FBERG · TUKE · 2025/2026

Objektové programovanie

Návrh dátového modelu

doc. Ing. Beáta Stehlíková, PhD.

Tento dokument vysvetľuje postup myslenia pri návrhu databázovej schémy. Na troch príkladoch (knižnica, fitko, receptár) ukážeme ako vyzerá zlý a dobrý model a prečo na správnom návrhu záleží ešte predtým, než napíšeme prvý riadok kódu.

01 — Čo je dátový model a prečo na ňom záleží

Dátový model je plán databázy — rozhodnutie o tom, aké entity existujú, aké majú atribúty a ako sú medzi sebou prepojené. Je to ako architektúra budovy: keď ju postavíš zle, môžeš neskôr pridávať priečky, ale základy nezmeníš bez búrania.

Dôsledky zlého návrhu

- **Duplicita dát —** tá istá informácia je uložená na viacerých miestach. Zmena jednej veci si vyžaduje update na N miestach — a vždy zabudneš na jedno.
- **Nemožnosť rozšírenia —** chceš pridať nový atribút? Musíš meniť štruktúru tabuľky, migrovať dáta a prepísať SQL dotazy.
- **Strata integrity —** keď vymažeš záznam, môžu ostať osirelé záznamy v iných tabuľkách ktoré na neho odkazujú.
- **Pomalé dotazy —** zle navrhnutá schéma núti písať komplikované JOINy a poddotazy na veci, ktoré by mali byť jednoduché.

02 — Postup rozmýšľania— od reality k tabuľkám

Pred samotným návrhom tabuliek pre projekt si sadni a odpovedz na otázky v tomto poradí:

Krok 1 — Aké entity pre môj projekt existujú v reálnom svete?

Entity sú "veci" o ktorých chceme uchovávať dáta. Napr. Kniha, Čitateľ, Recept, Surovina. Každá entita bude tabuľka.

Krok 2 — Aké atribúty má každá entita?

Atribúty = stĺpce. Pýtaj sa: "Čo o tej veci potrebujem vedieť?" Napr. Kniha má názov, ISBN, rok vydania. Nepridávaj atribúty "pre istotu" — len tie čo naozaj použiješ.

Krok 3 — Aký je vzťah medzi entitami?

1:N — jeden autor môže mať viac kníh, ale každá kniha má len jedného autora. N:M — jeden recept má viac surovín a jedna surovina môže byť vo viacerých receptoch.

Krok 4 — Čo je číselník a čo je plnohodnotná entita?

Číselník (lookup tabuľka) = malý zoznam možností ktoré sa nemenia (jednotky, kategórie). Entita = plnohodnotný objekt s vlastným životným cyklom (používateľ, objednávka, ingrediencia).

Krok 5 — Ako to v budúcnosti rozšírim bez straty dát?

Navrhuješ na budúcnosť. Ak vieš že neskôr pribudnú alergény, používatelia, výživové hodnoty — nechaj na ne miesto v schéme od začiatku.

Krok 6 — čo od aplikácie očakávam aké dátové toky, prehľady, funkcie ?

Pred finalizáciou schémy si polož tieto otázky:

Prehľady — čo chcem zobraziť?

Funkcie — čo chcem robiť?

Validácia modelu — viem to všetko? Ak na niektorú otázku odpoveď je *"to z tejto schémy neviem vybrať"* — schéma je neúplná a treba ju doplniť **teraz**, nie po napísaní kódu.

03 — Zlé vs. dobré modely — 3 príklady

Príklad 1 — Evidencia kníh (Knižnica)

Problém: Chceme evidovať knihy a ich autorov. Zlý návrh uloží autora priamo do tabuľky kníh.

✗ Zlý návrh

```
CREATE TABLE books (
  id           INT AUTO_INCREMENT PRIMARY
KEY,
  title        VARCHAR(200),
  -- Autor ulozeny ako text -- duplicita!
  author       VARCHAR(100),
  author_email VARCHAR(150)
);

-- Problem: Tolkien je v DB 50x
-- Oprava priezviska = 50 UPDATE dotazov
```

☑ Dobrý návrh

```
CREATE TABLE authors (
  id    INT AUTO_INCREMENT PRIMARY KEY,
  name  VARCHAR(100) NOT NULL,
  email VARCHAR(150)
);

CREATE TABLE books (
  id        INT AUTO_INCREMENT PRIMARY
KEY,
  title     VARCHAR(200) NOT NULL,
  author_id INT NOT NULL,
  FOREIGN KEY (author_id)
    REFERENCES authors(id)
);
-- Tolkien je v DB 1x
```

→ **Autor je entita, nie atribút. Každá entita si zaslúži vlastnú tabuľku.**

Príklad 2 — Rezervácia cvičení (Fitko)

Problém: Používateľ sa môže prihlásiť na viac cvičení, jedno cvičenie môže mať viac účastníkov. Vzťah N:M.

✗ Zlý návrh

```
CREATE TABLE users (
  id       INT AUTO_INCREMENT PRIMARY
KEY,
  name     VARCHAR(100),
  -- Cvicenia ulozene ako text?!
  workouts TEXT
  -- 'yoga,spinning,pilates'
  -- Neda sa filtrovat, joinovat,
  -- ani zmazat jedno cvicenie
);
```

☑ Dobrý návrh

```
CREATE TABLE users (
  id   INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL
);
CREATE TABLE workouts (
  id   INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL
);
-- M:N pivot tabulka
CREATE TABLE user_workouts (
  user_id    INT NOT NULL,
  workout_id INT NOT NULL,
  booked_at  TIMESTAMP DEFAULT NOW(),
  PRIMARY KEY (user_id, workout_id)
);
```

→ **Nikdy neukladaj zoznamy ako TEXT. Pre N:M vzťah vždy pivot tabuľka.**

Príklad 3 — Receptár (náš projekt)

Problém: Ako uložiť suroviny receptu tak, aby sme vedeli filtrovať podľa suroviny, jednotky, kategórie aj výživových hodnôt?

✗ Zlý návrh

```
CREATE TABLE recipes (
  id          INT AUTO_INCREMENT PRIMARY
KEY,
  title       VARCHAR(200),
  -- Suroviny ulozene ako jeden text!
  ingredients TEXT,
  -- '200g tofu, 1 cibula, sol'
);

-- Problemy:
-- Neda sa filtrovat podla suroviny
-- Jednotky nie su konzistentne
-- Nie je mozne pridat vyz. Hodnoty,
robiť súčty, hmotností, kalorických
hodnôt, minerálov, bielkovín, cukrov....
```

☑ Podmienečne dobrý návrh, základ pre ďfalšie uvažovanie

```
-- Ciselniky
CREATE TABLE units ( id, name );
CREATE TABLE ingredient_categories (
  id, name );

-- Suroviny ako entita
CREATE TABLE ingredients (
  id          INT AUTO_INCREMENT PRIMARY
KEY,
  name        VARCHAR(100) NOT NULL,
  unit_id     INT NOT NULL,
  category_id INT NOT NULL,
  FOREIGN KEY (unit_id)
    REFERENCES units(id),
  FOREIGN KEY (category_id)
    REFERENCES ingredient_categories(id)
);
```

Prehľad tabuliek

Tabuľka	Typ	Popis
units	číselník	Merné jednotky: g, ml, ks, lyžica...
ingredient_categories	číselník	Kat. surovín: strukoviny, zelenina...
ingredients	entita	Suroviny — napojené na unit a category
nutrition_types	číselník	Typy výž. hodnôt: kalórie, bielkoviny...
ingredient_nutrition	M:N pivot	Výživové hodnoty ku každej surovine
recipe_categories	číselník	Kat. receptov: raňajky, obed, dezert...
users	entita	Registrovaní používatelia
recipes	entita	Recepty — napojené na user a recipe_category
recipe_ingredients	M:N pivot	Suroviny receptu + množstvo

Úvahy: riešime priblíženie ku realite **Čo sme riešili a prečo**
Kategórie receptov — prečo M:N? Jedno jedlo môže byť súčasne obed aj večera, dezert aj desiata. Keby sme dali category_id priamo do receptu (1:N), recept by mohol mať len jednu kategóriu. Preto pivot tabuľka recipe_categories — recept môže patriť do ľubovoľného počtu kategórií.

Prečo jednotka ostala na surovine a nie na pivote? Zvažovali sme dať unit_id do recipe_ingredients aby každý recept mohol použiť inú jednotku pre tú istú surovinu. Rozhodli sme sa že jednotka patrí k surovine — tofu je vždy v gramoch, mlieko vždy v ml. Toto zjednodušuje model a zabraňuje nekonzistentnosti.
Prečo nutrition_types ako číselník a nie stĺpce priamo v ingredients? Keby sme dali kalórie, bielkoviny, tuky ako stĺpce priamo do ingredients, každé pridanie novej výživovej hodnoty (železo, horčík, vláknina...) by znamenalo ALTER TABLE = zmenu štruktúry databázy a prepísanie kódu. S číselníkom nutrition_types stačí jeden INSERT — žiadna zmena schémy, žiadna strata dát.
Prečo users nie sú teraz a ako ich pridáme? Users zatiaľ nie sú potrební. Keď ich pridáme, nebudeme robiť ALTER TABLE recipes ADD COLUMN user_id dodatočne — to by bol technický dlh. Namiesto toho vytvoríme tabuľku users hneď teraz s jedným systémovým záznamom (admin, id=1), stĺpec user_id bude v recipes od začiatku s DEFAULT 1 a cudzím kľúčom. Všetky recepty zatiaľ patria adminovi. Keď prídu skutoční používatelia, systém je pripravený — žiadna zmena schémy, len nové záznamy v users a update user_id v receptoch. Roly používateľov (viewer, editor, admin) prídu ako ďalšie rozšírenie.

04 — Náš model: Receptár vegánskych jedál

Toto je finálna schéma ktorú budeme používať celý semester. Je navrhnutá tak, aby sa dala rozšíriť (alergény, jazyky, hodnotenia) bez zmeny existujúcich tabuliek.

Prehľad tabuliek

#	Tabuľka	Typ	Poznámka
1	units	číselník	g, ml, ks, lyžica...
2	ingredient_categories	číselník	strukoviny, zelenina, koreniny...
3	nutrition_types	číselník	kalórie/kcal, bielkoviny/g, tuky/g...
4	categories	číselník	raňajky, desiata, obed, večera, dezert...
5	users	entita	zatiaľ 1 záznam — admin
6	ingredients	entita	surovina má unit + category
7	recipes	hlavná entita	má user_id FK → users
8	recipe_categories	M:N pivot	recept ↔ kategória
9	ingredient_nutrition	M:N pivot	surovina ↔ výživová hodnota + value
10	recipe_ingredients	M:N pivot	recept ↔ surovina + amount

SQL schéma

```
-- ── ČÍSELNÍKY ──────────────────────────────────────────

CREATE TABLE units (
    id   INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(30) NOT NULL    -- g, ml, ks, lyžica, lyžička...
);

CREATE TABLE ingredient_categories (
    id   INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(80) NOT NULL    -- strukoviny, zelenina, koreniny...
```

```
);

CREATE TABLE nutrition_types (
    id   INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(80) NOT NULL,   -- kalórie, bielkoviny, tuky, vláknina...
    unit VARCHAR(20)             -- kcal, g, mg...
);

CREATE TABLE categories (
    id   INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(80) NOT NULL    -- raňajky, desiata, obed, večera, dezert...
);

-- ── ENTITY ──────────────────────────────────────────────

CREATE TABLE users (
    id       INT AUTO_INCREMENT PRIMARY KEY,
    name     VARCHAR(100) NOT NULL,
    email    VARCHAR(150) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    created  TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Systémový záznam — všetky recepty zatiaľ patria adminovi
INSERT INTO users (id, name, email, password)
VALUES (1, 'admin', 'admin@localhost', 'zatial_nepouzivame');

CREATE TABLE ingredients (
    id          INT AUTO_INCREMENT PRIMARY KEY,
    name        VARCHAR(100) NOT NULL,
    unit_id     INT NOT NULL,
    category_id INT NOT NULL,
    FOREIGN KEY (unit_id)     REFERENCES units(id),
    FOREIGN KEY (category_id) REFERENCES ingredient_categories(id)
);

CREATE TABLE recipes (
    id           INT AUTO_INCREMENT PRIMARY KEY,
    title        VARCHAR(200) NOT NULL,
    description  TEXT,
    instructions TEXT,
    user_id      INT NOT NULL DEFAULT 1,
    created      TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
    FOREIGN KEY (user_id) REFERENCES users(id)
);

-- ── M:N PIVOT TABUĽKY ──────────────────────────────────────

CREATE TABLE recipe_categories (
    recipe_id   INT NOT NULL,
    category_id INT NOT NULL,
    PRIMARY KEY (recipe_id, category_id),
    FOREIGN KEY (recipe_id)   REFERENCES recipes(id),
    FOREIGN KEY (category_id) REFERENCES categories(id)
);

CREATE TABLE ingredient_nutrition (
    ingredient_id     INT NOT NULL,
    nutrition_type_id INT NOT NULL,
    value             DECIMAL(8,2) NOT NULL,
    PRIMARY KEY (ingredient_id, nutrition_type_id),
    FOREIGN KEY (ingredient_id)     REFERENCES ingredients(id),
    FOREIGN KEY (nutrition_type_id) REFERENCES nutrition_types(id)
);

CREATE TABLE recipe_ingredients (
    recipe_id     INT NOT NULL,
    ingredient_id INT NOT NULL,
    amount        DECIMAL(8,2),
    PRIMARY KEY (recipe_id, ingredient_id),
    FOREIGN KEY (recipe_id)     REFERENCES recipes(id),
    FOREIGN KEY (ingredient_id) REFERENCES ingredients(id)
);
```

Budúce rozšírenia bez zmeny schémy: Pridanie alergénov = nová tabuľka allergens + pivot ingredient_allergens. Pridanie nového typu výživovej hodnoty = nový riadok v nutrition_types. Žiadna existujúca tabuľka sa nemení, žiadne dáta sa nestratia.

Používatelia users podľa oprávnení keď príde čas.....

05 — Pravidlá rozšíriteľného návrhu

1 **Každá entita = vlastná tabuľka**

Ak o niečom uchovávaš viac ako 1 údaj, alebo ak sa ten istý objekt vyskytuje vo viacerých záznamoch — daj mu vlastnú tabuľku.

2 **Nikdy TEXT tam kde má byť vzťah**

Zoznam surovín neukladáš ako 'tofu, cícer, soľ" do jedného stĺpca. Každý prvok zoznamu = riadok v pivot tabuľke.

3

Číselníky sú tabuľky, nie ENUM

Jednotky, kategórie, typy — ukladaj ich do samostatných tabuliek. ENUM v MySQL sa mení len ALTER TABLE, tabuľka len INSERT.

4

Cudzí kľúč = ochrana integrity

FOREIGN KEY ti zabráni pridať recept s neexistujúcim user_id. Nikdy nezabudni na ON DELETE pravidlo (CASCADE, SET NULL, RESTRICT).

5

Navrhuješ na budúcnosť

Pýtaj sa: Čo ak zajtra príde požiadavka X? Ak by si musel meniť štruktúru tabuľky, návrh nie je dostatočne flexibilný.

6

Over model otázkami z praxe

Napíš si 5 viet začínajúcich slovom "Viem..." — prehľady a operácie ktoré aplikácia musí zvládnuť. Ak na niektorú nevieš odpovedať kladne, schéma nie je hotová. Oprav ju teraz — zmena modelu pred kódom trvá minúty, po kóde trvá hodiny.

Dátový model je rozhodnutie ktoré ovplyvní každý riadok kódu ktorý napíšeš. Venuj mu čas ešte pred otvorením editora.

---

06 — Ako používať AI na návrh databázy

AI vie vygenerovať databázovú schému za 30 sekúnd. Problém je že vygeneruje *nejakú* schému — nie nevyhnutne *správnu pre tvoju aplikáciu*. Tvoja úloha nie je nechať AI navrhnúť model, ale vedieť posúdiť či je navrhnutý model dobrý.

Ak použiješ AI, urob to takto:

1. Najprv sformuluj očakávania čo najpresnejšie Nedávaj AI iba tému. Povedz jej konkrétne čo aplikácia musí vedieť robiť — vypíš funkcie, prehľady, filtre. Čím presnejší vstup, tým použiteľnejší výstup. Rozdiel medzi *"navrhni databázu pre receptár"* a *"navrhni databázu pre receptár kde recept môže mať viac kategórií, surovina má jednotku a výživové hodnoty, a systém musí byť pripravený na pridanie používateľov bez zmeny schémy"* je zásadný.

2. Explicitne definuj vzťahy Povedz AI aké vzťahy očakávaš — inak ich odhadne a nemusí odhadnúť správne. Napríklad: *"recept a kategória sú M:N, surovina a jednotka sú 1:N, výživové hodnoty sú M:N cez číselník typov"*. Ak vzťahy nedefinuješ ty, definuje ich AI — a potom ich obhajuješ ty.

3. Výsledok over podľa postupu z tohto dokumentu Keď AI vygeneruje schému, prečítaj ju kriticky — nie či vyzerá dobre, ale či obstojí:

- Má každá entita vlastnú tabuľku?
- Nie sú niekde zoznamy uložené ako TEXT?
- Sú číselníky ako tabuľky, nie ENUM?
- Sú cudzie kľúče správne?
- Vieš odpovedať kladne na všetkých 5 *"Viem..."* otázok?

Ak nie — model oprav. Na obhajobe budeš vysvetľovať každé rozhodnutie ty, nie AI.