Университет ИТМО

Факультет программной инженерии и компьютерной техники
Кафедра вычислительной техники

Курсовая работа по дисциплине "Системы баз данных"

Разработка базы данных сети магазинов

Выполнили: Айтуганов Д. А.
Чебыкин И. Б.

Группа: P3301

Проверяющий: Беликов П. А.

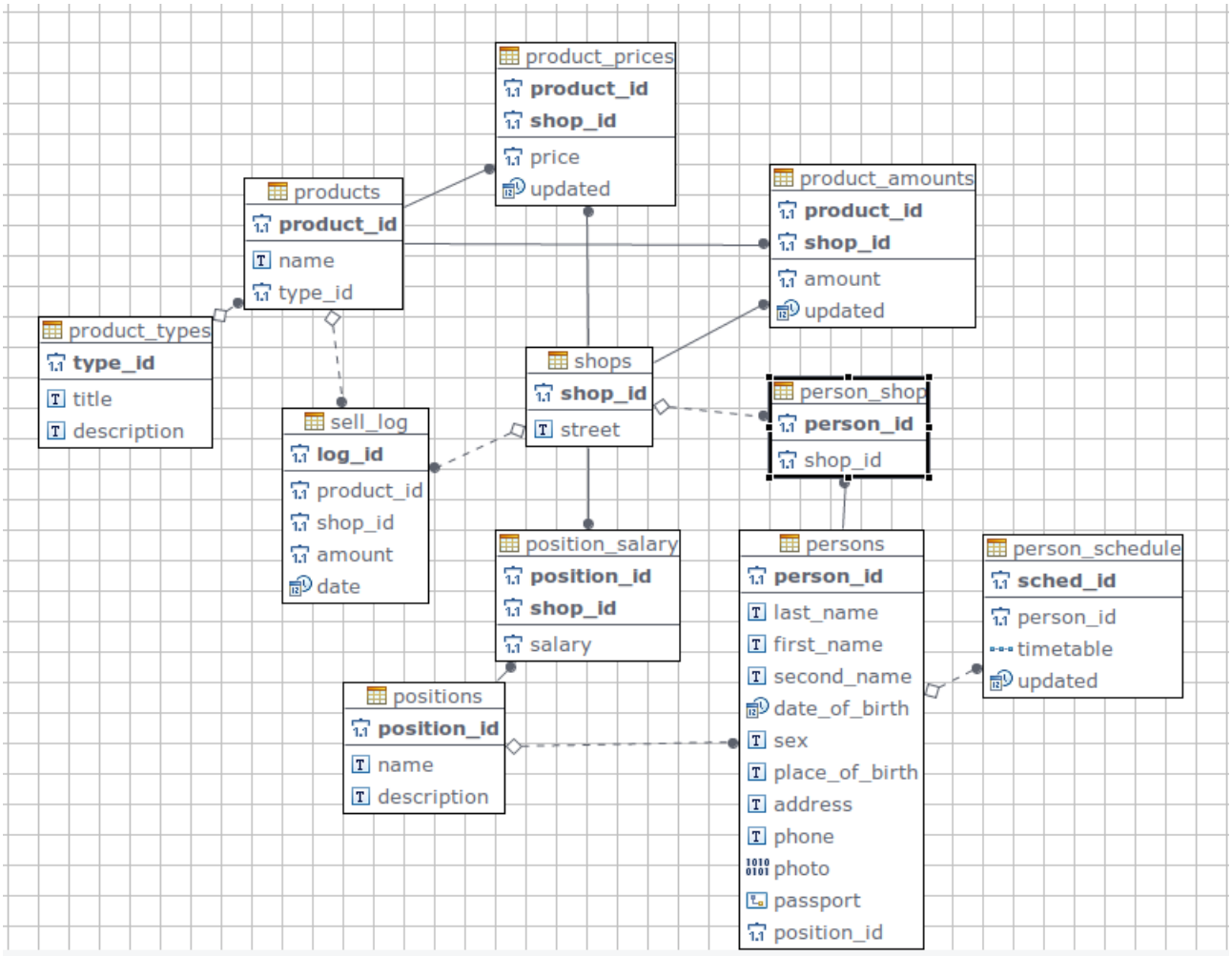
СПб, 2017

Содержание

1 Описание предметной области

В качестве предметной области была выбрана сеть магазинов, которые продают различные товары. В модели учитываются расписание сотрудников, логирование продаж, категории товаров и т. д.

2 Модель первой части



2.1 Примеры CRUD-кода

2.1.1 Хранимые процедуры

```
CREATE FUNCTION add_product(name text, type_id integer) RETURNS integer
    LANGUAGE sql
    AS $$
INSERT INTO products VALUES(NULL, name, type_id) RETURNING product_id;
 $$;
```

2.1.2 API на прикладном языке программирования

```
    private <R extends UpdatableRecord, T extends TableImpl<R>, F extends TableField<R,
    Integer>>
    int doCommand(T table, F[] pkey, CmdType type, int[] id, String fieldName, Object[] args,
    boolean skip_id) {
        R record = null;
```

```
        if (type == CmdType.ADD || type == CmdType.FIELDS) {
            record = ctx.newRecord(table);
        }

        if (type == CmdType.READ || type == CmdType.UPDATE || type == CmdType.DELETE) {
            if (pkey.length == 1) {
                record = ctx.fetchOne(table, pkey[0].equal(id[0]));
            } else {
                record = ctx.fetchOne(table, pkey[0].equal(id[0]).and(pkey[1].equal(id[1])));
            }
            if (record == null && type != CmdType.READ) {
                throw new IllegalArgumentException("No such row");
            }
        }
        if (type == CmdType.DELETE) {
            record.delete();
            return 0;
        }

        if (type == CmdType.READ) {
            if( id[0] > 0 && record == null ){
                throw new IllegalArgumentException("No such row");
            }
            Result<R> records;
            if( record != null ) {
                records = ctx.newResult(table);
                records.add(record);
            }else{
                records = ctx.fetch(table);
            }
            boolean printed = false;
            for (R r : records) {
                if (fieldName != null) {
                    Object data = r.getValue(fieldName);
                    System.out.println(data);
                } else {
                    if( !printed ) {
                        for (Field<?> f : r.fields()) {
                            System.out.print(f.getName()+" ");
                        }
                        printed = true;
                        System.out.println();
                    }
                    for (Field<?> f : r.fields()) {
                        Object data = r.getValue(f);
                        System.out.print(data + " ");
                    }
                    System.out.println();
                }
            }
            return 0;
        }

        Field<?>[] fields = record.fields();
        if (type == CmdType.FIELDS) {
            for (Field f : fields) {
                System.out.println(f);
            }
            return 0;
        }
        int i = skip_id ? -id.length : 0;
        for (Field f : fields) {
            // Skip id
            if (i < 0) {
                i++;
                continue;
            }
            record.set(f, args[i++]);
            System.out.println(f);
        }
        record.store();
        return (int) record.getValue(0);
    }
```

3 Схема второй части

```
persons
var mongoose = require('mongoose');
var table = 'persons';
var schema = mongoose.Schema({
  last_name: {type: String, required: true},
  first_name: {type: String, required: true},
  second_name: String,
  date_of_birth: {type: String, required: true},
  sex: {type: String, validate: /M|F/},
  place_of_birth: {type: String, required: true},
  address: {type: String, required: true},
  phone: {type: String, required: true},
  photo: Buffer,
  passport: {
    type: String,
    unique: true,
    validate: /\d{4},\d{6}/,
    required: true
  },
  position: {
    name: {type: String, required: true},
    description: String,
    shop_id: {
      type: Number,
      required: true
    },
    salary: {type: Number, required: true},
  }
});

module.exports = {
  schema: schema,
  model: mongoose.model(table, schema)
}

products
var mongoose = require('mongoose');
var table = 'products';
var schema = mongoose.Schema({
  name: {type: String, index: {unique: true}},
  type: {
    name: {type: String, required: true},
    description: String
  },
  sell_info: {
    shop_id: Number,
    price: Number,
    amount: Number,
  }
});

module.exports = {
  schema: schema,
  model: mongoose.model(table, schema)
}

sell_logs
var mongoose = require('mongoose');
var table = 'sell_log';
var schema = mongoose.Schema({
  name: {type: String, index: {unique: true} },
  product_id: { type: Number, unique: true },
  shop_id: { type: Number, unique: true },
  amount: Number,
  date: Date,
});

module.exports = {
  schema: schema,
  model: mongoose.model(table, schema)
}

shops
```

```
var mongoose = require('mongoose');
var table = 'shops';
var schema = mongoose.Schema({
  street: {
    type: String,
    required: true
  },
  number: {
    type: Number,
    required: true,
  },
});
schema.index({street: 1, number: 1}, {unique: true});
module.exports = {
  schema: schema,
  model: mongoose.model(table, schema)
}
```

3.1 Примеры CRUD-кода

```
'add': function(splitted_input) {
  var schema = get_schema_by_name(splitted_input[1]);
  if(schema == null) return;
  model = fill_fields(schema, null);
  console.log(model);
  model.save((err) => print_errors(err));
}
```

4 Модели взамодействия с Redis

4.1 Первая часть

При выполнении read операции результат запроса сохраняется для конкретной команды. При выполнении add, update или delete, кэш очищается.

4.2 Вторая часть

При выполнении read операции запрос кэшируется на определенный промежуток времени.