Systém na predaj lístkov na zimnom štadióne

Projekt na predmet Databázy (2)

Správa (1)

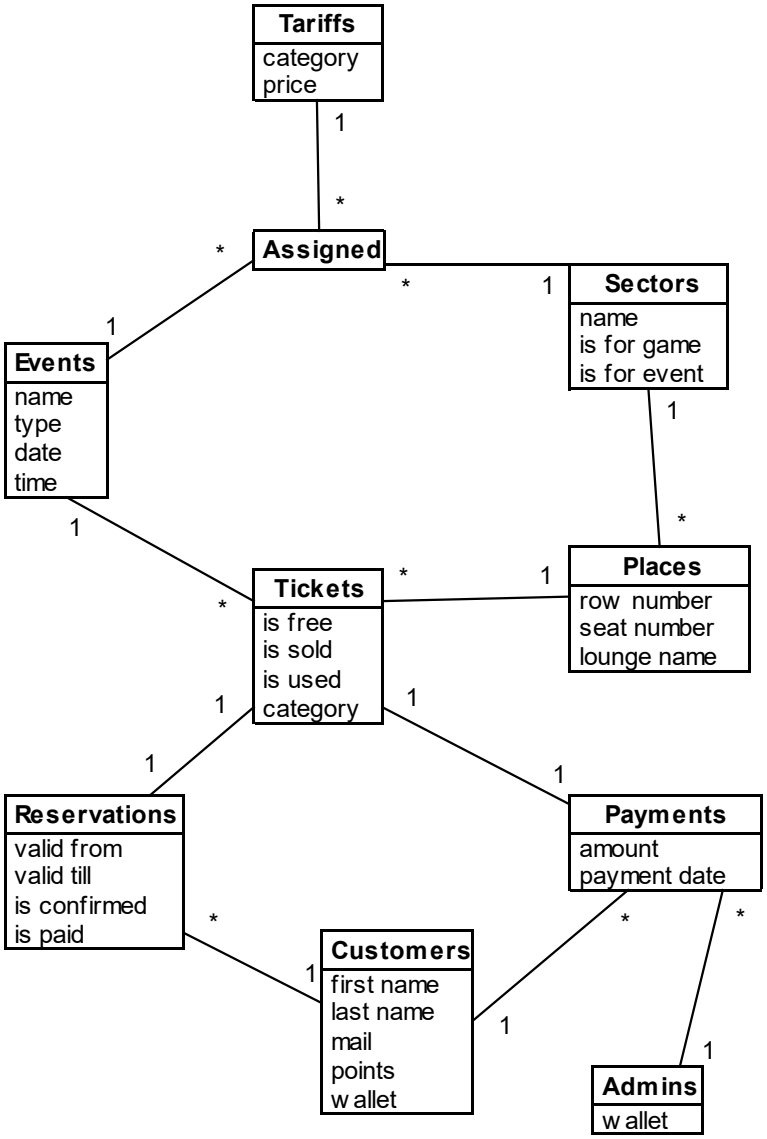
Jozef Bača

10. mája 2020

1. O Dokumente

Tento dokument je prvou verziu záverečnej správy o projekte predmetu Databázy (2). Zaoberá sa vytvoreným systémom na predaj a rezerváciu lístkov a evidovaním zákazníkov a podujatí na zimnom štadióne.

2. Dátový model



Entitno relačný model databázy systému.

Systém si uchováva informácie o zákazníkoch v množine ***Customers***. Na to, aby si zákazník pomocou svojho konta kúpil lístok,

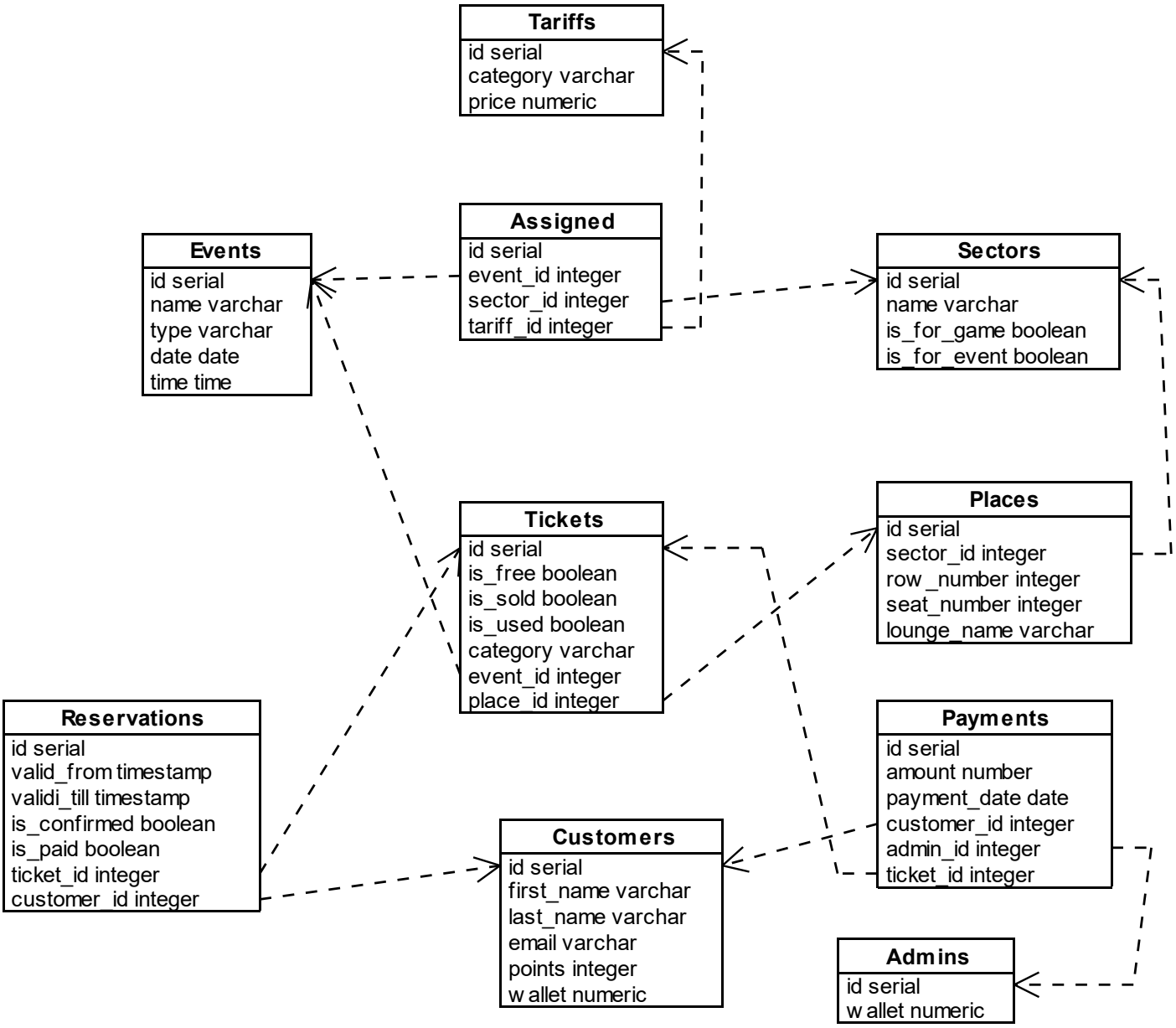
musí prebehnúť rezervácia, a preto sa ***Customers*** neviaže na lístky, ale na rezervácie uložené v množine ***Reservations***. Až tá sa viaže na množinu ***Tickets***.

Množina *Payments* si uchováva informácie o platbách za lístky. Sumy za lístky prechádzajú na konto jedného z adminov, uložených v množine ***Admins***.

Podujatia sa uložia v množine ***Events***. Jednou z atribút je *Type*, ktorá hovorí o type eventu. Môžu to byť hodnoty podujatie, zakladna cast alebo play-off.

Miesta na štadióne sú rozdelené do sektorov v množine ***Sectors***. V každom sektore môže byť ľubooľný počet miest uložených v množine ***Places***. Buď to môžu byť sedadlá v hľadisku alebo VIP lóže. Ceny lístkov sa ukladajú v množine ***Tariffs***.

3. Relačný model



Relačný model databázy systému

Relačný model vznikol transformáciou entitno relačného modelu.

4. Organizácia kódu

Projekt je naprogramovaný v jazyku *Java*. Prístup do databázy je riešený pomocou JDBC. Využíva vzory *Row Data Gateway* a *Transaction script*. Kód je rozdelený do balíkov:

main: Obsahuje triedu DBContext, kde si aplikácia udržiava jedno vytvorené spojenie s databázou, a triedu Main, kde sa aplikácia spúšťa.

rdg: Obsahuje triedy, pomocou ktorých sa realizuje *Row Data Gateway* databázy. Každej *Gateway* je vytvorená trieda *Finder*.

ui: Abstraktná trieda Menu je základom používateľského prostredia. Pomocou dedenia sú všetky ostatné triedy tvoriace používateľské prostredie jej podmnožinou.

Exceptions: Obsahuje všetky potrebné výnimky.

Transactions: Obsahuje triedy s transakciami podľa vzoru *Transaction script*, ako aj ďalšie metódy narábajúce s dátami databázy.

sql: Obsahuje create_script a generate_script na vygenerovanie databázy a testovacích dát

5. Optimalizácia

Pravidelná kontrola systému bola od začiatku pomalá. Robil som ju jedným sql príkazom:

```
UPDATE tickets
SET is_used = true
FROM tickets ti INNER JOIN events e ON (e.id = ti.event_id)
WHERE ti.is_used = false AND e.date < (SELECT now());
```

To bolo veľmi pomalé. Ako riešenie som zvolil rozdeliť tento príkaz na dva osobitné:

V prvom získam všetky id potrebných lístkov

```
SELECT t.id FROM tickets t
INNER JOIN events ON (e.id = t.event_id)
WHERE t.is_used = false AND e.date < (SELECT now());
```

V druhom spravím update na týchto lístkoch

```
UPDATE tickets SET
is_used = true
WHERE id = ANY ( (SELECT ?)::int[] );
```

Výsledkom je značné zrýchlenie tejto funkcie:

Počet riadkov *Tickets*:	Neoptimalizované:	Optimalizované:
140250	11,201s	1,4s
541750	61,523s	2,211s
2228200	viac ako 10 min	12,516s

6. Riešenie problému

Problém: Pôvodne som mal v databázovom modeli vzťah *one to many* medzi ***Tariffs*** a ***Events*** a vzťah *many to many* medzi ***Events*** a ***Sectors***. Umožňovalo to jednoduché priradenie cenníku pre podujatie pre všetky sektory. Trebalo však pridať možnosť, aby niektoré sektory mohli byť drahšie či lacnejšie, čiže priradiť viac cenníkov jednému podujatiu a pre daný sektor.

Riešenie: Implementovanie množiny ***Assigned***, ktorá slúži ako keby implementovanie vzťahu *„many to many to many“* a mohlo mať vlastné ID pre potreby aplikácie.

7. Inšpirácia

Štruktúry kódu aj tejto správy boli inšpirované Vzorovým projektom: Telekomunikačný systém, Alexander Šimko, 2020.