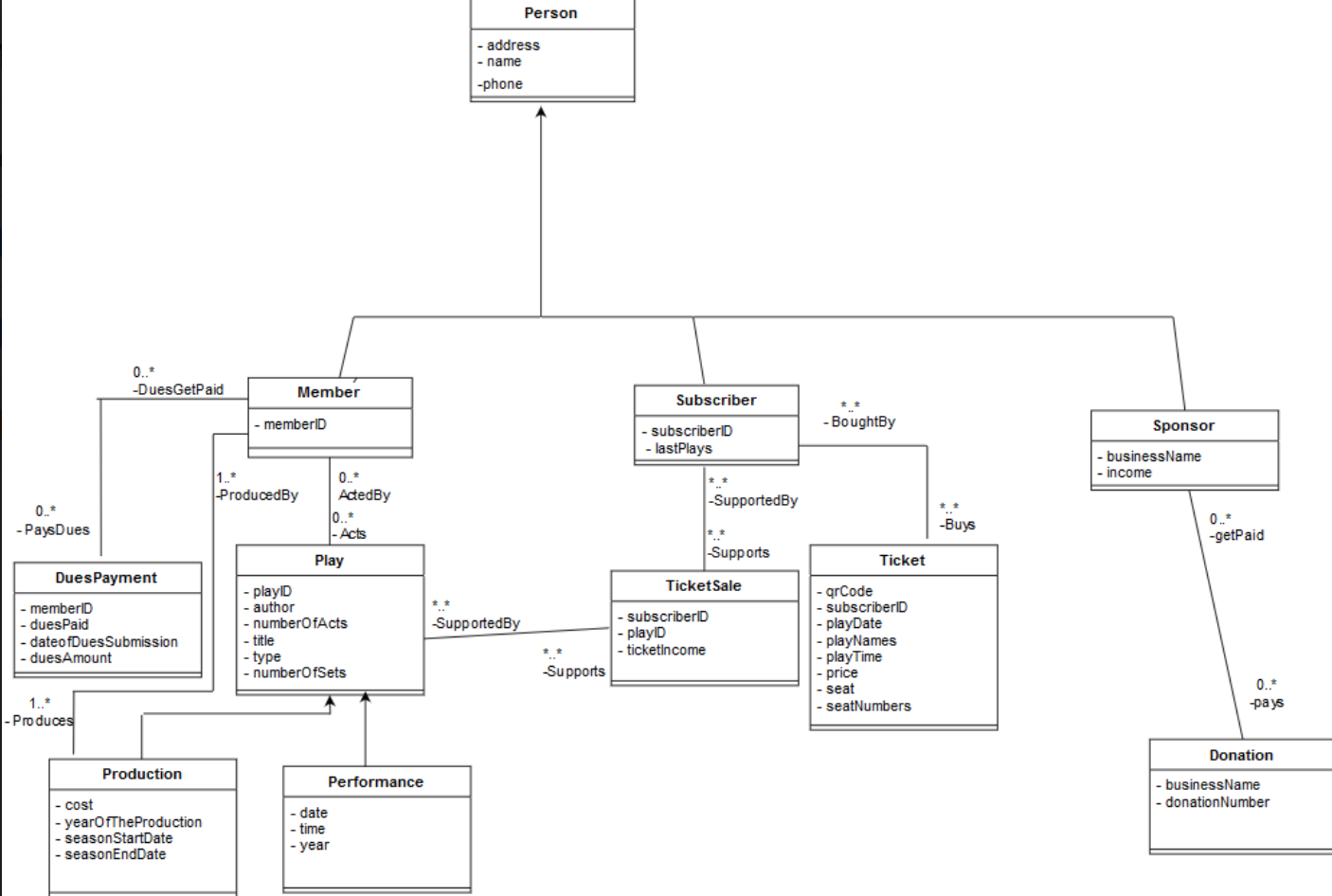
**Student Projects Chapter 7 - Drawing a UML Diagram and Creating an Object-Oriented Database for the Student Project**

Read the sample project steps for this chapter and apply the same techniques to the student project that you are developing.

* 7.1 - Create a UML diagram for the Student Project



* 7.2 - Convert the UML diagram into an Object-Oriented database model and write the ODL for the schema.

class Person

{

attribute Struct AddressType(string street, sting city, string state, string zip) address;

attribute Struct PhoneType(string areaCode, string telephoneNumber) phone;

string getName( );

void setName(string newName);

string getAddress( );

void setAddress(AddressType newAddress);

string getPhone( );

void setPhone(PhoneType newPhone);

};

class Member extends Person

(extent Member)

{

attribute int memberID;

int getMemberID();

void setMemberID(int newmemberid);

relationship Set<DuesPayment> PaysDues Inverse DuesPayment::DuesGetPaid;

relationship Set<Play> Acts Inverse Play::ActedBy;

};

class Subscriber extends Person

(extent Subscriber)

{

attribute int subscriberID;

attribute string lastPlays;

int getSubscriberID();

void setSubscriberID(int newsubscriberid);

string getPlays();

void setPlays (string newsubscriberplays);

relationship Set<Ticket> Buys Inverse Ticket::BoughtBy;

relationship Set<TicketSale> Supports Inverse Ticket::SupportedBy;

};

class Sponsor extends Person

(extent Sponsor)

{

attribute NameType businessName;

attribute real(8,2) income;

NameType getbusinessName();

void setbusinessName(NameType newsponsorbusinessName);

real getincome();

void setincome(real newsponsorIncome);

relationship Set<Donation> pays Inverse Donation:: getPaid;

};

class DuesPayment

(extent DuesPayment)

{

attribute int memberID;

attribute boolean duesPaid;

attribute string dateofDuesSubmission;

attribute int duesAmount;

int getmemberID();

void setmemberID(int newduespaymentmemberID)

boolean getDuesPaid();

void setDuesPaid(boolean newduespaymentPaid);

string getDateOfDuesSubmission();

void setDate(string newduespaymentdateofSubmission);

int getDuesAmount();

void setAmount(int newduespaymentAmount);

};

class Play

{

attribute int playID;

attribute string author;

attribute int numberOfActs;

attribute string title;

attribute string type;

attribute int numberOfSets;

int getPlayID();

void setID (int newplaygetPlayID);

string getAuthor();

void setAuthor (string newplayAuthor);

int getNumberOfActs();

void setNumberOfActs (int newplayNoActs);

string getTitle();

void setTitle (string newplayTitle);

string getType();

void setType (string newplayType);

int getNumberOfSets();

void setNumberOfSets (int newplaySets);

relationship set <Member> acts Inverse Member::ActedBy;

relationship set <TicketSale> supports Inverse TicketSale::SupportedBy;

};

class Production extends Play

(extent Production)

{

attribute real(8,2) cost;

attribute int yearOfTheProduction;

attribute string seasonStartDate;

attribute string seasonEndDate;

real getcost();

void setcost(real newproductionCost);

int getYearOfTheProduction();

void setYearOfTheProduction (int newproductionyear);

string getseasonStartDate();

void setseasonStartDate (string newproductionStart);

string getseasonEndDate();

void setEnd (string newproductionEnd);

};

class Performance extends Play

(extent Performance)

{

attribute DateType date;

attribute DateType time;

atribute DateType year;

DateType getDate();

void setDate(DateType newperformanceDate);

DateType getTime();

void setTime (DateType newperformanceTime);

DateType getYear();

void setYear (DateType newperformanceYear);

};

class TicketSale

(extent TicketSale)

{

attribute int subscriberID;

attribute int playID;

attribute real(8,2) ticketIncome;

int getsubscriberID();

void setsubscriberID(int newticketsalesubscriberID);

int getplayID();

void setplayID(int newticketsaleplayID);

real getIncome();

void setIncome(real newticketsaleincome);

relationship Set<Play> SupportedBy Inverse Play::Supports;

relationship Set<Subscriber> SupportedBy Inverse Subscriber::Supports;

};

class Ticket extends (Subscriber)

extent(Ticket)

{

attribute string qrCode;

attribute string subscriberID;

attribute DateType playDate;

attribute NameType playNames;

attribute int playTIme;

attribute real(8,2) price;

attribute string seat;

attribute int searNumbers;

string getqrCode();

void setqrCode(string newTicketqr);

string getsubscriberID();

void setsubscriberID(string newTicketSubscriberID);

DateType getplayDate();

DateType setplayDate(string newTicketplayDate);

NameType getplayNames();

NameType setplayNames(string newTicketplayNames);

int getplayTIme();

void setplayTIme(int newTicketplayTIme);

real getprice();

void setprice(real newTicketPrice);

string getseat();

void setseat(real newTicketSeat);

int getseatNumbers();

void setseatNumber(real newTicketSeatNumbers);

};

class Donation extends Sponsor

(extent Donation)

{

attribute NameType businessName;

attribute int donationNumber();

NameType getName();

void setName(NameType newDonationName);

int getdonationNumber();

void setdonationNumber(int newDonationNumber);

};

* 7.3 - Using InterSystems Caché or another object-oriented DBMS, create an object-oriented database for your schema. Note: If you are using InterSystems Caché you should complete the student laboratory exercise for this topic prior to doing this step. You might also watch the demonstrations and do the tutorials on this topic on the InterSystems Caché website.

Class Theater.Address Extends %SerialObject

{

/// street

Property street As %String(MAXLEN = 60);

/// city

Property city As %String;

/// zip

Property zip As %String(MAXLEN = 10);

/// state

Property state As %String;

}

Class Theater.telephone Extends %SerialObject

{

/// The area code

Property areaCode As %String(MAXLEN = 3);

/// The 7 digit phone number

Property phoneNumber As %String(MAXLEN = 8);

}

Class Theater.Person Extends %Persistent

{

/// The person's address

Property personAddress As Address;

/// The person's name

Property personName As %Name;

/// The person's phone

Property personPhone As telephone;

}

Class Theater.Member Extends (Theater. Person, %Persistent, %XML.Adaptor, %ZEN.DataModel.Adaptor)

{

Property memberID As %Int(MAXLEN = 9);

Relationship Acts As Theater.play [ Cardinality = many, Inverse = ActedBy ];

}

Class Theater.Play Extends %Persistent

{

/// Unique ID of the play

Property playID As %Int(MAXLEN = 8);

/// The author of the play

Property author As %String(MAXLEN = 30);

/// The number of Acts of the play

Property numberOfActs As %Int(MAXLEN = 20);

/// The title of the play

Property title As %String(MAXLEN = 20);

/// The type of the play

Property Type As %String(MAXLEN = 10);

/// The number of sets in the play

Property numberOfSets As %Int(MAXLEN = 20);

Relationship ActedBy As Theater.Member [Cardinality = many, Inverse = Acts]

Relationship DuesGetPaid As Theater.DuesPayment [ Cardinality = one, Inverse = DuesGetPaid ];

Relationship producedBy As Theater.Production [ Cardinality = many, Inverse = producedBy ];

Index ActedBy Index On ActedBy;

Index DuesGetPaid Index On DuesGetPaid ;

Index producedBy Index On producedBy ;

}

Class Theater.Donation Extends %Persistent

{

Property businessName As %String;

Property donationNumber As %Integer;

Relationship Pays As Theater.Sponsor [ Cardinality = one, Inverse = Pays ];

Index PaysIndex On Pays;

Relationship getPaid As Theater.Sponsor [ Cardinality = many, Inverse = getPaid ];

}

Class Theater.DuesPayment Extends %Persistent

{

Property memberID As %Integer;

Property duesPaid As %Boolean;

Property dateOfDuesSubmission As %Date;

Property duesAmount As %Double;

Relationship PaysDues As Theater.Member [ Cardinality = one, Inverse = PaysDues ];

Relationship DuesGetPaid As Theater.Member [ Cardinality = one, Inverse = DuesGetPaid ];

Index DuesGetPaidIndex On DuesGetPaid;

}

Class Theater.Performance Extends Theater.Play

{

Property date As %Date;

Property time As %Time;

Property year As %Integer;

}

Class Theater.Production Extends Theater.Play

{

Property cost As %Double;

Property yearOfTheProduction As %Integer;

Property seasonStartDate As %Date;

Property seasonEndDate As %Date;

Relationship Produces As Theater.Member [ Cardinality = many, Inverse = Produces ];

Relationship producedBy As Theater.Member [ Cardinality = one, Inverse = producedBy ];

Index producedByIndex On producedBy;

}

Class Theater.Sponsor Extends %Persistent

{

Property businessName As %String;

Property income As %Double;

Relationship Pays As Theater.Donation [ Cardinality = many, Inverse = Pays ];

Relationship getPaid As Theater.Donation [ Cardinality = one, Inverse = getPaid ];

Index getPaidIndex On getPaid;

}

Class Theater.Subscriber Extends (Theater.Person, %Persistent, %XML.Adaptor, %ZEN.DataModel.Adaptor)

{

Property subscriberID As %Integer;

Property lastPlays As %String;

Relationship Buys As Theater.Ticket [ Cardinality = many, Inverse = Buys ];

Relationship BoughtBy As Theater.Ticket [ Cardinality = one, Inverse = BoughtBy ];

Index BoughtByIndex On BoughtBy;

Relationship Supports As Theater.TicketSale [ Cardinality = many, Inverse = Supports ];

Relationship SupportedBy As Theater.TicketSale [ Cardinality = many, Inverse = SupportedBy ];

}

Class Theater.Ticket Extends %Persistent

{

Property qrCode As %String;

Property subscriberID As %Integer;

Property playDate As %Date;

Property playNames As %String;

Property playTime As %Time;

Property price As %Double;

Property seat As %String;

Property seatNumbers As %String;

Relationship Buys As Theater.Subscriber [ Cardinality = one, Inverse = Buys ];

Index BuysIndex On Buys;

Relationship BoughtBy As Theater.Subscriber [ Cardinality = many, Inverse = BoughtBy ];

}

Class Theater.TicketSale Extends %Persistent

{

Property subscriberID As %Integer;

Property playID As %Integer;

Property ticketIncome As %Double;

Relationship Supports As Theater.Subscriber [ Cardinality = one, Inverse = Supports ];

Index SupportsIndex On Supports;

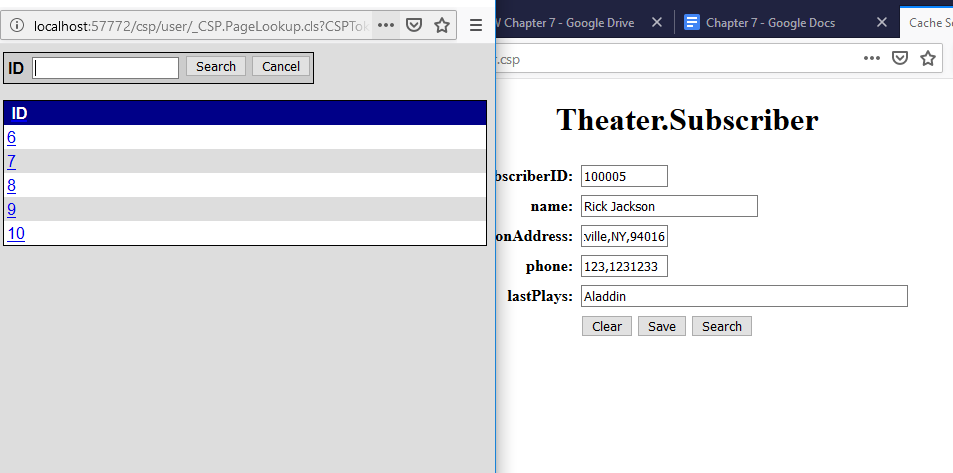
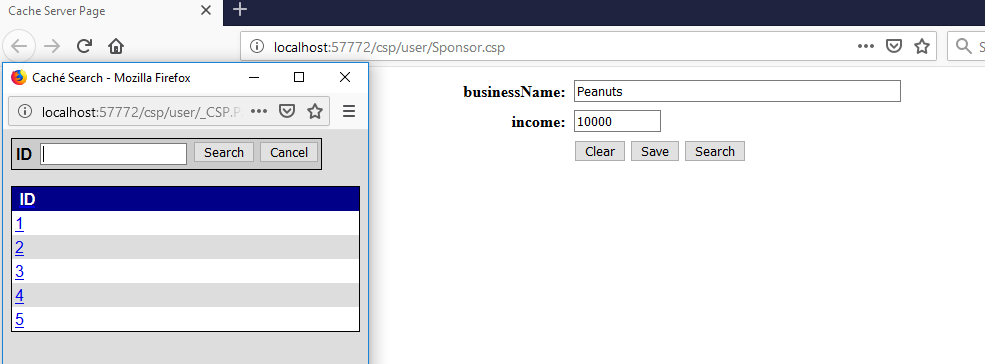
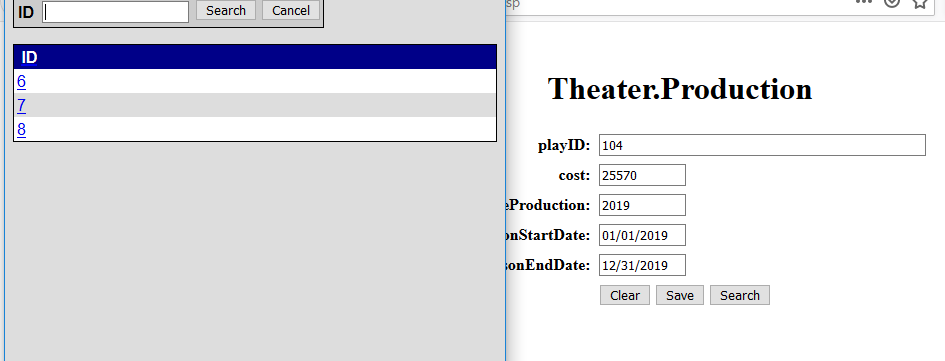
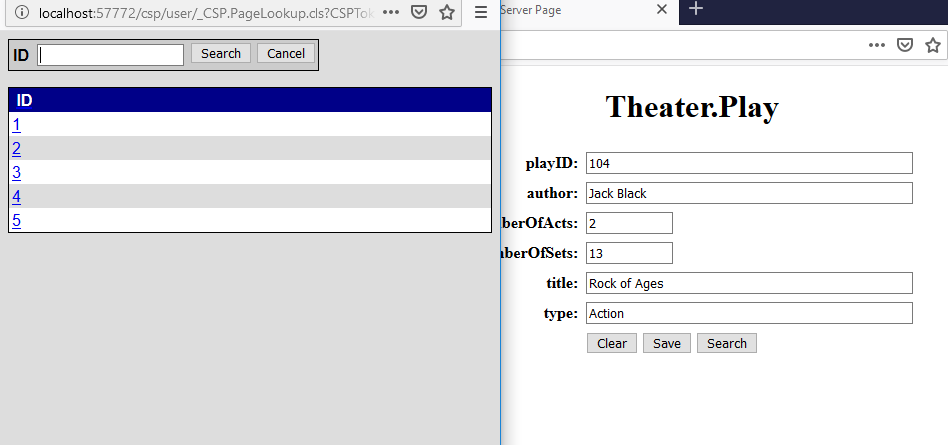
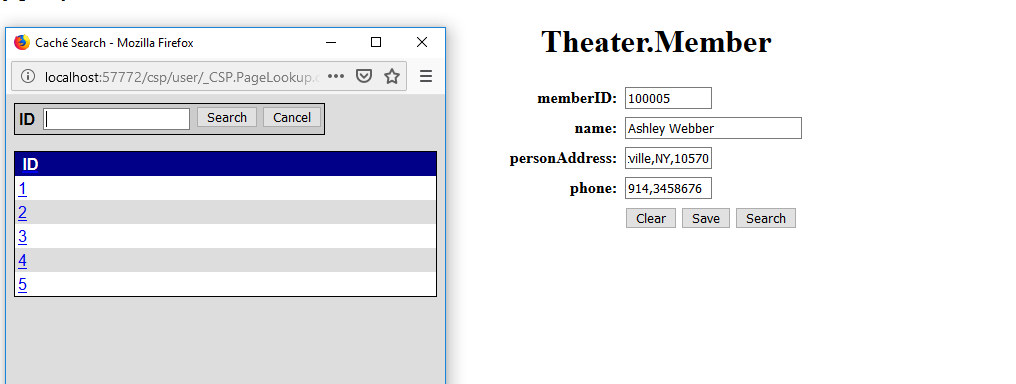
Relationship SupportedBy As Theater.Subscriber [ Cardinality = one, Inverse = SupportedBy ];

Index SupportedByIndex On SupportedBy;

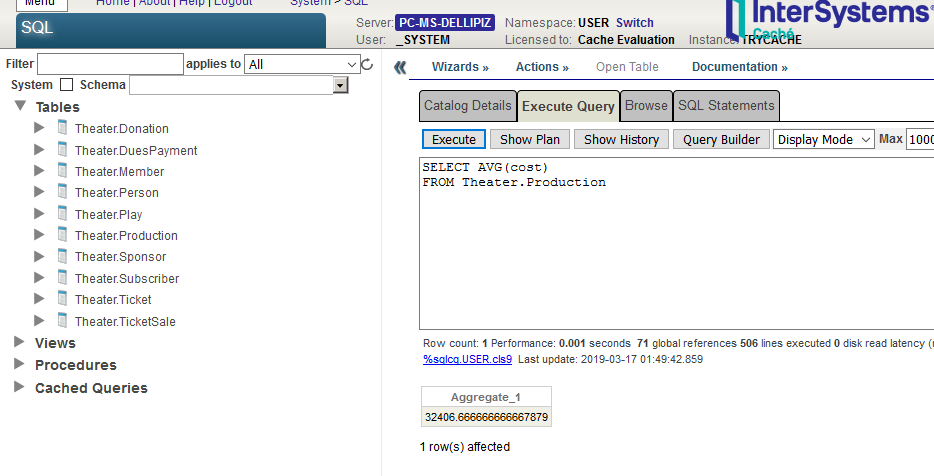
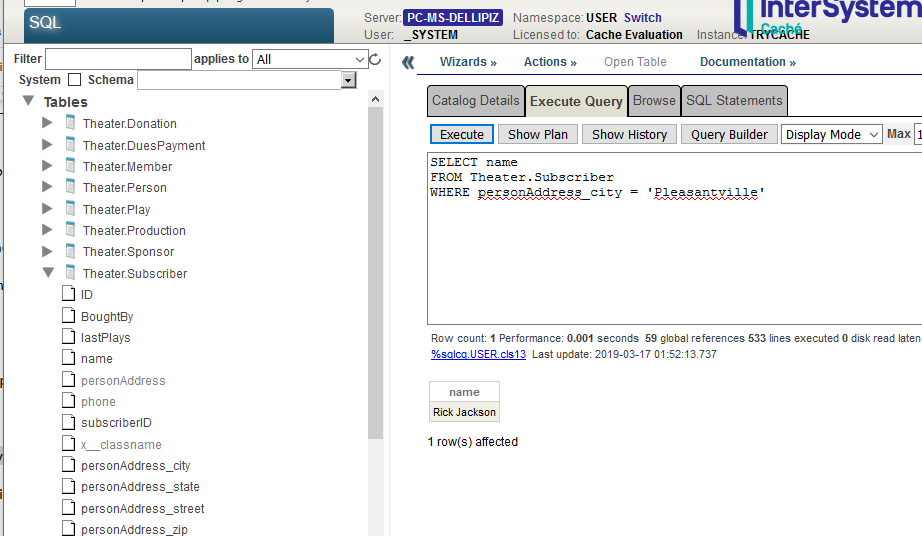
}

* 7.4 - Enter at least five records for each class in your database. If you are using InterSystems Caché you should complete the lab for this topic prior to doing this step. Be sure to create relationships between records, as shown in the laboratory exercise, using the Caché Terminal.

We used Cache’s Zen framework to develop a simple server page for each table, and entered data in each one by hand. Here are a few sample pages. The relationships were added through the terminal.



* 7.5 - Write out some queries for your database and print out the questions, the queries, and the results.

1. Get the average cost of a production
2. Find subscribers from Pleasantville
3. Find the plays where the number of sets is greater than the average for all the plays