Санкт-Петербургский государственный политехнический университет

Институт информационных технологий и управления

Кафедра компьютерных систем и программных технологий

ОТЧЕТ

По лабораторной работе

по дисциплине "Базы данных"

Выполнили: студент гр. 43501/13 Баринов Д.С.

студент гр. 43501/13 Сухинин А.А.

Проверил: М.Ю. Моисеев

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/подпись преподавателя, дата/

Санкт-Петербург

2014

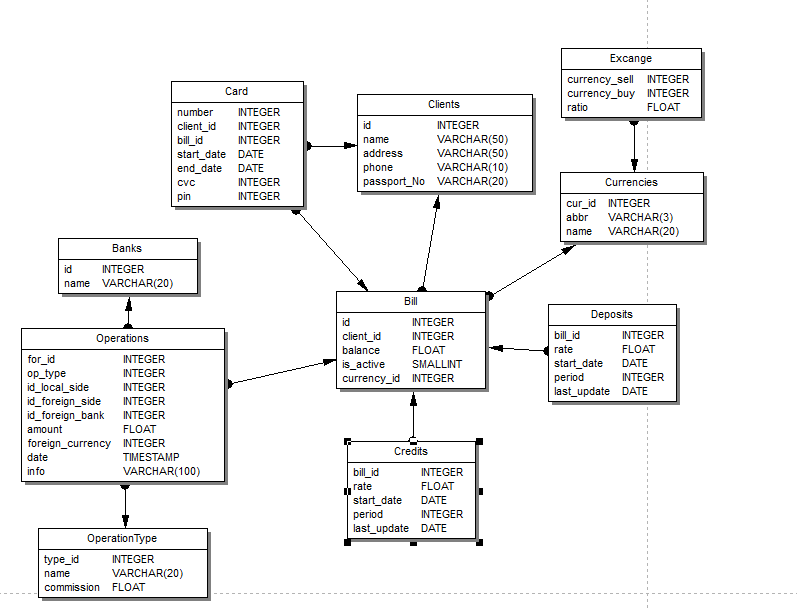
# Задание

Банк (счета, кредиты, переводы)

Банк предлагает различные услуги: ведение счета, предоставление кредита, переводы денег в другие банки, конвертацию из одной валюты в другую. ИС хранит информацию о клиентах, их счетах, операциях на счетах, тарифах, позволяет начислять проценты и осуществлять списания, поддерживает переводы с одного счёта на другой, включая другие банки с соответствующими списаниями за услуги. ИС также обеспечивает работу с кредитами клиентов. Интерфейс банкомата и интерфейс монитора.

# Решение.

Для решения поставленной задачи была спроектирована база данных, имеющая следующую архитектуру:



Скрипт для создания базы данных:

CREATE TABLE "Currencies"(

"cur\_id" INTEGER NOT NULL,

"abbr" VARCHAR(3),

"name" VARCHAR(20));

ALTER TABLE "Currencies" ADD CONSTRAINT "PK\_Currencies" PRIMARY KEY ("cur\_id");

CREATE TABLE "Excange"(

"currency\_sell" INTEGER,

"currency\_buy" INTEGER,

"ratio" FLOAT);

ALTER TABLE "Excange" ADD CONSTRAINT "FK\_ex\_sell\_cur" FOREIGN KEY ("currency\_sell") REFERENCES

"Currencies" ("cur\_id");

ALTER TABLE "Excange" ADD CONSTRAINT "FK\_ex\_buy\_cur" FOREIGN KEY ("currency\_buy") REFERENCES

"Currencies" ("cur\_id");

ALTER TABLE "Excange" ADD CONSTRAINT "UQ\_ratio" UNIQUE ("currency\_sell", "currency\_buy");

CREATE TABLE "Banks" (

"id" INTEGER NOT NULL,

"name" VARCHAR(20));

CREATE TABLE "Card" (

"number" INTEGER NOT NULL,

"client\_id" INTEGER,

"bill\_id" INTEGER,

"start\_date" DATE,

"end\_date" DATE,

"cvc" INTEGER,

"pin" INTEGER);

CREATE TABLE "Clients" (

"id" INTEGER NOT NULL,

"name" VARCHAR(50) NOT NULL,

"address" VARCHAR(50),

"phone" VARCHAR(10),

"passport\_No" VARCHAR(20) NOT NULL) ;

ALTER TABLE "Clients" ADD CONSTRAINT "PK\_id" PRIMARY KEY ("id");

ALTER TABLE "Clients" ADD CONSTRAINT "UQ\_pass\_no" UNIQUE ("passport\_No");

CREATE TABLE "Bill"(

"id" INTEGER NOT NULL,

"client\_id" INTEGER NOT NULL,

"balance" FLOAT,

"is\_active" SMALLINT,

"currency\_id" INTEGER NOT NULL);

ALTER TABLE "Bill" ADD CONSTRAINT "PK\_bill\_id" PRIMARY KEY ("id");

ALTER TABLE "Bill" ADD CONSTRAINT "FK\_Acc\_Cur\_id" FOREIGN KEY ("currency\_id") REFERENCES

"Currencies" ("cur\_id");

ALTER TABLE "Bill" ADD CONSTRAINT "FK\_bill\_cl\_id" FOREIGN KEY ("client\_id") REFERENCES

"Clients" ("id");

CREATE TABLE "OperationType" (

"type\_id" INTEGER NOT NULL,

"name" VARCHAR(20),

"commission" FLOAT);

ALTER TABLE "Banks" ADD CONSTRAINT "PK\_Banks" PRIMARY KEY ("id");

ALTER TABLE "Card" ADD CONSTRAINT "PK\_Card" PRIMARY KEY ("number");

ALTER TABLE "OperationType" ADD CONSTRAINT "PK\_OperationType" PRIMARY KEY ("type\_id");

ALTER TABLE "Card" ADD CONSTRAINT "FK\_client\_id" FOREIGN KEY ("client\_id") REFERENCES "Clients" ("id");

ALTER TABLE "Card" ADD CONSTRAINT "FK\_Card\_bill\_id" FOREIGN KEY ("bill\_id") REFERENCES "Bill" ("id");

CREATE TABLE "Deposits"(

"bill\_id" INTEGER UNIQUE,

"rate" FLOAT,

"start\_date" DATE,

"period" INTEGER,

"last\_update" DATE);

ALTER TABLE "Deposits" ADD CONSTRAINT "FK\_de\_bill\_id" FOREIGN KEY ("bill\_id") REFERENCES

"Bill" ("id");

CREATE TABLE "Credits"(

"bill\_id" INTEGER UNIQUE,

"rate" FLOAT,

"start\_date" DATE,

"period" INTEGER,

"last\_update" DATE);

ALTER TABLE "Credits" ADD CONSTRAINT "FK\_cr\_bill\_id" FOREIGN KEY ("bill\_id") REFERENCES

"Bill" ("id");

CREATE TABLE "Operations"(

"for\_id" INTEGER NOT NULL,

"op\_type" INTEGER,

"id\_local\_side" INTEGER,

"id\_foreign\_side" INTEGER,

"id\_foreign\_bank" INTEGER,

"amount" FLOAT,

"foreign\_currency" INTEGER,

"date" TIMESTAMP,

"info" VARCHAR(100));

ALTER TABLE "Operations" ADD CONSTRAINT "PK\_Foreign\_Operations" PRIMARY KEY ("for\_id");

ALTER TABLE "Operations" ADD CONSTRAINT "FK\_foreign\_op\_bank\_id" FOREIGN KEY ("id\_foreign\_bank") REFERENCES

"Banks" ("id");

ALTER TABLE "Operations" ADD CONSTRAINT "FK\_fo\_op\_lo\_si" FOREIGN KEY ("id\_local\_side") REFERENCES

"Bill" ("id");

ALTER TABLE "Operations" ADD CONSTRAINT "FK\_fo\_op\_type" FOREIGN KEY ("op\_type") REFERENCES

"OperationType" ("type\_id");

COMMIT;

Для обеспечения защиты данных при работе с бд все операции с бд было решено проводить только с помощью хранимых процедур.

Реализованы следующие хранимые процедуры:

* CHECKCARDCVC – проверка авторизации пользователя по номеру карты и коду cvc;
* CHECKPASSWD - проверка авторизации пользователя по номеру карты и пин коду карты;
* GETRURCHANGE - позволяет получить коэффициент перевода из текущей валюты в рубли;
* GETBALANCERURCVC – получение баланса счета в рублях с проверкой авторизации по коду cvc;
* GETBALANCERUR - получение баланса счета в рублях с проверкой авторизации по пин коду;
* WITHDRAWRUR – снятие денег со счета с проверкой авторизации по пин коду (имитация получения денег в банкомате);
* GETOPERATIONS – получение списка всех операций с картой;
* LOCALMONEYTRANSFER – перевод денег со счета на счет с вычислением комиссии за операцию.
* ADD\_PERCENTS\_CREDIT – пересчет нового значения баланса счета для кредитных счетов.
* ADD\_PERCENT\_DEPOSIT - пересчет нового значения баланса счета для вкладов.

Исходный код процедур приведен в приложении 1.

# Реализация веб интерфейса.

Веб интерфейс реализован в виде веб сайта на ASP.NET. Данная технология позволяет создать удобную базу для описания поведения сайта.

Возможности веб интерфейса:

* Просмотр информации о счете
* Просмотр истории переводов
* Перевод средств на счет

Идентификация и авторизация пользователя происходит по номеру карты и cvc коду карты.

Для обеспечения взаимодействия с базой данных на основе Firebird используется библиотека FirebirdSql.Data.FirebirdClient.

Для работы сайта необходим веб сервер IIS.

# Реализация терминала.

//здесь могла быть ваша реклама

Приложение 1. Процедуры.

SET TERM ^ ;

create or alter procedure CHECKPASSWD (

C\_NUMBER integer,

C\_PIN integer)

returns (

EXISTING integer)

as

begin

existing = 0;

for select "number" FROM "Card"

WHERE "Card"."number"=:c\_number and "Card"."pin"=:c\_pin

into :existing

do begin

end

suspend;

end^

SET TERM ; ^

SET TERM ^ ;

create or alter procedure CHECKCARDCVC (

C\_NUMBER integer,

C\_CVC integer)

returns (

EXISTING integer)

as

begin

existing = 0;

for select "number" FROM "Card"

WHERE "Card"."number"=:c\_number and "Card"."cvc"=:c\_cvc

into :existing

do begin

end

suspend;

end^

SET TERM ; ^

SET TERM ^ ;

create or alter procedure GETBALANCERUR (

C\_NUMBER integer,

C\_PIN integer)

returns (

amount float)

as

DECLARE VARIABLE login INTEGER = 0;

DECLARE VARIABLE currency INTEGER = 0;

DECLARE VARIABLE changeRur FLOAT = 0;

begin

login = 0;

amount = -1;

EXECUTE PROCEDURE checkPasswd(C\_NUMBER, C\_PIN) RETURNING\_VALUES :login;

IF (:login =0) THEN BEGIN

SUSPEND;

EXIT;

END

FOR SELECT "Bill"."balance", "Bill"."currency\_id" FROM "Bill", "Card"

WHERE "Card"."number"=:c\_number AND "Bill"."id" = "Card"."bill\_id"

INTO :amount, :currency

DO BEGIN END

EXECUTE PROCEDURE getRurChange(:currency) RETURNING\_VALUES :changeRur;

amount = :amount\* :changeRur;

SUSPEND;

end^

SET TERM ; ^

SET TERM ^ ;

create or alter procedure GETBALANCERURCVC (

C\_NUMBER integer,

C\_CVC integer)

returns (

amount float)

as

DECLARE VARIABLE login INTEGER = 0;

DECLARE VARIABLE currency INTEGER = 0;

DECLARE VARIABLE changeRur FLOAT = 0;

begin

login = 0;

amount = -1;

EXECUTE PROCEDURE checkCardCvc(C\_NUMBER, C\_CVC) RETURNING\_VALUES :login;

IF (:login =0) THEN BEGIN

SUSPEND;

EXIT;

END

FOR SELECT "Bill"."balance", "Bill"."currency\_id" FROM "Bill", "Card"

WHERE "Card"."number"=:c\_number AND "Bill"."id" = "Card"."bill\_id"

INTO :amount, :currency

DO BEGIN END

EXECUTE PROCEDURE getRurChange(:currency) RETURNING\_VALUES :changeRur;

amount = :amount\* :changeRur;

SUSPEND;

end^

SET TERM ; ^

SET TERM ^ ;

create or alter procedure GETRURCHANGE (

src\_currency INTEGER)

returns (

ratio float)

as

DECLARE VARIABLE rur integer = -1;

begin

ratio = -1;

for select "Currencies"."cur\_id" from "Currencies"

where "Currencies"."abbr" = 'RUR'

INTO :rur

DO BEGIN END

IF (rur<0) THEN

BEGIN

SUSPEND;

EXIT;

END

IF (:src\_currency=:rur) then

begin

ratio = 1;

suspend;

exit;

end

for select "Excange"."ratio" from "Excange"

where "Excange"."currency\_sell"=:src\_currency AND "Excange"."currency\_buy"=:rur

into :ratio

do begin end

suspend;

end^

SET TERM ; ^

SET TERM ^ ;

create or alter procedure WITHDRAWRUR (

C\_NUMBER integer,

C\_PIN integer,

AMOUNT float)

returns (

SUCCESS float)

as

declare variable ID integer;

declare variable SOURCE\_BILL integer;

declare variable ISCREDIT integer;

declare variable CURRENCY integer;

declare variable COMMISION float;

declare variable CHANGERATIO float;

declare variable EQUITY float = 0;

begin

success = 0;

EXECUTE PROCEDURE getbalancerur(C\_NUMBER, C\_PIN) RETURNING\_VALUES :equity;

iscredit = -1;

for select "Credits"."bill\_id" from "Credits", "Card"

where "Credits"."bill\_id"="Card"."bill\_id" and "Card"."number"=:c\_number

into :iscredit do begin end

IF (amount<0 or (equity<amount and :iscredit<0)) THEN

BEGIN

SUSPEND;

EXIT;

END

FOR SELECT "Bill"."currency\_id" FROM "Bill", "Card"

WHERE "Card"."number"=:c\_number AND "Bill"."id" = "Card"."bill\_id"

INTO :currency

DO BEGIN END

EXECUTE PROCEDURE getrurchange(:currency) RETURNING\_VALUES :changeratio;

UPDATE "Bill" SET "Bill"."balance" = (:equity-:amount)/:changeratio

WHERE "Bill"."id" IN (SELECT "Card"."bill\_id" FROM "Card" WHERE "Card"."number"=:C\_NUMBER);

FOR select count (\*) from "Operations" into :id DO BEGIN END

FOR select "Card"."bill\_id" from "Card" where "Card"."number" = :c\_number into :source\_bill DO BEGIN END

INSERT INTO "Operations" ("for\_id","op\_type","id\_local\_side","id\_foreign\_side","id\_foreign\_bank","amount","foreign\_currency","date","info")

VALUES (:id+1,0, :source\_bill, null, null,:amount,2, current\_timestamp,'From terminal');

FOR select "OperationType"."commission" from "OperationType" where "OperationType"."type\_id" = 0 into :commision DO BEGIN END

success=:amount/:changeratio\*:commision;

suspend;

end^

SET TERM ; ^

SET TERM ^ ;

create or alter procedure GETOPERATIONS (

BILL\_ID integer)

returns (

AMOUT integer,

OPDATE timestamp,

INFO varchar(100),

"TYPE" varchar(20),

ID\_FOREIGN\_BANK integer,

ID\_FOREIGN\_SIDE integer)

as

declare variable OPTYPE integer;

begin

for SELECT "Operations"."op\_type",

"Operations"."amount",

"Operations"."date",

"Operations"."id\_foreign\_side",

"Operations"."id\_foreign\_bank",

"Operations"."info"

from "Operations" where "Operations"."id\_local\_side"=:bill\_id

INTO

:optype, :amout, :opdate, :id\_foreign\_side, :id\_foreign\_bank, :info

DO begin

for select "OperationType"."name" from "OperationType" where "OperationType"."type\_id"=:optype into :type do begin end

suspend;

end

end^

SET TERM ; ^

SET TERM ^ ;

create or alter procedure LOCALMONEYTRANSFER (

INFO varchar(100),

AMOUNT float,

DST\_BILL integer,

SRC\_BILL integer)

returns (

COMMISION float)

as

declare variable DST\_EQUTY integer;

declare variable SRC\_EQUITY float;

declare variable ISCREDIT integer;

declare variable ID integer;

declare variable CHANGERAIO float;

declare variable SRC\_CURRENCY integer;

declare variable DST\_AMOUT integer;

declare variable DST\_CURRENCY integer;

begin

commision = -1;

for select "Bill"."currency\_id" from "Bill" where "Bill"."id"=:src\_bill into :src\_currency do begin end

for select "Bill"."currency\_id" from "Bill" where "Bill"."id"=:dst\_bill into :dst\_currency do begin end

for select "Excange"."ratio" from "Excange" where "Excange"."currency\_sell"=:src\_currency and "Excange"."currency\_buy"=:dst\_currency

into :changeraio do begin end

for select "Bill"."balance" from "Bill" where "Bill"."id"=:src\_bill into :src\_equity do begin end

for select "Bill"."balance" from "Bill" where "Bill"."id"=:dst\_bill into :dst\_equty do begin end

iscredit = -1;

for select "Credits"."bill\_id" from "Credits"

where "Credits"."bill\_id"=:src\_bill

into :iscredit do begin end

IF (:amount<0 or (:src\_equity<:amount and :iscredit<0)) THEN

BEGIN

SUSPEND;

EXIT;

END

FOR select count (\*) from "Operations" into :id DO BEGIN END

FOR select "OperationType"."commission" from "OperationType" where "OperationType"."type\_id"=2 into :commision do begin end

dst\_amout=:amount\*:changeraio;

commision=:dst\_amout\*:commision;

UPDATE "Bill" SET "Bill"."balance" = (:src\_equity-:amount) WHERE "Bill"."id"=:src\_bill;

UPDATE "Bill" SET "Bill"."balance" = (:dst\_equty+:dst\_amout-:commision) WHERE "Bill"."id"=:dst\_bill;

INSERT INTO "Operations" ("for\_id","op\_type","id\_local\_side","id\_foreign\_side","id\_foreign\_bank","amount","foreign\_currency","date","info")

VALUES (:id+1,2, :src\_bill, :dst\_bill, null,:amount,null, current\_timestamp,:info);

INSERT INTO "Operations" ("for\_id","op\_type","id\_local\_side","id\_foreign\_side","id\_foreign\_bank","amount","foreign\_currency","date","info")

VALUES (:id+2,3, :dst\_bill, :src\_bill, null,:dst\_amout, null, current\_timestamp,:info);

commision=:commision/:changeraio;

suspend;

end^

SET TERM ; ^

SET TERM ^ ;

create or alter procedure ADD\_PERCENTS\_CREDIT (

BILL\_ID integer)

returns (

AMOUNT float)

as

declare variable BALANCE float;

declare variable DAYS integer;

declare variable LAST\_UPDATE date;

declare variable RATE float;

begin

amount = 0;

for select "Credits"."rate", "Credits"."last\_update", "Bill"."balance" from "Credits", "Bill"

where "Credits"."bill\_id"=:bill\_id and "Bill"."id"=:bill\_id

into :rate,:last\_update, :balance do

begin

if (balance>0) then

begin

suspend;

exit;

end

days = (current\_date-:last\_update);

rate = :rate/365;

amount = :days\*:rate\*:balance;

UPDATE "Bill" SET "Bill"."balance" = :balance+:amount

WHERE "Bill"."id"=:bill\_id;

update "Credits" SET "Credits"."last\_update" = current\_date

where "Credits"."bill\_id"=:bill\_id;

suspend;

end

end^

SET TERM ; ^

SET TERM ^ ;

create or alter procedure ADD\_PERCENT\_DEPOSIT (

BILL\_ID integer)

returns (

AMOUNT float)

as

declare variable DAYS integer;

declare variable BALANCE float;

declare variable LAST\_UPDATE date;

declare variable RATE float;

begin

amount = 0;

for select "Deposits"."rate", "Deposits"."last\_update", "Bill"."balance" from "Deposits", "Bill"

where "Deposits"."bill\_id"=:bill\_id and "Bill"."id"=:bill\_id

into :rate,:last\_update, :balance do

begin

if (balance<0) then

begin

suspend;

exit;

end

days = (current\_date-:last\_update);

rate = :rate/365;

amount = :days\*:rate\*:balance;

UPDATE "Bill" SET "Bill"."balance" = :balance+:amount

WHERE "Bill"."id"=:bill\_id;

update "Deposits" SET "Deposits"."last\_update" = current\_date

where "Deposits"."bill\_id"=:bill\_id;

suspend;

end

end^

SET TERM ; ^

SET TERM ^ ;

create or alter procedure ADD\_ALL\_PERCENTS

returns (

AMOUNT float,

BILL integer)

as

declare variable BILL\_ID integer;

begin

for select "Credits"."bill\_id" from "Credits" into :bill\_id do

begin

EXECUTE PROCEDURE add\_percents\_credit(:bill\_id) RETURNING\_VALUES :amount;

bill = :bill\_id;

suspend;

end

for select "Deposits"."bill\_id" from "Deposits" into :bill\_id do

begin

EXECUTE PROCEDURE add\_percent\_deposit(:bill\_id) RETURNING\_VALUES :amount;

bill = :bill\_id;

suspend;

end

end^

SET TERM ; ^