Pokročilé informační systémy

Principy, data, modely, architektury

prof. Ing. Tomáš Hruška, CSc.
doc. Ing. Radek Burget, Ph.D.
burgetr@fit.vutbr.cz

Předmět PIS – Cíle předmětu

- Návrh informačního systému – rozšíření znalostí z IIS
  - Analýza domény a procesů (datové modelování, workflow)
  - Pokročilé architektury systémů
  - Systémy pro podporu rozhodování
- Zvládnutí pokročilých technologií
  - Databázová vrstva – objektový datový model, alternativy
  - Aplikační vrstva – aplikační rámce pro enterprise aplikace
  - Prezentační vrstva – klientské programování
- Business intelligence (OLAP systémy)

Technologie

- Cílem je pochopit základní principy **nezávisle na implementační platformě**
- Prakticky zvládnout pokročilá technická řešení
  - Jakarta EE (dříve Java Enterprise Edition), Microprofile
    - Alternativy na jiných platformách
- Praktické ukázky (a projekt) budou využívat zejména
  - Eclipse Microprofile (aplikační logika, REST rozhraní, provoz, monitorování)
  - Objektově-relační mapování (JPA)
  - Pokročilé API architektury (GraphQL, SOAP, …)

Hodnocení

- **Půlsemestrální zkouška:** 19 bodů
- **Semestrální zkouška:** 51 bodů
- **Projekt:** 30 bodů (8 + 22)
  - Realizace IS na dané téma v týmu (zadání z AIS)
  - ORM datová vrstva
  - Pokročilý aplikační framework umožňující oddělení datové, aplikační a prezentační vrstvy
    - Monolitický backend (Java nebo .NET) + JS frontend (Angular, Vue.js, React, …)
    - Mikroslužby (microprofile + libovolné další), Docker, orchestrace

Kontakty

- **doc. Ing. Radek Burget, Ph.D.**
  burgetr@fit.vut.cz
  - **Přednášky**, zkoušky, projekty, všechno ostatní
- **Ing. Jiří Hynek, Ph.D.**
  hynek@vut.cz
  - Vybrané přednášky

Data – informace – znalosti

Data

- Hodnota schopná přenosu, uchování, interpretace či zpracování
- Z hlediska IT jde o *hodnoty* různých *datových typů*
- Data sama o sobě *nemají sémantiku* (význam), jsou to věty nějakého formálního jazyka
  - Viz pojem *databáze*
- Hodnoty dat obvykle udávají *stav* nějakého systému

Informace

- *Informace* jsou interpretovaná data
- Mají *sémantiku* (význam)
- Transformaci dat na informace neprovádí informační systém, ale *uživatel*
  - Systém ukládá a transformuje *data*
  - Pro uživatele výsledek znamená *informaci*
- Je nezbytné zajistit shodnou interpretaci dat u všech uživatelů informace
  - Vzdělání, školení, zavedení konvencí

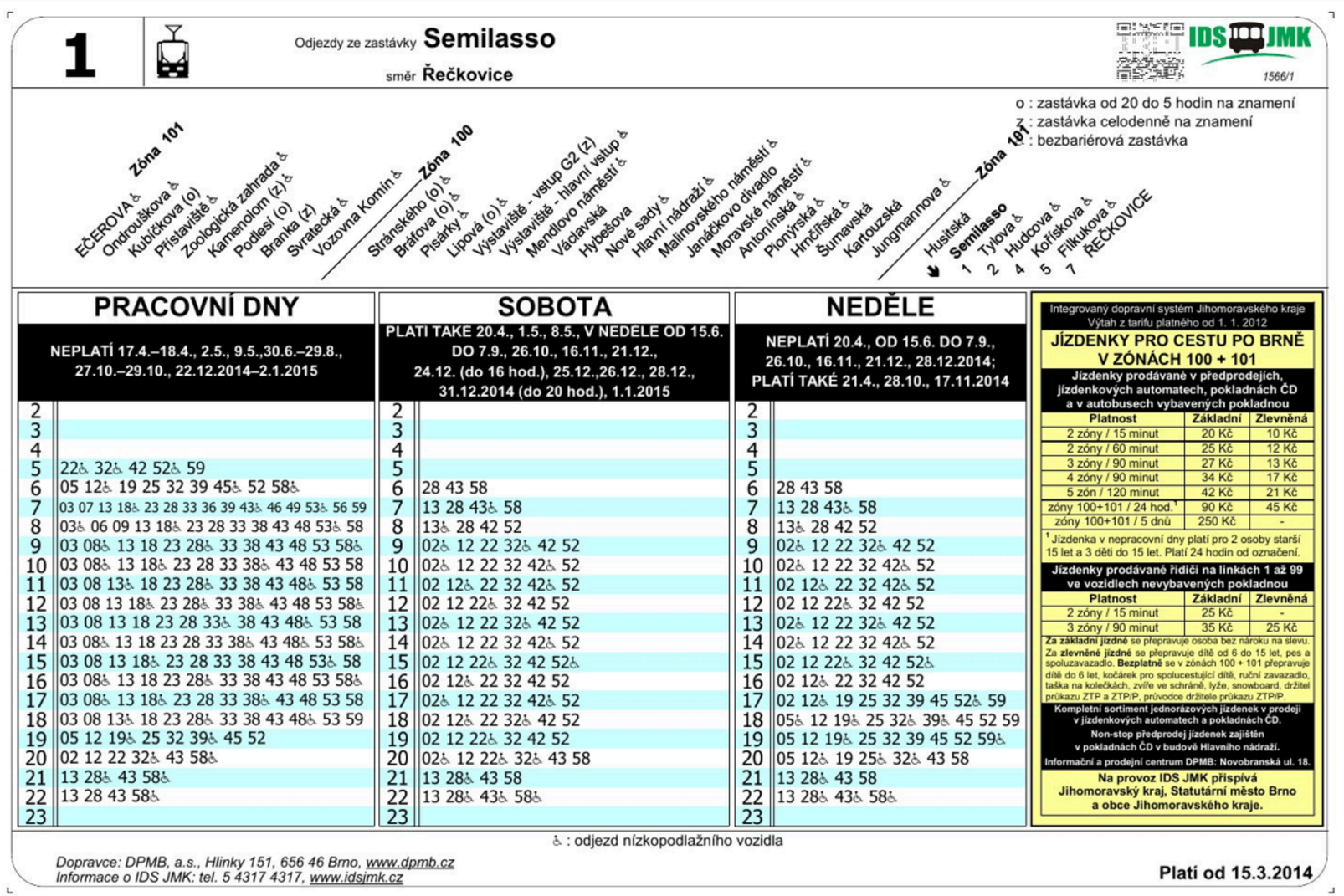
Příklad rozdílné interpretace dat

- Údaj 10-12-2005
  - V Evropě informace 10. prosince 2005
  - V USA informace 12. října 2005
- Pro totožná data vznikne *rozdílná informace* jinou *interpretací* dat
- Podobně např. jméno a příjmení

Znalost

- Informace zařazená do souvislostí
- Jejich interpretace je však ještě hůře definovatelná, neboť může jít o celé shluky informací
- Znalosti chápeme často jako *sekundární odvozené informace*
- Některé informační systémy se zabývají pouze *informacemi (transakční)*, některé pracují se *znalostmi (pro podporu rozhodování a plánování)*
- Problematika *získávání znalostí z dat* (knowledge discovery, data mining)
  - Předmět Získávání znalostí z databází (ZZN)

Příklad: jízdní řád



1

Odjezdy ze zastávky **Semilasso**

směr **Řečkovice**

IDS JMK

1566/1

o : zastávka od 20 do 5 hodin na znamení
z : zastávka celodenně na znamení
♿ : bezbariérová zastávka

Zóna 101: ♿ EČEROVA, ♿ Ondrouškova (o), ♿ Kubíčkova (o), ♿ Přístaviště, ♿ Zoologická zahrada, ♿ Kamenolom (z), ♿ Podlesí (o), ♿ Branka (z), ♿ Svratecká, ♿ Vozovna Komín

Zóna 100: ♿ Stránského (o), ♿ Bráfova (o), ♿ Písárky, ♿ Lipová (o), ♿ Výstaviště - vstup G2 (z), ♿ Výstaviště - hlavní vstup, ♿ Mendlovo náměstí, ♿ Václavská, ♿ Hybešova, ♿ Nové sady, ♿ Hlavní nádraží, ♿ Malinovského náměstí, ♿ Janáčkovo divadlo, ♿ Moravské náměstí, ♿ Antonínská, ♿ Pionýrská, ♿ Hrnčířská, ♿ Šumavská, ♿ Kartouzská, ♿ Jungmannova

Zóna 101: ♿ Husitská, **Semilasso**, ♿ Tylova, ♿ Hudcova, ♿ Kořískova, ♿ Filkuková, ♿ ŘEČKOVICE

1 2 4 5 7

PRACOVNÍ DNY

NEPLATÍ 17.4.–18.4., 2.5., 9.5.,30.6.–29.8., 27.10.–29.10., 22.12.2014–2.1.2015

2	
3	
4	
5	22♿ 32♿ 42 52♿ 59
6	05 12♿ 19 25 32 39 45♿ 52 58♿
7	03 07 13 18♿ 23 28 33 36 39 43♿ 46 49 53♿ 56 59
8	03♿ 06 09 13 18♿ 23 28 33 38 43 48 53♿ 58
9	03 08♿ 13 18 23 28♿ 33 38 43 48 53 58♿
10	03 08♿ 13 18♿ 23 28 33 38♿ 43 48 53 58
11	03 08 13♿ 18 23 28♿ 33 38 43 48♿ 53 58
12	03 08 13 18♿ 23 28♿ 33 38♿ 43 48 53 58♿
13	03 08 13 18 23 28 33♿ 38 43 48♿ 53 58
14	03 08♿ 13 18 23 28 33 38♿ 43 48♿ 53 58♿
15	03 08 13 18♿ 23 28 33 38 43 48 53♿ 58
16	03 08♿ 13 18 23 28♿ 33 38 43 48 53 58♿
17	03 08♿ 13 18♿ 23 28 33 38♿ 43 48 53 58
18	03 08 13♿ 18 23 28♿ 33 38 43 48♿ 53 59
19	05 12 19♿ 25 32 39♿ 45 52
20	02 12 22 32♿ 43 58♿
21	13 28♿ 43 58♿
22	13 28 43 58♿
23	

SOBOTA

PLATÍ TAKÉ 20.4., 1.5., 8.5., V NEDĚLE OD 15.6. DO 7.9., 26.10., 16.11., 21.12., 24.12. (do 16 hod.), 25.12.,26.12., 28.12., 31.12.2014 (do 20 hod.), 1.1.2015

2	
3	
4	
5	
6	28 43 58
7	13 28 43♿ 58
8	13♿ 28 42 52
9	02♿ 12 22 32♿ 42 52
10	02♿ 12 22 32 42♿ 52
11	02 12♿ 22 32 42♿ 52
12	02 12 22♿ 32 42 52
13	02♿ 12 22 32♿ 42 52
14	02♿ 12 22 32 42♿ 52
15	02 12 22♿ 32 42 52♿
16	02 12♿ 22 32 42 52
17	02♿ 12 22 32 42♿ 52
18	02 12♿ 22 32♿ 42 52
19	02 12 22♿ 32 42 52
20	02♿ 12 22♿ 32♿ 43 58
21	13 28♿ 43 58
22	13 28♿ 43♿ 58♿
23	

NEDĚLE

NEPLATÍ 20.4., OD 15.6. DO 7.9., 26.10., 16.11., 21.12., 28.12.2014; PLATÍ TAKÉ 21.4., 28.10., 17.11.2014

2	
3	
4	
5	
6	28 43 58
7	13 28 43♿ 58
8	13♿ 28 42 52
9	02♿ 12 22 32♿ 42 52
10	02♿ 12 22 32 42♿ 52
11	02 12♿ 22 32 42♿ 52
12	02 12 22♿ 32 42 52
13	02♿ 12 22 32♿ 42 52
14	02♿ 12 22 32 42♿ 52
15	02 12 22♿ 32 42 52♿
16	02 12♿ 22 32 42 52
17	02 12♿ 19 25 32 39 45 52♿ 59
18	05♿ 12 19♿ 25 32♿ 39♿ 45 52 59
19	05 12 19♿ 25 32 39 45 52 59♿
20	05 12♿ 19 25♿ 32♿ 43 58
21	13 28♿ 43 58
22	13 28♿ 43♿ 58♿
23	

Integrovaný dopravní systém Jihomoravského kraje
Výtah z tarifu platného od 1. 1. 2012

JÍZDENKY PRO CESTU PO BRNĚ V ZÓNÁCH 100 + 101

Jízdenky prodávané v předprodejích, jízdenkových automatech, pokladnách ČD a v autobusech vybavených pokladnou

Platnost	Základní	Zlevněná
2 zóny / 15 minut	20 Kč	10 Kč
2 zóny / 60 minut	25 Kč	12 Kč
3 zóny / 90 minut	27 Kč	13 Kč
4 zóny / 90 minut	34 Kč	17 Kč
5 zón / 120 minut	42 Kč	21 Kč
zóny 100+101 / 24 hod.[1]	90 Kč	45 Kč
zóny 100+101 / 5 dnů	250 Kč	-

[1] Jízdenka v nepracovní dny platí pro 2 osoby starší 15 let a 3 děti do 15 let. Platí 24 hodin od označení.

Jízdenky prodávané řidiči na linkách 1 až 99 ve vozidlech nevybavených pokladnou

Platnost	Základní	Zlevněná
2 zóny / 15 minut	25 Kč	-
3 zóny / 90 minut	35 Kč	25 Kč

Za základní jízdné se přepravuje osoba bez nároku na slevu. Za **zlevněné jízdné** se přepravuje dítě od 6 do 15 let, pes a spoluzavazadlo. **Bezplatně** se v zónách 100 + 101 přepravuje dítě do 6 let, kočárek pro spolucestující dítě, ruční zavazadlo, taška na kolečkách, zvíře ve schráně, lyže, snowboard, držitel průkazu ZTP a ZTP/P, průvodce držitele průkazu ZTP/P.

Kompletní sortiment jednorázových jízdenek v prodeji v jízdenkových automatech a pokladnách ČD.

Non-stop předprodej jízdenek zajištěn v pokladnách ČD v budově Hlavního nádraží.

Informační a prodejní centrum DPMB: Novobranská ul. 18.

Na provoz IDS JMK přispívá Jihomoravský kraj, Statutární město Brno a obce Jihomoravského kraje.

♿ : odjezd nízkopodlažního vozidla

Dopravce: DPMB, a.s., Hlinky 151, 656 46 Brno, www.dpmb.cz
Informace o IDS JMK: tel. 5 4317 4317, www.idsjmk.cz

Platí od 15.3.2014

Správa informací

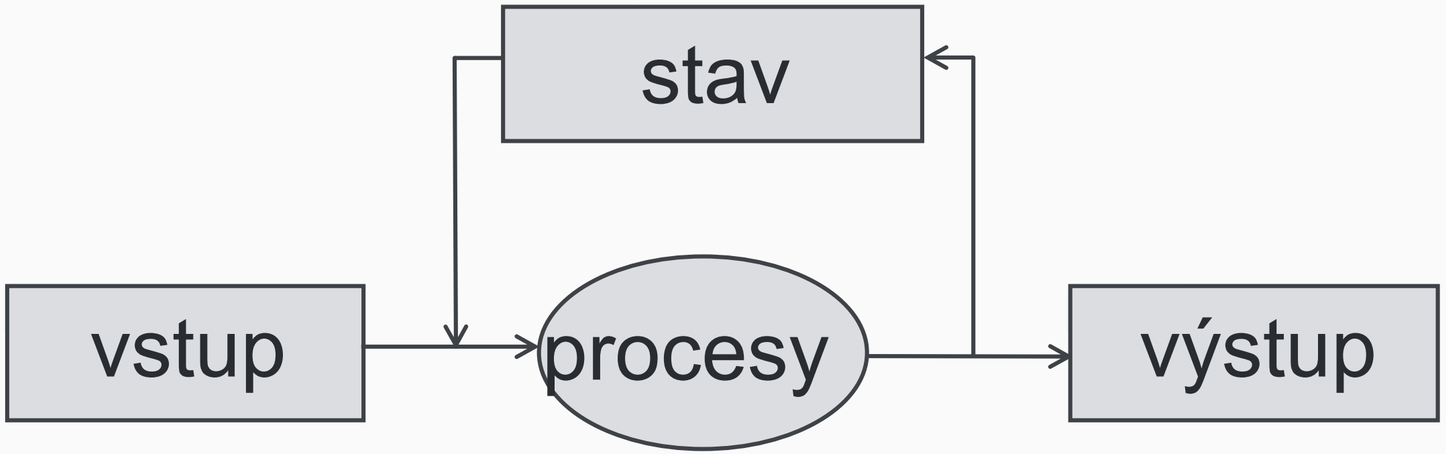
1. Sběr,
2. Uspořádání a příprava,
3. Užití,
4. Rušení a náhrada.

Správa probíhá dle základních funkcí *systému*

- stav, data (zpětná vazba)
- transformace a procesy
- vstup a výstup (komunikace)

Informační systém

Schéma informačního systému



- Modifikované schéma obecného systému
- Data uchovávající *stav* systému a
- *Procesy* realizující transformace často ve formě *transakcí*

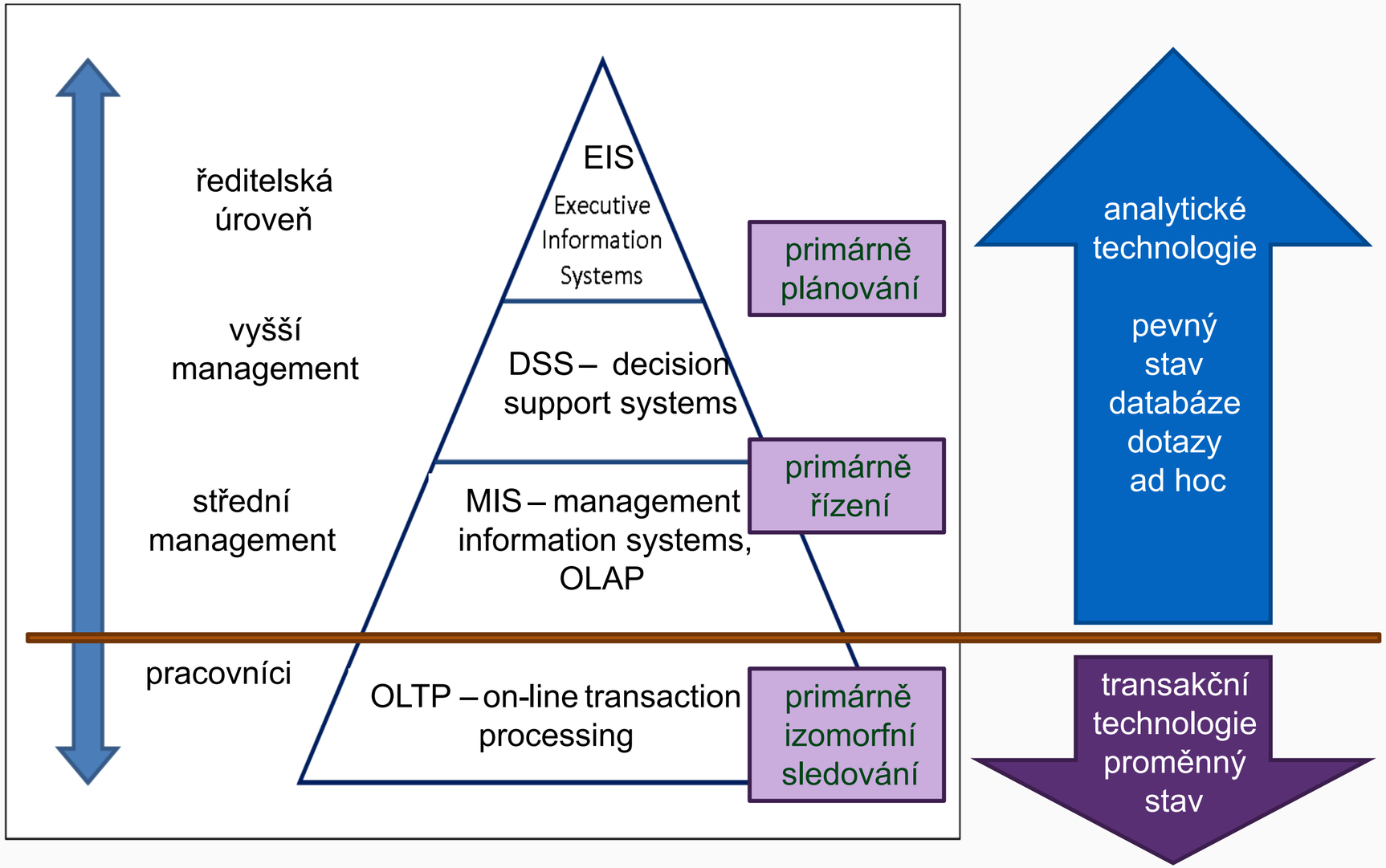
Stav informačního systému

- Stavem informačního systému jsou hodnoty dat (typicky reprezentované pomocí nějakého *modelu*) a musíme se zabývat jejich
  - *Persistencí* (přetrváváním),
  - *Konzistencí* (splňování jistých pravidel o možných kombinacích hodnot údajů ve stavu) apod.

IS podle úrovně rozhodování

- Klasické *pyramidové schéma*
- Odráží hierarchii úrovně rozhodování v organizaci:
  - Systém pro zpracování transakcí
  - Management information systems
  - Decision support systems
  - Executive information systems

Pyramidové schéma



OLTP – On-Line Transaction Processing

- Třída informačních systémů, které zpracovávají transakčně orientované aplikace
- Termín transakční je dvojznačný:
  - *databázové transakce*
  - *komerční (business) transakce*
- (mohou se ovšem překrývat)
- Systém odpovídá na požadavky uživatele okamžitě *změnou stavu*.

MIS - Management Information Systems

- Překládáme *Informační systémy pro podporu řízení*
- Poskytují informace, které jsou potřebné pro efektivní řízení organizace
- MIS je obecně užívány pro skupinu metod zpracování informací určených k automatizaci a podpoře rozhodování
- Nemusejí nutně pracovat nad aktuálním modelem fyzického systému (povoleno zpoždění)
- Nejčastěji jde o:
  - Systémy pro podporu rozhodování (DSS)
  - Expertní systémy (ES)
  - Informační systémy pro exekutivu (EIS)
  - OLAP (Online Analytical Procesing)

Návrh informačního systému

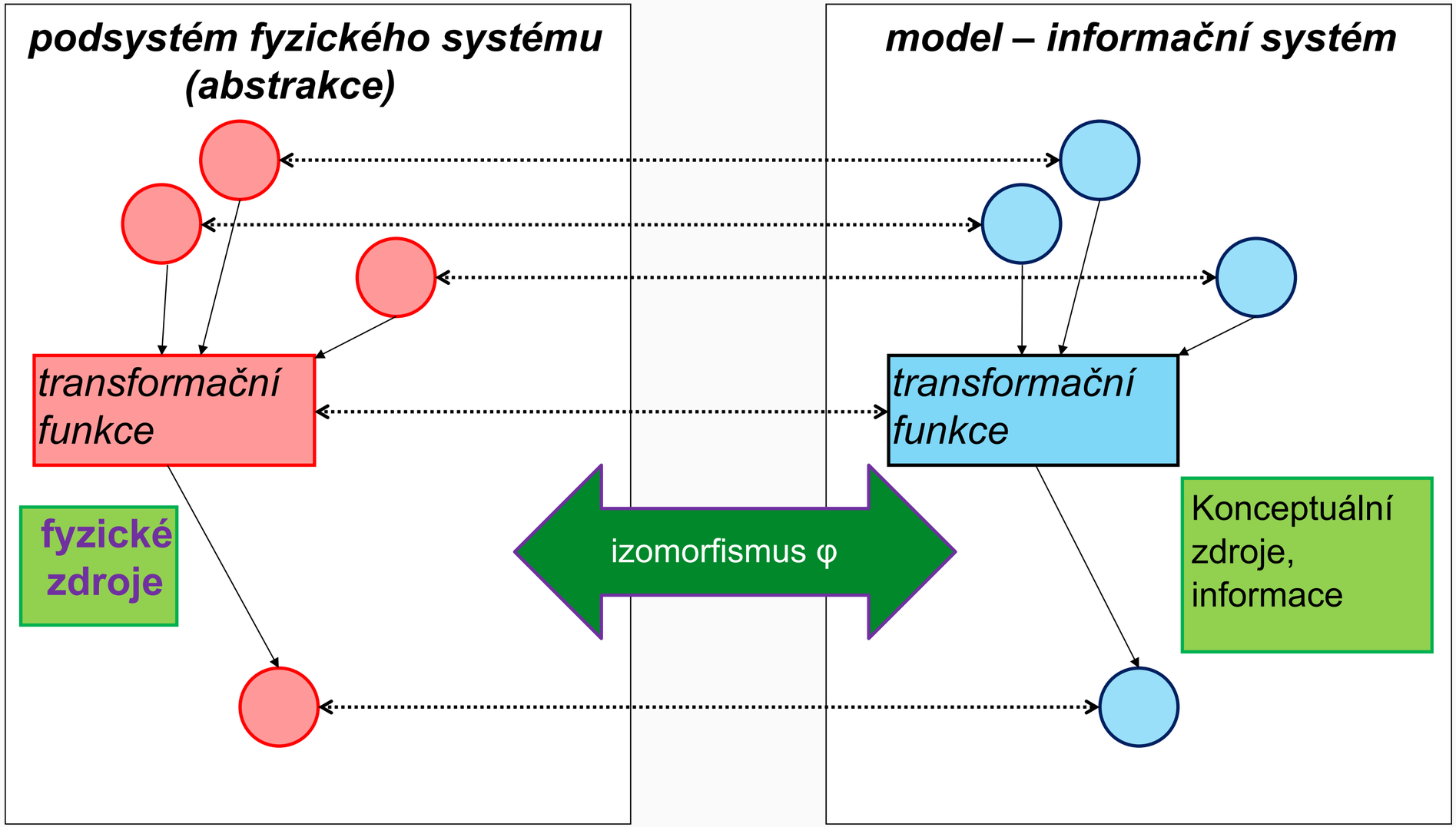
Informační systém jako model

- Informace **modelují skutečné zdroje jiného** – obvykle **fyzického systému** (např. podniku)
- Informační systém tedy na nehmotné – virtuální úrovni **modeluje** svůj fyzický vzor, pro jehož řízení je obvykle vytvářen. Vzhledem k tomu, že model nikdy **nemůže postihnout veškeré chování a vlastnosti svého vzoru**, je virtuální kopie pořizována vždy na vhodné úrovni **abstrakce**.

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.

OLTP jako model



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**.

OLTP jako model

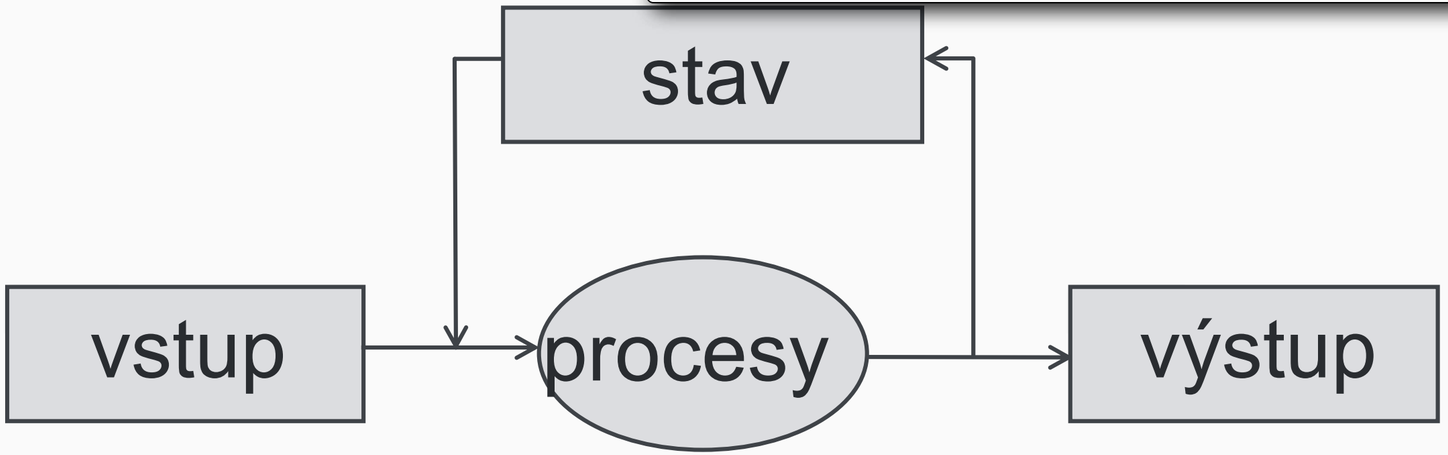
- Říkáme, že OLTP modeluje nějaký fyzický podsystém.
- Mezi OLTP a jeho fyzickým vzorem existuje izomorfismus φ.
- Pokud je v původním systému funkce nad zdroji, potom v OLTP existuje obraz této funkce pracující s obrazy zdrojů.
- Pokud funkce v původním systému má za parametry jisté zdroje a dává jistý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

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.

Návrh informačního systému

- **S jakými daty pracujeme?**
- Tzn. jak je reprezentován **stav IS**?
- Analýza domény, model, persistence, konzistence, …



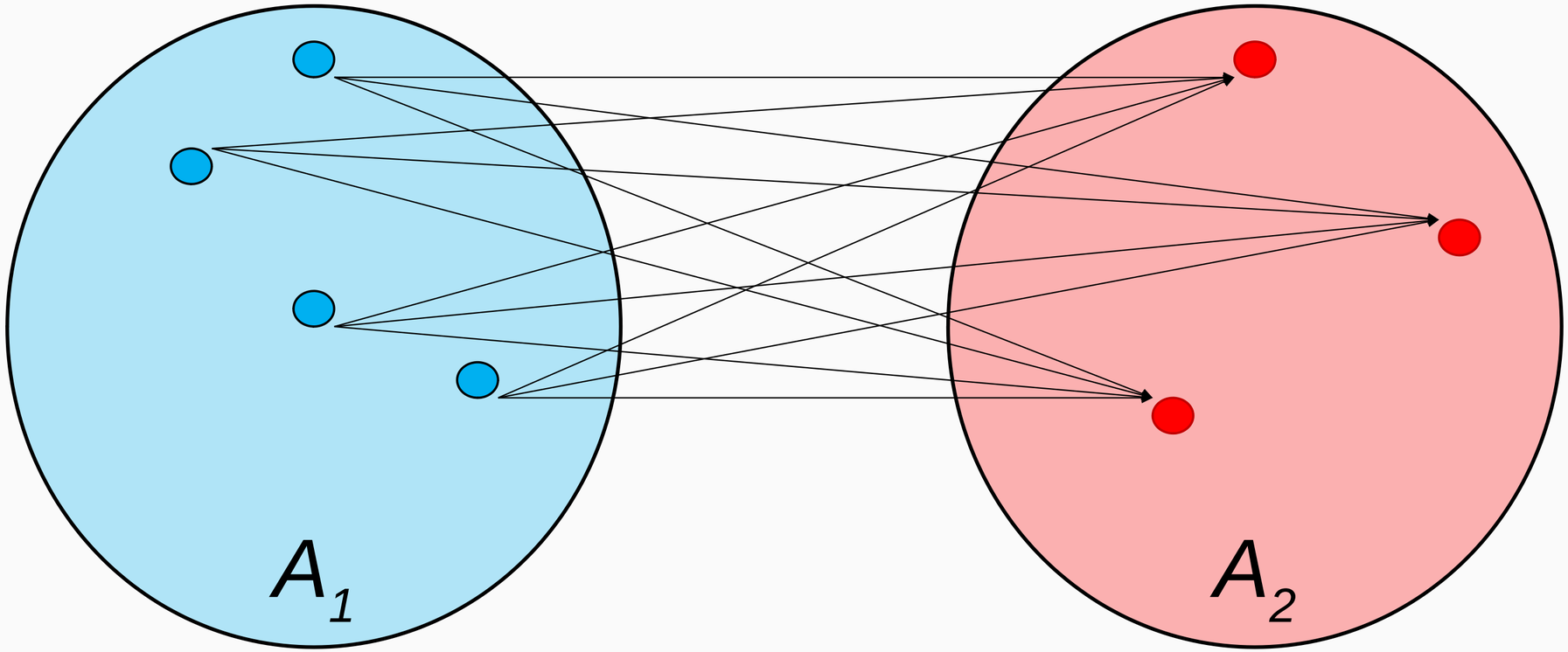
- **Jaké jsou k dispozici vstupy?**
- Jak se informace pořizují, kde jsou zdroje?

- **Jak je třeba data transformovat?**
- Jaké jsou procesy a postupy v cílovém řešení?

- **Jak mají vypadat výstupy?**
- Vždy to nejlépe vídalo účelu systému?

Data a metadata

Kartézský součin A1 x A2



Výsledkem je **množina dvojic** – celkem $4x3=12$ dvojic.

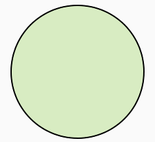
Kartézský součin

- *Uspořádaná n-tice* `$(a_1, a_2, \dots, a_n)$`
- *Kartézský součin množin* `$A_1 \times A_2 \times \dots \times A_n$` je množina všech uspořádaných n-tic takových, že `$a_1 \in A_1, a_2 \in A_2, \dots, a_n \in A_n$`
- Podstatné je, že v uspořádané n-tici je každá hodnota prvkem jediné z množin v kartézském součinu a to té, která jí indexem odpovídá

Struktura a kolekce

Základní typy

- Základní nestrukturované datové typy:
  - Celočíselné
  - Reálná čísla
  - Znaky / řetězce
  - Datum/čas
  - Výčtové typy apod.



Strukturované datové typy

- Strukturovaný datový typ = datová struktura = *metadata*
  - Jak z jednodušších datových typů (ať už základních nebo i jednodušších strukturovaných) budovat složitější.
- Existují základní dva způsoby, jak strukturované datové typy vytvářet:
  - *struktura* a
  - *kolekce*.
- Vše je definováno předem, před vznikem hodnoty

Struktura

- Strukturované hodnoty vytvářené:
  - **Pevným počtem** dílčích hodnot obecně **různých** typů
  - Tedy uspořádané n-tice, které jsou prvky kartézského součinu množin dílčích datových typů
  - Hodnoty jsou pojmenované, tzn. přistupuje se k nim přes jejich unikátní jméno
- Jako synonymum pro uspořádanou n-tici (tedy hodnotu) je velmi často užíván termín *struktura* nebo *záznam*. Jako synonymum pro kartézský součin (tedy datový typ) budeme často používat *typ struktura* nebo *typ záznam*.

Schéma struktury

Příklad datového typu struktura

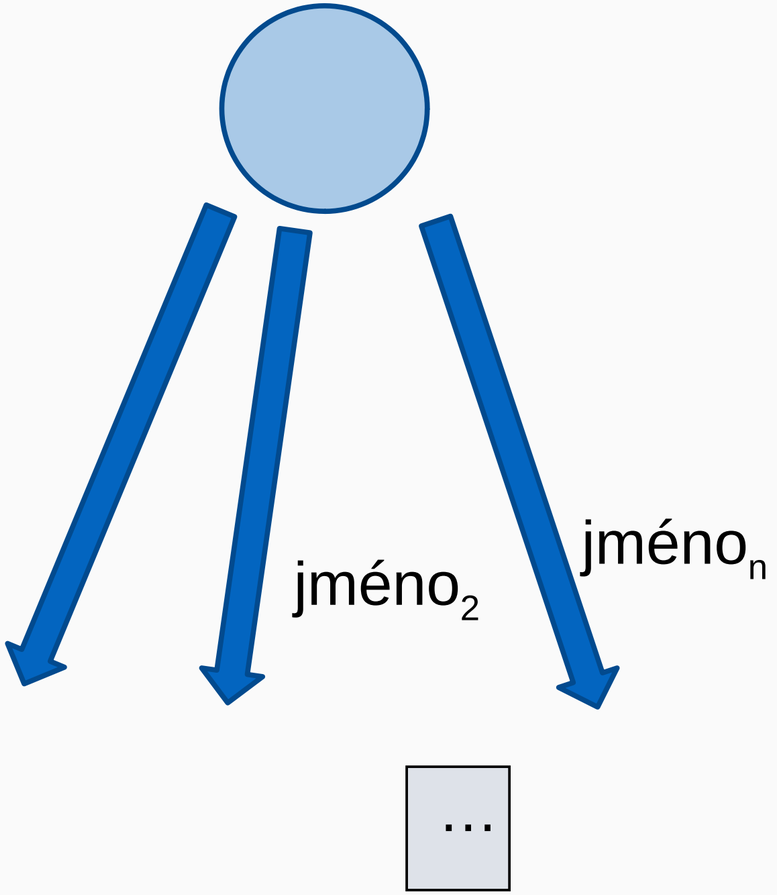
```
structure FyzOsoba
   properties
      UplneJmeno:   string
      Jmeno:        string
      Prijmeni:     string
      DatumNaroz:   date
   end structure
```

Definujeme *metadata*.

Hodnota struktury

jména vlastností	**hodnoty vlastností**
UplneJmeno:	"Prof. Ing. Jan Novák, CSc."
Jmeno:	"Jan"
Prijmeni:	"Novák"
DatumNaroz:	24.5.1954

jméno vlastnosti$_1$



Kolekce

- *Kolekce* (synonyma jsou *řetězec*, *posloupnost*, *seznam*, *soubor*) je, na rozdíl od struktur, tvořena
  - ***Předem neomezeným počtem** hodnot **stejných** datových typů.*

Kolekce

- Množina obsahuje obvykle každý prvek pouze jednou. Pokud je povoleno, aby daný prvek byl v množině vícekrát, mluvíme o *multimnožině*
- Tradiční seznam je *uspořádanou multimnožinou*
- Obecně lze vytvářet kolekce s prvky libovolných datových typů.
- Časté omezení je vytvářet *pouze kolekce s prvky datového typu struktura*

Operace nad množinou

- Vkládání prvku do kolekce (*add*),
- Získání prvku z kolekce (*item*),
- Určení počtu prvků kolekce (*count*) a
- Rušení prvku kolekce (*remove*)

případně

- Provádění operací nad všemi prvky (*forall*)

Vlastnosti kolekce

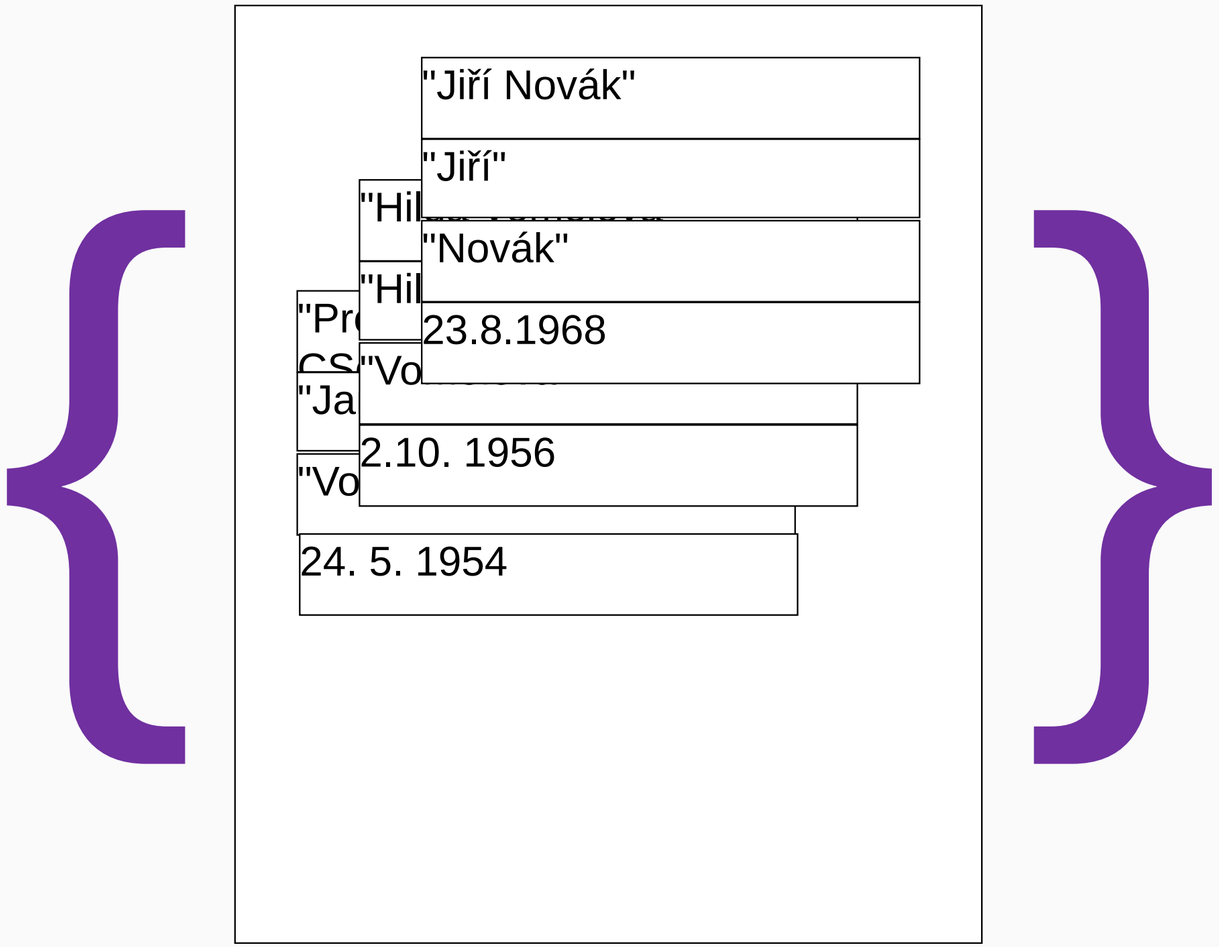
- *Kurzor (**iterator**)*, což je ukazovátko do kolekce, kterým lze posunovat oběma směry a nastavovat je do různých pozic v kolekci podle různých kriterií.
- Protože v průběhu práce s kurzorem se může kolekce měnit co do obsahu i počtu prvků, dělíme kurzory na *stabilní*, které na tuto skutečnost neberou zřetel a *nestabilní*, které reflektují změny
- Nad kolekcí může existovat jedno nebo více definovaných *uspořádání* jejich prvků podle různých klíčů.

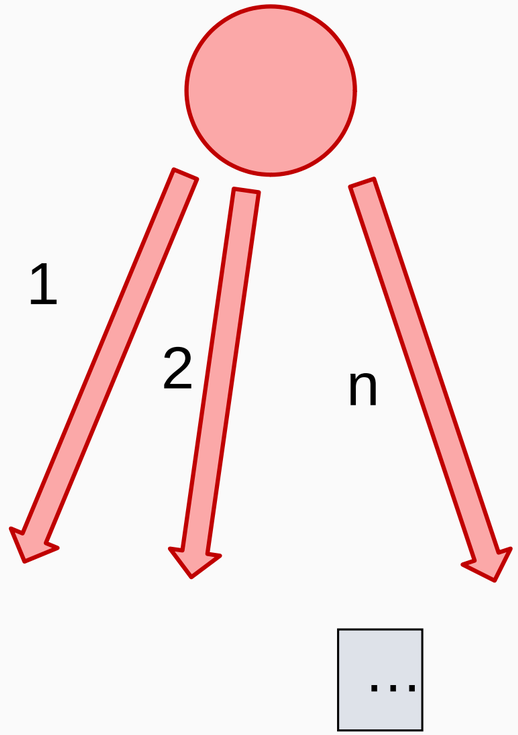
Schéma kolekce

Příklad datového typu kolekce struktur

```
collection FyzickeOsoby of
    structure FyzOsoba
       properties
        UplneJmeno: string
        Jmeno:      string
        Prijmeni:   string
        DatumNaroz: date
    end structure
```

Hodnota kolekce





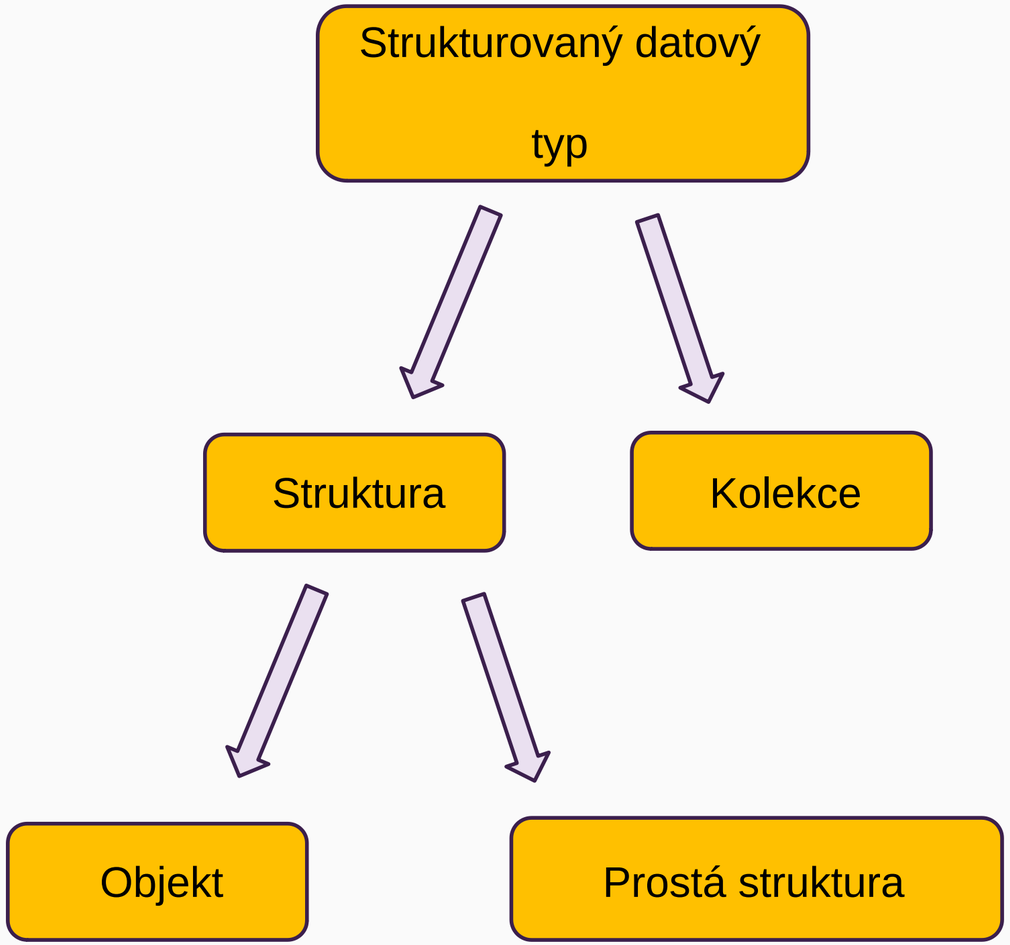
Agregáty

- Vlastnostmi kolekce jsou nejčastěji *agregáty (agregované hodnoty)*, což jsou hodnoty statisticky popisující prvky *kolekce* nejčastěji *číselných hodnot*.
  - *počet prvků,*
  - *maximum*,
  - *minimum*,
  - *součet hodnot*,
  - *průměr* atd.

Objekt a prostá struktura

- *Objekt je struktura s identifikací.*
- Každému objektu v systému přiřazena *jednoznačná identifikace* nazývaná *OID* (***object identification***).
- Objekt je tedy *struktura, jejíž systémovou a obvykle první vlastností je OID*. Hodnotu OID generuje databázový systém při vzniku objektu a po celou dobu činnosti ji nemění.
- Tím, že má objekt OID, je *identifikovatelný* a tudíž i *odkazovatelný*. Má to za následek, že může figurovat jako *člen* ve *vztazích*. To struktura bez identifikace nemůže. Takovou strukturu bez OID budeme nadále nazývat *prostou strukturou*.

Strukturované datové typy



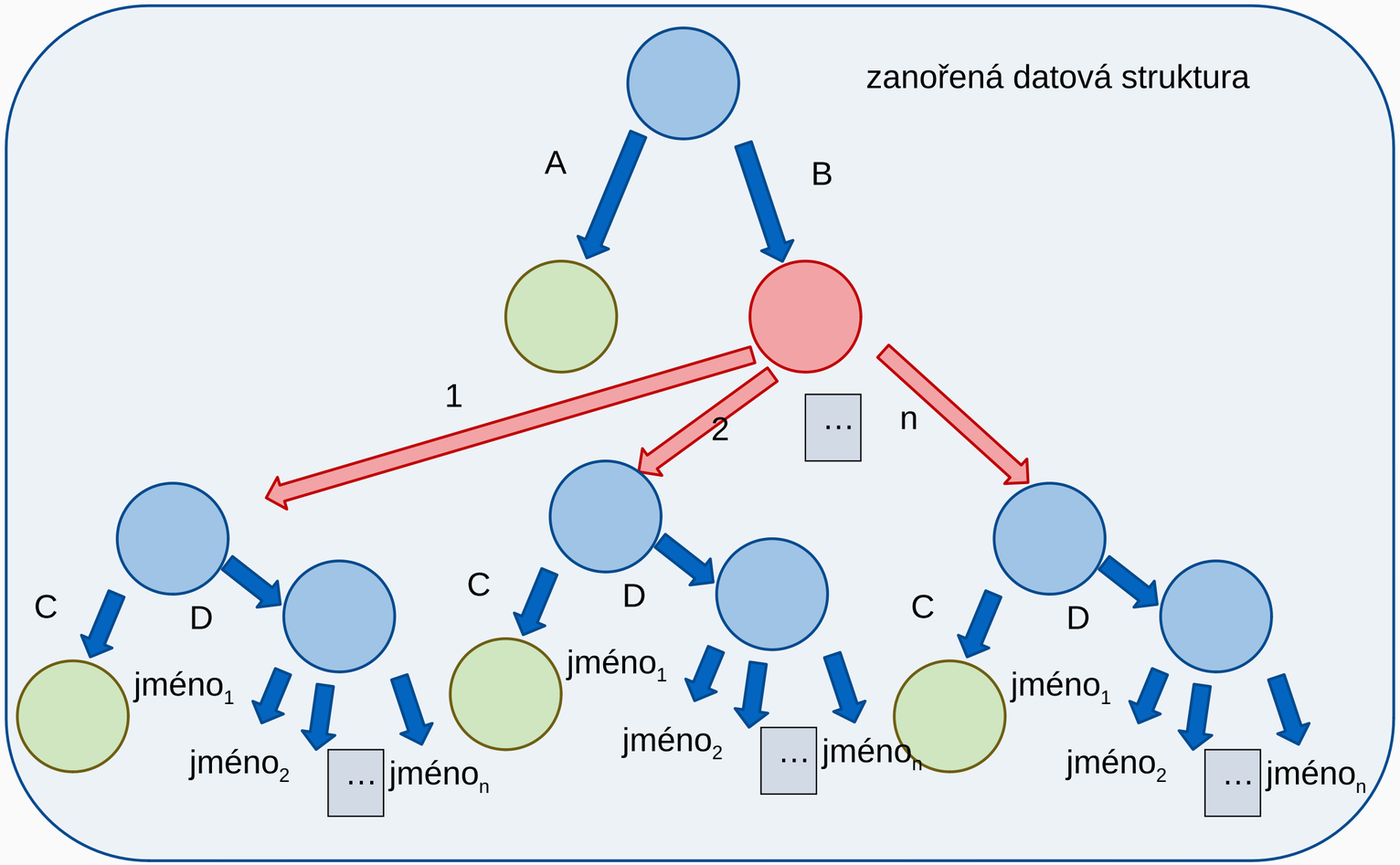
Zanořené kolekce a struktury

- Obecně lze struktury a kolekce libovolně vzájemné vnořovat

```
structure ZANORENA
   properties
      A: integer
      B: collection of structure
            properties
               C: integer
               D: structure
               …
                  end structure

         end structure
end structure
```

Graf hodnoty zanořených typů



Datové modelování

Databázové modely

- Modely, které je schopen interpretovat systém pro řízení databázového systému SŘBD
- Jinak též zvané **produkční modely**
- V jejich definičním jazyku musejí být zapsána **metadata** pro všechny datové struktury uložené v databázi
- Prozatím budeme uvažovat jako produkční **relační a objektový datový model**.

Databázové modely

- Jednoduché (NoSQL)
  - Key-value (MUMPS, Redis, …)
  - Dokumentové (MongoDB, CouchDB, …)
  - Sloupcové (Apache HBase, …)
- **Relační datový model**
  - Mnoho implementací
- **Objektový datový model**
  - Objektově-relační mapování (ORM)
- Grafové
  - Grafové databáze (Neo4J, OrientDB, …)
  - Sémantická úložiště (sémantický web, RDF)

Konceptuální modely

- Slouží pro komunikaci mezi návrháři, případně se zákazníky
- Jsou formálně přesné a **převoditelné** na produkční modely
- Často jsou grafické pro větší přehlednost
- Nejběžnější konceptuální modely **diagram tříd (UML)**, **E-R diagram** a **CDL**.

Transformace mezi datovými modely

- Slouží nejčastěji pro transformaci konceptuálních modelů na produkční.
- Transformace je tím složitější, čím jsou modely více sémanticky odlišné.
- Nejčastěji se uvažuje **transformace E-R diagramu na relační datový model**.

Relační model dat

- Tabulka (= **relace**) v relačním modelu je *kolekcí struktur*, přičemž datové typy vlastností jsou *jednoduché* (tedy především *ne odkazy/vztahy*)
- Srovnej: *Podmnožina kartézského součinu*

```
collection of
    structure
        properties
            jméno vlastnosti1: jednoduchý datový typ1
            jméno vlastnosti2: jednoduchý datový typ2
            …
            jméno vlastnostin: jednoduchý datový typn
    end structure
```

Vztahy

- Umožňují odkazovat z jedné (strukturované) hodnoty (vlastníka) jinou (člen)
- Musí existovat datový typ *jednoznačné identifikující (odkazující) strukturovanou hodnotu* (např. OID)
- Vztah je definován prvkem vlastníka typu odkaz (reference) a členem, který je hodnotou odkazu identifikován.

Vztahy

- Relační model dat vztahy přímo neobsahuje
  - Vytváří se až v okamžiku dotazování (JOIN apod.)
  - (Neplést s referenční integritou!)
- Objektový model
  - Vztahy lze tvořit pomocí OID

Objektový model dat

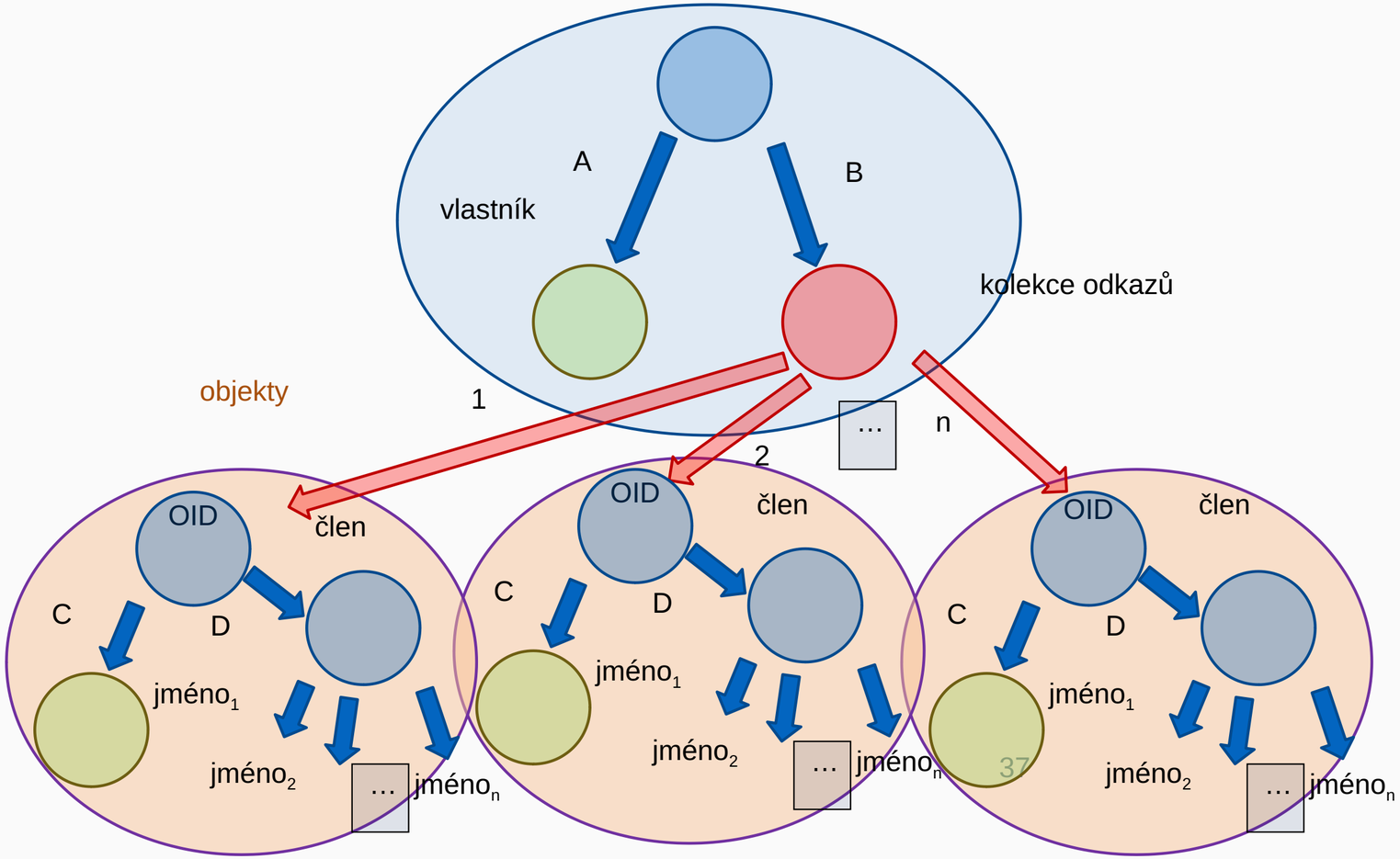
- Základní typy + datový typ OID
- Objekt je vždy strukturou na nejvyšší úrovni
- Dva druhy neomezeně zanořených struktur
  - Kolekce (někdy omezení pouze na kolekce prostých struktur a OID)
  - Prosté struktury (ostatní)
- Další vlastnosti – dědičnost apod.
- **Odpadá nutnost transformace objektového modelu na schéma relační databáze**

Odkazované struktury (objekty)

```
structure VLASTNIK
   properties
      A: integer
      B: CLEN
end structure

object CLEN
   properties
      C: integer
      D: structure
      …
end object
```

Graf hodnoty odkazovaných typů



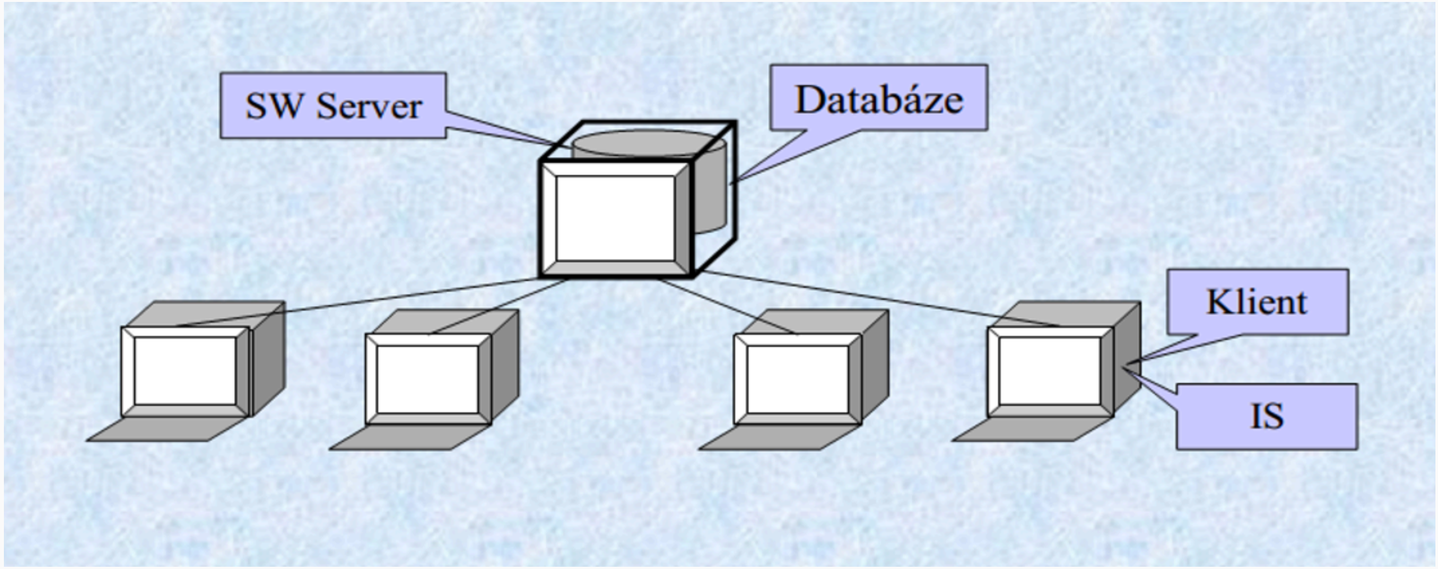
Shrnutí

- Strukturovaná data modelujeme jako **kolekce** a **struktury**
- Metadata – popisují data, se kterými systém pracuje
- Konceptuální modely
  - Vyjádření metadat pro účely modelování
  - E-R diagram, Class diagram, CDL
- Produkční (databázové) modely
  - Definice metadat pro konkrétní databázi
  - Relační model, objektový model
  - Alternativní modely – dokumentové, grafové, …

Architektury informačních systémů

Architektura klient-server (dvouvrstvá)

- Užity dva druhy oddělených výpočetních systémů **klient** a **server**.
- **Tloušťka** klienta odpovídá jeho "**inteligenci**"



Třívrstvá architektura

- *(three-tier architecture)*
- **Prezentační vrstva** – **vizualizuje** informace pro uživatele, většinou formou grafického uživatelského rozhraní, může kontrolovat zadávané vstupy, neobsahuje však zpracování dat
- **Aplikační vrstva** – jádro aplikace, logika a funkce, výpočty a zpracování dat
- **Datová vrstva** – nejčastěji databáze. Může zde být ale také (síťový) souborový systém, webová služba nebo jiná aplikace.

Terminologická odbočka

- **Tier** – fyzická vrstva – jednotka nasazení (deployment)
  - Fyzické členění systému – klient, aplikační server, DB server
  - Tomu odpovídá volba technologií pro realizaci jednotlivých částí
- **Layer** – logická vrstva – jednotka organizace kódu
  - Obvykle řešena v rámci aplikační vrstvy
  - *Data layer* – část řešící komunikaci s databází
  - *Business layer* – část implementující logiku aplikace
  - *Presentation layer* – komunikace s klientem

Schéma třívrstvé architektury

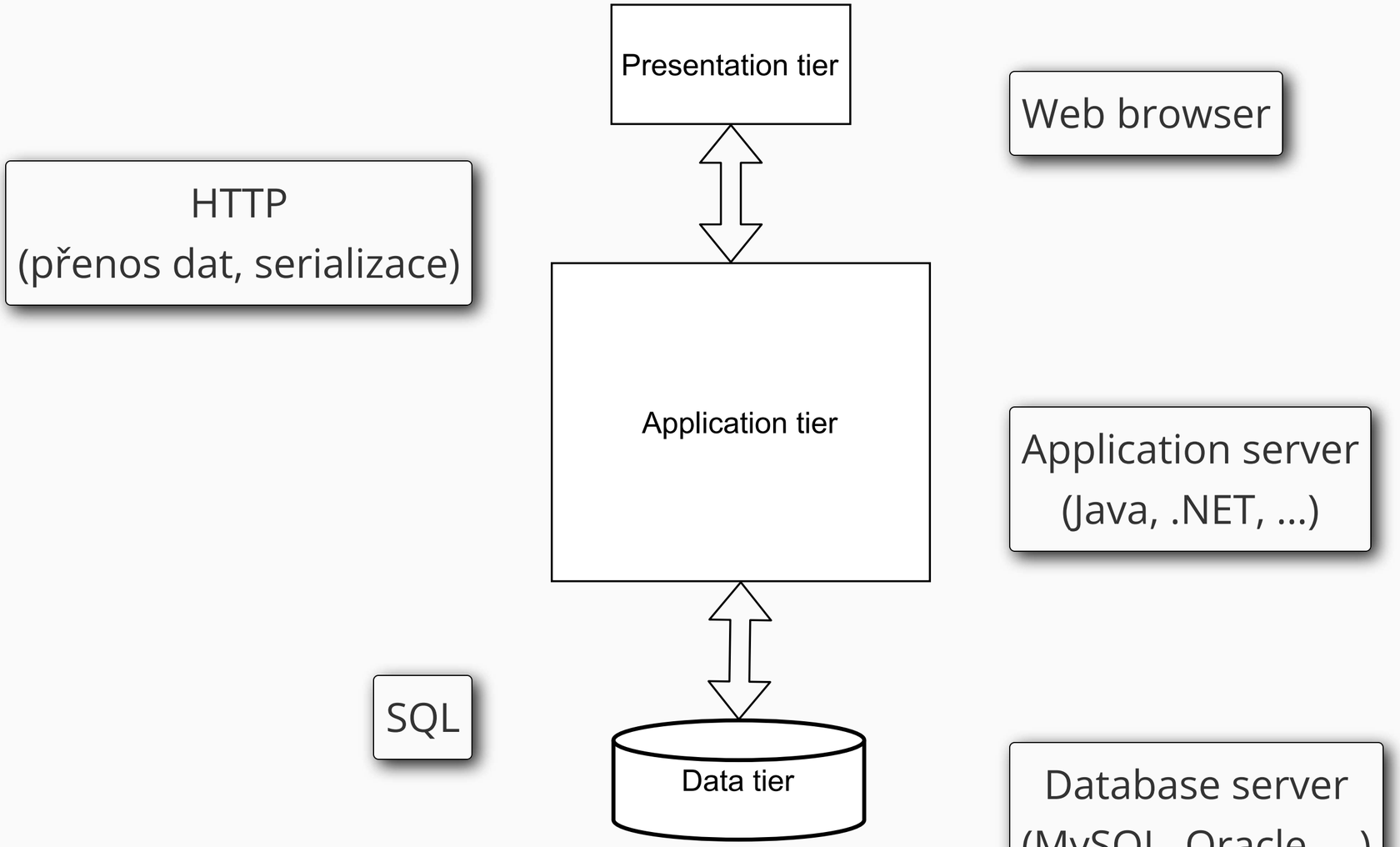
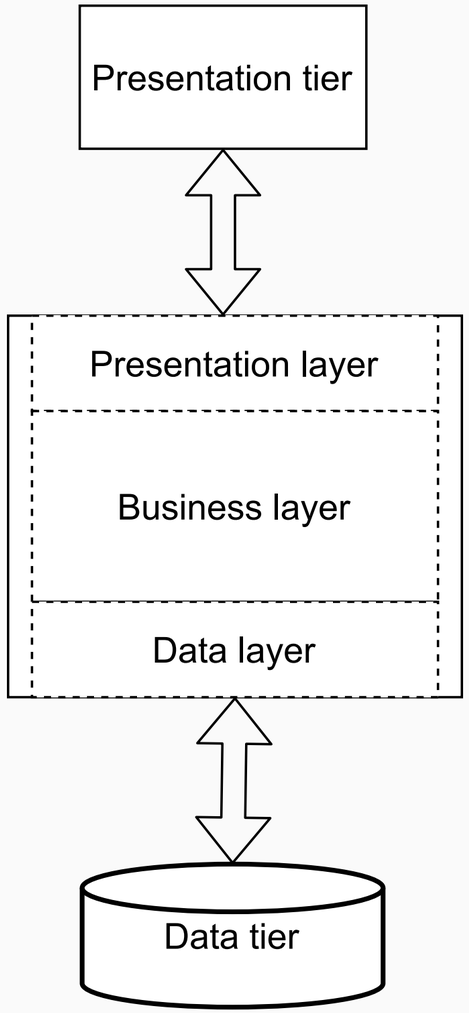


Schéma třívrstvé architektury (II)



Tenčí nebo tlustší klient v prohlížeči

Java, .NET, PHP …
Různá rámcová řešení (framework)

Datový model (objektový, relační, …)

Aplikační vrstva – Jakarta EE

Jakarta EE umožňuje (kromě jiného) implementovat *monolitický* IS s *třívrstvou architekturou:*

1. Databázová vrstva
   - JPA – definice entit, persistence (*PersistenceManager*)
   - Alternativně: Relační databáze (JDBC), NoSQL (MongoDB), …
2. Logická (business) vrstva
   - Enterprise Java Beans (EJB) nebo CDI beans
   - Dependency injection – volné propojení
3. Prezentační vrstva
   - Webové rozhraní (JSF) nebo API (REST, JAX-RS)

Další platformy – přehled

- Java
  - Existuje mnoho možností kromě „standardní“ J EE
- .NET (Core / Framework)
  - Mnoho řešení na všech vrstvách
- PHP
  - Různé frameworky, důraz na webovou vrstvu
- JavaScript
  - Node.js + frameworky, důraz na web a mikroslužby
- Python, Ruby, … - podobné principy

Distribuované architektury

- Monolitický systém (typické pro třívrstvou architekturu)
  - Vyvíjí se a nasazuje jako jeden celek
  - + snáze zvládnutelný vývoj, testování
  - - obtížnější a pomalejší nasazování nových verzí
- Distribuované architektury
  - Service-oriented architecture (SOA)
  - Microservices (mikroslužby)

Mikroslužby (Microservices)

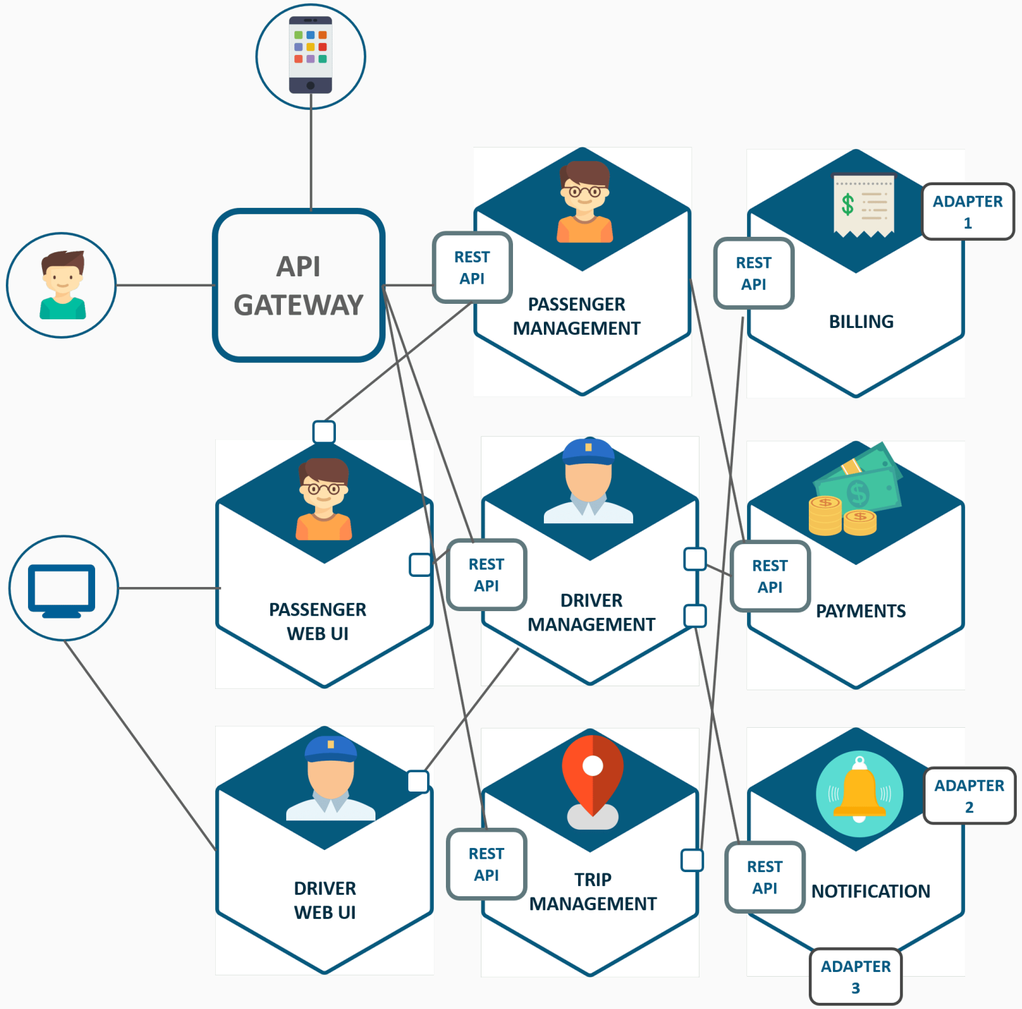
Monolitická architektura

- 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

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, …

Mikroslužby (příklad: Uber)



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)

Implementace mikroslužeb

- V čemkoliv – spojovacím bodem je pouze API
- Node.js (+ express + MongoDB)
  - Populární rychlé řešení
- Java
  - Spring Boot
  - Ultralehké frameworky
    - Např. Spark - https://github.com/perwendel/spark
  - **Microprofile**

A to je vše!

Dotazy?