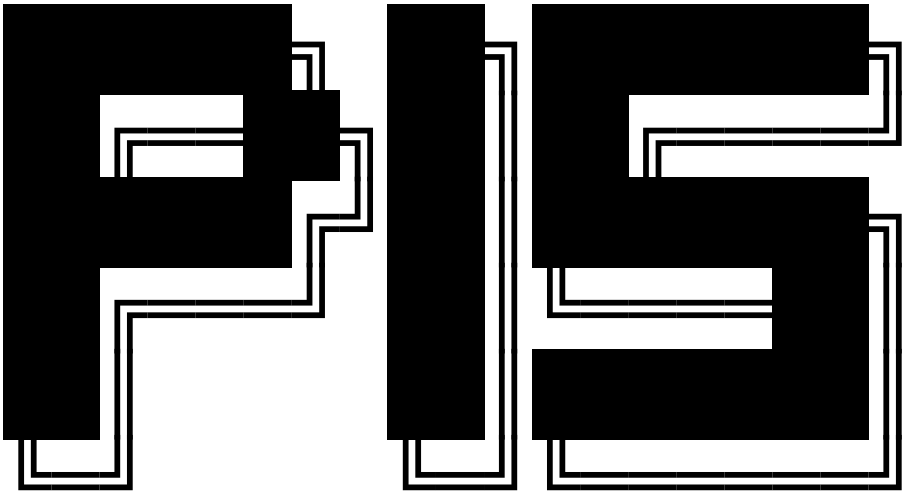


2020

1) OLAP charakteristika, ROLAP, jak se ukládá do relační DB - schémata hvězda a vločka popsat
2) SPARQL, nad jakými daty, co vrací, ukázat SELECT
3) GraphQL, co to je, srovnání s REST API, Co musí být implementované na straně serveru a co v klientovi
4) Zotavitelná fronta, operace, použití
5) Workflow, účel v IS, kde se užívá, typy jazyků a porovnání mezi nimi
6) Typy geografickych grafu / vizualizace geografickych dat, co je GeoJSON a použití
7) Podvědomé vnímání + užití v dashboardech

2019

1. Definujte dimenziu a multidimenzionalnu kocku. Definujte sucet. Nakreslite kocku pre cas, miesto, zbozi, dodavatel.

- Dimenze je uspořádatelná množina hodnot diskrétního základního typu (integer, výčet, čas) nebo množina jejich struktur hierarchicky organizovaných

Definice multidimezionální kostky

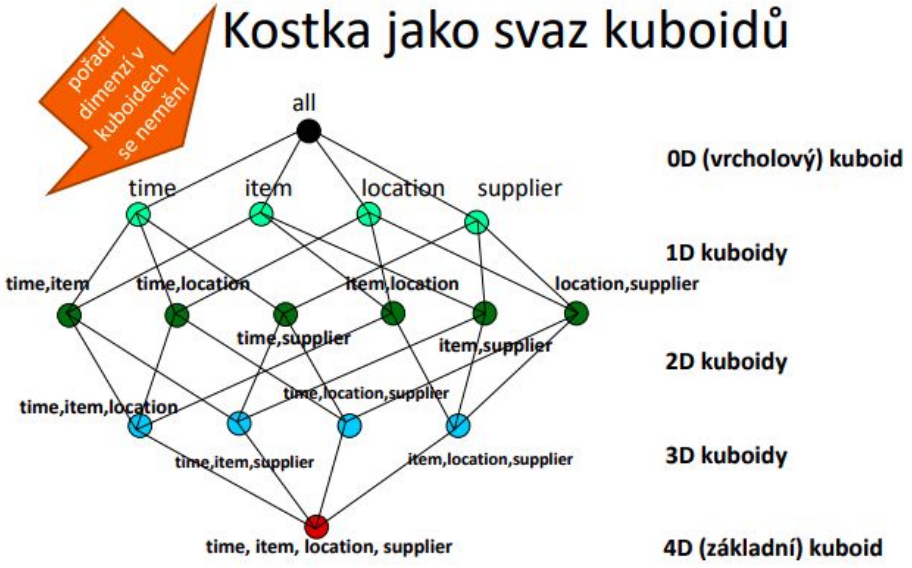
- ***Multidimenzionální kostka*** je funkce $\boldsymbol{g}_m$ (A_1 x A_2 x A_3 x …x A_m) = ***F***, kde $f \in \boldsymbol{F}$ nazýváme ***fakt (míra, measure)*** a A_1 x A_2 x A_3 x …x A_m je ***kartézský součin*** dimenzí.
- ***Fakt (míra, measure)*** je libovolná ***agregovatelná*** hodnota (lze ji sčítat, průměrovat, řetězit apod.), tedy existují kostky počtů, aritmetických průměrů, součtů apod.

- *počet různých prvků* relace *R* je *n!* (počet ***permutací*** nad *n*, počet různých uspořádání)
- je to také ***počet stěn n-dimenzionální kostky***. Proto má ***3D kostka na vrhcáby*** 6, tj. 3! stěn a dvoudimenzionální čtverec 2, tj. 2! stěn.

1	čas	katalog	místo
2	čas	místo	katalog
3	katalog	čas	místo

4	katalog	místo	čas
5	místo	katalog	čas
6	místo	čas	katalog

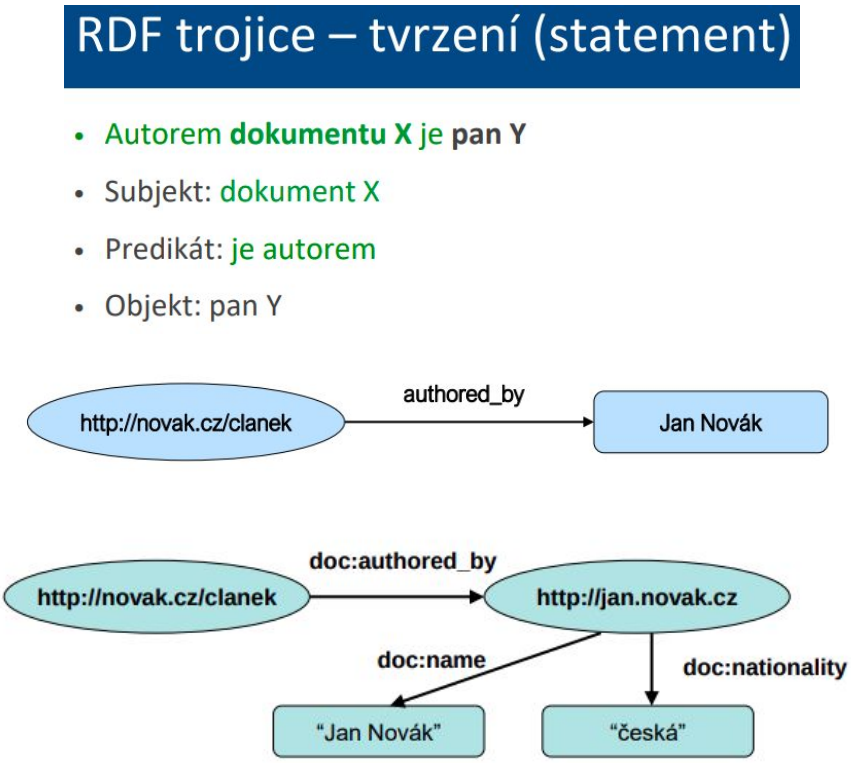
Kostka jako svaz kuboidů



2. Co to RDF, popiste RDF trojice. Popiste serializaciu a napiste dva sposoby serializacie. Nakreslite priklad nejakeho RDF graf s 5-6 uzlami.

- Je datový model standardizovaný W3C
- grafová struktura
- zaměřeno na data sdílená na venek.

Snadné propojení dat z různých zdrojů a na různých schématech.



Serializace - Official RDF/XML serialization, N3 Notation (cleaner), Turtle (Simplified N3)

3. Popiste klientske a vyvolavane aplikacie vo WF. Popiste aplikacne, vecne, riadace data vo WF. Napiste, ktore casti WF pracuju s akymi datami.

1. Klientské aplikace workflow
    a. Provádí jednotlivé úkoly
    b. Interakce uživatelů s workflow
2. Vyvolané aplikace
    a. Spouštěné v souvislosti se započetím úkolu apod.

1. Řídicí data workflow
    a. Interní data WF systému nutná pro zajištění chodu příp. zotavení po havárii
    b. Nedostupná externím aplikacím
2. Věcná data workflow
    a. Zpracování jádrem workflow systému
    b. Používána pro rozhodování o dalším postupu
    c. Dostupná i aplikacím
3. Aplikační data workflow
    a. Specifická data aplikací podporujících proces
    b. Nejsou přístupná WF systému

4. Autentizacia v RESTe. Preco nie je mozne pouzit sessions? Uvedte mechanizmy autentizacie pre REST. Popiste JSON Web Token, co to je, z coho sa sklada a ako prebieha autentizacia pomocou JWT. --P01 str 60-63

Protokol REST je definován jako bezstavový.
Požadavek musí obsahovat vše, žádné ukládání stavu na serveru
JWT Řetězec složený ze 3 částí kodovane base64
1. Header (hlavička) – účel, použité algoritmy (JSON)
2. Payload (obsah) – JSON data obsahující id uživatele, jeho práva, expiraci apod.
3. Signature (podpis) – pro ověření, že token nebyl podvržen nebo změněn cestou
Klient kontaktuje autentizační server a dodá autentizační údaje. Autentizační server vygeneruje podepsaný JWT a vrátí klientovi. Klient předá JWT při každém volání API.
Authorization: Bearer xxxxx.yyyyy.zzzzz

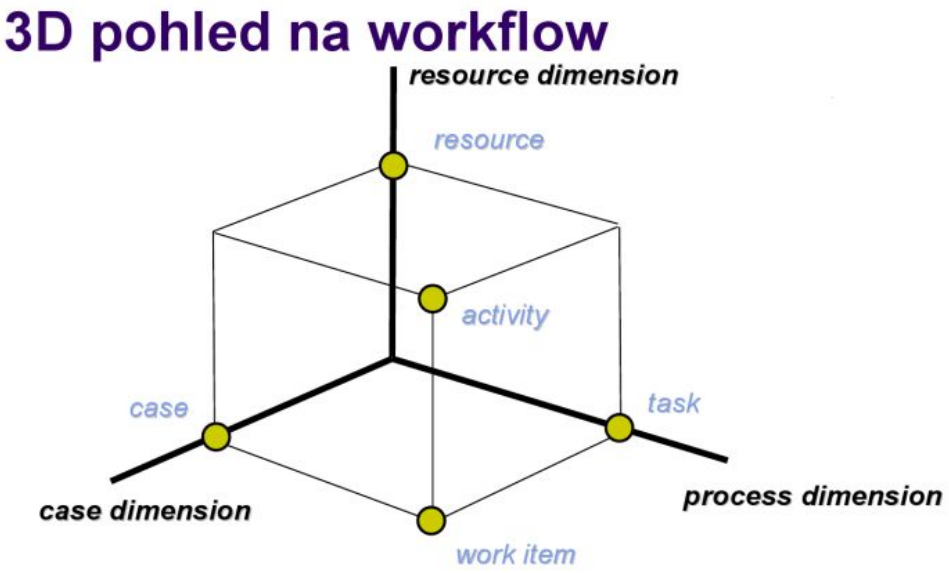
5. Definujte zotavitelnu frontu. Uvedte, aka je jej suvislost s procesom a transakciami. Popiste zakladne operacie nad touto frontou. Uvedte jej vlastnosti. Uvedte jednoduchy priklad jej pouzitia.

Zotavitelná fronta je mechanismus na plánování transakcí pro budoucí vykonání a zajištění, že vykonání bylo skutečně v aplikaci provedeno. Musí být trvanlivá, obsahuje operace:
• *Vlož*. Transakce vkládá do fronty záznam o práci naplánované k provedení, právě když je transakce potvrzena.
• *Vyber. Z*áznam 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).
Použití při paralelismu?

6. Definujte pojmy proces, uloha, pripad, zdroj, pracovna polozka, aktivita. Uvedte ako spolu suvisia.

Proces je komplexnější činnost.
- případ (case)
    - konkrétní řešený problém (žádost o půjčku)
- úloha (task)
    - krok provádění procesu
- zdroj (resource)
    - zařízení (fax, tiskárna) nebo osoba (účastník, dělník, zaměstnanec)
- pracovní položka, požadavek (work item)
    - úkol řešený pro konkrétní případ, např. „vrátit panu Novákovi peníze za reklamované zboží“
- činnost (activity)
    - úkol řešený pro konkrétní případ a využívající konkrétní zdroj
    - vytváří frontu požadavků (worklist)

Jak to spolu souvisí? - Určují nám pohled na workflow a taky jej reprezentují.



7. Mikrosluzby. Popiste architekturu s mikrosluzbami, uvedte jej vyhody a nevyhody. Porovnajte mikrosluzby s monolotickou architekturou. Nakreslite schemu tejto architektury s webovym rozhranim. -- P01 str 87-91

Mikroslužby

- Aplikace je rozdělena na malé části
  - Vlastní databáze (nepřístupná vně)
  - Business logika
  - Aplikační rozhraní (REST)
- Typicky malý tým vývojářů na každou část (2 pizzas rule)
- Výhody
  - Technologická nezávislost
  - Snadné aktualizace, kontinuální vývoj
- Nevýhody
  - Testovatelnost – závislosti na dalších službách
  - Režie komunikace, riziko nekompatibility, řetězové selhání, …

Vlastnosti mikroslužby

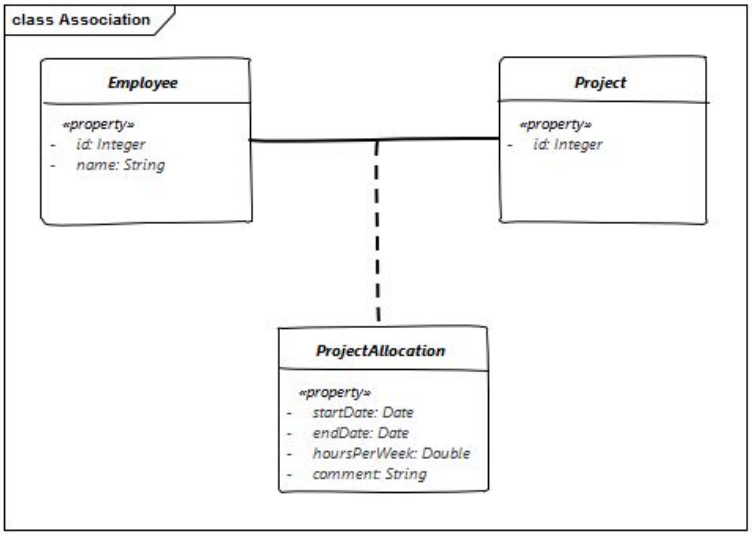
- Vnější API
  - Dostatečně obecné – reprezentuje logiku, ne např. schéma databáze (která je skrytá)
- Externí konfigurace
- Logování
- Vzdálené sledování
  - Telemetrie – metriky (počty volání apod.), výjimky
  - Sledování živosti (Health check)

Monolitická architektura

- Jedna aplikace
  - Jedna databáze, webové (aplikační) rozhraní
  - Business moduly – např. objednávky, doprava, sklad, …
- Výhody
  - Jednotná technologie, sdílený popis dat
  - Testovatelnost
  - Rychlé nasazení – jeden balík
- Nevýhody
  - Rozměry aplikace mohou přerůst únosnou mez
  - Neumožňuje rychlé aktualizace částí, reakce na problémy
  - Pokud použité technologie zastarají, přepsání je téměř nemožné

1, Co su metadata v objektovom db modeli, co obsahuju, ako su reprezentovane vztahy medzi entitami a kardinalita. Uvedte priklad vztahu 1:n
https://blog.zvestov.cz/software%20development/2015/04/15/jpa-vazebni-tabulky-s-metadaty

Existuje ovšem elegantnější řešení, jak se z *Employee* dostat přímo k entitám *Project* a zároveň mít případně k dispozici i metadata v *embeddable* objektu *ProjectAllocation*. (V anglické terminologii *Relationship State* či *Association class*)



2, Popiste dokumentove nosql db. Na com je model db zalozeny, ako su data ulozene, atd. Dalej, ako prebieha dotazovanie. Priklad takejto dokumentovej nosql db.

noSQL:

- NoSQL podporují nerelační datový model.
  (klíč-hodnota, dokumentové, grafové, atd.)
- NoSQL podporují distribuovanou architekturu.
  (lze použít jako centrální db., ale jejich síla je v distribuovanosti)
- Většina NoSQL je open-source,
  mají různý přístup k práci s daty a jejich dotazování.
- NoSQL většinou řeší CAP omezením konzistence dat.
  (BASE = Basically Available Soft-state services with Eventual-consistency)

Dokumentová noSQL:

- V podstatě „klíč-hodnota“, ale hodnota je strukturovaná.
  (databáze vidí „dovnitř“, hodnota je pochopena, analyzována)
- Hodnota např. jako XML/JSON, nebo jako objekt.
  (možnost referení na jiné záznamy, vnořování struktur, kolekce)
- Dotazy i složitější, než přes klíče.
  (např. XPath nebo jako v objektových databázích)

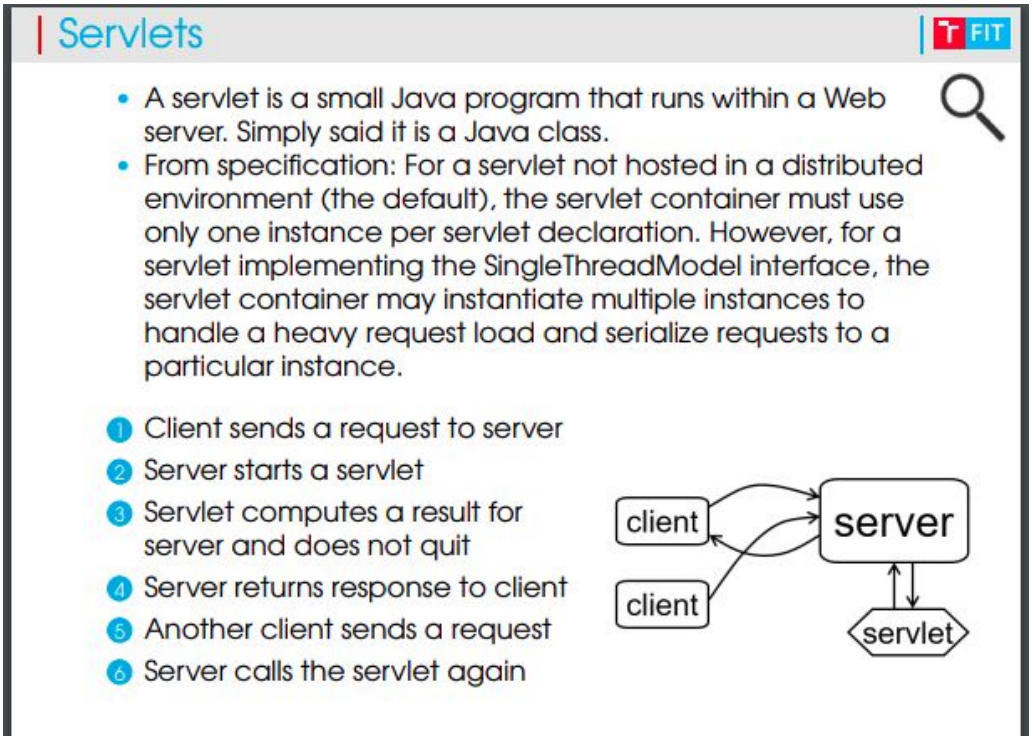
Ukázka práce s mongoDB

```
db.article.insert({
  "name" : "My Article",
  "publish date" : new Date("2013-10-15"),
  "comment" : [],
  "tag" : [ "adventure", "fiction" ]
})

db.article.find({"tag" : "adventure"}).pretty()
{
  "_id" : ObjectID("525c7ed0cc9374393401f5fd"),
  "name" : "My Article",
  "publish date" : ISODate("2013-10-15"),
  "comment" : [],
  "tag" : [
    "adventure",
    "fiction"
  ]
}

db.article.update(
  {"_id" : new ObjectID("525c7ed0cc9374393401f5fd")},
  {$push : { comment : { name : "Alice", comment : "Awesome post!" } } }
)
```

Pozn.: „$push“ je atomický update operátor (další jsou $pop, $pull, . . .).

3, Co je servlet v java EE, na co sluzi, ako sa znaci, ako sa vytvaraju instance. Popiste priebeh spracovania http poziadavku.

slajdy z GJA



```
public class HelloServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
                      throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String docType =
            "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 " +
            "Transitional//EN\">\n";
        out.println(docType +
                    "<HTML>\n" +
                    "<HEAD><TITLE>Hello</TITLE></HEAD>\n" +
                    "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                    "<H1>Hello World</H1>\n" +
                    "</BODY></HTML>");
    }
}
```

4, Popiste CDI, ako sa definuju CDI objekty, kto ich vytvara. Popiste pojem scope v tomto kontexte, vysvetlite request, session, application scope.

Contexts and Dependency Injection (CDI) - Obecný mechanismus pro DI mimo EJB
Omezuje závislosti mezi třídami přímo v kódu

- Flexibilita (výměna implementace), lepší testování, ...

Injektovatelné objekty

- Třídy, které nejsou EJB
- Různé vlastnosti pomocí anotací

Použití objektu

- Anotace @Inject
- CDI kontejner zajistí získáni a dodání instance
- dodání přes property, konstruktor, setter

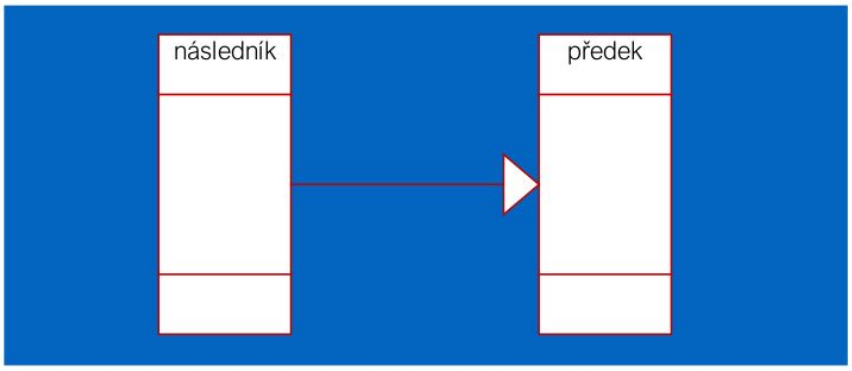
Scope - poskytuje objektu dobře definovaný kontext životního cyklu

- @Dependent – vzniká pro konkrétní případ, zaniká s vlastníkem (default)
- @RequestScoped – trvá po dobu HTTP požadavku
- @SessionScoped – trvá po dobu HTTP session
- @ApplicationScoped – jedna instance pro aplikaci

1) co je obecne dedicnost v objektovem modelu. definice jednoduche a vicenasobne dedicnosti + jejich grafy.

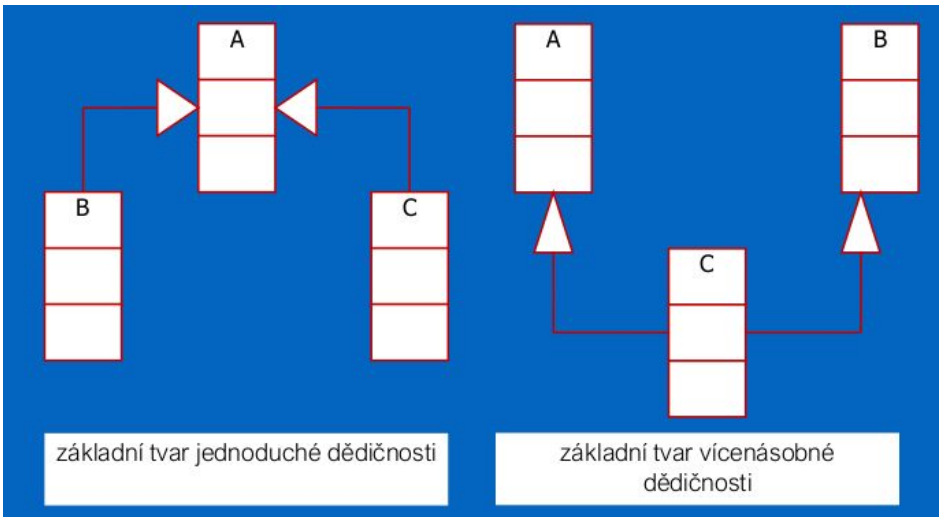
Vazby ***mezi typy*** struktur. Všechny možné vazby diskutované zde se vyskytují pouze separátně mezi stejnými typy struktur, tedy mezi:

- ***objekty*** a
- ***prostými strukturami***.



Ve schématu podle UML jde šipka ve směru generalizace.

- U **jednoduché dědičnosti** každý následník smí mít pouze jediného předka. V grafické podobě dědičnosti to znamená, že ze žádného typu nesmí vycházet více, nežli jedna šipka. Takto zakreslený graf je potom stromem.
- U **vícenásobné dědičnosti** není počet předků omezen. V grafické podobě dědičnosti to znamená, že z každého typu smí vycházet libovolný počet šipek. Takto zakreslený graf je obecný acyklický graf.



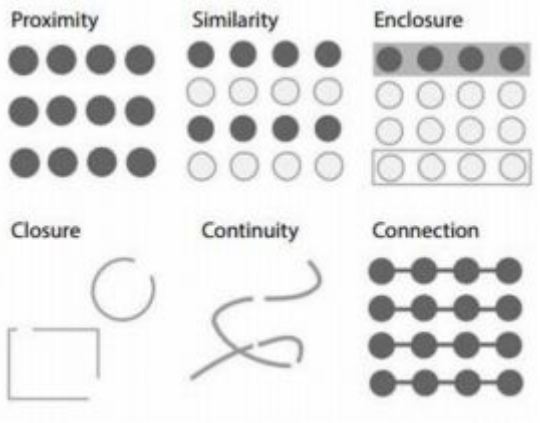
3) jak je resena dedicnost objektove relacnich mapovanich v relacnich tabulkach (nepamatuju si jak to presne bylo formulovane)

OLD>>

1) Definujte multidimenzionalni kostku součtu. Nakreslete svaz pro 4 dimenze, nad touto kostkou definujte operaci rollUp (8b)

2) Zadány Dimenze (lokace, produkty, množství - viz PIS300OLAP.pdf slajd 32 a 34 a 43) a v jazice MDX (asi) napsat dotaz, který pro danou lokaci, daný produkt vypíše součet množství (7b)

3) Gestalt principy, vyjmenovat a nakreslit a popsat (7b)



psychologicka disciplina zamerena na vnimani podnetu clovekem (gestalt - vzor)
- pravidlo **blízkosti** (tendence seskupovat blizke objekty do skupin a uvazovat je jako jeden nadrazeny objekt)
- pravidlo **podobnosti** (tendence seskupovat objekty majici podobne vlastnosti, barva, velikost, tvar, orientace)
- pravidlo **propojování** (tendence seskupovat objekty ktere jsou explicitne propojeny carou, *silnejsi nez blizkost a podobnost*)
- pravidlo **ohraničení** (tendence seskupovat objekty, ktere jsou explicitne ohraniceny, *silnejsi nez blizkost a podobnost, propojeni*)
- pravidlo **návaznosti** (tendence intuitivne dokreslovat objekty, ktere se nachazeji v podobnem smeru)

- pravidlo uzavreni, zavirat prvky do chlivecku, tvaru..

4) Popsat rozhraní a prvky WorkFlow (7b)

Rozhrania:
- pro nástroje pro definici procesů
- pro workflow klienty
- pro volané aplikace
- pro komunikaci s jinými WFM systémy
- pro administraci a monitorování

Prvky:
- klientské aplikace (provádí úkoly, interakce uživatelů s workflow)
- vyvolané aplikace (spouštěné v souvislosti se započetím úkolu)
- nástroje pro definici procesů (obvykle grafické), model obsahuje zprávy, události, rozhodnutí
- nástroje pro simulaci procesů (what if?, ověření, predikce)
- nástroje pro verifikaci (bude každá objednávka vyřízena?, matematické modely - Petriho síťě)
- nástroje pro administraci

5) Popsat RDF, RDF trojici, Graf, Serializaci a význam ontologie (8b)

RDF - odporúčania od W3C, definujú RDF trojice, grafy a prácu s nimi. Popisuje reprezentáciu dát tak, aby boli čitateľné strojovo aj ľudsky. Je to formalismus (datový model) pro grafovou reprezentaci dat – speciálně metadat na WWW.
RDF trojica (zakladny prvok) - subjekt-predikát-objekt (napr. Autorem dokumentu X je pan Y)
Graf - graficky znázorňuje RDF trojice
Serializace - Official RDF/XML serialization, N3 Notation (cleaner), Turtle (Simplified N3)
Ontologie - formálne definuje sémantiku entít a vzťahy medzi nimi. Zameriava sa na určitú doménu napr. rodinné vzťahy, medicínska doména atď.

6) Co je to SPARQL, nad jakými daty pracuje, jaký tvar má Select, jaká data vrací? (7b)

- dotazovací jazyk pro dotazování nad RDF dokumenty
- dotazuje se na RDF trojice (subjekt, predikát, objekt)
- Dotaz zjišťující v kterém státu je Brno:
- `SELECT ?stat FROM <ns> WHERE { ?x ns:jeV ?stat ; ?x ns:nazev "Brno" }`
- **vrací:** `?stat = "Česká republika"`

7) Zadán objektový model (obrázek s třídama, šipkama...) a v ODMG popsat jeden objekt (7b)

Vícetypovost - k čemu je potřeba, jaký je s ní spojen problém, jak se řeší, uvést příklad

vícetypovost je vícenásobná dědičnost pro persistentní objekty prováděná v čase běhu
role: změna role objektu v čase u perzistentních objektů
Pokud objektem modelujeme jistou realnou skutecnost, je predpoklad existence jedineho koncoveho typu je nedostatecna. Pokud modelujeme Osobu, tak muze mit vice roli, muze byt student, muze byt ctenar ve knihovne …
V db aplikacich jsou objekty persistentni a neni mozne predem predpokladat jejich vsechny mozne koncove typy. Napr clovek jak bude rust, tak bude zakem, studentem, pracujicim, duchodcem … ruzne role. Navic lze vselijak kombinovat. Je tedy nutne umoznit vytvareni rychznych kombinaci koncovych typu v jedinem objektu behem jeho existence. A to je prave ta vicetypovost.
Problemy?
Problém kolize: zamezení výskytu nedovolených kombinací mohou zajistit relace současné a výlučné existence místo relace dědičnosti.

Nakreslit tabulku která bude výsledkem MDX dotazu ve stylu:
SELECT [Measures].[Amount] ON COLUMNS FROM [Sales]
(Výsledkem bude jediná hodnota představující součet všech prodejů napříč celou databází)

Definovat funkce pro map a reduce pro vytvoření tabulky součtů group by (Product, Country) z multidimenzionální kostky (Time, Product, Country, ...)

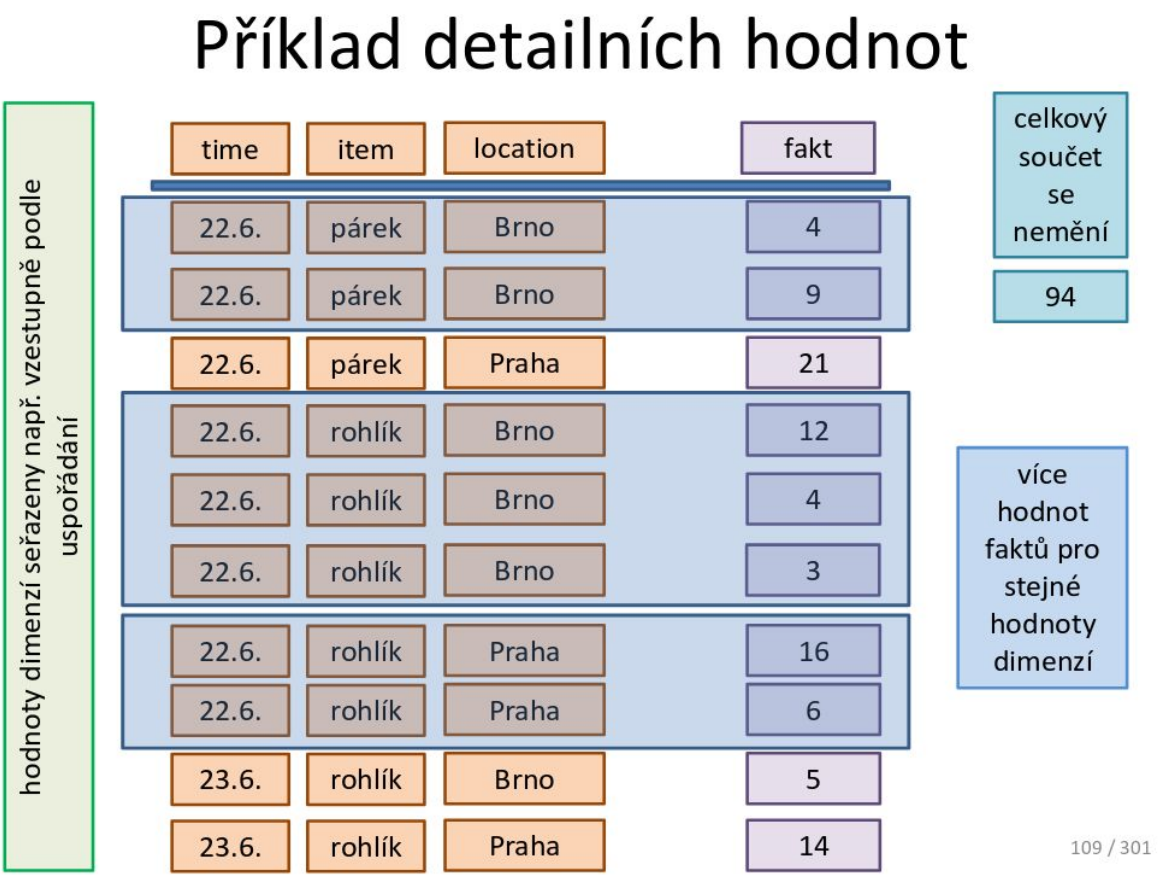
PIS příprava na semestrálku 2017

jako vždy - doplňujte obsah, komentujte pomocí komentářů

Loňská příprava: https://goo.gl/w92VC3

Riadny termin 2017

1, definovat multidimenzionalni kostku, nakreslit 4D kostku pro time, item, location, supplier. Popísať kostku pro počet



Ako by ste robili tu ten pocet ? Ma byt ten pocet = 10 alebo 5 ? Maju sa najskvor zlucit tie riadky, ktore maju rovnake hodnoty a az potom sa rata ten pocet?

Otázky 2016 řádný termín

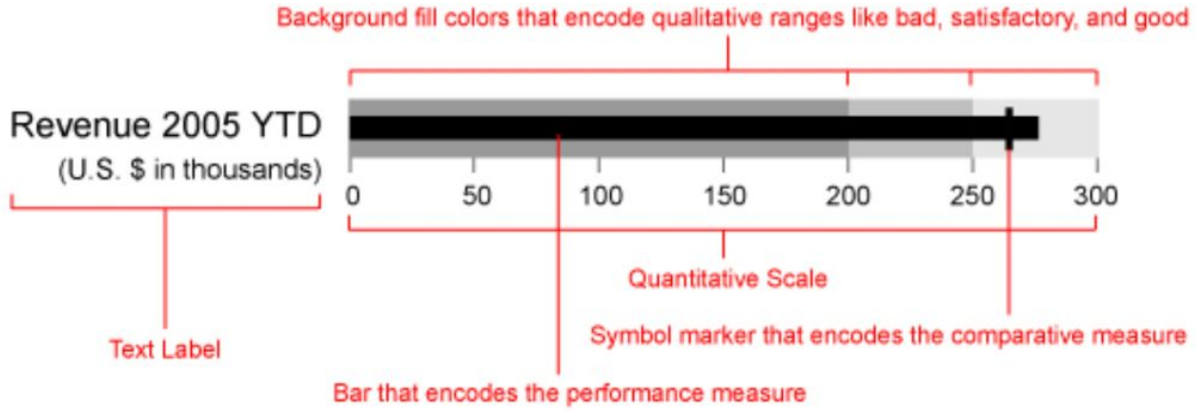
1. **Vícetypovost - k čemu je potřeba, jaký je s ní spojen problém, jak se řeší, uvést příklad**

- vícetypovost je vícenásobná dědičnost pro persistentní objekty prováděná v čase běhu
- **koncové typy** - objekt je standardně vytvářen tak, že je vybrán jediný z typů a objekt je pak instancí tohoto typu. Je dále zaručeno, že není již žádného dalšího typu, který je následníkem vybraného typu (tzv. koncový typ).
- Opravovat něco na poslední chvíli, to už je Pokud objektem modelujeme jistou reálnou skutečnost, je předpoklad existence jediného koncového typu nedostatečný.
- v DB aplikacích však objekt persistentní je a není možné předem předpokládat všechny možné kombinace koncových typů, obzvláště u složitých grafů dědičnosti,
- je tedy nutné umožnit vytváření různých kombinací koncových typů v jediném objektu během jeho existence. Tento jev se nazývá **vícetypovost** .
- **Problém kolize:** některé kombinace typů se nemohou vyskytovat zároveň (např. osoba nemůže být zároveň dítě i dospělý)

- zamezení výskytu nedovolených kombinací mohou zajistit **relace současné** a **výlučné existence** místo relace dědičnosti.

2. Jak vypadá Bullet graf, co z něj lze vyčíst

- funguje na stejném principu jako stupnice na teploměru
- šikovná náhrada za kruhová měřidla (tachometry , gauges)
- kompaktní, jednorozměrné
- jednotlivé bullet grafy je možné skládat vedle sebe



3. Nakreslit tabulku která bude výsledkem MDX dotazu ve stylu: SELECT [Measures].[Amount] ON COLUMNS FROM [Sales]

Výsledkem bude jediná hodnota představující součet všech prodejů napříč celou databází.

Amount
<číslo>

Příklad se slidů:

pouze sloupce

```
SELECT ([Product].[Prod].[Licence].[Corporate],
        [Geography].[Geo].[Continent].[America]) ON COLUMNS
FROM [Sales]
```

- Výpis uspořádané n-tice v jednom sloupci, celková utržená částka za zboží.

Corporate
America
768

```
SELECT
    ([Product].[Prod].[Licence].[Corporate],
     [Geography].[Geo].[Continent].[America]) ON COLUMNS
FROM
    [Sales]
```

Corporate
America
768

4. **Definovat funkce pro map a reduce pro vytvoření tabulky součtů group by (Product, Country) z multidimenzionální kostky (Time, Product, Country, ...)**

Obecně:
Map/Reduce: funkcie z LISPu
Map: parametrami sú funkcia a množina hodnôt, výstupom je množina výsledkov po aplikácii funkcie na všetky hodnoty
Reduce: parametrom je binárny operátor a množina hodnôt, výstupom je jeden výsledok po aplikovaní binárneho operátora na všetky hodnoty reduce '+' (0 1 2 3) -> ((0+1)+2)+3=6

//TODO pro tento konkrétní příklad?
vubec nevim je to spravne nebo ne, ale podle meho nazoru je neco takoveho:
Map (key: (Product, Country), value: ~~Time~~ fakt (například utržené peníze)) z ceho dostaneme paralelne seznam hodnot, a pak pomoci
Reduce (+) (k_i , list (v_i)) spocitame kolik vyskytu kazdeho (Product, Country) mame.

Ale je to jen **muj nazor.**

Pokus 2 len teoreticky:
Uvazujme, ze multidimenzionalna kocka je reprezentovana ako dvojdimenzionalne pole respektive tabulka. Stlpce su jednotlive dimenzi v poradi podla zadania. Posledny stlpec obsahuje fakt.
Map funkcia pomocou operatora (prvy argument) transformuje kocku (druhy argument), tak ze sa vynechava stlpec cas tj. prva polozka v kazdom riadku.

Takto aggregovane hodnoty z viacerych uzlov vstupuju do reduce funkcie ako druhy parameter. Prvy parameter je binarna funkcia, ktorej vysledkom je tabulka v ktorej kazdy riadok resp. kombinacia produkt a cas je unikatna. Duplikovane riadky su aggregovane scitanim faktu.

5. Popište prvky Workflow systému a s jakými daty pracují (řídící/věcná/aplikační)

Prvky WF:

1. WES servery (složen z 1 nebo více Workflow engine)
2. klientské aplikace (provádí úkoly, interakce uživatelů s workflow)
3. vyvolané aplikace (spouštěné v souvislosti se započetím úkolu)
4. nástroje pro definici procesů (obvykle grafické) , model obsahuje zprávy, události, rozhodnutí
5. nástroje pro simulaci procesů (what if?, ověření, predikce)
6. nástroje pro verifikaci (bude každá objednávka vyřízena?, matematické modely - Petriho síťě)
7. nástroje pro administraci

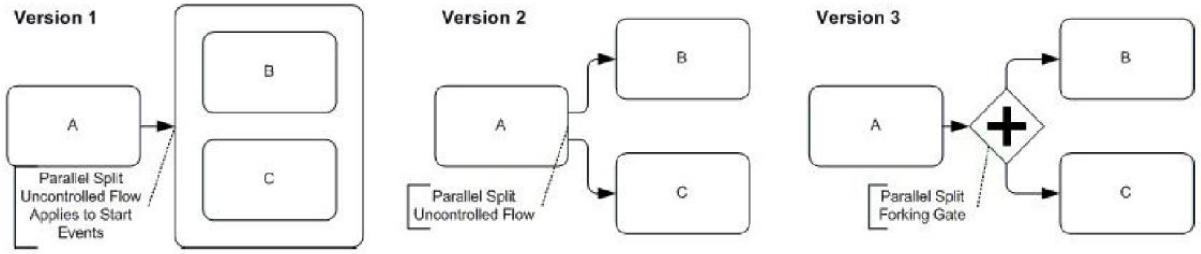
Data WF:

- Model organizační struktury
  - Role, vztahy nadřízený – podřízený
- Definice procesu
  - Činnosti, přidělení rolím, rozhodovací pravidla
- Seznam úkolů
  - Aktuální úkoly pro konkrétní uživatele
  - Uživateli buď skryt (postupné přidělování úkolů) nebo přístupný (uživatel si volí pořadí, možno i více úkolů současně)
- Řídicí data workflow
  - Interní data WF systému nutná pro zajištění chodu příp. zotavení po havárii
  - Nedostupná externím aplikacím
- Věcná data workflow
  - Zpracování jádrem workflow systému
  - Používána pro rozhodování o dalším postupu
  - Dostupná i aplikacím
- Aplikační data workflow
  - Specifická data aplikací podporujících proces
  - Nejsou přístupná WF systému

6. Nakreslete a popište co dělají jednotlivé prvky řízení toku ve workflow: AND-split, AND-join, XOR-split, XOR-merge

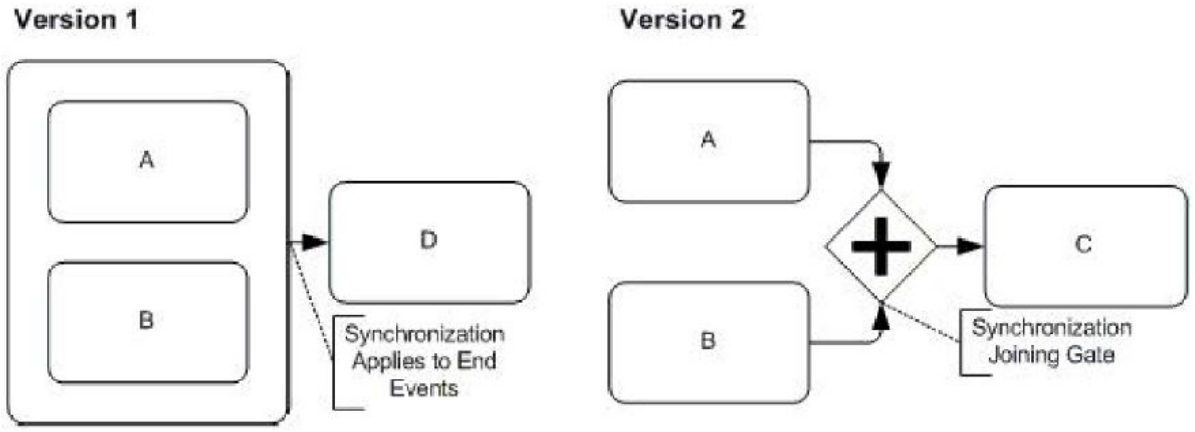
Parallel split (AND - split)

- Rozděluje tok procesů (workflow) do dvou a více paralelních vláken



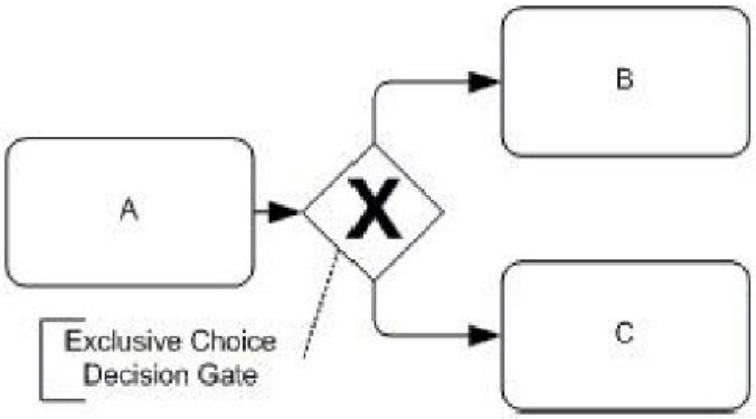
Synchronizace (AND - join)

- Dalším úkolem se pokračuje, až po dokončení všech předchozích vláken.



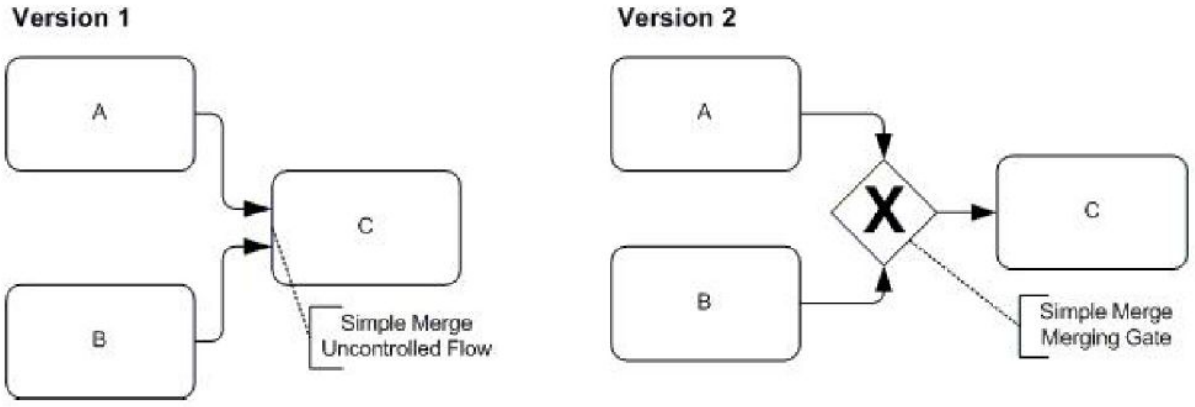
Výlučné rozhodnutí (XOR - split)

- Rozděluje tok procesů na dvě nebo více větví, které jsou vzájemně výlučné. Podle podmínky v Gateway se vstupuje do jedné z větví.



Jednoduché spojení (XOR - merge)

- Spojení dvou nebo více nezávislých větví do jedné. Navazující aktivita začne okamžitě, jakmile jedno vlákno dosáhne svého konce.



Základní vlastnosti transakce

Atomičnost (Atomicity) – každá transakce je dokončena zcela nebo vůbec (zajišťuje TPS)

Konzistence (Consistence) – databázová konzistence (správná reflexe stavu reálného světa a
dodržování omezujících pravidel pro hodnoty) **(zajišťuje uživatel)**

Izolovanost (Isolation, Independence) – souběžné provádění má totožný efekt jako sekvenční
(TPS)
Trvanlivost (Durability) – odolnost proti ztrátě již dokončených změn (TPS)
Ez
Pozn.: TPS = Transactional Processing System

Kdyby náhodou:

úroveň	popis	příklad pro relační databázový model	příklad pro objektový model
meta^{n}data	atd. cestou odvození uvedených pojmů k základním definicím, se zvyšující se mocninou se používá obecnějších modelů vesměs matematických		
meta^{5}data	jak vypadá množina, uspořádání atd.	uspořádání je binární relace na množině tranzitivní, reflexivní, antisymetrická	
meta^{4}data	jak vypadá struktura a kolekce	kartézský součin je množina uspořádaných dvojic; uspořádaná multimnožina je množina obecně obsahující více stejných prvků s definovaným uspořádáním	
meta^{3}data	jak vypadá databázový model – základní stavební kameny struktura, kolekce	struktura je kartézský součin dvojic; kolekce je uspořádaná multimnožina (zde a výše už to nezáleží na modelu)	
meta^{2}data	jak vypadá katalog – databázový model (schéma)	relace je kolekce struktur	objekt je pojmenovaná struktura s libovolnou úrovní vnoření struktur a kolekcí
meta^{1}data = metadata	jak vypadá výskyt - katalog	relace faktura má domény číslo, adresát a klíčem je vázána na relaci položky	objekt faktura má položku číslo a vnořenou prostou struktura adresa, vnořenou kolekci struktur položka atd.
meta^{0}data = data	výskyt	faktura číslo 1200005, Jan Novák, 1 ks telefon, 2500 Kč, celkem 2500 Kč atd. (výskyt prvku relace nebo objektu)	

Dělení grafů podle účelů:

Dělení dle účelu

- **hodnota v rozsahu:** gauge, bullet graph
- **porovnání, ranking:** bar chart, spider/radar, wordcloud, parallel, …
- **distribuce hodnot:** histogram, boxplot, density, violin, …
- **korelace:** scatter plot, heatmap, bubble, …
- **vývoj (v čase):** line plot, sparkline, candlestick chart, area, stacked area, , …
- **část celku, hierarchie:** pie/donut, stacked bars, treemap, circular packing, dendrogam, …
- **flow:** Sankey, chord, edge bundling, network, …
- **mapy, geovizualizace:** choropleth, map with markers (bubble map), connection, cartogram, …

https://www.d3-graph-gallery.com

https://datavizcatalogue.com