Prírodovedecká fakulta UPJŠ Košice

Nám z DBS manuál nestačí I.

(**neoficiálne skriptá pre zimný semester Databázových systémov**)

prednáša: RNDr. Stanislav Krajči, PhD.

IMPRIMATUR
Košice, 7. mája 2002
RNDr. Stanislav Krajči, PhD.

Verzia 1. 6. 2002 (X)
Zostavil Róbert Novotný, `r.novotny@szm.sk`
`http://skmi.science.upjs.sk/~novotnyr`

Obsah

1 Úvod a základné pojmy

1.1 Úrovne pohľadu na databázový systém

- rozlišujeme 4 úrovne pohľadu na databázový systém:
  1. *reálny svet* – stále (nemenné) vlastnosti objektov
  2. *konceptuálna schéma* – dáta sú kategorizované – zuniformovanie dát do štruktúr
  3. *databázová schéma* – zdokonalenie konceptuálnej schémy, algoritmami sa vytvorí databázový systém
  4. *fyzická schéma DB*– súbory – stará sa o ňu databázová aplikácia
- tieto úrovne sú nezávislé, spravidla pracujú na nich rôzne tímy.
- súvisia s vytváraním aplikácie – životný cyklus informačného systému
  - *analýza* (funkčná a dátová)
  - *návrh*
  - *implementácia* – fyzické naprogramovanie

1.2 Databázová technológia

- unifikovaný systém pojmov, nástrojov a techník na vytváranie IS a efektívne spracovanie dát
- riadenie veľkého množstva dát (v TB)
- *perzistentnosť* (trvácnosť) – dáta sa v DB zachovávajú, aj keď sa s nimi nepracuje
- *zdieľateľnosť* – používateľnosť medzi viacerými užívateľmi
- *spoľahlivosť*
  - integrita (integrity) – ucelenosť, dáta nie sú sporné – správnosť údajov a väzieb medzi nimi
  - bezpečnosť (security) – ochrana pred neoprávneným prístupom (autorizácia)

Ďalšie vlastnosti dobrej databázy

- *nerendundantnosť* – dáta sa zbytočne neopakujú
  - dáta nezaberajú veľa miesta
  - pri zmene hodnoty netreba upravovať tie isté dáta na ostatných miestach (znižuje sa riziko chyby)
- *nezávislosť* – programy, ktoré dáta používajú, sú nezávislé na tom, kde sa dáta nachádzajú a naopak.

1.3 Databázový systém

- Má 2 časti:
  - *databáza* – miesto uloženia údajov (a pomocných údajov – *metadát*)
  - *systém riadenia databázy* – programy pracujúce s údajmi
- potreba dvoch jazykov

  - jazyk na definovanie štruktúry dát
  - jazyk na manipuláciu s dátami a ich vyhľadávanie

  Obe požiadavky spĺňa jazyk *SQL* (Structured Query Language).

  - *embedded SQL*
    - je súčasťou iného programovacieho jazyka (C, Pascal)
    - má ďalšie črty (napr. cykly)

- používateľské pohľady (user view)
  - ďalšia črta
  - umožňuje poskytnutie obmedzeného množstva dát

1.4 História

- sieťové databázy
  - spájanie záznamov zo súborov smerníkmi
  - zjednodušenie programovania väzieb medzi dátami

  1965 – konferencia o jazykoch databázových systémov (CODASYL). Vznikol výbor Database Task Group (DTBG)

  1971 – DBTG vydal správu o základných databázových pojmoch

- hierarchické databázy
  - 2. pol. 60. rokov – program Apollo – potreba organizovať $2 \cdot 10^6$ súčiastok hierarchickým spôsobom. Vzniká systém IMS (Information Management System) od spoločnosti IBM – najväčší databázový projekt v histórii v histórii

- relačné databázy
  - nezávislé súbory
  - silné prostriedky na dopyt a aktualizáciu
  - silný rozvoj v 80. rokoch

  1970 – E. F. Codd – článok o relačných databázach

  1974 – jazyk SQL

- objektové databázy
  - vznikajú v polovici 80. rokov

Súčasnosť

- relačné, objektové relačné, textové ..... a ich kompromisy, nič však nie je „všeliekom“
- stále dominujú relačné databázy
- známe databázové systémy: DB2, Informix (IBM), Oracle, SyBase, Paradox

2 Konceptuálny pohľad na dáta

2.1 Entitno-relačný (E-R) konceptuálny model

- systém pojmov, ktoré popisujú (budúci) používateľský program s cieľom vytvoriť špecifikáciu štruktúry databázy

2.1.1 Pojmy entita a vzťah

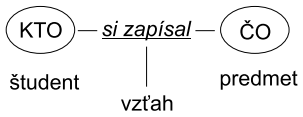
- *entita* (angl. entity) – objekt reálneho sveta, ktorý je schopný nezávislej existencie a je jednoznačne odlíšiteľný od ostatných objektov
  Príklad: študent Jozef Púčik s rodným číslom 82XXXX/XXXX
- *vzťah* (často aj *relácia*) (angl. relationship) – väzba medzi dvoma (alebo viacerými) entitami
  Príklad: študent JP má zapísaný predmet DBS
  Vymedzenie entita vs. vzťah je voľné a neexistuje jednoznačné pravidlo (spravidla entita zodpovedá podmetu resp. podstatnému menu, vzťah prísudku)

2.1.2 Ďalšie pojmy – atribút a vzťah

- *atribút* (angl. attribute) – funkcia, ktorá priraďuje entitám alebo vzťahom nejakú hodnotu určujúcu podstatnú (dôležitú) vlastnosť
  Príklad: farba auta
- *doména* (angl. domain) – možné hodnoty atribútov, resp. množina funkčných hodnôt
  Príklad: celé čísla, reťazce znakov, dátumy

2.1.3 Fázy vytvárania E-R modelu

- *identifikovanie typov entít* ako množín objektov rovnakého druhu (typu)
  Príklad: študent, predmet,...
- *identifikovanie typov vzťahov*, do ktorých entity môžu vstupovať
  Príklad: študent MÁ ZAPÍSANÝ predmet

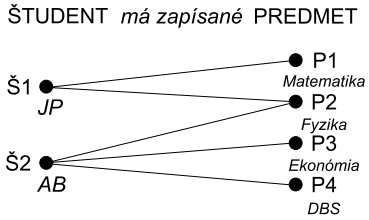


Obr. 1: Identifikovanie typov entít a vzťahov

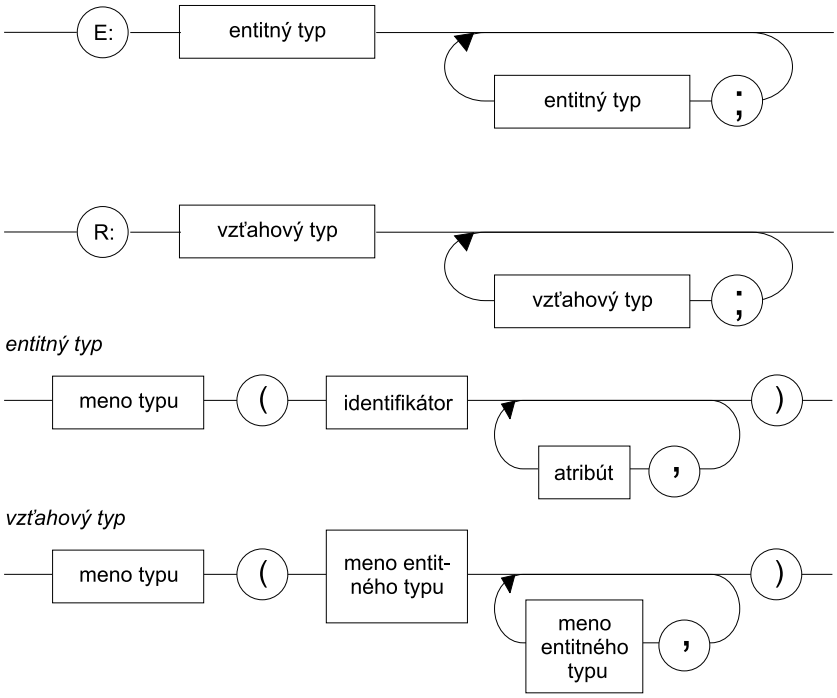
- *priradenie relevantných atribútov* ako entitám, tak i vzťahom
  Príklad: u študenta jeho meno, u predmetu jeho názov, kód, počet kreditov
- *formulácia integritných obmedzení* vyjadrujúcich s väčšou či menšou presnosťou súlad s modelovanou realitou (určenie doménových typov – doménou atribútu Cena je množina reálnych čísel, dátum „od“ nesmie presiahnuť dátum „do“)

2.1.4 Typ (entity alebo vzťahu)

- entity študent JP s rodným číslom a študent AB s rodným číslom sú 2 rôzne entity, ale majú *čosi* spoločné
- obe entity vystupujú v role študenta, sú to entity rovnakého typu
- sú to jeho *výskyty* alebo *inštancie*
  - výskyt entitného typu „študent“: $Š_1$, $Š_2$
  - výskyt entitného typu „predmet“: P_1, P_2, P_3, P_4
  - výskyty relačného typu „má zapísaný“: $Š_1P_1$, $Š_1P_2$, $Š_2P_3$, $Š_2P_4$, $Š_2P_2$
- Často sa pojmy entita a entitný typ zamieňajú, záleží od kontextu



Obr. 2: Výskyty entitných typov a vzťahov a ich inštancie



Obr. 3: Syntaktické diagramy pre entitný a relačný typ

2.1.5 Identifikačný kľúč

- každá entita musí byť jednoznačne identifikovateľná a to niektorým atribútom (alebo skupinou atribútov)
  Príklad: zamestnanci podľa osobného čísla alebo rodného čísla alebo trojicou (meno, priezvisko, dátum narodenia)

- pre jednoznačnosť sa často používa *umelý kľúč* (číslo)

- názov entity resp. entitného typu je vzhľadom na celú databázu jednoznačný

2.1.6 Zápis konceptuálnej schémy

Príklad:
E: Študent(*kód*, *meno*, *priezvisko*,...);
E: Predmet(*kód*, *názov*, *počet_kreditov*,...);
R: MáZapísaný(Študent, Predmet);

Grafické znázornenie (UML)

- entity sa zapisujú do obdĺžnikov
- vzťahy do kosoštvorcov
- atribúty pod názov entity
- pozri obr. 4

Obr. 4: Znázornenie entít a vzťahov v UML

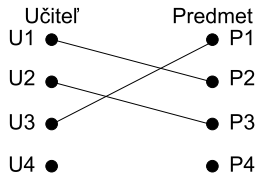
2.1.7 Atribúty

- typy sú popísané až vtedy, keď sú každému pridelené príslušne popísané atribúty
- atomické (jednoduché) atribúty priraďujú entite najviac 1 nedeliteľnú hodnotu
- vlastnosti atribútov
  - *meno* – je jednoznačne určené vzhľadom na príslušný entitný typ, v rámci celého systému sa môže opakovať
  - *typ atribútu* – množina jeho hodnôt (doména) – napr. celé čísla
  - *kľúčovosť* – príslušnosť k identifikačnému kľúču
  - či pripúšťa *prázdnu hodnotu* (nezistiteľnú, neznámu) – NULL
  - či má *jednoznačnú hodnotu* – UNIQUE (funkcia je prostá)
- čo nie sú atribúty entity
  - nepodstatnosti (nepodstatné vlastnosti) – o podstatnosti rozhoduje analytik
  - informácie, ktoré sú vlastnosťami iného objektu
    Príklad: u pacienta počet lôžok izby na ktorej leží (je to atribút izby) – inak by sa táto informácia opakovala u každého pacienta z tej izby – vznikla by redundancia dát.
    Preto návrh
    E: PACIENT(*meno, priezvisko, č_izby, počet_lôžok*);
    treba transformovať na
    E: PACIENT(*meno, priezvisko, č_izby*); E: IZBA(*číslo izby, počet_lôžok*);
    R: JEUMIESTNENÝNA(PACIENT, IZBA);
    Odpoveď na počet lôžok izby, v ktorej leží pacient, je potom dostupná cez nadviazaný vzťah JEUMIESTNENÝNA.
- neatomické atribúty
  - (zatiaľ(?)) iba na konceptuálnej úrovni
    - *skupinové atribúty* (zložené) – napr. adresa: ulica, číslo popisné, číslo evidenčné, PSČ, ...
      - štruktúra nemusí byť dvojúrovňová (ako adresa), môže to byť celá hierarchia
      - používa sa vtedy, ak sa pracuje aj s celkom aj so zložkami osobitne
    - *viachodnotové atribúty*
      - Príklad: ZAMESTNANEC(...,*deti*:MULTIVALUED);

2.2 Integritné obmedzenia pre vzťahy

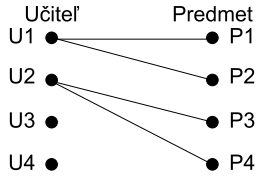
2.2.1 Kardinalita vzťahu

- binárny vzťah
  - Príklad: vzťah UČÍ medzi entitami Učiteľ, Predmet
  - 1:1
    - každý učiteľ učí najviac jeden predmet
    - každý predmet je učený najviac jedným učiteľom
    - sú tu zahrnuté aj vzťahy 1:0 (učiteľ neučí žiaden predmet), 0:1 (predmet nie je učený žiadnym učiteľom)
    - v prísnejšom chápaní sa však vzťahy 1:0, 0:1 nepripúšťajú



Obr. 5: Učiteľ a predmet vo vzťahu 1:1

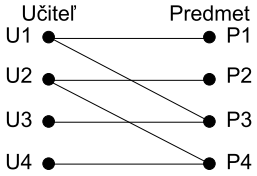
  - 1:N
    - učiteľ môže učiť viacej predmetov, ale predmet môže byť učený najviac jedným učiteľom
    - vo všeobecnosti môžu byť zahrnuté aj vzťahy 1:1, 1:0, 0:1
    - v prísnejšom chápaní sa tieto možnosti nepripúšťajú (teda každý učiteľ učí viac než jeden predmet, každý predmet je učený práve jedným učiteľom)
    - je tu dôležitý smer – v prípade N:1 je vhodné zmeniť názov vzťahu na „je učený“



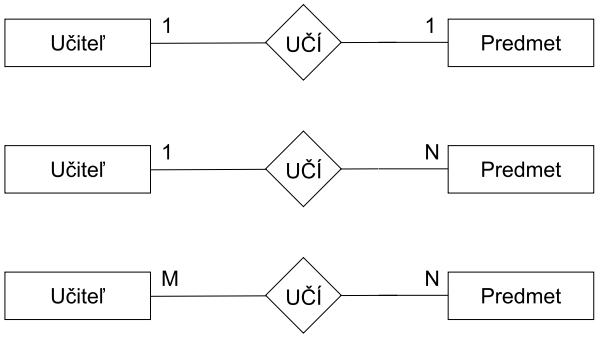
Obr. 6: Učiteľ a predmet vo vzťahu 1:N

  - M:N
    - učiteľ môže učiť viac predmetov
    - predmet môže byť učený viacerými učiteľmi
    - vo všeobecnosti sú zahrnuté aj vzťahy 1:1, 1:N, 0:1, 1:0
  - pozri obr. 8
- ternárny vzťah
  - rozlišujeme vzťahy 1:1:1, 1:1:M, 1:M:N, M:N:P
  - pozri obr.9

Zložitejšie kardinality sa vyjadrujú obrázkami problematicky.



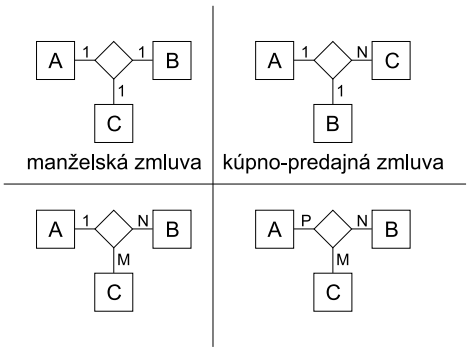
Obr. 7: Učiteľ a predmet vo vzťahu M:N



Obr. 8: Príklady na rôzne typy binárnych vzťahov

2.2.2 Determinanty

- entita jedného typu determinuje (určuje) jednoznačne entitu druhého entitného typu
  - 1:1 – meno učiteľa jednoznačne určuje predmet, predmet jednoznačne určuje učiteľa
  - 1:N – učiteľ jednoznačne neurčuje predmet, ale predmet určuje učiteľa
  - M:N – ani učiteľ nedeterminuje predmet, ani naopak
- členstvo vo vzťahu
  - povinné (obligatórne) vs. nepovinné – vyjadruje, že inštancia jedného typu nemôže existovať bez zapojenia do vzťahu s inštanciou druhého typu
    Príklad: FILM a jeho KÓPIA
  - musí vs. môže
  - totálna vs. parciálna účasť



Obr. 9: Príklady na rôzne typy ternárnych vzťahov

2.2.3 Slabé a silné entitné typy

- slabý entitný typ
  - ak môžu existovať 2 rôzne inštancie majúce rovnaké kombinácie vlastných atribútov a sú sami osebe nerozlíšiteľné. Sú však identifikovateľné tým, že sú v povinnom (tzv. *identifikačnom*) vzťahu k inštancii iného entitného typu (tzv. *identifikačný vlastník*)
  - Príklad: film a jeho kópia. Typ KópiaFilmu (slabý entitný typ, pretože každý film môže mať kópiu s číslom 1) je existenčne závislá od entitného typu Film (identifikačný vlastník).
  - súčasťou identifikačného kľúča slabého entitného typu musí byť (popri prípadných vlastných atribútoch), identifikačný kľúč jeho identifikačného vlastníka – slabý entitný typ „zdedí“ jeho kľúčové atribúty
  - ak by bolo číslo kópie jednoznačné v rámci celej požičovne (nezávisle od filmu), nebol by to už slabý entitný typ, ale existenčná závislosť na type Film by ostala
- silný (regulárny) entitný typ
  - ten typ entity, ktorý nie je slabý

2.2.4 Jemnejšie vyjadrenie integritného obmedzenia vzťahov

- min-max integritné obmedzenie
- pre binárny typ vzťahu $R(E_1, E_2)$ označme

  $$E_i : (\min_i, \max_i)$$

  minimálny a maximálny počet výskytov entity typu E_i vo vzťahovej množine R.
  Zapisujeme

  $$R(E_1 : (\min_1, \max_1), E_2 : (\min_2, \max_2))$$

- $\min_i = 0$ – vyjadruje nepovinné členstvo E_i vo vzťahu, t.j. možnosť existencie entity daného typu nezávisle na existencii entity druhého typu
- $\min_i > 0$ – vyjadruje existenčnú závislosť na entite druhého typu
- (niektoré) možnosti
  - 1:1
    - $E_1(0,1); E_2(0,1)$ ... E_1 môže, E_2 môže (sa podieľať na vzťahu)
    - $E_1(0,1); E_2(1,1)$ ... E_1 môže, E_2 musí
    - $E_1(1,1); E_2(0,1)$ ... E_1 musí, E_2 môže
    - $E_1(1,1); E_2(1,1)$ ... E_1 musí, E_2 musí
  - 1:N
    - $E_1(0,N); E_2(0,1)$ ... E_1 môže, E_2 môže (sa podieľať na vzťahu)
    - $E_1(0,N); E_2(1,1)$ ... E_1 môže, E_2 musí
    - $E_1(1,N); E_2(0,1)$ ... E_1 musí, E_2 môže
    - $E_1(1,N); E_2(1,1)$ ... E_1 musí, E_2 musí

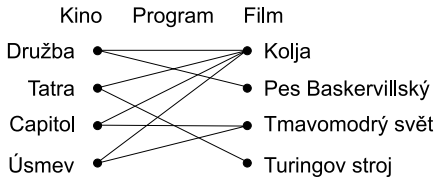
    Všimnite si, že 1 v relácii 1:N sa vzťahuje na entitný typ E_2, N sa vzťahuje na entitný typ E_1
  - M:N
    - $E_1(0,N); E_2(0,M)$ ... E_1 môže, E_2 môže (sa podieľať na vzťahu)
    - $E_1(0,N); E_2(1,M)$ ... E_1 môže, E_2 musí

- $E_1(1,N); E_2(0,M)$ ... E_1 musí, E_2 môže
- $E_1(1,N); E_2(1,M)$ ... E_1 musí, E_2 musí

- zopár užitočných príkladov
  - PÔŽIČKA(*čitateľ*:(0,4), *exemplár*:(0,1)) ... 1:N
    Čitateľ nemusí mať požičanú žiadnu knihu a môže mať vypožičané najviac 4 knihy.
    Daný exemplár môže byť vypožičaný najviac raz, v danom okamihu ho buď má nejaký čitateľ, alebo sa nachádza v regáli.
  - REZERVÁCIA(*čitateľ*:(0,4), *kniha*(0,m))
    Daný čitateľ nemusí mať rezervovanú žiadnu knihu a môže mať rezervované najviac 4 knihy.
    Daná kniha môže byť rezervovaná ľubovoľným počtom čitateľom (aj nulovým).
  - MÁEXEMPLÁR(*kniha*:(1,n), *exemplár*:(1,1))
    V knižnici môže byť niekoľko exemplárov danej knihy (ale aspoň 1).
    Každý exemplár musí mať (práve jednu) väzbu na konkrétny titul.
  - MÁEXEMPLÁR(*kniha*:(0,n), *exemplár*:(1,1))
    V knižnici môže byť aj nulový počet exemplárov jednej knihy – kniha je objednaná príp. rozkradnutá (resp. všetky exempláre sú vyradené)
  - MOSLIMSKÉMANŽELSTVO(*muž*:(0,4), *žena*:(0,1))
    Muž môže mať najviac 4 ženy, žena však najviac jedného muža.

2.2.5 Dekompozícia vzťahu M:N

- majme daný vzťah M:N – KINO-FILM

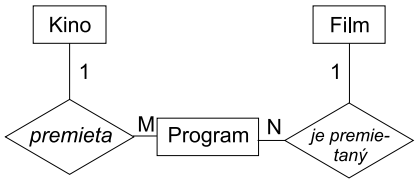


Obr. 10: Vzťah KINO-FILM ako vzťah M:N

- medzi entitné typy KINO, FILM vložíme nový entitný typ PROGRAM (tzv. *prienikový entitný typ*)

Obr. 11: Stav pred transformáciou

- vzťah M:N sa takto rozloží na 2 vzťahy kardinality 1:N
- prienikový entitný typ PROGRAM je existenčne závislý na oboch typoch
- spravidla je vo všeobecnosti závislý na oboch typoch aj identifikačne (jeho identifikačný kľúč je po dekompozícii tvorený dvojicou identifikátorov oboch základných typov a teda nemá vlastné kľúčové atribúty)
- alternatívou je vyrobenie umelého identifikačného kľúča



Obr. 12: Stav po transformácii

- niekedy sa robí na úrovni konceptuálneho modelu („na papieri“), niekedy až vo fáze (možno aj automatickej) transformácie entitno-relačnej schémy do databázovej schémy

2.3 Ďalšie črty konceptuálneho modelu

2.3.1 IS-A hierarchia

- študenti a učitelia majú niekoľko spoločných atribútov, ktoré možno zhrnúť do OSOBA(*rodné_číslo, meno, priezvisko, adresa*)

- niektoré vlastnosti majú špecifiká:
  u študenta: odbor, ročník, počet dosiahnutých kreditov
  u učiteľa: akademická hodnosť

- aj niektoré vzťahy môžu byť iné:
  u študenta: zapísal si u učiteľa: učí predmet, je členom katedry

- jedným extrémnym riešením je
  OSOBA(*rodné_č*, *meno*, *priezvisko*, *adresa*, *typ_osoby*, *odbor*, *ročník*, *počet_kreditov*, ..., *akad_hodnosť*, ...)
  pričom *typ_osoby* je študent resp. učiteľ. Potom však musíme pripustiť prázdne hodnoty pre tie atribúty, ktoré sú pre daný typ osoby irelevantné.

- druhým extrémnym riešením je zaviesť nezávislé entitné typy:
  ŠTUDENT(*rodné_č, meno, priezvisko, adresa|, odbor, ročník,...*)
  UČITEĽ(*rodné_č, meno, priezvisko, adresa|, akad_hodnosť,...*)
  Týmto sa však stratí možnosť mať pokope spoločné atribúty a výhody z toho vyplývajúce.

- kompromisné riešenie zavádza IS-A hierarchiu
  E: OSOBA(*rodné_č, meno, priezvisko, adresa,...*);
  ŠTUDENT(*odbor, ročník, počet_kreditov*) IS A OSOBA;
  UČITEĽ(*akad_hodnosť,...*) IS A OSOBA

- typ entity, ktorý je v IS-A hierarchii nižšie, dedí atribúty typu entity vyššie v hierarchii

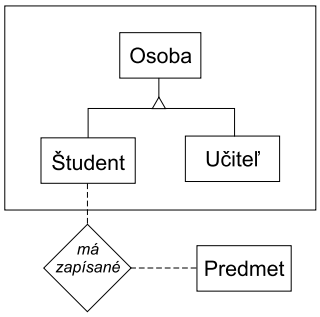
- IS-A vzťah je reflexívny a tranzitívny

- IS-A hierarchia z hľadiska teórie grafov

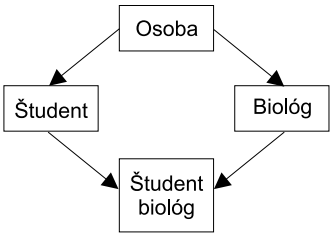
  – nemusí byť nutne strom, prípustné je aj viacnásobné dedenie

  – pre zjednodušenie sa zavádza podmienka, že existuje iba jeden uzol, z ktorého nevedie žiadna IS-A hrana, tzv. zdroj IS-A hierarchie. Táto podmienka je iba formálna – prinajhoršom sa urobí nový (umelý) uzol. V celej hierarchii sa potom môže ľahko dediť identifikačný kľúč zdroja.

  – IS-A hrany preto nemôžu tvoriť orientovaný cyklus



Obr. 13: Príklad IS-A hierarchie



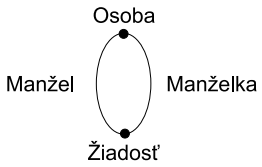
Obr. 14: Viacnásobné dedenie

- typ entity v IS-A hierarchii, ktorý nie je zdrojom, nie je identifikačne závislý na žiadnom entitnom type (mimo IS-A hierarchie), t.j. je identifikovateľný svojím zdrojom v IS-A hierarchii
- zdroj IS-A hierarchie má identifikačný kľúč, ostatné entitné typy v IS-A hierarchii ho nemajú (vlastný)
- mená atribútov musia byť jednoznačné aj vzhľadom na zdedenie
- prostriedky pre vznik IS-A hierarchie
  - *špecializácia* – definícia *podtypu*
    Príklad: PROFESOR je podtypom typu PRACOVNÍK VŠ
  - *generalizácia* – definícia *nadtypu*
    Príklad: PUBLIKÁCIA je nadtypom (generalizáciou) typu KNIHA, ČLÁNOK,...

2.3.2 Hierarchia identifikačných vzťahov

- analógia IS-A hierarchie
- ani identifikačné vzťahy nemôžu tvoriť orientovaný cyklus, ale tu sa podmienka jedinečnosti identifikačného zdroja nevyžaduje
- ak je identifikovaný slabý entitný typ E pre dva rôzne vzťahy ten istý, tak:
  (1) pre E existujú 2 rôzne identifikačné zdroje
  Príklad: ŽIADANKA by mohla byť identifikované pomocou identifikačných kľúčov typu PRACOVNÍK a SKLAD
  (2) alebo pre ten istý identifikačný zdroj vystupuje v dvoch rolách a tie nie je nutné rozlíšiť (označením)
  Príklad: ŽIADOSŤOMANŽELSKÚPÔŽIČKU je identifikovaný pomocou identifikátorov

oboch manželov, ktorí žiadosť podávajú (identifikačné vzťahy sú 2) a to k tomu istému silnému entitnému typu



Obr. 15: V žiadosti o manželskú pôžičku sa môžu vyskytovať dva identifikačné vzťahy

2.3.3 Ďalšie črty konceptuálneho modelu

Agregácia umožňuje vzťah medzi objektami nižšieho typu uvažovať ako objekt vyššieho typu
Príklad: dvojica typu (DÁTUM, FILM) môže tvoriť typ PROGRAM.

Zoskupovanie slúži na vytváranie podmnožín množiny objektov daného typu.
Príklad: nad typom PREDMET môžeme vytvoriť zoskupovaním typ PLÁNPREDMETOV, čo je trieda predmetov, ktoré možno študovať na škole v jednom odbore.

3 Relačný model dát

3.1 Základné pojmy

3.1.1 Relácia

- konflikt pojmov relácia v matematike a relácia (vzťah) v DB
- *relácia* z matematického hľadiska je množina prvkov tvare $(a_1, a_2, \ldots, a_n)$, kde n je *rád relácie*
- prvok $(a_1, a_2, \ldots, a_n)$ nazývame *n-tica* (*tica* $\equiv$ tuple (angl.)). Jeho zložka a_i je hodnota zodpovedajúca atribútu A_i príslušnej entity $R(A_1, \ldots, A_n)$.
- A_i je *meno atribútu*
- *doména* (označenie: dom(A_i)) je špecifikovaná množina hodnôt, ktoré môže atribút nadobúdať. Z matematického hľadiska je teda dom zobrazenie priraďujúce menu atribútu jeho doménu. Domény môžu byť pre rôzne atribúty aj rovnaké (t.j. zobrazenie nemusí byť prosté).
- reláciu pomenovanú R teda možno popísať ako

  $$R(A_1 : D_1, A_2 : D_2, \ldots, A_n : D_n); \qquad \text{pričom } D_i = \text{dom}(A_i)$$

  čím dostávame *schému relácie*.
- relácia R je potom podmnožina karteziánskeho súčinu

  $$D_1 \times D_2 \times \ldots \times D_n.$$

  Naviac táto podmnožina môže byť aj prázdna.
- *aktuálna doména* (označenie: adom(A)) je množina všetkých hodnôt atribútu A, ktoré sa v relácií vyskytnú (z matematického hľadiska ide o obor hodnôt).
  Príklad:

FILM(*meno*: CHAR(50), *herec*: CHAR(50), *rok*: INTEGER) s hodnotami

A_1 = meno	A_2 = herec	A_3 = rok
Statočné srdce	Gibson	1997
Forrest Gump	Hanks	1997
Zelená míľa	Hanks	2000
Winnetou	Brice	1963
Prázdniny v Ríme	Hepburnová	1965
Pán prsteňov	Wood	2002

3.1.2 Tabuľková terminológia

- relácia je tabuľka
- schéma relácie je záhlavie tabuľky
- tice sú riadky tabuľky,
- atribúty sú stĺpce tabuľky
- rozdiely:
  - relácie neobsahujú duplicitné riadky (sú to množiny), v tabuľkách sa vyskytovať môžu
  - v relácii nezáleží na poradí riadkov, v tabuľke je vždy dané nejaké poradie.

3.1.3 1. normálna forma

- znamená, že hodnoty atribútov sú nedeliteľné (atomické); je to základný predpoklad.
- Príklad: ak pri filme treba sledovať viac hercov, môžeme do tabuľky FILM vložiť riadok | Černí baroni | Vetchý, Landovský | 1994 |, stratí sa však atomickosť – dopyt na filmy, v ktorých hrá Landovský by viedol k spracovaniu reťazcov na časti a značne by sa skomplikoval – bol by za rámcom relačného modelu dát.

3.1.4 Projekcia

- ak $u = (a_1, a_2, \ldots, a_n) \in R(A_1, \ldots, A_n)$ a $\mathcal{B} = (A_{i_1}, \ldots, A_{i_k})$, tak

$$u[\mathcal{B}] = (a_{i_1}, \ldots, a_{i_k})$$

  je *projekcia* u na $\mathcal{B}$.
- pod $R[\mathcal{B}]$ budeme rozumieť množinu projekcií všetkých prvkov $u \in R$ na $\mathcal{B}$

$$R[\mathcal{B}] = \{u[\mathcal{B}] : u \in R\}$$

3.1.5 Kľúčové schémy

- *kľúč* $\mathcal{K}$ schémy $R(\mathcal{A})$ je minimálna (nezmenšiteľná) množina atribútov z $\mathcal{A}$, ktorých hodnoty budú jednoznačne určovať tice relácie R.
- prvky $u[\mathcal{K}]$ (je to projekcia) sa nazývajú *hodnoty kľúča* $\mathcal{K}$
- platí teda, že ak sú u, v dve rôzne tice z relácie $R(\mathcal{A})$, tak existuje aspoň 1 atribút $A_i \in \mathcal{K}$ (kľúčový) taký, že $u[A_i] \neq v[A_i]$.
- z podstaty relačného modelu dát vyplýva, že nás zaujímajú iba také schémy relácií, pre ktoré možno definovať kľúč (v najhoršom prípade celá množina atribútov). Zabezpečí sa tak neduplicita prvkov. V praxi tabuľka môže obsahovať aj duplicitné riadky, vtedy hovoríme, že ide o *kolekciu* tíc. Odstrániť ju možno pomocou klauzuly SQL `DISTINCT`.

- pretože kľúčov môže byť aj viac, vyberá sa jeden – *primárny kľúč* (primary key), ostatné sú *sekundárne*, resp. *alternatívne*.
- kľúč, ktorý pozostáva len z jedného atribútu sa nazýva *jednoduchý*, ostatné nazývame *zloženými*.
- v schéme budeme kľúčové atribúty podčiarkovať
- ak pridáme ku kľúču ľubovoľné nekľúčové atribúty, vlastnosť jednoznačnej identifikácie ostáva zachovaná, ale stráca sa minimálnosť. Výsledok je *nadkľúč* schémy $R(\mathcal{A})$.

3.1.6 Integritné obmedzenia

- zatiaľ sme sa zaoberali len štruktúrou dát, tu pôjde o zabezpečenie ich správnosti
- integritné obmedzenia sú ďalšie znalosti o dátach, ktoré musia dáta v databáze spĺňať
- je potrebné mať mechanizmy umožňujúce kontrolovať dáta a ich zmeny v databáze
- Príklad:
  V modeli
  KINO(*názov*, *adresa*);
  FILM(*meno*, *herec*, *rok*);
  PROGRAM(*názov kina*, *meno filmu*, *dátum*);
  môžu byť tieto integritné obmedzenia:
  - v kinách sa hrá najviac 2-krát za týždeň
  - jeden film sa hrá najviac v 3 kinách v meste
  - jeden film nemôžu dávať v jednom kine viackrát (v rámci jedného sledovaného obdobia) – vyplýva to z definície kľúča v relácii PROGRAM.

3.1.7 Referenčná integrita

- integritné obmedzenie popisujúce vzťah medzi dátami z dvoch relácií
- atribút, ktorého sa integritné obmedzenie týka, sa volá *cudzí kľúč* (foreign key). Ide o atribút (alebo skupinu atribútov), ktorého hodnota v každej tici je hodnotou primárneho kľúča inej relácie (alebo je prázdna, teda neobsahuje nič).
  Príklad: v predošlom prípade je v schéme PROGRAM atribút *názov kina* cudzím kľúčom k relácii KINO a atribút *názov filmu* cudzím kľúčom k relácii FILM.
- kontrola referenčnej integrity sa uplatní ako pri vkladaní do závislej relácie, tak aj pri vymazávaní z hlavnej relácie
- formálnejšie: ak je $\mathcal{K}$ kľúčom hlavnej relácie H a $\mathcal{C}$ cudzím kľúčom závislej relácie Z, tak

$$Z[\mathcal{C}] \subseteq H[\mathcal{K}](\cup\{\text{NULL}\})$$

3.1.8 Spôsoby vyjadrenia integritných obmedzení

- *procedurálne vyjadrenie integritných obmedzení* – v minulosti bolo možné len málo integritných obmedzení špecifikovať v definícii databázy, museli sa implementovať na úrovni aplikácie (programu), ktorá databázu používa
- *deklaratívna definícia integritných obmedzení* – v súčasnosti je možné väčšinu integritných obmedzení formulovať na úrovni databázy, starostlivosť o ne je presunutá na databázový stroj – pri pokuse o ich narušenie by malo dôjsť k chybovému hláseniu a automatickému zotaveniu systému.

- *kontrolné procedúry* tvoria samostatné programové moduly, ktoré sú uložené v systémovej časti databázy a vykonáva ich databázový server. Vytvárajú sa spravidla automaticky podľa definície integritného obmedzenia
- špeciálnymi kontrolnými procedúrami sú tzv. *triggery*, ktorá sa aktivujú niektorou spúšťacou udalosťou, obvykle úpravou niektorej databázovej tabuľky.

3.1.9 Špecifikácia databázy

- *relačná schéma* databázy je dvojica $(\mathbb{R}, \mathbb{I})$, kde $\mathbb{R}$ je množinová schéma, $\mathbb{I}$ je množina integritných obmedzení.
- niekedy je vhodné rozlišovať *lokálne* integritné obmedzenia týkajúce sa jednotlivých schém a *globálne* obmedzenia týkajúce sa väzbami medzi tabuľkami
- ak schémy relácií vyhovujú predpísaným integritným obmedzeniam, množina relácií je *konzistentná*.
- relácie v databázach možno upravovať, databáza sa tak môže meniť z jedného stavu do druhého, v každom stave by však mala byť konzistentná.

3.1.10 Manipulácia s dátami

- dopytovacie prostriedky (`SELECT` v SQL) nemenia stav databázy, len čítajú jej obsah.
- operácie na aktualizáciu
  - pridanie prvku do tabuľky (`INSERT`)
  - odstránenie prvku (`DELETE`)
  - úprava prvku v množine (`UPDATE`)
- pri manipulácii sa používa (na kontrolu a/alebo vyhľadávanie) kľúč
- v praxi existujú aj iné operácie – napr. dávkové naplnenie

3.2 Relačná algebra

- Coddov matematický pohľad na databázové štruktúry ako na relácie (v matematickom zmysle slova), od tejto chvíle môžu databázy používať aparát predikátovej logiky prvého rádu.
- *Relačná algebra* je množina operácií, ktorých aplikáciou na nejaké relácie (tabuľky) dostaneme ďalšie relácie (tabuľky)

3.2.1 Základné operácie

Keďže relácie sú množiny, ide o množinové operácie. V príkladoch budeme využívať tabuľky:

Chlapec

Meno	Priezvisko
Janko	Hraško
Juraj	Truľo
Ali	Baba

Dievča

Meno	Priezvisko
Ružena	Šípková
Ali	Baba

Súčin $\times$ (karteziánsky súčin)

súčin Chlapec $\times$ Dievča

Meno	Priezvisko	Meno	Priezvisko
Janko	Hraško	Ružena	Šípová
Janko	Hraško	Ali	Baba
Juraj	Truľo	Ružena	Šípová
Juraj	Truľo	Ali	Baba
Ali	Baba	Ružena	Šípová
Ali	Baba	Ali	Baba

Zjednotenie `UNION` a prienik `INTERSECT`

Chlapec $\cup$ Dievča

Meno	Priezvisko
Ján	Hraško
Juraj	Truľo
Ali	Baba
Ružena	Šípová

Chlapec $\cap$ Dievča

Meno	Priezvisko
Ali	Baba

Rozdiel $-, \setminus$ `EXCEPT`

Chlapec $\setminus$ Dievča

Meno	Priezvisko
Ján	Hraško
Juraj	Truľo

Dievča $\setminus$ Chlapec

Meno	Priezvisko
Ružena	Šípová

Na súčin sa nekladú žiadne obmedzenia, v ostatných prípadoch ($\cap, \cup, \setminus$) musia mať relácie kompatibilné operandy (mená, počet a domény atribútov musia byť rovnaké). V praxi sa zhodnosť mien atribútov nepožaduje.

Vytvorme si teraz ďalšiu tabuľku:

Pes

Meno
Dunčo
Rex

Pre túto tabuľku a predchádzajúcu tabuľku Chlapec existuje Chlapec $\times$ Pes, ale neexistuje Chlapec $\cup$ Pes, Chlapec $\cap$ Pes, Chlapec $\setminus$ Pes

Bodková notácia a operácia premenovania

- Pri súčine môže nastať konflikt mien (t.j. dva stĺpce sa volajú rovnako), v takom prípade použijeme *bodkovú notáciu*.
  - ak sú dané schémy $R(\mathcal{A}), S(\mathcal{B})$, tak $R \times S$ môže mať atribúty
    $$R.A_1, \ldots, R.A_n, S.B_1, \ldots, S.B_n$$
- Ak by aj potom nastal konflikt (napríklad ak násobíme tou istou tabuľkou, resp. $R = S$) použijeme operáciu *premenovania*
  - ak má relácia R schému $(A_1, \ldots, A_n)$, možno operáciu premenovania zadať zoznamom substitúcií
    $$R\langle A_{i_1} \to B_{i_1}, \ldots, A_{i_m} \to B_{i_m}\rangle$$
    alebo alternatívne
    $$R\langle (A_{i_1}, \ldots, A_{i_m}) \to (B_{i_1}, \ldots, B_{i_m})\rangle.$$

- Príklad:
  $\textsc{Chlapec}\langle(\textit{meno, priezvisko}) \to (\textit{meno1chlapca, priezvisko1chlapca})\rangle \times$
  $\times\ \textsc{Chlapec}\langle(\textit{meno, priezvisko}) \to (\textit{meno2chlapca, priezvisko2chlapca})\rangle$

3.2.2 Špecificky relačné operácie

Projekcia $R[\mathcal{B}]$ (v SQL zodpovedá `SELECT`)

- ak R je relácia so schémou $\mathcal{A}$ a $\mathcal{B}$ je postupnosť niektorých atribútov z $\mathcal{A}$, tak

$$R[\mathcal{B}] = \{u[\mathcal{B}] : u \in R\}$$

- pripomeňme, že ak $u = (a_1, \ldots, a_n)$, tak $u[\mathcal{B}] = (a_{i_1}, \ldots, a_{i_k})$, je projekcia u na $\mathcal{B}$, pričom $(a_{i_1}, \ldots, a_{i_k})$ je postupnosť hodnôt atribútov z $\mathcal{B}$.
- operácia vytvorí reláciu so schémou $\mathcal{B}$ a ticami, ktoré vzniknú z pôvodnej relácie výberom niektorých stĺpcov, prípadné duplicity sú odstránené.

Selekcia (reštrikcia) $R(\varphi)$ (zodpovedá `WHERE`)

- ak R je relácia so schémou $\mathcal{A}$ a φ je logická podmienka, tak

$$R(\varphi) = \{u \in R : \varphi(u)\}$$

- operácia vytvorí reláciu s tou istou schémou $\mathcal{A}$, ale ponechá len tie tice (riadky), ktoré spĺňajú podmienku φ.
- podmienka φ je zadaná logickým výrazom – pomocou logických spojok $\wedge, \vee, \neg$ z atomických formúl tvaru $t_1 \mathbin{@} t_2, t \mathbin{@} a$, kde $@$ je jedno z porovnaní, t_1, t_2, t sú mená atribútov, a je konštanta.
- $(t_1 \mathbin{@} t_2)(u)$ sa vyhodnocuje ako $u.t_1 \mathbin{@} u.t_2$
  $(t \mathbin{@} a)(u)$ sa vyhodnocuje ako $u.t \mathbin{@} a$
  logické spojky sa vyhodnotia podľa pravidiel Booleovej logiky
- Príklad:
  Vo výraze

$$\textsc{Chlapec}(\textit{meno} = \textit{priezvisko} \vee \textit{priezvisko} \neq \text{'Baba'})$$

  je relácia $R = \textsc{Chlapec}$ a podmienka φ je $(\textit{meno} = \textit{priezvisko} \vee \textit{priezvisko} \neq \text{'Baba'})$.
  Vyhodnotenie výrazu je potom nasledovné:

$$\begin{aligned}
&\varphi((\text{'Jan', 'Hrasko'})) \sim \\
&(\textit{meno} = \textit{priezvisko} \vee \textit{priezvisko} \neq \textit{'Baba'})(\text{'Jan', 'Hrasko'}) \sim \\
&(\underbrace{\textit{meno}}_{t_1} = \underbrace{\textit{priezvisko}}_{t_2})(\underbrace{\text{'Jan', 'Hrasko'}}_{u}) \vee (\underbrace{\textit{priezvisko}}_{t} \underbrace{\neq}_{@} \underbrace{\text{'Baba'}}_{a})(\underbrace{\text{'Jan', 'Hrasko'}}_{u}) \sim \\
&\Big((\text{'Jan', 'Hrasko'}).\textit{meno} = (\text{'Jan', 'Hrasko'}).\textit{priezvisko}\Big) \vee \\
&\vee\Big((\text{'Jan', 'Hrasko'}).\textit{meno} \neq \text{'Baba'}\Big) \sim \\
&\text{'Jan'} = \text{'Hrasko'} \vee \text{'Hrasko'} \neq \text{'Baba'} \sim \\
&\text{NEPRAVDA} \vee \text{PRAVDA} \sim \\
&\text{PRAVDA} \\
&\text{t.j.('Jan','Hrasko')} \in R(\varphi)
\end{aligned}$$

Spojenie $*$ (zodpovedá `JOIN, WHERE`)

- ak $R(\mathcal{A})$ a $S(\mathcal{B})$ sú relácie, tak

$$R * S = \{u : u[\mathcal{A}] \in R \wedge u[\mathcal{B}] \in S\}$$

- ak $\mathcal{A} = (a_1, \ldots, a_n)$, $\mathcal{B} = (b_1, \ldots, b_m)$, tak operácia vytvorí reláciu so schémou $(a_1, \ldots, a_n, b_{i_1}, \ldots, b_{i_k})$, kde $\{b_{i_1}, \ldots, b_{i_k}\} \subseteq \mathcal{H}(\mathcal{B}) \setminus \mathcal{H}(\mathcal{A})$ a $i_1 < i_2 < \ldots < i_k$
- prvky relácie teda vznikajú spojením tíc z oboch relácií cez rovnosť hodnôt na maximálnej množine spoločných atribútov (je to prirodzené spojenie podľa zásady „spoj vo všetkých kombináciách všetko, čo sa spojiť dá“)
- ak $\mathcal{H}(\mathcal{A}) \cap \mathcal{H}(\mathcal{B}) = \emptyset$ (t.j. neexistuje žiaden spoločný atribút), tak $R * S = R \times S$
- Príklad:
  Majme dané relácie:

MÁZapísané

Študent	Predmet
Hraško	Analýza
Hraško	Algebra
Polienko	Analýza
Baba	Algebra

Predmet

Predmet	Učiteľ
Analýza	Komenský
Algebra	Cantor

  Potom:

MÁZapísané $*$ Predmet

Študent	Predmet	Učiteľ
Hraško	Analýza	Komenský
Hraško	Algebra	Cantor
Polienko	Analýza	Komenský
Baba	Algebra	Cantor

Predmet $*$ MÁZapísané

Predmet	Učiteľ	Študent
Analýza	Komenský	Hraško
Analýza	Komenský	Polienko
Algebra	Cantor	Hraško
Algebra	Cantor	Baba

  Ak by v tabuľke Predmet bola uvedená položka | DBS | Vinař |, neobjavila by sa ani v tabuľke MÁZapísané $*$ Predmet ani v Predmet $*$ MÁZapísané.

Poznámka k operáciám
Uvedená množina operácií $\{*, [\mathcal{B}], (\varphi), \cap, \cup, \times, \setminus, \text{premenovanie}\}$ nie je minimálna

- $R \cap S = \Big((R \cup S) \setminus (R \setminus S)\Big) \setminus (S \setminus R)$
- operáciu $*$ možno definovať pomocou operácií premenovania, $\times, (\varphi), [\mathcal{B}]$

3.3 Dopytovací jazyk

3.3.1 Dopyty a dopytovací jazyk

- ak je $\mathbb{B}$ množina operácií, tak *dopytovací jazyk – relačná algebra* $RA_{\mathbb{B}}$ je množina výrazov, ktoré vzniknú skladaním relačných operácií z $\mathbb{B}$ nad tabuľkami danej databázy.
- prvok takéhoto jazyka sa nazýva *dopyt („dotaz“, query)*
- *odpoveď* na dopyt je relácia, ktorá vznikne vyhodnotením výrazu dopytu
- ak výrazy E_1, E_2 dávajú rovnakú odpoveď, sú *ekvivalentné*
- dopytovací jazyk, ktorý má prostriedky na realizáciu všetkých operácií relačnej algebry je *relačne úplný*
  - $RA_{\mathbb{B}}$ je relačne úplný

- všetky operácie možno skladať
  - schéma, akú bude mať výsledné zloženie týchto operácií je daná takouto funkciou sch (schéma relačného výrazu E):
    - ak $E = R$, kde R má schému $\mathcal{A}$, tak $\text{sch}(E) = \mathcal{A}$
    - ak $E = E_1 \cup E_2$, resp. $E = E_1 \cap E_2$, resp. $E = E_1 \setminus E_2$, tak
      $$\text{sch}(E) = \text{sch}(E_1)\,(= \text{sch}(E_2))$$
    - ak $E = E_1 \times E_2$ alebo $E = E_1 * E_2$, tak
      $$\text{sch}(E) = \text{sch}(E_1) \cup \text{sch}(E_2)$$
    - ak $E = F[\mathcal{X}]$, tak $\text{sch}(E) = \mathcal{X}$
    - ak $E = F(\varphi)$, tak $\text{sch}(E) = \text{sch}(F)$
    - ak $E = F\langle \mathcal{A} \to \mathcal{B} \rangle$, kde $\mathcal{A}$, $\mathcal{B}$ sú množiny atribútov rovnakej mohutnosti, tak
      $$\text{sch}(E) = (\text{sch}(F) \setminus \mathcal{A}) \cup \mathcal{B}$$
  - Príklad:
    Majme dané relácie:

ŠTUDENT

Meno	Adresa
Hraško	Hraškova 2
Šípová	Zámocká 4
Ali Baba	Serail 1001
Polienko	Drevená dedina 16

PREDMET

Názov	Učiteľ	Kredit
Databázové systémy	Vinař	2
Databázové systémy	Krajči	4
OOP	Eliáš	4
OOP	Marcinčák	2
Matematická logika	Bukovský	2
Fraktálna geometria	Bukovský	3
Teória informácií	Bukovský	4

MÁZAPÍSANÉ

Študent	Predmet	Hodnotenie
Hraško	OOP	A
Hraško	Matematická logika	B
Ali Baba	Fraktálna geometria	D
Ali Baba	OOP	C
Ali Baba	Matematická logika	B

  - postupne robíme operácie
    - $R_1 = \text{MÁZAPÍSANÉ}(\textit{študent} = \text{'Ali Baba'})$ – selekcia

Študent	Predmet	Hodnotenie
Ali Baba	Fraktálna geometria	D
Ali Baba	OOP	C
Ali Baba	Matematická logika	B

    - $R_2 = R_1[\textit{predmet, hodnotenie}]$ – projekcia

Predmet	Hodnotenie
Fraktálna geometria	D
OOP	C
Matematická logika	B

    - $R_3 = R_2\langle \textit{predmet} \to \textit{názov} \rangle$ – premenovanie

Názov	Hodnotenie
Fraktálna geometria	D
OOP	C
Matematická logika	B

▷ $R_4 = R_3 * \text{PREDMET}$

Predmet	Hodnotenie	Učiteľ	Kredit
Fraktálna geometria	D	Bukovský	3
OOP	C	Eliáš	4
OOP	C	Marcinčák	2
Matematická logika	B	Bukovský	2

▷ $R_5 = R_4[učiteľ]$

Učiteľ
Bukovský
Eliáš
Marcinčák
Bukovský

- v tomto príklade sme vlastne riešili dopyt „Nájdite všetkých učiteľov, ktorí učia Aliho Babu“ a to pomocou nasledovného výrazu (bez medzivýsledkov):

  $$\text{MÁZAPÍSANÉ}(\check{s}tudent = \text{'Ali Baba'})[predmet, hodnotenie]\langle predmet \to názov\rangle * \text{PREDMET}[učiteľ]$$

- atribút *hodnotenie* nie je pre celkový výsledok podstatný, mohli by sme ho teda vynechať; dostali by sme ekvivalentný výraz, ktorého vyhodnocovanie by bolo efektívnejšie.

- dopyty, ktoré využívajú len operácie spojenia, projekcie, selekcie a premenovania atribútu (teda nie $\times, \setminus, \cap, \cup$) nazývame *jednoduché dopyty*, v praxi je ich drvivá väčšina.

- plán vyhodnotenia dopytu
  - v systéme riadenia relačnej databázy existuje tzv. *jadro relačného stroja* – skupina programov schopná vyhodnotiť vlastnosti relačných operácií, ktoré pracujú pomocou rôznych algoritmov
  - algoritmy využívajú zotriedenie relácií, indexy, štatistiky (napr. veľkosť relácie, počet rôznych hodnôt daného atribútu), optimalizátor pomocou týchto metainformácií rozhodne, ktorý z algoritmov je na vyhodnotenie daného dopytu optimálny, príp. sám nahradí neefektívne napísaný výraz efektívnejším.

3.3.2 Príklady na vytváranie výrazov

Uvedieme ďalšie príklady na vytváranie výrazov (využívame tabuľky z predchádzajúceho príkladu)

- Úloha: treba nájsť všetkých študentov, ktorí majú zapísané okrem iných všetky predmety Bukovského.
  - $P = \text{MÁZAPÍSANÉ}[\check{s}tudent]$
  - $Q = \text{PREDMET}(učiteľ = \text{'Bukovský'})[názov]\langle názov \to predmet\rangle$ (formálne premenujeme názov na predmet)

  Relácia P: ŠTUDENT

Študent
Hraško
Baba

  Relácia Q: PREDMET

Predmet
Matematická logika
Fraktálna geometria
Teória informácií

  V relácii P sú študenti, ktorí majú zapísaný aspoň jeden predmet.
  V relácii Q sú predmety, ktoré vyučuje Bukovský.

– $R = P \times Q$
Relácia $R = P \times Q$:

Študent	Predmet
Hraško	Matematická logika
Hraško	Fraktálna geometria
Hraško	Teória informácií
Ali Baba	Matematická logika
Ali Baba	Fraktálna geometria
Ali Baba	Teória informácií

Zoznam všetkých kombinácií študentov, ktorí majú zapísaný aspoň jeden predmet „hocikoho“ a predmetov, ktoré vyučuje Bukovský (bez ohľadu na to, či ich má študent zapísané alebo nie).

– $S = R \setminus \text{MÁZAPÍSANÉ}[\check{s}tudent, predmet]$
Relácia S:

Študent	Predmet
Hraško	Matematická logika
Hraško	Fraktálna geometria
Ali Baba	Matematická logika
Ali Baba	Fraktálna geometria

Dvojice študentov a predmetov, ktoré učí Bukovský, ale študent ich nemá zapísané.

– $T = S[\check{s}tudent]$
Relácia T:

Študent
Hraško
Baba

Študenti, ktorým chýba čo i len jeden predmet Bukovského, ale zároveň majú čosi zapísané.

– $U = P \setminus T$
Relácia U:

Študent
—

Zoznam študentov, ktorí majú zapísané všetky predmety Bukovského.
Záver:
$U = \text{MÁZAPÍSANÉ}[\check{s}tudent] \setminus$
$\Big(\Big(\text{MÁZAPÍSANÉ}[\check{s}tudent] \times$
$\times \text{PREDMET}(u\check{c}itel' = \text{'Bukovský'})[n\acute{a}zov]\langle n\acute{a}zov \to predmet\rangle\Big) \setminus$
$\setminus \text{MÁZAPÍSANÉ}[\check{s}tudent, predmet]\Big)[\check{s}tudent]$

- Predstavme si, že by si Ali Baba zapísal aj Teóriu informácií. Potom:
$T = S[\check{s}tudent]$:

Priezvisko študenta
Hraško

$U = P \setminus T$:

Priezvisko študenta
Ali Baba

- Úloha: treba nájsť všetkých študentov, ktorý nemajú zapísaný žiaden predmet Bukovského.
Riešenie: Od všetkých študentov odčítame takých, ktorí majú zapísaný aspoň jeden predmet Bukovského.
$\text{ŠTUDENT}[meno]\langle meno \to \check{s}tudent\rangle \setminus$

$\Big($PREDMET($učiteľ = $'Bukovský')[$*názov*$]\langle$*názov* $\rightarrow$ *predmet*$\rangle *$
MÁZAPÍSANÉ[*študent*, *predmet*]$\Big)$
Všimnime si, že výsledok obsahuje aj mená študentov, ktorí nemajú nič zapísané (*flákačov*): Ružena Šípová, Ján Polienko.

3.3.3 Ďalšie operácie relačnej algebry

@-spojenie ($R[t_1@t_2]S$, kde $@ \in \{<, >, =, \geq, \leq, \neq\}$)

- ak $R(\mathcal{A})$ a $S(\mathcal{B})$ sú relácie, pričom $\mathcal{H}(\mathcal{A}) \cap \mathcal{H}(\mathcal{B}) = \emptyset$, tak

$$R[a_i@b_j]S = \{u : u[\mathcal{A}] \in R \wedge u[\mathcal{B}] \in S \wedge u.a_i@u.b_j\}$$

- Operácia vytvorí reláciu, ktorá je podmnožinou karteziánskeho súčinu $R \times S$ a obsahuje tie tice u, ktoré spĺňajú podmienku $u.a_i@u.b_j$; je to zloženie karteziánskeho súčinu a selekcie.
- Príklad:

PRVÝCHLAPEC

Poradie 1. chlapca	Meno 1. chlapca
1	Ali Baba
2	Ján Hraško
3	Ján Polienko

DRUHÝCHLAPEC

Poradie 2. chlapca	Meno 2. chlapca
1	Ali Baba
2	Ján Hraško
3	Ján Polienko

Spojenie
PRVÝCHLAPEC[*Poradie1Chlapca* $<$ *Poradie2Chlapca*]DRUHÝCHLAPEC:

Por. 1. chlapca	Meno 1. chlapca	Por. 2. chlapca	Meno 2. chlapca
1	Ali Baba	2	Ján Hraško
1	Ali Baba	3	Ján Polienko
2	Ján Hraško	3	Ján Polienko

- Najčastejšie sa používa $=$-spojenie.

@-polospojenie, kde $@ \in \{<, >, =, \geq, \leq, \neq\}$

- ľavé @-polospojenie $R\langle t_1@t_2]S$
  - $R\langle t_1@t_2]S = (R[t_1@t_2]S)[\mathcal{A}]$, kde $\mathcal{A}$ je schéma R.
  - operácia vytvorí reláciu, ktorá je podmnožinou R a obsahuje tie tice u, ktoré sú @-spojiteľné aspoň s 1 prvkom z S.
  - Príklad:
    Budeme využívať tabuľky PRVÝCHLAPEC, DRUHÝCHLAPEC.
    PRVÝCHLAPEC$\langle$*Poradie1Chlapca* $<$ *Poradie2Chlapca*]DRUHÝCHLAPEC

Poradie 1. chlapca	Meno 1. chlapca
1	Ali Baba
2	Ján Hraško

- pravé @-polospojenie $R[t_1@t_2\rangle S$
  - $R[t_1@t_2\rangle S = (R[t_1@t_2]S)[\mathcal{B}]$, pričom $\mathcal{B}$ je schéma S.
- pri zrejmom smere sa hovorí len o @-spojení.
- najčastejšou možnosťou je $=$, vtedy ide o polospojenie cez rovnosť; v prípade, že ide o rovnosť všetkých zodpovedajúcich atribútov, hovoríme o *prirodzenom* polospojení.

Ľavé polospojenie označujeme $R\langle^* S$, pravé $R^*\rangle S$.
Ľavé ani pravé @-polospojenie nie je komutatívne, t.j. $R\langle^* S$ a $R^*\rangle S$ nie sú ekvivalentné.

Delenie ($R \div S$)

- ak $R(\mathcal{A})$ a $S(\mathcal{B})$ sú relácie a $\mathcal{H}(\mathcal{B}) \subseteq \mathcal{H}(\mathcal{A})$, tak

$$R \div S = \{t : (\forall s \in S) : t \oplus s \in R\},$$

  pričom operácia $\oplus$ označuje *zreťazenie*. (Príklad: $(a, b) \oplus (c, d, e) = (a, b, c, d, e)$)

- ak je $R = T \times S$, tak $T = R \div S$
- je to teda inverzná operácia ku karteziánskemu súčinu (pre disjunktné atribúty)
- delenie sa dá odvodiť:

$$R \div S = R[\mathcal{A} \ominus \mathcal{B}] \setminus \Big((R[\mathcal{A} \ominus \mathcal{B}] \times S) \setminus R\Big)[\mathcal{A} \ominus \mathcal{B}]$$

- Príklad:
  Nech $\mathcal{A} = (a, b), \mathcal{B} = (b), \mathcal{A} \ominus \mathcal{B} = (a)$

R

a	b
3	1
3	2
4	1
4	2
5	1

S

b
1
2

$R \div S$

a
3
4

$R[\mathcal{A} \ominus \mathcal{B}]$

a
3
4
5

$R[\mathcal{A} \ominus \mathcal{B}] \times S$

a	b
3	1
3	2
4	1
4	2
5	1
5	2

$(R[\mathcal{A} \ominus \mathcal{B}] \times S) \setminus R$

a	b
5	2

$\Big((R[\mathcal{A} \ominus \mathcal{B}] \times S) \setminus R\Big)[\mathcal{A} \ominus \mathcal{B}]$

a
5

$R[\mathcal{A} \ominus \mathcal{B}] \setminus \Big((R[\mathcal{A} \ominus \mathcal{B}] \times S) \setminus R\Big)[\mathcal{A} \ominus \mathcal{B}]$

a
3
4

- nájdime iný výraz vyjadrujúci všetkých študentov, ktorí majú zapísané všetky predmety Bukovského:

$$\begin{aligned}&\text{MÁZAPÍSANÉ}[\textit{študent, predmet}] \div \\ &\text{PREDMET}(\textit{učiteľ} = \text{'Bukovský'})[\textit{názov}]\langle \textit{názov} \to \textit{predmet} \rangle\end{aligned}$$

- podľa definície obsahuje výsledok takých študentov $š$, že pre každý predmet p učený Bukovským existuje riadok m z relácie MÁZAPÍSANÉ taký, že

$$m[\textit{študent}] = š, m[\textit{predmet}] = p$$

3.3.4 Operácie za relačným modelom dát

- prázdna hodnota (NULL)
  - patrí do všetkých domén
  - označuje, že hodnota chýba, alebo je neznáma alebo neaplikovateľná
  - ak je použitá v logických operáciách, dostávame logickú hodnotu UNKNOWN (prechod do trojhodnotovej logiky)
- vonkajšie spojenie ($R *_F S$) – z angl. *full outer join*

– ľavé (pravé) vonkajšie spojenie ($R *_L S$, $R *_R S$) – *left* resp. *right outer join*

$$R *_L S = (R * S) \cup (R_n \times \{\underbrace{\text{NULL}, \ldots, \text{NULL}}_{|\mathcal{B}|\text{-krát}}\})$$

$$R *_R S = (R * S) \cup (\{\underbrace{\text{NULL}, \ldots, \text{NULL}}_{|\mathcal{A}|\text{-krát}}\} \times S_n)$$

$$R *_F S = (R *_L S) \cup (R *_R S)$$

– R_n, S_n obsahujú tice nespojiteľné s S, resp. s R.

– na rozdiel od vonkajších spojení sa normálnemu spojeniu hovorí *vnútorné* (inner join)

Príklad:

ŠTUDENT

Meno	Číslo izby
Ali Baba	120A
Ružena Šípová	220B
Ján Polienko	NULL

IZBA

Číslo izby	Typ
120A	dvojka
220B	trojka
1X	jednotka

ŠTUDENT * IZBA (`JOIN .. ON`)

Meno	Číslo izby	Typ
Ali Baba	120A	dvojka
Ružena Šípová	220B	trojka

ŠTUDENT $*_L$ IZBA (`LEFT OUTER JOIN .. ON`)

Meno	Číslo izby	Typ
Ali Baba	120A	dvojka
Ružena Šípová	220B	trojka
Ján Polienko	NULL	NULL

ŠTUDENT $*_R$ IZBA (`RIGHT OUTER JOIN .. ON`)

Meno	Číslo izby	Typ
Ali Baba	120A	dvojka
Ružena Šípová	220B	trojka
NULL	1X	jednotka

ŠTUDENT $*_F$ IZBA (`FULL OUTER JOIN .. ON`)

Meno	Číslo izby	Typ
Ali Baba	120A	dvojka
Ružena Šípová	220B	trojka
Ján Polienko	NULL	NULL
NULL	1X	jednotka

3.4 Relačný kalkulus

- ticový relačný kalkulus
  - vyskytol sa v Coddovej definícii relačného modelu dát.
  - poskytuje veľmi silný dopytovací prostriedok porovnateľný s relačnou algebrou
  - používa prirodzený pohľad na reláciu ako na množinu dát
- doménový relačný kalkulus
  - objavil sa neskôr
  - pracuje s hodnotami jednotlivých tíc
  - je to vhodný teoretický základ pre dopytovací jazyk QBE (*Query by Example*) známy napr. zo systému PARADOX.

- oba jazyky úzko súvisia s predikátovým počtom 1. rádu
- zameriame sa na doménový relačný kalkulus

3.4.1 Základné prvky doménového relačného kalkulu

Termy

- doménové premenné – reťazce znakov; zodpovedajú atribútom. Dosadzujú sa za ne hodnoty z prislúchajúcich domén.
- konštanty
  - nemusia sa vyskytovať v databáze
  - Príklad: $plat < 100000$ vyjadruje hodnoty premennej *plat*, ktoré sú menšie ako 100000, pričom konštanta 100000 sa nemusí v databáze vyskytovať
- na rozdiel od predikátového počtu vo všeobecnosti už nemá iné termy
- na odlíšenie reťazcových konštánt od premenných ich budeme uzatvárať do apostrofov

Predikátové symboly

- mená relácií z databázy
- binárne porovnávacie predikáty z množiny $\{=, >, <, \leq, \geq, \neq\}$

Formuly

- atomické formuly – zodpovedajú predikátovým symbolom
  - $R(t_1, \ldots, t_n)$, R je predikátovým symbolom
    Príklad: ŠTUDENT(*meno*, 'Baba', *adresa*)
  - $t_1 @ t_2$, kde $@ \in \{=, >, <, \leq, \geq, \neq\}$, t_1, t_2 sú termy
    Príklad: *priezvisko* = 'Baba'
- skladanie formúl
  - pomocou logických spojok $(\wedge, \vee, \rightarrow, \neg)$
    - ak F_1, a F_2 sú formuly, tak $F_1 \wedge F_2, F_1 \vee F_2, F_1 \rightarrow F_2$ sú formuly
    - ak F je formula, tak $\neg F$ je formula
  - pomocou kvantifikátorov $\forall, \exists$
    - ak $F(x)$ je formula, tak aj $(\forall x)F(x); (\exists x)F(x)$ sú formuly
- univerzum (univerzálna doména)
  - množina hodnôt, ktoré možno dosadzovať za premenné
  - je to napr. zjednotenie všetkých domén použitých v schéme databázy
  - prakticky je to však zjednotenie aktuálnych domén databázy a konštánt, ktoré sa vyskytujú v tom-ktorom dopyte
- interpretácia formúl
  - napr. keď vo formule ŠTUDENT(x, y) ohodnotíme premenné x a y, t.j. priradíme im konkrétne hodnoty a, b, tak táto formula má ohodnotenie TRUE, ak dvojica (a, b) patrí do relácie ŠTUDENT, a ohodnotenie FALSE, ak $(a, b) \notin$ ŠTUDENT
  - zložené formuly sú ohodnotené v súlade so zmyslom logických spojok

  - vyhodnocovanie formúl $(\exists)F(x)$ a $(\forall x)F(x)$ si možno predstaviť ako mechanizmus dosadzujúci za x relevantné konštanty k a vyhodnocujúci platnosť formuly $F(k)$
    - Príklad:
      V prípade formuly $(\exists \textit{učiteľ})\textsc{Predmet}(\text{'OOP'}, \textit{učiteľ}, 4)$ by vyhodnotil formuly:

$$\begin{aligned}
\textsc{Predmet}(\text{'OOP'}, \text{'Vinař'}, 4) &\;--\; \text{FALSE}\\
\textsc{Predmet}(\text{'OOP'}, \text{'Krajči'}, 4) &\;--\; \text{FALSE}\\
\textsc{Predmet}(\text{'OOP'}, \text{'Eliáš'}, 4) &\;--\; \text{TRUE}\\
\textsc{Predmet}(\text{'OOP'}, \text{'Marcinčák'}, 4) &\;--\; \text{FALSE}\\
\textsc{Predmet}(\text{'OOP'}, \text{'Bukovský'}, 4) &\;--\; \text{FALSE}
\end{aligned}$$

      takže odpoveď by bola {'Eliáš'}
    - tento mechanizmus môžeme použiť na formuláciu dopytov
      - ▷ Príklad: Výraz $\{(x, y) : \textsc{Student}(x, y)\}$ znamená dopyt „nájdi všetky mená a adresy všetkých študentov“
      - ▷ Príklad: Výraz $\{\textit{učiteľ}, \textit{kredit} : \textsc{Predmet}(\text{'OOP'}, \textit{učiteľ}, \textit{kredit})\}$ znamená dopyt „ktorí učitelia a za koľko kreditov učia OOP“

- typy výskytov premennej
  - *viazaný výskyt* – ak je v dosahu príslušného kvantifikátora
    Príklad: premenná x má vo formule $(\exists x)F(x)$ viazaný výskyt.
  - *voľný výskyt* – v inom prípade

  Budeme sa vyhýbať formulám typu $G(x) \wedge (\exists x)F(x)$ v ktorých má x aj voľný aj viazaný výskyt – časť $(\exists x)F(x)$ jednoducho ekvivalentne nahradíme $(\exists y)F(y)$. Na názve viazanej premennej nezáleží, označuje len väzbu medzi miestami jej výskytu. Môžeme preto povedať, že každá premenná vo formule je buď viazaná alebo voľná.

3.4.2 Sprehľadnenia a zjednodušenia zápisov

- aby bolo jasné, ktorá hodnota zodpovedá ktorému atribútu, budeme používať notáciu typu

$$\{(x, y) : \textsc{Student}(\textit{meno} : x, \textit{adresa} : y)\}$$

- ak sa vo formule dopytu vyskytne $(\exists x)R(\ldots, A : x, \ldots)$ a x sa už viackrát vo formule nevyskytne, budeme $(\exists x)$ a $A : x$ vynechávať.

- Príklad:
  Formulu

$$\{u : (\exists k)\textsc{Predmet}(\textit{názov} : \text{'OOP'}, \textit{učiteľ} : u, \textit{kredit} : k)\}$$

  nahradíme formulou

$$\{u : \textsc{Predmet}(\textit{názov} : \text{'OOP'}, \textit{učiteľ} : u)\}$$

  (Môžeme sa riadiť pravidlom: „Čo tam nie je, považujeme za existenčný kvantifikátor.“)

3.4.3 Vzťah relačného kalkulu k relačnej algebre

- Príklad:
  Formula

$$\{u : \textsc{Predmet}(\textit{názov} : \text{'OOP'}, \textit{učiteľ} : u)\}$$

  v relačnej algebre znamená

$$\textsc{Predmet}(\textit{názov} = \text{'OOP'})[\textit{učiteľ}]$$

- Príklad:
  Formula

$$\{(š, a, p) : \text{MÁZAPÍSANÉ}(\textit{študent} : š, \textit{predmet} : p) \wedge \text{ŠTUDENT}(\textit{meno} : š, \textit{adresa} : a)\}$$

  v relačnej algebre znamená

$$\Big(\text{MÁZAPÍSANÉ} * (\text{ŠTUDENT}\langle \textit{meno} \rightarrow \textit{študent}\rangle)\Big)[\textit{študent, adresa, predmet}]$$

3.4.4 Problémy s vyhodnocovaním formúl doménového relačného kalkulu

- v doménovom relačnom kalkule možno formulovať výrazy, ktoré pri klasickom matematickom spôsobe vyhodnocovania vedú pri (v princípe) nekonečnej doméne (napr. INTEGER) k nekonečným reláciám alebo k nekonečnému počtu vyhodnocovaných formúl
  - Príklad: aspoň jedna z relácií $\{x : R(x)\}$ a $\{x : \neg R(x)\}$ je nekonečná.
  - Príklad: pri vyhodnocovaní formuly $(\exists x)F(x)$ resp. $(\forall x)F(x)$ sa za x dosadzuje nekonečné množstvo konštánt.
  - Príklad: ak je v $\{(x, y) : R(x) \vee S(y)\}$ formula $R(x)$ ohodnotená TRUE, za y možno dosadiť čokoľvek a dvojica (x, y) bude zreteľne vyhovovať.
- riešením takýchto prípadov je obmedziť pojem relačného kalkulu
  - *Ullmanov syntaktický variant* doménového relačného kalkulu zavádza pojem tzv. *bezpečných formúl*, ktorých všetky voľné premenné sú ohraničené.
  - vyhodnocovací algoritmus pre kvantifikátory obmedzíme na hodnoty z aktuálnych domén jednotlivých atribútov – hovoríme o *obmedzenej interpretácii*
- Takto upravený model relačný dopytovací prostriedok je relačne úplný[1]. Znamená to, že relačná algebra je dostatočne silná, lebo zodpovedá prostriedkom logiky 1. rádu.

3.4.5 Rozšírenia relačných kalkulov

ÁNO/NIE dopyty

- ak formula dopytu neobsahuje voľné premenné, testujeme vlastne, či nadobúda hodnotu FALSE (prázdna množina) alebo TRUE (neprázdna množina)

Agregačné funkcie

- relačnou algebrou nemôžeme zistiť, koľko predmetov učí konkrétny učiteľ, potrebovali by sme funkciu, ktorá priradí množine jej mohutnosť.
- prax ukazuje potrebu takých agregačných funkcií, ktoré sa aplikujú na reláciu vzniknutú ako poddopyt.

COUNT	počet prvkov
SUM_A	súčet hodnôt (číselného) atribútu A v danej relácii
MIN_A	minimum z hodnôt atribútu A.
MAX_A	maximum z hodnôt atribútu A.
AVG_A	(aritmetický) priemer z hodnôt atribútu A.

- relačný kalkulus môžeme rozšíriť tak, že medzi termy kalkulu zaradíme okrem jednoduchých premenných a konštánt výrazy tvaru

$$\text{AGREGAČNÁFUNKCIA}(\textit{výraz\_relačného\_kalkulu, atribút})$$

[1] Dôsledok: „Ullman nebol blbý.“

a následne aj príslušné z nich zložené termy.

- Príklad:
  Výraz

$$\{(\check{s}, počet) : počet = \text{COUNT}(p : \text{MÁZAPÍSANÉ}(študent : \check{s}, predmet : p))\}$$

  vytvára tabuľku študentov spolu s počtom zapísaných predmetov.

Tranzitívny uzáver relácie

- niektoré typy údajov sú tranzitívne, t.j. ak je A vo vzťahu k B a B je vo vzťahu k C, tak A je vo vzťahu k C.
  Príklad: hierarchia „nadriadený-podriadený“, „predok-potomok“

- reprezentovať takéto vzťahy v databáze možno jednoducho pomocou binárnej relácie

$$R(A, B), \quad \text{kde } \text{dom}(A) = \text{dom}(B)$$

  pričom stačí, aby v R boli iba bezprostredné vzťahy (najbližší nadriadený, rodič a pod.)

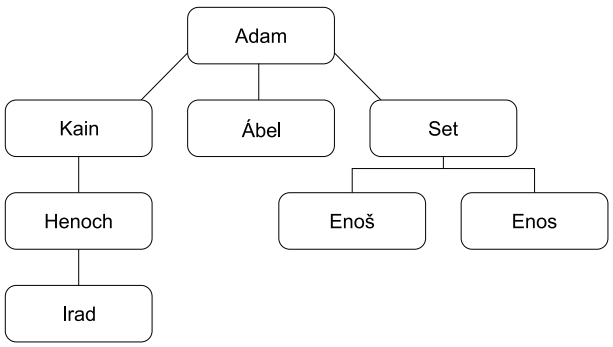
- výrazy definujúce úrovne potom budú

$$\begin{aligned} R_0 &= R(A, B) \\ R_1 &= \Big(R_0[B = A'](R\langle A \to A', B \to B'\rangle)\Big)[A, B']\langle B' \to B\rangle \\ R_2 &= \Big(R_1[B = A'](R\langle A \to A', B \to B'\rangle)\Big)[A, B']\langle B' \to B\rangle \\ &\vdots \end{aligned}$$

- výraz pre *tranzitívny uzáver* (zjednotenie všetkých dvojíc predkov a potomkov) je potom zjednotenie týchto úrovní.

$$\text{TU} = \bigcup_{i=1}^{\infty} R_i$$

  Vo všeobecnosti sa však nedá zostrojiť, lebo nie je známy počet úrovní (je obsiahnutý v databáze (implicitne)).



Obr. 16: Rodokmeň Adama

- Príklad:

R

otec	syn
Adam	Kain
Adam	Ábel
Adam	Set
Kain	Henoch
Henoch	Irad
Set	Enos
Set	Enoš[2]

$R_0[B = A']R\langle A, B \to A', B'\rangle$

predok	predok = potomok	potomok
Adam	Kain	Henoch
Adam	Ábel	
Adam	Set	Enos
Adam	Set	Enoš
Kain	Henoch	Irad
Henoch	Irad	
Set	Enos	
Set	Enoš	

R_1

predok	potomok
Adam	Henoch
Adam	Enos
Adam	Enoš
Kain	Irad

$R_1[B = A']R\langle A, B \to A', B'\rangle$

predok	predok = potomok	potomok
Adam	Henoch	Irad

R_2

predok	potomok
Adam	Irad

$R_3 = \emptyset, R_4 = \emptyset, R_5 = \emptyset, \ldots$

[2] „Keď mal Set stopäť rokov, narodil sa mu Enos“ (Gn 5:6, SÚ 1995). „A Set žil sto päť rokov a splodil Enoša“ (Gn 5:6, preklad Roháček.) Kvôli tomu je v relácii umiestnená táto položka.