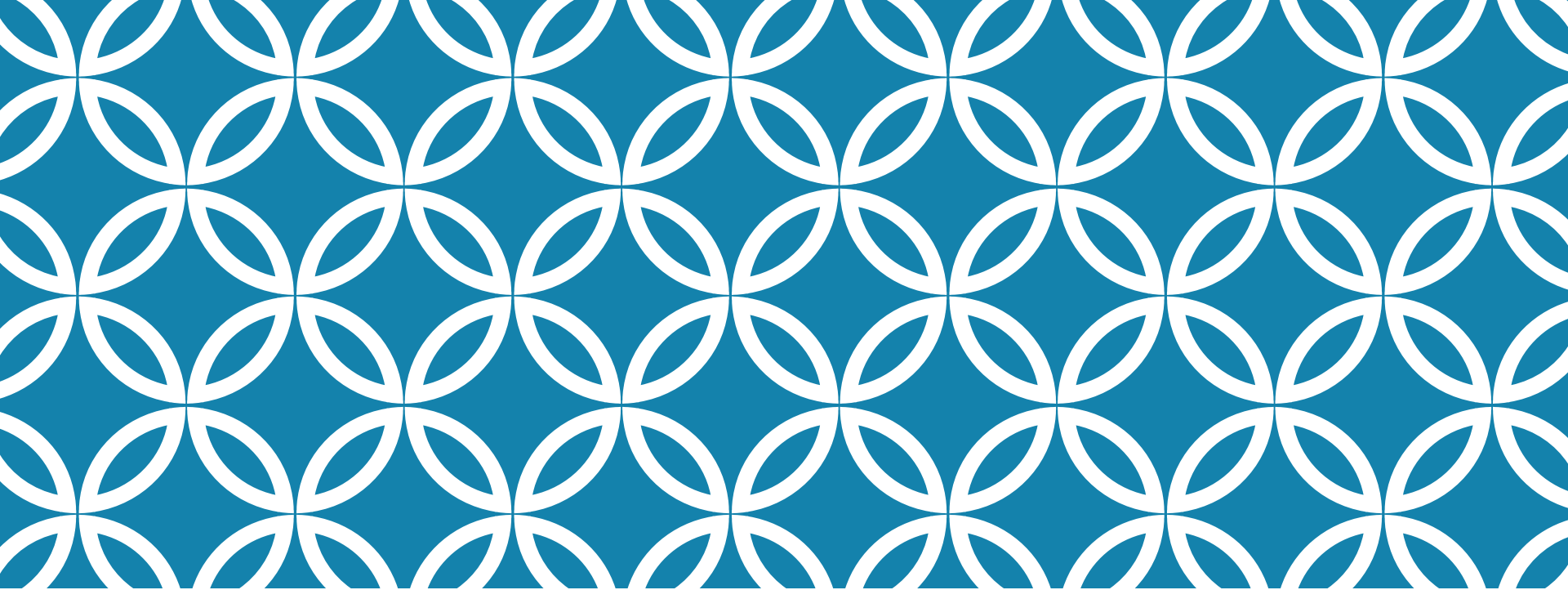


ISKB12 INFORMAČNÍ SYSTÉMY
- DATABÁZE

Ing. Mgr. Pavel Synek

10.3.2023

DNEŠNÍ AGENDA

1. Databáze a jejich historie
2. Modely
3. Architektura IS
4. Notace modelovacích jazyků
5. Datové modelování

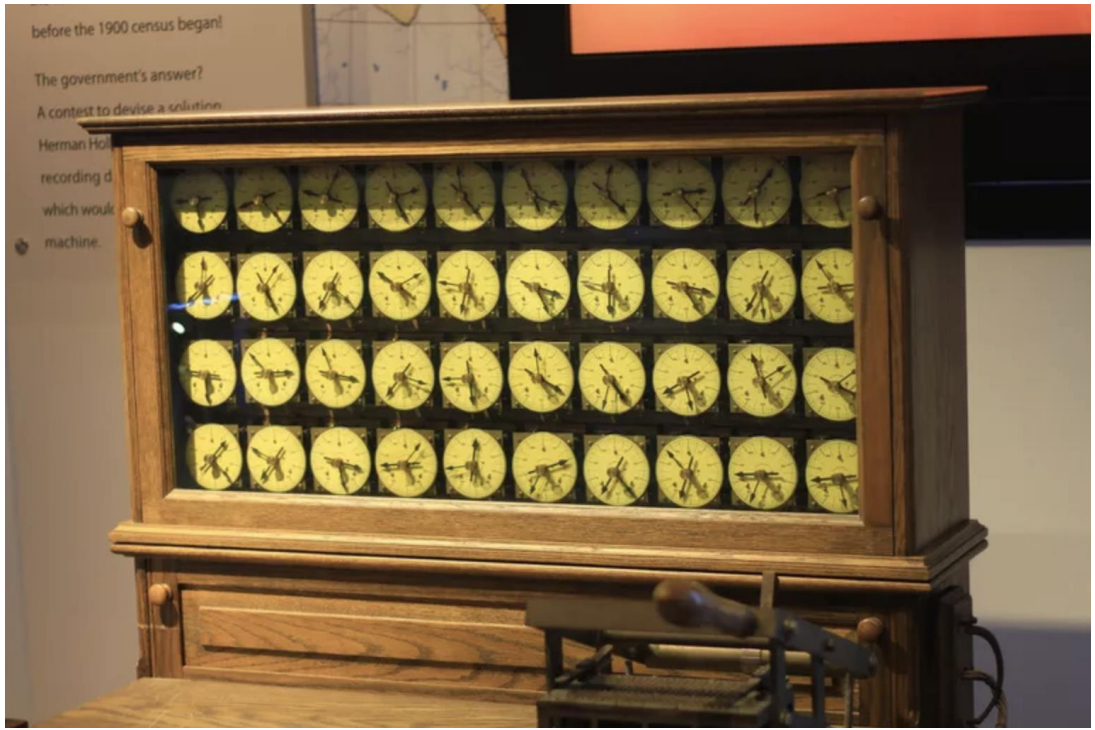
DATABÁZE

- Nárůst množství dat a vzájemných vazeb - potřeba je hierarchizovat a spravovat
- Databáze = soubor dat organizovaných dle předem stanoveného řádu uložených k dalšímu zpracování
- Databáze je zjednodušeným modelem reálného světa a popisuje děje v něm probíhající
- Rozsáhlejší systémy pro zpracování dat se pak nazývají informační systémy, které můžeme definovat jako „systémy pro sběr, uchovávání, vyhledávání a zpracovávání dat za účelem poskytování informací“.

DATABÁZE

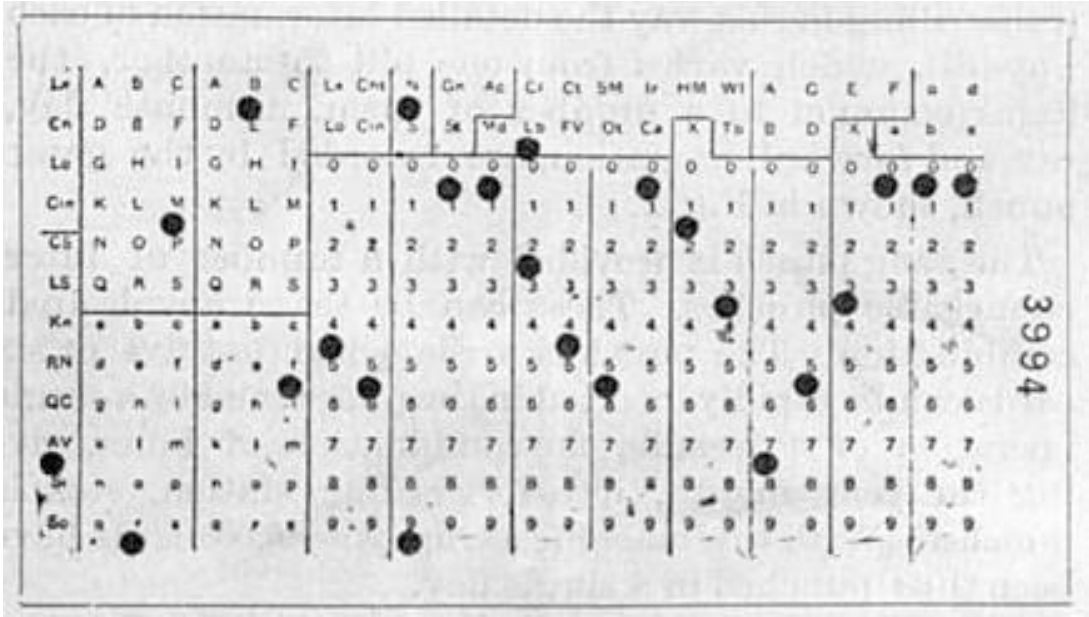
Knihovní katalog

DATABÁZE



Herman Hollerith`s Tabulating Machine - 1890

DATABÁZE



Děrný štítek

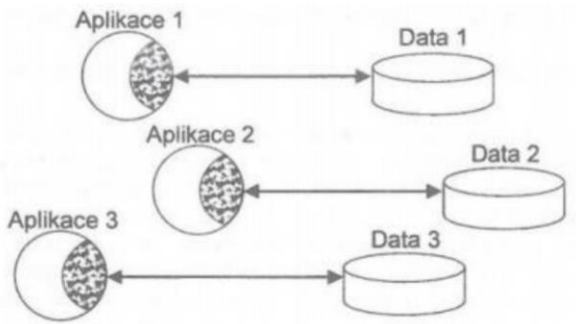
DATABÁZE

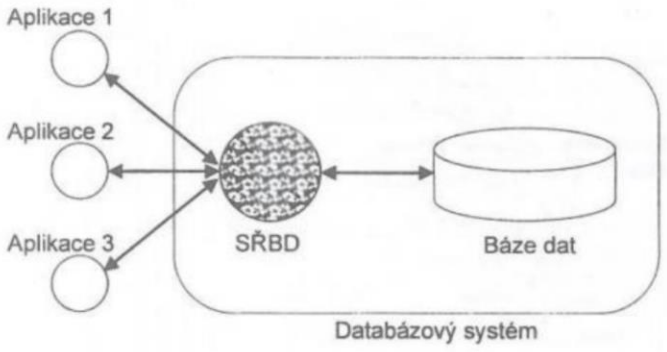
- 1888 – Oberlin Smith publikoval článek "Some Possible Forms of the Phonograph" v časopisu Electrical World, kde představuje tezi o technologickém řešení záznamu informací na elektromagnetický nosič
- 1959 - první jazyk pro databázové systémy - COBOL (common business-oriented language)
- 1965 - První Systémy řízení báze dat provozované na sálových počítačích
- 1969 – Edgar. F. Codd : Návrh relačního datového modelu
- 1974 - vzniká databázový jazyk Sequel (později SQL)
- 1980 - První SQL datáze Oracle, spol. Relational Software, Inc. (1980),
- následovala DB2 od IBM (a řada další Progres, Informxix, SyBase,...).

HISTORIE DATABÁZOVÉHO ZPRACOVÁNÍ

- V historii hromadného zpracování dat lze sledovat dva přístupy k datům:
  - Souborový přístup – data uložena do jednoho nebo více datových souborů uložených na paměťovém médiu, součástí souboru dat je i jejich popis. Aplikace je přímo navázána na strukturalizaci dat. Toto řešení přineslo celou řadu problémů (redundance, izolovanost dat, nekonzistence dat, nezajištění integrity dat, nemožnost přístupu více uživatelů, atd.)
  - Databázový přístup – odstraňuje nevýhody souborového přístupu. Báze dat je řízena Systémem řízení báze dat.

SOUBOROVÝ VS. DATABÁZOVÝ PŘÍSTUP





VÝHODY DATABÁZOVÉHO PŘÍSTUPU

- Zamezení redundance
- Zajištění konzistence dat
- Integrita dat
- Sdílení dat
- Ochrana dat před zneužitím
- Nezávislost dat na aplikaci
- Přístup k datům výhradně prostřednictvím databázových programů
- Možnost využití grafických přehledů
- Možnost ukládání velkých objemů dat
- Využití jazyka SQL jako standardu pro práci s daty

SYSTÉM ŘÍZENÍ BÁZE DAT (SŘBD)

- SŘBD je programový systém, který umožňuje definování struktury, ukládání, výběr a ochranu dat, zabezpečuje databázi a komunikaci mezi uživatelem a systémem.
- Zjednodušeně jde tedy o softwarový prostředek, který řídí sdílený přístup k bázi dat a poskytuje mechanismy určené k zajištění bezpečnosti a integrity dat.
- SŘBD tvoří souhrn procedur a datových struktur, které zajišťují nezávislost databázových aplikací na detailech vytváření, výběru, uchování, modifikaci a zabezpečení ochrany databází na fyzických paměťových strukturách počítače.
- Příklady SŘBD: Microsoft SQL server, My SQL, Oracle, Informix, SyBase, Microsoft Access a další.

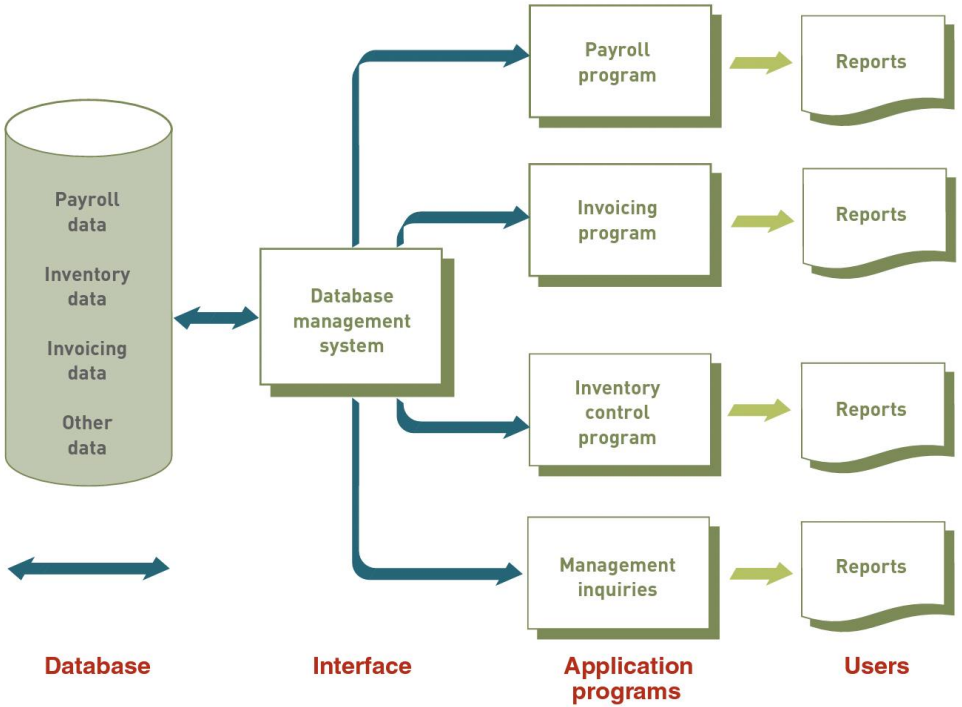
SYSTÉM ŘÍZENÍ BÁZE DAT (SŘBD)

- SŘBD umožňují:
  - vytvoření báze dat
  - vkládání dat
  - aktualizace dat
  - rušení dat
  - výběr z báze dat
  - tvorbu vstupních a výstupních formulářů, výstupních sestav a vytváření aplikací.

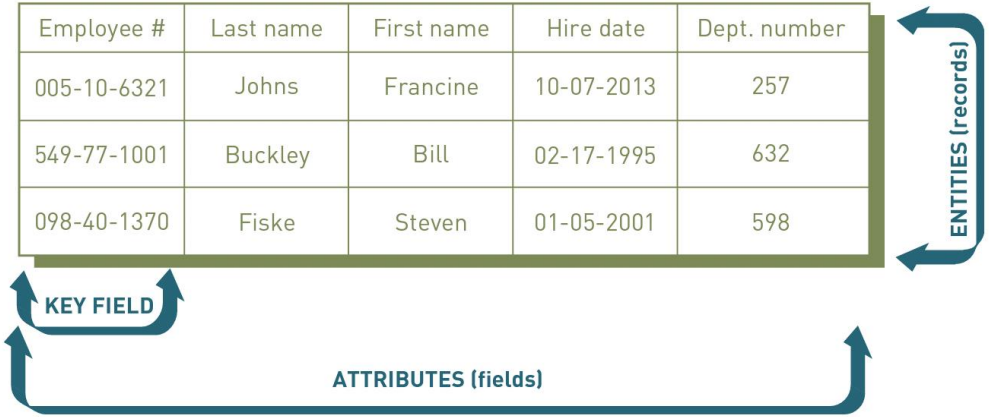
SYSTÉM ŘÍZENÍ BÁZE DAT (SŘBD)

- Systém řízení báze dat zahrnuje:
  - **Jazyk pro definici dat** – DDL (data definition language) – prostředky pro popis dat, sloužící k vytvoření všech definic uživatelských dat potřebných v aplikaci, včetně určení omezujících podmínek.
  - **Jazyk pro manipulaci s daty** – DML (data manipulation language) – prostředky pro popis algoritmu, které se používají k aktualizaci dat (přidávání, změny a rušení dat) a k výběru dat z databáze na základě kladených požadavků.

DATABÁZOVÝ SYSTÉM = BÁZE DAT + SYSTÉM ŘÍZENÍ BÁZE DAT



DATABÁZE


Employee #
Last name
First name
Hire date
Dept. number
005-10-6321
Johns
Francine
10-07-2013
257
549-77-1001
Buckley
Bill
02-17-1995
632
098-40-1370
Fiske
Steven
01-05-2001
598
ENTITIES (records)
KEY FIELD
ATTRIBUTES (fields)

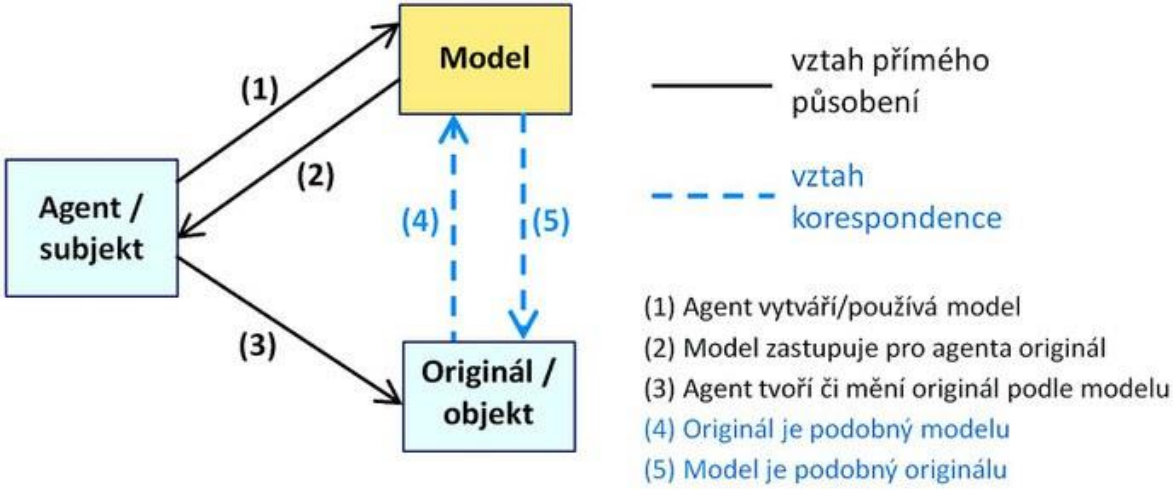
BÁZE DAT (BD) A.K.A. DATABÁZE

- BD je množina souborů a jejich popisu, které jsou vzájemně v určitém logickém vztahu.
- Jde o komplikovanou centrálně zpracovávanou strukturu dat.
- Pro databáze je vytvořena jediná interní organizace dat, společná pro všechny oblasti a způsoby využití.

CO JE MODEL?

- Model budeme chápat jako umělý výtvor, tj. artefakt, záměrně za určitým cílem vytvářený agentem ve funkci subjektu. Tímto výtvorem může být jakákoli myšlenková nebo materiální konstrukce. Od ostatních umělých výtvorů se model odlišuje svým specifickým účelem, jímž je reprezentace originálu ve funkci objektu (předmětu).

CO JE MODEL?



Znázornění modelu
https://knihovnarevue.nkp.cz/archiv/2018-2/recenzovane-prispevky/pojem-modelu-a-pojmovy-model-v-informacni-vede

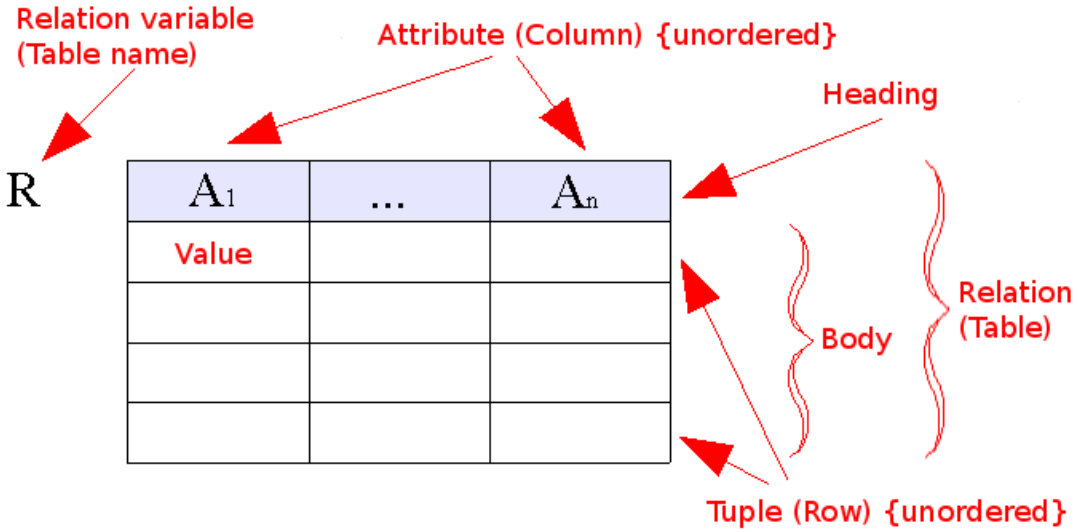
JAKÉ JSOU DRUHY MODELŮ?

- Mentální model
- Konceptuální model
- Matematický model
- Procesní model
- Fyzický model
- Architektonický model
- Datový model
- ….

DATABÁZOVÉ MODELY

- Databázový model je typ datového modelu, který určuje logickou strukturu databáze a zásadně určuje, jakým způsobem lze data ukládat, organizovat a manipulovat.
- Podle typu modelovaných vztahů mezi záznamy se v databázi na technologické úrovni rozlišují následující modely:
  - relační
  - hierarchický
  - síťový
  - objektový model

RELAČNÍ DATOVÝ MODEL



RELAČNÍ DATOVÝ MODEL

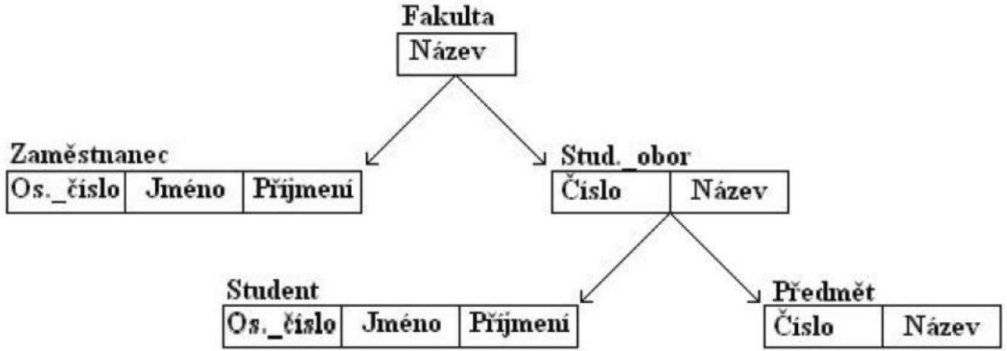
- Relace = tabulka se sloupci a řádky vyjadřující jejich vzájemný vztah/vazby
- Atribut = pojmenovaný sloupec s nějakou "relací"
- Doména = rozsah hodnot přípustných v daném atributu
- N-tice (Tuple) = řádek tabulky

- Výhody - přirozená reprezentace dat, jednoduché zachycení struktury a vazeb

HIERARCHICKÝ MODEL

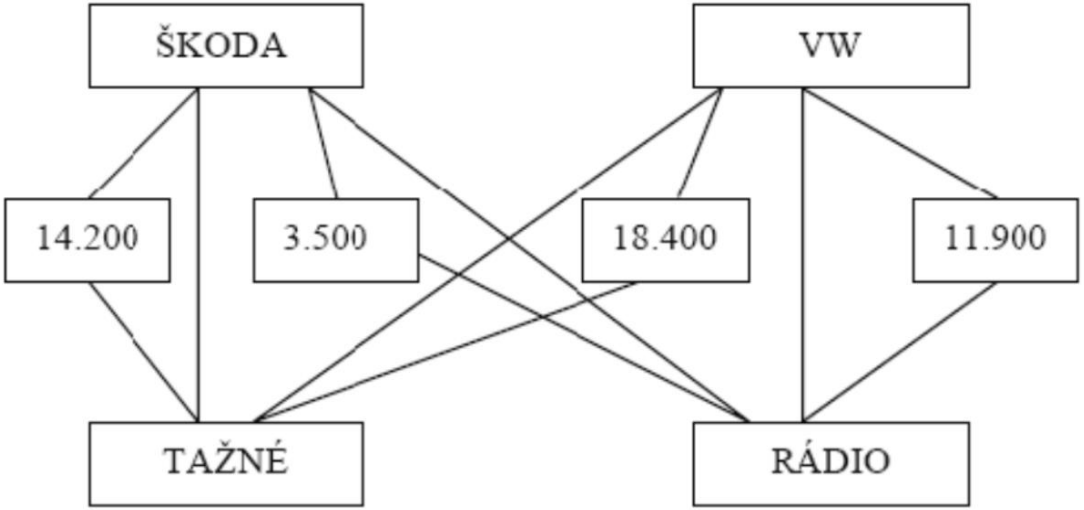
- Logické uspořádání dat v tomto modelu je založeno na stromové struktuře, která umožňuje vyjádřit jednosměrné vztahy typu 1 - více ve směru shora dolů.
- Každý záznam představuje uzel ve stromové struktuře a vzájemný vztah mezi záznamy je typu Rodič/Potomek.
- Použití hierarchického modelu je vhodné tam, kde i realita má hierarchickou strukturu, např. organizační či skladové systémy.
- Nalezení dat v této struktuře vyžaduje navigaci přes záznamy směrem dolů (Potomek), nahoru (Rodič), do stran (Sourozenec).

HIERARCHICKÝ MODEL



SÍŤOVÝ MODEL

- V tomto modelu jsou data logicky i fyzicky uspořádána jako uzly rovinného grafu, v němž může být každý záznam spojený s libovolným počtem dalších záznamů.
- Tím může obsahovat odkazy na jiné entity v bázi dat, které s ní logicky souvisí – je tedy možné využít mnohonásobného rodičovství.
- Jde v podstatě o zobecněný hierarchický model, který je doplněn o mnohonásobné vztahy. Jednotlivé entity tak mohou vytvořit síť s velmi rozmanitou strukturou.

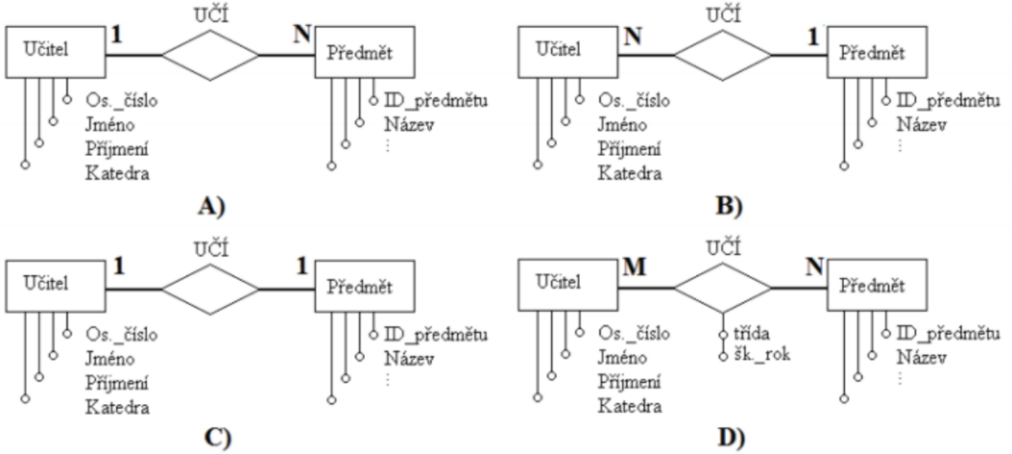


SÍŤOVÝ MODEL

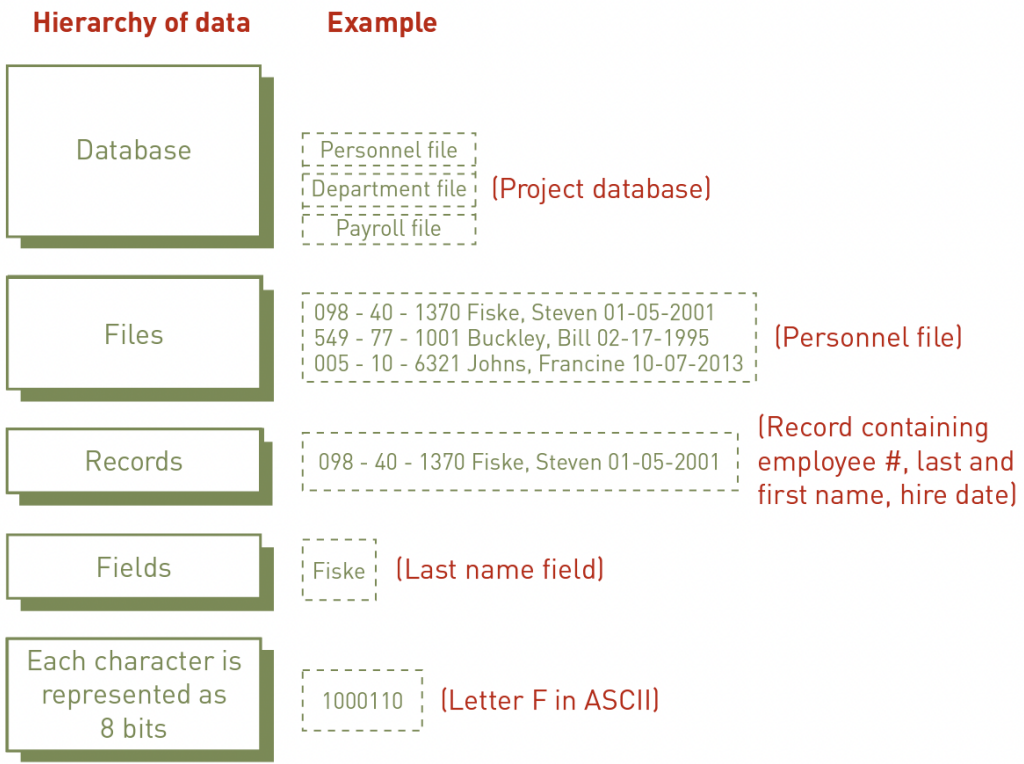
RELAČNÍ MODEL

- Relační databázový model byl do nedávna nejvíce rozšířený a používaný databázový model.
- Relační databázový model má jednoduchou strukturu, kdy data jsou organizována v tabulkách, které se skládají z řádků a sloupců.
- Všechny databázové operace jsou pak prováděny na těchto tabulkách.

RELAČNÍ MODEL



HIERARCHIE DAT

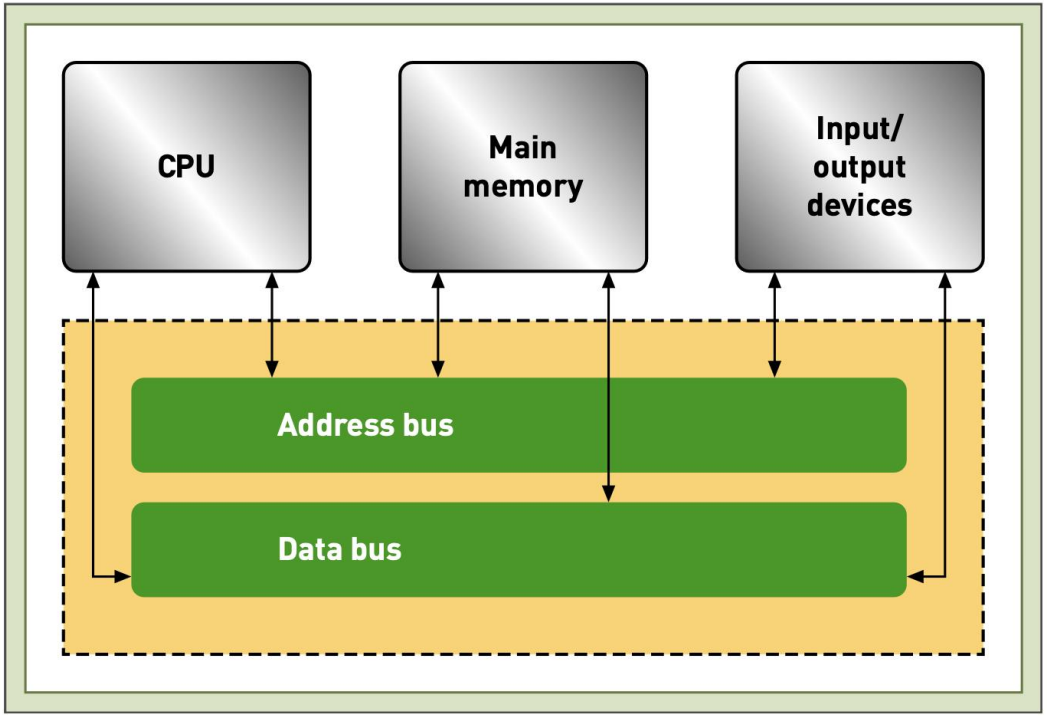


OBJEKTOVÝ MODEL

- Pod obecné označení „objektové databáze“ patří dva odlišné datové modely:
  - Objektově relační datový model, který je doplněním relačního datového modelu o možnost práce s některými datovými strukturami známé z oblasti objektově orientovaných programovacích jazyků. Objektově relační datový model však ve svých principech zůstává původním relačním datovým modelem.
  - Objektově orientovaný datový model je nový datový model, který není odvozen od relačního datového modelu. Do jisté míry jde o přetvoření a vylepšení původního síťového datového modelu, který je doplněn o možnost práce s objekty známými z objektového programování.

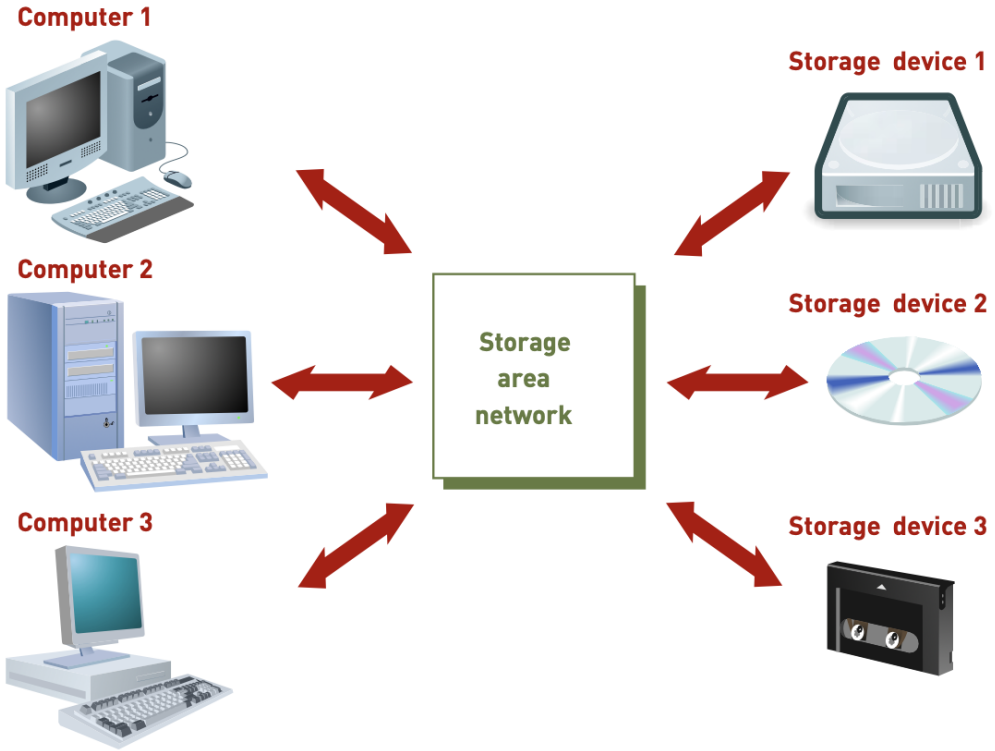
ARCHITEKTURA DATABÁZOVÝCH SYSTÉMŮ

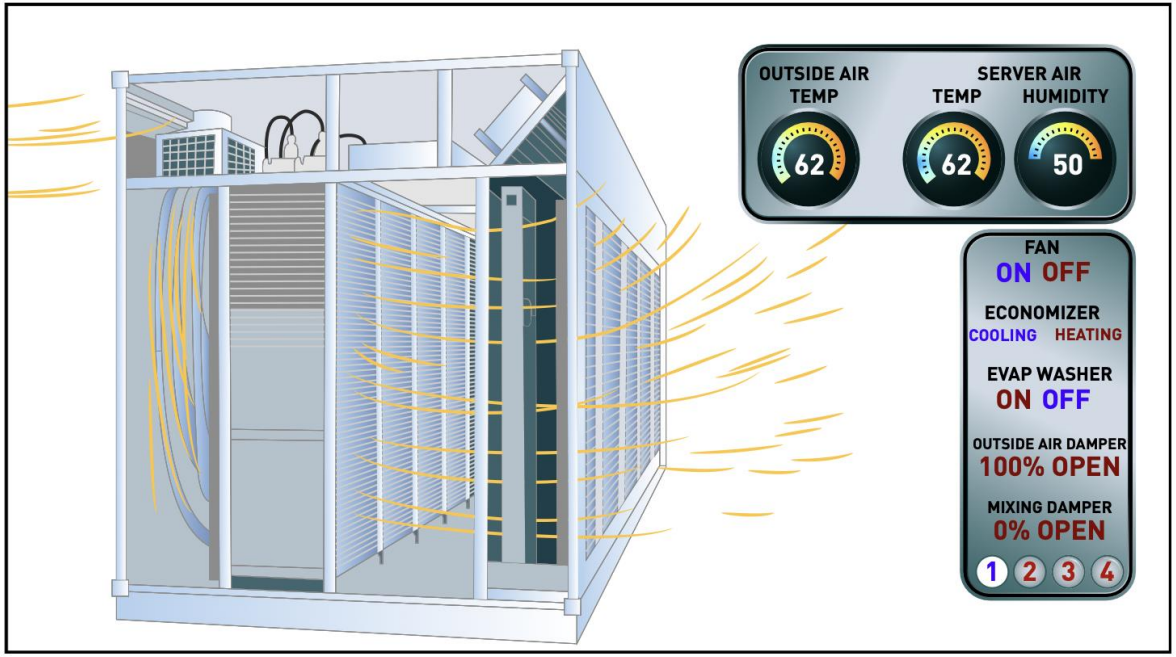
ZJEDNODUŠENÁ ANATOMIE POČÍTAČE



STORAGE AREA NETWORK

Dedikovaná datová síť, která umožňuje připojení externích datových nosičů se serverem.

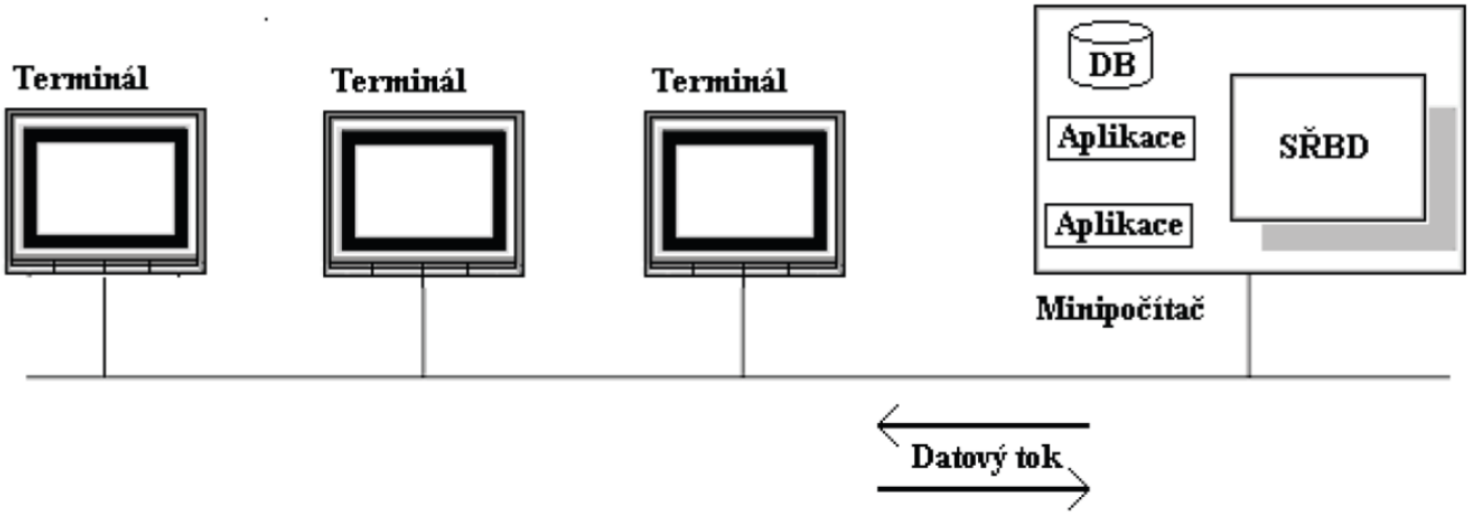




MODULÁRNÍ DATOVÉ CENTRUM

ARCHITEKTURA DATABÁZOVÝCH SYSTÉMŮ

- Rozlišujeme následující typy databázových architektur:
  - Jednovrstvá (centralizovaná) architektura
  - Dvouvrstvá architektura (File-Server nebo Klient-Server)
  - Vícevrstvá architektura



JEDNOVRSTVÁ (CENTRALIZOVANÁ) ARCHITKTURA

JEDNOVRSTVÁ (CENTRALIZOVANÁ) ARCHITEKTURA

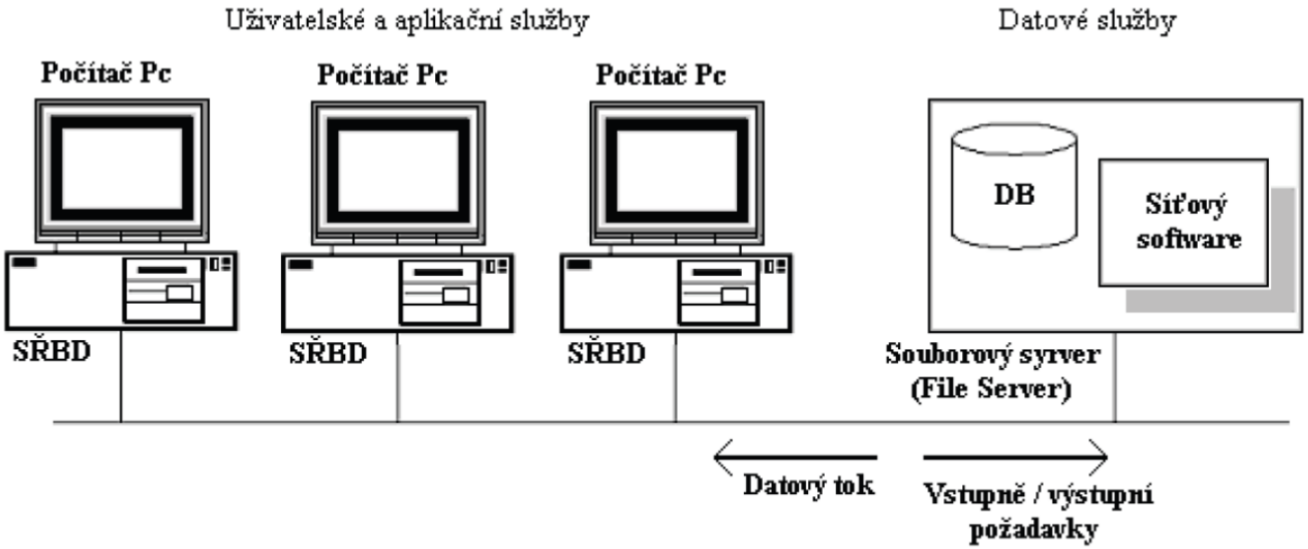
- Zastaralý model architktury db systémů s použitím centrálního počítače
- BD dat i SŘBD jsou společně na centrálním počítači a komunikaci s uživatelem zprostředkovává pouze terminál umístěný na pracovišti uživatele.
- Vstupní data a požadavky se přenášejí z terminálu po síti do centrálního počítače, kde dochází ke zpracování a následnému zpětnému odeslání na terminál uživatele.

JEDNOVRSTVÁ (CENTRALIZOVANÁ) ARCHITEKTURA

- Vhodné pro aplikace (databáze), ve kterých se informace nesdílejí mezi několika uživateli.
- Řešení má výhodu v rychlosti přístupu a zpracování dat, protože data jsou uložena lokálně a nevyžadují samostatný databázový server a počítačovou síť.

DVOUVRSTVÁ ARCHITEKTURA

- Tvorbou dvouvrstvých aplikací poskytneme víceuživatelskou podporu a získáme možnost používat velké vzdálené databáze, které mohou ukládat značně velké množství informací.
- Dvouvrstvou architekturu lze rozdělit do dvou skupin:
  - Architektura File-Server - výkon spojený s aplikačními službami je umístěn na straně klienta
  - Architektura Klient-Server - výkon spojený s aplikačními službami je umístěn na straně serveru



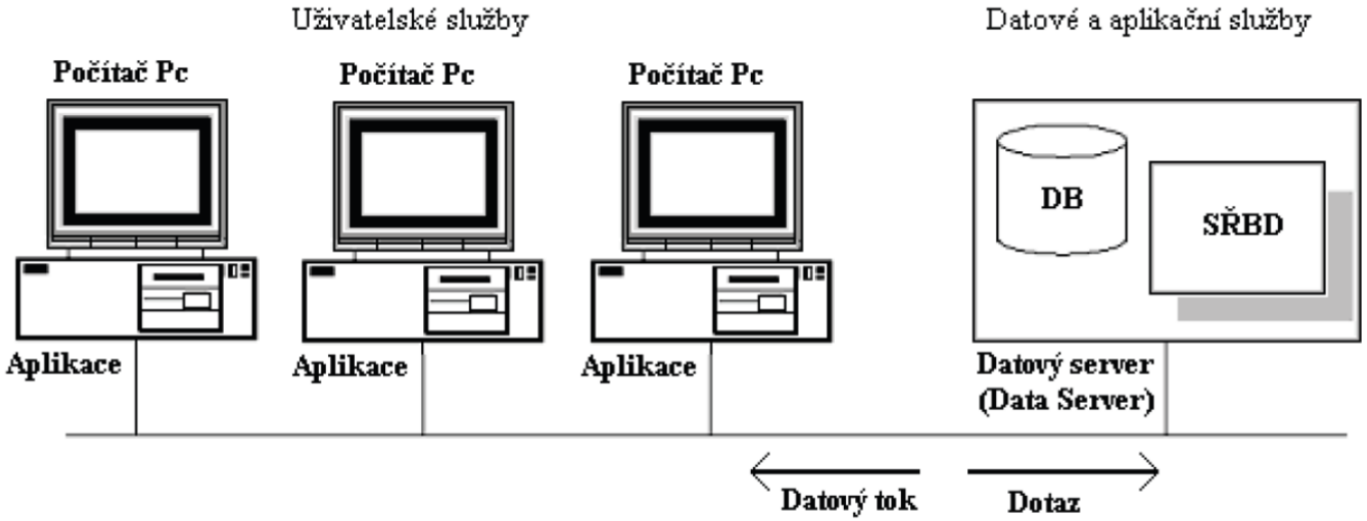
ARCHITEKTURA FILE-SERVER

ARCHITEKTURA FILE-SERVER

- Databáze (DB) s daty je umístěna na počítači, který pracuje jako File-Server a prostřednictvím sítě a jednotlivých systémů řízení báze dat (SŘBD), umístěných na počítačích uživatelů, poskytuje jednotlivá data a umožňuje sdílení.
- V tomto případě může k datům přistupovat více aplikací v podobě SŘBD najednou a proto je nutné zajistit ochranu používaných záznamů.

ARCHITEKTURA FILE-SERVER

- Nevýhodou je, že veškeré aplikační a uživatelské služby se zpracovávají u klienta, kterého v tomto případě nazýváme Tlustý klient.
- Velkou slabinou této architektury jsou rovněž velké nároky na přenosové kapacity, kdy mezi klientem a serverem musí probíhat velký počet datových přenosů.



ARCHITEKTURA KLIENT-SERVER

ARCHITEKTURA KLIENT-SERVER

- U této architektury běží na počítačích aplikace, které předávají dotazy a požadavky na datový server, který je zpracovává a výsledky posílá zpět na počítač uživatele.
- Server je v tuto chvíli nejvíce zatíženým počítačem, protože na něm běží SŘBD, který vše zpracovává.

ARCHITEKTURA KLIENT-SERVER

- K uživateli jsou přesunuty pouze uživatelské služby a získává pouze požadované informace. V tomto případě hovoříme, že uživatel je tzv. Tenký klient.
- Aplikační a datové služby probíhají na serveru. Tato architektura snižuje požadavky na množství dat pohybujících se v síti a tím vyhovuje i rozsáhlým aplikacím.
- Výhodou je tedy minimální zatížení sítě, vysoká pružnost aplikací a rozdělení zpracování záznamů.

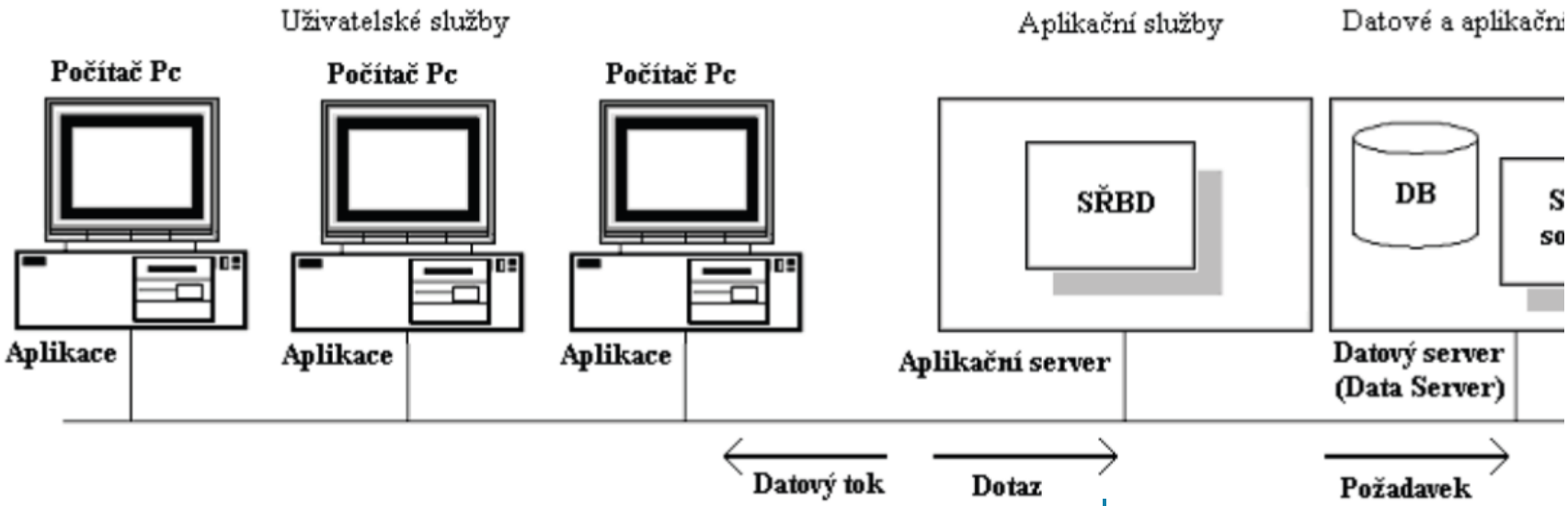
ROZDÍL MEZI FILE-SERVER A KLIENT-SERVER

Architektura Klient-Server redukuje přenos dat po síti, protože dotazy jsou prováděny přímo na databázovém serveru a na počítač jsou posílány pouze výsledky. Např. pokud je mezi 10 000 záznamy pouze 100 záznamů, které splňují podmínku dotazu, pak na personální počítač putuje pouze těchto 100 záznamů.

V případě architektury File-Server je však nutné poslat všech 10 000 záznamů na personální počítač (tedy celou tabulku – blok), tam se teprve provede dotaz a zpracuje nalezených 100 záznamů.

VÍCEVRSTVÁ ARCHITEKTURA

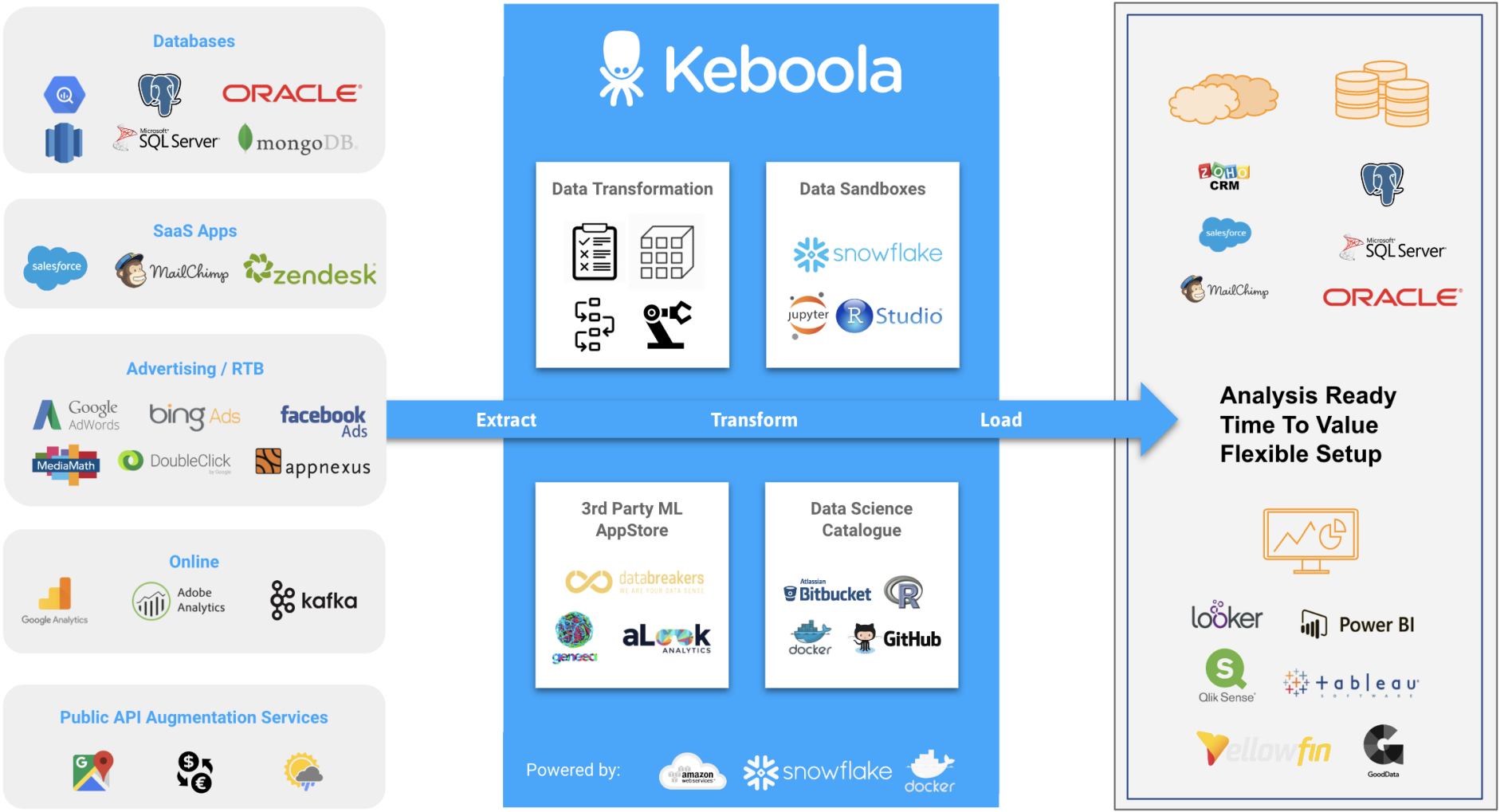
- Obsahují-li databázové informace komplikované vazby mezi několika tabulkami nebo zvyšuje-li se počet uživatelů (klientů), pak lze používat vícevrstvé aplikace.
- Vícevrstvé aplikace obsahují střední vrstvu, která centralizuje logiku ovládání databázových interakcí (umožňuje různým klientským aplikacím používat stejná data se zajištěním konzistentní datové logiky).

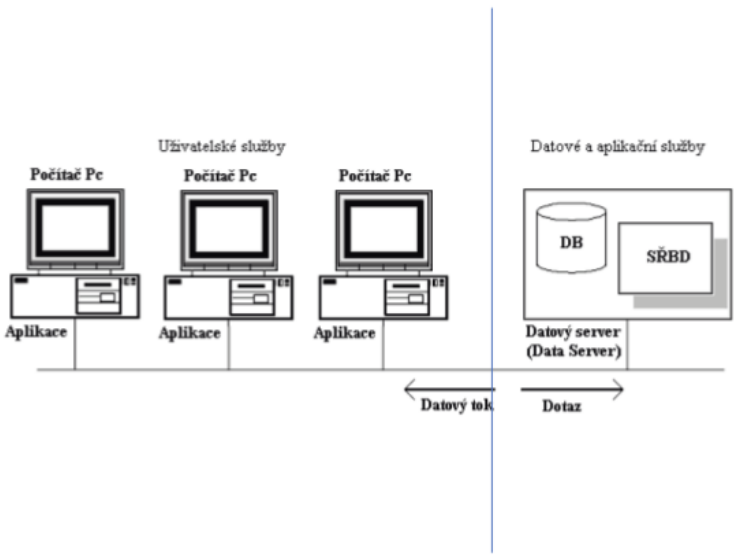


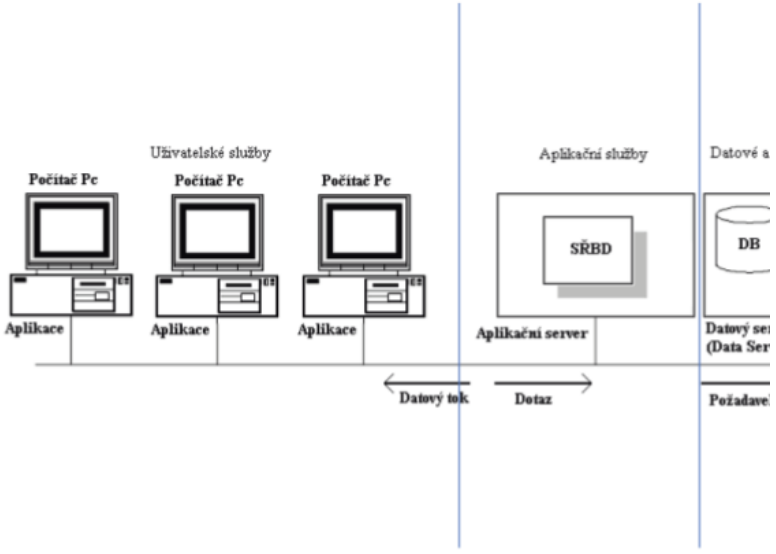
VÍCEVRSTVÁ ARCHITEKTURA

VÍCEVRSTVÁ ARCHITEKTURA

- U vícevrstvé architektury lze shledat určitou podobnost s již uvedeným modelem architektury Klient-Server, kdy je výkon spojený s aplikačními službami soustředěn na serveru.
- Klient, potažmo uživatel pracuje pouze s uživatelským rozhraním, přičemž datové a aplikační služby jsou od sebe odděleny do samostatných logických celků, které mohou být umístěny buď na stejném serveru, nebo na dvou různých serverech.
- Vícevrstvá architektura umožňuje získat vyšší úroveň stability, protože provozní zátěž může být rozložena na dva nebo více servery.


Databases
ORACLE
SQL Server
mongoDB
SaaS Apps
salesforce
MailChimp
zendesk
Advertising / RTB
Google AdWords
bing Ads
facebook Ads
MediaMath
DoubleClick
appnexus
Online
Google Analytics
Adobe Analytics
kafka
Public API Augmentation Services
Keboola
Data Transformation
Data Sandboxes
snowflake
jupyter
R Studio
3rd Party ML AppStore
databreakers
aLook ANALYTICS
Data Science Catalogue
Atlassian Bitbucket
docker
GitHub
Powered by:
amazon web services
snowflake
docker
Extract
Transform
Load
ZOHO CRM
salesforce
MailChimp
Microsoft SQL Server
ORACLE
Analysis Ready
Time To Value
Flexible Setup
looker
Power BI
Qlik Sense
tableau
yellowfin
GoodData

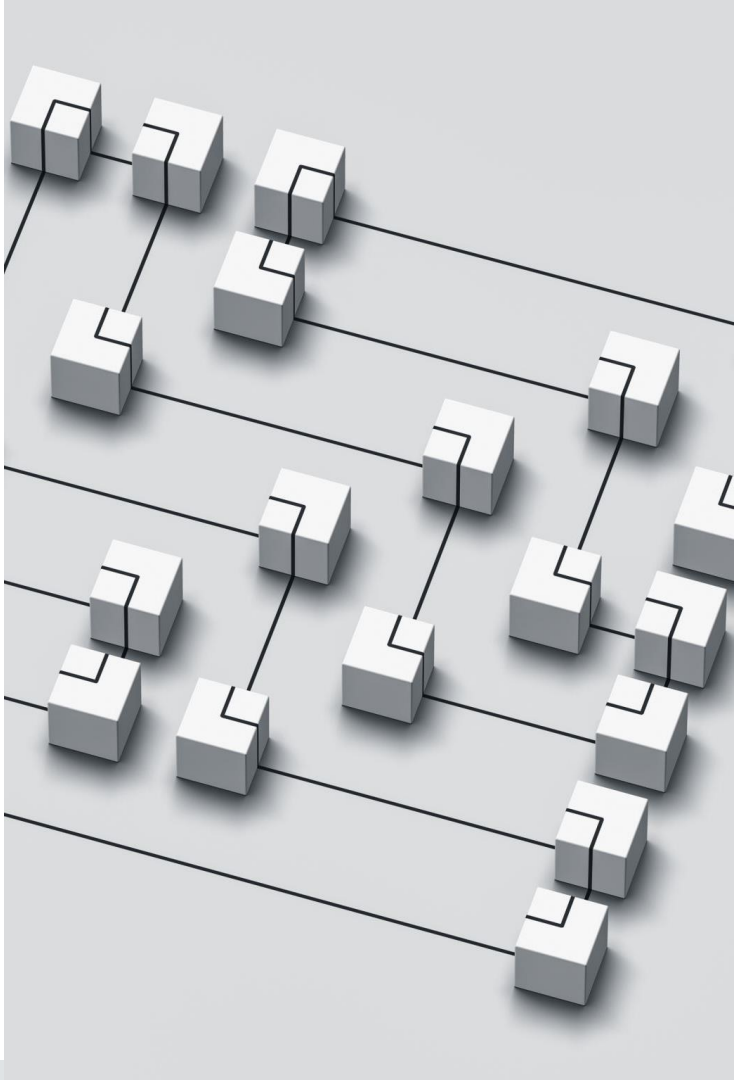




DVOUVRSTVÁ VS. TŘÍVRSTVÁ ARCHITEKTURA

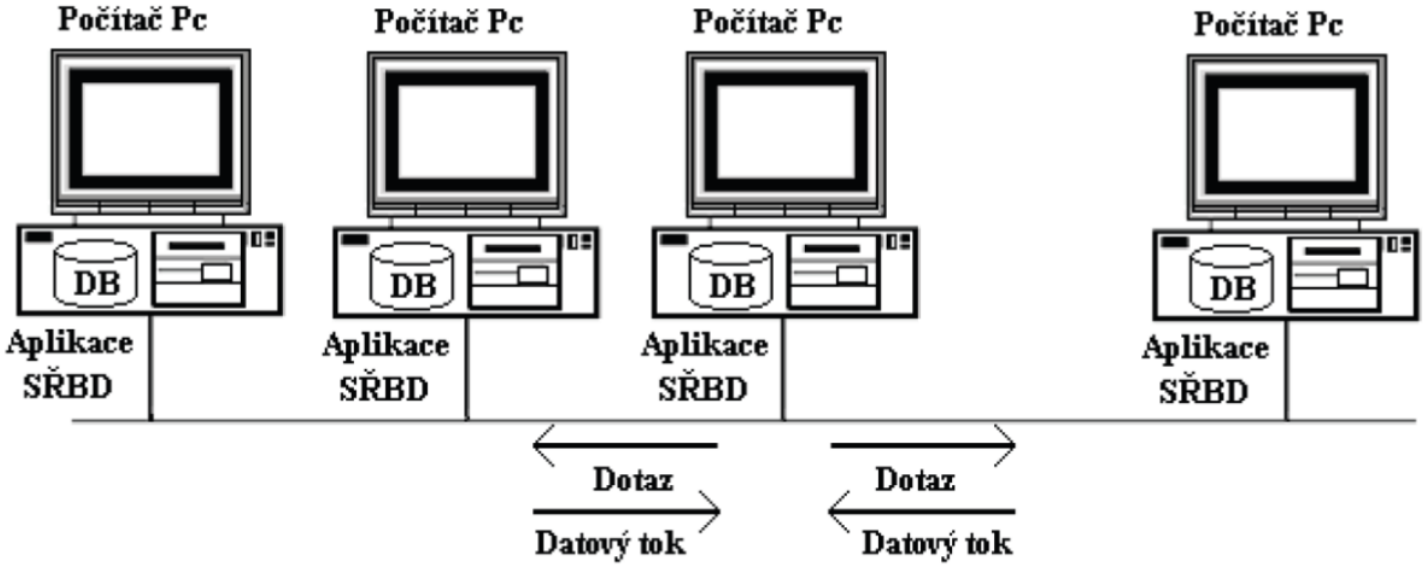
ARCHITEKTURA DISTRIBUOVANÝCH DB SYSTÉMŮ

- Distribuovaná databáze je kolekce dat, které logicky patří do jednoho systému, ale fyzicky jsou rozprostřena mezi jednotlivými místy (uzly) počítačové sítě.
- K rozvoji DDBS vedla řada faktorů:
  - distribuovaná povaha některých databázových aplikací
  - zvýšená spolehlivost a dostupnost
  - řízené sdílení dat
  - zvýšený výkon.

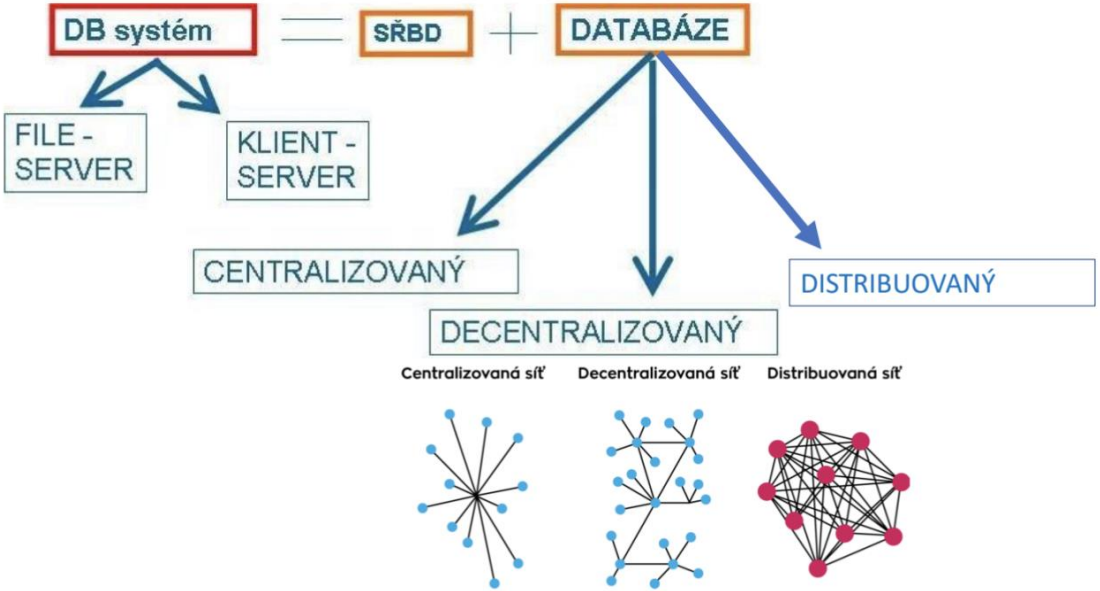


ARCHITEKTURA DISTRIBUOVANÝCH DB SYSTÉMŮ

- Při použití této architektury jsou data rozložena v několika počítačích, což znamená, že databáze je rozdělena do několika částí.
  - Navenek se však uživateli jeví jako jediná celistvá databáze.
  - Distribuované databázové systémy se vyznačují třemi základními vlastnostmi:
  - Transparentnost
  - Autonomnost
  - Nezávislost na typu sítě



ARCHITEKTURA DISTRIBUOVANÝCH SYSTÉMŮ



KONTEXT DBS = SŘBD + DB

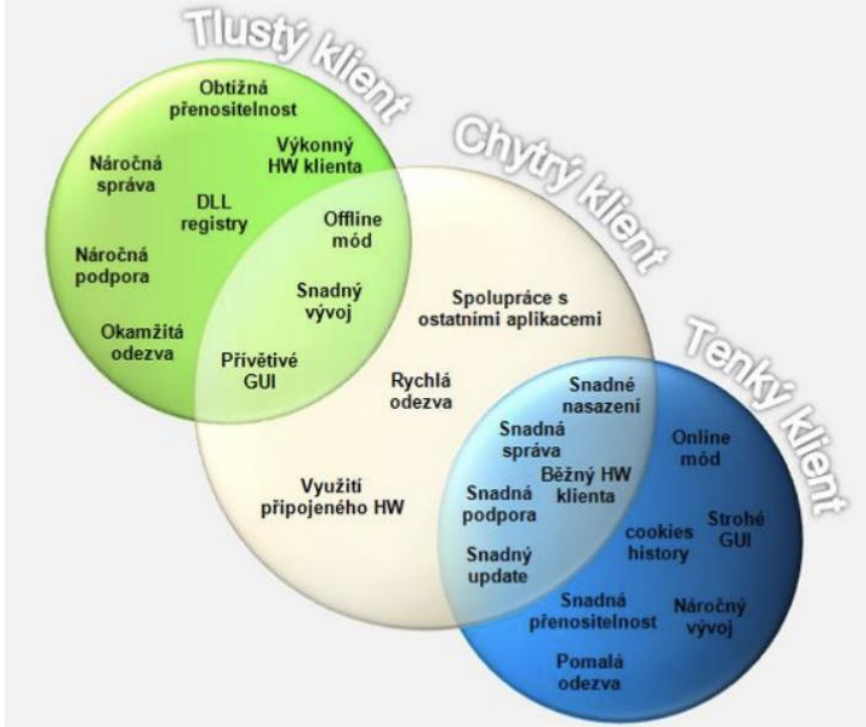
TYPY SW KLIENTŮ A JEJICH VÝHODY A NEVÝHODY

- **Tlustý klient** v sobě obvykle obsahuje jak presentační tak i aplikační vrstvu a připojuje se přímo k databázovému nebo jinému serveru. Další typickou vlastností tlustého klienta je, že si přes síť stahuje velký objem dat, která zpracuje a výsledek pak přenese zpět na server.
- **Tenkým klientem** je obvykle webový prohlížeč, který s presentační vrstvou komunikuje přes bez stavový HTTP protokol a stará se tak pouze o zobrazování dat. Na tenkém klientovi neprobíhá žádná rozhodovací logika, pokud nebereme v úvahu např. validaci dat, zadávaných do webového formuláře.

TYPY SW KLIENTŮ A JEJICH VÝHODY A NEVÝHODY

- Chytrý klient vhodně kombinuje výhody tenkého a tlustého klienta a potlačuje jejich nevýhody. Aby mohl pracovat off-line, obsahuje určitou logiku a drží data, která se v okamžiku navázání spojení synchronizují. Využívá místní systémové zdroje jako je např. paměť, procesor a diskový prostor, ale může komunikovat a využívat I připojených zařízení.

Vlastnost/klient	Tlustý klient (desktopová aplikace)		Chytrý klient		Tenký klient (webová aplikace)	
Nároky na HW konfiguraci klienta	-	vysoké, je potřeba výkonný HW	0	postačí běžný HW	+	nízké, není potřeba výkonný HW
Nároky na HW konfiguraci serveru	+	nízké, není potřeba výkonný HW	0	postačí běžný HW	-	vysoké, je potřeba výkonný HW
Objem přenesených dat	-	značný, neboť data se zpracovávají lokálně	0	malý, pořízená data se musí přenést na server	+	nepatrný, neboť data se zpracovávají na serveru
Nároky na vývoj aplikace	+	nízké, snadný a levný vývoj	0	střední, snadný, ale drahý vývoj	-	vysoké, náročný a drahý vývoj
Nároky na nasazení aplikace	-	časově náročný instalační proces setup.exe	0	nainstalovaný po zadání URL	+	k dispozici po připojení na dané URL
Nároky na update aplikace	-	nutno naplánovat způsob jak update proběhne	0	po připojení proběhne automatické update	+	zobrazena vždy aktuální verze
Nároky na správu aplikace	-	vysoké	0	střední/nízké	+	nízké/žádné
Nároky na podporu uživatelů aplikace	-	vysoké	0	střední/nízké	+	nízké/žádné
Přenositelnost aplikace	-	musí se instalovat různé verze pro různé platformy	0	požaduje runtime nebo pluginy do prohlížeče	+	běží v každém internetovém prohlížeči
Možnost práce uživatele offline	+	ano	0	ano – synchronizace	-	ne
Uživatelské rozhraní aplikace	+	přívětivé	0	přívětivé	-	strohé
Odezva aplikace	+	okamžitá	0	rychlá (JAVA, Flash, Silverlight, AJAX)	-	pomalá (reload stránky po každém kliknutí)
Stopy aplikace v systému	-	velký (vlastní program, DLL, registry)	0	střední (vlastní program a soubory v tempu)	+	malý (cookies, soubory v tempu, historie)
Způsob uložení dat	0	data jsou stahovaná na klienta	0	data se nachází dočasně na klientovi	0	data jsou uložena výhradně na serveru



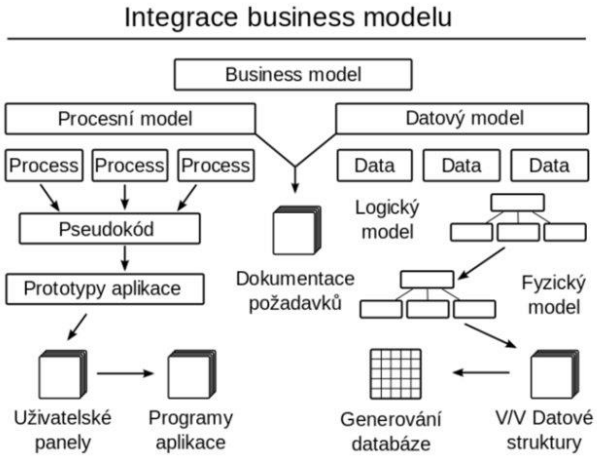
DATOVÉ MODELOVÁNÍ

DATOVÉ MODELOVÁNÍ

- Datové modelování je jednou z disciplín softwarového inženýrství
- Je to proces, při němž se definují a analyzují požadavky na strukturu dat, s nimiž pracuje informační systém.
- Datové modely slouží jako prostředek pro komunikaci mezi těmi, kteří definují požadavky na informační systém, a těmi, kdo tento systém vytvářejí.

DATOVÉ MODELOVÁNÍ

- Cílem datového modelování je zachytit a popsat tu část reality, o které chceme uchovávat informace.
- Cílem je převést reálné objekty na objekty datové (data)
- Datové modelování je součástí všech projektů vyžadujících analýzu dat

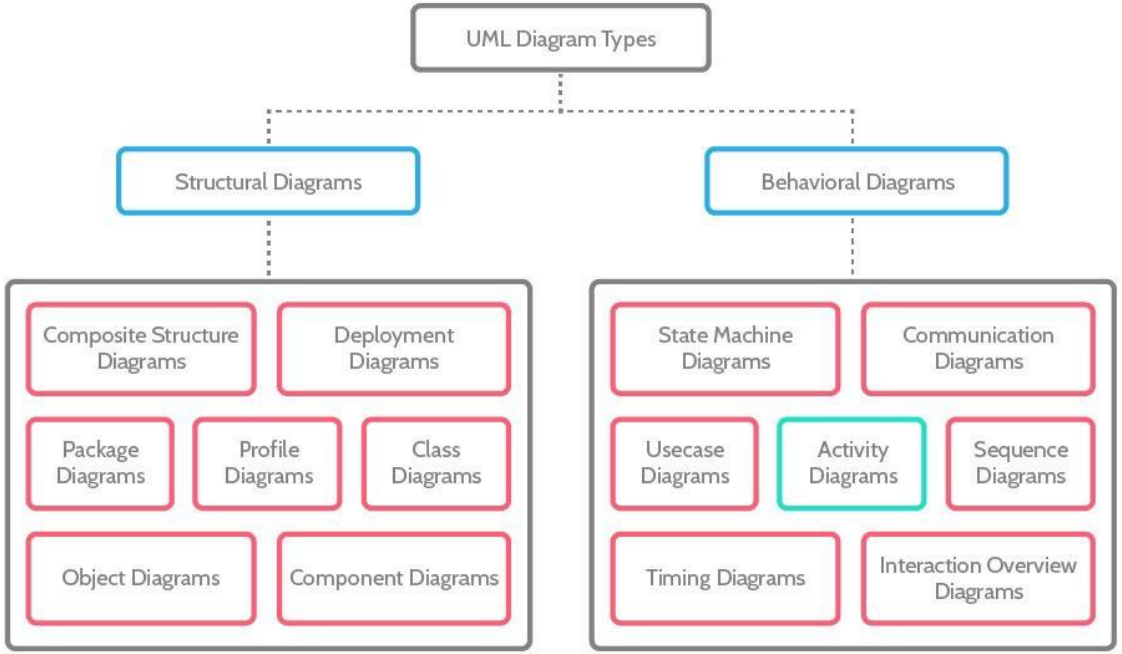


DATOVÉ MODELOVÁNÍ

- Cílem datového modelování je navrhnout kvalitní datovou strukturu pro konkrétní aplikaci a databázový systém, který bude tato aplikace využívat k uložení dat.
- Databázový model je nástroj pro reprezentaci struktury a funkcionality databáze = souhrn pravidel pro reprezentaci logické organizace dat v databázi.

UML

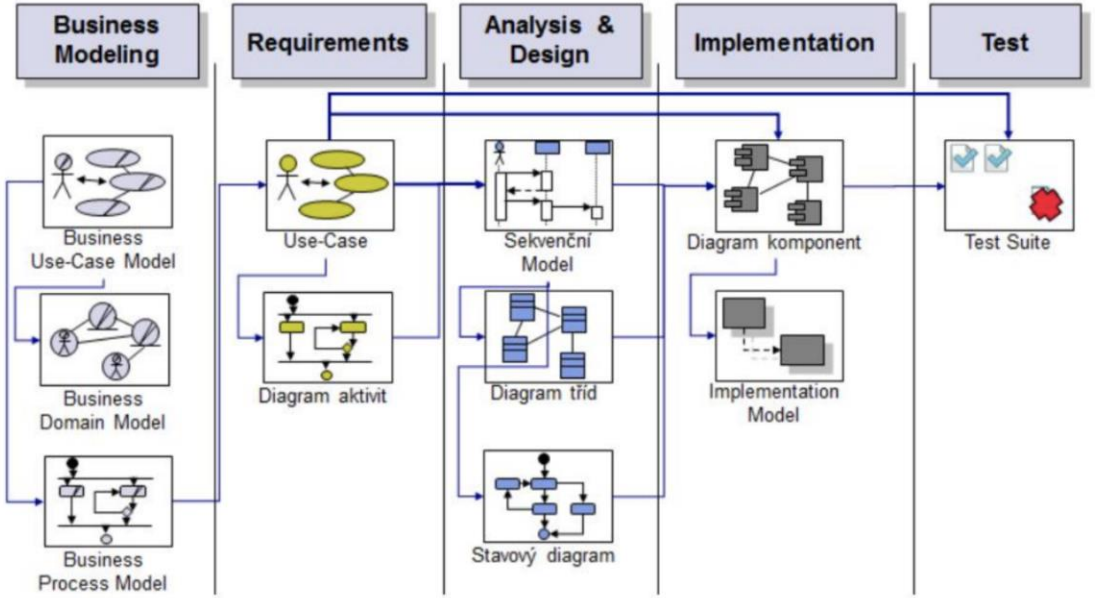
- UML - **Unified Modeling Language**
  - Standard vytv. v 90. letech 20. století
  - Primárním účelem bylo vytvoření standardu pro softwarové inženýrství
  - Vývojem se zabývá OMG (Object Management Group)
  - UML je uznáván jako ISO standard
  - Aktuálně verze UML 2.5
  - Vhodné při objektově orientované analýze
  - Pomocí různých typů diagramů umožňuje zachytit celistvý stav systému z různých úhlů pohledu
  - Více informací k UML například zde: https://www.youtube.com/watch?v=WnMQ8HlmeXc
  - UML eLearning kurz: https://www.itnetwork.cz/navrh/uml



UML – TYPY DIAGRAMŮ

DĚLENÍ UML

1. Funkční náhled
   - Diagram případů užití (use cases)
2. Logický náhled
   - Diagram tříd
   - Objektový diagram
3. Dynamický náhled popisující chování
   - Stavový diagram
   - Diagram aktivit
   - Interakční diagramy
   - Sekvenční diagramy
   - Diagramy spolupráce
4. Implementační náhled
   - Diagram komponent
   - Diagram nasazení



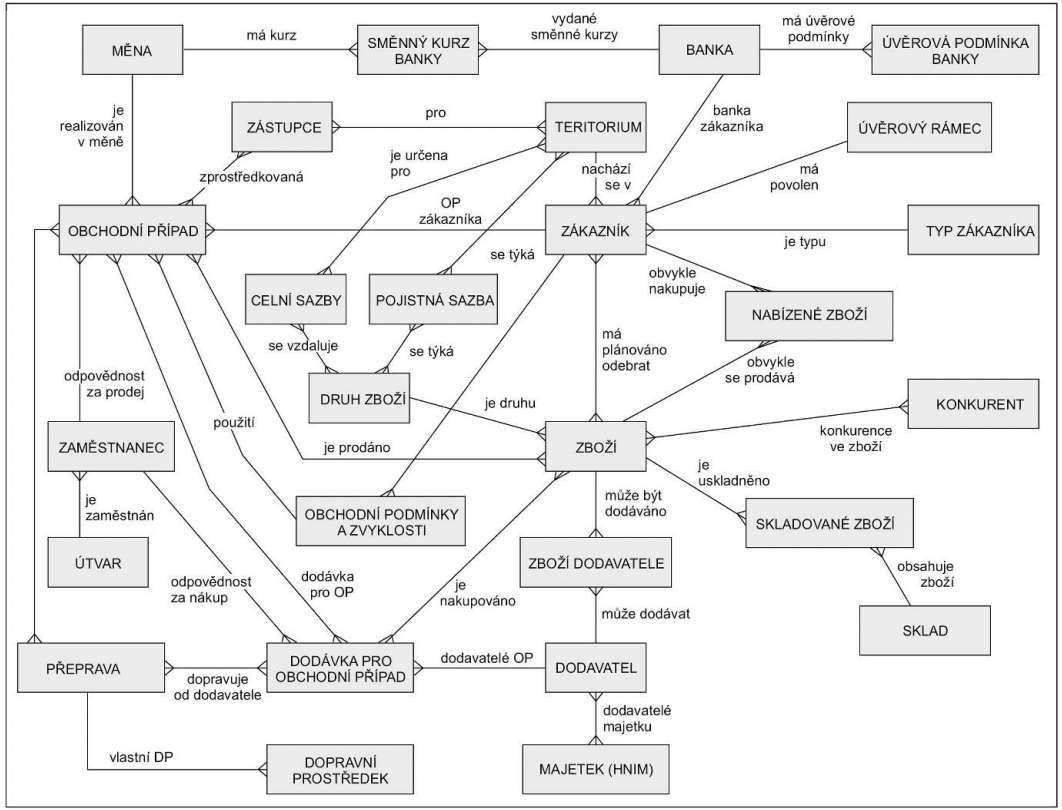
,

ROLE UML VE VÝVOJI IS

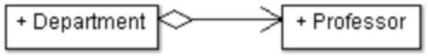
DIAGRAM TŘÍD (CLASS DIAGRAM)

- Diagram tříd (Class diagram) - diagram struktury tříd poskytuje logický náhled na systém.
- Znázorňuje datové struktury, operace u objektů a také jejich vazby.
- Diagram tříd se využívá pro tvorbu ERD a při návrhu implementace.

CLASS DIAGRAM

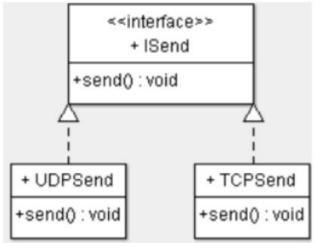


VAZBY V DIAGRAMU TŘÍD

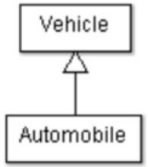


Agregace - vztah vyjdařující katedra má profesory, zánik department neznamená zánik professor

Kompozice - nejsilnější asociace, existence odkazovaného objektu (department) bez majitele (faculty) nemá smysl a zaniká i s ním

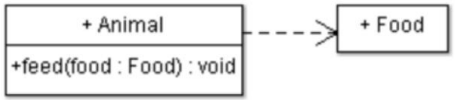


Realizace - implementace rozhraní



Generalizace (druhý obr.) - dědičnost - automobil dědí z vehicle

Asociace - vztah mezi instancemi - posílají si zprávy



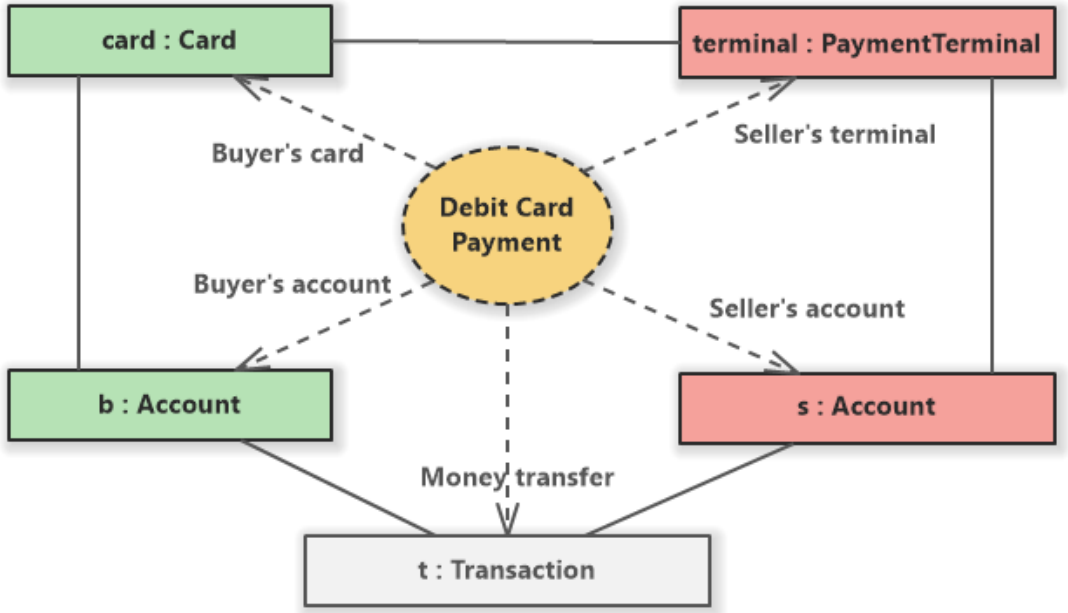
Závislost - třída animal používá třídu food (je na ni závislá)

DIAGRAM SLOŽENÝCH STRUKTUR (COMPOSITE STRUCTURE DIAGRAM)

- Diagram tříd lze větvit do libovolného detailu, avšak na úkor srozumitelnosti celku
- Diagram složených struktur se zaměřuje na konkrétní třídu (zařízení) a izolovaně popisuje její hlavní prvky a součásti, kterými je objekt propojen s okolními objekty a s vnějším světem
- Cílem je popsat interní části, třídy, komunikační porty či užití daného objektu/třídy

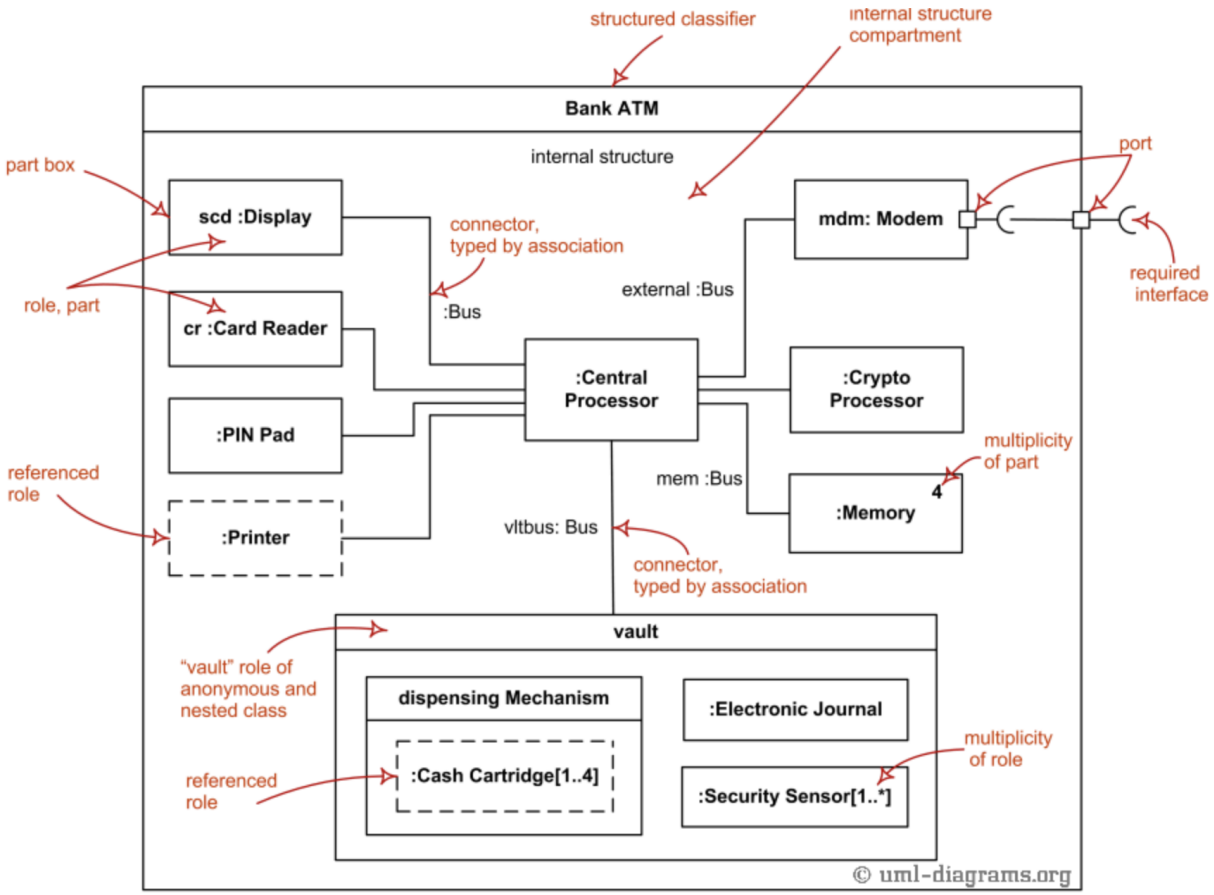
UML : COMPOSITE STRUCTURE DIAGRAM

Debit Card Payment



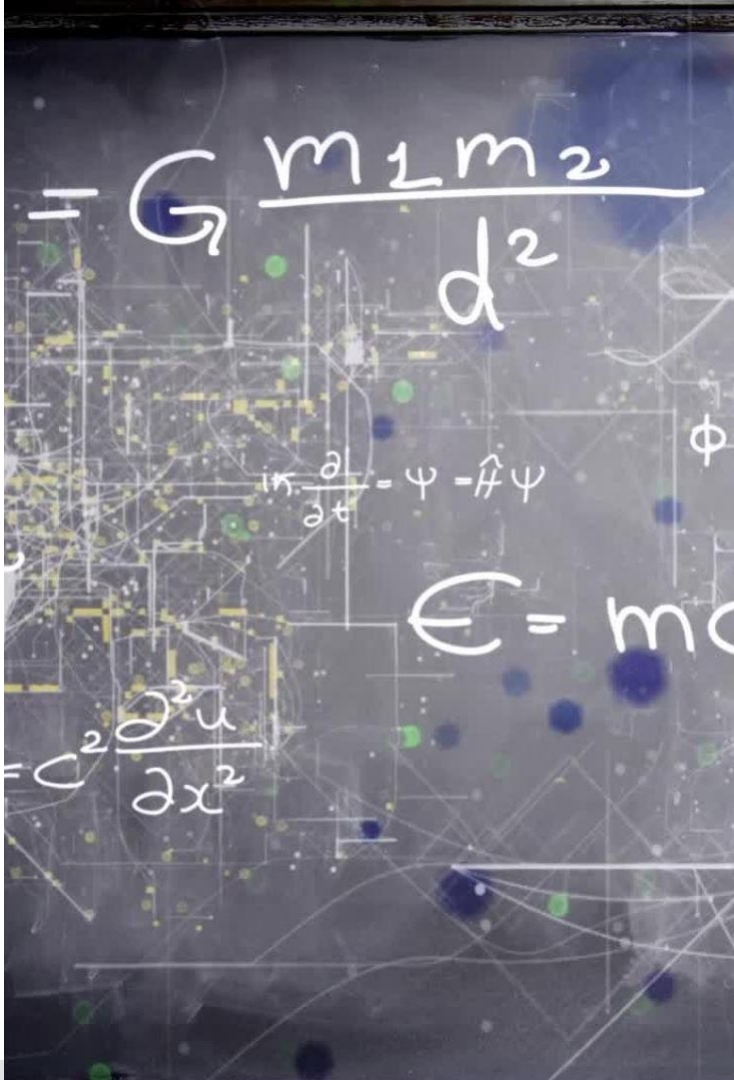
UML : COMPOSITE STRUCTURE DIAGRAM

Bank ATM



UML : USE CASE DIAGRAM

- UML diagram který se zabývá aktéry, vztahy mezi nimi a přístupy aktérů k systému.
- Součástí diagramu je:
  - **Aktér** - ten co sysstém používá a přistupuje k systému k různým případům užití
  - **Use case** - případ využití systému (znázorňuje funkci systému)
  - **Relace** – vazby a vztahy mezi aktéry a případy užití
    - <<include>> - používá se uvnitř systému, kdy jeden use case je využit i v jiném
    - <<extend>> - v případě, kdy proces může být rozšířen o jiný
    - specializace/generalizace

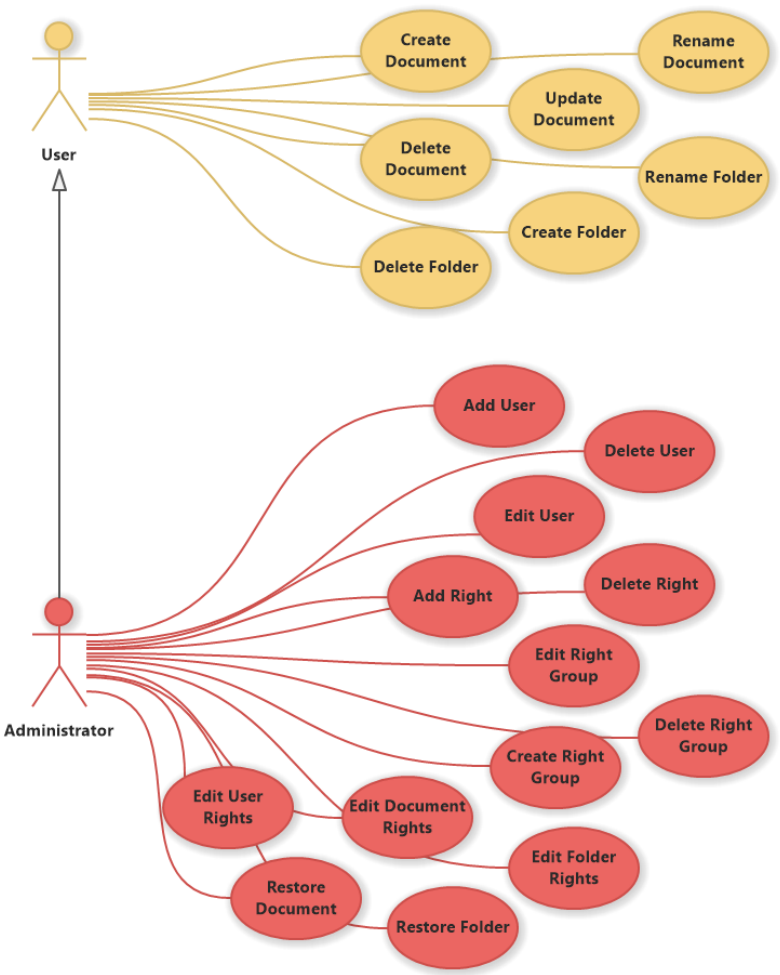


UML : USE CASE

- má vyjadřovat co systém dělá (ne jak) a co od něj očekávají aktéři
- měly by být používány jen pojmy problémové domény - žádné neznámé termíny
- případy užití by měly být co nejjednodušší - ať jim rozumí i zadavatelé - nepoužívat příliš <include> a <extend>
- nepoužívat příliš funkční dekompozici (specializaci)
- primární aktéři umístěni vlevo a pojmenováni krátkým podstatným jménem
- každý aktér by měl být propojen s minimálně jedním use-case
- základní use case vlevo a další kreslit směrem doprava, rozšiřující use-case směrem dolů

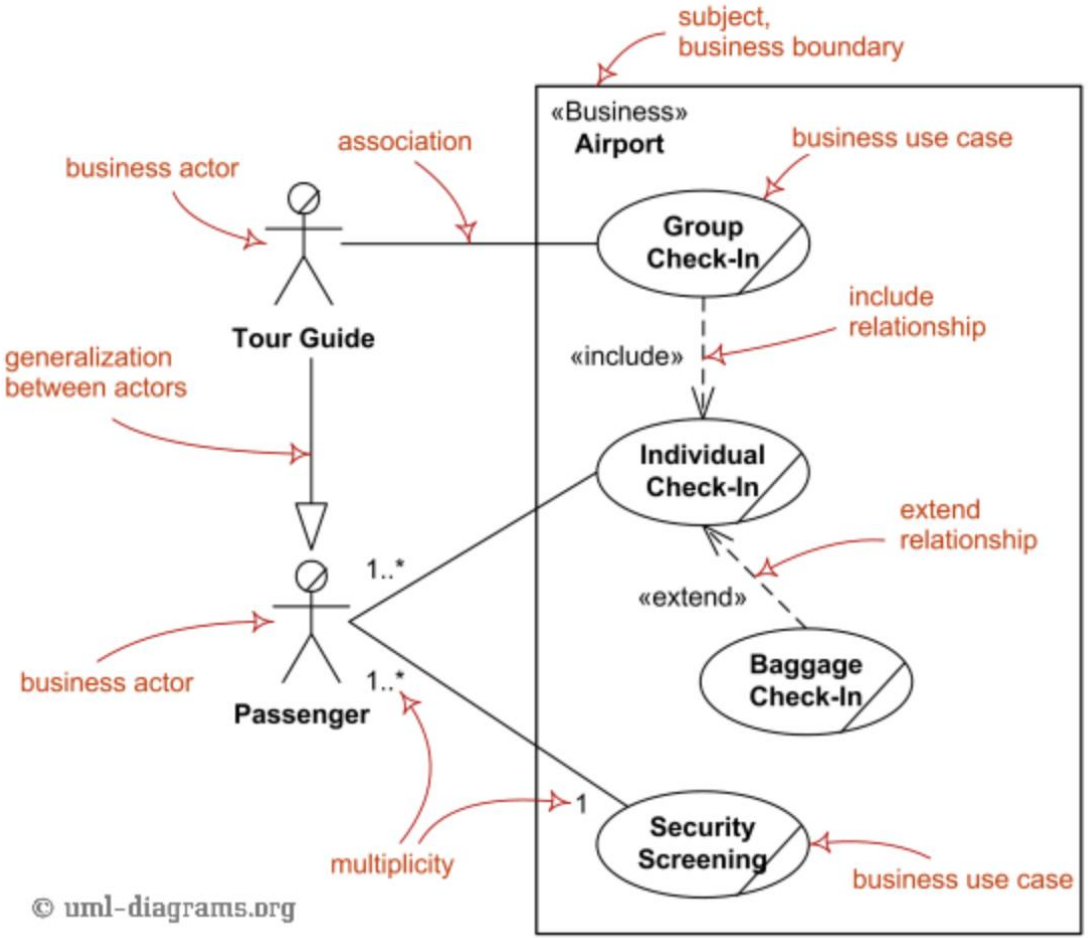
UML : USE CASE DIAGRAM

Document Management System



UML : USE CASE DIAGRAM

Hotel Check-in

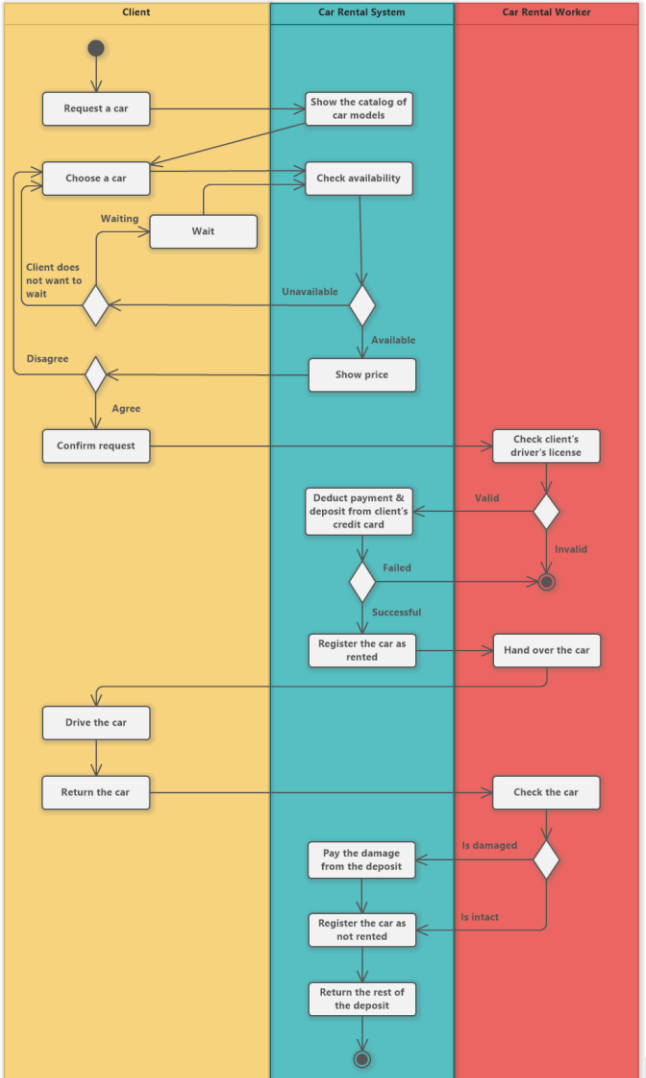


UML : ACTIVITY DIAGRAM

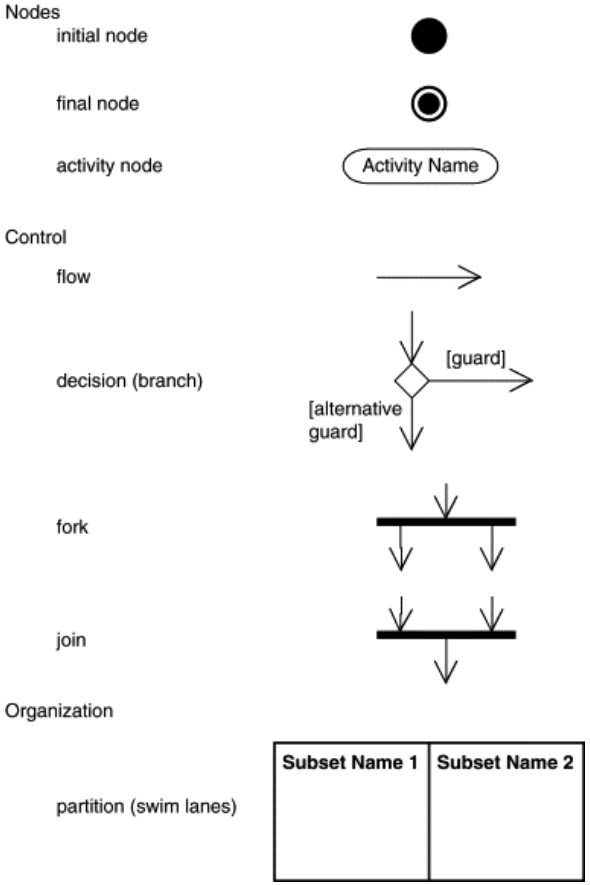
- Použití pro modelování systémů pracujících v reálném čase, systémů pro řízení technologických procesů, nebo paralelních procesů a jejich synchronizaci.
- Další použití pro znázornění složitého scénáře a doplnění sekvenčního diagramu.
- Jsou zvláštním případem stavových diagramů, kde stavy jsou vyjádřeny jako akce a kde přechody jsou spouštěny automaticky po ukončení předchozích akcí nebo aktivit.
- Používají obvykle pouze malou podmnožinu bohaté syntaxe stavových diagramů UML.
- Lze používat symbolů rozhodování (tzv. hodnocení přechodů), symbolů rozvětvení (jeden vstup několik výstupů), spojení (více vstupů jeden výstup), plavecké dráhy (Swimlanes) pro specifikace osob, oddělení nebo tříd zodpovědných za aktivitu.

UML : ACTIVITY DIAGRAM

- Car Rental System



UML – ACTIVITY DIAGRAM NOTACE



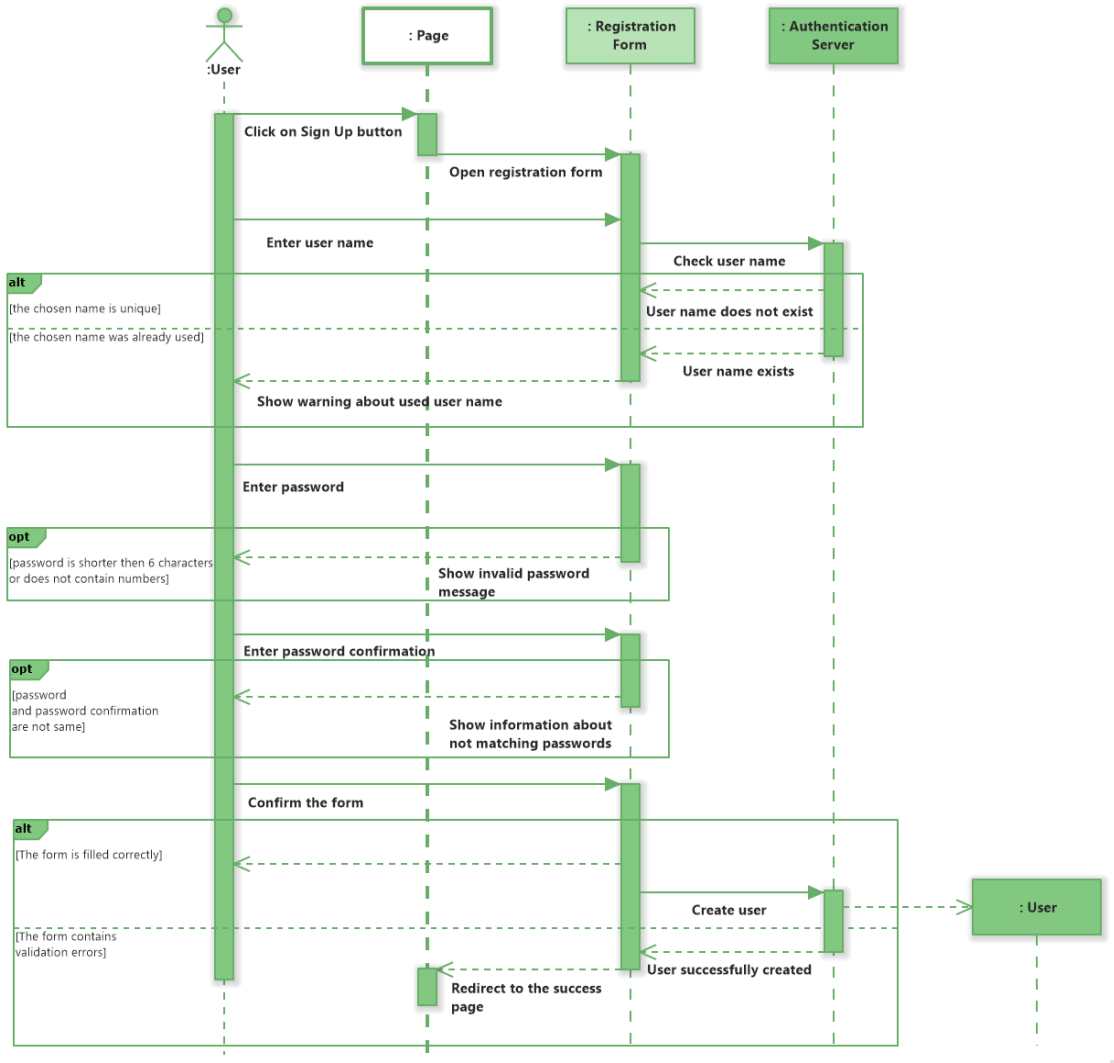
UML : SEQUENCE DIAGRAM

- Zachycuje interakci mezi objekty, zachycuje zasílání zpráv mez objekty v rámci systému.
- Zachycuje dynamické chování s orientací na čas.

- Vlastnosti sekvenčního diagramu:
  - Objekty sekvenčního diagramu spolu komunikují pomocí zasílání zpráv.
  - Popisuje jeden průchod zpráv systémem.
  - Nemá přímé výrazové prostředky pro smyčky, větvení a podmínky.
  - Pro jednoduché případy použiji poznámky.
  - Složité případy řeším separátními diagramy.

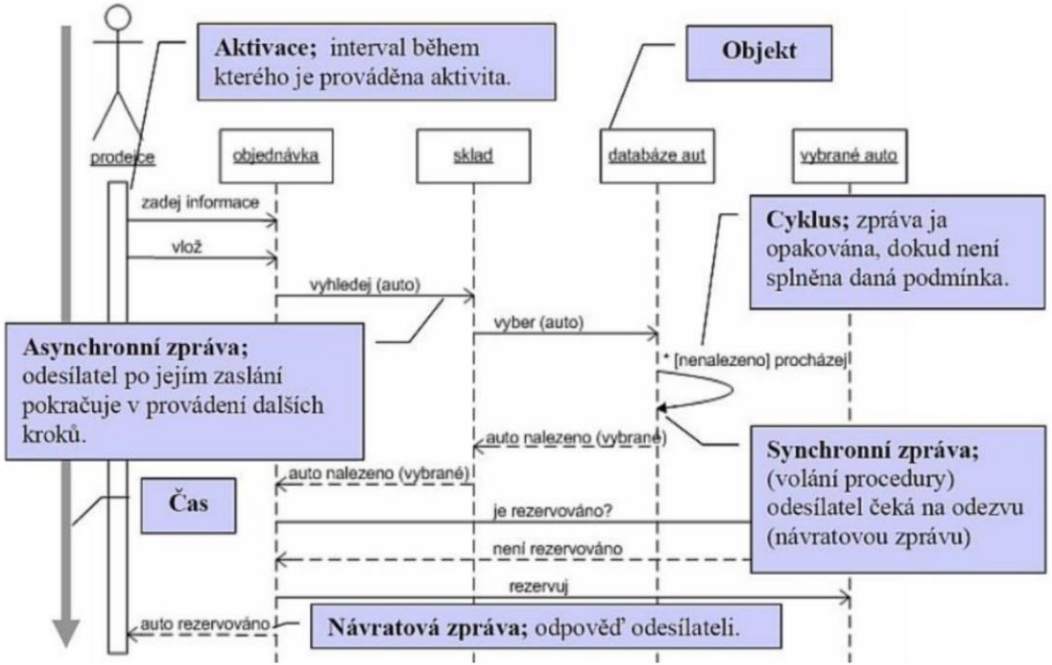
UML : SEQUENCE DIAGRAM

- User Registration



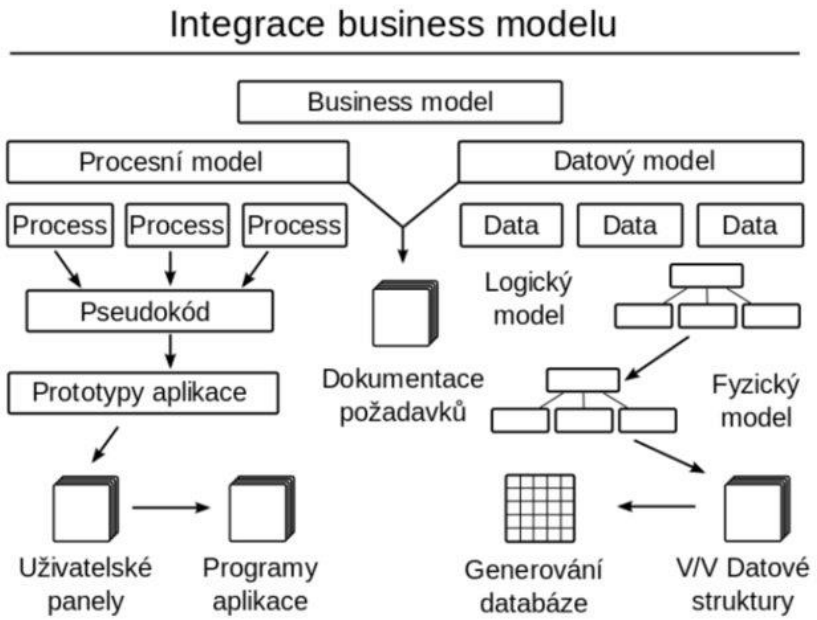
UML : SEQUENCE DIAGRAM

- Product Order



PROCESNÍ MODELOVÁNÍ

- **Business model = process model + data model**
- **Data model** se zabývá tím, JAKÁ DATA business potřebuje, aby mohl úspěšně fungovat.
- **Process model** se zabývá tím, CO business dělá NYNÍ a CO by měl dělat v BUDOUCNU, aby fungoval co nejefektivněji



BPMN

- BPMN = Business Process Modeling Notation
  - Standard, původně vyvíjený v BPMI (Business Process Modelling Initiative), později v OMG
  - Primární účel je pro modelování business procesů - s tím i srozumitelnost pro kohokoliv (mimo IT svět)

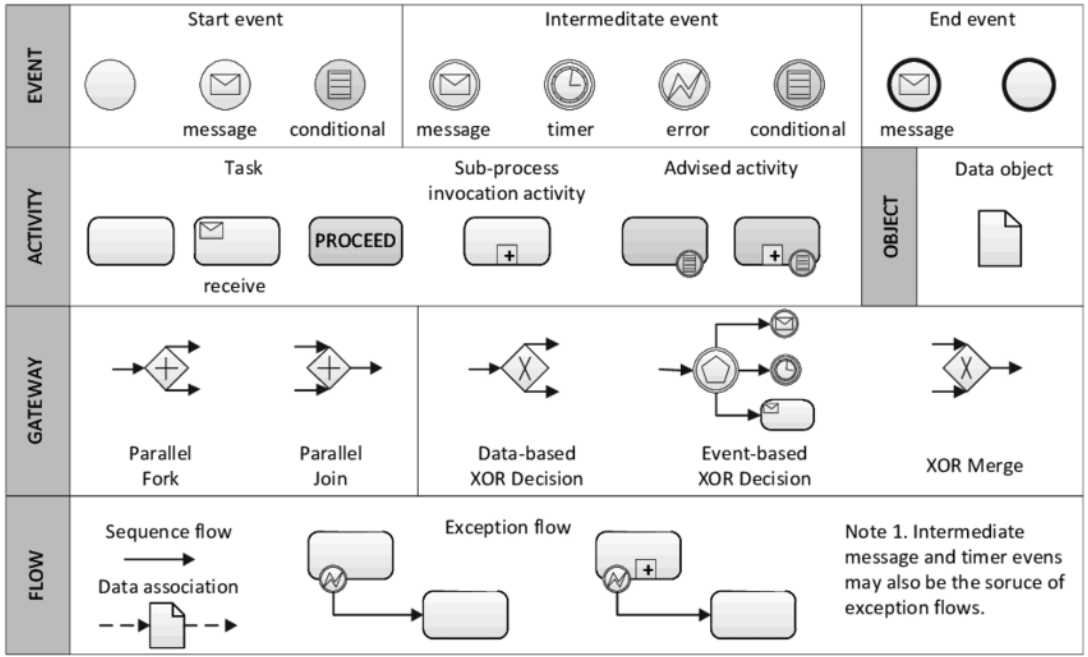
VÍCE INFORMACÍ K TÉMATU PROCESNÍHO MODELOVÁNÍ

- Úvod do BPMN: https://www.youtube.com/watch?v=Uk6WaW9QWn8
- Více o technikách modelování procesů a životní cyklus BPM: https://www.youtube.com/watch?v=eNJ2zdP1BrA
- BPMN - stručné vysvětlení notace (pro audiovizuálně založené): https://www.youtube.com/watch?v=8bp0IYVjrYw
- Přednáška - Essential Business Process Modeling Using BPMN: https://www.youtube.com/watch?v=Wg6330UGKL0

BPMN VS. UML

View	UML Representation		BPMN Representation	
Requirements view	✓	*Use case diagram*: requirements shown as use cases, stakeholders as actors	×	*No representation*. Stakeholder information only found as swimlanes (pools and lanes) on business process diagram. No concept of requirements.
Process structure view	✓	Class diagram	×	No representation.
Process content view	✓	*Class diagram*: processes shown as classes with artefacts shown as attributes and activities as operations	×	*No representation*. Only way to understand content of each process is to look at business process diagram for each process.
Stakeholder view	✓	*Class diagram*: each stakeholder shown as a class	×	*No representation*. Stakeholder information only found as swimlanes (pools and lanes) on business process diagram.
Process behaviour view	✓	*Activity diagram*: stakeholders shown as swimlanes, activities as activity invocations, artefacts as objects	✓	*Business process diagram*: stakeholders shown as swimlanes (pools and lanes), activities as activities, artefacts as data objects
Information view	✓	*Class diagram*: artefacts shown as classes	×	*No representation*. Artefacts only found as data objects on business process diagram.
Process instance view	✓	*Sequence diagram*: each process shown as a lifeline	✓	*Business process diagram*: Sub-processes can be connected together to show executions of processes in sequence.

BPMN NOTACE


EVENT
Start event
message
conditional
Intermeditate event
message
timer
error
conditional
End event
message
ACTIVITY
Task
receive
PROCEED
Sub-process
invocation activity
Advised activity
OBJECT
Data object
GATEWAY
Parallel
Fork
Parallel
Join
Data-based
XOR Decision
Event-based
XOR Decision
XOR Merge
FLOW
Sequence flow
Data association
Exception flow
Note 1. Intermediate message and timer evens may also be the soruce of exception flows.

BPMN VS. UML

- Peixoto, Daniela & A. Batista, Vitor & Atayde, Ana & Borges, Eduardo & Resende, Rodolfo & Isaías, Clarindo & Pádua, P. (2008). A Comparison of BPMN and UML 2.0 Activity Diagrams. Dostupné online: https://www.researchgate.net/publication/228965384_A_Comparison_of_BPMN_and_UML_20_Activity_Diagrams (+dostupné ve Studijních materiálech)

STANDARDY V POPISU PROCESŮ

- Více příkladů užití v praxi:
  - https://www.softwareideas.net/c/39/Diagrams
- BPMN – Infografika
  - http://www.bpmb.de/images/BPMN2_0_Poster_EN.pdf

ÚKOL 2: NÁVRH PROCESU

- **Zadání:** Provozujete malou firmu poskytující cloudové služby (virtuální úložiště, servery, autentizační službu) v modelu B2B. Firma má jednoho account manažera, který se stará o cca 100 zákazníků. Navrhněte pro něho proces pro sledování využívanosti služby Vašimi zákazníky, který bude implementován do Vašeho IS.
- Sledované metriky, resp. účtované parametry (stanovené na měsíční bázi) u zákazníků:
  - Počet uživatelských účtů
  - Množství využitého prostoru (TB)
  - Využití serverových kapacit (strojový čas)
- **Cíl:**
  1. Identifikace účtů k navýšení tarifu (=upsell) a co možná největší zjednodušení celého procesu
  2. Proces navýšení tarifu u zákazníka (v případě, že převyšuje aktuální smlouvou stanovené limity tarifu)
  3. Identifikace účtů, u nichž může dojít ke zrušení předplatného (=propenzitní model aka churn prediction model)

ÚKOL 2: NÁVRH PROCESU

- Termín odevzdání:
  - do **5. května 2023** v 10:00 v Odevzdávárně
  - Pozn.: studenti na výměnném zahraničním pobytu a studenti kombinovaného studia mají termín do začátku zkouškového období (v takovém případě prosím o odevzdání prostřednictvím e-mailu)

ÚKOL 3: STUDIJNÍ LITERATURA

- Nastudujte následující článek:
  - BETZ, Charles T. ITIL®, COBIT®, and CMMI®: Ongoing Confusion of Process and Function. *BP Trends*, 2011, 1: 13. https://www.bptrends.com/bpt/wp-content/publicationfiles/10-04-2011-ART-Ongoing%20Confusion%20of%20Process%20and%20Function-Betz-Final.pdf

DĚKUJI ZA POZORNOST.