ZÁPADOČESKÁ
UNIVERZITA
V PLZNI

FAV Fakulta aplikovaných věd

Databázové systémy 2

Databáze řidičů

Jméno a příjmení: Jan Tichava
Osobní číslo: A07103
Studijní skupina: čtvrtek, 4 – 5
Obor: ININ – SWIN
E-mail: jtichava@students.zcu.cz

Databázové systémy II. – KIV/DB2 – LS – 2007/2008

Zadání semestrální práce

Jméno a příjmení: **Jan Tichava** **Osobní číslo:** **A07103**
E-mail: **jtichava@students.zcu.cz**

Název práce: **Databáze řidičů**

Popis programu: Databáze obsahuje řidiče ve firmě, kteří mohou řídit určitá vozidla. Ke každému řidiči bude přiřazeno bydliště, u vozidel bude uvedeno, kde byla koupena a u řidičů i vozidel bude pobočka firmy, ke které patří.

Vyznačte software, který použijete:

- **SŘBD:** PostgreSQL
- **Klient:** JSP

1. Výčet SQL dotazů (alespoň 3, jeden agregovaný)

Seznam řidičů vybrané pobočky, včetně adres a vozidel, která mohou řídit

Vypíše pobočky a autosalony, které sídlí ve stejném městě

Průměrný věk řidiče (AVG)

2. Uložené PL/SQL procedury nebo funkce (alespoň dvě, volány v triggerech)

Počet vozidel patřící jedné pobočce

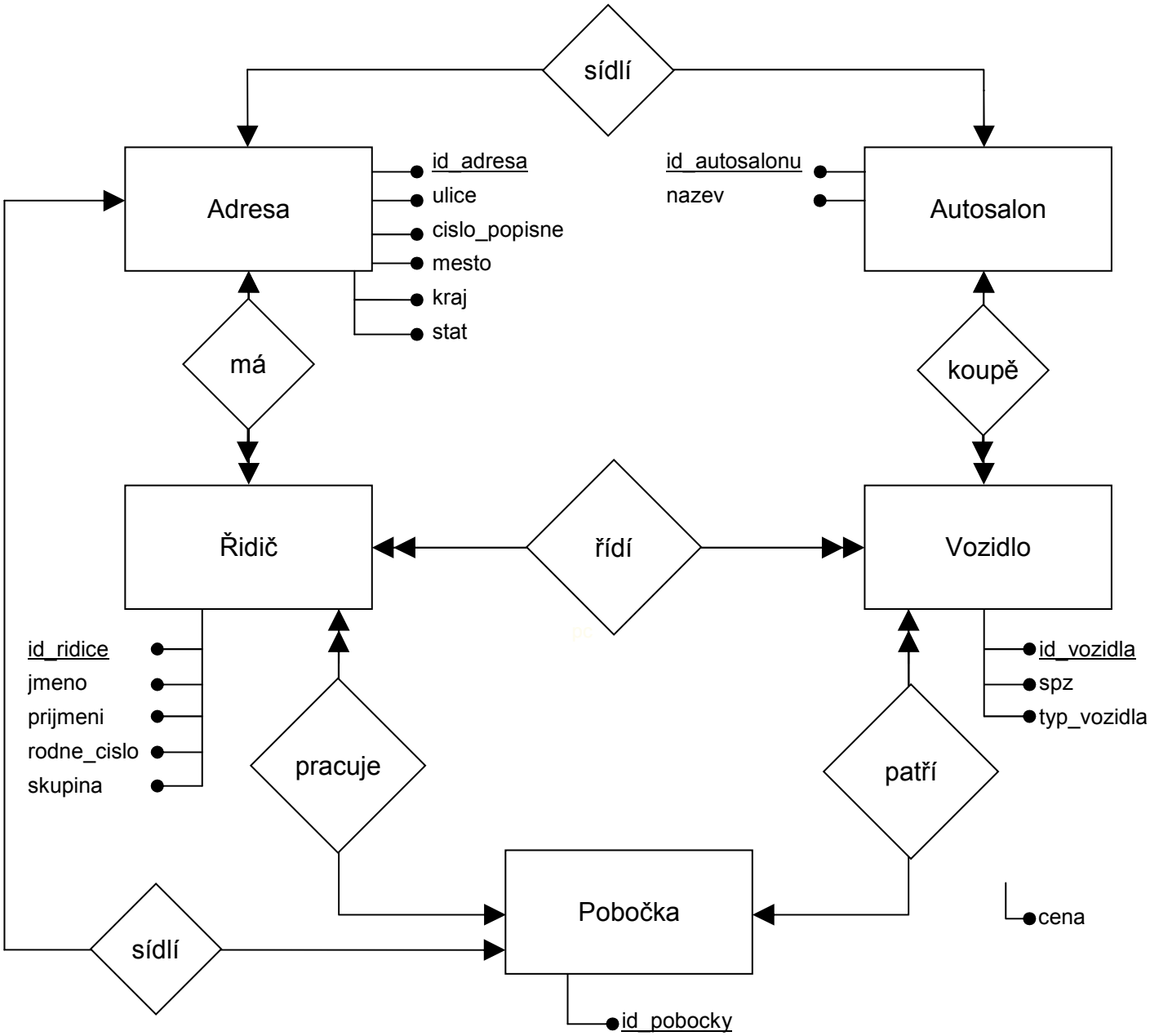
Test na jedinečnost adresy

3. Triggery (alespoň dva)

Kontrola překročení maximálního počtu vozidel

Zabrání přidání duplicitní adresy

ERA model


sídlí
Adresa
id_adresa
ulice
cislo_popisne
mesto
kraj
stat
id_autosalonu
nazev
Autosalon
má
koupě
Řidič
řídí
Vozidlo
id_ridice
jmeno
prijmeni
rodne_cislo
skupina
pracuje
patří
id_vozidla
spz
typ_vozidla
Pobočka
cena
sídlí
id_pobocky

Datová analýza

Tabulka `adresa`

Tabulka uchovává kompletní údaje o adresách. Pokud bude konkrétní adresa někde použita, nepůjde smazat. Každá adresa bude unikátní.

```
CREATE TABLE db.adresa
(
  id_adresa integer NOT NULL DEFAULT nextval('db.adresa_adresa_seq'::regclass),
  ulice character varying(50) NOT NULL,
  cislo_popisne character varying(10) NOT NULL,
  mesto character varying(50) NOT NULL,
  kraj character varying(50) NOT NULL,
  stat character varying(50) NOT NULL,
  CONSTRAINT adresa_pkey PRIMARY KEY (id_adresa)
)
```

Tabulka autosalon

Tabulka obsahuje údaje o každém autosalonu, odkazuje na konkrétní existující adresu.

```
CREATE TABLE db.autosalon
(
  id_autosalonu serial NOT NULL,
  nazev text NOT NULL,
  id_adresa integer NOT NULL,
  CONSTRAINT autosalon_pkey PRIMARY KEY (id_autosalonu),
  CONSTRAINT autosalon_id_adresa_fkey FOREIGN KEY (id_adresa)
    REFERENCES db.adresa (id_adresa) MATCH SIMPLE
    ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Tabulka pobocka

Informace o pobočce jsou uloženy v této tabulce. Podobně jako autosalon, tak také tato tabulka odkazuje na adresu.

```
CREATE TABLE db.pobocka
(
  id_pobocky serial NOT NULL,
  max_vozidel numeric(5) NOT NULL,
  id_adresa integer NOT NULL,
  CONSTRAINT pobocka_pkey PRIMARY KEY (id_pobocky),
  CONSTRAINT pobocka_id_adresa_fkey FOREIGN KEY (id_adresa)
    REFERENCES db.adresa (id_adresa) MATCH SIMPLE
    ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Tabulka ridic

Každý řidič zde musí mít záznam. Odkazuje na adresu, kde bydlí a na pobočku ve které pracuje. Rodné číslo musí být unikání.

```
CREATE TABLE db.ridic
(
  id_ridice serial NOT NULL,
  jmeno text NOT NULL,
  prijmeni text NOT NULL,
  rodne_cislo text NOT NULL,
  skupina text NOT NULL,
  id_adresa integer NOT NULL,
  id_pobocky integer NOT NULL,
  rok_narozeni integer NOT NULL,
  plat integer NOT NULL,
  CONSTRAINT ridic_pkey PRIMARY KEY (id_ridice),
  CONSTRAINT ridic_id_adresa_fkey FOREIGN KEY (id_adresa)
    REFERENCES db.adresa (id_adresa) MATCH SIMPLE
    ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT ridic_id_pobocky_fkey FOREIGN KEY (id_pobocky)
    REFERENCES db.pobocka (id_pobocky) MATCH SIMPLE
    ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT ridic_rodne_cislo_key UNIQUE (rodne_cislo)
)
```

Tabulka vazba

Slouží pouze k rozložení M:N relace. Uchovává jednotlivé řidiče a jejich vozidla.

```
CREATE TABLE db.vazba
(
  id serial NOT NULL,
  id_ridice integer NOT NULL,
  id_vozidla integer NOT NULL,
  CONSTRAINT vazba_pkey PRIMARY KEY (id),
  CONSTRAINT vazba_id_ridice_fkey FOREIGN KEY (id_ridice)
    REFERENCES db.ridic (id_ridice) MATCH SIMPLE
    ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT vazba_id_vozidla_fkey FOREIGN KEY (id_vozidla)
    REFERENCES db.vozidlo (id_vozidla) MATCH SIMPLE
    ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Tabulka vozidlo

Zde jsou vedeny záznamy o tom, kde bylo vozidlo zakoupeno a k jaké pobočce patří.

```
CREATE TABLE db.vozidlo
(
  id_vozidla serial NOT NULL,
  spz text NOT NULL,
  typ_vozidla text NOT NULL,
  id_pobocky integer NOT NULL,
  id_autosalonu integer NOT NULL,
  CONSTRAINT vozidlo_pkey PRIMARY KEY (id_vozidla),
  CONSTRAINT vozidlo_id_autosalonu_fkey FOREIGN KEY (id_autosalonu)
    REFERENCES db.autosalon (id_autosalonu) MATCH SIMPLE
    ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT vozidlo_id_pobocky_fkey FOREIGN KEY (id_pobocky)
    REFERENCES db.pobocka (id_pobocky) MATCH SIMPLE
    ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT vozidlo_spz_key UNIQUE (spz)
)
```

Funkční analýza

pc

SQL dotazy – SELECT

```
SELECT * FROM db.adresa ORDER BY id_adresa;
SELECT * FROM db.adresa WHERE id_adresa = 'ID';
SELECT * FROM db.adresa ORDER BY id_adresaů;
SELECT * FROM db.adresa WHERE id_adresa = 'ID';
SELECT * FROM db.autosalon ORDER BY id_autosalonu;
SELECT * FROM db.autosalon WHERE id_autosalonu = 'ID';
SELECT * FROM db.autosalon ORDER BY id_autosalonu;
SELECT * FROM db.autosalon WHERE id_autosalonu = 'ID';
SELECT * FROM db.pobocka ORDER BY id_pobocky;
SELECT * FROM db.pobocka WHERE id_pobocky = 'ID';
SELECT * FROM db.pobocka ORDER BY id_pobocky;
SELECT * FROM db.vozidlo ORDER BY id_vozidla;
SELECT * FROM db.vozidlo WHERE id_vozidla = 'ID';
SELECT * FROM db.vozidlo ORDER BY id_vozidla;
SELECT * FROM db.vozidlo WHERE id_vozidla = 'ID';
SELECT * FROM db.ridic ORDER BY id_ridice;
SELECT * FROM db.ridic WHERE id_ridice = 'ID';
SELECT * FROM db.ridic ORDER BY prijmeni;
SELECT * FROM db.ridic WHERE id_ridice = 'ID';
SELECT * FROM db.vazba ORDER BY id;
SELECT * FROM db.vazba WHERE id = 'ID'
SELECT mesto FROM db.adresa GROUP BY mesto;
```

Průměrný plat:

```
SELECT AVG(plat) FROM db.ridic;
```

Průměrný věk řidiče:

```
SELECT (SELECT EXTRACT(YEAR FROM TIMESTAMP 'now()'))-avg(rok_narozeni)
FROM db.ridic;
```

Všichni řidiči z konkrétní kanceláře:

```
SELECT db.ridic.id_ridice, db.vazba.id_vozidla FROM db.ridic, db.vazba
WHERE ((db.ridic.id_pobocky = 'ID') AND (db.ridic.id_ridice = db.vazba.id_ridice));
```

Všechny autosalony/pobočky v konkrétním městě:

```
SELECT id_autosalonu FROM db.adresa, db.autosalon
WHERE db.adresa.mesto = 'MESTO' AND db.adresa.id_adresa = db.autosalon.id_adresa;

SELECT id_pobocky FROM db.adresa, db.pobocka
WHERE db.adresa.mesto = 'MESTO' AND db.adresa.id_adresa = db.pobocka.id_adresa;
```

SQL dotazy – INSERT

```
INSERT INTO db.adresa VALUES (nextval('db.adresa_adresa_seq'::regclass),
'street', 'number', 'city', 'area', 'country';

INSERT INTO db.autosalon VALUES
(nextval('db.autosalon_id_autosalonu_seq'::regclass), name', 'id_address';

INSERT INTO db.pobocka VALUES
(nextval('db.pobocka_id_pobocky_seq'::regclass), 'maxCars', 'id_address'";

INSERT INTO db.vozidlo VALUES
(nextval('db.vozidlo_id_vozidla_seq'::regclass), 'numberPlate', 'type',
'office', 'motorShow'";

INSERT INTO db.ridic VALUES (nextval('db.ridic_id_ridice_seq'::regclass),
'name', 'lastName', 'boringNumber', 'group', 'address', 'office',
'birthYear', 'pay';

INSERT INTO db.vazba VALUES (nextval('db.vazba_id_seq'::regclass),
'id_driver', 'id_car';
```

SQL dotazy – DELETE

```
DELETE FROM db.adresa WHERE id_adresa = 'ID';
DELETE FROM db.autosalon WHERE id_autosalonu = 'ID';
DELETE FROM db.pobocka WHERE id_pobocky = 'ID';
DELETE FROM db.vozidlo WHERE id_vozidla = 'ID';
DELETE FROM db.ridic WHERE id_ridice = 'ID';
DELETE FROM db.vazba WHERE id = 'ID';
```

SQL dotazy – UPDATE

```
UPDATE db.adresa SET ulice = 'street', cislo_popisne = 'number',
mesto = 'city', kraj = 'area', stat = 'country' WHERE id_adresa = id;

UPDATE db.autosalon SET nazev = 'name', id_adresa = 'id_address'
WHERE id_autosalonu = id;

UPDATE db.pobocka SET max_vozidel = 'maxCars', id_adresa = 'id_address'
WHERE id_pobocky = id;

UPDATE db.vozidlo SET spz = 'numberPlate', typ_vozidla = 'type',
id_pobocky = 'office', id_autosalonu = 'motorShow' WHERE id_vozidla = id;

UPDATE db.ridic SET jmeno = 'name', prijmeni = 'lastName',
rodne_cislo = 'boringNumber', skupina = 'group', id_adresa = 'address',
id_pobocky = 'office', plat = 'pay', rok_narozeni = 'birthYear'
WHERE id_ridice = id;

UPDATE db.vazba SET id_ridice = 'id_driver', id_vozidla = 'id_car'
WHERE id = id;
```

Funkce `freespace()`

Funkce nejprve zjistí, jaký je maximální počet vozidel pro danou pobočku, pak zjistí aktuální počet vozidel, který má takto pobočka přiděleno. Na základě porovnání těchto dvou čísel se rozhodne, jestli povolí nebo zabrání vložení.

```
CREATE OR REPLACE FUNCTION db.freespace() RETURNS "trigger" AS $BODY$

DECLARE
   maxCars INTEGER;
   currentCars INTEGER;

BEGIN
   SELECT max_vozidel INTO maxCars FROM db.pobocka WHERE id_pobocky = NEW.id_pobocky;
   SELECT count(*) INTO currentCars FROM db.vozidlo WHERE id_pobocky = NEW.id_pobocky;

   IF (currentCars >= maxCars) THEN
      RAISE EXCEPTION 'Maximum vozidel';
   END IF;

   RETURN NEW;
END;
```

Funkce `existsAddress()`

Fukce vyhledá v databázi stejnou adresu, jako je zadaná, pokud existuje, zabrání vložení.

```
CREATE OR REPLACE FUNCTION db. existsAddress() RETURNS "trigger" AS $BODY$

DECLARE
  addressCount INTEGER;

BEGIN
  SELECT count(*) INTO addressCount FROM db.adresa
    WHERE ulice = NEW.ulice AND cislo_popisne = NEW.cislo_popisne
    AND mesto = NEW.mesto AND kraj = NEW.kraj AND stat = NEW.stat;

  IF (addressCount > 0) THEN
    RAISE EXCEPTION 'Adresa existuje';
  END IF;

  RETURN NEW;
END;
```

Tabulka `vozidlo`*, trigger* `beforeAddNewCar`

Trigger zavolá funkci `freespace()` před vložení nového řádku do tabulky.

```
CREATE TRIGGER "beforeAddNewCar"
  BEFORE INSERT ON db.vozidlo
  FOR EACH ROW EXECUTE PROCEDURE db.freespace();
```

Tabulka `adresa`*, trigger* `noDupliciteAddress`

Trigger zavolá funkci `existsAddress()` před vložení nového řádku do tabulky.

```
CREATE TRIGGER "noDupliciteAddress"
  BEFORE INSERT ON db.adresa
  FOR EACH ROW EXECUTE PROCEDURE db."existsAddress"();
```

Uživatelský manuál

Program je realizovaný jako webová aplikace, která je přístupná na adrese:

`http://db2.tichava.cz/`

Ovládání je intuitivní, v každém formuláři je nutné vyplnit všechna textová pole příslušnou hodnotou, pokud je některé pole prázdné nebo obsahuje zjevně chybnou hodnotu, pak nedojde k přidání do databáze (poznámka: rodné číslo musí mít minimálně 9 číslic, není kontrolované na validitu).
V zadání byl navíc agregovaný dotaz, který není uvedený v hlavním menu. Výsledek daného dotazu byl umístěn na stránku s řidiči vedle rodného čísla.

Zhodnocení a závěr

Webovou aplikaci jsem naprogramoval v Javě a v PostgresSQL. Javu jsem si zvolil proto, že je mým oblíbeným jazykem pro webové aplikace. PostgresSQL jsem si zvolil proto, že jsem se chtěl naučit něco nového, což se splnilo jen částečně. Kvůli JDBC je přístup do databáze stejný jako pro MySQL (nebo jinou databázi), ale na druhou stranu jsem se naučil pracovat s pgAdmin, který považuji za výborný nástroj.
Velice se mi líbí triggery a funkce, které zjednoduší a zautomatizují některé věci, které by jinak bylo nutné řešit programově.