

## **Project Assignment 1 – Database Design**

Bezpečnost databázových systémů

Vypracovali: Martin Nečas, Natálie Klimková

# Úvod

V našem projektu jsme se snažili vytvořit databázi potenciální aplikace pro rozvoz jídla. Proč? Protože máme rádi jídlo. A co je lepší, než když vám ho někdo uvaří a doveze až domů. Při plánování projektu jsme vycházeli z již existujících aplikací Dáme jídlo a Wolt, díky tomu jsme si zjednodušili fázi vymýšlení jednotlivých tabulek, které by měla databáze obsahovat. Databáze funguje docela jednoduše. Naším úkolem je dostat jídlo k uživateli. K tomu potřebujeme poměrně dost informací, jako třeba adresu uživatele, kontakty pro doručení a co hlavně, jaké jídlo si uživatel vybral a v jaké restauraci. Stejně jako u adresy, tak i u samotné objednávky má uživatel možnost napsat vlastní požadavky. Jakmile tyto informace zjistíme, můžeme přejít k části, která vyžaduje trochu logistických schopností. My se toho ale nebojíme, protože tabulky nám pomůžou. Nejen, že se z nich dozvíme, který řidič má zrovna jakou směnu, ale třeba i to, jakým jezdí autem. Všechny zásadní informace potřebné k doručení jídla se spojí v tabulce delivery, která mimo jiné ukazuje třeba čas doručení. U toho ale nekončíme! Zpětná vazba je pro nás velice důležitá, a proto jsme se rozhodli dát uživatelům možnost vyjádřit své dojmy a připomínky nejen u samotných jídel, ale i u celé restaurace. Samozřejmostí je také hodnocení řidičů, kteří rádi uslyší pozitivní zprávy, a ty negativní je snad někde v životě posunou.

## Vysvětlení jednotlivých tabulek:

Pro vše, co obsahuje `_id` jsme zvolili typ INT. Práce s čísly je rychlejší a jednodušší než práce s řetězcem.

### person:

Tato tabulka je určena pro zadávání informací o uživateli nebo řidičích. Pro `first_name`, `last_name`, `password` a `username` jsme zvolili typ VARCHAR, protože zde očekáváme, že uživatel zadá znakový řetězec. U `date_of_birth` je typ DATE, protože očekávaným vstupem je datum.

person	
person_id	INT
person_type_id	INT
first_name	VARCHAR(45)
last_name	VARCHAR(45)
password	VARCHAR(45)
username	VARCHAR(45)
date_of_birth	DATE

### contact:

Tabulka bude obsahovat kontakty uživatelů a řidičů z tabulky person. Pro `type` a `value` jsme zvolili VARCHAR, protože zde očekáváme vložení čísla/mailu a označení daného kontaktu.

contact	
contact_id	INT
person_id	INT
type	VARCHAR(45)
value	VARCHAR(45)

### person\_type:

Tabulka dělí tabulku person na řidiče a uživatele. Pro rozdělení jsme zvolili typ VARCHAR.

person_type	
person_type_id	INT
type	VARCHAR(45)

### shift, driver\_has\_shift:

Tyto tabulky používáme na přehled směn jednotlivých řidičů. V shift tabulce jsou vypsané jen směny, proto jsme pro start\_time a end\_time zvolili typ DATETIME. Tabulka driver\_has\_shift přiřadí řidiče k dané směně.

### car:

Tabulka se používá na evidenci aut. Pro plate, type a brand jsme zvolili VARCHAR, protože očekáváme na vstupu znakový řetězec.

### address:

Tabulka zaznamenává jednotlivé údaje, které souvisí s adresou uživatele. Pro city, street, zip a description jsme zvolili typ VARCHAR, protože očekáváme na vstupu znakový řetězec. V případě street\_number a floor jsme zvolili INT, protože všude očekáváme jako vstup číslo.

### user\_has\_address:

Tabulka spojuje adresy s uživateli. Type je použito pro pojmenování typu adresy (doma, práce, ...), proto je použito VARCHAR.

### restaurant:

Tuto tabulku používáme pro jednotlivé restaurace. Name je VARCHAR, protože očekáváme na vstupu znakový řetězec (jméno restaurace).

### cuisine:

Tabulka je spojená s restaurací a určuje typ kuchyně, kterou restaurace vaří. U cuisine\_name je VARCHAR, protože očekáváme na vstupu znakový řetězec (název kuchyně).

### dish:

Tabulka je věnovaná hlavnímu důvodu našeho projektu, a to jednotlivým jídlům. U name je VARCHAR, protože očekáváme na vstupu znakový řetězec (název jídla).

#### shift

- shift\_id INT
- start\_time DATETIME
- end\_time DATETIME

#### driver\_has\_shift

- driver\_id INT
- shift\_id INT
- person\_id INT

#### car

- car\_id INT
- person\_id INT
- plate VARCHAR(45)
- type VARCHAR(45)
- brand VARCHAR(45)

#### address

- address\_id INT
- city VARCHAR(45)
- street VARCHAR(45)
- street\_number INT
- zip VARCHAR(20)
- description VARCHAR(45)
- floor INT

#### user\_has\_address

- person\_id INT
- address\_id INT

#### restaurant

- restaurant\_id INT
- address\_id INT
- name VARCHAR(45)
- cuisine\_id INT

#### cuisine

- cuisine\_id INT
- cuisine\_name VARCHAR(45)

#### dish

- dish\_id INT
- name VARCHAR(45)

### dish\_has\_restaurant:

Tabulka spojující jednotlivá jídla a restaurace. Najdeme v ní taky ceny, pro které používáme typ FLOAT.

dish_has_restaurant ▼	
🔑	dish_id INT
🔑	restaurant_id INT
💎	price FLOAT

### delivery:

Tabulka delivery se týká samotné objednávky. Zjistíme zde, který řidič komu objednávku doveze a také můžeme zjistit čas příjezdu (jde o čas a datum, proto DATETIME) a cenu doručení (INT, protože počítáme s číselným vstupem).

delivery ▼	
🔑	delivery_id INT
🔑	person_id INT
🔑	driver_id INT
💎	arrival DATETIME
💎	delivery_fee INT

### delivery\_has\_dish\_has\_restaurant:

Tabulka spojující tabulku delivery, dish a restaurant. Plus má možnost uživatel dopsat poznámky k jídlu (například „bez koriandru“). Pro user\_requirements jsme zvolili typ TEXT s délkou 200, kdyby těch požadavků bylo hodně.

delivery_has_dish_has_restaurant ▼	
🔑	delivery_id INT
🔑	dish_id INT
🔑	restaurant_id INT
💎	user_requirements TEXT(200)

### review:

Poslední ale ne méně důležitá je tabulka review. Dělá přesně to, co by od ní člověk očekával. Díky ní se dozvíme, jestli je vybrané jídlo přesolené, jestli je restaurace tak dobrá, jak se o ní říká, nebo jaké chování má ten hezký řidič, co nám jídlo přivezl. Rating je typ TINYINT, protože nám to stačí. Text je typu TEXT s délkou 500, aby se člověk mohl pořádně rozepsat.

review ▼	
🔑	review_id INT
🔑	person_id INT
🔑	restaurant_id INT
🔑	dish_id INT
🔑	driver_id INT
💎	rating TINYINT
💎	text TEXT(500)

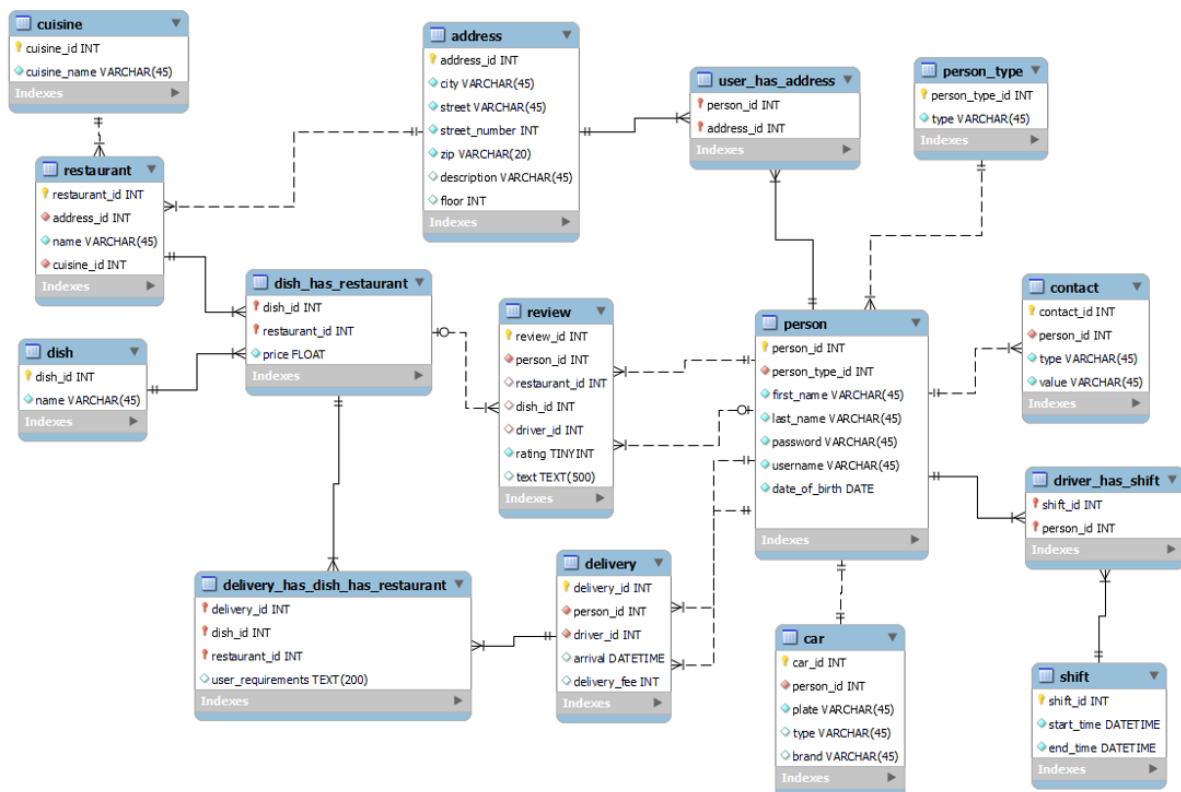
## Github repozitáře




































<https://github.com/mnecas/bds-project>

<https://github.com/natalieklimek/bds-db-design>

**3. normálová forma:** Myslíme, že databáze je v třetí normálové formě, protože všechny sloupce v jednotlivých tabulkách jsou závislé na primárních klíčích, a ne na jiném jiném sloupci.

**Celý návrh databáze vypadá takto:**



- ▼  Schemas (1)
  - ▼  bpc\_bds
    - >  Collations
    - >  Domains
    - >  FTS Configurations
    - >  FTS Dictionaries
    - >  FTS Parsers
    - >  FTS Templates
    - >  Foreign Tables
    - >  Functions
    - >  Materialized Views
    - >  Procedures
    - >  1..3 Sequences
    - ▼  Tables (15)
      - >  address
      - >  car
      - >  contact
      - >  cuisine
      - >  delivery
      - >  delivery\_has\_dish\_has\_re
      - >  dish
      - >  dish\_has\_restaurant
      - >  driver\_has\_shift
      - >  person
      - >  person\_type
      - >  restaurant
      - >  review
      - >  shift
      - >  user\_has\_address
      - >  Trigger Functions
      - >  Types
      - >  Views
    - >  Subscriptions
  - >  Login/Group Roles
  - >  Tablespaces

