Posloupnost operací transakcí se nazývá plán**

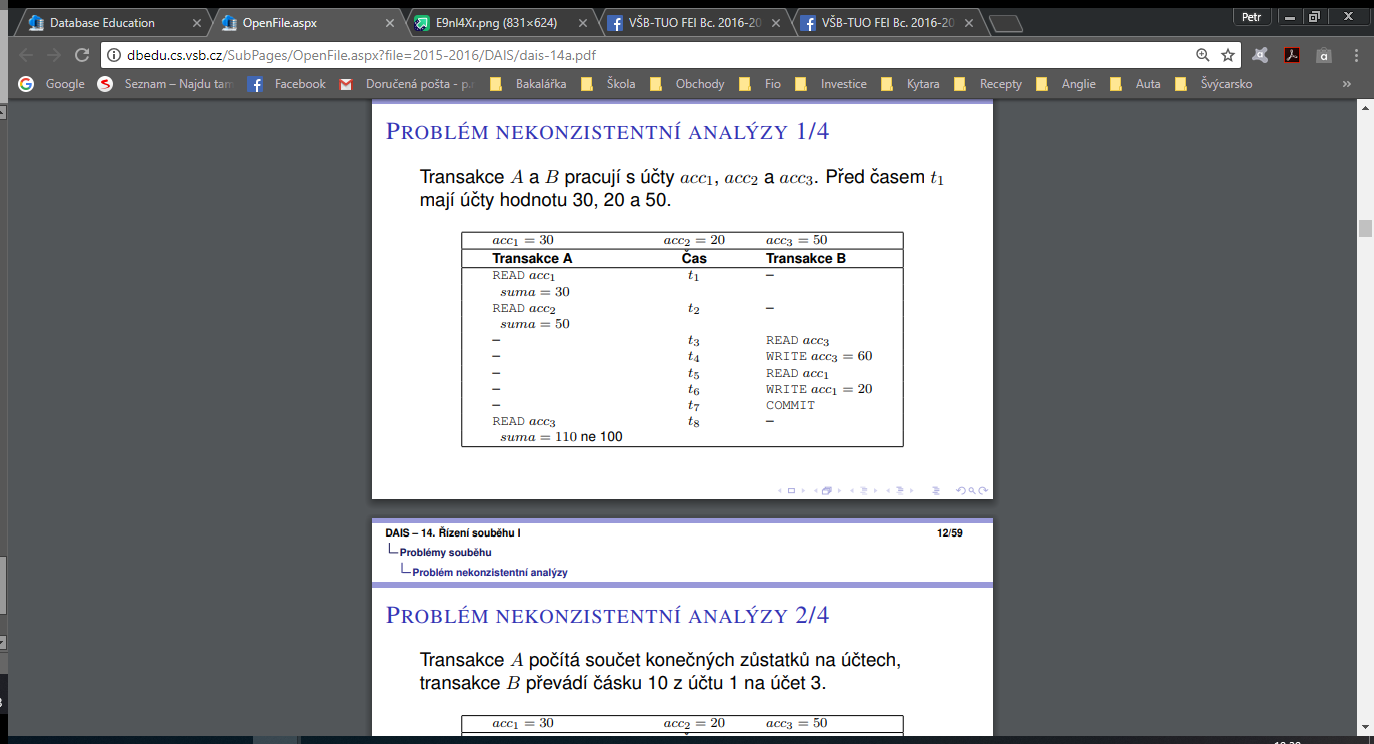
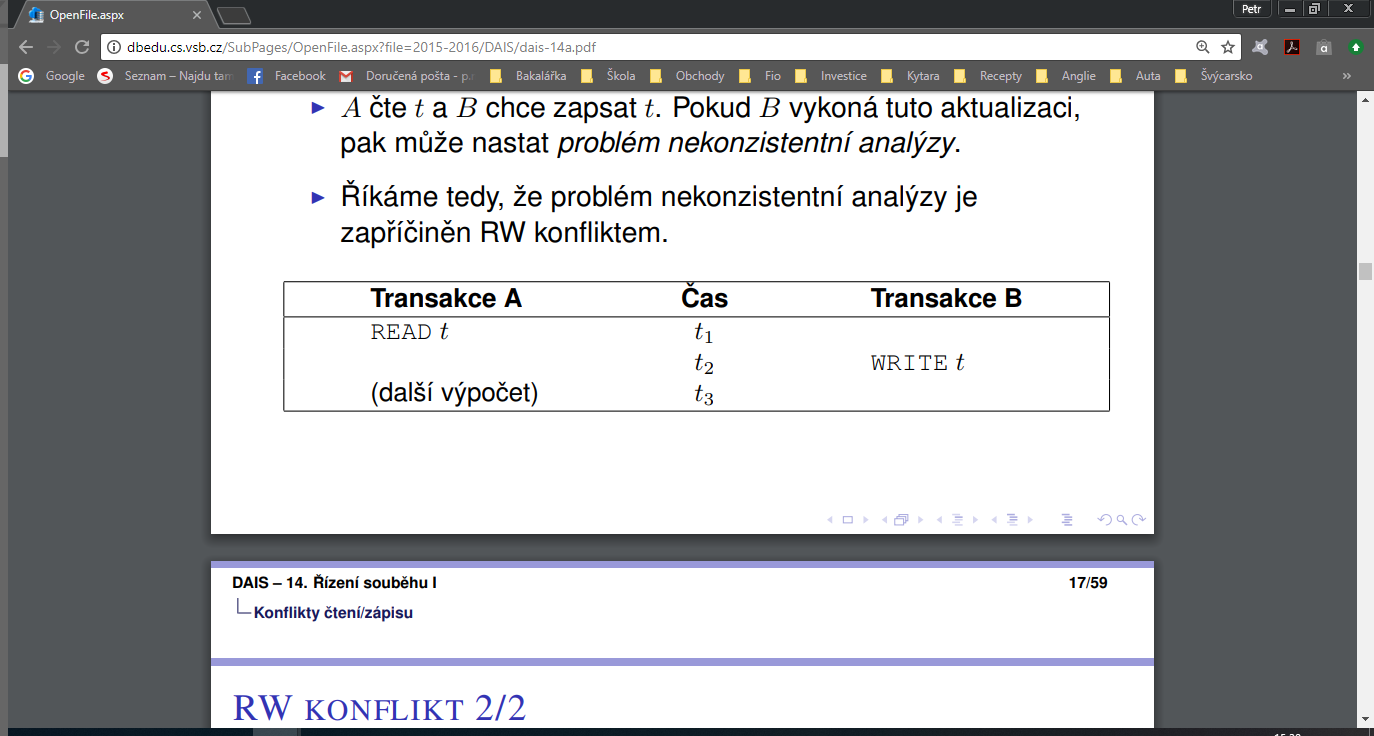
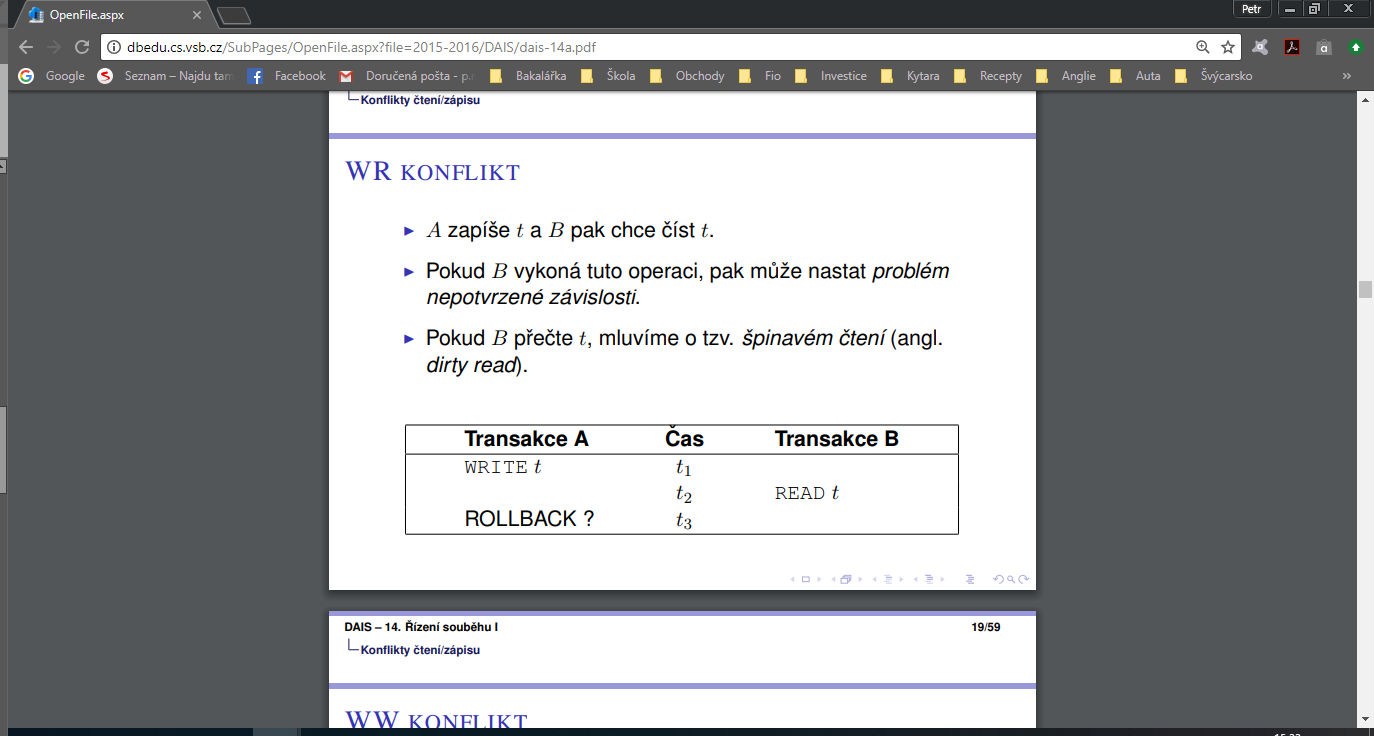
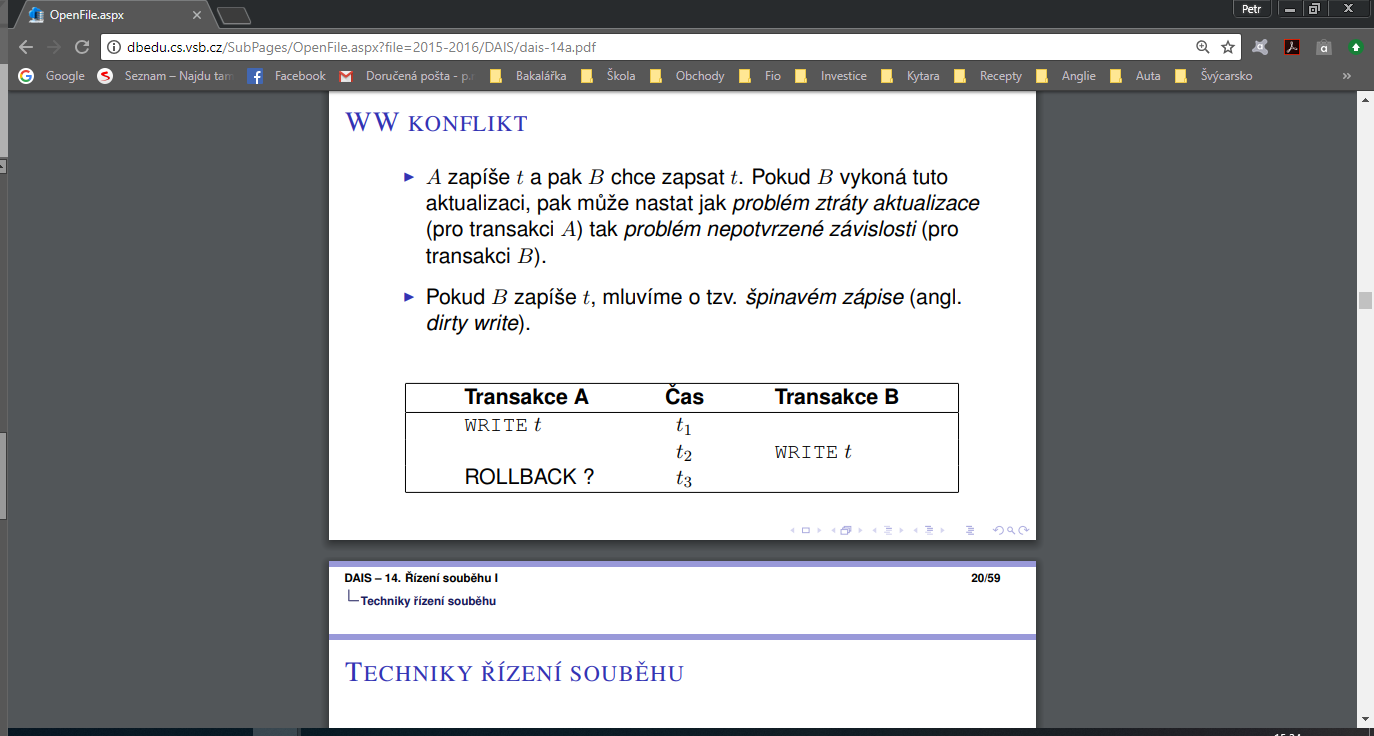
**Problémy souběhu**

* **Ztráta aktualizace**



* + Transakce A získá v čase t1 z DB entici t
  + B získá stejnou entici v t2
  + A změní hodnotu entice v t3 na základě hodnoty v t1
  + B aktualizuje entici t v čase t4 -> Dojde ke ztrátě aktualizace provedené A v t3
* **Nepotvrzená závislost**
  + A načetla v t2 nepotvrzenou aktualizaci entice t provedenou transakcí B
  + V t3 je transakce B zrušena
  + A pak pracuje s chybnými hodnotami entice t, ačkoliv platné hodnoty jsou z času před t1

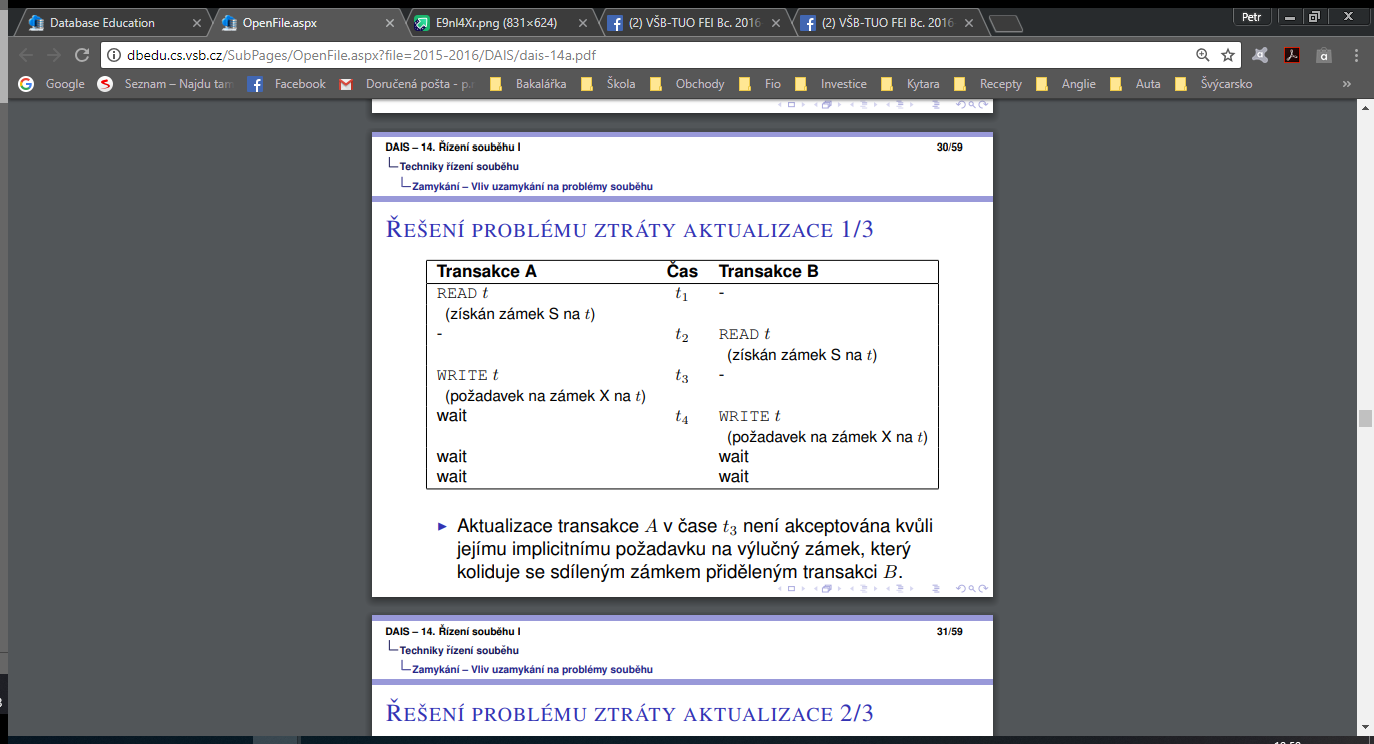
V případě, že u A je místo READ – WRITE:

* + A se stala závislou na nepotvrzené změně z času t1
  + Došlo ke ztrátě aktualizace provedení v t2, ROLLBACK z t3 způsobí nastavení entice na hodnoty z času před t1
* **Nekonzistentní analýza**
  + A počítá sumu zůstatků, B převádí 10 z účtu 1 na 3
  + Transakce A má k dispozici nekonzistentní DB a vykoná nekonzistentní analýzu
* **Konflikty čtení/zápisu**
  + problém nastává pokud dvě transakce A a B chtějí číst nebo zapisovat stejnou entici
  + **RW konflikt**
    - A čte t a B chce zapsat t. Pokud B vykoná tuto aktualizaci, pak může nastat problém nekonzistentní analýzy
    - 
    - Pokud B vykoná aktualizace a A načte t znovu, pak A získá odlišné hodnoty = neopakovatelné čtení
      * Obrázek stejný, místo (další výpočet) je READ t
  + **WR konflikt**
    - A zapíše t a B pak chce číst t
    - Pokud B vykoná tuto operaci, pak může nastat problém nepotvrzené závislosti
    - Pokud B přečte t, mluvíme o tzv. špinavém čtení (dirty read)
    - 
  + **WW konflikt**
    - A zapíše t a pak B chce zapsat t. Pokud B vykoná tuto aktualizaci, pak může nastat jak problém ztráty aktualizace (pro A) tak problém nepotvrzené závislosti (pro B)
    - Pokud B zapíše t, mluvíme o tzv. špinavém zápise (dirty write)
    - 

**Techniky řízení souběhu**

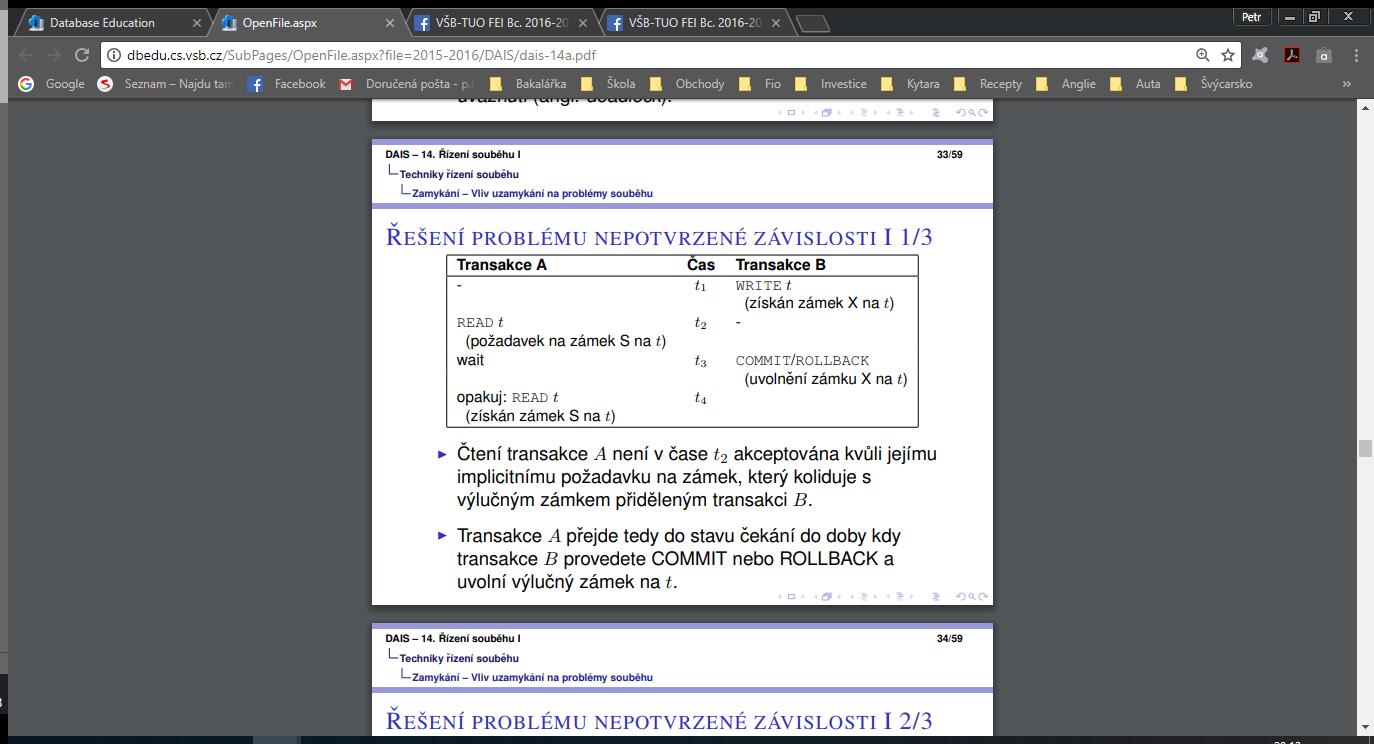
* **Zamykání**
  + Pokud transakce A chce provést čtení nebo zápis nějakého objektu v DB, pak požádá o zámek na tento objekt
  + **Význam zámků** – žádná jiná paralelní transakce nemůže získat získat zámek a nemůže tedy provést čtení či aktualizaci do doby, než A zámek uvolní
  + **Výlučný zámek X, Sdílený zámek S**
  + **Uzamykací protokol (dvoufázové uzamykání):**
  + Transakce, která chce získat entici z DB, musí požádat o sdílený zámek
  + -||- aktualizovat entici musí požadovat výlučný zámek
  + Pokud zámek požadovaný transakcí B nemůže být přidělen okamžitě, pak B přejde do stavu čekání
  + Zámky jsou uvolněny na konci transakce

**Řešení problému ztráty aktualizace při uzamykání**

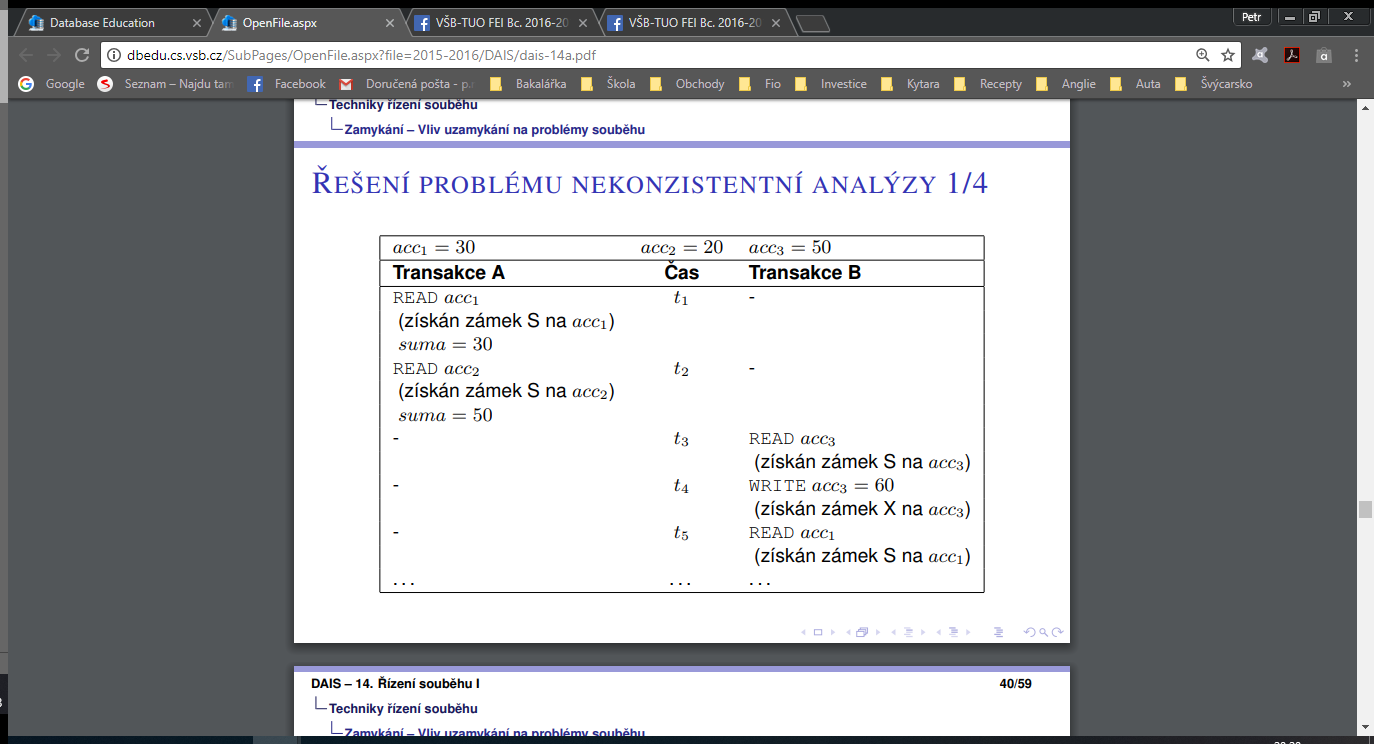


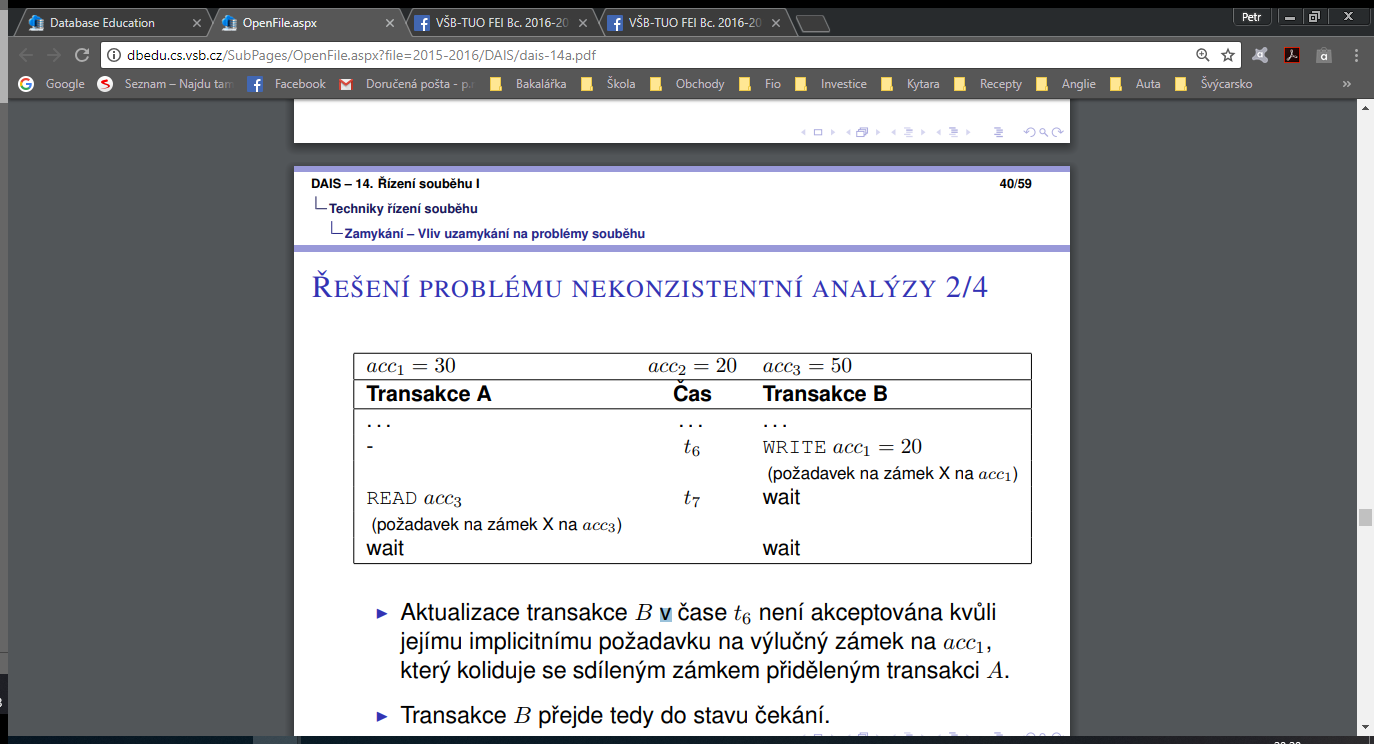
* Aktualizace transakce A v čase t3 není akceptování kvůli požadavku na výlučný zámek, který koliduje se sdíleným zámkem přiděleným transakci B
* A přejde do čekání. B také přejde do čekání v t4 ze stejného důvodu -> nastává uváznutí

**Řešení problému nepotvrzené závislosti I**



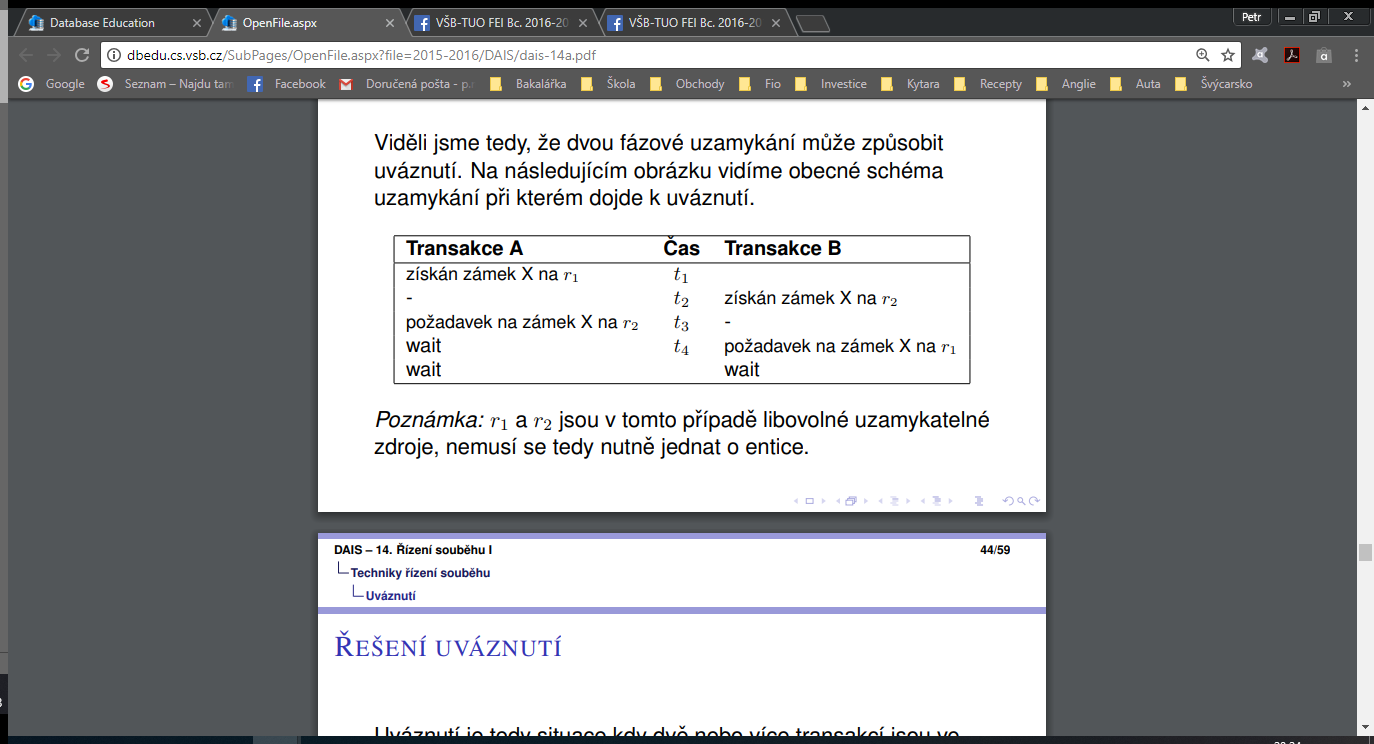
* Čtení A není v čase T2 akceptování kvůli implicitnímu požadavku na zámek, který koliduje s výlučným zámkem přiděleným transakci B
* A přejde do čekání do doby, než B provede COMMIT nebo ROLLBACK a uvolní výlučný zámek
* Po uvolnění, A pokračuje. Pokud B provede COMMIT – A pracuje s hodnotami z času t1, pokud B provede ROLLBACK, pak A pracuje s hodnotami před časem t1
* II. – u A místo READ – WRITE, princip stejný

**Řešení problému nekonzistentní analýzy**



* Aktualizace B v čase t6 není akceptování kvůli požadavku na výlučný zámek na acc1, který koliduje se sdíleným zámkem přiděleným transakci A. B přejde do čekání
* Čtení A v čase t7 není akceptováno kvůli požadavku na sdílený zámek na acc3. A přejde do čekání
* Nastalo uváznutí

**Uváznutí**



* Je situace, kdy více transakcí jsou ve stavu čekání a čekají na uvolnění zámků
* Řešení
  + Detekce uváznutí – nastavení časových limitů, detekce cyklu v grafu Wait-For
  + Prevence uváznutí pomocí časových razítek
* **Nastavení časových limitů:**
  + Systém předpokládá, že transakce může trvat nejdéle nějakou dobu
  + Pokud trvá déle, systém detekuje uváznutí
* **Detekce cyklu v grafu:**
  + Efektivnější
  + Zaznamenává jaké transakce na sebe vzájemně čekají
  + Řešení spočívá ve výběru jedné z uváznutých transakcí a provedení ROLLBACK (uvolní zámky). Ostatní transakce pokračují v činnosti
  + Zrušená transakce je spuštěna znovu nebo vygenerována výjimka o zrušení transakce
* **Řešení uváznutí**
  + Dvě verze protokolu: Wait-Die, Wound-Wait
  + Každé transakci je přiděleno časové razítko – čas začátku transakce.
  + Pokud A požaduje zámek na entici, která je uzamčena transakcí B, pak:
    - **Wait-Die:** pokud A je starší než B, pak A čeká. Pokud A je mladší, A je zrušena ROLLBACKem a spuštěna znovu (zemře)
    - **Wound-Wait**: pokud A je mladší, A čeká. Pokud A je starší, B je zrušena ROLLBACKem a spuštěna znovu (wound)

**Sériový a serializovatelný plán**

* **Sériový plán** – transakce jsou provedeny za sebou. Zapisujeme jako entici uspořádanou dle vykonávání transakcí. (A, B) – A je vykonána před B
* **Serializovatelný plán**
  + Plán vykonávání dvou transakcí je korektní tehdy a jen tehdy pokud je serializovatelný: plán je ekvivalentní s výsledkem libovolného sériového plánu