Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»



**Домашнее задание №1 по дисциплине**

«Постреляционные базы данных»

**ИСПОЛНИТЕЛЬ:**

Березин И. С.

Группа ИУ5-23М

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

"\_\_"\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

Москва, 2019

**Цель работы:**

Освоить технологии построения моделей базы данных: ER, объектной, объектно-реляционной и полуструктуруированных данных. Сравнить возможности этих моделей с возможностями реляционной модели.

1. **На основе описания предметной области выделить сущности, их атрибуты и связи. Указать ключевые атрибуты и типы связей. Построить ER-модель. Продемонстрировать в модели связи один-ко-многим (1-М) и многие-ко-многим (М-М), связь типа ISA**



1. **На основе описания предметной области построить объектную модель БД на языке ODL. Привести описания классов, их свойств, методов и связей. Указать ключи и экстенты.**

Методы классов следует выделять так, чтобы их можно было использовать в типовых запросах вместо вложенных подзапросов. Продемонстрировать наследование, интерфейсы, сложные типы данных (коллекции и структуры)

class client (extent clients

key id\_c)

{

attribute integer id\_c;

attribute integer passport;

attribute string fio;

attribute integer age;

attribute string city;

attribute Struct Addr

{

string street,

integer house,

integer apart

} addr;

attribute Struct Cont

{

string phone,

string email,

} cont;

relationship Set< order> ords

inverse order::cli;

// метод возвращает информацию о клиенте, если такого заказа не существует возвращает noclient

string Info() raise(noclient);

};

class order (extent orders

key id\_o)

{

attribute integer id\_o;

attribute string date;

attribute integer summary;

relationship Set< client > cli

inverse client::ords;

relationship branch br

inverse branch::ords;

// метод возвращает информацию о заказе, если такого заказа не существует возвращает noorder

string Info() raise(noorder);

};

class branch (extent branches

key number)

{

attribute integer number;

attribute Struct Addr

{

string street,

integer house,

integer apart

} addr;

relationship Set <order> ords

inverse orders::br;

};

class payment extends order (extent ords)

{

attribute string payment\_method;

};

interface iclient

{

attribute string fio;

attribute integer age;

attribute string city;

attribute Struct Cont

{

string phone,

string email,

} cont;

relationship Set<order> ords

inverse order::cli;

};

1. **На основе объектной модели построить реляционную модель. Привести перечень отношений с указанием их атрибутов и ключей. Составить на языке SQL описание схемы базы данных (создание таблиц, задание первичных и внешних ключей, ограничений и т.д.). Продемонстрировать реализацию отношения наследования тремя различными методами**

client (id\_c, passport, fio, age, address, contacts, city)

cliord (id\_c, id\_o)

branch (number, address)

1. **Набор сущностей со всеми их атрибутами**

order (id\_o, date, summary, branch(FK))

payment (id\_o, date, summary, branch(FK), payment\_method)

1. **Базовая сущность + набор сущностей с дополнительными атрибутами и ключом**

order (id\_o, date, summary, branch(FK))

payment (id\_o, payment\_method)

1. **Одна таблица со всеми полями**

order (id\_o, date, summary, place(FK), payment\_method)

create\_table Client

(

id\_c integer NOT NULL,

fio varchar(100) NOT NULL,

passport integer NOT NULL,

age integer,

city varchar(100),

address varchar(128),

contacts varchar(128),

CONSTRAINT idc PRIMARY KEY(id\_c),

CONSTRAINT age CHECK(age >= 18));

create\_table Cliord

(

id\_c NOT NULL,

id\_o NOL NULL,

CONSTRAINT idc PRIMARY KEY(id\_c, id\_o),

CONSTRAINT idcf FOREIGN KEY(id\_c) REFERENCES Client(id\_c)

ON UPDATE CASCADE ON DELETE SET NULL,

CONSTRAINT idof FOREIGN KEY(id\_o) REFERENCES Order(id\_o)

ON UPDATE CASCADE ON DELETE SET NULL

);

create\_table Branch

(

Number integer NOT NULL,

Address varchar(128)

CONSTRAINT number PRIMARY KEY(number)

);

**Первый вариант**

create\_table Order

(

Id\_o integer NOT NULL,

date varchar(15) NOT NULL,

summary integer,

branch varchar(15),

CONSTRAINT idop PRIMARY KEY(id\_o),

CONSTRAINT branchf FOREIGN KEY (branch) REFERENCES Branch(number)

ON UPDATE CASCADE ON DELETE SET NULL

);

create\_table Payment

(

Id\_o integer NOT NULL,

date varchar(15) NOT NULL,

summary integer,

branch varchar(15),

payment\_method varchar(30) NOT NULL,

CONSTRAINT idop PRIMARY KEY(id\_o),

CONSTRAINT branchf FOREIGN KEY (branch) REFERENCES Branch(number)

ON UPDATE CASCADE ON DELETE SET NULL

);

**Второй вариант**

create\_table Order

(

Id\_o integer NOT NULL,

date varchar(15) NOT NULL,

summary integer,

branch varchar(15),

CONSTRAINT idop PRIMARY KEY(id\_o),

CONSTRAINT branchf FOREIGN KEY (branch) REFERENCES Branch(number)

ON UPDATE CASCADE ON DELETE SET NULL);

create\_table Payment

(

Id\_o integer NOT NULL,

payment\_method varchar(30) NOT NULL,

CONSTRAINT idop PRIMARY KEY(id\_o),

);

**Третий вариант**

create\_table Order

(

Id\_o integer NOT NULL,

date varchar(15) NOT NULL,

summary integer,

branch varchar(15),

payment\_method varchar(30) NOT NULL,

CONSTRAINT idop PRIMARY KEY(id\_o),

CONSTRAINT branchf FOREIGN KEY (branch) REFERENCES Branch(number)

ON UPDATE CASCADE ON DELETE SET NULL

);

1. **На основе описания предметной области построить объектно-реляционную модель. Привести описания типов данных UDT с указанием их свойств и методов. На основе пользовательских типов составить схемы отношения. Для каждого отношения указать ключи, ссылочный атрибут, ограничения и т.д. Задать связи между отношениями. Для любых двух пользовательских типов задать правила сравнения: 1) на равенство и полное, 2) поэлементное и через функцию. Привести описание функций сравнения.**

**Продемонстрировать наследование, ссылки и сложные типы данных. Принцип определения методов UDT соответствует принципам определения методов класса**

create type Client AS

(

Id\_c integer ,

fio varchar(100),

passport integer,

age integer,

city varchar(100),

address varchar(128),

contacts varchar(128)

)

// объявление метода

method Info( integer id\_c) returns integer;

// определение метода

create method Info() returns integer;

for Client

begin

return self.fio && self.passport

end;

create type Order AS

(

Id\_o integer,

date varchar(15) ,

summary integer ,

branch ref(Branch) [scope Br]

)

create type Addr as

(

Street varchar(50),

House integer,

Apart integer

);

create type Branch as

(

Number integer,

add Addr

);

create type Payment under Order as

(

Payment\_method varchar(30)

);

Cli (id\_c, fio, passport, age, address, contacts, city)

Cliord (id\_c(\*(Cli)), id\_o(\*(Ord)))

Br (number, addr)

Ord (id\_o, date, summary, branch(\*(PlOrds)))

Pay (id\_o, payment\_method)

create table Cli of Client

(

PRIMARY KEY(id\_c),

Id\_c NOT NULL,

fio NOT NULL,

passport NOT NULL,

age,  
 address,

contacts,

city,

age CHECK(age >= 18)

);

create table Cliord

(

Id\_c ref(Client) scope Cli,

Id\_o ref(Order) scope Ord

);

create table Br of Branch

(

PRIMARY KEY(number)

);

create table Ord of Order

(

PRIMARY KEY(id\_o),

Id\_o NOT NULL,

date NOT NULL,

summary,

numer ref(Branch) scope Br

);

create table Pay of Payment under Ord

(

Payment\_method NOT NULL

);

create ordering for Addr

<equals only | order full | relative with func> by state

create function func(x1 Addr, x2 Addr) returns int

if x1.street > x2.street then return 1;

else if x1.street < x2.street then return -1;

else if x1.house > x2.house then return 1;

else if x1.house < x2.house then return -1;

else if x1.apart > x2.apart then return 1;

else if x1.apart < x2.apart then return -1;

else return 0;

end if;

1. **На основе описания предметной области построить модель полу-структурированных данных. Привести пример графа полу-структурированных данных, соответствующий ему XML-документ и DTD-определение.**

**<?xml** version="1.0" encoding="UTF8" standalone="yes"**?>**

**<db>**

**<order** ido="o1" tocli="c1" tobr="br1"**>**

**<id\_o>**1**</id\_o>**

**<date>**20.03.2019**</date>**

**<summary>**500**</summary>**

**</order>**

**<client** idc="c1" too="o1"**>**

**<id\_c>**1**</id\_c>**

**<fio>** Berezin Ivan Sergeevich**<\fio>**

<passport> 4616240251 <\passport>

<age> 23 <\age>

<city> Moscow region <\city>

**<addr>**

**<street>**First**<\street>**

**<house>**3**<\house>**

**<apart>**5**<\apart>**

**<\addr>**

**<cont>**

**<phone>**+79851740745**<\phone>**

**<email>**uivan2009@rambler.ru**<\email>**

**<\cont>**

**</client>**

**<Branch** idbr="br1" too="o1"**>**

**<number>**1**</number>**

**<addr>**

**<city>**Moscow**<\city>**

**<street>**street **1<\street>**

**<\addr>**

**</branch>**

**</db>**

**<?xml** version="1.0" encoding="UTF8" standalone="yes"**?>**

<!DOCTYPE db [

<!ELEMENT order (id\_o, date, summary)>

<!ELEMENT id\_o (#PCDATA)>

<!ELEMENT date (#PCDATA)>

<!ELEMENT summary (#PCDATA)>

<!ATTLIST order ido ID #REQUIRED tocli IDREFS tobr IDREF>

<!ELEMENT client (id\_c, fio, passport, age, addr, cont, city)>

<!ELEMENT id\_c (#PCDATA)>

<!ELEMENT fio (#PCDATA)>

<!ELEMENT passport (#PCDATA)>

<!ELEMENT age (#PCDATA)>

<!ELEMENT addr (street,house,apart)>

<!ELEMENT street (#PCDATA)>

<!ELEMENT house (#PCDATA)>

<!ELEMENT apart (#PCDATA)>

<!ELEMENT cont (phone,email)>

<!ELEMENT phone (#PCDATA)>

<!ELEMENT email (#PCDATA)>

<!ELEMENT city (#PCDATA)>

<!ATTLIST client idc ID too IDREFS>

<!ELEMENT branch (number, addr)>

<!ELEMENT number (#PCDATA)>

<!ELEMENT addr(city, street)>

<!ELEMENT city (#PCDATA)>

<!ELEMENT street (#PCDATA)>

<!ATTLIST branch idbr ID #REQUIRED too IDREFS>

]>

**<db>**

...

**</db>**