Navrhovanie databáz

Ján Mazák

FMFI UK Bratislava

Návrh databázových schém

Databázová schéma: čo sú atribúty a aký majú dátový typ, aké relácie máme, vzťahy medzi nimi.

Pri návrhu databázovej schémy (voľne databázy) sa najmä snažíme vyhnúť známym nedostatkom. Neraz ide o kompromis medzi blízkosťou k ideálnemu návrhu a výkonom v praxi.

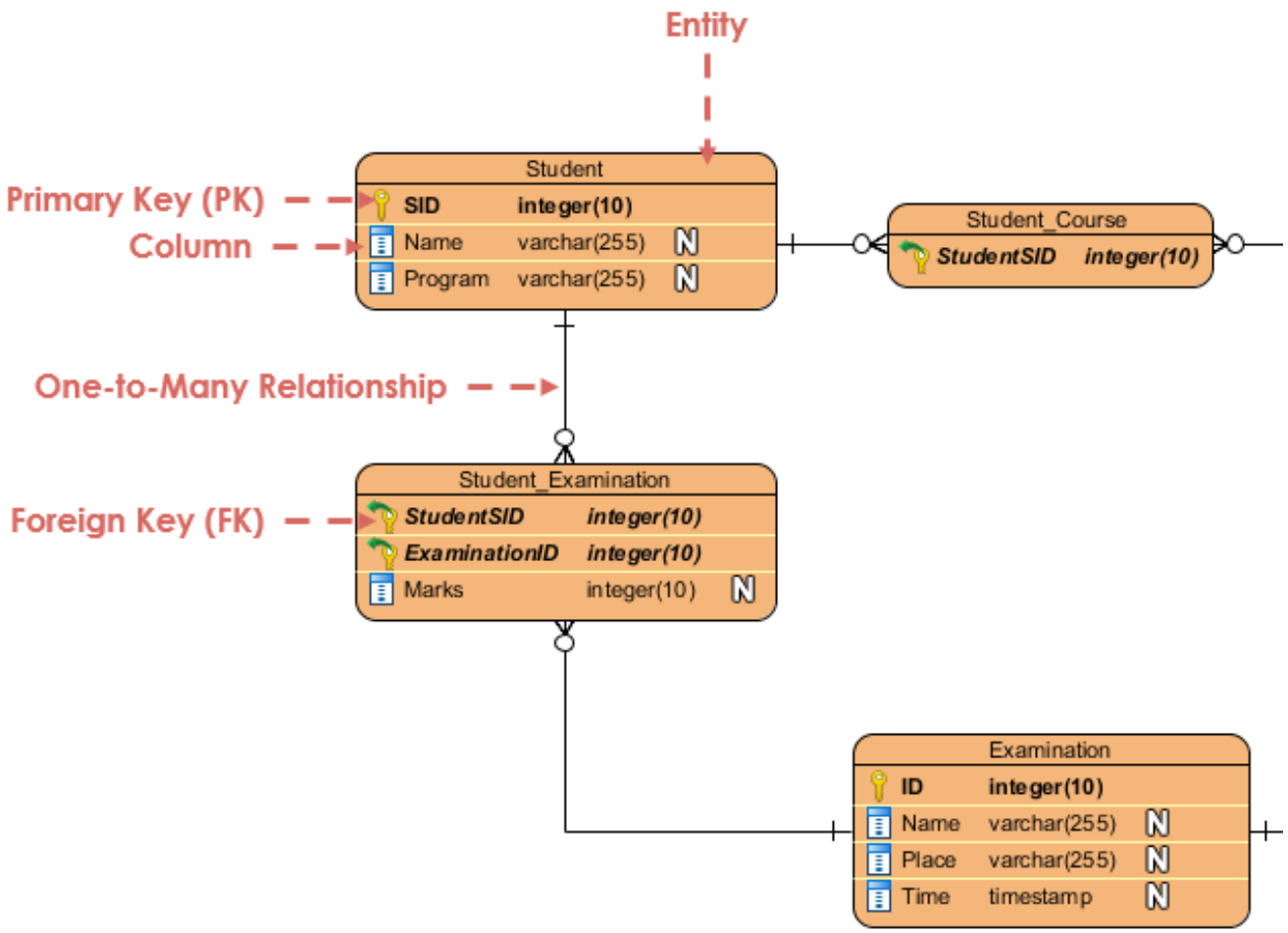
Relačný model, jazyk SQL a bežné DBMS poskytujú dostatok prostriedkov na zabezpečenie integrity dát.

Entitno-relačný model

Pri návrhu vychádzame zo zjednodušeného modelu sveta — *dátový model*. Tieto modely sa bežne kreslia v podobe entitno-relačných diagramov (ERM/ERD).

- entitám v tomto modeli zodpovedajú tabuľky, ich atribútom stĺpce
- vzťahy 1:N reprezentujeme atribútmi
- vzťahy M:N zvyčajne reprezentujeme tabuľkami (atribút, ktorého dátovým typom je pole, je takmer vždy v rozpore s dobrou praxou)

Entitno-relačný model



Nedostatky v návrhu

- zlý či neaktuálny dátový model
  (bežné pri digitalizácii verejnej správy)
- nejasnosti v tom, aké dáta sú potrebné a či je možné ich získať
- schéma, ktorú je ťažké upraviť pri zmene dátového modelu (napr. kvôli nadmernému využitiu custom riešení miesto štandardizovaných; voľte jednoduchosť)
- zlé pomenovania (nejasné, málo špecifické, nekonzistentné, opakovanie názvu pre rôzne veci)
- nadmerné použitie indexov (netreba ich „pre každý prípad“ pre každý atribút)
- chýbajúca dokumentácia, najmä zdôvodnenia netriviálnych rozhodnutí

Nedostatky v návrhu — redundancia

Zamestnanec	*Pracovisko*	*Adresa pracoviska*
Adam	SAV	Patrónka
Adam	KAGDM	Mlynská dolina
Cyril	KAGDM	Mlynská dolina

Redundancy: Informácia o adrese pracoviska je v tabuľke viacnásobne.

Nedostatky v návrhu — anomálie pri updatovaní

Zamestnanec	*Pracovisko*	*Adresa pracoviska*
Adam	SAV	Patrónka
Adam	KAGDM	Mlynská dolina → Staré grunty
Cyril	KAGDM	Mlynská dolina

UPDATE anomaly: keď upravíme adresu pracoviska v jednom zázname, bude to v rozpore s ostatnými záznamami.

Nedostatky v návrhu — anomálie pri mazaní

Zamestnanec	*Pracovisko*	*Adresa pracoviska*
Adam	SAV	Patrónka
Adam	KAGDM	Mlynská dolina
Cyril	KAGDM	Mlynská dolina

DELETE anomaly: keď zmažeme všetkých zamestnancov pracoviska, stratíme informáciu o jeho adrese.

Nedostatky v návrhu — anomálie pri vkladaní

Zamestnanec	*Pracovisko*	*Adresa pracoviska*
Adam	SAV	Patrónka
Adam	KAGDM	Mlynská dolina
Cyril	KAGDM	Mlynská dolina

INSERT anomaly: nemožno zmysluplne pridať adresu pracoviska bez pridania zamestnancov (resp. NULL).

Nedostatky v návrhu — ako sa im vyhnúť?

Riešenie: dekompozícia relácií.

Zamestnanec	*Pracovisko*
Adam	SAV
Adam	KAGDM
Cyril	KAGDM

Pracovisko	*Adresa prac.*
SAV	Patrónka
KAGDM	Mlynská d.

- žiadna redundancia
- žiadne anomálie (INSERT, UPDATE, DELETE)

Naplnenie relácií v dekompozícii získame projekciou z pôvodnej relácie. Pôvodné dáta získame pomocou operácie JOIN.

Funkčné závislosti

Ako vieme, podľa ktorých atribútov robiť dekompozíciu a kedy už je dostatočná? Matematický model: funkčné závislosti.

Uvažujme reláciu R a dve množiny atribútov X, Y.
Funkčná závislosť $X \to Y$ platí práve vtedy,
keď pre každé naplnenie relácie R platí:
ak sa dva záznamy zhodujú na všetkých atribútoch z X,
zhodujú sa aj na všetkých atribútoch z Y.

Príklad: R = (Meno, RodnéČíslo, ČísloPasu); FZ:
$\{\text{RodnéČíslo}\} \to \{\text{Meno}\}$
$\{\text{RodnéČíslo, Meno}\} \to \{\text{ČísloPasu, Meno}\}$

ČísloPasu môže byť NULL, preto sa nehodí na identifikáciu záznamu. FZ to nepokazí, lebo NULL hodnota nie je rovná žiadnej inej, ani NULL, a môžeme použiť UNIQUE.

Funkčné závislosti a kľúče

FZ nemožno identifikovať z naplnenia databázy (pri inom by nemuseli platiť), musia byť preto súčasťou návrhu. Pomocou FZ vieme vyjadriť napr. pojem kľúča.

Množina atribútov X je v nejakej relácii s atribútmi R nadkľúč, ak platí $X \rightarrow R$. Nadkľúč minimálny vzhľadom na inklúziu sa nazýva kľúč. (V angličtine sa pre dvojicu pojmov nadkľúč/kľúč používa superkey/key alebo key/candidate key.)

Aké kľúče má relácia (Meno, RodnéČíslo, ČísloPasu)?

Primary keys

V relácii (Meno, RodnéČíslo, ČísloPasu) sú atribúty RodnéČíslo a ČísloPasu kľúčmi. Pre externé nástroje je výhodné, keď možno záznamy identikovať úplne jednoznačne, preto si zo všetkých kľúčov chceme vybrať primárny.

- PostgreSQL: PRIMARY KEY = NOT NULL UNIQUE.
- SQLite: PRIMARY KEY je unique, ale *nie* NOT NULL.

  `https://www.sqlitetutorial.net/sqlite-primary-key/`

Primárny kľúč môže zahŕňať viac atribútov:

```
CREATE TABLE ta (
  a1 integer, a2 integer,
  PRIMARY KEY (a1, a2)
);
```

Primary keys

Pre niektoré tabuľky nemáme žiadnych prirodzených kandidátov na primárny kľúč (napr. ak tabuľka len zachytáva vzťah M:N). Vtedy môžeme pridať ako primárny kľúč umelý identifikátor, ktorý jednoznačne identifikuje záznam.

- PostgreSQL: SERIAL.
- SQLite: atribút definovaný ako INTEGER PRIMARY KEY je automaticky aliasom pre ROWID (stĺpec s automaticky inkrementovanými hodnotami).

Umelé id možno využiť, aj keď máme iné kľúče, napr. ak sa chceme vyhnúť kompozitným kľúčom či pridlhým hodnotám kľúčových atribútov (index nad celými číslami zaberá menej miesta a je rýchlejší než nad dlhými reťazcami).

Funkčné závislosti

Reprezentácia funkčných závislostí v SQL: pomocou UNIQUE.

Pre závislosť $AB \rightarrow C$ spravíme tabuľku

```
CREATE TABLE t (
        A <data_type_of_A>,
        B <data_type_of_B>,
        C <data_type_of_C>,
        UNIQUE(A, B)
);
```

Keďže sa do nej nedajú vložiť záznamy (a_1, b_1, c_1) a (a_1, b_1, c_2), platí aj požadovaná funkčná závislosť.

Funkčné závislosti

Aké funkčné závislosti musia platiť pre nasledujúcu reláciu?
(Hľadáme závislosti vynútené SQL kódom.)

```
CREATE TABLE customers (
    id          integer PRIMARY KEY,      -- I
    address     text NOT NULL,            -- A
    name        text NOT NULL,            -- N
    email       text NOT NULL UNIQUE,     -- E
    phone       text NOT NULL UNIQUE,     -- P
    UNIQUE(name, address)
);
```

Funkčné závislosti

Aké funkčné závislosti musia platiť pre nasledujúcu reláciu? (Hľadáme závislosti vynútené SQL kódom.)

```
CREATE TABLE customers (
    id          integer PRIMARY KEY,      -- I
    address     text NOT NULL,            -- A
    name        text NOT NULL,            -- N
    email       text NOT NULL UNIQUE,     -- E
    phone       text NOT NULL UNIQUE,     -- P
    UNIQUE(name, address)
);
```

- PRIMARY KEY: $I \to ANEP$
- UNIQUE: $E \to I$, $P \to I$, $AN \to I$
- Z významu stĺpcov to vyzerá tak, že iné FZ (okrem tých, čo vyplývajú z uvedených) už medzi týmito atribútmi nie sú (a nechceme ich v SQL vynucovať).

Funkčné závislosti

Funkčné závislosti treba vnímať so všetkými ich dôsledkami. Predpokladajme, že platí

$$A \to C, \quad B \to D, \quad CD \to E.$$

Potom aj

$$\begin{aligned} AB &\to D \\ A &\to A \\ AB &\to E \\ &\dots \end{aligned}$$

Funkčné závislosti

Pravidlá pre odvodzovanie — Armstrongove axiómy.

1. Ak $X \to Y$, tak $XZ \to YZ$.
2. Ak $Y \subseteq X$, tak $X \to Y$.
3. Ak $X \to Y$ a $Y \to Z$, tak $X \to Z$.

Ich platnosť vyplýva priamo z definície FZ.

Odvodiť sa dajú ďalšie pravidlá, napr.
$A_1 A_2 \ldots A_n \to B_1 B_2 \ldots B_n \;\leftrightarrow\; \forall i \; A_1 A_2 \ldots A_n \to B_i$.

Množinu všetkých FZ, ktoré možno odvodiť z danej množiny FZ M, nazývame uzáver množiny FZ M.
(Jeho veľkosť môže byť exponenciálna vzhľadom na veľkosť M, čo spôsobuje algoritmické problémy.)

Funkčné závislosti

Uzáver množiny atribútov $\{A_1, A_2 \ldots, A_n\}$ vzhľadom na nejakú množinu FZ M: maximálna množina atribútov B taká, že platí $A_1 A_2 \ldots A_n \rightarrow B$.

Uzáver možno vypočítať v polynomiálnom čase tak, že začneme pôvodnou množinou atribútov, a v každom kroku prejdeme všetky FZ z M a pre každú z nich pridáme do uzáveru pravú stranu, ak sa tam už nachádza ľavá strana. Keď sme nič nepridali počas celého prechodu M, končíme.

Uzáver možno využiť na overenie, či je daná množina atribútov kľúč (ako?), alebo na výpočet všetkých FZ platných v danej relácii (vyskúšame všetky možné podmnožiny atribútov ako ľavé strany a na pravú stranu pripíšeme uzáver ľavej strany).

Dekompozícia relácií

V princípe každú databázu možno vnímať ako jedinú reláciu (predstavte si join všetkých tabuliek). Preto dekompozícia relácie na menšie je pri návrhu kľúčová, aj keď ju explicitne nevnímame.

Pri dekompozícii nechceme polámať funkčné závislosti.

Príklad: relácia $order\_items(Order, Product, Amount)$ so závislosťou

$$\{Order, Product\} \rightarrow \{Amount\}.$$

Ak ju akokoľvek dekomponujeme, stratíme informáciu o tom, že $\{Order, Product\}$ je kľúč. Pri kontrole integrity budeme musieť pri každej zmene niektorej z čiastkových relácií spočítať join a overiť, či nedošlo k porušeniu FZ.

Dekompozícia relácií — bezstratovosť

Z dekomponovanej relácie musíme vedieť získať pôvodné záznamy. Inak povedané, ak spravíme join, dostaneme práve pôvodné záznamy a nič navyše — nestratili sme žiadnu informáciu, čiže ide o bezstratovú dekompozíciu (lossless decomposition).

Otestovať, či je dekompozícia bezstratová, možno polynomiálnym algoritmom (link: chase algorithm).

Jednoduchšie riešenie pre 2 relácie: požadujeme, aby množina spoločných atribútov bola v aspoň jednej z nich nadkľúčom.

Dekompozícia relácií — bezstratovosť

$r(Zamestnanec, Pracovisko, AdresaPracoviska)$
dekomponujeme na
$r_1(Zamestnanec, Pracovisko)$,
$r_2(Pracovisko, AdresaPracoviska)$.
Je táto dekompozícia bezstratová?

Dekompozícia relácií — bezstratovosť

$r(\mathit{Zamestnanec}, \mathit{Pracovisko}, \mathit{AdresaPracoviska})$
dekomponujeme na
$r_1(\mathit{Zamestnanec}, \mathit{Pracovisko})$,
$r_2(\mathit{Pracovisko}, \mathit{AdresaPracoviska})$.
Je táto dekompozícia bezstratová?

Označme $Z = \mathit{Zamestnanec}$, $P = \mathit{Pracovisko}$,
$A = \mathit{AdresaPracoviska}$.

V r platia FZ $Z \to P$, $P \to A$. Preto P $(= r_1 \cap r_2)$ je kľúč v r_2 a do (natural) joinu $r_1 \bowtie r_2$ sa nemôže dostať záznam, ktorý v r nebol, pretože tento join ku každému riadku r_1 len doplní hodnotu A z r_2.

Dekompozícia relácií

Požiadavky na dekompozíciu:

- Je bezstratová. (Nutné.)
- Neláme funkčné závislosti. (Z tohto sa dá trocha zľaviť, ale je oveľa lepšie FZ zachovať, umožňuje to kontrolu integrity štandardnými prostriedkami.)
- Odstraňuje čo najviac redundancie a anomálií.

Redundancia sa odstraňuje normalizáciou — cieleným dekomponovaním. Mieru normalizácie popisujú normálne formy.

Normálne formy

1NF Nemáme duplikáty ani kompozitné atribúty (hodnotou atribútu nie je relácia).

3NF Odstraňuje väčšinu redundancie a problémov s anomáliami. Vždy existuje a ľahko sa hľadá.

BCNF Čosi ako 3.5NF, ideál. Relácia v BCNF neobsahuje žiadnu redundanciu vzhľadom na FZ, ani nedochádza k spomínaným anomáliám.

Normálne formy vytvárajú hierarchiu (ak je niečo vo vyššej, je aj v nižšej). Existujú aj vyššie normálne formy (napr. 4NF), tie sa venujú odstraňovaniu tzv. multivalued dependencies. V praxi sú zväčša irelevantné.

BCNF

Relácia r je v BCNF, ak ľavá strana každej netriviálnej FZ je nadkľúč r.

Problémy:

- Pre niektoré relácie neexistuje (napr. $r(ABC)$, kde FZ sú $AB \to C$, $C \to A$).
- Pre danú reláciu je NP-úplné rozhodnúť, či je v BCNF.

Ak relácia R nie je v BCNF kvôli FZ $X \to Y$, dekomponujeme ju do XY a $R - Y$. (Zachováva bezstratovosť.)
(V literatúre možno nájsť iné, hoci ekvivalentné, definície jednotlivých normálnych foriem. BCNF nemá pekné číslo: zámerom bolo už v 3NF odstrániť všetku redundanciu, ale vyšlo to až na druhý pokus.)

3NF

Relácia r je v 3NF, ak ľavá strana každej netriviálnej FZ je nadkľúč r alebo každý atribút na pravej strane je súčasťou nejakého kľúča.

Vlastnosti:

- Pre danú reláciu je NP-úplné rozhodnúť, či je v 3NF.
- Dekompozícia do 3NF vždy existuje.
- Existuje polynomiálny algoritmus, ktorý aspoň jednu 3NF nájde (cez minimálne pokrytie). Nájdená dekompozícia je bezstratová a zachováva FZ.

Schéma vytvorená na základe dobrého entitno-relačného modelu je zväčša už v 3NF.

Normalizácia

Normalizujme reláciu R:

Zamestnanec Z	*Pracovisko P*	*Adresa pracoviska A*
Adam	KAGDM	Mlynská dolina
Cyril	KAGDM	Mlynská dolina

Normalizácia

Normalizujme reláciu R:

Zamestnanec Z	*Pracovisko P*	*Adresa pracoviska A*
Adam	KAGDM	Mlynská dolina
Cyril	KAGDM	Mlynská dolina

FZ: $Z \to P$, $P \to A$. Jediný kľúč je Z.

FZ $P \to A$ porušuje BCNF, lebo ľavá strana nie je nadkľúč.
Aj $3NF$, lebo atribút A na pravej strane nie je súčasťou žiadneho kľúča.

Preto dekomponujeme: do PA a $R - A = ZP$.
Tie sú binárne, preto v BCNF. FZ ostali zachované.

Užitočná redundancia

Ak máme veľa dát, možno nechceme opakovane rátať agregačné funkcie, povedzme súčet. Jeho hodnotu H možno vypočítať dopredu, napr. tak, že záznamy dostávajú timestamp, podľa ktorého je spravený BTREE index, a v rámci dotazu počítame len H k nejakému timestampu plus záznamy, ktoré pribudli.

V takom prípade treba strážiť konzistenciu dát: pri zmene hodnôt originálnych záznamov (napr. zmazanie najstarších) aktualizovať H a naopak nedovoliť zmenu H bez úpravy originálnych záznamov.

Mechanizmus: trigger — funkcia uložená v databáze, ktorá sa spustí zakaždým, keď nastane definovaná udalosť, napr. pred pridaním či po pridaní riadka.

Normalizácia

Prehnaná normalizácia vedie k binárnym reláciám
(tie sú vždy v BCNF).

Príklad: binárne relácie
(Id, Meno), (Id, Priezvisko), (Id, RodneCislo)
miesto
(Id, Meno, Priezvisko, RodneCislo).

Zaužívané pravidlá je pri normalizácii občas výhodné porušiť, treba však mať jasno, že prečo, a uviesť to v dokumentácii. Kompromis medzi zachovaním FZ a znížením redundancie nemá jediné správne objektívne riešenie, treba sa riadiť skúsenosťou.

Foreign keys

Atribút obsiahnutý v niekoľkých reláciách:

- V matematickom relačnom modeli vždy „ten istý“.
- V SQL rovnako pomenované stĺpce v rôznych tabuľkách nemajú medzi sebou žiaden vzťah.

Foreign keys (cudzie kľúče) — mechanizmus na zabezpečenie súladu hodnôt toho istého atribútu v rôznych tabuľkách (referenčná integrita).

Foreign keys

```
CREATE TABLE ta (
  a1 integer REFERENCES tb,
  a2 integer,
  a3 integer,
  FOREIGN KEY (a2, a3) REFERENCES tc (c1, c2)
);
```

Cudzie kľúče možno pomenovať (ľahšie sa tak identifikuje dôvod zamietnutia operácie):

```
CONSTRAINT fk_name1 FOREIGN KEY a1 REFERENCES tb (b1)
CONSTRAINT fk_name2 FOREIGN KEY (a2, a3) REFERENCES tc (c1, c2)
```

Foreign keys

```
CREATE TABLE products (
  product_no integer PRIMARY KEY,
  price numeric
);
CREATE TABLE orders (
  order_id integer PRIMARY KEY
);
CREATE TABLE order_items (
  order_id integer REFERENCES orders ON DELETE CASCADE,
  product_no integer REFERENCES products
      ON DELETE RESTRICT ON UPDATE CASCADE,
  quantity integer,
  PRIMARY KEY (product_no, order_id)
);
```

Foreign keys

Možnosti pre ON DELETE

- CASCADE – zmaže riadok, ktorý sa odkazoval na mazaný riadok
- RESTRICT – nezmaže nič, operácia DELETE skončí s chybou
- SET NULL – nastaví odkaz na NULL
  (zlyhá pre NOT NULL stĺpce)
- SET DEFAULT – nastaví hodnotu odkazu na defaultnú
  (zlyhá, ak takto vznikne nekorektný odkaz)
- NO ACTION – toto je default, DELETE skončí s chybou

Foreign keys

Možnosti pre ON UPDATE

- CASCADE – zmení odkazujúcu hodnotu na novú hodnotu odkazovanej (zachová väzbu medzi riadkami)
- RESTRICT – zabráni zmene v odkazovanej tabuľke, UPDATE zlyhá
- SET NULL – nastaví odkaz na NULL (väzba zanikne)
- SET DEFAULT – nastaví hodnotu odkazu na defaultnú (zlyhá, ak takto vznikne nekorektný odkaz)
- NO ACTION – toto je default, väzba sa pokazí

Foreign keys

Kedy sa vyhodnocuje platnosť odkazu?

- DEFERRED – až pri commite transakcie
- IMMEDIATE – okamžite po vykonaní operácie

Foreign keys

Kedy sa vyhodnocuje platnosť odkazu?

- DEFERRED – až pri commite transakcie
- IMMEDIATE – okamžite po vykonaní operácie

Možnosti pri definícii cudzieho kľúča:

- NOT DEFERRABLE – vždy okamžité vyhodnocovanie
- DEFERRABLE INITIALLY IMMEDIATE – v rámci tx možno zvoliť, kedy vyhodnocovať, default je ihneď
- DEFERRABLE INITIALLY DEFERRED – v rámci tx možno zvoliť, kedy vyhodnocovať, default je commit

Literatúra

- entity-relationship diagrams
- normal forms
- normalization example 1
- normalization example 2
- `https://www.postgresql.org/docs/current/ddl-constraints.html`
- `https://www.postgresql.org/docs/current/sql-set-constraints.html`
- trigger v PostgreSQL
- trigger v SQLite
- `https://www.db-book.com/slides-dir/PDF-dir/ch6.pdf`
- `https://www.db-book.com/slides-dir/PDF-dir/ch7.pdf`
- `https://cs186berkeley.net/notes/note13/`

Literatúra

Online normalization tools (limited functionality)

- https://www.ict.griffith.edu.au/normalization_tools/normalization/index.html
- https://arjo129.github.io/functionalDependencyCalculator/
- http://raymondcho.net/RelationalDatabaseTools/RelationalDatabaseTools.html
- https://uisacad5.uis.edu/cgi-bin/mcrem2/database_design_tool.cgi

Úloha

Máme databázu kníh; predpokladáme, že každú napísal jediný autor. Atribúty:

- ISBN
- Názov
- Autor
- NárodnosťAutora
- Formát
- Cena
- Témy (zoznam tém)
- PočetStrán
- Vydavateľstvo
- KontaktVydavateľstvo
- Žáner
- DefiníciaŽánru

Identifikujte funkčné závislosti a navrhnite vhodnú databázovú schému v BCNF alebo aspoň v 3NF. Overte bezstratovosť dekompozície.