5 Model údajov

- identifikovanie údajov, ktoré systém prijíma, s ktorými pracuje a ktoré produkuje
- vyjadrenie vzťahov medzi identifikovanými údajmi a ich časťami
- určenie obsahu údajov

Cieľ analýzy a návrhu údajov

Zabezpečenie transformácie všetkých potrebných informácií do údajových štruktúr a návrh čo najvhodnejšieho spôsobu uchovávania týchto informácií z hľadiska ich následného spracovania.

Dátová entita (entitná množina)

- reprezentuje akúkoľvek informáciu, ktorú treba uchovávať:
  - hľadisko používateľa, t.j. problémovej oblasti
  - hľadisko práce s entitou, t.j. počítačovej reprezentácie
- pomenovanie: podstatné meno v jednotnom čísle
  - typ entity: vyjadruje *spoločné vlastnosti*, čo umožňuje pracovať s viacerými entitami podobne
  - *inštancia:* špecifický, jedinečný výskyt entity

Rôzne pohľady na údaje

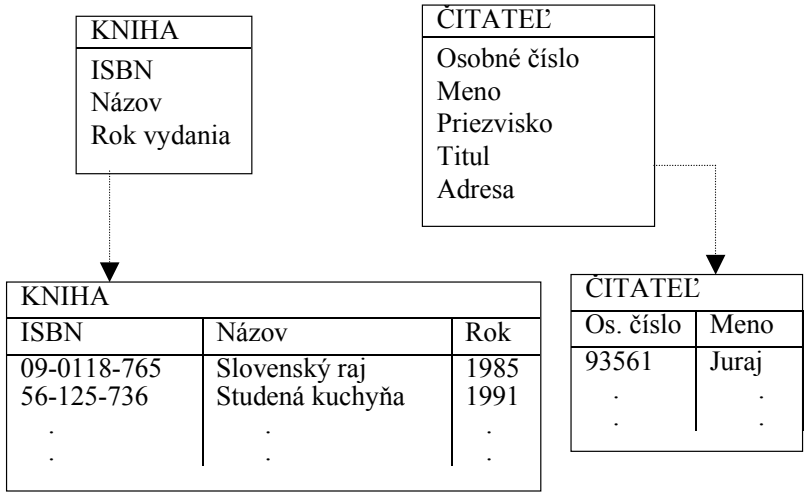
Pri tvorbe modelu údajov treba brať do úvahy rozdielne pohľady jednotlivých používateľov na údaje v systéme. Predpokladajme napr. troch používateľov údajov o telefónoch. Obchodník s realitami pozná adresu a potrebuje vyhľadať meno a číslo telefónu nájomníka. Telefónna operátorka v informačnej službe na základe mena vyhľadáva číslo telefónu, prípadne adresu. A nakoniec úradník, ktorý vystavuje účty za telefón potrebuje meno a adresu zákazníka, pričom jeho telefónne číslo je dané. Vidíme tri rôzne pohľady na tie isté údaje.

Atribút

- pomenovaná vlastnosť entity
- opisuje *čo treba vedieť* o entite
- entity sa odlišujú menom a zoznamom atribútov
- inštancie sa odlišujú *hodnotami atribútov*

Atribút alebo kombinácia viacerých atribútov, ktorých hodnoty (ako celok) jednoznačne charakterizujú inštanciu sa nazýva *kľúč*.

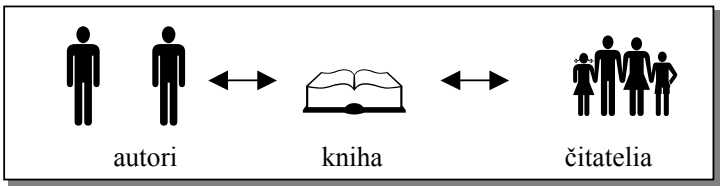
- *primárny kľúč:* „najdôležitejší“ kľúč



Vzťahy medzi entitami

- väzba medzi entitami: *sloveso*
- dôležité pri sprístupňovaní údajov
- **kvantifikácia vzťahu:** stanovenie počtu výskytov určitej inštancie jednej dátovej entity pre jednu inštanciu druhej dátovej entity vo vzťahu
- **kardinalita:**
  - 1 ku 1 – vzťah manželstva medzi mužom a ženou
  - 1 ku *M* – vzťah medzi knihou a vydavateľstvom
  - *M* ku *N* – vzťah medzi pracovníkom a projektom

☞ *Dve entity v rôznych problémových oblastiach môžu mať inú kardinalitu vzťahu (napr. vodič-autobus)!*



Konceptuálny model: pohľad používateľa(ov) (problémová oblasť).

Logický model: implementačne nezávislý pohľad na údaje v systéme

- vyčerpávajúci: všetky potrebné údaje
- neuvažuje počítačovú reprezentáciu
- minimálny: žiadna redundancia.

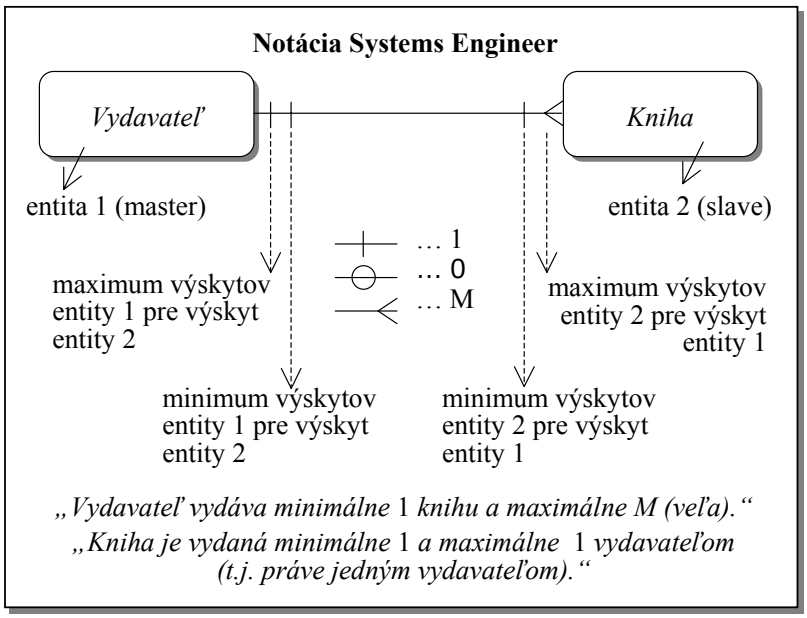
Fyzický model: pohľad reprezentácie v počítači (napr. relačná databáza).

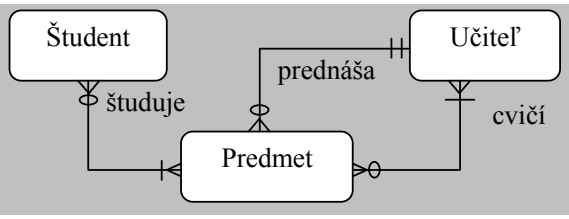
Diagram modelu údajov

angl. Data Model Diagram

Diagram modelu údajov znázorňuje entity a vzťahy medzi nimi. Neznázorňuje vznik, modifikáciu a zánik údajových entít a ani tok spracovania údajov.

- vychádza z entitno-relačného diagramu (ERD) (Chen, 1976)
- graf obsahujúci ako uzly dátové entity, hrany reprezentujú vzťahy medzi entitami
- v ERD sa vzťahy medzi entitami znázorňujú uzlami, t.j. umožňuje definovať aj atribúty vzťahov; v diagrame modelu údajov použijeme väzobnú entitu
- rôzne notácie na vyjadrenie kardinality vzťahu





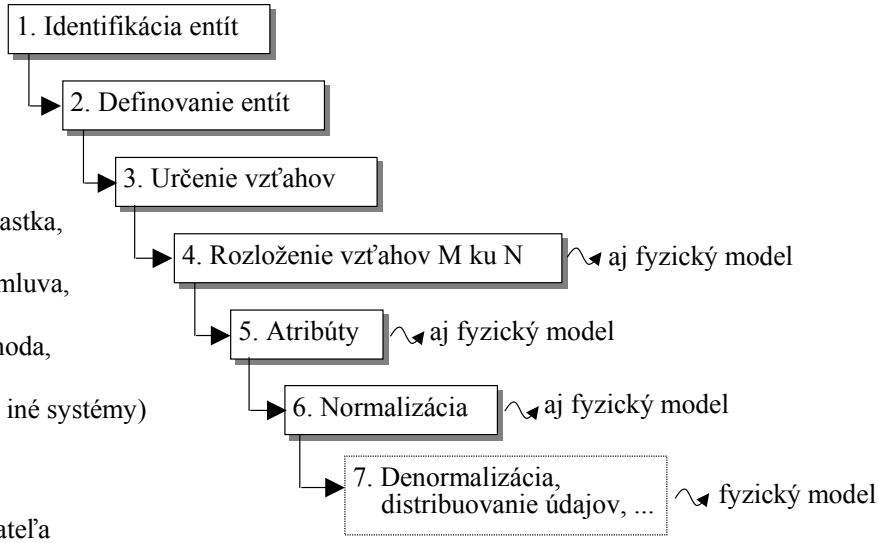
Entity: *študent*, *predmet*, *učiteľ*

Vzťahy:

študent	*študuje*	predmet
učiteľ	*prednáša*	predmet
učiteľ	*cvičí*	predmet

Tvorba modelu údajov

1. Identifikácia entít



Identifikujú sa všetky entity, ktoré sú dôležité v danej problémovej oblasti a o ktorých treba niečo vedieť

- *konkrétne veci* (najčastejšie hmatateľné): zamestnanec, súčiastka, kniha, …
- *konceptuálne entity:* zmluva, organizácia, …
- *udalosti:* dopravná nehoda, nákup, príchod, …
- *rozhranie* (ľudia alebo iné systémy)

Entita → podstatné meno!

Metódy identifikácie entít:

Interview: verifikácia používateľa

Na základe špecifikácie požiadaviek a inej dokumentácie

- gramatická analýza (podstatné mená)
- vychádza sa z opisu správania sa systému, existujúceho funkčného modelu, údajového slovníka, scenárov

Brainstorming: metóda hľadania riešenia problémov, kde členovia skupiny produkujú čo najviac myšlienok a čo najrýchlejšie

Fázy:

1. generovanie myšlienok
   - zaznačí sa každá myšlienka nezávisle od jej hodnoty – neexistujú zlé myšlienky
   - nevyhodnocujú sa
   - každý sa musí cítiť užitočný
2. vyhodnotenie myšlienok

☞ *Dôležitosť zručností a schopností osoby, ktorá vedie stretnutie!*

Metóda DELPHI: rozmýšľanie (brainstorming) na diaľku (prostredníctvom pošty)

- každý na základe pokynov vytvorí zoznam entít
- spoločný zoznam sa distribuuje účastníkom
- vyjadrenie a doplnenie
- distribúcia a doplňanie sa opakuje dovtedy, kým sa nezíska zoznam entít, s ktorým sú účastníci spokojní

2. Definovanie entít

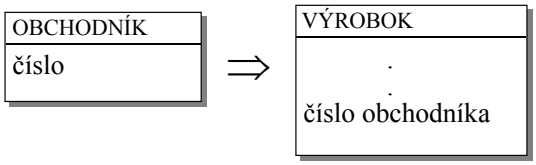
- kontrola, či identifikované entity sú skutočne dátové entity a či sú dôležité pre tento projekt
- pre každú entitu sa skúma: na čo slúži (účel) a z čoho sa skladá (atribúty)

Kontroluje sa:

Identita entity: pýtame sa „Existuje spôsob ako rozlíšiť navzájom inštancie entity?“

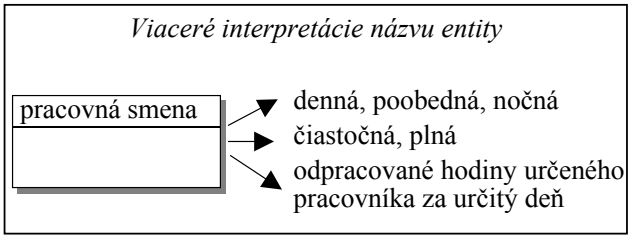
Nevadí, ak identifikátor, ktorý určíme nie je úplne jednoznačný. Dôležité je, že existuje spôsob na rozlíšenie. Ak je entita dôležitá, tak jej inštancie sú rozlíšiteľné. Opak nemusí platiť.

Príklad: To, že dvaja ľudia môžu mať rovnaké rodné číslo (hoci zriedka) neznamená, že nemajú identitu.



Definícia každej entity: pýtame sa „Čo je ____?’’

- jazyk používateľa
- možnosť odhaliť nejednoznačnosť, nepochopenie, chyby, nevyhodnotené predpoklady, rôzne pohľady
- odhalenie podtypov entity
- identifikovaná entita (meno) v skutočnosti reprezentuje niekoľko odlišných dátových entít



Príklady atribútov (a správania): pýtame sa „Čo treba vedieť o _____?’’

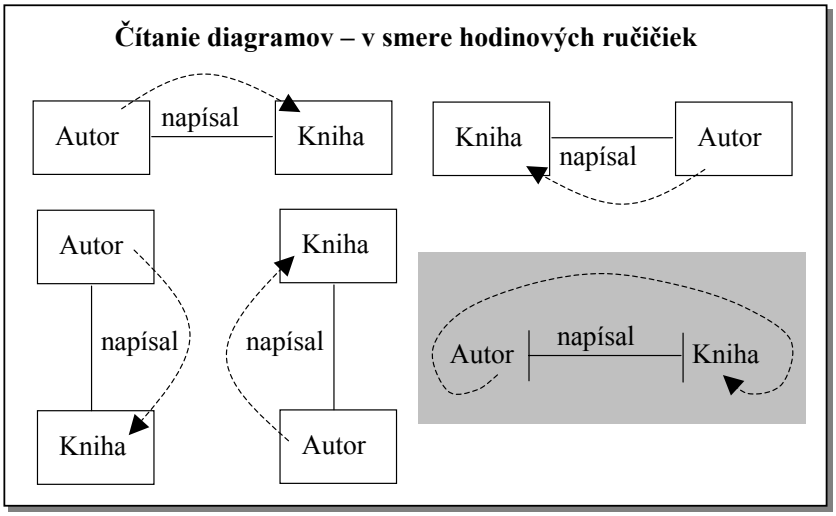
- ak je entita dôležitá, vieme stanoviť atribúty (okrem identifikácie), s ktorými sa pracuje alebo ktoré sa uchovávajú.

3. Určenie vzťahov

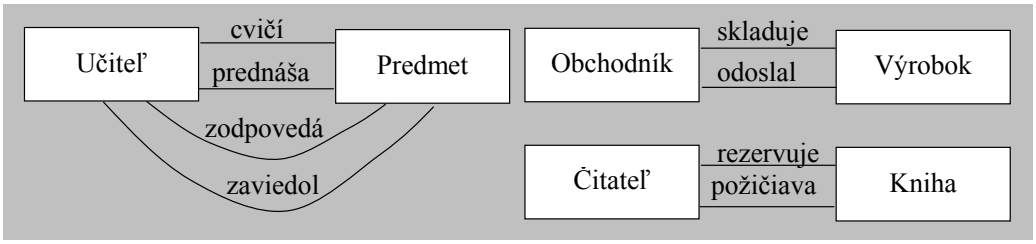
„Čo robí jeden z týchto (inštancia entity1 pre jedného z tamtých (inštancia entity2)?“

Pomenovanie vzťahu: sloveso

- opisuje vzájomnú väzbu medzi entitami
- pomenovanie je dôležité z hľadiska riešeného problému



Viacnásobné vzťahy: uistenie, že sú rôzne a nezávislé



Zahrnutie všetkých možných vzťahov

- treba postupovať systematicky

Príklad:

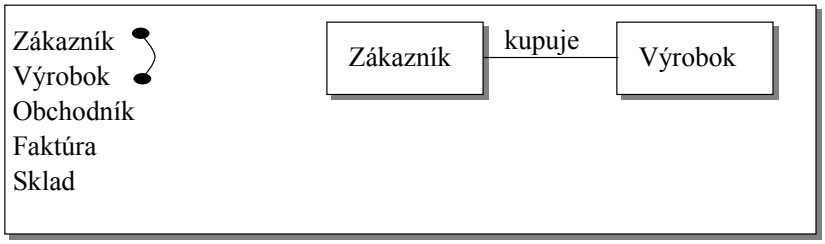
V prostredí obchodnej firmy sme identifikovali tieto entity: zákazník, výrobok, obchodník, faktúra, sklad. Postupne skúmame vzťahy medzi dvojicami entít:

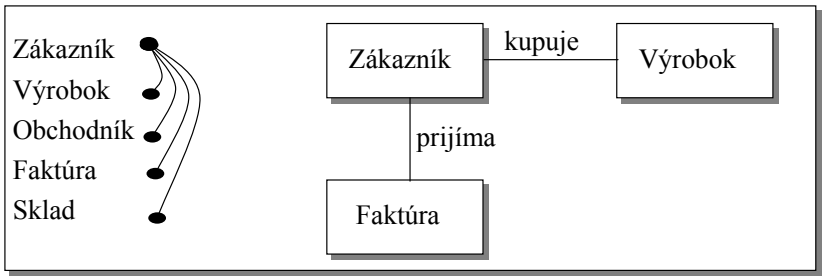
- *Čo robí zákazník s výrobkom?*

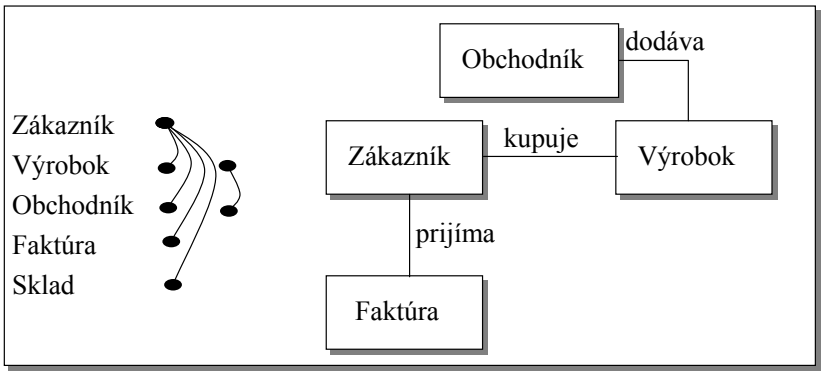
Odpoveď bude zrejme:

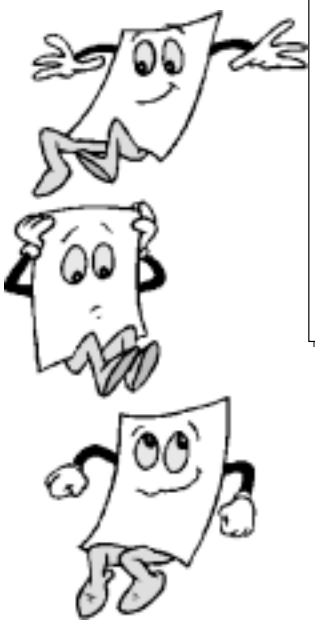
- *Kupuje ho.*

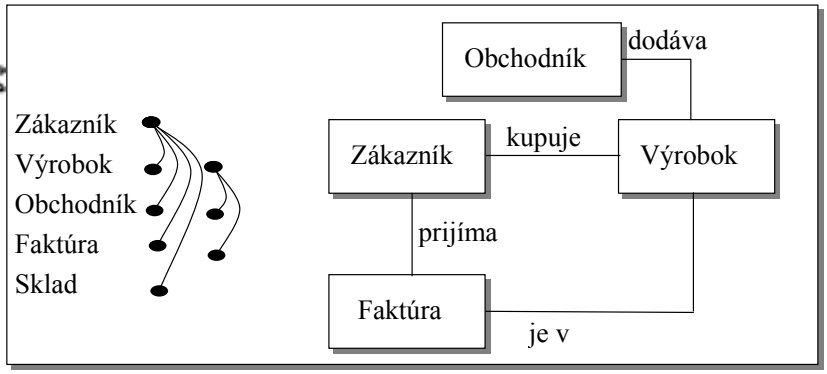
Ďalej sa možno pýtať rôzne otázky súvisiace so vzťahmi jednotlivých entít. Najlepšie je, ak postupujeme systematicky, t.j. postupne skúšame (a značíme si) všetky dvojice entít. Vyhneme sa zabudnutiu niektorého zo vzťahov, čo môže neskôr spôsobiť problémy.

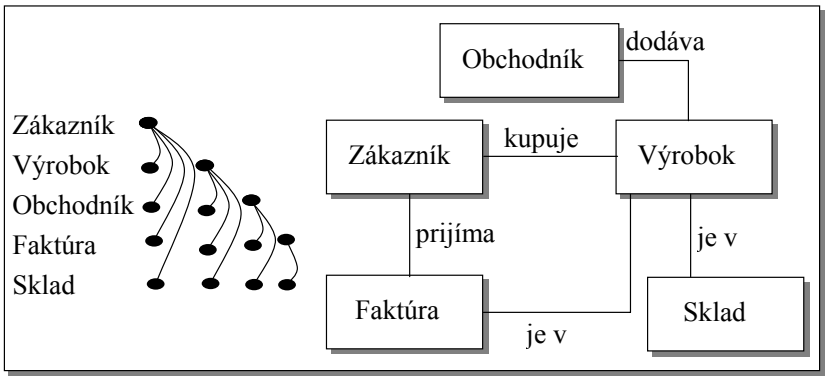












Kvantifikácia vzťahov

„Koľko inštancií sa zúčastňuje na vzťahu (v čase)?“

– *maximálna kardinalita*

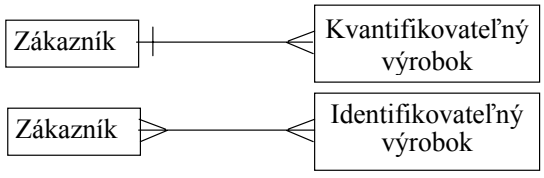
a) „Je možné, aby inštancia entity A bola vo vzťahu s viac ako jednou inštanciou entity B?“

b) A naopak

a) áno	b) nie	1 ku N	A — B
a) nie	b) áno	N ku 1	A — B
a) áno	b) áno	M ku N	A — B
a) nie	b) nie	1 ku 1 **!**	A — B

„Koľko výrobkov kupuje zákazník?“
Veľa.

„Koľko zákazníkov kupuje (jeden) výrobok?“
Veľa alebo jeden. ZÁVISÍ OD SITUÁCIE!



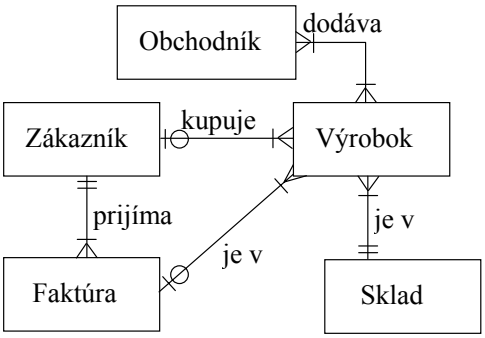
– *minimálna kardinalita*

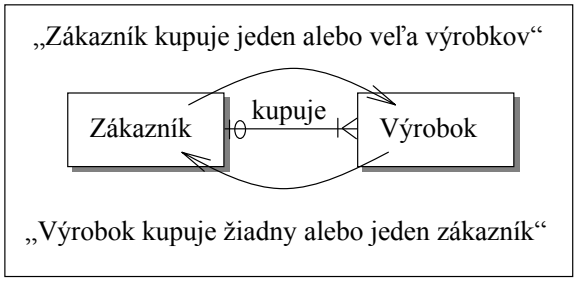
a) „Musí každá inštancia entity A byť vo vzťahu s aspoň jednou inštanciou entity B?“

b) A naopak

a) áno	A — B
a) nie	A — B

Príklad:

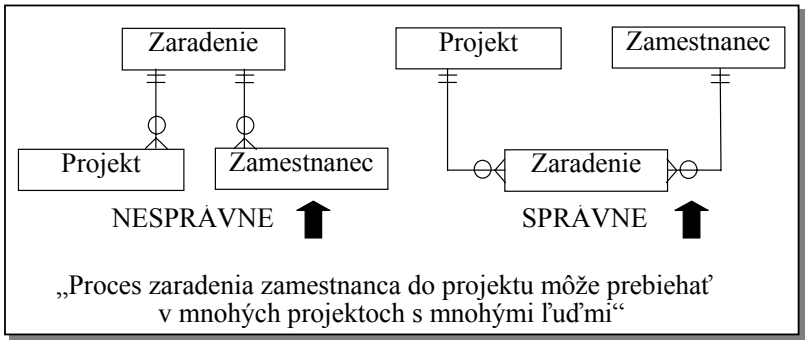




V prípade problémov pri určovaní kardinality sa treba vrátiť k definícii entity.

Poznámka:

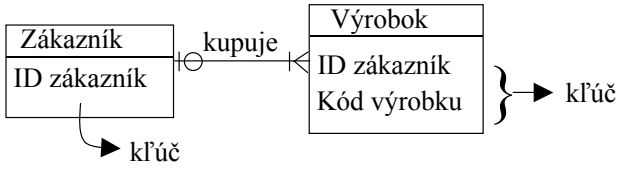
1. Pri definovaní vzťahov dobre zvážte časové hľadisko
2. Dôležitá je zrozumiteľná topológia diagramu
3. Identifikuje entity a nie procesy



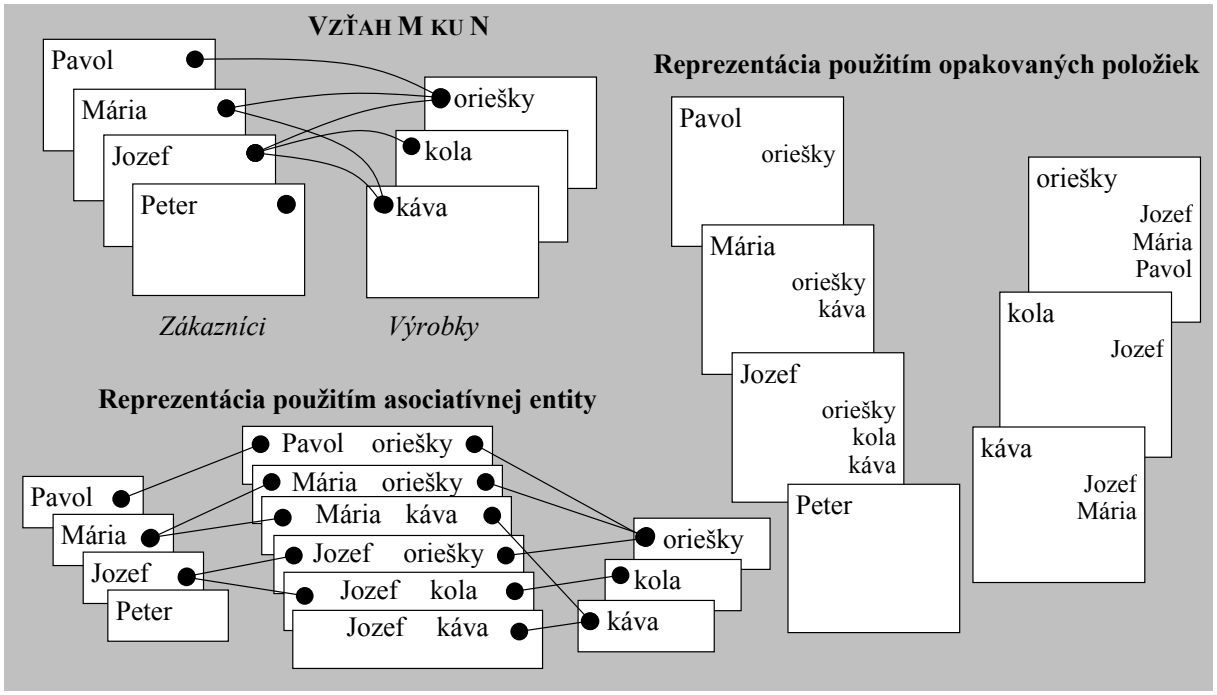
4. Rozloženie vzťahov M ku N

– **jednoduchosť vzťahov 1 ku M** z hľadiska počítačovej reprezentácie
  - smerník
  - cudzí kľúč (relačná databáza)

Treba, aby každá entita mala definovaný kľúč!



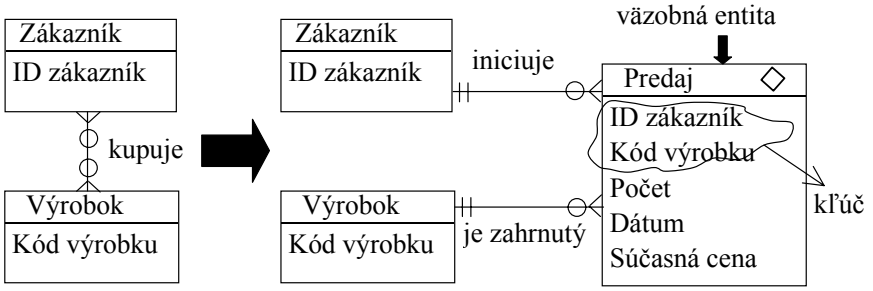
– **zložitosť vzťahov M ku N** z hľadiska počítačovej reprezentácie
  - potrebovali by sme „veľa“ (nevieme presne koľko) cudzích kľúčov



Vyjadrenie atribútov vzťahu (inštancie vzťahu M ku N)

– väzobná entita

Novú entitu treba definovať spolu so vzťahmi (kroky 2, 3).



5. Atribúty

– určia sa pre každú entitu
– treba ich vždy pozorne opísať
– dôležitosť prehodnotenia entít
  - rozhodnutie, či ide o entitu alebo atribút (entity, ktoré tvorí iba identifikátor)
  - entita s jedinou inštanciou

6. Normalizácia

Transformácia modelu údajov s cieľom zaistenia efektívnej počítačovej reprezentácie

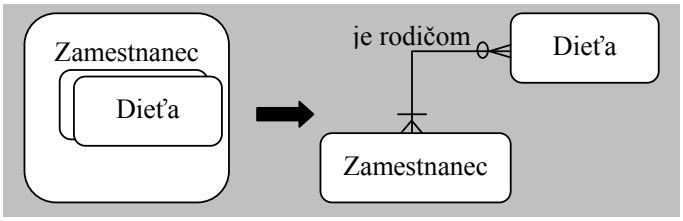
– neberie sa do úvahy sémantika entít; nemusí vyhovovať pri požiadavkách na dobu odozvy systému
– sleduje opakovanie položiek entity a závislosti medzi atribútmi
– snaha o odstránenie redundancie
– automatická normalizácia

Normalizácia modelu údajov

Štruktúra údajov a model údajov navrhnutý používateľom nie vždy zodpovedá požiadavkám na efektívne uchovávanie a spracovávanie informácii. Z tohoto dôvodu sa vykonáva transformácia tohoto modelu do modelu, ktorý tieto požiadavky spĺňa. Táto transformácia sa nazýva normalizáciou modelu údajov.

Normalizácia predstavuje postupnú procedúru transformácie modelu údajov do jednotlivých normálnych foriem. V praxi sa používajú 1., 2. a 3. normálna forma. 1. normálna forma napr. rieši opakujúce sa skupiny položiek v entite (pozri obrázok nižšie: zamestnanec a jeho deti).

Teoreticky bolo odvodených viac foriem, tieto sa však v praxi nepoužívajú. Cieľom normalizácie je postupná transformácia údajov do 3. normálnej formy, ktorá sa z hľadiska efektívnosti uchovávania a spracúvania informácii považuje vo väčšine prípadov za dostatočnú.



Modelovanie údajov a UML

Booch and Rumbaugh začali prácu na UML (Unified Modelling Language) v roku 1994 pod ochranou firmy Rational Inc. V roku 1997 Rational Inc. a Hewlett-Packard navrhli UML ako štandard pre objektovo-orientovanú analýzu a návrh. Tento si osvojila aj firma OMG. V súčasnosti predajcovia modifikujú svoje CASE prostriedky tak, aby boli konzistentné aj s notáciou UML.

UML nie je „nový“ jazyk, ale dopĺňa a zjednocuje doteraz používané metódy, konkrétne Booch (1991), OMT (Object Modelling Technique, Rumbaugh, 1991) a OOSE (Object Oriented Software Engineering, Jacobsen, 1992).

Vznik UML je spojený s objektovo-orientovanou analýzou a návrhom. UML však neslúži iba na modelovanie počas objektovo-orientovanej tvorby softvéru. UML je jazyk na vizualizáciu, špecifikáciu, konštrukciu a dokumentovanie (artefaktov) softvérovo intenzívnych systémov.

Jazyk UML definuje množinu diagramov, ktoré sa používajú v procese tvorby softvéru. Pritom treba zdôrazniť, že UML nedefinuje metodiku pre tvorbu diagramov (t.j. proces, ako tieto diagramy tvoriť a v ktorej etape tvorby softvéru), ale len syntax a sémantiku diagramov. UML definuje tieto typy diagramov:

– diagram tried (angl. class diagram)
– diagram objektov (angl. object diagram)
– diagram prípadov použitia (angl. use case diagram)
– sekvenčný diagram (angl. sequence diagram)
– diagram spolupráce (angl. collaboration diagram)
– stavový diagram (angl. statechart diagram)
– diagram aktivít (angl. activity diagram)
– diagram súčiastok (angl. component diagram)
– diagram rozmiestnenia (angl. deployment diagram)

Diagramy slúžia na modelovanie systému. Každý typ diagramu môže poskytnúť rôzny pohľad na ten istý systém alebo časť systému. Pre porozumenie, resp. pri tvorbe netriviálneho systému, nestačí jeden pohľad a typicky treba vytvoriť množinu diagramov viacerých typov.

UML definuje iba jadro jazyka. Pre prípadné rozširovanie alebo dopĺňanie sa definuje presný spôsob rozšírenia (tzv. mechanizmus rozširovania). UML teda možno dodefinovať tak, aby vyhovoval špecifickým potrebám a požiadavkám.

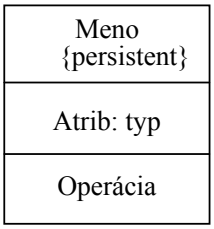
Diagram tried

angl. class diagram

- najpoužívanejší a základný diagram v objektovo-orientovanej analýze a návrhu
- zachytáva štruktúru tried, rozhraní a spoluprác spolu s ich vzájomnými vzťahmi

Entity → triedy, rozhrania

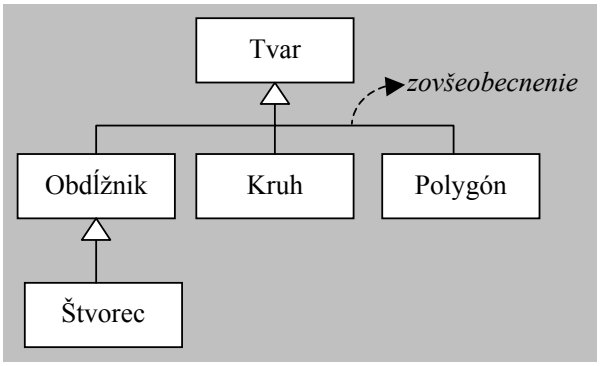
Vzťahy →
- zovšeobecnenie
- závislosť
- väzba
- kompozícia
- zoskupenie

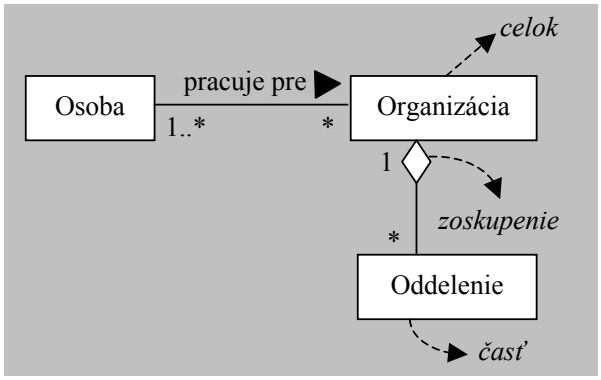


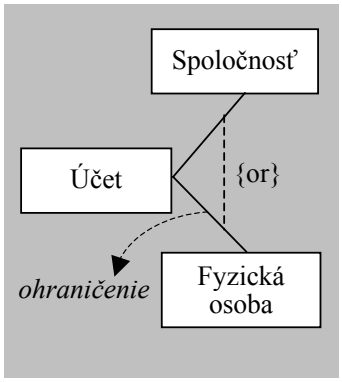
Použitie diagramu tried:

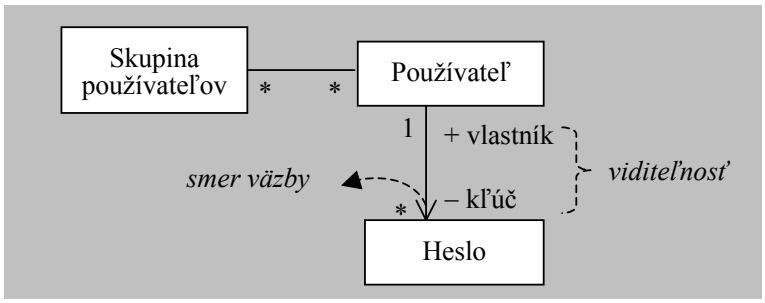
na rôznych úrovniach abstrakcie, t.j. metódy a atribúty tried sa špecifikujú na rozličnej úrovni podrobností:

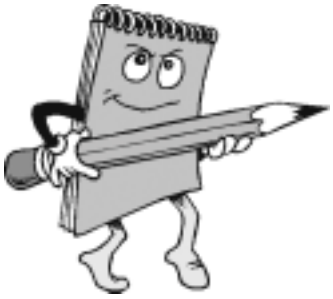
- *konceptuálne diagramy tried:* reprezentujú základné pojmy - triedy a ich vzťahy, nezávisia od jazyka,
- *špecifikačné diagramy tried:* reprezentujú triedy ako rozhrania, implementácia metód v rozhraní nie je dôležitá
- *implementačné diagramy tried:* podrobné diagramy reprezentujúce triedy a ich vzťahy tak, ako budú implementované v programovacom jazyku.











Príklad modelu údajov z prostredia univerzity