# Poľnohospodársky systém

*Ján Čegiň*

*Cvičiaci: Ing. Eduard Kuric*

*Databázové systémy*

*Iterácia 1*

# 1 Zadanie

Vo vami zvolenom prostredí vytvorte databázovú aplikáciu, **ktorá komplexne rieši minimálne 6 scenárov** vo vami zvolenej doméne. Presný rozsah a konkretizáciu scenárov si dohodnete s Vašim cvičiacim na cvičení. Aplikáciu vytvoríte v dvoch iteráciach. V prvej iterácii, postavenej nad relačnou databázou, musí aplikácia realizovať tieto všeobecné scenáre:

- Vytvorenie nového záznamu,
- Aktualizácia existujúceho záznamu,
- Vymazanie záznamu,
- Zobrazenie prehľadu viacerých záznamov (spolu vybranou základnou štatistikou),
- Zobrazenie konkrétneho záznamu,
- Filtrovanie záznamov spĺňajúcich určité kritériá zadané používateľom.

Aplikácia môže mať konzolové alebo grafické rozhranie. Je dôležité aby scenáre boli realizované realisticky - teda aby aplikácia (a teda aj jej používateľské rozhranie) naozaj poskytovala časť funkcionality tak, ako by ju očakával zákazník v danej doméne.

Scenáre, ktoré menia dáta musia byť realizované **s použitím transakcií** a aspoň jeden z nich musí zahŕňať **prácu s viacerými tabuľkami** (typicky vytvorenie záznamu a naviazanie cudzieho kľúča).

V druhej iterácii do aplikácie pridáte min. 1 scenár postavený na nerelačnej databáze Redis alebo ElastiSearch (dohoda s cvičiacim na inom type nerelačnej db je samozrejme možná). Konkrétny scenár si dohodnete s vašim cvičiacim v závislosti od použitej databázy a domény vašej aplikácie (napr. štatistiky o interakciách s jednotlivými záznamami aplikácie v Redise alebo vyhľadavávanie záznamov cez Elasticsearch).

Bez odovzdanej (teda cvičiacim akceptovanej) prvej iterácie nie je možné odovzdať druhú.

Pre získanie zápočtu je potrebné odovzdať (a cvičiaci musí akceptovať minimálnu úroveň kvality) obidve iterácie projektu.

# 2 Špecifikácia scenárov

Aplikácia realizuje v danej doméne nasledovné scenáre a to:

- Pridanie nového zamestnanca do databázy
- Pridanie nového zvieraťa do databázy
- Priradenie zvieraťa do chlieva
- Odstránenie zvieraťa z chlieva podľa typu zvieraťa a jeho zadanej váhy
- Nájdenie zvierat podľa viacerých filtrov
- Priradenie zamestnanca na prácu na poli či chlieve
- Priradenie traktoru zamestnancovi na prácu na poli, ak na poli pracuje
- Nájdenie všetkých pracovných záznamov zamestnanca s detailmi
- Zobrazenie štatistík a to:
    - Priemerný plat zamestnanca
    - Najčastejšie meno zvierat
    - Obsadenosť chlievov
    - Množstvo dostupného krmiva
    - Počet rôznych druhov zvierat v chlievoch

# 3 Logický a Fyzický dátový model

Poľnohospodársky systém, ktorý aplikácia obsluhuje v sebe obsahuje zamestnancov, ktorých údaje sú meno, priezvisko, plat, pozícia, dátum, kedy bol zamestnanec prijatý, resp. pridaný do systému. Jeho primárnym kľúčom je id, ktoré sa mu automaticky prideľuje pri pridelení.

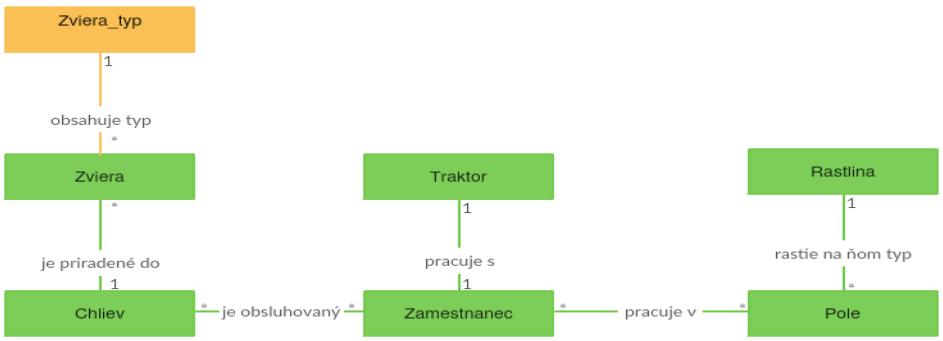
Ďalšou entitou je zviera, ktoré má svoje meno, váhu a id, ktoré sa tiež automaticky prideľuje. Obsahuje aj cudzie kľúče, od entít chliev a zviera_typ, podľa čoho vieme určiť, o aký typ zvieraťa sa jedná a v akom chlievy je priradené, ak vôbec je.

Entita chliev v sebe zahŕňa kapacitu, deň postavenia chlieva a jeho stav. Taktiež má svoje id, ktoré predstavuje jeho fyzické označenie. Chlievy z dôvodu, že sa často nemenia, v našich scenároch nepridávam ani nijako inak nemením.

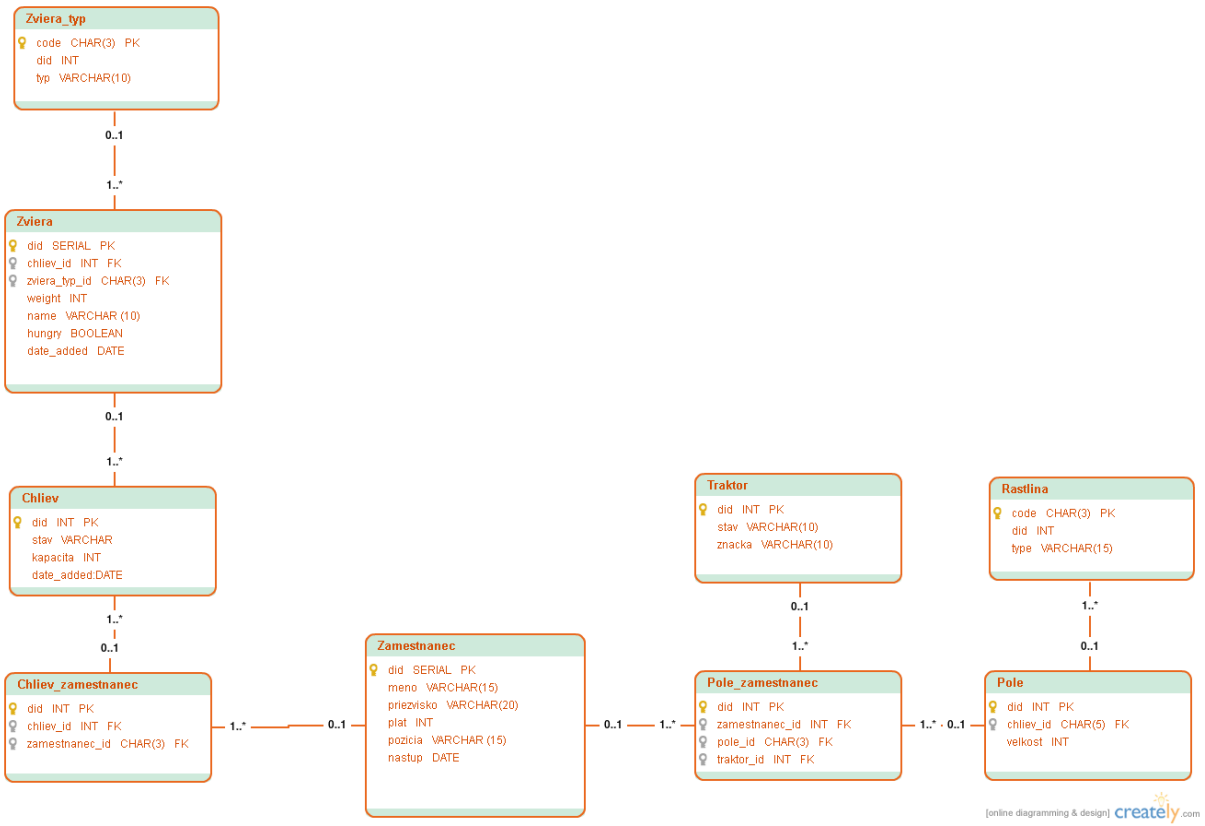
Podobnou entitou je pole, ktorá v sebe zahŕňa údaje o kapacite poľa, type rastliny (krmiva), ktoré sa na ňom pestuje, čo zodpovedá cudziemu kľúču rastlina_id, kde sú detaily o jednotlivých rastlinách. Rastliny pestované na poliach slúžia ako krmivo pre zvieratá. Entita pole, podobne ako entita chliev, má svoje id, ktoré odpovedá označeniu v dokumentoch poľnohospodárskej spoločnosti a preto, že sa často nemení, tak sa v scenároch nepridáva ani nijako inak nemení.

Entita traktor v sebe zahŕňa stav traktoru a jeho značku, teda značku výrobcu traktoru. Táto entita má rovnaký prípad ako entity pole a chliev, teda jeho id odpovedá fyzickému označeniu traktora pri evidencii vozidiel a túto entitu nemením.

Posledné dve tabuľky v databáze predstavujú väzobné entity pre vzťah many-to-many a sú to pole_zamestnanec a chliev_zamestnanec. Chliev_zamestnanec obsahuje id zamestnance, ktorý pracuje v chlievy a id chlievu, v ktorom pracuje zamestnanec. Zamestnanec môže pracovať a mať na starosti rôzne polia a chlievy, preto sú potrebné tieto väzobné tabuľky. Rozdiel v tabuľke pole_zamestnanec je ten, že zamestnanec, ak pracuje na poli, môže, ale aj nemusí na ňom pracovať s traktorom. Jeden zamestnanec môže pracovať práve s jedným traktorom. Preto obsahuje stĺpec traktor_id, kde ak zamestnanec s traktorom pracuje, tak sa to zobrazí na každom zázname pre neho platnom.




Obr.1 Logický dátový model




Obr.2 Fyzický dátový model

# 4 Opis návrhu a implementácie

Implementácia bola naprogramovaná v programovacom jazyku JAVA v prostredí Eclipse MARS 4.1. Pri implementácii boli využité prvky OOP. GUI je urobené cez prostredie JAVA FX, čo je Smart Client. Použitá databáza je PostgreSQL 9.5. Údaje na prístup do databázy sú – meno: postgres, heslo: chidorinagashi, port: 5121 (z dôvodu už bežiacej inej databázy na porte 5432).

K databáze sa pristupuje pomocou objektu Connection, ktorý sa vytvorí ako prvý. Následne sa vytvorí objekt Statement pomocou metódy createStatement() inštancie objektu Connection. Inštancia tohto objektu následne vykoná zadanú query a výpis sa uloží ako inštancia Objektu ResultSet. Následne sa výstup naparsuje podľa zvolených požiadaviek.

Po úspešnej exekúcii sa zavrie Connection, Statement aj ResultSet. V aplikácii sú použíte transakcie, autocommit je deaktivovaný. Transakcie sú využité najmä ak robím update či inú operáciu, ktorá mi záleži na výsledku iného selectu. Ak ten nespĺňa kritéria, alebo nastane SQLError, tak dôjde k rollbacku a zmeny nie sú komitnuté.

Pri implementovaní jednotlivých scenárov sa dbalo nato, aby používateľ zadal hodnoty požadované pre úspešné operácie. Ak sa operácia týka ID, tak sa dbá nato, aby ak ID objektu neexistuje, operácia neprebehne a dôjde k rollbacku.

Čo sa týka splnenia jednotlivých kritérií na zadanie tak tie sú: Agregačné funkcie - COUNT je použitý pri štatistikách ako aj pri výskyte daného ID v podmienkach, AVG je použité v štatistike priemerného platu, MAX je použité pri štatistike najčastejšieho mena zvierat. JOIN je využitý na viacerých miestach kde je potrebná štatistika či nájdenie viacerých údajov z rôznych tabuliek. GROUP BY sa využíva pri štatistikách.

**Zdroje**

http://www.tutorialspoint.com/postgresql/postgresql_java.htm

http://zetcode.com/db/postgresqljavatutorial/

http://www.postgresql.org/docs/9.5/static/docguide.html