Procesy, pokročilé modely procesů, cesta k workflow

INTEGRITA A KONZISTENCE

Databázová integrita

- databáze vyhovuje zadaným pravidlům – ***integritním omezením (IO)*.** Tato integritní omezení bývají nejčastěji součástí definice databáze a za jejich splnění zodpovídá **systém řízení báze dat (SŘBD)**
- mohou být zadána výrazem (*deklarativně*) nebo programem (*procedurálně*)
- Integritní omezení se mohou týkat *jednotlivých hodnot* vkládaných do polí databáze (například známka z předmětu musí být v rozsahu 1 až 5)

zdroj: http://cs.wikipedia.org/wiki/Databázová_integrita

Databázová integrita

- může jít o podmínku na *kombinaci hodnot* v některých polích jednoho záznamu (například datum narození nesmí být pozdější než datum úmrtí).
- může se týkat *i celé množiny záznamů daného typu*
- může jít o požadavek na *unikátnost hodnot daného pole či kombinace polí* v rámci celé množiny záznamů daného typu, které se v databázi vyskytují (například číslo průkazu v záznamech o osobách).

Integrita datovým typem

- *Datový typ* je množina hodnot spolu s operacemi, které je možné nad těmito hodnotami provádět.
- je vlastností jisté části modelu (proměnné, části jiného datového typu apod.) a *omezuje* její použití (jde vlastně o *integritní omezení* části modelu). Omezuje je tak, že tato část modelu může:
  - *nabývat pouze jisté množiny hodnot* a
  - *může s ní být prováděna pouze jistá omezená množina operací*.
- Výhodou zavedení datového typu pro jistou část modelu je zejména možnost kontrolovat (a to nejčastěji předem), zda se s touto částí zachází korektně (zda ukládaná hodnota je správná a použitá operace správně použita).

zdroj: studijní opora Pojem informačního systému, data, procesy, transakce

Konzistence

- DB musí splňovat všechna *integritní omezení* (IO)
- Udržování *interní konzistence* (redundantních, vícenásobně uložených/replikovaných dat v distribuovaných systémech)
- Dodržování *pravidel daných modelovaným systémem*

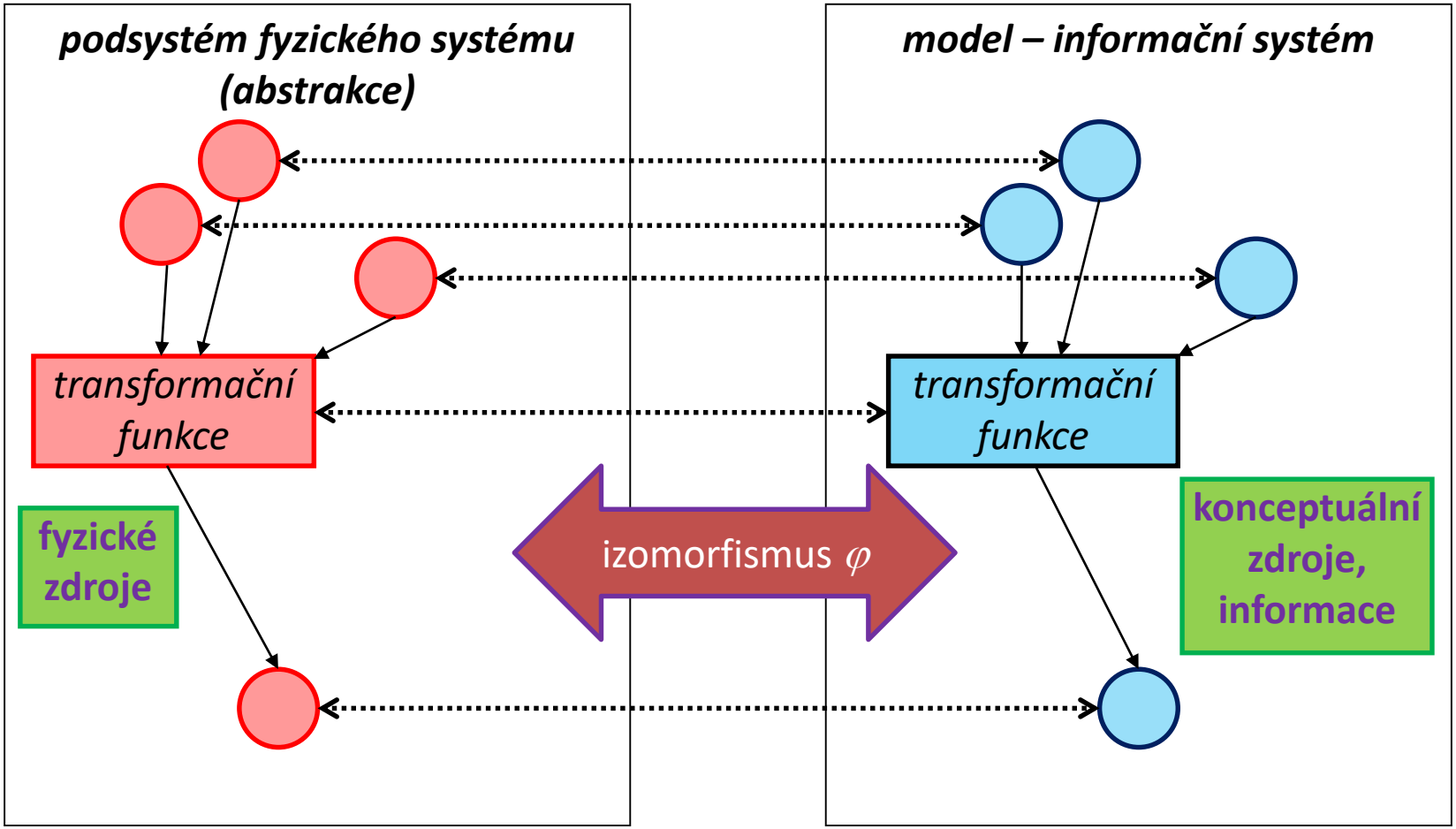
OLTP JAKO MODEL

Izomorfismus

- ***Izomorfismus*** je zobrazení mezi dvěma matematickými strukturami, které je vzájemně jednoznačné (bijektivní) a zachovává všechny vlastnosti touto strukturou definované.
- Jinými slovy, každému prvku první struktury odpovídá právě jeden prvek struktury druhé a toto přiřazení zachovává vztahy k ostatním prvkům.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

OLTP jako model



zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Nezbytnost abstrakce

- Není možné modelovat všechny zdroje i procesy fyzického systému. Vždy se vybírají jen ty, které jsou pro úroveň řízení, pro kterou OLTP budujeme, podstatné – modelujeme ***podsystém*** původního fyzického systému – ***abstrahujeme***
- OLTP je proto vždy modelem jisté ***abstrakce původního fyzického vzoru***.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

OLTP jako model

- říkáme, že OLTP modeluje nějaký fyzický podsystém.
- mezi OLTP a jeho fyzickým vzorem existuje izomorfismus φ
- pokud je v původním systému funkce nad zdroji, potom v OLTP existuje obraz této funkce pracující s obrazy zdrojů.
- pokud funkce v původním systému má za parametry jisté zdroje a dává jistý výsledek, pak obraz funkce v OLTP mající za parametry obrazy původních zdrojů dává za výsledek obraz původního výsledku.
- To platí i naopak.

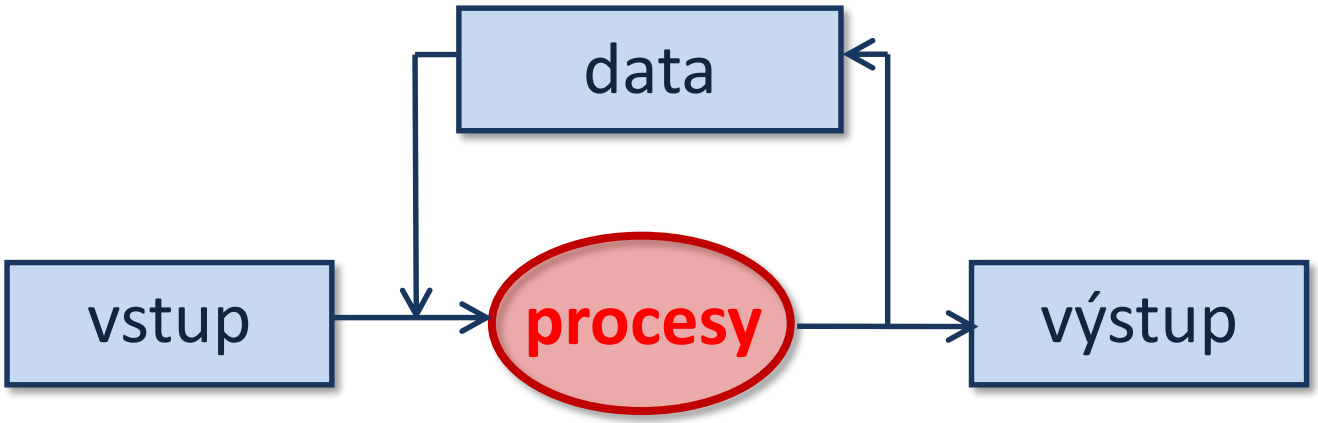
zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Příklad OLTP systému

- ve fyzickém systému se pracuje s peněžními zdroji, tj. ***skutečnými penězi***, pak se v informačním systému pracuje s jejich ***virtuálním obrazem***.
- pokud ve fyzickém systému je provedena funkce, která na základě objednávky vytvoří skutečnou fakturu, pak v informačním systému je vytvořen její obraz.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Procesy ve schématu informačního systému



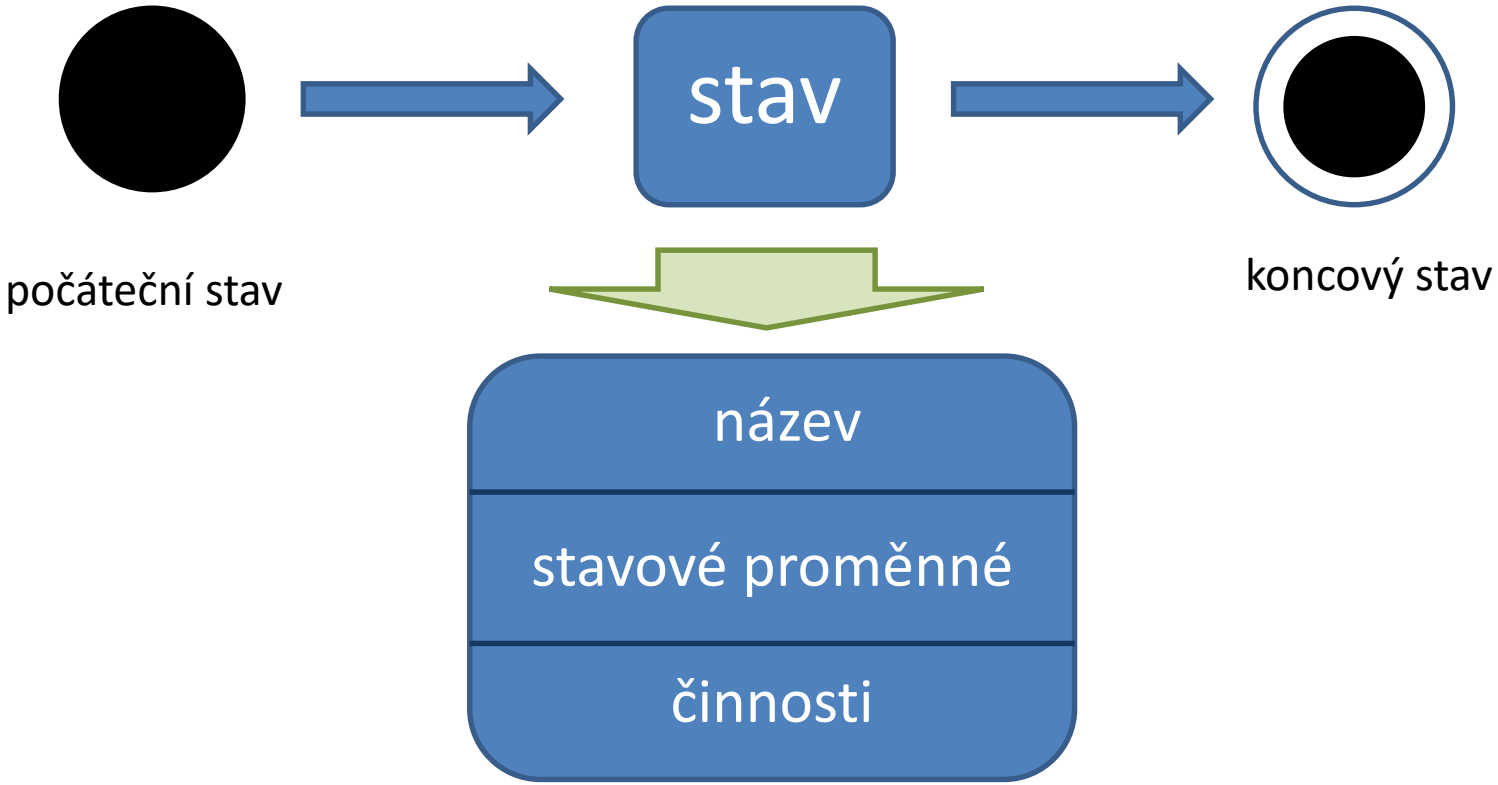
- ***data*** uchovávající ***stav*** systému a
- ***procesy*** realizující transformace často ve formě ***transakcí***.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

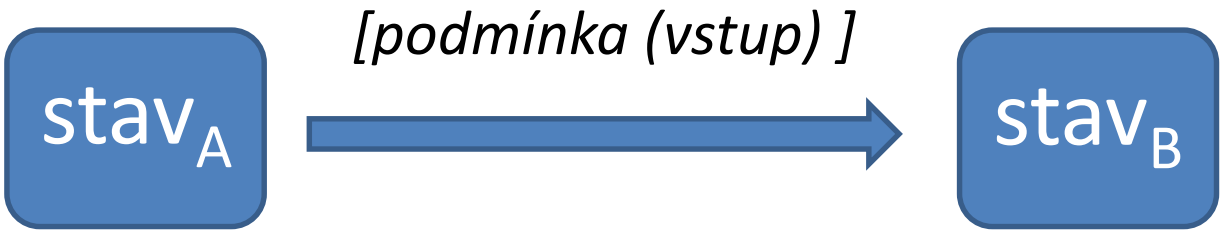
PROCESY A JEJICH DEFINICE

Stavový diagram UML

- Symboly UML používané ve stavových diagramech



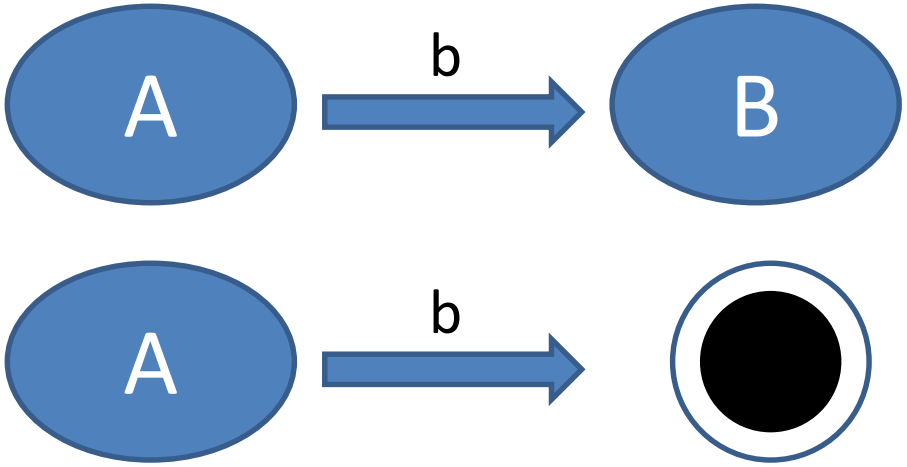
Strážní podmínky



Sekvenční procesy

- model = regulární gramatiky, konečné automaty

A -> bB nebo A->b



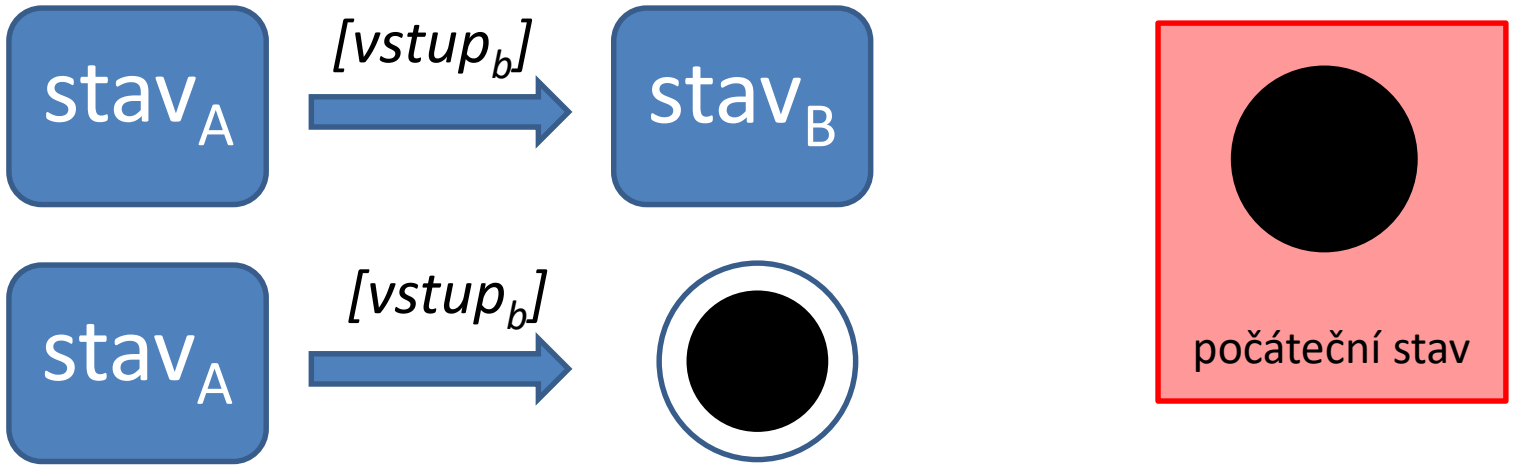
koncový stav

Sekvenční procesy

- stavový diagram (UML)

  $stav_A$ -> *[$vstup_b$]* $stav_B$

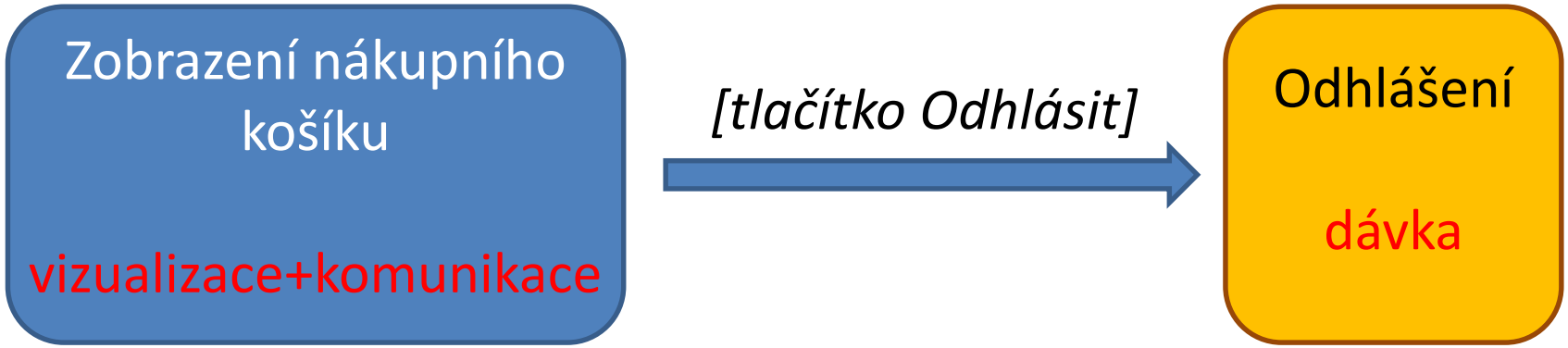
  $stav_A$ -> *[$vstup_b$]* (a konec)



- vstup může být i prázdný

Příklad

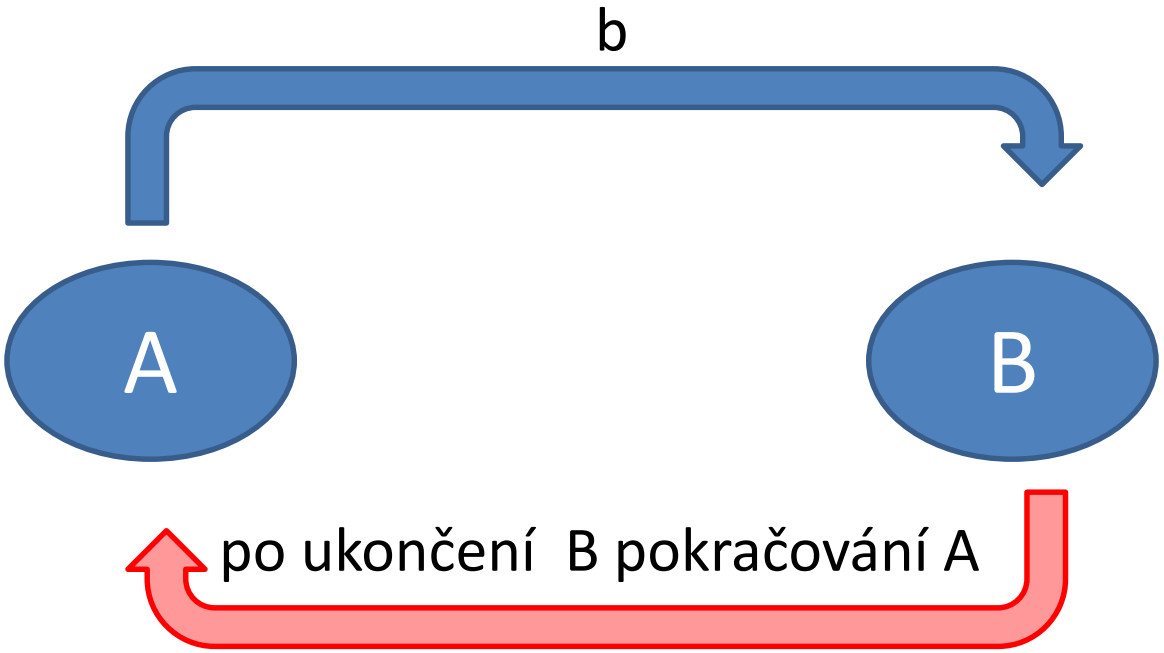
'Zobrazení nákupního košíku' -> *[tlačítko Odhlásit]*
'Odhlášení'



Hierarchické procesy

- model = bezkontextové gramatiky (speciální pravidlo), zásobníkový automat

  A -> b B pokračování$_A$

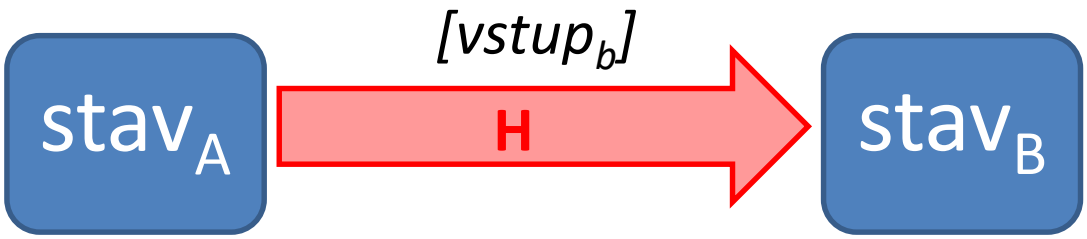


může pokračovat do libovolné hloubky zanoření

Hierarchické procesy

- stavový diagram (varianta ***UML***)

$stav_A$ -> *[*$vstup_b$*]* $stav_B$ pokračování$_A$



- Vstup musí být vždy označen, jinak může vzniknout nekonečný cyklus

Příklad

'Zobrazení nákupního košíku' -> *[tlačítko Přepočítat]*
'Úprava množství v košíku'

- v dávkových stavech se často vyskytují transakce

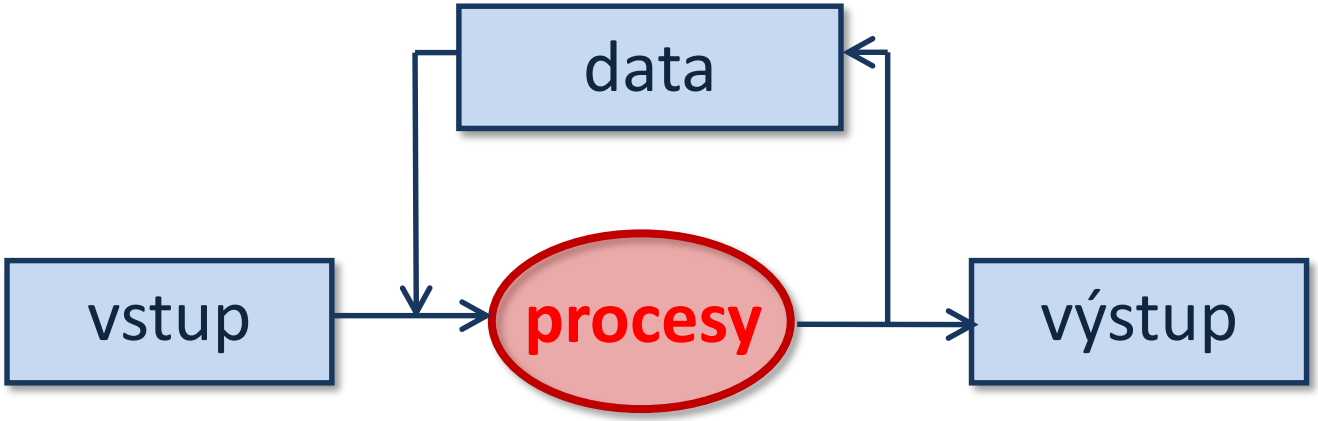
Obecné procesy

- model = neomezené gramatiky, Turingův stroj

  a A b -> c B d

- diagram neexistuje, popisuje se v nějakém programovacím jazyku

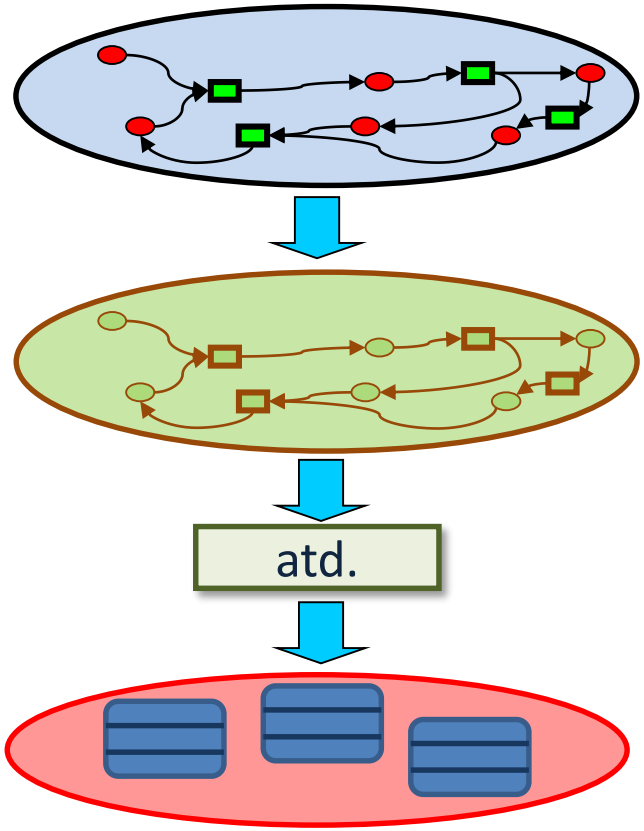
Procesy ve více úrovních



- typy ***procesů*** se liší využitím paměti (všechno jsou modelovány stavovými stroji s různým typem paměti - kontextu)
- pokud bychom považovali za paměť data IS v databázi, jde většinou o obecné procesy.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Řešení obecností dekompozicí



řízení - sekvence, hierarchie

dekompozice stavů

na dně hierarchie obvykle obecný proces - ***transakce***

(DATABÁZOVÉ) TRANSAKCE

Motivační otázky vzniku transakcí

- Co se stane, pokud dojde k poruše během práce s důležitými zdroji dat?
- Které operace prováděné nad daty systém stihl před poruchou skutečně provést a které ne?
- Co se stane, když více uživatelů současně bude modifikovat tentýž údaj?
- Budou údaje v databázi stále smysluplné ?
- Hledáním odpovědí a jejich aplikací při zajištění spolehlivosti se zabývají ***transakční modely*** a celý obor ***transakčního zpracování***.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Pojem transakce

- ***Transakce*** představuje jednotku práce vykonávanou v databázovém (informačním nebo podobném) systému nad databází a zpracovávanou ***souvislým a bezpečným způsobem nezávisle na jiných transakcích***. Transakce v databázovém prostředí má dva základní účely:
  - Poskytnout bezpečnou jednotku práce, která dovoluje správné ***zotavení z poruch*** a udržuje databázi v konzistentním stavu i v případě poruchy systému, když je zastaveno provádění (úplně nebo částečně) a některé operace nad databází zůstávají nedokončené nebo v nejistém stavu
  - Poskytnout ***izolaci programům přistupujícím k databázi současně***. Pokud tato izolace není poskytnuta, výstupy programů jsou potenciálně chybové.

zdroj: Wikipedie: Database transaction

Pojem transakce

- ***skupina operací*** (akcí) prováděných jako celek (buď celá dávka nebo nic)
- modelování stavu popisovaného výseku reálného světa
  - popis a provádění nerozlučných příkazů
  - první historické zmínky – 60. léta
  - důležitý pojem v oblasti databází
- ***Transakce*** je speciální druh programu, který je spouštěn v aplikaci ***OLTP*** (On Line Transaction Processing)

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Systém pro zpracování transakcí - TPS

- systém (platforma, databázový systém) ***podporující provádění transakcí*** – transakční systém
- zajišťuje speciální ***vlastnosti transakcí*** (atomičnost, nezávislost, trvanlivost)

- angl. ***Transactional Processing System*** (zkratka **TPS**)

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Základní vlastnosti transakce

důležitý pojem: ACID

- Žádoucí vlastnosti transakcí jsou:
  - ***Atomičnost*** (**A**tomicity) – každá transakce je dokončena zcela nebo vůbec
  - ***Konzistence*** (**C**onsistence) – databázová konzistence (správná reflexe stavu reálného světa a dodržování omezujících pravidel pro hodnoty)
  - ***Izolovanost*** (**I**solation, **I**ndependence) – souběžné provádění má totožný efekt jako sekvenční
  - ***Trvanlivost*** (**D**urability)– odolnost proti ztrátě již dokončených změn
- V databázové praxi se pro tyto vlastnosti užívá akronym ***ACID***.

zdroj: Wikipedie: Database transaction

Kdo co zajišťuje

- Programátor je zodpovědný za vytváření konzistentních transakcí. TPS považuje
  - ***konzistenci*** za zajištěnou programátorem (případně částečně systémem pro kontrolu integritních omezení)
- a zaopatřuje
  - ***atomičnost***,
  - ***izolovanost*** a
  - ***trvanlivost***,
- což jsou vlastnosti nutné pro zajištění ***souběžného spouštění konzistentních transakcí*** a případného ***zotavení z chyb či poruch***.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Úkol transakce

- Úkolem transakce je udržovat model (abstraktní, datový) stavu skutečného světa během jeho změn v ***konzistentním stavu***.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

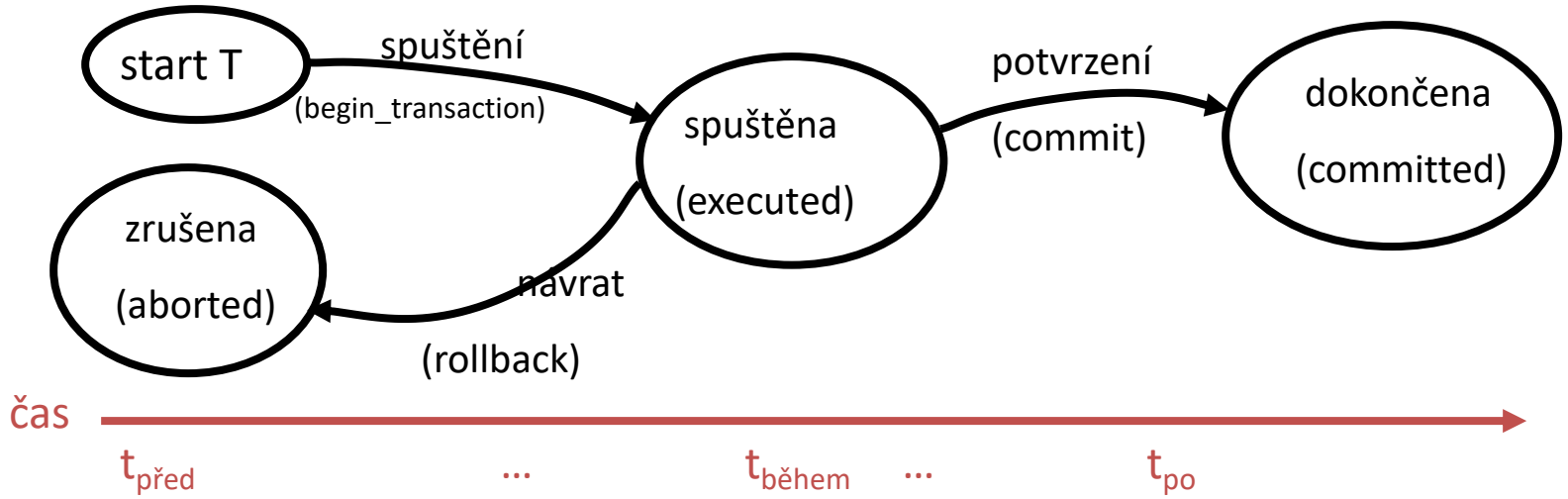
Atomičnost

BUĎ

- Provedení kompletní transakce

NEBO

- Neprovedení ani jedné operace z transakce
  - Implementačně řešeno návratem do původního stavu před spuštěním transakce



zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Důvody zrušení prováděné transakce

Nepředvidatelné:

- havárie systému

V režii TPS:

- porušení integritního omezení
- porušení izolovanosti souběžných transakcí
- detekované uváznutí (deadlock)

V režii transakce/programátora:

- na požadavek samotné transakce (rollback)

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Bankomat

- Transakce výběru hotovosti z bankomatu se skládá z nejméně dvou základních akcí:
  - snížení stavu účtu o vybranou částku a
  - vydání příslušného obnosu hotovosti.
- Atomické spuštění této transakce znamená, že pokud je potvrzena, tak jsou provedeny obě akce a pokud je zrušena, tak ani jedna.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Konzistence

- zajišťuje uživatel
- Databáze má dvě role vzhledem k modelovanému nosiči:

1. pasivní = kontrola a hlášení chyb
2. aktivní = vynucování platnosti pravidel daných aplikační doménou

- Konzistence souvisí s oběma rolemi a má dvě formy:

1. Konzistence datového modelu
2. Konzistence s reálným světem (izomorfismus modelu)

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Konzistence datového modelu

- DB musí splňovat všechna integrit. omezení (IO)
  1. Udržování ***interní konzistence*** (redundantních dat)
  2. Dodržování ***pravidel*** daných reálným světem

- ***Transakce nesmí po svém dokončení porušit/porušovat žádné integritní omezení!***
- U nekonzistentní DB nedefinujeme chování transakcí.

PS: Díky atomičnosti transakcí nevadí dočasná nekonzistence během provádění.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Izolovanost

- zabývá se vícenásobným souběžným přístupem
- řeší ji TPS

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Sekvenční zpracování, plán

- ***sekvenční zpracování transakcí*** = v jeden okamžik je rozpracována nejvýše 1 transakce (zákaz souběžnosti více transakcí)
  - \+ zachování konzistence
  - – špatná efektivita/propustnost
- ***souběžné zpracování*** *(concurrent execution)*= využití paralelismu (několik CPU, několik I/O jednotek)
- ***plán transakce*** = pořadí operací uvnitř této transakce
- (vykonávací) ***plán*** = ***sloučení*** (NE pouze zřetězení) plánů souběžných transakcí (operace se mohou promíchat)

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Izolovanost, nezávislost

- ***Izolovanost*** = výsledný efekt vykonání plánu souběžných transakcí je stejný jako jejich sekvenční zpracování.
- ***Uspořadatelný plán*** = splňuje izolovanost a transakce jsou konzistentní

- Analogie s problémem kritické sekce (v OS) při přístupu ke sdíleným prostředkům
  - zamykání záznamů v DB (trpí výkonnost)
- V praxi – různé **úrovně izolovanosti**
  - volba optima mezi správnou funkčností a rychlostí

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Trvanlivost

- zajišťuje TPS
- ***Trvanlivost*** (stálost) změn zanesených úspěšně dokončenými transakcemi
- ***Dostupnost*** = rychlost uvedení systému do původního funkčního stavu po havárii
  - Nonstop dostupnost (např. zrcadlení disků)
  - Pomalejší dostupnost (např. obnova DB z pásky)
- Úrovně kvality trvanlivosti
  - např. odolnost vůči selhání CPU, selhání 1/více disků, živelné pohromě, úmyslnému útoku
- **kvalita × cena**

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

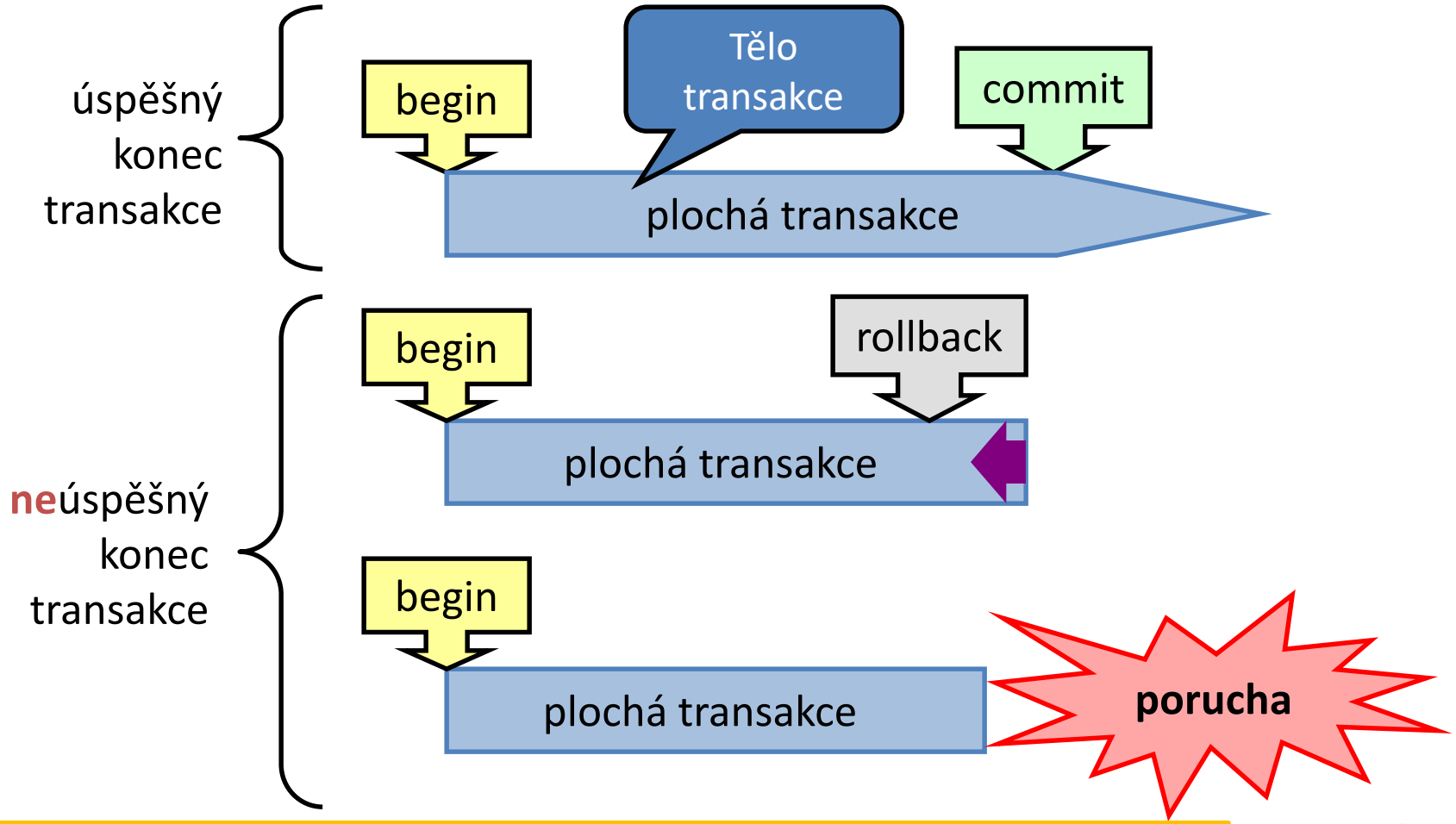
MODELY PROCESŮ (TRANSAKCÍ)

Plochá transakce

- Minimální model, který není vnitřně strukturovaný
- Obvykle obecný proces = popis v programovacím jazyce
- začátek transakce (begin)
- tělo transakce (sekvence operací)
- konec transakce
  - Úspěšný (commit)
  - Neúspěšný (rollback či abort nebo porucha)
- Žádné ***částečné zotavení*** ani ***savepoints***

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Plochá transakce (Flat transaction)



zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Struktura ploché transakce

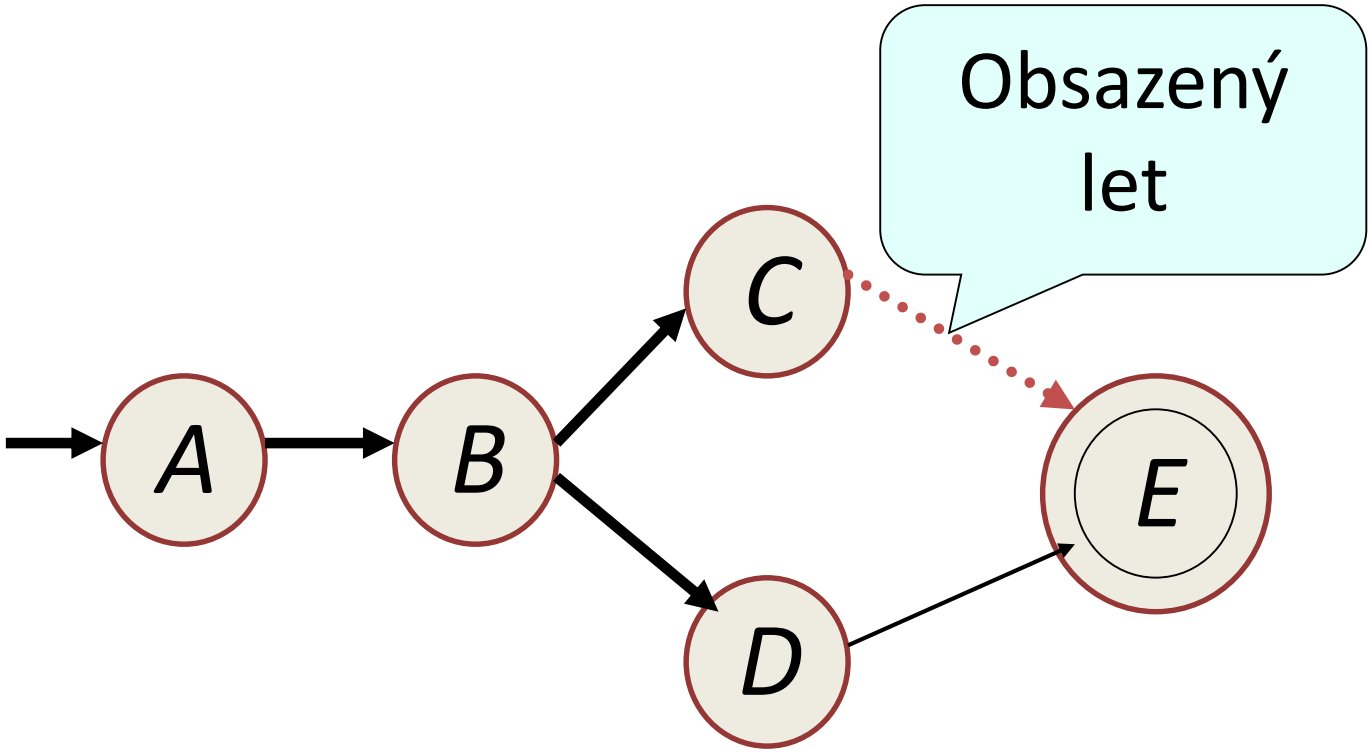
begin_transaction();

... blok programu ...

commit();

- kde blok může obsahovat příkaz abort();
- v programu se mohou vyskytovat ***lokální nedatabázové proměnné***

Příklad ploché transakce hledání volného letu z A do E



zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Příklad ploché transakce

- transakce plánování cest, která provádí rezervace letů na cestě z bodu *A* do bodu *E*.
- prvním pokusem je uspořádat cestu přes body *A*, *B*, *C*, *E*.
- druhou možností je cesta přes body *A*, *B*, *D*, *E*.
- v prvním kroku rezervuje transakce let z *A* do *B*, následně z *B* do *C*.
- nyní však plánovač zjistí, že z bodu *C* do *E* již není žádný volný let, takže ploché transakci nezbude nic jiného, než ***provést zrušení všech prozatím provedených rezervací*** místo toho, aby provedla pouze částečný návrat o jednu rezervaci a provedla rezervaci z bodu *B* do bodu *D*, odkud je již volný let do cílového bodu *E*.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Dekompozice konečným automatem

SEKVENČNÍ MODELY

Body návratu (Savepoints)

- ***savepoint*** = bod návratu (synonymum = ***milník***)v transakci pro částečný návrat (partial rollback)
- i více bodů návratu v 1 transakci
  - odlišení identifikátorem/číslem
  - sp_i := create_savepoint();
- ***částečný návrat*** obnoví DB kontext
  - rollback(sp_i);
  - ***nemění se hodnoty lokálních proměnných***
  - ***poté se pokračuje*** za příkazem návratu ve vykonávání transakce
- není aktivních více transakcí, vše v rámci jediné transakce

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Příklad

```
begin_transaction();
   S1;
   sp1 := create_savepoint();
   S2;
   sp2 := create_savepoint();
   S3;
   sp3 := create_savepoint();
    . . .
   if (condition)
   {
     rollback(sp2);
     . . .
   }
   . . .
commit();
```

- po provedení rollback(sp2); již nemá smysl pracovat s proměnnou sp3, protože identifikuje již neexistující milník. Součástí částečného návratu bylo i odstranění milníků mezi místem volání návratu (rollback(sp2);) a cílem návratu (sp2 := create_savepoint();).

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Body návratu

- ***Lokální proměnné*** v transakci ***nejsou*** provedením částečného návratu (ani návratu) ***změněny***.
- *Provedením částečného návratu tedy není iluze, jako by se daná část transakce vůbec neprovedla, ale pouze jako by neprovedla žádné změny v databázi.*
- Důležitým rozdílem mezi návratem (rollback) a zrušením transakce (abort) je, že ***abort*** transakce dále nepokračuje v provádění transakce, kdežto po ***rollback*** (v případě umístění milníku na úplný začátek transakce lze mluvit i o návratu) se provede zbytek transakce.

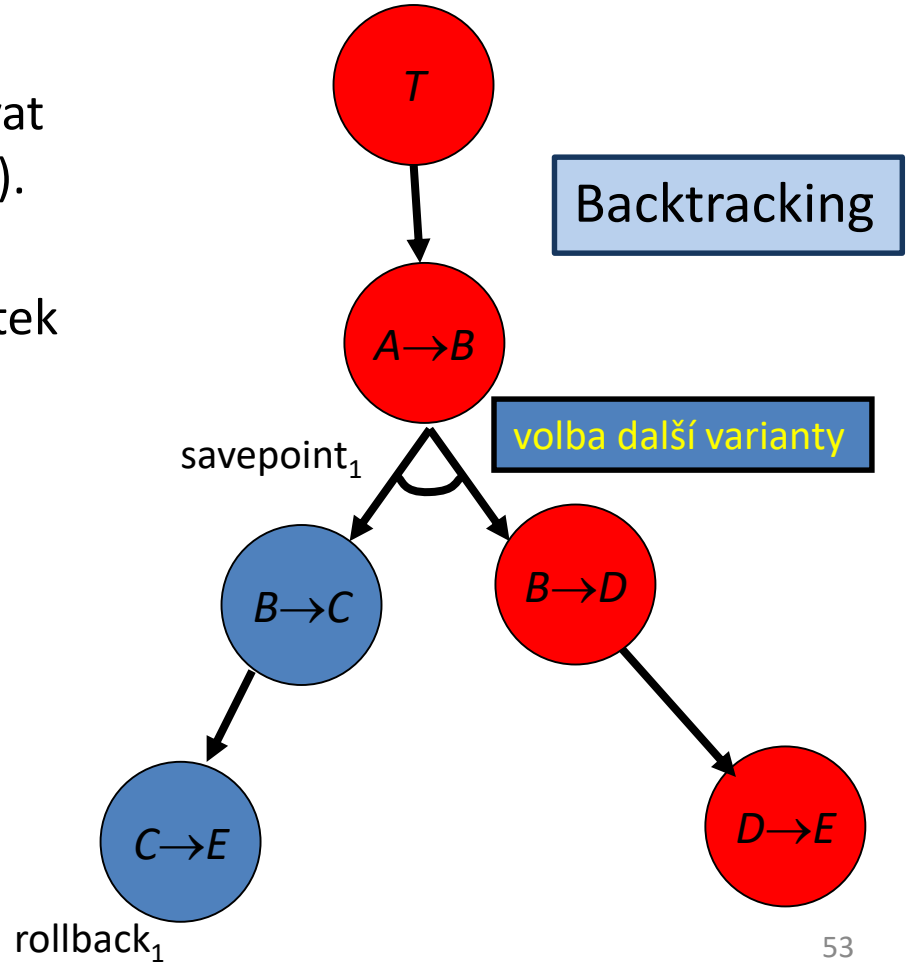
zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Příklad bodů návratu

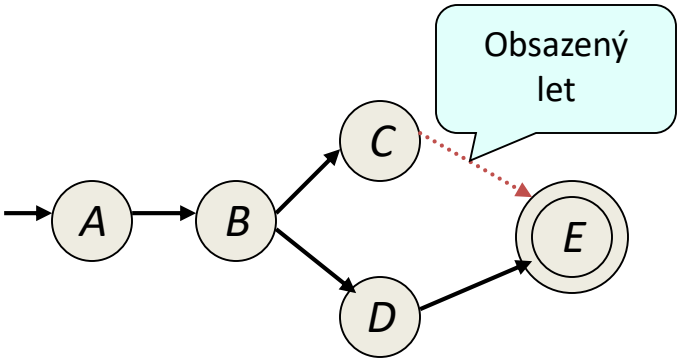
Model transakce *T* pro rezervaci cesty z *A* do *E*:

Je rozpracována jediná transakce se zásobníkovou strategií (musí existovat programově ***zásobník*** bodů návratu).

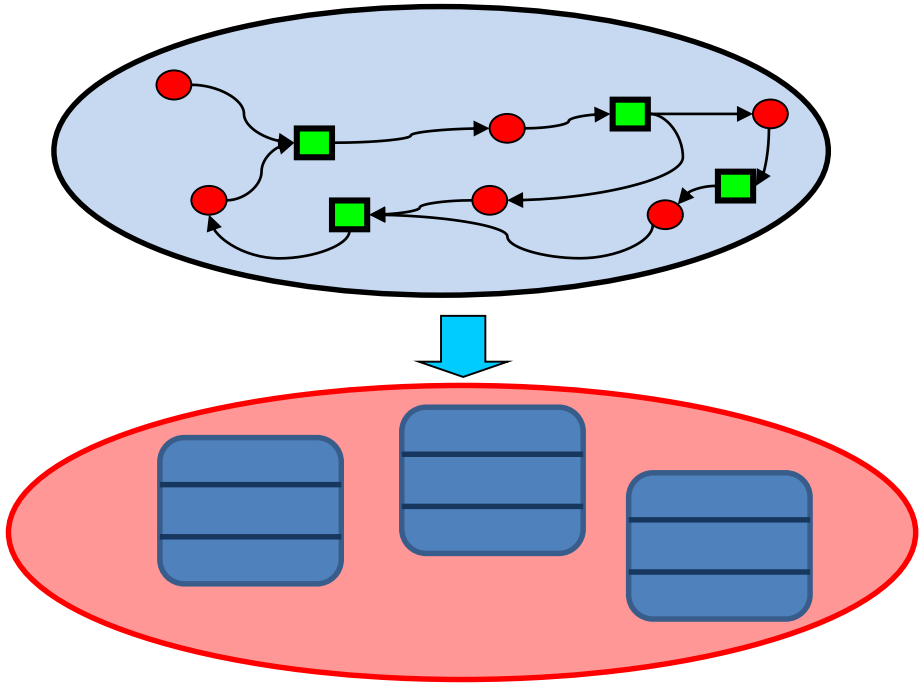
Na začátku a v každém větvení se vytvoří bod návratu. Návrat na začátek znamená neúspěch.



Možné cesty z *A* do *E*:



Dvouúrovňové schéma



řízení - sekvence, hierarchie

dekompozice stavů

stavy - obecný proces
někdy ***transakce***

- Standardně jsou procesy dekomponovány na ***řízení*** a ***činnost stavů*** - ***transakce***

Dvouúrovňové schéma

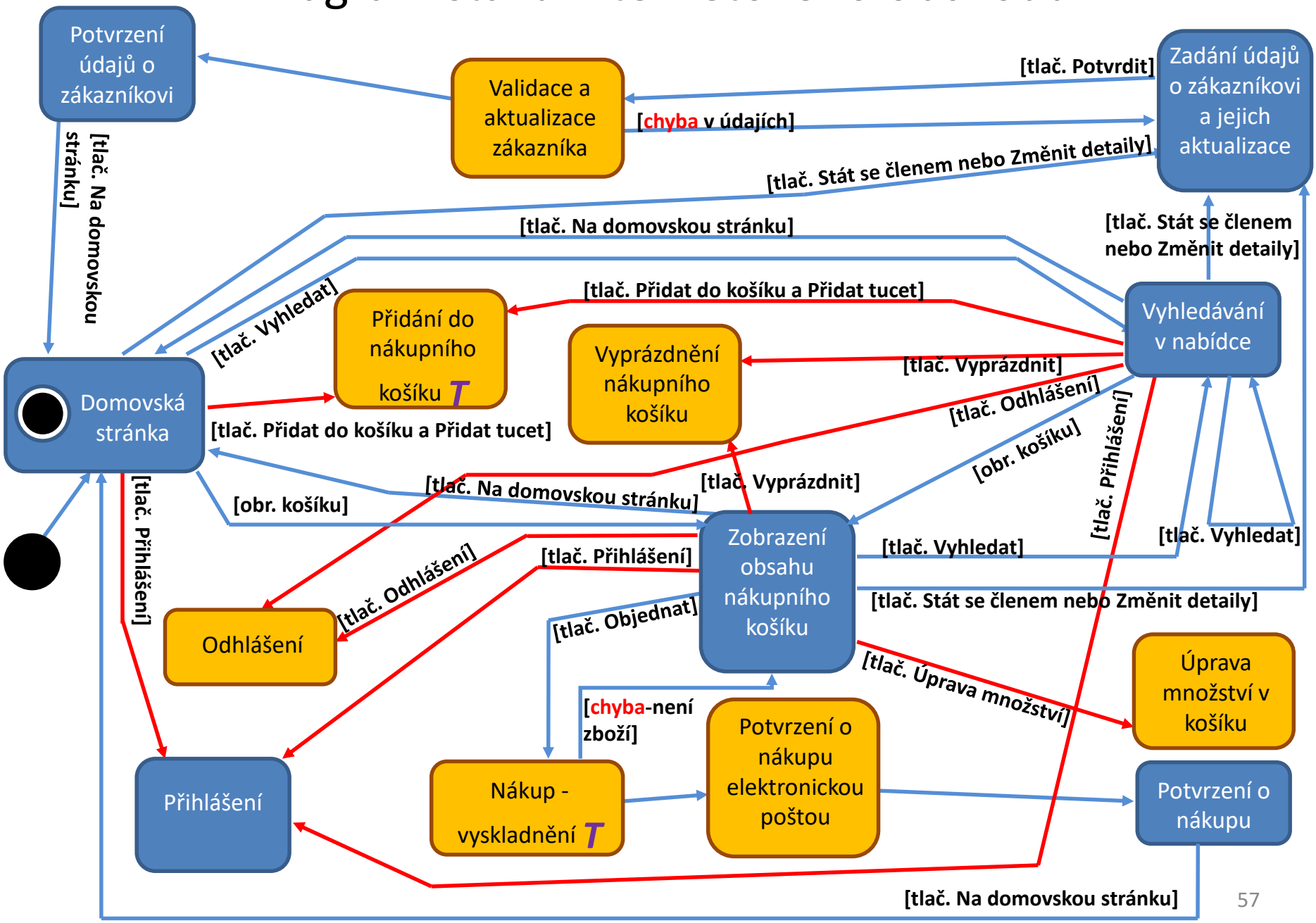
- Obecné procesy lokalizujeme do jednotlivých stavů

- Potom se proces přechodů mezi stavy značně zjednoduší

Dvouúrovňové schéma

- pro modelování informačních systémů na nejvyšší úrovni ***vystačíme se sekvenčními a hierarchickými procesy*** (širší kontext kromě zanoření se většinou nemodeluje)
- obecné procesy ve stavech (pracující s databází) pak modelujeme v obecném programovacím jazyce, někdy jako ***transakce*** podle různých modelů.

Diagram stavů internetového obchodu



Poznámky k diagramu stavů

- Větvení
  - převažující v komunikačních stavech (vizualizováno tlačítky) - událost způsobená uživatelem na klientovi
  - v dávkových stavech porušením konzistence (chyby) - kontrola konzistence prováděná na serveru
- Zabránit vzniku cyklů z hran bez označení
- ***Transakce*** ***T*** v dávkových stavech (ne ve všech)

Chained transaction

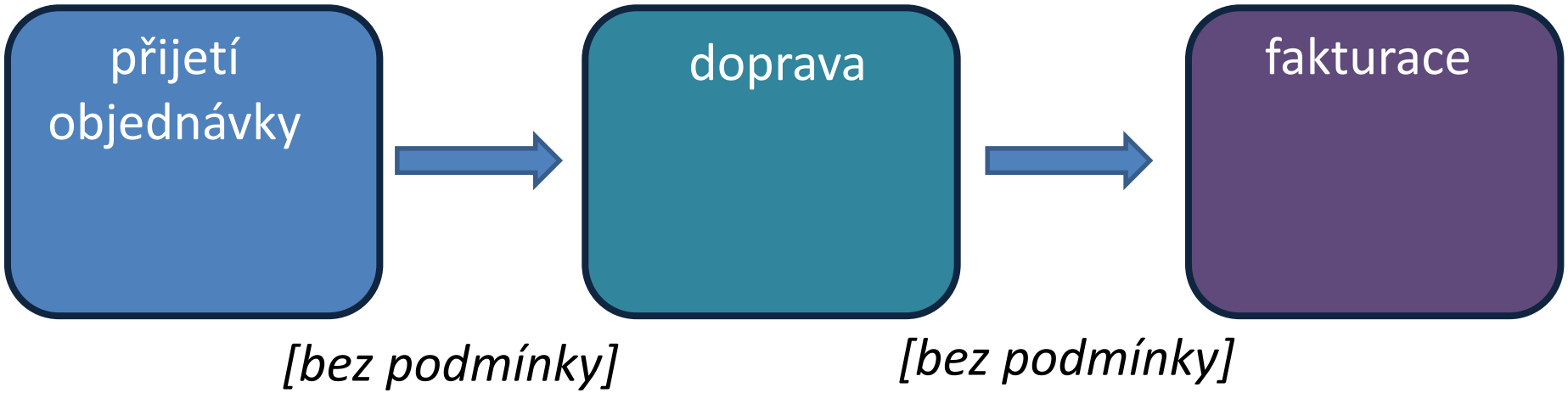
ZŘETĚZENÉ TRANSAKCE – DVOUÚROVŇOVÁ DEKOMPOZICE

Motivace

- Většina aplikací se skládá ze sekvence transakcí.
- Například objednávkový katalogový systém, který se skládá ze tří transakcí:
  - přijetí objednávky,
  - doprava,
  - fakturace.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Slučování řízení a transakcí



- Převedení sekvence stavů s přechody bez podmínek na jedinou modelovanou strukturu
- Dochází ke slučování vrstev dvouúrovňového modelu

Zřetězené transakce - úvod

- v některých situacích je nutno ***dekomponovat*** na menší i transakce, které to nevyžadují ani kvůli distribuovanosti nebo lepší strukturovanosti, ale ***z důvodu vylepšení výkonnosti***.
- především ***dlouho trvající transakce*** (ať už z důvodu velkého množství prováděných operací nebo nutnosti delších čekání mezi jednotlivými operacemi).
- dalším důvodem dekompozice je v případě poruchy ***zabránění ztrátě celé zatím provedené práce***, kterou transakce vykonala

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Zřetězené transakce

- Jednoduchou optimalizací je tzv. ***zřetězení* (angl. *chaining*)**, kdy po potvrzení jedné transakce je automaticky vytvořena nová transakce, která v dané sekvenci transakcí následuje.
- Tímto např. omezíme režii nutnou na zasílání příkazu begin_transaction() databázovému systému

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Schéma zřetězené transakce

begin_transaction();
S_1;
commit();
S_2;
commit();
. . .
S_{n-1};
commit();
S_n;
commit();

- kde S_i je tělo *i*-té transakce, kterou budeme dále označovat jako podtransakci ST_i.

Chování zřetězené transakce

- každý příkaz commit() způsobí ***trvanlivost změn předchozí podtransakce***, takže v případě havárie v podtransakci ST_i budou změny z podtransakcí ST_1, …, ST_{i-1} zachovány.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Odlišné vlastnosti zřetězených transakcí

- Při využití pro dekompozici dlouhých transakcí (například připisování úroků na desítky tisíc bankovních účtů) je potřeba uvažovat následující soubor vlastností, kterými se zřetězené transakce ***odlišují*** od prozatím celkem striktních požadavků na ***ACID*** vlastnosti transakcí

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Trvanlivost

- při poruše nejsou ztraceny výsledky předchozích podtransakcí v řetězu (sekvenci) na rozdíl třeba od klasické ***sekvenční dekompozice transakcí s milníky*** (savepoints), kdy je celá transakce zrušena (proveden kompletní návrat).
- Tím pádem zřetězená transakce jako celek (celý řetěz podtransakcí) ***není atomická***. TPS nemá žádnou zodpovědnost za restartování řetězu při zotavování z poruchy.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Komunikace mezi transakcemi v řetězu

- Problém využívání ***lokálních proměnných*** pro komunikaci mezi podtransakcemi (například předávání identifikátoru posledního zpracovaného účtu při procházení seznamu účtů) ***nastane v případě poruchy***, kdy všechny obsahy lokálních proměnných zaniknou a nemohou být tak využity při dokončení zřetězené transakce.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Komunikace mezi transakcemi v řetězu

- Alternativou je komunikace přes ***databázové proměnné***. Před potvrzením podtransakce jsou lokální proměnné uloženy do databázových položek a stanou se tak součástí databázového kontextu, který dokáže přežít pád systému. Při dokončování havarované transakce pak lze například zjistit, kterým záznamem dále pokračovat.
- Nevýhodou komunikace přes databázové proměnné je jejich ***viditelnost ostatním souběžným transakcím*** (např. v jiných aplikacích). Toto musí být ošetřeno (např. využitím ***zámků*** nad záznamy databázových proměnných).

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Databázový kontext

- není udržován mezi dvěma sousedními podtransakcemi v řetězu. Například pokud je potvrzena podtransakce ST_i, tak se uvolní všechny jí aktivované zámky nebo používané databázové kurzory.
- souběžné transakce tak mají v časovém úseku mezi dokončením ST_i a startem ST_{i+1} volný přístup i k položkám, které byly v ST_i zamknuty.
- může se tak stát, že ST_{i+1} již bude pracovat s jinými hodnotami v těchto položkách, protože budou změněny nějakou souběžnou transakcí.
- v kontrastu s dlouho trvajícími transakcemi je sice ***každá podtransakce řetězu izolovaná***, ale ***celá zřetězená transakce izolovaná není***.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Databázový kontext

- Porušení izolovanosti zřetězené transakce jako celku při správném návrhu a použití tohoto typu transakcí vede ke zvýšení výkonnosti, protože je zkrácena doba zamykání záznamů v databázi a je tak větší prostor pro souběžné provádění transakcí

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Konzistence

- podtransakce při potvrzení uvolňují databázový kontext a zviditelňují tak provedené změny ostatním souběžným transakcím.
- v případě, že tyto souběžné transakce vyžadují spouštění v konzistentní databázi, tak musíme požadovat, aby každá podtransakce v řetězu byla konzistentní. Všimněme si, že v případě použití ***milníků*** tento problém neřešíme, protože je celá transakce ***atomická*** a ***izolovaná***.
- Konzistence podtransakcí je vyžadována také z důvodu, že při havárii systému ***nezajišťuje TPS***, aby byla přerušená zřetězená transakce ***dokončena*** (je proveden návrat pouze aktuálně spuštěné podtransakce).
- ***Zřetězené transakce nejsou izolované ani atomické***.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Alternativní sémantika pro zřetězené transakce

- pokouší se eliminovat některé předchozí nevýhody a porušení ACID

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Schéma alternativní sémantiky

begin_transaction();
  S_1;
  chain();
  S_2;
  chain();
  . . .
  S_{n-1};
  chain();
  S_n;
commit();

- kde chain() odlišuje starou a novou sémantiku a jedná se o příkaz ***potvrzení podtransakce ve zřetězené transakci*** s lehce pozměněnou sémantikou oproti příkazu commit().

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

chain()

- Příkaz chain() potvrzuje podtransakci ST_i, začíná novou ST_{i+1}, ale
- na rozdíl od commit() ***mezitím neuvolňuje databázový kontext*** (např. aktivované zámky nad položkami databáze) ani databázové kurzory.
- Pokud byla tedy nějaká položka zamknuta v ST_i, tak bude zamknuta i v ST_{i+1}, pokud nebyl zámek explicitně uvolněn.
- Souběžné transakce navíc nevidí stav databáze ani v časovém bodě mezi ST_i a ST_{i+1}. Pokud to ST_{i+1} očekává, tak dokonce může ST_i zanechat databázi nekonzistentní. ***Zřetězena transakce s alternativní sémantikou je jako celek izolovaná***, i když na úkor výkonnosti.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Dopředný návrat (roll forward)

- daní za toto řešení je ***komplikovanější zotavovací fáze***.
- nyní již striktně nevyžadujeme konzistenci po dokončení podtransakce uvnitř řetězu, tak může při návratu ke stavu po poslední dokončené podtransakci ***zůstat databáze v nekonzistentním stavu***.
- ***Úplný návrat nelze provést***, protože změny již dokončených podtransakcí byly potvrzeny a porušili bychom tak trvanlivost.
- Řešením je rozšíření zotavovací procedury, která po restartu TPS zajistí ***dokončení zřetězené transakce***. Tj. obnoví se databázový kontext (včetně zámků) a znovu se provede kvůli havárii nedokončená podtransakce a všechny po ní následující až do konce řetězu. Pak zůstane ***zachována izolovanost i atomičnost celé zřetězené transakce s alternativní sémantikou***. Tento způsob zotavení se nazývá ***dopředný návrat*** (angl. ***roll forward***).

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Kompenzující transakce

- mějme zřetězenou transakci, kterou jsme vytvořili místo jedné dlouho trvající transakce, a předpokládejme, že po potvrzení (dokončení) několika podtransakcí potřebuje transakce provést úplný návrat (abort).
- bohužel protože zřetězené transakce ***nezaručují izolovanost ani atomičnost***, tak nelze garantovat, že nebyla databáze před spuštěním návratu změněna nějakou souběžnou transakcí, což je zásadní problém.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Možnost implementace návratu

- Možnosti implementace ***částečného návratu a úplného návratu transakcí***:
- v plochých a zanořených transakcích, které zaručují ACID vlastnosti, lze použít tzv. ***fyzickou obnovu*** neboli ***fyzické logování*** (angl. *physical restoration, logging*), kdy si uložíme stav dané databázové položky před její změnou, poznačíme si, že byla změněna, a při návratu provedeme obnovení její původní hodnoty.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Nevýhody fyzického logování

- Odčinění změn způsobených zřetězenou transakcí (neuvažuji alternativní sémantiku) je však složitější, protože pouhé fyzické logování je nedostatečné, protože mezi jednotlivými podtransakcemi je databázový kontext viditelný i ostatním souběžným transakcím a ty mohou provést nějakou změnu předtím, než spustíme návrat.
- Provedením fyzického obnovení bychom tak mohli potenciálně porušit trvanlivost změn od jiných transakcí. Tento problém řeší tzv. ***kompenzace*** (angl. *compensation*).

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Kompenzující transakce

- ***Kompenzující transakce*** je speciální druh transakce, který zajišťuje provedení ***logické obnovy důsledků spuštění jiné transakce***.
- Obsahuje posloupnost akcí nutných pro uvedení databáze do stavu, v jakém by byla při neprovedení transakce, kterou kompenzační transakce kompenzuje. Kompenzační transakce ***nemusí revertovat nutně všechny změny***, což je dáno především aplikační doménou a důležitostí těchto změn pro konzistenci databáze i konzistenci s reálným světem.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Příklad kompenzující transakce

- ***Registraci studentů do předmětu.***
- Odregistrační (kompenzační) transakce logicky ***ruší efekt úspěšné registrační transakce***.
- ***Registrace zvýší*** počet studentů ve třídě a naopak ***deregistrace*** jej o jednoho ***sníží***.
- Kompenzace tak správně provede návrat i v případě, že byla souběžně provedena registrace dalšího studenta.
- Příkladem operace z registrační transakce, která nemusí být nutně kompenzována může být registrace i neúspěšně zaregistrovaného studenta do bulletinu, jehož účel je informovat o nových předmětech na daném ústavu čistě z reklamních důvodů

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

ZOTAVITELNÉ FRONTY

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Zotavitelné fronty - motivace

- aplikace nevyžaduje úzké semknutí sekvence akcí do jediné transakce (izolované jednotky)
- postačí, když je zaručeno provedení jisté sekvence akcí (často transakcí)
- požadavkem je, aby po dokončení jedné akce byla ***někdy provedena další***.
- na rozdíl však od ***zřetězených transakcí*** může být mezi jednotlivými akcemi podstatná ***časová proluka***.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Příklad aplikace pro zotavitelnou frontu

- ***Zadání a specifikace problému katalogového a objednávkového systému.***
- Hlavní aktivita v tomto systému se skládá z vytvoření
  - objednávky,
  - expedice a
  - fakturace.
- Tyto tři úkoly lze vykonat třemi oddělenými transakcemi. Jediné co požadujeme, aby transakce ***fakturace a expedice byla provedena kdykoli po úspěšném dokončení objednávky***, a to i v případě havárie systému bezprostředně po objednání.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Zotavitelná fronta - definice

- ***Zotavitelná fronta*** (angl. ***recoverable queue***) je mechanismus na plánování transakcí pro budoucí vykonání a zajištění, že vykonání bylo skutečně v aplikaci provedeno. Základní sémantika vychází z klasické fronty, která obsahuje operace:
- ***Vlož.*** Transakce vkládá do fronty záznam o práci naplánované k provedení, právě když je transakce potvrzena.
- ***Vyber*****.** Záznam je pak někdy později vyzvednut jinou transakcí, která danou práci provede. Tato transakce bývá většinou spuštěna serverem, který periodicky kontroluje frontu a vybírá z ní pracovní požadavky.
- ***Vkládaný/vybíraný záznam*** obsahuje informaci o akci, která je plánována, a o datech, která je potřeba mezi jednotlivými transakcemi v řetězu předávat (například identifikátor objednávky).

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Vlastnosti zotavitelné fronty

- Zotavitelná fronta ***musí být trvanlivá***, aby byla schopna přežít havárii systému. Navíc atomicita transakce vyžaduje od vložení/výběru z fronty následující koordinaci s potvrzením (resp. způsobem ukončení) transakce:
  - Vloží-li transakce do fronty záznam a později je zrušena, tak musí být tento záznam z fronty odstraněn.
  - Vybere-li transakce z fronty záznam a později je zrušena, tak musí být tento záznam do fronty navrácen.
  - Dokud není transakce *T* dokončena (potvrzena), tak nelze jinými transakcemi vybírat záznamy, jež byly transakcí *T* vloženy, protože může být transakce *T* ještě případně zrušena.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Trvanlivost zotavitelné fronty

- Trvanlivost zotavitelné fronty lze sice implementovat prostřednictvím tabulky v databázi, ale častý přístup k této tabulce pak tvoří ***výkonnostní slabinu*** celého systému (angl. ***bottleneck***). Proto je vhodné využít k realizaci ***oddělený aplikační modul***.
- Jde tedy o další funkční blok systému

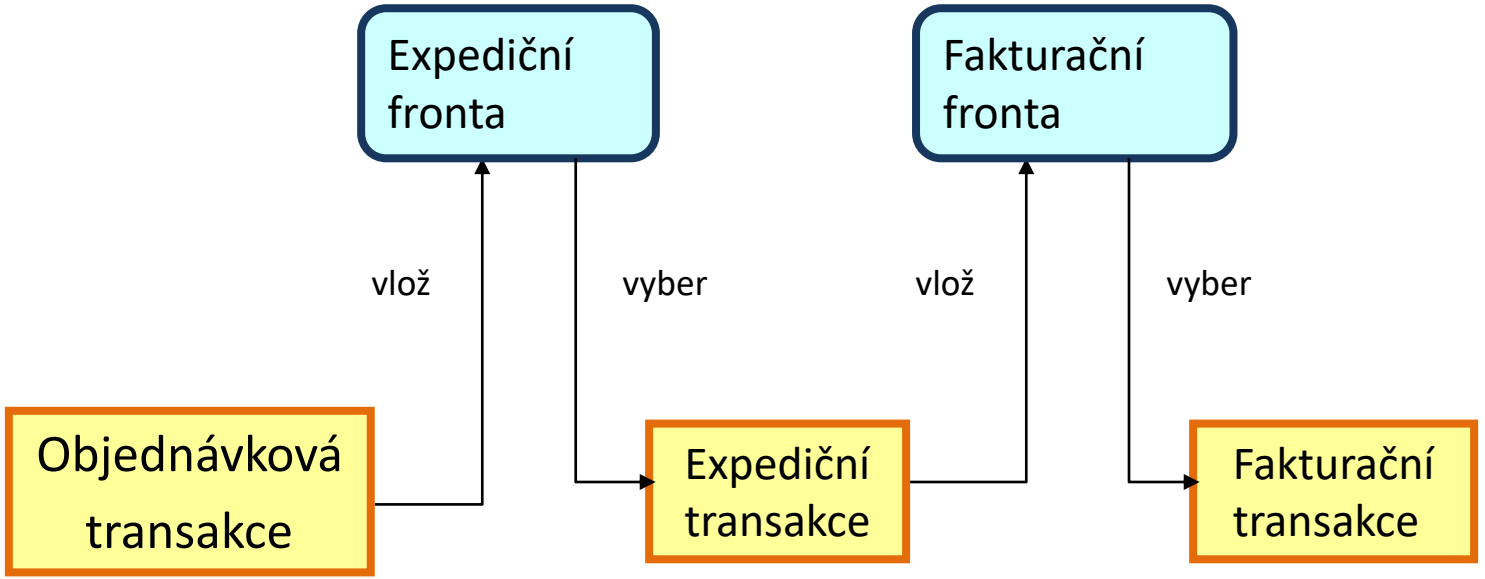
zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Scénáře využití zotavitelných front

- Sekvence a paralelismus směřují k pozdějším schématům ve workflow
- Doposud se paralelismus v modelech ***neřešil***. Předpokládalo se ***řešení pomocí TPS.***

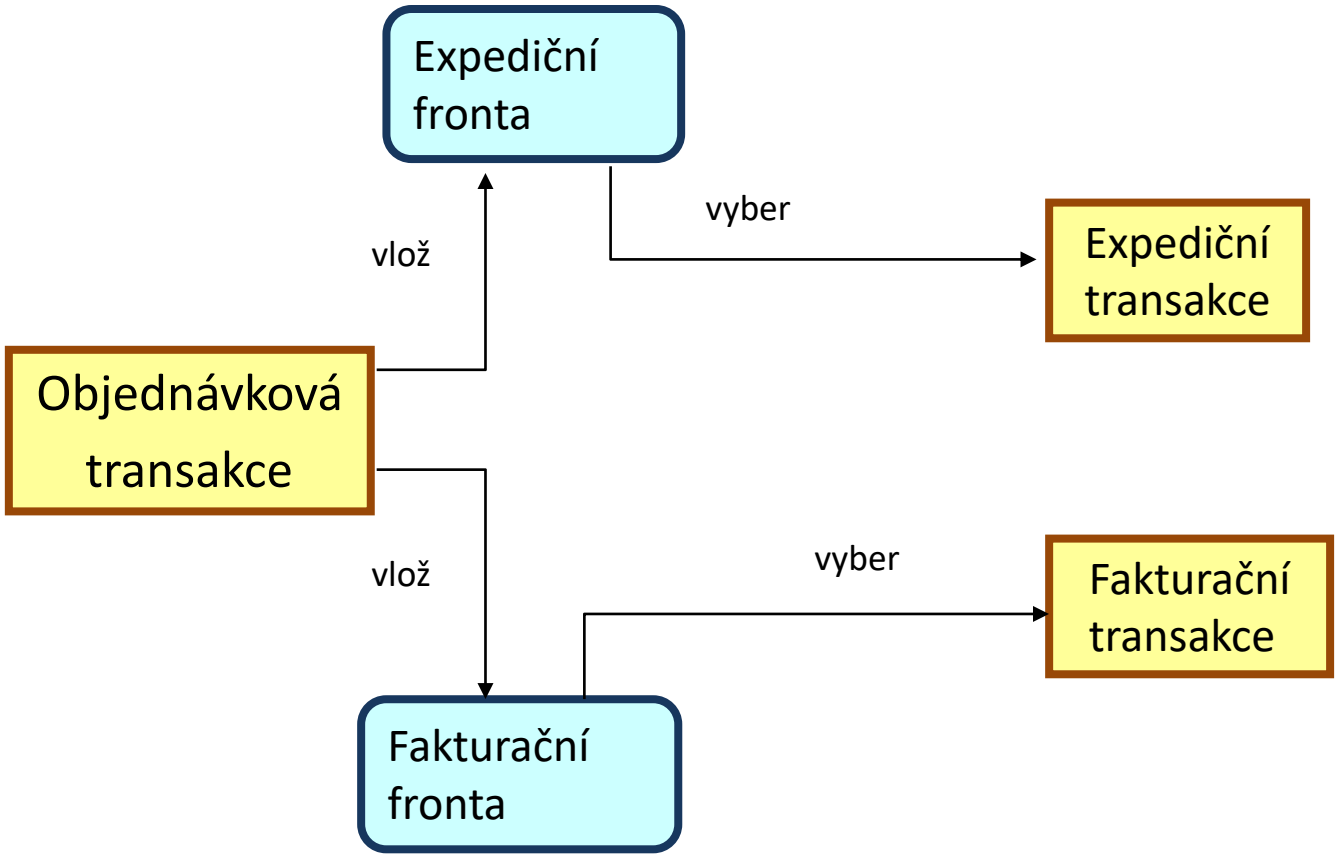
zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Organizace zotavitelné fronty – řetěz s pevným uspořádáním (pipeline)



zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Organizace zotavitelné fronty – podpora paralelismu



zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Paralelismus

- využití ***paralelismu***, protože povoluje, aby byla transakce fakturační i expediční spuštěna v libovolný okamžik po dokončení objednávky (nevýhoda – zákazník může dostat dříve fakturu než samotné zboží, ale obchodně je to v pořádku).

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

REÁLNÉ UDÁLOSTI

Reálné události

- transakce s reálnou událostí může být zrušena.
- problém je v dosažení atomičnosti, protože reálné události nelze většinou vrátit zpět (odčinit).
- Například pokud bankomat vydá zákazníkovi hotovost a před potvrzením transakce systém havaruje, tak TPS nedokáže zajistit, aby byly navráceny všechny provedené akce. Může sice vrátit databázi do původního stavu, ale od zákazníka ***lze jen těžko požadovat navrácení hotovosti*** (alespoň ne se 100% jistotou ☺).

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Reálné události

- ***Reálné (fyzické) události*** (angl. ***real-world events***) nelze vrátit zpět, a proto musí být atomičnost transakcí s takovými událostmi dosažena jiným způsobem než návratem (*rollback*).
- Nejpřirozenější je použít ***dopředný návrat*** (*forward roll*) s využitím zotavitelných front pro dosažení sémantiky – fyzická událost je provedena, když a jen když je transakce potvrzena.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Reálné události

- Transakce *T* vloží požadavek *R* na provedení fyzické události na zotavitelnou frontu *Q* před svým potvrzením. Pokud je však *T* zrušena, tak je *R* odstraněno z *Q*. Je-li T úspěšně potvrzena, tak je díky trvanlivosti *Q* někdy v budoucnu zajištěno obsloužení *R*.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Havárie reálné události

- Problém nastává v případě havárie transakce pro fyzickou událost T_{RW}, kdy nevíme, zda byla fyzická akce již provedena nebo ještě ne.

Možné řešení

- Předpokládejme, že fyzické zařízení provádějící reálnou událost obsahuje ***čítač*** *C*, který se inkrementuje atomicky s provedením akce. Dále předpokládejme, že tento čítač může číst i transakce T_{RW}. Pak po provedení akce provede T_{RW} načtení a uložení aktuální hodnoty čítače do databáze (***záznam*** označme *D*) před vlastním potvrzením T_{RW}. Když se pak musí systémem zotavit z náhlé havárie, tak načte čítač *C* a porovná s hodnotou v *D*. Porovnání může mít dva výsledky:

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Možné řešení

- $C = D$. Tedy žádná další fyzická akce nebyla provedena od potvrzení poslední T_{RW}.
- $C \neq D$. C musí být o jedničku větší, což znamená, že fyzická akce byla provedena, ale T_{RW} byla zrušena (a navrácena, včetně případného provedení aktualizace záznamu *D* i obnovení záznamu pr o T_{RW} v zotavitelné frontě).

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce

Zotavení

- Systém tedy z této situace ($C \neq D$) usoudí, že fyzická akce již byla provedena a cíl transakce T_{RW} byl splněn, i když nebyla potvrzena, a proto se před znovu-nastartováním celého systému provede odstranění záznamu pro T_{RW} ze zotavitelné fronty.

zdroj: studijní opora: Pojem informačního systému, Data, Procesy, Transakce