Dodávateľ plynu

Záverečná správa

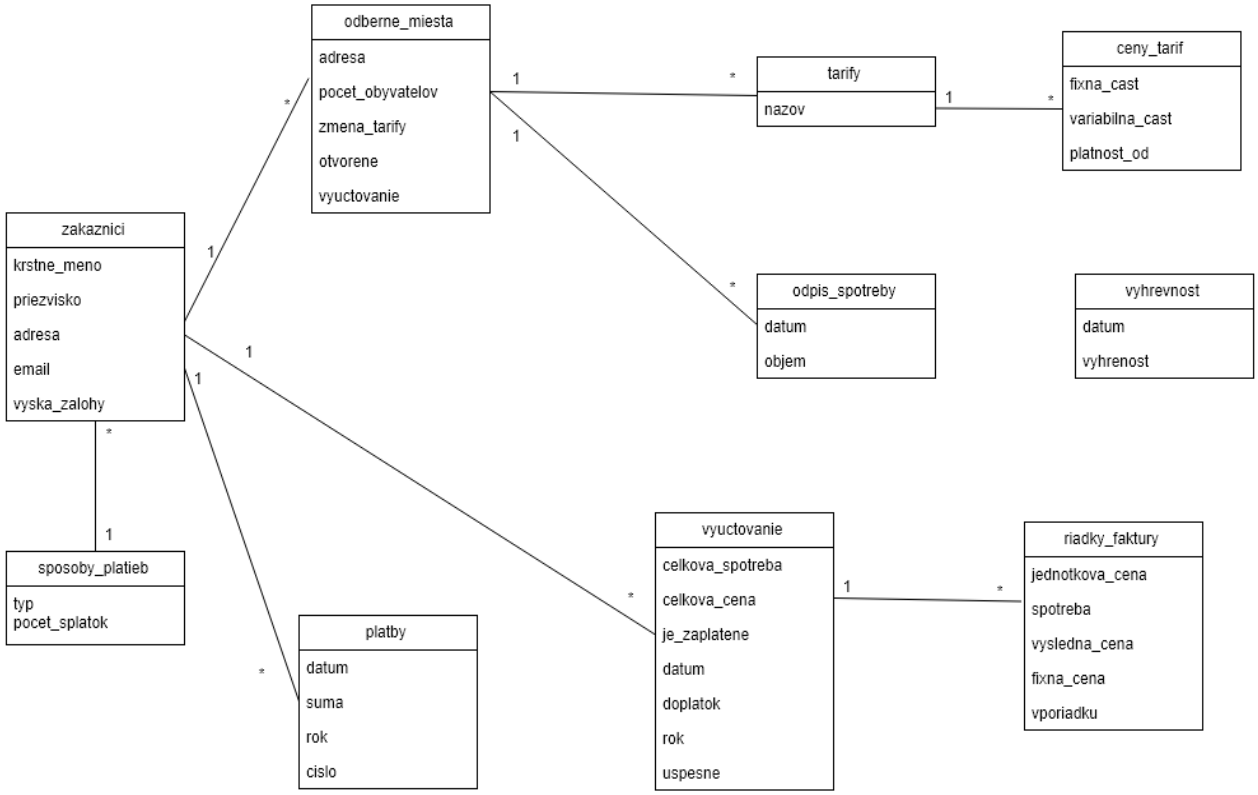
Databázy(2)

Daniel Kyselica 17. mája 2017

1 O dokumente

Tento dokument popisuje výsledný systém dodávateľa plynu, ktorý sa zaoberá evidovaním zákazníkov, odberných miest a zaznamenávaním ich spotreby, platieb.

2 Dátový model



Obr 1: Entitno relačný model dát dodávateľa plynu

Systém si uchováva údaje o zákazníkoch (**zakaznici**). Zákazníci si na začiatku zvolia spôsob platby **(sposob_platby**) – ročne, polročne štvrťročne. Zákazník má prihlásené odberné miesta, o ktorých sú informácie uložené v tabuľke **odberne_miesta**. Odberné má určenú tarifu, ktoré sú uložené v tabuľke **tarify.** Táto tarifa môže byť raz ročne zmenená – atribút zmena_tarify. Tarify majú určené ceny uložené v tabuľke **ceny_tarif**, ktoré sa skladajú z dvoch zložiek fixnej a variabilnej časti platenej za kwh. Taktiež obsahujú datum začatia platnosti ceny. Na určenie koľko kwh jedného metra kubického plynu slúži tabuľka **vyhrevnost**, kde sú uložené merania výhrevnosti plynu. Aktuálna spotreba odberných miest je uložená v **odpis_spotreby.** Všetky platby zákazníkov sú zaznamenávané (**platby**). Každá platba aj nesie informáciu o tom za aké obdobie mala byť zaplatená. Na konci vyúčtovacieho obdobia je zákazníkovi vytvorená faktúra, uložená v tabuľke **vyuctovanie.** Ak neexistuje dostatočný počet meraní je to zaznamenané. Taktiež si uchováva informácie o doplatku resp. nedoplatku. Keďže sa cena plynu môže počas vyúčtovacieho meniť jednotlivé ceny a spotreba za dané obdobia sú uložené v tabuľke **riadky_faktury**.

3 Zoznam funkcií

Systém umožňuje, pridávať, upravovať, mazať, zobrazovať informácie

- Zákazníkov
- Odberných miest
- Taríf a ich cien
- Meraní (spotreby alebo výhrevnosti)

Ďalej systém

Simuluje operácie týkajúce sa odberných miest a to

- Zriadenie - kde sa okrem pridania odberného miesta vytvorí aj počiatočné meranie spotreby pre dané miesto
- Uzavretie – Odberné miesto sa uzavrie a teda v tom momente sa končí jeho vyúčtovacie obdobie a už sa nové neotvorí
- Zmena tarify – túto operáciu je vždy možné vykonať maximálne jedenkrát ročne, pričom táto nová tarifa bude pri výpočte použitá pre celé vyúčtovacie obdobie

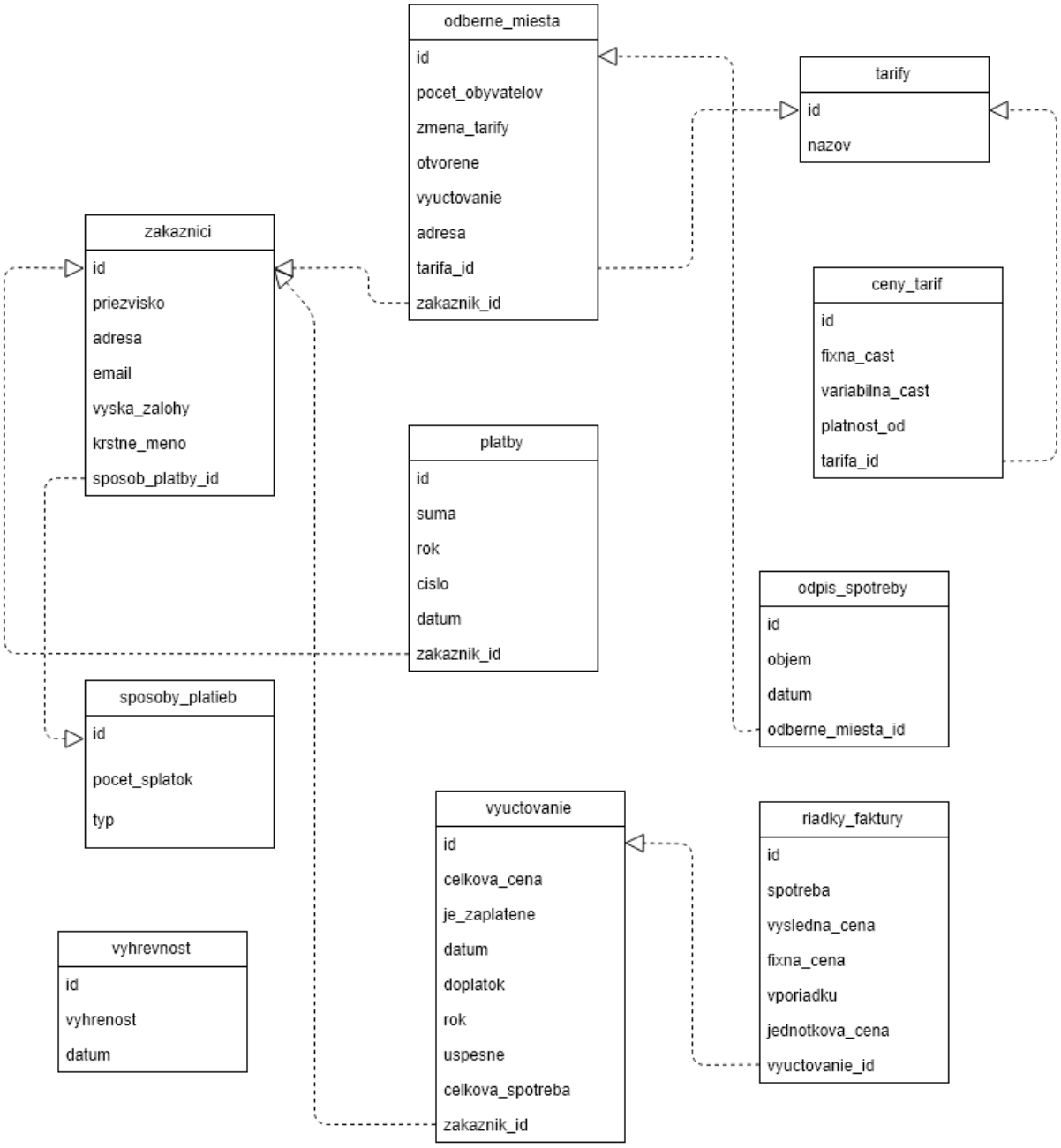
Vyúčtovanie – Fakturácia

- Systém pre všetky časové úseky v ktorých sa cena pre jednotlivé jeho odberné miesta cena menila vytvorí riadky faktúry, ktoré zodpovedajú týmto obdobiam a zaznamenávajú spotrebu a cenu za plyn v tomto období
- Zobrazí informácie o celkovej spotrebe, cene, súčte platieb a doplatku resp. nedoplatku za dané fakturačné obdobie (obyčajne kalendárny rok)

Systém ďalej zobrazuje nasledovné štatistiky

- Porovnanie spotreby – zákazníkovi ukáže ako sa mení a menila spotreba v jeho odberných miestach voči minulosti ale aj voči ostatným odberným miestam. Tieto informácie sa zobrazia pre každý rok a každé odberné miesto zákazníka.
- Oneskorení platcovia – zoznam obsahujúci iba zákazníkov, ktorý svoje platby či už zálohové alebo nedoplatkové realizovali aspoň mesiac po dohodnutom termíne.

4 Relačná databáza



Obr. 2: Relačný model dát systému dodávateľa plynu

5 Organizácia kódu

Aplikácia je naprogramovaná v jazyku Java. Využíva vzory Row Data Gateway a Transaction Script. Prístup do databázy je riešený cez JDBC. Zdrojový kód je rozdelený do nasledovných balíkov.

Main

Tento balík obsahuje Triedu ***Main***, cez ktorú sa spúšťa aplikácia. Trieda ***DbContext*** sa stará o spojenie s databázou ktoré je udržiavané počas celého chodu aplikácie.

Gateway

Tento balík obsahuje pre každú tabuľku v databáze Data Gateway – iba samotný ***Gateway***. Opakujúci sa kód je vyčlenený do tried ***BaseGateway*** od ktorých ostatné triedy dedí (túto triedu som použil zo vzorového riešenia).

Finder

Tento balík obsahuje druhú časť Data Gateway pre každú tabuľku a to ***Finder***. Všetky findre sú podtriedy ***BaseFinder***, ktorá nesie v sebe opakujúci sa kód (táto trieda taktiež bola použitá zo vzorového projektu).

Printers

Tento balík obsahuje triedy návrhového vzoru ***singleton***, ktoré sa starajú o zobrazenie danej triedy v používateľskom rozhraní. Používateľské rozhranie môže byť zmenené preto používam printre a nie jednoduché funkcie ako toString.

Exceptions

Tento balík obsahuje mnou vytvorené výnimky, ktoré lepšie popisujú možné vzniknuté situácie

ComplexOperations

Tento balík sa stará o ***Databázové tranzakcie***, teda o zložitejšie doménové operácie.

UI

Tento balík obsahuje triedy pre hlavné menu a podmenu, ktoré tvoria užívateľské rozhranie

6 Optimalizácia SQL

Pri cieľom množstve dát bola štatistika spotreby (script je v prílohe) zákazníka veľmi pomalá

Pseudokód:

Pre každý rok a každé odberné miesto zákazníka:

- Vypočítaj spotrebu za tento rok

Pre každý rok:

- Vypočítaj priemernú spotrebu odberných miest

Pre všetky získané údaje:

- Vypíš spotrebu miesta za rok, rok, rozdiel voči priemernej spotrebe všetkých miest
- Ak existuje záznam pre toto miesto o rok starší vypíš rozdiel v spotrebe

Preto som ju prerobil cez with konštrukciu aby databáza optimalizovala čo sa dá.

```
WITH om_spotreba_za_rok AS (
     SELECT
          om.id as id,
        om.adresa as adresa,
        om.pocet_obyvatelov as pocet_obyvatelov,
        om.zmena_tarify as zmena_tarify,
        om.otvorene as otvorene,
        om.vyuctovanie as vyuctovanie,
        om.zakaznik_id as zakaznik_id,
        om.tarifa_id as tarifa_id,
        sum(os.objem -
           (SELECT os2.objem FROM odpis_spotreby as os2
           WHERE os2.odberne_miesta_id = om.id AND os2.datum <
os.datum ORDER BY datum DESC LIMIT 1) )
          as spotreba,
        seq.i as rok

    FROM odberne_miesta as om JOIN odpis_spotreby as os ON om.id =
os.odberne_miesta_id
    JOIN generate_series(1,date_part('year',now())::INTEGER) as
seq(i) ON seq.i = date_part('year', os.datum)
    WHERE om.zakaznik_id  = 105
    GROUP BY om.id, seq.i
    ORDER BY om.id, rok ASC
),
primerna_spotreba_rocna AS (
     SELECT
        avg(os.objem) as priemer,
        seq.i as rok
```

```
    FROM odberne_miesta as om JOIN odpis_spotreby as os ON om.id =
os.odberne_miesta_id
    JOIN generate_series(1,date_part('year',now())::INTEGER) as
seq(i) ON seq.i = date_part('year', os.datum)
    GROUP BY seq.i
    ORDER BY rok
)
SELECT
     om1.id,
     om1.adresa,
     om1.pocet_obyvatelov,
     om1.zmena_tarify,
     om1.otvorene,
     om1.vyuctovanie,
     om1.zakaznik_id,
     om1.tarifa_id,
     CASE WHEN om1.spotreba IS NOT NULL THEN om1.spotreba
     ELSE 0 END as spotreba,
     om1.rok,
     CASE WHEN om1.spotreba - om2.spotreba IS NOT NULL
     THEN om1.spotreba - om2.spotreba
     ELSE 0
     END AS rocny_rozdiel,
     CASE
          WHEN om1.spotreba - psr.priemer IS NOT NULL
          THEN om1.spotreba - psr.priemer
          ELSE 0
     END as rozdiel_voci_priemeru

FROM om_spotreba_za_rok as om1 LEFT JOIN om_spotreba_za_rok as om2
ON om1.id = om2.id AND om1.rok = om2.rok +1
     JOIN primerna_spotreba_rocna as psr ON psr.rok = om1.rok;
```

Takto som presunul všetok výpočet s javy do SQL čím som operáciu značne urýchlil.

Avšak aj po tejto úprave bola vykonanie dopytu veľmi pomalé preto som použil index na urýchlenie, ktorý výrazne zvýšil rýchlosť vykonania dopytu.

Veľkosť tabuľky odpis_spotreby (v riadkoch)	Čas spracovania dopytu PRED optimalizáciou (ms)	Čas spracovania dopytu PO optimalizáciou (ms)
10 000	1 350	164
100 000	18 633	782
1 000 000	126 262	7100

7 Vybraný riešený problém

V pôvodnom návrhu som nepoužil tabuľku riadkov faktúry. Keď zákazník chcel zistiť akú ma spotrebu, koľko platil, aká bola cena plynu za určité obdobie musel som tieto informácie získať pomerne zložitým dopytom z viacerých tabuliek (platby, odpis_spotreby, vyhrevnost, cena_tarif, tarify). Tento dopyt je

už raz vykovaný pri vyúčtovaní preto som tieto vypočítané, získané dáta uložil do novej tabuľky riadky_faktury. Týmto som dokázal ušetriť veľa času aby som nemusel vždy pri prezeraní faktúry znova a znova vypočítavať dáta ale stačí ich prečítať už iba z jednej tabuľky.