

Курсовая работа по предмету "Базы данных"

Андрей Козлов, гр. 4538

25 января 2013г.

1 Постановка задачи

Некоторая компания предоставляет своим сотрудникам возможность бесплатно общаться по мобильному телефону. Для этого был заключен контракт с оператором мобильной связи на следующих условиях:

- компании выдается набор корпоративных номеров, общение между которыми не тарифицируется;
- цены на все остальные услуги устанавливаются в соответствии с действующими тарифами оператора.

Сотрудник компании может заключить договор на использование корпоративного номера. В договоре указывается тариф оператора, по которому будут тарифицироваться услуги. Один сотрудник может иметь более одного номера в любой момент времени.

Чтобы контролировать расходы компании на мобильную связь, требуется реализовать биллинговую систему, умеющую выполнять следующие операции:

- добавить новый корпоративный номер;
- добавить нового сотрудника;
- заключить контракт с сотрудником на использование корпоративного номера с указанным тарифным планом;
- расторгнуть контракт с сотрудником на использование корпоративный номер;
- сменить тарифный план контракта;
- рассчитать расходы сотрудника за заданный период;
- рассчитать все расходы сотрудника;
- вычислить процент расходов на личные разговоры.

Данная курсовая работа представляет собой реализацию такой биллинговой системы.

2 Модель

Список сущностей:

- сотрудник;
- номер корпоративного телефона;
- тарифный план оператора;
- услуга (звонок, sms/mms-сообщение, gprs-соединение и т.д.);
- единица тарификации услуги (секунды, штуки, кБ и т.д.);
- мобильная операция.

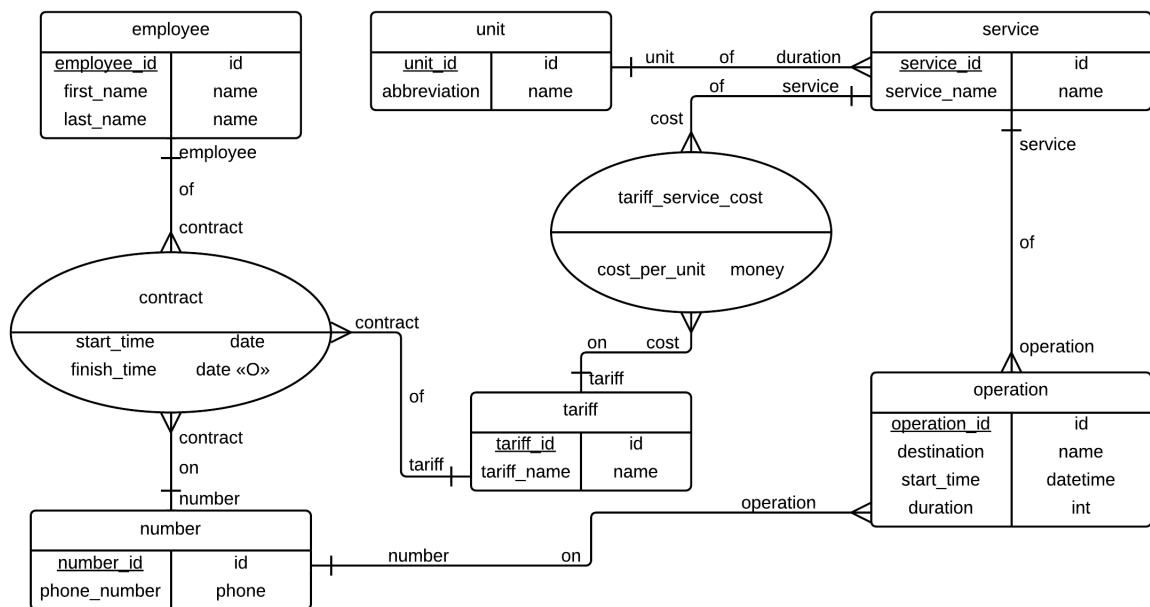


Рис. 1: Модель сущность-связь

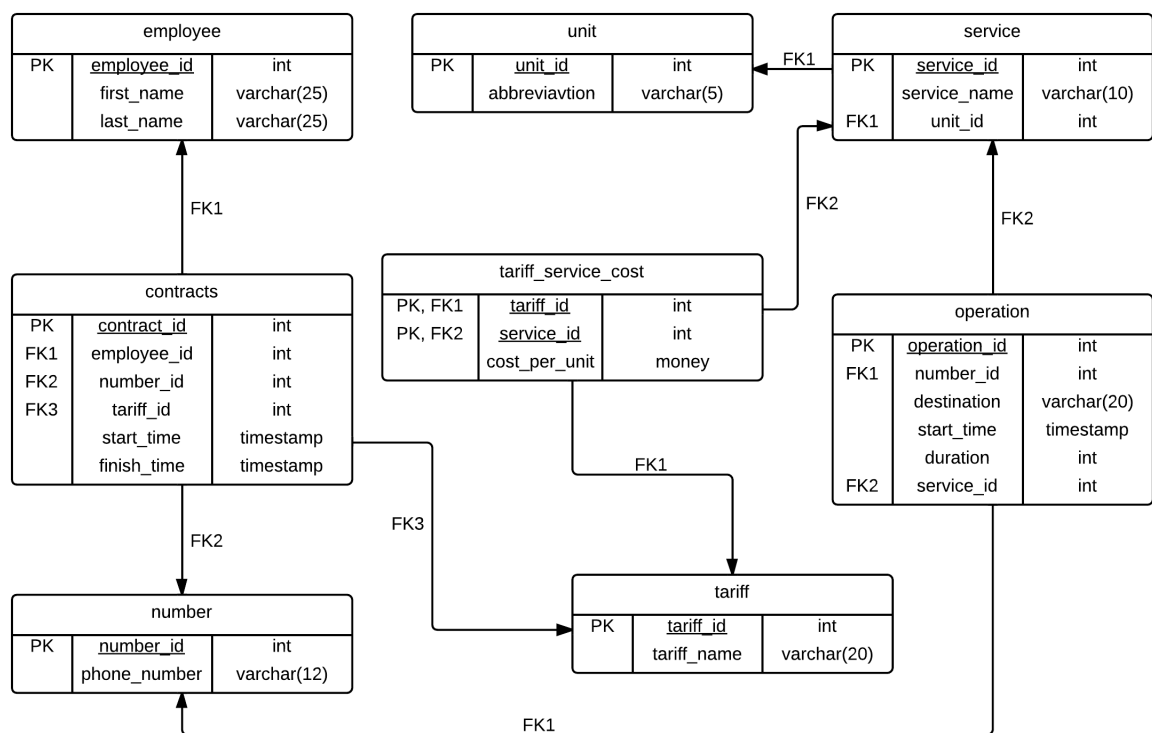


Рис. 2: Физическая модель

3 Исходный код

3.1 Создание таблиц

```
1
2 create table if not exists employees (
3     employee_id serial primary key,
4     first_name varchar(25) not null check (first_name != ''),
5     last_name varchar(25) not null check (last_name != '')
6 );
7 create index lowercased_employees_idx on employees (lower(first_name),
8     lower(last_name));
9
10 create table if not exists numbers (
11     number_id serial primary key,
12     phone_number varchar(12) not null unique check (phone_number != '')
13 );
14 create unique index numbers_idx on numbers (phone_number);
15
16 create table if not exists tariffs (
17     tariff_id serial primary key,
18     tariff_name varchar(20) not null unique check (tariff_name != '')
19 );
20
21 create function inf () returns timestamp as $$
22     begin
23         return '9999-12-31 23:59:59';
24     end;
25 $$ language plpgsql;
26
27 create table if not exists contracts (
28     employee_id int not null references employees (employee_id) on delete
29         restrict on update restrict,
30     number_id int not null references numbers (number_id) on delete
31         restrict on update restrict,
32     tariff_id int not null references tariffs (tariff_id) on delete
33         restrict on update restrict,
34     start_time timestamp not null,
35     finish_time timestamp not null default inf() check (finish_time >=
36         start_time)
37 );
38
39 create table if not exists units (
40     unit_id serial primary key,
41     abbreviation varchar(5) not null unique check (abbreviation != '')
42 );
43
44 create table if not exists services (
45     service_id serial primary key,
46     service_name varchar(10) not null unique check (service_name != '')
47 );
48
49 create table if not exists service_units (
50     service_id int not null unique references services (service_id) on
51         delete restrict on update restrict,
52     unit_id int not null references units (unit_id) on delete restrict on
53         update restrict
54 );
```

```

48
49 create table if not exists tariff_service_costs (
50     tariff_id int not null references tariffs (tariff_id) on delete
51         restrict on update restrict ,
52     service_id int not null references services (service_id) on delete
53         restrict on update restrict ,
54     cost_per_unit money not null ,
55     unique (tariff_id , service_id)
56 );
57
58 create table if not exists operations (
59     operation_id serial primary key ,
60     number_id int not null references numbers (number_id) on delete
61         restrict on update restrict ,
62     destination varchar(20) not null ,
63     operation_time timestamp not null ,
64     duration int not null ,
65     service_id int not null references services (service_id) on delete
66         restrict on update restrict
67 );
68
69 create index operations_idx on operations (number_id, operation_time,
70     service_id);

```

3.2 Создание представлений, функций и триггеров

```

1
2 create or replace function get_employee_id (first_name_ varchar , last_name_
3     varchar) returns int as $$
4     declare
5         employee_id_ int;
6     begin
7         set transaction isolation level serializable read only;
8
9         select employee_id into employee_id_ from employees where lower(
10             first_name) = lower(first_name_) and lower(last_name) = lower(
11                 last_name_);
12
13         if (employee_id_ is null) then
14             raise exception 'there is no employee ' '% %' , first_name_ ,
15                 last_name_;
16         else
17             return employee_id_;
18         end if;
19     end;
20 $$ language plpgsql;
21
22 create or replace function get_number_id (phone_number_ varchar) returns
23     int as $$
24     declare
25         number_id_ int;
26     begin
27         set transaction isolation level serializable read only;
28
29         select number_id into number_id_ from numbers where phone_number =
30             phone_number_;
31
32         if (number_id_ is null) then

```

```

27         raise exception 'there is no phone number ' '%', phone_number_
28         ;
29     else
30         return number_id_;
31     end if;
32 end;
33 $$ language plpgsql;
34 create or replace function get_tariff_id (tariff_name_ varchar) returns int
35 as $$
36 declare
37     tariff_id_ int;
38 begin
39     set transaction isolation level serializable read only;
40     select tariff_id into tariff_id_ from tariffs where lower(
41         tariff_name) = lower(tariff_name_);
42     if (tariff_id_ is null) then
43         raise exception 'there is no tariff ' '%', tariff_name_;
44     else
45         return tariff_id_;
46     end if;
47 end;
48 $$ language plpgsql;
49
50 create or replace function new_employee (first_name_ varchar, last_name_
51     varchar) returns int as $$
52 begin
53     set transaction isolation level serializable read write;
54     insert into employees (first_name, last_name) values (first_name_,
55         last_name_);
56     return currval('employees_employee_id_seq');
57 end;
58 $$ language plpgsql;
59 create or replace function new_number (phone_number_ varchar) returns int
60 as $$
61 begin
62     set transaction isolation level serializable read write;
63     insert into numbers (phone_number) values (phone_number_);
64     return currval('numbers_number_id_seq');
65 end;
66 $$ language plpgsql;
67
68 create or replace function open_contract_on_date (employee_id_ int,
69     number_id_ int, tariff_id_ int, start_time_ timestamp) returns void as
70 $$
71 begin
72     set transaction isolation level serializable read write;
73     insert into contracts (employee_id, number_id, tariff_id,
74         start_time) values (employee_id_, number_id_, tariff_id_,
75         start_time_);
76 end;
77 $$ language plpgsql;

```

```

75
76 create or replace function open_contract_now (employee_id_ int, number_id_
    int, tariff_id_ int) returns void as $$
77     begin
78         set transaction isolation level serializable read write;
79
80         insert into contracts (employee_id, number_id, tariff_id,
            start_time) values (employee_id_, number_id_, tariff_id_, now ()
            );
81     end;
82 $$ language plpgsql;
83
84 create or replace function check_contract_is_open () returns trigger as $$
85     begin
86         if exists (select * from contracts where employee_id = new.
            employee_id and number_id = new.number_id and finish_time = inf
            ()) then
87             raise exception 'contract of employee ''' on number ''' is
                already open', new.employee_id, new.number_id;
88         end if;
89         return new;
90     end;
91 $$ language plpgsql;
92
93 create trigger check_contract_is_open before insert on contracts
94     for each row execute procedure check_contract_is_open ();
95
96 create or replace function close_contract_on_date (employee_id_ int,
    number_id_ int, finish_time_ timestamp) returns void as $$
97     begin
98         set transaction isolation level serializable read write;
99
100        update contracts set finish_time = finish_time_ where employee_id =
            employee_id_ and number_id = number_id_ and finish_time = inf()
            ;
101    end;
102 $$ language plpgsql;
103
104 create or replace function close_contract_now (employee_id_ int, number_id_
    int) returns void as $$
105     begin
106         set transaction isolation level serializable read write;
107
108         perform close_contract_on_date (employee_id_, number_id_, now ());
109     end;
110 $$ language plpgsql;
111
112 create or replace function check_contract_is_closed () returns trigger as
    $$
113     begin
114         if not exists (select * from contracts where employee_id = new.
            employee_id and number_id = new.number_id and finish_time = inf
            ()) then
115             raise exception 'there is no opened contract of employee '''
                on number ''', new.employee_id, new.number_id;
116         end if;
117         return new;
118     end;

```

```

119 $$ language plpgsql;
120
121 create trigger check_contract_is_closed before update on contracts
122     for each row execute procedure check_contract_is_closed ();
123
124 create or replace function change_tariff (number_id_ int, new_tariff_id int
125 ) returns int as $$
126     declare
127         contract_id_ int;
128         employee_id_ int;
129         old_tariff_id int;
130     begin
131         set transaction isolation level serializable read write;
132
133         select contract_id into contract_id_ from contracts where number_id
134             = number_id_ and finish_time = inf();
135
136         if (contract_id_ is null) then
137             raise exception 'there is no opened contract on number ''%',
138                 number_id_;
139         else
140             select tariff_id into old_tariff_id from contracts where
141                 contract_id = contract_id_;
142
143             if (old_tariff_id != new_tariff_id) then
144                 update contracts set finish_time = current_timestamp where
145                     contract_id = contract_id_;
146                 select employee_id into employee_id_ from contracts where
147                     contract_id = contract_id_;
148                 return open_contract_now (employee_id_, number_id_,
149                     new_tariff_id);
150             else
151                 raise notice 'tariff on number ''%' is already ''%',
152                     number_id_, old_tariff_id;
153                 return contract_id_;
154             end if;
155         end if;
156     end;
157 $$ language plpgsql;
158
159 create or replace view all_contracts as
160     select * from contracts natural join employees natural join numbers
161         natural join tariffs;
162
163 create or replace function all_employee_contracts (first_name_ varchar,
164     last_name_ varchar) returns table(phone_number varchar, tariff_name
165     varchar, start_time timestamp, finish_time timestamp) as $$
166     begin
167         set transaction isolation level serializable read only;
168
169         return query select c.phone_number, c.tariff_name, c.start_time, c.
170             finish_time from all_contracts as c where employee_id =
171             get_employee_id (first_name_, last_name_);
172     end;
173 $$ language plpgsql;
174
175 create or replace function open_employee_contracts (first_name_ varchar,
176     last_name_ varchar) returns table(phone_number varchar, tariff_name

```

```

163      varchar, start_time timestamp) as $$
164      begin
165          set transaction isolation level serializable read only;
166          return query select c.phone_number, c.tariff_name, c.start_time
                        from all_employee_contracts (first_name_, last_name_) as c where
                        finish_time = inf();
167      end;
168  $$ language plpgsql;
169
170  create or replace view all_operations as
171      select * from operations natural join contracts natural join employees
            natural join tariff_service_costs;
172
173  create or replace function total_employee_charges (first_name_ varchar,
174      last_name_ varchar) returns money as $$
175      declare
176          charges money;
177      begin
178          set transaction isolation level serializable read only;
179          select sum (duration * cost_per_unit) into charges from
            all_operations where employee_id = get_employee_id (first_name_ ,
            last_name_) group by employee_id;
180      return coalesce (charges , 0.00::money);
181      end;
182  $$ language plpgsql;
183
184  create or replace function period_employee_charges (first_name_ varchar,
185      last_name_ varchar, start_time_ timestamp, finish_time_ timestamp)
186      returns money as $$
187      declare
188          charges money;
189      begin
190          set transaction isolation level serializable read only;
191          select sum (duration * cost_per_unit) into charges from
            all_operations where employee_id = get_employee_id (first_name_ ,
            last_name_) and operation_time >= start_time_ and
            operation_time < finish_time_ group by employee_id;
192      return coalesce (charges , 0.00::money);
193      end;
194  $$ language plpgsql;
195
196  create or replace function employee_charges_percent (first_name_ varchar,
197      last_name_ varchar) returns real as $$
198      declare
199          charges money;
200          total_charges money;
201      begin
202          set transaction isolation level serializable read only;
203          select sum (duration * cost_per_unit) into charges from
            all_operations where employee_id = get_employee_id (first_name_ ,
            last_name_) and destination not in (select phone_number from
            numbers) group by first_name, last_name;

```



```

204         select total_employee_charges (first_name_, last_name_) into
           total_charges;
205
206         if (total_charges = 0.00::money) then
207             raise notice 'employee ' || '%' || ' has no operations yet',
                first_name_, last_name_;
208             return 0.0;
209         else
210             return charges / total_charges;
211         end if;
212     end;
213 $$ language plpgsql;
214
215 create or replace function spenders () returns table (first_name varchar,
           last_name varchar, percent real) as $$
216     begin
217         set transaction isolation level serializable read only;
218
219         return query select e.first_name, e.last_name,
           employee_charges_percent (e.first_name, e.last_name) as p from
           employees as e order by p desc;
220     end;
221 $$ language plpgsql;
222
223 create or replace view all_operations_with_services as
224     select * from all_operations natural join services natural join
           service_units natural join units natural join numbers;
225
226 create or replace function period_employee_operations (first_name_ varchar,
           last_name_ varchar, start_time_ timestamp, finish_time_ timestamp)
           returns table (phone_number varchar, destination varchar, operation_time
           timestamp, cost money, duration int, unit varchar, service_name varchar
           ) as $$
227     begin
228         set transaction isolation level serializable read only;
229
230         return query select o.phone_number, o.destination, o.operation_time
           , o.duration * o.cost_per_unit, o.duration, o.abbreviation, o.
           service_name from all_operations_with_services as o where
           employee_id = get_employee_id (first_name_, last_name_) and o.
           operation_time >= start_time_ and o.operation_time <
           finish_time_;
231     end;
232 $$ language plpgsql;
233
234 create or replace function period_employee_work_operations (first_name_
           varchar, last_name_ varchar, start_time_ timestamp, finish_time_
           timestamp) returns table (phone_number varchar, destination varchar,
           operation_time timestamp, cost money, duration int, unit varchar,
           service_name varchar) as $$
235     begin
236         set transaction isolation level serializable read only;
237
238         return query select * from period_employee_operations (first_name_,
           last_name_, start_time_, finish_time_) as o where o.destination
           in (select n.phone_number from numbers as n);
239     end;
240 $$ language plpgsql;

```

```

241
242 create or replace function period_employee_not_work_operations (first_name_
      varchar, last_name_ varchar, start_time_ timestamp, finish_time_
      timestamp) returns table (phone_number varchar, destination varchar,
      operation_time timestamp, cost money, duration int, unit varchar,
      service_name varchar) as $$
243   begin
244     set transaction isolation level serializable read only;
245
246     return query select * from period_employee_operations (first_name_ ,
      last_name_ , start_time_ , finish_time_ ) except all (select *
      from period_employee_work_operations (first_name_ , last_name_ ,
      start_time_ , finish_time_ ));
247   end;
248 $$ language plpgsql;

```