Doručovanie

Záverečná správa

Projekt z predmetu Databázy (2)

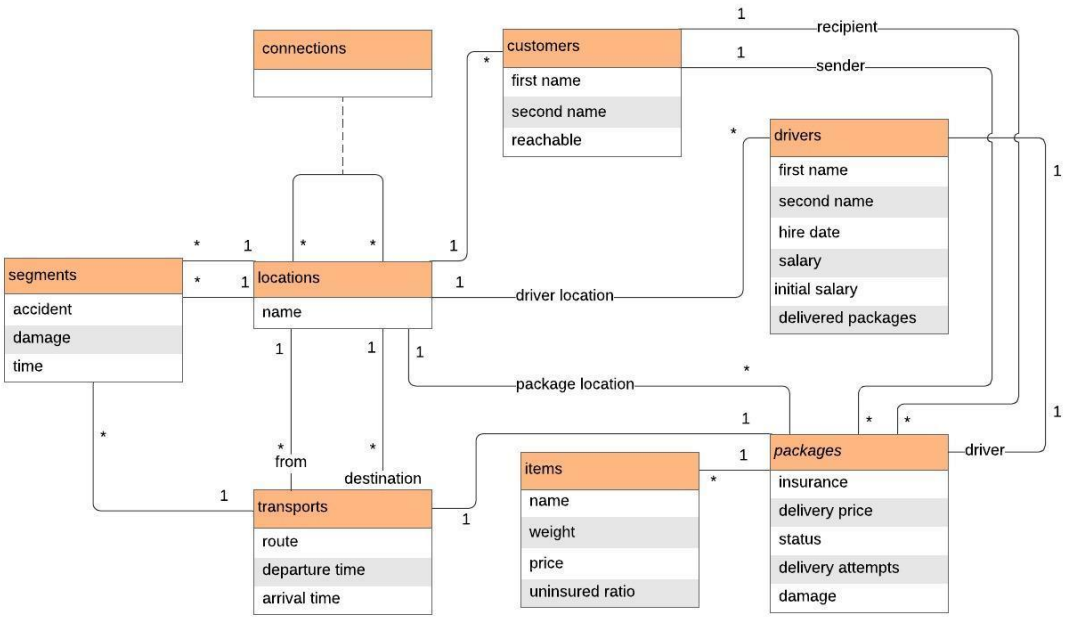
Andrej Paluch

10.5.2020

1 O dokumente

Tento dokument je záverečná správa k projektu z predmetu Databázy (2). Opisuje vytvorenú databázovú aplikáciu, ktorá simuluje doručovaciu spoločnosť t.j. eviduje si zákazníkov, zamestnancov, balíky, položky v nich aj samotné prepravy balíkov. Okrem samotného doručovania balíkov sa projekt snaží simulovať aj situácie ako nehody, alebo nezastihnutie adresáta, ktoré sa v bežnom živote stávajú.

2 Dátový model



Obrázok 1. Entitno relačný model dát doručovacej spoločnosti

Systém (Obr. 1) si uchováva zákazníkov v množine **customers**, kde si okrem základných údajov (meno, priezvisko) zaznamenáva aj to, či je zákazník zastihnuteľný.

Svojich zamestnancov (vodičov) si spoločnosť uchováva v množine **drivers**, o každom vodičovi si eviduje meno, priezvisko, dátum prijatia do zamestnania, plat, počiatočný plat, počet doručených balíkov, lokácia, v ktorej sa aktuálne nachádza a balík, ktorý práve prevára. Množiny vodičov a balíkov sú vo vzťahu 1:1, keďže vodič môže prepravovať iba jeden balík naraz, Množiny vodičov a lokácií sú vo vzťahu *:1, keďže spoločnosť môže zamestnávať veľa vodičov a kľudne sa môže stať že je ich viac v tej istej lokácií.

V množine **locations** sú uložené všetky lokácie, lokácie môžu byť medzi sebou vo vzťahu, to predstavuje množina **connections** (znamená že tie dve lokácie sú prepojené). Množina lokácií je vo vzťahu 1:* s inými množinami, to znamená, že objekty z tých množín sa nachádzajú na tej ktorej lokácií (alebo sa viažu na danú lokáciu).

V množine **packages** sú uložené balíky, o každom balíku sa eviduje: čo je poistený, cena za doručenie, počet pokusov o doručenie, prípadne poškodenie a stav. Stav môže byť :

- new – novo zaevidovaný balík, ešte sa v ňom nenachádzajú položky

- ready – už sa v ňom nachádzajú položky, cena za doručenie je vypočítaná, je pripravený na doručenie
- on_way – je práve na ceste, t.j. nejaký vodič ho doručuje
- delivered – už doručený balík
- damaged – úplne poškodený balík
- company_failed – balík, ktorý sa nepodarilo doručiť a prepadol spoločnosti
- refunded – úplne poškodený, alebo čiastočne poškodený doručený balík, ktorý už bol refundovaný

z balíku je uvedený aj odosielateľ a príjemca (množina customers), balík ma vždy len jedného odosielateľa a jedného príjemcu, preto sú množiny packages a customers vo vzťahu *:1

v množine **items** sa evidujú všetky položky, ktoré môžu (ale ešte nemusia) byť pridelené k balíkom. Každá položka má svoje meno, hmotnosť, cenu a podiel (v percentách) z ceny, ktorý sa vráti odosielateľovi v prípade poškodenia balíku. V jednom balíku môže byť viac položiek, položka však nemôže byť vo viacerých balíkoch, preto sú množiny items a packages vo vzťahu *:1.

transporty sú uložené v množine **transports**. Každý transport má svoju cestu, po ktorej sa balík prepravuje, čas odchodu a príchodu. Každý transport sa skladá zo segmentov (uložených v množine **segments**). Segment je úsek trasy (dvojica lokácií), pamätá si čas, to či sa stala nehoda a miera poškodenia balíka, ak sa nehoda stala. Množiny segmentov a transportov sú vo vzťahu *:1.

3 Relačná databáza

Obr. 2 predstavuje výslednú relačnú databázu, ktorá vznikla transformovaním entitno relačného modelu z Obr. 1.

4 Organizácia kódu

Aplikácia je naprogramovaná v jazyku Java. V aplikácií sú použité vzory Row Data Gateway a Transaction Script. Prístup do databázy je riešený pomocou JDBC. Zdrojový kód je rozdelený do nasledovných priečinkov.

java

Tento priečinok obsahuje triedu Main, cez ktorú sa aplikácia spúšťa. Aplikácia má po celý čas vytvorené jedno spojenie s databázovým systémom, ktoré sa udržiava v triede DbContext. Okrem toho sa v tomto priečinku nachádzajú aj všetky nasledujúce priečinky.

java.row_data_gateway

Tento priečinok obsahuje pre väčšinu tabuliek v databáze Data Gateway. Row Data Gateway je realizovaný dvojicou tried - jedna pre Gateway a druhá pre Finder. Tabuľky connections, locations a customers nie sú mapované do Row Data Gateway, keďže to nebolo potreba.

java.complex_operations

Tento priečinok obsahuje triedy zložitejších doménových funkcií. Je realizovaný pomocou vzoru Transaction Script. Každá zložitejšia doménová operácia sa nachádza na samostatnom súbore.

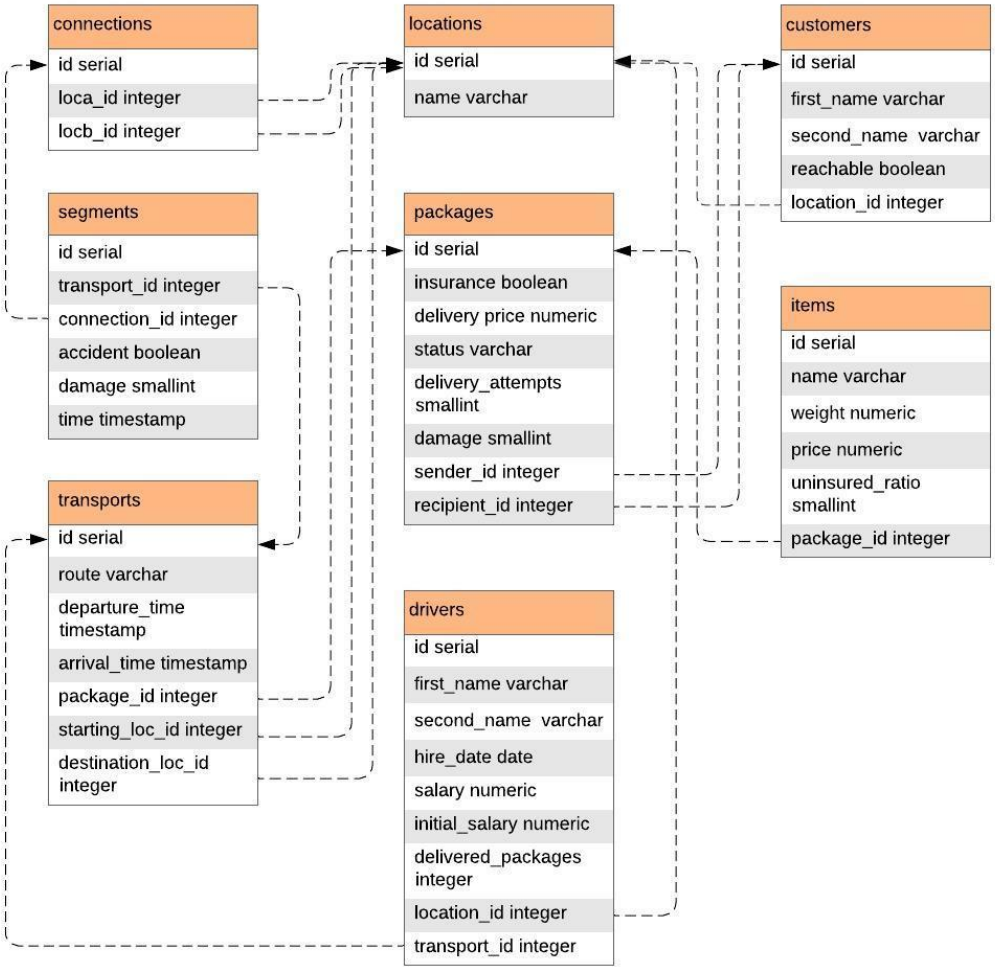
java.user_interface

Tento priečinok obsahuje kód používateľského rozhrania. Ide o jednoduché používateľské rozhranie ovládané z príkazového riadka. Rozhranie funguje spôsobom: používateľ si vyberie nejakú z ponúknutých možností zadaním jej čísla. Každé menu je realizované samostatnou triedou. To čo majú všetky menu súbory rovnaké sa nachádza v triede menu.java, ktorá je abstraktná a ostatné menu triedy od nej dedia. Aby hlavné menu nebolo príliš neprehľadné, je rozdelené na 5 menu súborov:

- MenuDriver – CRUD operáciu s vodičom
- MenuItem – CRUD operáciu s položkami
- MenuPackage – CRUD operáciu s balíkmi
- MenuTransport – CRUD operácie s transportmi
- MenuOther – menu pre zložitejšie doménové operácie

Java.db_context

Tu sa nachádza DbContext, ktorý udržiava spojenie z databázou, a súbor MyExceptions kde sú definované výnimky, ktoré používam v projekte.



Obrázok 2. Relačný model dát doručovacej spoločnosti

5 Optimalizácia SQL

Pri hľadaní trasy, po ktorej sa prepraví balík, som použil rekurzívnu funkciu (názorný príklad pre hľadanie trasy medzi najvzdialenejšími lokáciami)

```
WITH RECURSIVE paths(src,dst,path) AS (
    SELECT e.loca_id, e.locb_id, ARRAY[e.loca_id,e.locb_id]
    FROM connections e
    UNION
    SELECT p.src, e.locb_id, p.path || ARRAY[e.locb_id]
    FROM paths p JOIN connections e ON p.dst = e.loca_id AND e.locb_id != ALL(p.path)
)
SELECT path FROM paths where src = 1 and dst = 49 limit 1;
```

Optimalizácia, keďže hľadáme konkrétnu cestu, zbytočne vyberáme všetky dvojice z connections v prvom SELECTE rekurzívnej funkcie, stačí keď vyberieme tie, ktoré obsahujú našu začiatočnú lokáciu.

```
WITH RECURSIVE paths(src,dst,path) AS (
    SELECT e.loca_id, e.locb_id, ARRAY[e.loca_id,e.locb_id]
    FROM connections e where  e.loca_id = 1
    UNION
    SELECT p.src, e.locb_id, p.path || ARRAY[e.locb_id]
    FROM paths p JOIN connections e ON p.dst = e.loca_id AND e.locb_id != ALL(p.path)
)
SELECT path FROM paths where  dst = 49 limit 1;
```

Výsledok,

Pre trasy dĺžky 7 a menej skoro nepoznať rozdiel, no z tabuľky možno spozorovať, že neoptimalizovaná verzia rastie exponenciálne z každou ďalšou o jedna dlhšou trasou, zatiaľ čo optimalizovaná verzia rastie iba lineárne. Pri trase dlhej 13 je rozdiel viac ako o jeden rád.

(dĺžka cesty) *cesta*	*rýchlosť pred optimalizáciou*	*rýchlosť po optimalizácií*
(2) 1-2	267ms	236ms
(3) 1-8-9	223ms	251ms
(4) 5-12-19-26	254ms	222ms
(5) 10-17-24-23-30	220ms	205ms
(6) 15-16-23-24-25-26	254ms	220ms
(7) 20-19-18-17-24-23-30	252ms	221ms
(8) 15-16-23-30-31-32-33-34	303ms	205ms
(9) 8-9-16-23-24-25-26-27-28	393ms	227ms
(10) 1-8-9-16-17-24-25-32-33-40	536ms	235ms
(11) 9-10-17-18-25-26-33-34-35-42-49	1148ms	239ms
(12) 49-48-41-34-27-20-13-12-11-10-9-8	5205ms	247ms
(13) 43-44-37-30-23-16-9-2-3-4-5-6-7	8332ms	253ms

6 Vybraný riešený problém

Zvýšenie výplaty zamestnancom. Pôvodne som plánoval spraviť spúšťač, ktorý by zvýšil výplatu vodičovi vždy keď by doručil 10000 balíkov, avšak výplata sa ma zvýšiť aj po jednom odpracovanom roku, a vytvoriť spúšťač, ktorý by sa spúšťal v pravidelných časových intervaloch a kontroloval by či už prešiel ďalší rok. Okrem toho, operácia sa ma vykonávať plošne pre všetkých vodičov, nie pre každého zvlášť.

Riešenie, upravil som tabuľku drivers pridaním stĺpca initial_salary, takže vždy pri zavolaní operácie pay rise sa pre každého vodiča nanovo vypočíta jeho mzda jednoduchým matematickým výrazom.