12-31-1990

Implementation of object oriented university database using VODAK/VML-0 prototype

Abhay V. Bhave
New Jersey Institute of Technology

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

2) **Implementation of Object Oriented**

University Database

using

VODAK/VML-0 Prototype

Submitted to
Department of Computer and Information Science
New Jersey Institute of Technology

in partial fulfillment

of

the requirements for the degree
of Master of Science

by

1) Abhay Bhave

Advisor: Dr. Yehoshua Perl

Approval Sheet

Title of Thesis: Implementation of Object Oriented
University Database
using VODAK/VML-0 Prototype

Name of candidate: Abhay V. Bhave

Thesis and abstract approved: _____ 12.17.90

Dr. Y. Perl — Date
Professor
Dept of Computer & Informaton Sc.
New Jersey Institute of Technology

VITA

Name	:	Abhay V. Bhave
Permanent Address	:	
Degree and date to be conferred	:	MS in CIS, Dec 1990
Date of birth	:	
Place of birth	:	

Collegiate inst. attended	Dates	Degree	Date of Degree
N.J.I.T.	1988-1990	MS	Dec. 1990
V.J.T.I., India	1985-1988	BS	Jul. 1988
V.J.T.I., India	1981-1985	LTM	Jul. 1985

Major : Computer Science

ABSTRACT

In recent years object - oriented programming has gained a tremendous popularity in the design and implementation of emerging data - intensive application systems. Object oriented knowledge based approaches have proved to be very powerful vehicle when developing, integrating complex systems.

Using a University database demonstrated here is an object oriented model. This model uses VODAK/VML-0 prototype which was developed by GMD - IPSI, Darmstadt FRG. The Conceptual Schema of this database was done using The Dual Model, but as the current VODAK/VML version does not support dual model, the schema was modified before use. There are 167 classes defined and their interfaces consist of 60 methods coded in smalltalk. The prototype supports user defined datatypes and it is capable of supporting user defined queries on the database .

The VODAK datamodel prototype itself is developed using powerful features object oriented language "Smalltalk", which by itself make computer problem solving more human like activity.

ACKNOWLEDGEMENT

I would like to express my gratitude to Dr. Perl without whose valuable advice the completion of this project would not have been possible.

I would also like to thank Mr. Ashish Mehta at N.J.I.T. for the time and effort he put in our discussions.

TABLE OF CONTENTS

CHAPTER 1

Introduction

Object oriented programming offers a number of important advantages for applications over traditional control-oriented programming. One is the modelling of all conceptual entities with a single concept, namely, OBJECT.

An object represents anything from a simple number to complex entity. The state of an object is captured in the instance variables. The behavior of an object is captured in messages, to which an object responds. Thus the messages completely define the semantic behavior of an object.

Another advantage of object oriented programming is the notion of class hierarchy and inheritance of properties (instance variable and messages) along the class hierarchy. The class hierarchy captures roleof and catagoryof relations between a class and its subclass, equivalently a class and its superclass. All subclasses of a class, inherit all properties defined for the class and can have additional properties local to them. The notion of property inheritance along the hierarchy facilitates top-down design of the database.

The Dual model successfully differentiates between data types versus Object types, such that while data types represent structural behavior of an object, the object types describe semantic integrity of it in the object hierarchy. The Dual model has ability to capture two types of hierarchies by separating the semantic description from the structural parts.

The object class is defined to express that all instances of a class have the same structure and behavior, they are considered to be of same abstract data type, called the object type. Thus better abstraction mechanism is attained.

On one side there is a hierarchy of types (structures) and on the other a network of classes. Each class is related with one type of this hierarchy and the instances of that class are restricted to be instances of that type.

In this presentation an attempt has been made to emphasize the mentioned properties of the Dual Model using the University environment database as an example. The basic schema for this purpose is taken from the previous work [VAH] on Dual Model of University Database. VODAK which is used for implementing this schema is an object oriented prototype. Which is based on smalltalk-80 system. The necessary code is written in VODAK Modelling Language(VML).

The logical interface among the objects is written using Smalltalk . For storage purposes BOSS (Binary Object Stream System) is used, and the queries to access data from BOSS are also written using Smalltalk.

CHAPTER 2

VODAK DataModel and VML Datamodeling Language

VODAK is an object oriented database prototype, which was developed keeping in mind the Dual Model. VML is the modelling language used for this prototype.

The first version of VML is implemented in smalltalk-80 because of its rich features of rapid prototyping. As it is based on an object oriented language, it supports class constructs, object modeling and basic data types which are readily supported by smalltalk programming language.

The nature of the objects in which we are interested in is very diverse and a rich set of types is needed, in addition to the primitive types provided, in order to have adequate representation of the objects. For reasons of clarity there is a need to distinguish between data types typically found in programming languages and instance and class types developed as a basis for the University Database Model.

For Tuple type the syntax is:

[attribute1: type1, attribute2: type2,...]

For Enumerated the type syntax is:

(#enum1,#enum2,#enum3,...)

For Set type the syntax is:

{ type1 }

Following is a listing of the data types used in the example of University Database on VODAK with VML

datatype ZIPTYPE = a string of 5 digits

```
datatype AREACODE = an integer 3 digits

datatype LOCALPHONETYPE = an integer of 7 digits

datatype DAYTYPE = an integer between 1 and 31

datatype MONTHTYPE =

    ( #Jan, #Feb, #Mar, #Apr, #May, #Jun, #Jul, #Aug, #Sep, #Oct, #Nov, #Dec)

datatype YEARTYPE = an integer between 1800 and 2100

datatype SERIALTYPE = (#JUNIOR, #SENIOR, #THIRD, #FOURTH)

datatype EXTRATYPE =

    < tag1;maiden:STRING, serialno:SERIALTYPE >

datatype NAMETYPE = [first:STRING, middle: STRING,

     last: STRING, extra = EXTRATYPE)

datatype SEXTYPE = (#M, #F);

datatype STATUSTYPE = (#SINGLE, #MARRIED, #DIVORCED, #WIDOW)

datatype STREETADDRESSTYPE =

    [number: INTEGER, street: STRING, unit: UNITTYPE)

datatype CITYADDRESSTYPE =

    [city: STRING, state:STRING, ZIP: ZIPTYPE)

datatype ADDRESSTYPE = [streetaddress: STREETADDRESSTYPE,

    cityaddress: CITYADDRESSTYPE)

datatype TELEPHONETYPE =

    [ area : AREACODE, number: LOCALPHONETYPE]

datatype TELEPHONESTYPE = {TELEPHONETYPE}

datatype NATURAL = positive integer

datatype DEPARTMENTTYPE =

    [dept: STRING, companyname: STRING]

datatype COMPANYADDRESSTYPE =

    [dept: DEPARTMENTTYPE,
```

```
    streetaddress: STREETADDRESSTYPE,
    cityaddress: CITYADDRESSTYPE]
datatype DATETYPE =
    [day: DAYTYPE, month: MONTHTYPE, year: YEARTYPE]
datatype PERSDATATYPE =
    [name: NAMETYPE, sex: SEXTYPE,
     maritalstatus: STATUSTYPE, birthday:DATETYPE]
datatype UNDERYEARTYPE =
    (#FRESHMAN, #SOPHOMORE, #JUNIOR, #SENIOR)
datatype MAJMINTYPE = STRING
datatype MAJORTYPE = STRING
datatype PERCENTTYPE = a real number between 0 and 100
datatype BULDNAMETYPE = STRING
datatype ROOMTYPE =
    [buldname: BULDNAMETYPE, roomnumber: NATURAL]
datatype HOURTYPE = an integer between 1 and 12
datatype MINUTETYPE = an integer between 0 and 59
datatype DAYNIGHTTYPE = (#am,#pm)
datatype TIMETYPE =
    [hour: HOURTYPE, minute: MINUTETYPE,
    year: YEARTYPE, ampm: DAYNIGHTTYPE]
datatype DEGREETYPE = (#BS,#MS,#PhD,#Non-Mat)
datatype WEEKTYPE = NATURAL
datatype INVESTIGATORTYPE = STRING
```

The current VML version supports class constructs, class hierarchy and inheritance. The database is defined using above defined datatypes and top-down class constructs. The class constructs includes

identification section, subtype section and property section, wherein the attributes related to that particular class as well as its relationships are listed.The current VML version does not support semantic relations among instances like "roleof" or "catagoryof". For this reason the database schema made by previous workers [VAH] was changed to suite the available and supported facilities. The new database schema thus achieved is used for actually implementing it. Methods are utilized to gain required information from distant classes in the database by tracking down a path through the network of classes. Unfortunately "methods" are not supported by the current VML version.

To overcome this difficulty smalltalk modules are written seperately for each and every method related to the classes and then they were attched to the individual class defination.

CHAPTER 3

Implementation Procedures

After going through all the previous chapters and the referred documents, now it is time to learn about implementation procedures for each part, but before looking in to actual smalltalk code and logical flow underlying, it is better to go through some important smalltalk-80 terms.

Section 3.1

Definition of Classes and Compilation

The main way to find about classes in the system is to use a system class browser. A browser presents a hierarchical index to information. A smalltalk-80 class browser presents a hierarchical index to classes in the system. This index is independent of programming logic, it is designed solely for user access to class descriptions via subject categories. The browser presents categories that organize the classes within the system, and categories that organize messages within each class. The information about a class that a user can retrieve using a browser includes

* a comment about role of the class in the system
* a description of the part of the system class hierarchy
  in which the class is found
* a description of variables of a class
* a description of the messages and methods of the class,
  including comments about the use of the message and the
  design of the method
* a classification of the class with respect to other
  classes
* a classification of the messages of the class

* access to all methods in the system that send a particular message
* a list of messages sent in a particular method

The system browser also provides access to templates for defining new classes and templates for defining new messages.

the browser is made up of five sub views and two menu items labeled Class and Instance. Four of the sub views are fixed list menus; the fifth view in which methods can be defined and modified using the text editor.

Objects are instances of class. the objects and classes defined in the system communicate through messages. When an object receives a message, it responds to it by either returning a value by running its own method allocated to that type of message, or instead it generates a message for some other object or class. For each message that object can understand there is a method implemented in it's class.

There are instance methods or methods for objects. As classes are also objects, they respond to messages which are implemented as class methods. Class methods are used for interacting with class without creating its instance.

In the actual implementation of the university database there are total 167 classes defined, which are listed under Appendix A, to make these classes compatible with the VODAK system each class was defined as the VMLclass and then compiled. The new data types are also classes defined as the VMLclasses.

Section 3.2

Creation of the University Database

The next step after the Definition of the classes taking part in the data base , is to create instances for these classes and thus create the actual database. As mentioned before the size of this database is very large considering 167 classes and instances of each of these class plus methods which are written in smalltalk and attached to each class Definition. This is a too large size for the memory to store, to overcome this problem the schema is broken down in to four manageable size and logically self contained groups.

In a real database system an application database is stored on secondary storage and when ever required some part of it transferred to main memory by system. As VODAK does not have a database system to take care of this application, there was a necessity for some secondary storage system, to supplement the main prototype. Hence use of "BOSS" (Binary Object Stream system) was done. This system supports associative storage by using an index or a key of some kind which is called surrogate. For storage of the objects under the University Database a specific naming method is adopted. A file called VODAK.VSS is used for storing all objects. Thus by using correct surrogate access to any object is achieved. The methods which are written separately for each of these class VmlObject are also stored and loaded from this file. When a surrogate is passed to this VmlObject method to load it returns the object corresponding to that surrogate. Similarly if an object is passed to the method for getting persistent surrogate, it returns a surrogate for that object stored and thus this surrogate can ever be used for accessing that particular object.

The advantage of using such a storage is that when an object is made persistent, all other objects with which it has relationship are also made persistent. Similarly when an object is loaded, the other related objects are also loaded but they are accessible only through the main object. This advantages make the use of BOSS more supportive and the method of access is made easier.

The whole University Database, which is build on 167 classes, is divided into four manageable parts, thus any one of them or all can be loaded from VODAK.VSS into main memory or can be saved from memory in to VODAK.VSS. The four functional divisions of the University schema are

- Class Student, TRanscript, and related classes
- Class under Committee
- Class Employee, Resume and related classes
- Class University and other classes representing
  it's academic units.

Four database are such created are called

- Transcript
- Committee
- Resume
- University

All the databases are totally independent at a given time and individually can be loaded into main memory as per the need. Each one of the above databases has three files associated with it, namely

* Method
* Query
* Demonstration

Each demonstration file creates certain number of instances for the classes in its hierarchy. These instantiations are then saved on to the file VODAK.VSS. Thus for each one of the instances created, persistent storage is obtained and a surrogate is received in return through which it can be accessed for methods and queries related to it. The code related to each of this divisions is attached in Appendix B.

Section 3.3

Methods and Queries

As mentioned previously the current VML version does not support methods and hence separate smalltalk methods associated to each class Definition were written and then attached to the class Definition. More complex methods are built on other primitive methods. Such complex methods traverse through the schema via relationships provided in the class Definition.

Similarly VODAK not being a real database it does not support permanent storage hence BOSS was used and implementation of VODAK.VSS was done. As the attempt is to be made, so that this University Database is a real database, the important tool of any database the "Queries" are also written in smalltalk.

A set of queries have been prepared for each of the demonstration database in such a way that all methods defined for the classes, which that demonstration covers are used. When a query is run, the related methods are called to retrieve required data from the database. The part which retrieves information is constructed from many user defined methods. The part which formats the output is constructed by sending messages to predefined smalltalk classes.

A listing of all the methods and queries for each class are attached in Appendix C.

CHAPTER 4

CONCLUSION

The example of University Database used to demonstrate the object oriented database implementation is large. As the current prototype is having limitations regarding methods and it does not support relations the original dual model schema was flattened to suite it, thus loosing all the semantic relations. Separate methods were written and then used. VODAK not being a regular database, there is no sufficient storage to accommodate such a big database. Use of BOSS has supported the aim of making this database like a real database, even though the whole schema had to be broken down in four logical parts.

The four databases developed here, truly represent the original schema, and the demonstration developed for each of them show use of all the classes and methods under that database.

The other features of this model are the availability of composite data types needed for complex attributes and the ability to point out differences between data types and object types.

Though all the efforts are made to make this database as a real database, there are some limitations. There is no query language for this database thus no queries other than defined for the database can be used.

As the prototype is based on the smalltalk-80 system, the functioning is slower as this system is large and takes time to load, compile and execute.

The designed database can be extended to include some essential features one of them is the development of an user interface to support database functions like Create/Add/delete. Also some kind of consistency control has to be built into the system to support the above functionality.

BIBLIOGRAPHY

[ATK] M.Atkinson, F. Bancilhon, D.DeWitt, K.Dittrich, D.Maier, S. Zdonik, "The Object-oriented Database System Manifesto", Proceedings DOOD 89 Kyoto, December 1989.

[car] M.Carey, D.DeWitt, S.Vandenberg, "A Data Model and Query Language for EXODUS", ACM, 1989.

[Dit] K.Dittrich, "Object-oriented Database Systems A Workshop Report", Entity Relationship Approach, ERI 1987.

[Fis] D.Fishman, D.Beech, H.Cate, E. Chow, T.Connors, J. Davis, N.Derett, C. Hoch, P.Lyngbaek, B.Mahbod, M. Neimat, T.Ryan, M. Shah," IRIS : An object oriented Database Management Systems, VOL5, No1, January 1987.

[Hul] R.Hull, J. Su, "On Accessing Object-oriented Database : Expressive Power, Complexity and Restrictions", ACM 1989.

[Jag] D.Jagannathan, R. Guck, B.Fritchman, J.Thompson, T.Tolbert, "SIM : A Database system based on semantic data model", ACM 1988.

[Ken] W. Kent, " The leading edge of database technology", Information System Concept: An In depth Analysis, IFIP 1989.

[Kho] S. Khoshfian, G. Copeland, "Object Identity", OOPSLA'86 Proceedings, ACM 1986.

[Neh] E. Neuhold, Y. Perl, J. Geller, V. Turau,
" The Dual Model of Object-oriented Knowledge bases",
NJIT CIS TR No.23, 1989.

[Nie] O. Nierstrasz,"A survey of Object-oriented Concepts"
Object oriented Programming Languages, 1988.

[Pok] B. Pokknuri," Object oriented programming",
SIGPLAN Notices, Vol. 24, No. 11, 1988.

[Sny] A. Snyder, " Encapsulation and Inheritance in Object
oriented Programming Languages",OOPSLA'86 Proceedings
ACM 1986.

[Sto] M. Stonebraker, L. Rowe, B. Lindsay, J. Gray,
M. Carey, M. Brodie, P. Bernstein, D. Beech,
" Third generation Database System manifesto",
Proceedings of IFIP DS-4 Workshop OOD 1990.

[SMT] A. Goldberg, " Smalltalk-80, The Interactive
Programming Environment",
Addison-wesley Publishing Company, 1984.

[VAH] V. Teli, H. chao, "Development of University
Database using The Dual Model of Object oriented
Knowledge Bases", NJIT, Masters thesis 1990.

APPENDIX A

DEFINITION OF DATA TYPES AND CLASSES

"

DEFINITION OF DATATYPES

Following is the VML code for definition of the basic data types for this database. Each data type is defined as a Vml Class. A better way to do it is to define them as composite types using primitives like tuple, ennumerated and set types provided by Vml.
These definitions aer in the file **/OOUDB/impl/classDefs/datatypes.**

salil kulkarni
NJIT, Newark, NJ.
December 1990.

---"

```
DEFINE VmlClass Address
    category: 'UNIV-Db-Types'
    instanceType:
        [properties:
          [streetaddress: ↑StreetAddress;
            cityaddress: ↑CityAddress]].


DEFINE VmlClass StreetAddress
    category: 'UNIV-Db-Types'
    instanceType:
        [properties:
            [number: Integer;
             street: String;
             unit: String]].



DEFINE VmlClass CityAddress
    category: 'UNIV-Db-Types'
    instanceType:
        [properties:
            [city: String;
             state: String;
             zip: String]].



DEFINE VmlClass CompanyAddress
    category: 'UNIV-Db-Types'
```

```
    instanceType:
        [subtypeof: AddressType;
         properties:
            [department: String;
             companyName: String]].


DEFINE VmlClass DAte
    category: 'UNIV-Db-Types'
    instanceType:
        [properties:
            [day: Integer;
             month: (#Jan,#Feb,#Mar,#Apr,#May,#Jun,#July,#Aug,#Sep,#Oct,#Nov,#Dec);
             year: Integer]].


DEFINE VmlClass Name
    category: 'UNIV-Db-Types'
    instanceType:
        [properties:
            [first: String;
             middle: String;
             last: String;
             extra: String]].

DEFINE VmlClass PersonData
    category: 'UNIV-Db-Types'
    instanceType:
         [properties:
            [name: [first: String; middle: String; last: String; extra: String];
             sex: (#Male, #Female);
             maritalStatus: (#Single, #Married, #Divorced, #Widow);
             birthdate:
                 [day: Integer;
             month: (#Jan,#Feb,#Mar,#Apr,#May,#Jun,#July,#Aug,#Sep,#Oct,#Nov,#Dec);
                  year: Integer]]].



DEFINE VmlClass Room
    category: 'UNIV-Db-Types'
    instanceType:
        [properties:
            [buldName: String;
             roomNumber: String]].
```

```
DEFINE VmlClass Telephone
    category: 'UNIV-Db-Types'
    instanceType:
        [properties:
            [area: Integer;
             number: String]].


DEFINE VmlClass TIme
    category: 'UNIV-Db-Types'
    instanceType:
        [properties:
            [hour: Integer;
             minute: Integer;
             year: Integer;
             ampm: (#am,#pm)]].
```

" An example to demonstrate writing into and reading from tuple types.

```
| pd |
pd ← PersonData new.
pd name.first: 'Salil'; name.middle: 'Bachubhai'; name.last: 'Kulkarni';
    sex: 'Male'; maritalStatus: 'Married';
    birthdate.day: 22; birthdate.month: 'Oct'; birthdate.year: 1964.
pd name first.
pd birthdate."
```

"

DEFINITION OF ALL THE CLASSES IN THE UNIVERSITY DATABASE PROTOTYPE

Following is the VML code for definition of all the classes in the university database. These definitions are grouped according to the database in which a class participates, and the stored in nine files with **/OOUDB/impl/classDefs/.objs** *extensions.*

salil kulkarni
NJIT, Newark, NJ.
December 1990.

"----------"

"*Classes under, and related to Class Student*"

```
DEFINE VmlClass Person
    category: 'UNIV-Db'
    instanceType:
        [ properties:
            [perData: ↑PersonData;
             address: ↑Address;
             telephones: ↑Telephone;
             socialSecurityNr: Integer;
             visaStatus: String ]].


DEFINE VmlClass Student
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: PersonType;
         properties:
            [studentId: Integer;
             localAddress: ↑Address;
             degree: String;
             lastEducation: String;
             organization: ↑StudentOrganization;
             transcript: ↑TRanscript]].


DEFINE VmlClass Students
```

```
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: DbaseType;
         properties:
             [purpose: String;
              numStudents: Integer;
              setof: {↑Student}]].


DEFINE VmlClass NonMatriculate
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: StudentType;
         properties:
             [startingYear: Integer]].


DEFINE VmlClass Matriculate
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: StudentType;
         properties:
             [dateOfGraduation: ↑DAte]].


DEFINE VmlClass UnderGradStudent
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: MatriculateType;
         properties:
             [year: Integer;
              major: String;
              minor: String;
              project: String;
              supervisor: ↑FacultyMember]].


DEFINE VmlClass GraduateStudent
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: MatriculateType;
         properties:
             [underMajor: String;
```

```
        underMinor: String;
        major: String;
        supervisor: ↑Professor]].

DEFINE VmlClass MasterStudent
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: GraduateStudentType;
         properties:
            [project: String;
             thesis: String]].

DEFINE VmlClass PhDStudent
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: GraduateStudentType;
         properties:
            [masterMajor: String;
             masterProject: String;
             dissertation: String;
             specialArea: String;
             qualifyingStatus: String;
             dissertationComm: ↑Professors]].

DEFINE VmlClass StudentOrganization
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [numMembers: Integer;
             address: ↑CompanyAddress;
             chairPerson: ↑Student;
             members: ↑Students;
             workers: ↑Employees]].

DEFINE VmlClass CourseRecord
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [grade: Float;
             semesterTaken: String;
```

```
        course: ↑Course]].

DEFINE VmlClass CourseRecords
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numCourseRecords: Integer;
             transcript: ↑TRanscript;
             setof: {↑CourseRecord}]].

DEFINE VmlClass TRanscript
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [date: ↑DAte;
             student: ↑Student;
             currentSections: ↑Sections;
             courseRecords: ↑CourseRecords]].

DEFINE VmlClass Course
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [name: String;
             department: String;
             creditHours: Integer;
             number: String;
             preReq: ↑Courses;
             sections: ↑CrsSections]].

DEFINE VmlClass Courses
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
            numCourses: Integer;
            setof: {↑Course}]].
```

```
DEFINE VmlClass CrsSections
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [numSections: Integer;
             course: ↑Course;
             setof: {↑Section}]].


DEFINE VmlClass Section
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [sectionNo: String;
             noOfStudents: Integer;
             room: ↑Room;
             time: ↑TIme;
             capacity: Integer;
             students: ↑Students;
             instructor: ↑Instructor]].


DEFINE VmlClass Sections
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numSections: Integer;
             setof: {↑Section}]]


"----------------------------------------------------------------------------------------------------"

"Classes under, and related to Class Employee."


DEFINE VmlClass Employee
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: PersonType;
```

```
        properties:
            [employeeNr: Integer;
             totWeekHours: Integer;
             position: String;                    "selector 'type' is changed to 'position'"
             salaryRate: Float;
             started: Integer;
             officeAddress: ↑CompanyAddress;
             phoneNumber: ↑Telephone;
             supervisees: ↑Employees;
             resume: ↑Resume]].


DEFINE VmlClass Employees
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: DbaseType;
         properties:
            [purpose: String;
             numEmployees: Integer;
             setof: {↑Employee}]].


DEFINE VmlClass Instructor
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: EmployeeType;
         properties:
            [teachEval: Float;
             sections: ↑Sections]].


DEFINE VmlClass FacultyMember
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: InstructorType;
         properties:
            [officeHour: String;
             insurancePolicyNumber: String;
             grader: ↑Grader;
             committees: ↑Committees]].
"relationship 'committees' has been transfered from class Professor so that Special Lecturer can also be members
of some committees"


DEFINE VmlClass FacultyMembers
    category: 'UNIV-Db'
```

```
    instanceType:
        [properties:
            [purpose: String;
             numFacultyMembers: Integer;
             setof: {↑FacultyMember}]].


DEFINE VmlClass SpecialLecturer
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: FacultyMemberType;
         properties:
            [studiesStatus: String;
             contractPeriod: Integer;
             specialAreas: ↑Courses]].


DEFINE VmlClass Professor
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: FacultyMemberType;
         properties:
            [specialArea: String;
             supervisees: ↑Students;
             researchAss: ↑ResearchAssistants;
             facultyAss: ↑FacultyAssistants;
             department: ↑Department]].


DEFINE VmlClass Professors
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numProfessors: Integer;
             setof: {↑Professor}]].


DEFINE VmlClass AssiProfessor
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: ProfessorType;
         properties:
            [yearsOfContract: Integer;
             tenureConsider: String]].
```

```
DEFINE VmlClass VisitProfessor
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: ProfessorType;
         properties:
            [periodOfMonths: Integer;
             permanentCompAddress: ↑CompanyAddress;
             permanentCompPhone: {↑Telephone};
             permanentCompTitle: String]].


DEFINE VmlClass SeniorProfessor
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: ProfessorType;
         properties:
            [tenure: ↑DAte]].


DEFINE VmlClass AssocProfessor
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: SeniorProfessorType;
         properties:
            [assocAppointDate: ↑DAte]].


DEFINE VmlClass FullProfessor
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: SeniorProfessorType;
         properties:
            [fullAppointDate: ↑DAte]].


DEFINE VmlClass FullProfessors
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
```

```
            numFullProfs: Integer;
            setof: {↑FullProfessor}]].

DEFINE VmlClass DistProfessor
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: FullProfessorType;
         properties:
            [dateAppointed: ↑DAte]].


DEFINE VmlClass DistProfessors
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numDistProfs: Integer;
             setof: {↑DistProfessor}]].


DEFINE VmlClass Adjunct
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: InstructorType;
         properties:
            [permanentCompAddress: ↑CompanyAddress;
             permanentCompPhone: {↑Telephone};
             permanentCompTitle: String;
             numYearsThere: Integer]].

DEFINE VmlClass Adjuncts
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numAdjuncts: Integer;
             setof: {↑Adjunct}]].


DEFINE VmlClass FacultyUnion
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [name: String;
             members: ↑FacultyMembers;
             chair: ↑FacultyMember;
             represents: ↑FacultyMembers]].
```

```
DEFINE VmlClass Council
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [name: String;
             members: ↑Professors;
             chair: ↑Professor;
             represents: ↑Professors]].


DEFINE VmlClass Staff
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: EmployeeType;
         properties:
            [title: String]].


DEFINE VmlClass Staffs
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numStaffs: Integer;
             setof: {↑Staff}]].


DEFINE VmlClass PermanentStaff
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: StaffType;
         properties:
            [yearsOfEmplymnt: Integer;
             insurancePolicyNumber: String;
             salaryGridPoint: String]].


DEFINE VmlClass AdminStaff
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: PermanentStaffType;
```

```
        properties:
            [adminPosition: String;
             workFor: String]].


DEFINE VmlClass DeptStaff
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: PermanentStaffType;
         properties:
            [deptPosition: String;
             worksFor: ↑Department]].


DEFINE VmlClass StudentStaff
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: StaffType;
         properties:
            [semestersOfEmplymnt: Integer]].


DEFINE VmlClass AdminStudentStaff
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: StudentStaffType;
         properties:
            [adminPosition: String;
             workFor: String]].


DEFINE VmlClass OnCompusJob
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: AdminStudentStaffType;
         properties:
            [weekDays: String]].


DEFINE VmlClass DeptStudentStaff
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: StudentStaffType;
         properties:
            [deptPosition: String;
```

```
        workFor: ↑AcademicUnit]].


DEFINE VmlClass Grader
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: DeptStudentStaffType;
         properties:
            [courseLevel: String;
             sections: ↑Sections;
             supervisor: ↑FacultyMember]].


DEFINE VmlClass Assistant
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: DeptStudentStaffType;
         properties:
            [position: String]].


DEFINE VmlClass Assistants
    category: 'UNIV-Db'
    instanceType:
         [properties:
            [purpose: String;
             numAssistants: Integer;
             setof: {↑Assistant}]].


DEFINE VmlClass ResearchAssistant
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: AssistantType;
         properties:
            [postPercent: String;
             supervisor: ↑Professor]].


DEFINE VmlClass ResearchAssistants
    category: 'UNIV-Db'
    instanceType:
```

```
        [properties:
            [purpose: String;
             numResearchAssitants: Integer;
              setof: {↑ResearchAssistant}]].


DEFINE VmlClass GraduateAssistant
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: AssistantType;
         properties:
            [postPercent: String;
             supervisor: ↑AdminStaff]].



DEFINE VmlClass GraduateAssistants
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numGraduateAssitants: Integer;
             setof: {↑GraduateAssistant}]].

DEFINE VmlClass FacultyAssistant
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: AssistantType;
         properties:
            [postPercent: String;
             supervisor: ↑Professor]].



DEFINE VmlClass FacultyAssistants
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numFacultyAssitants: Integer;
              setof: {↑FacultyAssistant}]].


DEFINE VmlClass TeachingAssistant
    category: 'UNIV-Db'
    instanceType:
```

```
        [subtypeof: AssistantType;
         properties:
            [postPercent: String;
             numOfCourses: Integer;
             supervisor: ↑Professor;
             teachAssign: ↑Sections]].


DEFINE VmlClass TeachingAssistants
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numTeachingAssitants: Integer;
             setof: {↑TeachingAssistant}]].



DEFINE VmlClass AdministrativeAppointment
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: EmployeeType;
         properties:
            [name: String;
             office: ↑CompanyAddress;
             telephones: {↑Telephone};
             degree: String;
             responsibilities: String;
             comChair: ↑Committee]].
"relationship 'comChair' has been added to allow DeptChairPerson to be chair of DeptExecutiveCommittee "



DEFINE VmlClass President
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: AdministrativeAppointmentType;
         properties:
            [yearsInCharge: Integer;
             incharge: ↑University]].



DEFINE VmlClass VicePresident
    category: 'UNIV-Db'
    instanceType:
```

```
        [subtypeof: AdministrativeAppointmentType;
         properties:
            [yearsAssigned: Integer;
             incharge: String;
             supervisor: ↑President]].


DEFINE VmlClass VPAcademic
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: VicePresidentType;
         properties:
            [dummy: String]].


DEFINE VmlClass VPAdminTreasurer
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: VicePresidentType;
         properties:
            [dummy: String]].


DEFINE VmlClass VPDevelopement
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: VicePresidentType;
         properties:
            [dummy: String]].


DEFINE VmlClass Dean
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: AdministrativeAppointmentType;
         properties:
            [yearsAssigned: Integer;
             supervisor: ↑VPAcademic]].
```

```
DEFINE VmlClass CollegeDean
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: DeanType;
         properties:
            [adminRespon: String]].


DEFINE VmlClass SchoolDean
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: DeanType;
         properties:
            [ inCharge: String]].


DEFINE VmlClass DivisionDean
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: DeanType;
         properties:
            [ inCharge: String]].


DEFINE VmlClass AssociateDean
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: DeanType;
         properties:
            [ jobDescription: String]].


DEFINE VmlClass DeptChairPerson
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: AdministrativeAppointmentType;
         properties:
            [yearsAssigned: Integer;
             inCharge: ↑Department]].
```

"----------"

"*Classes under, and related to Class Resume.*"

```
DEFINE VmlClass Resume
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [jobTitle: String;
             belongsTo: ↑Employee;
             personalData: ↑PersonData;
             formalEducation: ↑FormalEducations;
             informalEducation: ↑InFormalEducations;
             teachingActivity: ↑TeachingActivities;
             experience: ↑Experiences;
             lisence: ↑ProfessionalLisence;
             grant: ↑ResearchGrants;
             honor: ↑Honors;
             patent: ↑PatentsAwarded;
             references: ↑References;
             publications: ↑Publications;
             presentataions: ↑ProfessionalPresents]].


DEFINE VmlClass FormalEducation
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [area: String;
             yrGranted: String;
             university: String;
             degree: String]].


DEFINE VmlClass FormalEducations
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numFormalEducations: Integer;
```

```
        bachelorDegree: ↑BachelorDegrees;
        masterDegree: ↑MasterDegrees;
        phdDegree: ↑PhDDegrees;
        setof: {↑FormalEducation}]].


DEFINE VmlClass BachelorDegree
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: FormalEducationType;
         properties:
            [projectTitle: String]].


DEFINE VmlClass BachelorDegrees
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [numBSDegrees: Integer;
             setof: {↑BachelorDegree}]].


DEFINE VmlClass MasterDegree
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: FormalEducationType;
         properties:
            [thesisTitle: String]].


DEFINE VmlClass MasterDegrees
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [numMSDegrees: Integer;
             setof: {↑MasterDegree}]].


DEFINE VmlClass PhDDegree
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: FormalEducationType;
```

```
        properties:
            [dissertationTitle: String]].


DEFINE VmlClass PhDDegrees
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [numPhDDegrees: Integer;
             setof: {↑PhDDegree}]].


DEFINE VmlClass InFormalEducation
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [course: String;
             location: ↑Address;
             description: String;
             date: ↑DAte]].


DEFINE VmlClass InFormalEducations
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numInformalEducations: Integer;
             setof: {↑InFormalEducation}]].


DEFINE VmlClass TeachingActivity
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [course: String;
             datesOffered: {↑DAte}]].


DEFINE VmlClass TeachingActivities
    category: 'UNIV-Db'
    instanceType:
```

```
    [properties:
        [numTeachingActivities: Integer;
         academic: ↑AcademicTeachings;
         nonAcademic: ↑NonAcademicTeachings;
         setof: {↑TeachingActivity}]].


DEFINE VmlClass AcademicTeaching
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: TeachingActivityType;
         properties:
            [courseListing: ↑Course]].


DEFINE VmlClass AcademicTeachings
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [numAcadTeachings: Integer;
             setof: {↑AcademicTeaching}]].


DEFINE VmlClass NonAcademicTeaching
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: TeachingActivityType;
         properties:
            [typeOfCourse: String;
             location: ↑CompanyAddress;
             periodOfTime: String;
             institution: String]].


DEFINE VmlClass NonAcademicTeachings
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [numNonAcadTeachings: Integer;
             setof: {↑NonAcademicTeaching}]].
```

```
DEFINE VmlClass Reference
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: PersonType;
         properties:
            [position: String;
             buisness: String;
             rank: String]].


DEFINE VmlClass References
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numReferences: Integer;
             setof: {↑Reference}]].


DEFINE VmlClass ResearchGrant
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [status: String;
             agency: String;
             title: String;
             amount: Float;
             dates: {↑DAte}]].


DEFINE VmlClass ResearchGrants
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numResearchGrants: Integer;
             setof: {↑ResearchGrant}]].


DEFINE VmlClass ProfessionalLisence
    category: 'UNIV-Db'
```

```
    instanceType:
        [properties:
            [title: String;
             lisenceNum: Integer;
             dates: {↑DAte}]].


DEFINE VmlClass ProfessionalLisences
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numProfLisences: Integer;
             setof: {↑ProfessionalLisence}]].


DEFINE VmlClass PatentAwarded
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [author: String;
             titles: String;
             patentNum: Integer;
             dates: {↑DAte}]].


DEFINE VmlClass PatentsAwarded
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numPatents: Integer;
             setof: {↑PatentAwarded}]].


DEFINE VmlClass Honor
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [honor: String;
         dates: {↑DAte}]].
```

```
DEFINE VmlClass Honors
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numHonors: Integer;
             setof: {↑Honor}]].


DEFINE VmlClass ServiceActivity
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [service: String;
             department: String;
             organization: String]].


DEFINE VmlClass ServiceActivites
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numServices: Integer;
             setof: {↑ServiceActivity}]].


DEFINE VmlClass Experience
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [title: String;
             startingDate: ↑DAte;
             endingDate: ↑DAte;
             employer: String;
             acadAppointments: ↑AcademicAppointments;
             nonAcadAppointments: ↑NonAcademicAppointments;
             consultings: ↑Consultings]].


DEFINE VmlClass Experiences
    category: 'UNIV-Db'
    instanceType:
```

```
        [properties:
            [numExperiences: Integer;
             setof: {↑Experience}]].


DEFINE VmlClass AcademicAppointment
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: ExperienceType;
         properties:
            [tenure: {↑DAte}]].


DEFINE VmlClass AcademicAppointments
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [numAcadAppointments: Integer;
             setof: {↑AcademicAppointment}]].


DEFINE VmlClass NonAcademicAppointment
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: ExperienceType;
         properties:
            [workDescriptions: String]].


DEFINE VmlClass NonAcademicAppointments
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [numNonAcadAppoints: Integer;
             setof: {↑NonAcademicAppointment}]].


DEFINE VmlClass Consulting
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: ExperienceType;
         properties:
```

```
        [organization: String;
         natureOfWork: String]].


DEFINE VmlClass Consultings
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [numConsultings: Integer;
             setof: {↑Consulting}]].
```

"----------"

"*Classes under, and related to Class Publication*"

```
DEFINE VmlClass Publication
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [authors: String;
             title: String;
             year: String]].


DEFINE VmlClass Publications
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [numPulications: Integer;
             journals: ↑RefereedJournalPapers;
             books: ↑Books;
             articles: ↑NonRefereedArticles;
             conferences: ↑RefereedConferencePapers;
             setof: {↑Publication}]].


DEFINE VmlClass Book
    category: 'UNIV-Db'
```

```
    instanceType:
        [subtypeof: PublicationType;
         properties:
            [editor: String;
             publisher: String;
             numPages: Integer;
             location: ↑Address]].


DEFINE VmlClass Books
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numBooks: Integer;
             setof: {↑Book}]].


DEFINE VmlClass RefereedJournalPaper
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: PublicationType;
         properties:
            [journal: String;
             volume: Integer;
             pageNum: Integer;
             typeOfReview: String]].


DEFINE VmlClass RefereedJournalPapers
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numRefJournalPapers: Integer;
             setof: {↑RefereedJournalPaper}]].


DEFINE VmlClass RefereedConferencePaper
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: PublicationType;
         properties:
```

```
            [conference: String;
             volume: Integer;
             pageNum: Integer;
             location: ↑CityAddress]].


DEFINE VmlClass RefereedConferencePapers
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numRefConfPapers: Integer;
             setof: {↑RefereedConferencePaper}]].


DEFINE VmlClass NonRefereedArticle
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: PublicationType;
         properties:
            [conference: String;
             volume: Integer;
             pageNum: Integer;
             location: ↑CityAddress]].


DEFINE VmlClass NonRefereedArticles
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numNonRefArticles: Integer;
             setof: {↑NonRefereedArticle}]].


DEFINE VmlClass ProfessionalPresent
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [meeting: String;
             location: ↑Address;
             date: ↑DAte]].
```

```
DEFINE VmlClass ProfessionalPresents
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numProfPresents: Integer;
             setof: {↑ProfessionalPresent}]].
```

"----------"

"*Classes under, and related to Class Committee.*"

```
DEFINE VmlClass Committee
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [name: String;
             code: String;                "this property is added to aid querying"
             functionality: String;
             timeOfMeeting: ↑TIme;
             maxCommitteNum: Integer]].


DEFINE VmlClass Committees
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: DbaseType;
         properties:
            [purpose: String;
             numCommittees: Integer;
             setof: {↑Committee}]].


DEFINE VmlClass UniversityCommittee
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: CommitteeType;
```

```
        properties:
            [administraRespon: String]].


DEFINE VmlClass UniversityCommittees
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numUnivComm: Integer;
             setof: {↑UniversityCommittee}]].


DEFINE VmlClass UniversityBCommittee
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: UniversityCommitteeType;
         properties:
            [chair: ↑FullProfessor;
             members: ↑FullProfessors]].


DEFINE VmlClass UniversityACommittee
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: UniversityCommitteeType;
         properties:
            [confidentialityStatus: Boolean]].


DEFINE VmlClass UnivDistProfCommittee
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: UniversityACommitteeType;
         properties:
            [chair: ↑DistProfessor;
             members: ↑DistProfessors]].


DEFINE VmlClass UnivSabaticalCommittee
    category: 'UNIV-Db'
    instanceType:
```

```
        [subtypeof: UniversityACommitteeType;
         properties:
            [numSabaticalPos: Integer;
             submissionDeadline: ↑DAte;
             chair: ↑Professor;
             members: ↑Professors]].


DEFINE VmlClass UnivPTCommittee
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: UniversityACommitteeType;
         properties:
            [submissionDeadLine: ↑DAte;
             decesionDeadLine: ↑DAte;
             chair: ↑FullProfessor;
             members: ↑FullProfessors]].


DEFINE VmlClass AcademicUnitCommittee
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: CommitteeType;
         properties:
            [academicRespon: String]].


DEFINE VmlClass UnderGradCommittee
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: AcademicUnitCommitteeType;
         properties:
            [numUnderGradCourses: Integer;
             members: ↑FacultyMembers;
             chair: ↑BSAdvisor]].


DEFINE VmlClass GraduateCommittee
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: AcademicUnitCommitteeType;
         properties:
              [members: ↑FacultyMembers;
```

```
            chair: ↑MSAdvisor]].


DEFINE VmlClass SeniorCommittee
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: AcademicUnitCommitteeType;
         properties:
            [confidentialityStatus: Boolean]].


DEFINE VmlClass ExecutiveCommittee
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: SeniorCommitteeType;
         properties:
            [members: ↑Advisors]].


DEFINE VmlClass SchoolExecutiveCommittee
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: ExecutiveCommitteeType;
         properties:
            [chair: ↑SchoolDean]].


DEFINE VmlClass DeptExecutiveCommittee
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: ExecutiveCommitteeType;
         properties:
            [chair: ↑DeptChairPerson]].


DEFINE VmlClass PhDCommittee
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: SeniorCommitteeType;
         properties:
            [numPhDStudents: Integer;
             dateQualifyExam: ↑DAte;
```

```
            chair: ↑PhDAdvisor;
            members: ↑Advisors]].


DEFINE VmlClass PTCommittee
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: SeniorCommitteeType;
         properties:
            [tenureDeadLine: ↑DAte;
             promotionDeadLine: ↑DAte;
             members: ↑FullProfessors]].


DEFINE VmlClass DeptPTCommittee
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: PTCommitteeType;
         properties:
            [chair: ↑DeptChairPerson]].


DEFINE VmlClass ResourceCommittee
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: AcademicUnitCommitteeType;
         properties:
            [members: ↑FacultyMembers;
             chair: ↑ResourceAdvisor;
             budget: ↑Budget]].


DEFINE VmlClass ColloquiumCommittee
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: AcademicUnitCommitteeType;
         properties:
            [colloquiumHour: ↑TIme;
             honorarium: String;
             members: ↑FacultyMembers;
             chair: ↑Professor;
             budget: ↑Budget]].
```

"----------"

"*Classes under, and related to Class Advisor.*"

```
DEFINE VmlClass Advisor
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: EmployeeType;
         properties:
            [appointmentDate: ↑DAte;
             committees: ↑Committees]].
```

"*relationship 'committees' has been added so that an advisor can be a member of a DepExecutiveCommittee*"

```
DEFINE VmlClass Advisors
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numAdvisors: Integer;
             setof: {↑Advisor}]].
```

```
DEFINE VmlClass ResourceAdvisor
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: AdvisorType;
         properties:
            [incharge: ↑ResourceCommittee]].
```

```
DEFINE VmlClass BSAdvisor
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: AdvisorType;
         properties:
            [incharge: ↑UnderGradCommittee]].
```

```
DEFINE VmlClass MSAdvisor
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: AdvisorType;
         properties:
            [incharge: ↑GraduateCommittee]].


DEFINE VmlClass PhDAdvisor
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: AdvisorType;
         properties:
            [incharge: ↑PhDCommittee]].
```

"----------"

"*Classes under AcademicFacility, AcademicUnit, Budget, and also Classes University, Division and College.*"

```
DEFINE VmlClass University
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: DbaseType;
        properties:
            [name: String;
             office: ↑CompanyAddress;
             telephones: {↑Telephone};
             president: ↑President;
             vpAcademic: ↑VPAcademic;
             vpAdminTreasu: ↑VPAdminTreasurer;
             vpDevelopement: ↑VPDevelopement;
             facultyMembers: ↑FacultyMembers;
             staffMembers: ↑Staffs;
             adjuncts: ↑Adjuncts;
             assistants: ↑Assistants;
             schools: ↑Schools;
             colleges: ↑Colleges;
             divisions: ↑Divisions]].
```

```
DEFINE VmlClass Division
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [name: String;
             office: ↑CompanyAddress;
             telephones: {↑Telephone};
             dean: ↑DivisionDean]].


DEFINE VmlClass Divisions
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numDivisions: Integer;
             setof: {↑Division}]].


DEFINE VmlClass College
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [name: String;
             office: ↑CompanyAddress;
             telephones: {↑Telephone};
             numDepartments: Integer;
             departments: ↑Departments;
             dean: ↑CollegeDean]].


DEFINE VmlClass Colleges
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numColleges: Integer;
             setof: {↑College}]].
```

```
DEFINE VmlClass AcademicUnit
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [name: String;
             office: ↑CompanyAddress;
             telephones: {↑Telephone};
             degreeOffered: String;
             majorOffered: String;
             offices: ↑Offices;
             computers: ↑Computers;
             laboratories: ↑Laboratories;
             budget: ↑Budget]].


DEFINE VmlClass School
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: AcademicUnitType;
         properties:
            [dean: ↑SchoolDean]].


DEFINE VmlClass Schools
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numSchools: Integer;
             setof: {↑School}]].


DEFINE VmlClass Department
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: AcademicUnitType;
         properties:
            [ college: ↑College;
             chairPerson: ↑DeptChairPerson;
             deptResourceAdvisor: ↑ResourceAdvisor;
             deptBSAdvisor: ↑BSAdvisor;
             deptMSAdvisor: ↑MSAdvisor;
             deptPhDAdvisor: ↑PhDAdvisor]].
```

```
DEFINE VmlClass Departments
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numDepts: Integer;
             setof: {↑Department}]]   .


DEFINE VmlClass AcademicFacility
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [facilityId: String;
             location: ↑CompanyAddress;
             belongsTo: ↑AcademicUnit]].


DEFINE VmlClass Computer
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: AcademicFacilityType;
         properties:
            [type: String;
             usage: String;
             computerDetails: String;
             computerAssist: ↑GraduateAssistant;
             supervisor: ↑ResourceAdvisor]].


DEFINE VmlClass Computers
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numComputers: Integer;
             setof: {↑Computer}]].


DEFINE VmlClass Laboratory
    category: 'UNIV-Db'
```

```
    instanceType:
        [subtypeof: AcademicFacilityType;
         properties:
            [telephone: {↑Telephone};
             instruments: String;
             labName: String;
             labAssistant: ↑GraduateAssistant;
             supervisor: ↑ResourceAdvisor]].


DEFINE VmlClass Laboratories
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numlabs: Integer;
             setof: {↑Laboratory}]].


DEFINE VmlClass Office
    category: 'UNIV-Db'
    instanceType:
        [subtypeof: AcademicFacilityType;
         properties:
            [telephone: {↑Telephone};
             noOfStaff: Integer;
             employeeInfor: ↑Staff]].


DEFINE VmlClass Offices
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [purpose: String;
             numOffices: Integer;
             setof: {↑Office}]].


DEFINE VmlClass Budget
    category: 'UNIV-Db'
    instanceType:
        [properties:
            [source: String;
```

APPENDIX B

IMPLEMENTATION OF THE INTERFACE OF CLASSES

CLASS WISE IMPLEMENTATION OF METHODS FOR CLASSES PARTICIPATING IN UNIVERSITY DATABASE.

The following Smalltalk code is in the file **/OOUDB/impl/classInterface/univDBif**

salil Kulkarni
NJIT, Newark, NJ.
December 1990

CLASS UniversityType

getAllColDeans
"*A query to displays names of deans of all colleges in this university on system transcript window*"

```
    self collegeDeans.
```

getAllDepChairs
"*A query to displays names of chair persons of all departments in this university on system transcript window*"

```
    self chairPersons.
```

getAllDivDeans
"*A query to displays names of deans of all divisions in this university on system transcript window*"

```
    self divisionDeans.
```

getAllSchDeans
"*A query to displays names of deans of all schools in this university on system transcript window*"

```
    self schoolDeans.
```

numColleges
"returns number of colleges (as a integer) this university has"

```
    ↑self colleges numColleges.
```

numDepartments
"returns number of departments (as a integer) this university has"

```
    | total |
    total ← 0.
    self colleges setof do:[:each |
                    total ← total + each numOfDepartments].
    ↑total.
```

numDivisions
"returns number of divisions (as a integer) this university has"

```
    ↑self divisions numDivisions.
```

numSchools
"returns number of schools (as a integer) this university has"

```
    ↑self schools numSchools.
```

chairPersons
"displays a set of department chair persons of this university on system transcript window, else if a college has no department, a message is displayed "

```
    | n |
    n ← 0.
    Transcript refresh; clear; cr;
                . next: 7 put: $ ;
                  show: 'Names of Chair Persons of each department in
UCLA' asUppercase; cr;
```

```
            next: 25 put: $ ;
            show: 'University Persident: ';
            show: self president personName; cr;
            show: '------------------------------------------------------------------'; cr; cr.

    ((self colleges) = nil)
    ifFalse:[                                   "if there are colleges in this university"
    self colleges setof do:[:each |
            Transcript cr;show: 'College name:';
                show: each name asUppercase; crtab.
                ((each departments) = nil )
                ifFalse:[    " if there are departments in this college "
                each departments setof do:[:dep |
                Transcript show: 'Department name: ';
                    show: dep name; crtab;
                    show: '          Chair: ';
                    show: dep chairPerson personName; crtab.
                    n ← n + 1] ]
                ifTrue:[    " if there are no departments in college "
              Transcript show: 'No Department in this College'; crtab] ].

    Transcript cr;
            show: '------------------------------------------------------------------'; cr; cr;
            show: 'Total chair persons: '; show: n printString; endEntry]

    ifTrue:[                                    "if there are no colleges in this university"
        Transcript show: 'No Colleges in this University !!'; endEntry].
```

collegeDeans

"*displays names of college deans of all colleges in this university on system transcript window else if a university has no colleges, a message is displayed* "

```
    | nl
    n ← 0.
    Transcript refresh; clear; cr;
            next: 7 put: $ ;
            show: 'Names of college deans of each college in UCLA' asUppercase; cr;
            next: 25 put: $ ;
            show: 'University Persident: ';
            show: self president personName; cr;
            show: '------------------------------------------------------------------'; cr;cr.

    ((self colleges) = nil)
    ifFalse:[                                   "if there are colleges in this university"
```

```
self colleges setof do:[:each | Transcript  show: 'College name: ';
                                            show: each name; crtab;
                                            show: '   Dean: ';
                                            show: each dean personName; cr.
                                            n ← n + 1].

Transcript cr;
            show: '----------------------------------------------------------------------'; cr; cr;
            show: 'Total college deans: '; show: n printString; endEntry]

ifTrue:[                                    "if there are no colleges in this university"
    Transcript show: 'No Colleges in this University !!'; endEntry].
```

divisionDeans

"displays names of division deans of all divisions in this university on system transcript window else if a university has no divisions, a message is displayed "

```
| nl
n ← 0.
Transcript refresh; clear; cr;
            next: 15 put: $ ;
            show: 'Names of division deans of each division in UCLA' asUppercase; cr;
            next: 25 put: $ ;
            show: 'University Persident: ';
            show: self president personName; cr;
            show: '----------------------------------------------------------------------'; cr;cr.

((self divisions) = nil)
ifFalse:[                                   "if there are divisions in this university"
self divisions setof do:[:each | Transcript  show: 'Division name: ';
                                            show: each name; crtab;
                                            show: '     Dean: ';
                                            show: each dean personName; cr.
                                            n ← n + 1].

Transcript cr;
            show: '----------------------------------------------------------------------'; cr; cr;
            show: 'Total division deans: '; show: n printString; endEntry]

ifTrue:[                                    "if there are no divisions in this university"
        Transcript show: 'No Divisions in this University !!'; endEntry].
```

schoolDeans
"displays names of school deans of all schools in this university on system transcript window else if a university has no schools, a message is displayed"

```
| n|
n ← 0.
Transcript refresh; clear; cr;
            next: 15 put: $ ;
            show: 'Names of school deans of each division in UCLA' asUppercase; cr;
           next: 25 put: $ ;
           show: 'University Persident: ';
           show: self president personName; cr;
           show: '------------------------------------------------------------------';cr;cr.

((self schools ) = nil )
ifFalse:[                                          "if there are schools in this university"
self schools setof do:[:each | Transcript  show: 'School name: ';
                                            show: each name; crtab;
                                            show: '    Dean: ';
                                            show: each dean personName; cr.
                                            n ← n + 1].

Transcript cr;
           show: '------------------------------------------------------------------'; cr; cr;
           show: 'Total school deans: '; show: n printString; endEntry]

ifTrue:[                                          "if there are no schools in this university"
       Transcript show: 'No Schools in this university !!'; endEntry].
```

CLASS CollegeType

chairPersons
"returns a set of department chair persons (as a EmployeesType) this college has"

```
| aSet |
aSet ← Employees new.
aSet purpose: 'Set of department chair persons a college has'.
self departments setof do:[:each |
                aSet setof add: each chairPerson].
```

aSet numEmployees: aSet setof size.

numOfDepartments
"returns a number of departments (as an integer) this college has"

↑self departments numDepts.
↑aSet.

CLASS WISE IMPLEMENTATION OF METHODS FOR CLASSES PARTICIPATING IN TRANSCRIPT DATABASE

The following Smalltalk code is in the file **/OOUDB/impl/classInterface/trnsDBif**

salil Kulkarni
NJIT, Newark, NJ.
December 1990

CLASS StudentsType

getCurrentSectionsFor: key
"*A query to displays sections currently registered for by a student who's ID = key, or a error message if id is not valid, on the system transcript window*"

```
    | id |
    id ← 0.

    ((self isValid: key) = 'true')
  ifTrue: [
        (key isInteger)
        ifFalse:[id ← (self getId: key) ]          "key is a student name"
        ifTrue:[id ← key].                         "key is a student ID"

        self setof do:[:each |
                        ((each studentId) = id)
                        ifTrue: [each transcript cmtSections]] ]

    ifFalse: [ 1 to: 3 do: [:i |Transcript refresh; clear;
              show: 'getCurr...: Invalid key used !!'; endEntry] ].
```

getNumOfStudents
"*A query to displays total number of students in the database on system transcript window*"

```
    Transcript refresh; clear; cr;cr;
                show: 'Total number of Students  in  the  database = ' asUppercase;
```

```
            show: self numStudents printString; endEntry.
```

getStudents
"*A query to displays names and IDs of all students in the database on system transcript window*"

```
    | n |
    n ← 0.
    Transcript refresh; clear; cr;
                next: 20 put: $ ;
                show: 'Students in the database' asUppercase; cr; cr; cr;
                show: '------------------------------------------------------------------------------';crtab;
                show: '    student name'; next: 15 put: $ ; show: 'student ID'; cr;
                show: '------------------------------------------------------------------------------';cr.

    self setof do:[:each |
                        Transcript show: (n + 1) printString; tab;
                                show: each personName;
                                next: (33 - each personName size)
                                put: $ ;
                                show: each studentId printString; cr.
                                n ← n + 1].

    Transcript show: '------------------------------------------------------------------------------';cr; cr;
            show: 'Total students : '; show: n printString; endEntry.
```

getStudentsIfGpa: g
"*A query to displays names and IDs of all students who's GPA >= g on system transcript window*"

```
    | n |
    n ← 0.
    Transcript refresh; clear; cr;
                next: 20 put: $ ;
                show: 'Students who''s GPA is >= ' asUppercase;
                show: g printString; cr; cr; cr;
                show:
'------------------------------------------------------------------------------';crtab;
                show: '    student name'; next: 10 put: $ ; show: 'student ID'; tab;
tab; tab;
                show: 'gpa'; cr;
```

```
                show: '----------------------------------------------------------------------------';cr.

    self setof do:[:each |
                        (each gpa isZero) ifFalse:[
                            ((each gpa) >= g) ifTrue:[
                            Transcript show: (n + 1) printString; tab;
                                        show: each personName;
                                        next: (30 - each personName size)
                                        put: $ ;
                                        show: each studentId printString; tab; tab; tab;
                                        show: each gpa printString; cr.
                                        n ← n + 1]] ].
    (n isZero)
    ifFalse:[
    Transcript  show: '----------------------------------------------------------------------------';cr;cr;
                show: 'Total students : '; show: n printString; endEntry]
    ifTrue:[Transcript cr; show: 'None '; endEntry].
```

getTranscriptFor: key

"*A query to displays transcript of a student who's ID = key, or an error message if key is not valid, on the system transcript window*"

```
    | id |
    id ← 0.

    ((self isValid: key) = 'true')
   ifTrue: [ (key isInteger)
                ifFalse:[id ← (self getId: key) ]
                ifTrue:[id ← key].

        self setof do:[:each |
                            ((each studentId) = id)
                            ifTrue: [each transcript courseListing] ] ]

    ifFalse: [ 1 to: 3 do: [:i |Transcript refresh; clear;
                show: 'getTran...: Invalid key used !!';
endEntry] ].
```

getId: n

"*returns studentId of the student who's name is n, else returns false if name is not a valid*

one."

```
    self setof do:[:each |
        ((each firstLastName ) sameAs: n )
        ifTrue:[↑each studentId]].
    ↑'false'.
```

getName: id
"returns name(first and last) of the student who's studentId = id, else returns false if id is not a valid one."

```
    self setof do:[:each |
        ((each studentId ) = id )
        ifTrue:[↑each firstLastName]].
    ↑'false'.
```

isValid: key
"returns true if key (name or studentId) is valid, else returns false"

```
    (key isInteger)
    ifFalse:[ ((self getId: key) = 'false')
                ifTrue:[↑'false']]
    ifTrue:[ ((self getName: key) = 'false')
                ifTrue:[↑'false']].
    ↑'true'.
```

CLASS TRanscriptType

credits
"returns total credits earned as a integer by owner of this transcript"

```
    | sum |
    sum ← 0.
    (self courseRecords setof) do:[:each | sum ← sum + (each creditPoints) ].
    ↑sum.
```

qualityPoints
"returns total quality points earned as a float by owner of this transcript"

```
| sum |
sum ← 0.
(self courseRecords setof) do:[:each | sum ← sum + (each qualityPoints) ].
↑sum.
```

gpa
"returns gpa as a float of owner of this transcript"

```
((self credits) = 0)
ifTrue: [↑0]
ifFalse: [↑ (self qualityPoints) / (self credits)].
```

courseListing
"display formatted details of courses finished by student in system Transcipt window"

```
| n tmp |
n ← 0.
tmp ← self student perData name.
Transcript clear;refresh;cr; next: 24 put: $ ;
show: ('Transcript of a student ') asUppercase; cr;cr;
show: 'Student''s name: '; show: tmp first, ' ', tmp middle, ' ', tmp last;cr;
show: '              id: '; show: (self student studentId)printString;cr;cr.

Transcript show: 'Total courses taken: ';
            show: (self courseRecords numCourseRecords)printString;
            cr;cr;cr.


Transcript show: '-------------------------------------------------------------------';cr;
            show: '  course no.'; tab;
            show: '    course name'; next: 20 put: $ ; show: 'semester taken  ';
            show: '         grade';cr;
            show: '-------------------------------------------------------------------';cr.

(self courseRecords setof)
      do:[:each |  Transcript show: (n + 1)printString; tab;
                   show: (each course number); tab;
                   show: (each course name);
                   next: (40 - (each course name size)) put: $ ;
```

```
                    show: (each semesterTaken);
                    next: (17 - (each semesterTaken size)) put: $ ;
                    show: (each grade)printString;cr.
                    n ← n + 1].

    Transcript show: '------------------------------------------------------------------';cr;cr;
                show: 'Quality points earned: '; show: (self qualityPoints)printString;cr;
                show: 'Credits earned      : '; show: (self credits)printString;cr;cr;
                show: 'Cumulative GPA      : '; show: (self gpa)printString;
            endEntry.
```

crntSections

"display formatted details of courses currently registered by student on system Transcript window"

```
    | n tmp |
    n ← 0.
    tmp ← self student perData name.
    Transcript clear;refresh;cr; next: 17 put: $ ;
    show: ('Courses curently registered by a student ') asUppercase; cr;cr;
    show: 'Student''s name: '; show: tmp first, '*', tmp middle, ' ', tmp last;cr;
    show: '              id: '; show: (self student studentId)printString;cr;cr.

    Transcript show: 'Total courses being taken: ';
                show: (self currentSections numSections)printString;
                cr;cr;cr.

    Transcript show: '------------------------------------------------------------------';cr;
                show: '  section no.';
                next: 16 put: $ ;
                show: 'instructor name '; cr;
                show: '------------------------------------------------------------------';cr.

    (self currentSections setof)
            do:[:each | Transcript show: (n + 1) printString; tab;
                        show: each sectionNo;
                        next: 10 put: $ .
                        ((each instructor) = nil)
                        ifFalse:[ Transcript show: (each instructor personName); cr]
                        ifTrue:[ Transcript show: (self student supervisor
personName); show: ' (guide)'; cr].
                        n ← n + 1].

    Transcript endEntry.
```

CLASS StudentType

credits
"*returns credits finished by a student as a integer*"

```
    ↑self transcript credits.
```

gpa
"*returns gpa of a student as a float*"

```
    ↑self transcript gpa.
```

showTranscript
"displays transcript of a student on system transcript window"

```
    self transcript courseListing.
```

CLASS PersonType

firstLastName
"*returns first and last name of person as a string*"

```
    | tmp |
    tmp ← self perData name.
    ↑tmp first, ' ', tmp last.
```

personName
"*returns full name of person as a string*"

```
    | tmp |
```

```
    tmp ← self perData name.
    ↑tmp first, ' ', tmp middle, ' ', tmp last, ' ', tmp extra.
```

CLASS CourseRecordsType

student
"*returns a Student as a StudentType who is owner of transcript which has this set of course records*"

```
    ↑self transcript student.
```

CLASS CourseRecordType

creditPoints
"*returns credit points of a courseRecord as a integer*"

```
    ↑self course creditHours
```

qualityPoints
"*returns quality points of a courseRecord as a float*"

```
    ↑(self grade) * (self creditPoints)
```

CLASS WISE IMPLEMENTATION OF METHODS FOR CLASSES PARTICIPATING IN RESUME DATABASE

The following Smalltalk code is in the file **/OOUDB/impl/classInterface/resDBif**

salil Kulkarni
NJIT, Newark, NJ.
December 1990

CLASS EmployeesType

getAllEmployees
"*A query to displays names and emplyoeeNr of all employees in the database on system transcript window*"

```
    | n |
    n ← 0.
    Transcript refresh; clear; cr;
                next: 24 put: $ ;
                show: 'Employees in the database' asUppercase; cr; cr; cr;
                show:
'-------------------------------------------------------------------------------';crtab;
                show: '   employee name'; next: 15 put: $ ; show: 'employeeNr';
                next: 10 put: $ ;
               show: 'employee type'; cr; show:
'-------------------------------------------------------------------------------';cr.

    self setof do:[:each |
                        Transcript show: (n + 1) printString; tab;
                                show: each personName;
                                next: (33 - each personName size)
                                put: $ ;
                                show: each employeeNr printString;
                                next: 13 put: $ ;
                                show: each resume jobTitle; cr.
                                n ← n + 1].

    Transcript cr; show: '-------------------------------------------------------------------------------';
cr; cr;
```

```
show: 'Total employees : '; show: n printString; endEntry.
```

getConsultingsFor: key
"A query to displays names of consultancies done by a member of the receiver of name or employee number = key on system transcript window"

```
| id |

((self isValid: key) = 'true')
ifTrue: [
    (key isInteger)
    ifFalse:[id ← (self getId: key) ]          "key is an employee name"
    ifTrue:[id ← key].                         "key is an employee number"

    self setof do: [:each |
        ((each employeeNr) = id)
        ifTrue:[each resume consultComps] ] ]

ifFalse: [ 1 to: 3 do: [:i |Transcript refresh; clear;
          show: 'getConsult...: Invalid key used !!'; endEntry] ].
```

getFormalEducationsOfType: code For: key
"A query to displays certain formal educations depending on code received by a member of the receiver of name or employee number = key on system transcript window"

```
| id |

( ((self isValid: key) = 'true') & ((self isValidDeg: code) = 'true') )
ifTrue: [
    (key isInteger)
    ifFalse:[id ← (self getId: key) ]          "key is an employee name"
    ifTrue:[id ← key].                         "key is an employee number"

    self setof do: [:each |
        ((each employeeNr) = id)
        ifTrue:[each resume forEds: code] ] ]


ifFalse: [ ((self isValid: key) = 'false')
           ifTrue:[
           1 to: 3 do: [:i |Transcript refresh; clear;
```

```
            show: 'For: Invalid KEY used !!'; endEntry] ]
              ifFalse:[
              1 to: 3 do: [:i |Transcript refresh; clear;
            show: 'getForEdu...: Invalid CODE used !!'; endEntry] ] ].
```

getHonorsFor: key

"A query to displays honors received by a member of the receiver of name or employee number = key on system transcript window"

```
    | id |

    ((self isValid: key) = 'true')
  ifTrue: [
        (key isInteger)
        ifFalse:[id ← (self getId: key) ]          "key is an employee name"
        ifTrue:[id ← key].                         "key is an employee number"

        self setof do: [:each |
            ((each employeeNr) = id)
            ifTrue:[each resume hnrs] ] ]

    ifFalse: [ 1 to: 3 do: [:i |Transcript refresh; clear;
              show: 'getHonors...: Invalid key used !!'; endEntry] ].
```

getPublicationsOfType: code For: key

"A query to display details of certain publications depending on code, published by a member of the receiver of name or employee number = key on system transcript window"

```
    | id |

    ( ((self isValid: key) = 'true') & ((self isValidPub: code) = 'true') )
  ifTrue: [
        (key isInteger)
        ifFalse:[id ← (self getId: key) ]          "key is an employee name"
        ifTrue:[id ← key].                         "key is an employee number"

        self setof do: [:each |
            ((each employeeNr) = id)
            ifTrue:[each resume pubs: code] ] ]
```

```
    ifFalse: [ ((self isValid: key) = 'false')
                ifTrue:[
                1 to: 3 do: [:i |Transcript refresh; clear;
               show: 'For: Invalid KEY used !!'; endEntry] ]
                ifFalse:[
                1 to: 3 do: [:i |Transcript refresh; clear;
               show: 'getPubl...: Invalid CODE used !!'; endEntry] ] ].
```

isValidDeg: code
"*returns true if code (degree code) is valid, else returns false*"

```
( (code = 'B') | (code = 'M') | (code = 'P') | (code = 'ALL') )
ifFalse:[↑'false']
ifTrue:[↑'true'].
```

isValidPub: code
"*returns true if code (publication code) is valid, else returns false*"

```
( (code = 'RCP') | (code = 'RJP') | (code = 'B') | (code = 'A') || (code = 'ALL') )
ifFalse:[↑'false']
ifTrue:[↑'true'].
```

isValid: key
"*returns true if key (name or employeeNr) is valid, else returns false*"

```
    (key isInteger)
    ifFalse:[ ((self getId: key) = 'false')
                ifTrue:[↑'false']]
    ifTrue:[ ((self getName: key) = 'false')
                ifTrue:[↑'false']].
    ↑'true'.
```

getName: id
"*returns name(first and last) of a member of the receiver who's employeeNr = id, else returns*

false if id is not a valid one."

```
    self setof do:[:each |
        ((each employeeNr ) = id )
        ifTrue:[↑each firstLastName]].
    ↑'false'.
```

getId: n

"returns employeeNr of a member of the receiver who's name is n, else returns false if name is not a valid one."

```
    self setof do:[:each |
        ((each firstLastName ) sameAs: n )
        ifTrue:[↑each employeeNr]].
    ↑'false'.
```

CLASS ResumeType

hnrs

"display honors held by the receiver on system transcript window"

```
    | n |
    n ← 0.

    ( (self honor) = nil)
    ifFalse:[

    Transcript clear;refresh;cr; next: 5 put: $ ;
    show: ('List of honors received by ') asUppercase;
    show: self belongsTo personName asUppercase; cr; cr;
    show: 'Employee number: '; show: self belongsTo employeeNr printString;cr;
    show: 'Employee type   : '; show: self jobTitle; cr; cr;
    show: '-----------------------------------------------------------------';cr; cr.

    self honor setof
            do:[:each |
                    Transcript show: (n + 1) printString; tab;
                            show: each honor; cr.
```

```
                        n ← n + 1].

    Transcript cr; show: '----------------------------------------------------------------';cr;cr;
                show: 'Total honors: '; show: n printString; endEntry]

    ifTrue:[
        Transcript refresh; clear; show: ' No honors received by ';
                    show: self belongsTo personName; endEntry].
```

consultComps

"display names of consulting companies of the receiver on system transcript window"

```
    | n |
    n ← 0.

    ((self experience) = nil)
    ifFalse:[

    Transcript clear;refresh;cr; next: 4 put: $ ;
    show: ('List of consulting positions held by ') asUppercase;
    show: self belongsTo  personName asUppercase; cr; cr;
    show: 'Employee number: '; show: self belongsTo employeeNr printString;cr;
    show: 'Employee type   : '; show: self jobTitle; cr; cr;
    show: '----------------------------------------------------------------';cr; cr.

    self experience setof
            do:[:each |
                    ( (each type) = ConsultingType)
                    ifTrue:[ Transcript show: (n + 1) printString; tab;
                                        show: each title; space;
                                        show: 'for ';
                                        show: each employer; cr.
                                n ← n + 1] ].
    (n isZero)
    ifTrue:[Transcript show: 'No consultings done '; cr; cr; endEntry]
    ifFalse:[
    Transcript cr; show: '----------------------------------------------------------------';cr;cr;
                show: 'Total consultings: '; show: n printString; endEntry]]

    ifTrue:[
        Transcript refresh; clear; show: ' No experience ';endEntry].
```

pubs: key

```
"display details of certain publications depending on key, published by receiver on system
 transcript window"

    ( (self publications) = nil)
    ifFalse:[

        (key = 'RCP')                    "RCP = Refereed Conference Papers"
        ifTrue:[ self RCpapers].

        (key = 'RJP')                    "RJP = Refereed Journal Papers"
        ifTrue:[ self RJpapers].

        (key = 'B')                      "B = Books"
        ifTrue:[ self allBooks].

        (key = 'A')                      "A = Articles"
        ifTrue:[ self allArticles].

        (key = 'ALL')                    "ALL = show all publications. Actually, some kind"
        ifTrue: [ self RCpapers.         "      of user control is required to go from one"
                  self RJpapers.         "      publication to the next"
                  self allBooks.
                  self allArticles] ]

    ifTrue: [Transcript refresh; clear; show: 'No Publications by ';
                  show: self belongsTo personName;
                  endEntry].


allRCpapers
"displays Refereed Conference Papers published by receiver on system transcript window"

    | n |
    n ← 0.

    Transcript clear;refresh;cr;
    next: 17 put: $ ;
    show: ('List of refereed conference papers published by ') asUppercase; cr;
    next: 35 put: $ ;
    show: self belongsTo personName asUppercase; cr; cr;
    show: 'Employee number: '; show: self belongsTo employeeNr printString;cr;
    show: 'Employee type   : '; show: self jobTitle; cr; cr;
    show: '------------------------------------------------------------------';cr; cr.

    ((self publications conferences) = nil)
    ifFalse:[
    self publications conferences setof
```

```
        do:[:each |
                Transcript show: (n + 1) printString; tab;
                        show: 'Authors    : '; show: each authors; crtab;
                        show: 'Title      : '; show: each title; crtab;
                        show: 'Conference: '; show: each conference;
                        show:', '; crtab; next: 15 put: $ ;
                        show: each year; cr;cr.
                        n ← n + 1 ].

    Transcript cr; show: '----------------------------------------------------------------------';cr;cr;
        show: 'Total refereed conference papers: '; show: n printString; endEntry]

    ifTrue:[Transcript show: 'No refereed conference papers '; endEntry].
```

allRJpapers

"*displays Refereed Journal Papers published by receiver on system transcript window*"

```
    | n |
    n ← 0.

    Transcript clear;refresh;cr;
    next: 17 put: $ ;
    show: ('List of refereed journal papers published by ') asUppercase; cr;
    next: 35 put: $ ;
    show: self belongsTo personName asUppercase; cr; cr;
    show: 'Employee number: '; show: self belongsTo employeeNr printString;cr;
    show: 'Employee type   : '; show: self jobTitle; cr; cr;
    show: '----------------------------------------------------------------------';cr; cr.

    n ← 0.

    ((self publications journals) = nil)
    ifFalse:[
    self publications journals setof
            do:[:each |
                    Transcript show: (n + 1) printString; tab;
                            show: 'Authors    : '; show: each authors; crtab;
                            show: 'Title      : '; show: each title; crtab;
                            show: 'Journal    : '; show: each journal;
    show:', '; crtab; next: 12 put: $ ;
                            show: each year; cr;cr.
                            n ← n + 1 ].

    Transcript cr; show: '----------------------------------------------------------------------';cr;cr;
```

```
        show: 'Total refereed journal papers: '; show: n printString; endEntry]

    ifTrue:[Transcript show: 'No refereed journal papers '; endEntry].
```

allBooks

"*displays the Books published by receiver on system transcript window*"

```
    | n |
    n ← 0.

    Transcript clear;refresh;cr;
    next: 28 put: $ ;
    show: ('List of books published by ') asUppercase; cr;
    next: 32 put: $ ;
    show: self belongsTo personName asUppercase; cr; cr;
    show: 'Employee number: '; show: self belongsTo employeeNr printString;cr;
    show: 'Employee type   : '; show: self jobTitle; cr; cr;
    show: '------------------------------------------------------------------------';cr; cr.

    ((self publications books) = nil)
    ifFalse:[
    self publications books setof
            do:[:each |
                    Transcript show: (n + 1) printString; tab;
                            show: 'Authors    : '; show: each authors; crtab;
                            show: 'Title      : '; show: each title; crtab;
                            show: 'Editor     : '; show: each editor; crtab;
                            show: 'Publisher  : '; show: each publisher;
crtab;
                            show: 'Total pages : '; show: each pageNum; crtab;
                            show: each year; cr;cr.
                            n ← n + 1 ].

    Transcript cr; show: '------------------------------------------------------------------------';cr;cr;
        show: 'Total Books: '; show: n printString; endEntry]

    ifTrue:[Transcript show: 'No Books '; endEntry].
```

allArticles

```
"displays the Non Refereed Articles published by receiver on system transcript window"

    | n |
    n ← 0.

    Transcript clear;refresh;cr;
    next: 17 put: $ ;
    show: ('List of Non Refereed Articles published by ') asUppercase; cr;
    next: 35 put: $ ;
    show: self belongsTo personName asUppercase; cr; cr;
    show: 'Employee number: '; show: self belongsTo employeeNr printString;cr;
    show: 'Employee type  : '; show: self jobTitle; cr; cr;
    show: '------------------------------------------------------------------------';cr; cr.

    ((self publications articles) = nil)
    ifFalse:[
    self publications articles setof
            do:[:each |
                    Transcript show: (n + 1) printString; tab;
                            show: 'Authors    : '; show: each authors; crtab;
                            show: 'Title      : '; show: each title; crtab;
                            show: 'Conference     : '; show: each conference; crtab;
                            show: 'Volume    : '; show: each volume;
crtab;
                            show: 'Page No.  : '; show: each pageNum; crtab;
                            show: each year; cr;cr.
                            n ← n + 1 ].

    Transcript cr; show: '------------------------------------------------------------------------';cr;cr;
        show: 'Total Articles: '; show: n printString; endEntry]

    ifTrue:[Transcript show: 'No Articles '; endEntry].



forEds: key
"display details of certain formal educations depending on key, undergone by receiver on
 system transcript window"

    ( (self formalEducation) = nil)
    ifFalse:[

        (key = 'B')                    "B = Bachelor Degree"
        ifTrue:[ self BSDeg].

        (key = 'M')                    "M = Master Degree"
```

```
        ifTrue:[ self MSDeg].

        (key = 'P')                    "P = PhD Degree"
        ifTrue:[ self PhDDeg].

        (key = 'ALL')                  "ALL = All Degrees Actually, some kind of user"
        ifTrue:[ self BSDeg.           "      control is required to go from one degree"
                 self MSDeg.           "      to the next"
                 self PhDDeg] ]


    ifTrue: [Transcript refresh; clear; show: 'No Formal Educations received                    by
';
                    show: self belongsTo personName;
                    endEntry].
```

BSDeg
"displays details of BS Degree of the receiver"

```
    | n |
    n ← 0.
    Transcript clear;refresh;cr;
    next: 7 put: $ ;
    show: ('Bachelor Degrees received by ') asUppercase;
    show: self belongsTo personName asUppercase; cr; cr;
    show: 'Employee number: '; show: self belongsTo employeeNr printString;cr;
    show: 'Employee type   : '; show: self jobTitle; cr; cr;
    show: '------------------------------------------------------------------';cr; cr.

Transcript show: 'Bachelor Degrees: '; cr; cr.

    self formalEducation setof
            do:[:each |
                    ( (each type) = BachelorDegreeType)
                    ifTrue:[Transcript show: (n+1)printString; tab;
                                show: 'area      : '; show: each area; cr;
                                next: 4 put: $ ;
                                show: 'year      : '; show: each yrGranted; cr;
                                next: 4 put: $ ;
                                show: 'University: '; show: each university; cr.
                                n ← n + 1 ]].
    (n isZero)
    ifTrue:[Transcript tab; show: 'None'; endEntry]
    ifFalse:[
```

```
    Transcript cr; show: '----------------------------------------------------------------------';cr;cr;
                show: 'Total Bachelor Degrees: '; show: n printString; endEntry].
```

MSDeg

"Display details of MS Degree of receiver"

```
    | n |
    n ← 0.
    Transcript clear;refresh;cr;
    next: 7 put: $ ;
    show: ('Master Degrees received by ') asUppercase;
    show: self belongsTo personName asUppercase; cr; cr;
    show: 'Employee number: '; show: self belongsTo employeeNr printString;cr;
    show: 'Employee type   : '; show: self jobTitle; cr; cr;
    show: '----------------------------------------------------------------------';cr; cr.

Transcript show: 'Master Degrees: '; cr; cr.

    self formalEducation setof
          do:[:each |
                ( (each type) = MasterDegreeType)
                ifTrue:[Transcript show: (n+1)printString; tab;
                          show: 'area      : '; show: each area; cr;
                          next: 4 put: $ ;
                          show: 'year      : '; show: each yrGranted; cr;
                          next: 4 put: $ ;
                          show: 'University: '; show: each university; cr.
                          n ← n + 1 ]].
    (n isZero)
    ifTrue:[Transcript tab; show: 'None'; endEntry]
    ifFalse:[
    Transcript cr; show: '----------------------------------------------------------------------';cr;cr;
                show: 'Total Master Degrees: '; show: n printString; endEntry].
```

PhDDeg

"Display details of Phd Degree of receiver"

```
    | n |
    n ← 0.
    Transcript clear;refresh;cr;
    next: 7 put: $ ;
```

```
    show: ('PhD Degrees received by ') asUppercase;
    show: self belongsTo personName asUppercase; cr; cr;
    show: 'Employee number: '; show: self belongsTo employeeNr printString;cr;
    show: 'Employee type   : '; show: self jobTitle; cr; cr;
    show: '------------------------------------------------------------------';cr; cr.

Transcript show: 'PhD Degrees: '; cr; cr.

    self formalEducation setof
            do:[:each |
                    ( (each type) = PhDDegreeType)
                    ifTrue:[Transcript show: (n+1)printString; tab;
                            show: 'area      : '; show: each area; cr;
                            next: 4 put: $ ;
                            show: 'year      : '; show: each yrGranted; cr;
                            next: 4 put: $ ;
                            show: 'University: '; show: each university; cr.
                            n ← n + 1 ]].
    (n isZero)
    ifTrue:[Transcript tab; show: 'None'; endEntry]
    ifFalse:[
    Transcript cr; show: '------------------------------------------------------------------';cr;cr;
                show: 'Total PhD Degrees: '; show: n printString; endEntry].
```

CLASS WISE IMPLEMENTATION OF METHODS FOR CLASSES PARTICIPATING IN COMMITTEES DATABASE.

The following Smalltalk code is in the file **/OOUDB/impl/classInterface/commDBif**

salil Kulkarni
NJIT, Newark, NJ.
December 1990

CLASS CommitteesType

getCommitteeMembersOf: comCode
"*A query to displays names and employee numbers of all members of a commitee having code = comCode*"

```
(self isValidCom: comCode)
ifTrue:[
    self setof do:[:each |
            ((each code) sameAs: comCode)
            ifTrue:[ each getMembers ] ] ]
ifFalse:[
    Transcript refresh; clear;
        show: ' getComm...: Invalid committee CODE used !!'; endEntry]
```

getAllCommittees: Ename
"*A query to get names of all committees in which Ename is a member*"

```
| flag n aEmp |
flag ← 'f'.
n ← 0.

Transcript refresh; clear; cr;
    next: 10 put: $ ;
    show: 'List of committees in which ' asUppercase;
    show: Ename asUppercase; show: ' is a member' asUppercase; cr;cr;
    show: '--------------------------------------------------------------------------------'; cr; cr.

self setof do:[:each |
    each members setof do: [:ele |
```

```
            ( (ele firstLastName) sameAs: Ename)
             ifTrue:[ flag ← 't'.
                      aEmp ← ele] ].
        (flag = 't')
        ifTrue:[ Transcript show: (n + 1)printString; tab;
                    show: each name; cr.
                    flag ← 'f'.
                    n ← n  + 1] ].

    (n isZero)
    ifFalse:[ Transcript show:
'------------------------------------------------------------------------------'; cr; cr;
            show: Ename; show: '''s position: '; show: aEmp position; cr;
            show: 'Total committees: '; show: n printString; endEntry]
    ifTrue:[ Transcript tab; show: 'None'; endEntry]
```

isValidCom: code

"*Returns true if the code is a valid committee code, else returns false*"

```
    ↑ ( (code = 'B') | (code = 'SAB') | (code = 'DP') | (code = 'G')
      | (code = 'R') | (code = 'DE') | (code = 'DPT') )
```

CLASS CommitteeType

getMembers

"*Displays names and employee numbers of all members of receiver*"

```
    | n |
    n ← 0.
    Transcript refresh; clear; cr;
        show: 'Members of ' asUppercase; show: self name asUppercase;
        cr; cr; show: 'Chair Person: '; show: self chair personName;
        show: '( '; show: self chair position; show: ' )';cr;
        show: '------------------------------------------------------------'; crtab;
        show: 'member names '; next: 12 put: $ ;
        show: 'employee number'; tab; tab;
        show: 'employee type'; cr;
        show: '------------------------------------------------------------'; cr; cr.
```

```
( (self members) = nil)
ifFalse:[
    self members setof do:[:each |
                    Transcript show: (n + 1)printString; tab;
                    show: each personName;
                    next: (30 - (each personName size)) put: $ ;
                    show: each employeeNr printString;
                    next: 14 put: $ ;
                    show: each position; cr.
                    n ← n + 1].
    (n isZero)
    ifFalse:[Transcript
    show: '------------------------------------------------------------------'; cr; cr;
    show: 'Total members: '; show: n printString; endEntry]
    ifTrue:[Transcript show: 'No members'; endEntry] ]

ifTrue:[ Transcript show: 'No members'; endEntry].
```

APPENDIX C

SOURCE CODE FOR CREATING THE DATABASE

"

SOURCE CODE FOR CREATING THE UNIVERSITY DATABASE
(academic units in a university)

This file contains Smalltalk code to create a database of the academic units in a university, like colleges and their departments, schools and divisions.. The following instances have been created:

univ . . . . . University (1)
sch1 to sch3 . . . . . .School (3)
div1 . . . . . Division (1)
Col1 and col2 . . . . . College (2)
dept1 to dept6 . . . . . Department (6)
office11 to 16 and
office21 to 23 . . . . . Office (9)
comp11 to 16 and
comp21 to 23 . . . . . .Computer (9)
lab11 to 16 and
lab21 to 23 . . . . . Laboratory (9)

This Smalltalk source code is in the file /OOUDB/impl/srcCodes/univDBsc
salil kulkarni
NJIT, New Jersey, USA.
Dec. 1990

---"

```
|
univ

univDivs
div1  d1Dean d1DeanPd

univSchs
sch1  s1Dean s1DeanPd
office21 s1Offices comp21 s1Comps lab21 s1Labs s1Budget

sch2  s2Dean s2DeanPd
office22 s2Offices comp22 s2Comps lab22 s2Labs s2Budget

sch3  s3Dean s3DeanPd
office23 s3Offices comp23 s3Comps lab23 s3Labs s3Budget
```

```
dept1  d1Chair d1ChairPd
office11 d1Offices  comp11  d1Comps  lab11 d1Labs  d1Budget

dept2 d2Chair  d2ChairPd
office12 d2Offices comp12  d2Comps lab12  d2Labs  d2Budget

dept3 d3Chair  d3ChairPd
office13 d3Offices comp13  d3Comps lab13  d3Labs  d3Budget

dept4 d4Chair  d4ChairPd
office14 d4Offices comp14  d4Comps lab14  d4Labs  d4Budget

dept5 d5Chair  d5ChairPd
office15 d5Offices comp15  d5Comps lab15  d5Labs  d5Budget

dept6 d6Chair  d6ChairPd
office16 d6Offices comp16  d6Comps lab16  d6Labs  d6Budget

univCols
col1 c1Dean c1DeanPd c1Depts

col2  c2Dean c2DeanPd c2Depts

|


"-------------------------------------------------------------------------------------------------------------------------"

univ ← University new.

univDivs ← Divisions new.
div1 ← Division new.
d1Dean ← DivisionDean new.
d1DeanPd ← PersonData new.

univSchs ← Schools new.
sch1 ← School new.
s1Dean ← SchoolDean new.
s1DeanPd ← PersonData new.
office21 ← Office new.
s1Offices ← Offices new.
comp21 ← Computer new.
s1Comps ← Computers new.
lab21 ← Laboratory new.
s1Labs ← Laboratories new.
s1Budget ← Budget new.
```

sch2 ← School new.
s2Dean ← SchoolDean new.
s2DeanPd ← PersonData new.
office22 ← Office new.
s2Offices ← Offices new.
comp22 ← Computer new.
s2Comps ← Computers new.
lab22 ← Laboratory new.
s2Labs ← Laboratories new.
s2Budget ← Budget new.

sch3 ← School new.
s3Dean ← SchoolDean new.
s3DeanPd ← PersonData new.
office23 ← Office new.
s3Offices ← Offices new.
comp23 ← Computer new.
s3Comps ← Computers new.
lab23 ← Laboratory new.
s3Labs ← Laboratories new.
s3Budget ← Budget new.

dept1 ← Department new.
d1Chair ← DeptChairPerson new.
d1ChairPd ← PersonData new.
office11 ← Office new.
d1Offices ← Offices new.
comp11 ← Computer new.
d1Comps ← Computers new.
lab11 ← Laboratory new.
d1Labs ← Laboratories new.
d1Budget ← Budget new.

dept2 ← Department new.
d2Chair ← DeptChairPerson new.
d2ChairPd ← PersonData new.
office12 ← Office new.
d2Offices ← Offices new.
comp12 ← Computer new.
d2Comps ← Computers new.
lab12 ← Laboratory new.
d2Labs ← Laboratories new.
d2Budget ← Budget new.

dept3 ← Department new.
d3Chair ← DeptChairPerson new.
d3ChairPd ← PersonData new.

office13 ← Office new.
d3Offices ← Offices new.
comp13 ← Computer new.
d3Comps ← Computers new.
lab13 ← Laboratory new.
d3Labs ← Laboratories new.
d3Budget ← Budget new.

dept4 ← Department new.
d4Chair ← DeptChairPerson new.
d4ChairPd ← PersonData new.
office14 ← Office new.
d4Offices ← Offices new.
comp14 ← Computer new.
d4Comps ← Computers new.
lab14 ← Laboratory new.
d4Labs ← Laboratories new.
d4Budget ← Budget new.

dept5 ← Department new.
d5Chair ← DeptChairPerson new.
d5ChairPd ← PersonData new.
office15 ← Office new.
d5Offices ← Offices new.
comp15 ← Computer new.
d5Comps ← Computers new.
lab15 ← Laboratory new.
d5Labs ← Laboratories new.
d5Budget ← Budget new.

dept6 ← Department new.
d6Chair ← DeptChairPerson new.
d6ChairPd ← PersonData new.
office16 ← Office new.
d6Offices ← Offices new.
comp16 ← Computer new.
d6Comps ← Computers new.
lab16 ← Laboratory new.
d6Labs ← Laboratories new.
d6Budget ← Budget new.

univCols ← Colleges new.
col1 ← College new.
c1Dean ← CollegeDean new.
c1DeanPd ← PersonData new.
c1Depts ← Departments new.

```
col2 ← College new.
c2Dean ← CollegeDean new.
c2DeanPd ← PersonData new.
c2Depts ← Departments new.

"-----------------------------------------------------------------------------------------------------------

DIVISIONS"

d1DeanPd name.first: 'Peter'; name.middle: 'Jake'; name.last: 'Appleton';
            sex: 'Male'; maritalStatus: 'Married';
            birthdate.day: 21; birthdate.month: 'Feb'; birthdate.year: 1939.

d1Dean perData: d1DeanPd;
        employeeNr: 1001;
        name: 'Dean of the Division of Law';
        inCharge: 'Division of Law'.

div1 name: 'Division of Law';
     dean: d1Dean.

"-----------------------------------------------------------------------------------------------------------
SCHOOLS: "

s1DeanPd name.first: 'Mike'; name.middle: 'Jack'; name.last: 'Gere';
            sex: 'Male'; maritalStatus: 'Married';
            birthdate.day: 15; birthdate.month: 'Aug'; birthdate.year: 1941.

s1Dean perData: s1DeanPd;
        employeeNr: 2001;
        name: 'Dean of School of Architecture'.

office21  facilityId: 'O-201';
            belongsTo: sch1.
s1Offices purpose: 'Offices in school of architecture'.
s1Offices setof add: office21.
s1Offices numOffices: (s1Offices setof) size.

comp21 facilityId: 'C-201';
            belongsTo: sch1.
s1Comps purpose: 'computers in school of architecture'.
s1Comps setof add: comp21.
s1Comps numComputers: (s1Comps setof) size.
```

```
lab21 facilityId: 'l-201';
        belongsTo: sch1.
s1Labs purpose: 'laboratories in school of architecture'.
s1Labs setof add: lab21.
s1Labs numlabs: (s1Labs setof) size.

s1Budget source: 'State Funds';
            amount: 80000.0;
            availabilityPeriod: '1989-1991'.

sch1  name: 'School of Architecture';
      dean: s1Dean;
      offices: s1Offices;
      computers: s1Comps;
      laboratories: s1Labs;
      budget: s1Budget.

            "-------------------------------------------"

s2DeanPd name.first: 'Edward'; name.middle: 'Gregory'; name.last: 'Haynes';
            sex: 'Male'; maritalStatus: 'Married';
            birthdate.day: 5; birthdate.month: 'Nov'; birthdate.year: 1943.

s2Dean perData: s2DeanPd;
        employeeNr: 2002;
        name: 'Dean of school of buisness management'.

office22  facilityId: 'O-202';
            belongsTo: sch2.
s2Offices purpose: 'Offices in school of buisness management'.
s2Offices setof add: office22.
s2Offices numOffices: (s2Offices setof) size.

comp22 facilityId: 'C-202';
            belongsTo: sch2.
s2Comps purpose: 'computers in school of buisness management'.
s2Comps setof add: comp22.
s2Comps numComputers: (s2Comps setof) size.

lab22 facilityId: 'l-202';
        belongsTo: sch2.
s2Labs purpose: 'laboratories in school of buisness management'.
s2Labs setof add: lab22.
s2Labs numlabs: (s2Labs setof) size.

s2Budget source: 'Guttenberg''s Fund';
            amount: 80000.0;
```

```
            availabilityPeriod: '1990-1991'.

sch2  name: 'School of Buisness Management';
      dean: s2Dean;
      offices: s2Offices;
      computers: s2Comps;
      laboratories: s2Labs;
      budget: s2Budget.

            "-------------------------------------------"

s3DeanPd name.first: 'Maurice'; name.middle: 'Haley'; name.last: 'Ricardo';
            sex: 'Male'; maritalStatus: 'Married';
            birthdate.day: 15; birthdate.month: 'Aug'; birthdate.year: 1937.

s3Dean perData: s3DeanPd;
        employeeNr: 2003;
        name: 'Dean of school of drama'.

office23  facilityId: 'O-203';
            belongsTo: sch3.
s3Offices purpose: 'Offices in school of drama'.
s3Offices setof add: office23.
s3Offices numOffices: (s3Offices setof) size.

comp23 facilityId: 'C-203';
            belongsTo: sch3.
s3Comps purpose: 'computers in school of drama'.
s3Comps setof add: comp23.
s3Comps numComputers: (s3Comps setof) size.

lab23 facilityId: 'l-203';
        belongsTo: sch3.
s3Labs purpose: 'laboratories in school of drama'.
s3Labs setof add: lab23.
s3Labs numlabs: (s3Labs setof) size.

s3Budget source: 'American Society for Drama ';
            amount: 10000.0;
            availabilityPeriod: '1990-1991'.

sch3  name: 'School of Drama';
      dean: s3Dean;
      offices: s3Offices;
      computers: s3Comps;
      laboratories: s3Labs;
      budget: s3Budget.
```

```
"-------------------------------------------------------------------------------------------------------------
DEPARTMENTS: "

d1ChairPd name.first: 'Larry'; name.middle: 'Gene'; name.last: 'Gomes';
            sex: 'Male'; maritalStatus: 'Single';
            birthdate.day: 22; birthdate.month: 'Oct'; birthdate.year: 1940.

d1Chair perData: d1ChairPd;
        employeeNr: 301;
        name: 'Chair person of CIS Department';
        inCharge: dept1.

office11  facilityId: '0-101';
            belongsTo: dept1.
d1Offices purpose: 'Offices in CIS department '.
d1Offices setof add: office11.
d1Offices numOffices: (d1Offices setof) size.

comp11 facilityId: 'C-101';
            belongsTo: dept1.
d1Comps purpose: 'Computers in CIS department'.
d1Comps setof add: comp11.
d1Comps numComputers: (d1Comps setof) size.

lab11 facilityId: 'L-101';
        belongsTo: dept1.
d1Labs purpose: 'Laboratories in CIS department'.
d1Labs setof add: lab11.
d1Labs numlabs: (d1Labs setof) size.

d1Budget source: 'IBM computers';
            amount: 300000.0;
            availabilityPeriod: '1989-1991'.

dept1  name: 'Department of Computer Science';
       offices: d1Offices;
       computers: d1Comps;
         laboratories: d1Labs;
         budget: d1Budget;
         college: col1;
         chairPerson: d1Chair.

    "-------------------------------------------"

d2ChairPd name.first: 'Kenny'; name.middle: 'Nicholson'; name.last: 'Hunts';
```

```
            sex: 'Male'; maritalStatus: 'Married';
            birthdate.day: 2; birthdate.month: 'May'; birthdate.year: 1937.

d2Chair perData: d2ChairPd;
          employeeNr: 302;
          name: 'Chair person of EE Department';
          inCharge: dept2.

office12  facilityId: '0-102';
            belongsTo: dept2.
d2Offices purpose: 'Offices in EE department '.
d2Offices setof add: office12.
d2Offices numOffices: (d2Offices setof) size.

comp12 facilityId: 'C-102';
            belongsTo: dept2.
d2Comps purpose: 'Computers in EE department'.
d2Comps setof add: comp12.
d2Comps numComputers: (d2Comps setof) size.

lab12 facilityId: 'L-102';
          belongsTo: dept2.
d2Labs purpose: 'Laboratories in EE department'.
d2Labs setof add: lab12.
d2Labs numlabs: (d2Labs setof) size.

d2Budget source: 'Digital Engineering Corporation';
            amount: 100000.0;
            availabilityPeriod: '1989-1991'.

dept2  name: 'Department of Electrical engineering';
        offices: d2Offices;
        computers: d2Comps;
         laboratories: d2Labs;
         budget: d2Budget;
         college: col1;
         chairPerson: d2Chair.

            "-----------------------------------"

d3ChairPd name.first: 'Barbara'; name.middle: 'Ronald'; name.last: 'Baise';
            sex: 'Female'; maritalStatus: 'Married';
            birthdate.day: 21; birthdate.month: 'Jan'; birthdate.year: 1939.

d3Chair perData: d3ChairPd;
          employeeNr: 303;
          name: 'Chair person of ME Department';
```

```
        inCharge: dept3.

office13  facilityId: '0-103';
            belongsTo: dept3.
d3Offices purpose: 'Offices in ME department '.
d3Offices setof add: office13.
d3Offices numOffices: (d3Offices setof) size.

comp13 facilityId: 'C-103';
            belongsTo: dept3.
d3Comps purpose: 'Computers in ME department'.
d3Comps setof add: comp13.
d3Comps numComputers: (d3Comps setof) size.

lab13 facilityId: 'L-103';
        belongsTo: dept3.
d3Labs purpose: 'Laboratories in ME department'.
d3Labs setof add: lab13.
d3Labs numlabs: (d3Labs setof) size.

d3Budget source: 'McDonald Doughlas Ltd.';
            amount: 90000.0;
            availabilityPeriod: '1990-1991'.

dept3  name: 'Department of Mechanical engineering';
       offices: d3Offices;
       computers: d3Comps;
        laboratories: d3Labs;
        budget: d3Budget;
        college: col1;
        chairPerson: d3Chair.

            "-------------------------------------"

d4ChairPd name.first: 'Jack'; name.middle: 'Pedro'; name.last: 'Fenster';
            sex: 'Male'; maritalStatus: 'Married';
            birthdate.day: 13; birthdate.month: 'Sep'; birthdate.year: 1938.

d4Chair perData: d4ChairPd;
        employeeNr: 304;
        name: 'Chair person of Department of Fine Arts ';
        inCharge: dept4.

office14  facilityId: '0-104';
            belongsTo: dept4.
d4Offices purpose: 'Offices in the department of fine arts'.
d4Offices setof add: office14.
```

```
d4Offices numOffices: (d4Offices setof) size.

comp14 facilityId: 'C-104';
        belongsTo: dept4.
d4Comps purpose: 'Computers in the department of fine arts'.
d4Comps setof add: comp14.
d4Comps numComputers: (d4Comps setof) size.

lab14 facilityId: 'L-104';
      belongsTo: dept4.
d4Labs purpose: 'Laboratories in the department of fine arts'.
d4Labs setof add: lab14.
d4Labs numlabs: (d4Labs setof) size.

d4Budget source: 'Revlon Pvt. Ltd.';
        amount: 30000.0;
        availabilityPeriod: '1990-1991'.

dept4  name: 'Department of Fine Arts';
      offices: d4Offices;
      computers: d4Comps;
       laboratories: d4Labs;
       budget: d4Budget;
       college: col2;
       chairPerson: d4Chair.

        "-------------------------------------"

d5ChairPd name.first: 'Michelle'; name.middle: 'Jake'; name.last: 'Pfiffer';
        sex: 'Female'; maritalStatus: 'Single';
        birthdate.day: 1; birthdate.month: 'Mar'; birthdate.year: 1943.

d5Chair perData: d5ChairPd;
       employeeNr: 305;
       name: 'Chair person of Department ofApplied Physics ';
       inCharge: dept5.

office15  facilityId: '0-105';
        belongsTo: dept5.
d5Offices purpose: 'Offices in the department of applied physics'.
d5Offices setof add: office15.
d5Offices numOffices: (d5Offices setof) size.

comp15 facilityId: 'C-105';
        belongsTo: dept5.
d5Comps purpose: 'Computers in the department of applied physics '.
d5Comps setof add: comp15.
```

```
d5Comps numComputers: (d5Comps setof) size.

lab15 facilityId: 'L-105';
        belongsTo: dept5.
d5Labs purpose: 'Laboratories in the department of applied physics'.
d5Labs setof add: lab15.
d5Labs numlabs: (d5Labs setof) size.

d5Budget source: 'Central Apllied Physics LAboratory';
            amount: 60000.0;
            availabilityPeriod: '1990-1992'.

dept5  name: 'Department of Applied Physics';
       offices: d5Offices;
       computers: d5Comps;
         laboratories: d5Labs;
         budget: d5Budget;
         college: col2;
         chairPerson: d5Chair.

            "-------------------------------------".

d6ChairPd name.first: 'Janet'; name.middle: 'Piquick'; name.last: 'Sable';
            sex: 'Female'; maritalStatus: 'Widow';
            birthdate.day: 11; birthdate.month: 'Sep'; birthdate.year: 1935.

d6Chair perData: d6ChairPd;
         employeeNr: 306;
         name: 'Chair person of IE Department ';
         inCharge: dept6.

office16  facilityId: '0-106';
            belongsTo: dept6.
d6Offices purpose: 'Offices in the department'.
d6Offices setof add: office16.
d6Offices numOffices: (d6Offices setof) size.

comp16 facilityId: 'C-106';
            belongsTo: dept6.
d6Comps purpose: 'Computers in the IE department'.
d6Comps setof add: comp16.
d6Comps numComputers: (d6Comps setof) size.

lab16 facilityId: 'L-106';
        belongsTo: dept6.
d6Labs purpose: 'Laboratories in the IE department '.
d6Labs setof add: lab16.
```

```
d6Labs numlabs: (d6Labs setof) size.

d6Budget source: 'General Motors';
            amount: 600000.0;
            availabilityPeriod: '1990-1992'.

dept6  name: 'Department of Industrial Engineering';
       offices: d6Offices;
       computers: d6Comps;
        laboratories: d6Labs;
        budget: d6Budget;
        college: col1;
        chairPerson: d6Chair.


"-------------------------------------------------------------------------------------------------------------
COLLEGES: "

c1DeanPd name.first: 'James'; name.middle: 'Peter'; name.last: 'Nicholson';
            "sex: 'Male'; maritalStatus: 'Married'; "
            birthdate.day: 1; birthdate.month: 'May'; birthdate.year: 1940.

c1Dean perData: c1DeanPd;
        employeeNr: 3001;
        name: 'Dean of College of Engineering and Technology'.

c1Depts purpose: 'Departments in the College of Engineering and Technology'.
c1Depts setof add: dept1; add: dept2; add: dept3; add: dept6.
c1Depts numDepts: (c1Depts setof) size.

col1 name: 'College of Engineering and Technology';
     dean: c1Dean;
     departments: c1Depts.

"-----------------------------------------------------------"

c2DeanPd name.first: 'Fedrick'; name.middle: 'Mike'; name.last: 'Monroe';
            sex: 'Male'; maritalStatus: 'Divorced';
            birthdate.day: 11; birthdate.month: 'Dec'; birthdate.year: 1941.

c2Dean perData: c2DeanPd;
        employeeNr: 3002;
        name: 'Dean of College of Arts and Science'.

c2Depts purpose: 'Departments in the College of Arts and Science'.
c2Depts setof add: dept4; add: dept5.
c2Depts numDepts: (c2Depts setof) size.
```

```
col2 name: 'College of Arts and Science';
      dean: c2Dean;
      departments: c2Depts.

"-------------------------------------------------------------------------------------------------------------------------------"

univDivs purpose: 'Divisions in UCLA'.
univDivs setof add: div1.
univDivs numDivisions: (univDivs setof) size.

univSchs purpose: 'schools in university'.
univSchs setof add: sch1; add: sch2; add: sch3.
univSchs numSchools: (univSchs setof) size.

univCols   purpose: 'Colleges in UCLA'.
univCols setof add: col1; add: col2.
univCols numColleges: (univCols setof) size.

"-------------------------------------------------------------------------------------------------------------------------------"
univ    name: 'University of California at Los Angeles';
        divisions: univDivs;
        schools: univSchs;
        colleges: univCols.


"-------------------------------------------------------------------------------------------------------------------------------"

"Above database is saved as follows: "


univ makePersistent surrogate.

"got 200
 old one -> 196,146"


"-------------------------------------------------------------------------------------------------------------------------------"
```

"

SOURCE CODE FOR CREATING THE TRANSCRIPT DATABASE

This file contains Smalltalk code to create a database of students, their transcripts containing all courses they have finished, the courses they are taking currently, and professors. The following instances have been created:

stud1 to stud10 . . . . . MasterStudent (10)
assProf1 to assProf3 . . . . . AssiProfessor (3)
fullProf1 to fullProf2 . . . FullProfessor (2)
cors1 to cors10 . . . . . Course (10)
sec1 to sec15 . . . . . Section (15)
univStudents . . . . . Students(1)

This Smalltalk source code is in the file /OOUDB/impl/srcCodes/trnsDBsc
salil kulkarni
NJIT, New Jersey, USA.
Dec. 1990

---"

|
pd1 stud1 tr1 tr1CorsRecds tr1CrntSecs
pd2 stud2 tr2 tr2CorsRecds tr2CrntSecs
pd3 stud3 tr3 tr3CorsRecds tr3CrntSecs
pd4 stud4 tr4 tr4CorsRecds tr4CrntSecs
pd5 stud5 tr5 tr5CorsRecds tr5CrntSecs
pd6 stud6 tr6 tr6CorsRecds tr6CrntSecs
pd7 stud7 tr7 tr7CorsRecds tr7CrntSecs
pd8 stud8 tr8 tr8CorsRecds tr8CrntSecs
pd9 stud9 tr9 tr9CorsRecds tr9CrntSecs
pd10 stud10 tr10 tr10CorsRecds tr10CrntSecs

apPd1 assProf1
apPd2 assProf2
apPd3 assProf3
fpPd1 fullProf1
fpPd2 fullProf2

cors1 cors2 cors3 cors4 cors5 cors6 cors7 cors8 cors9 cors10
sec1 sec2 sec3 sec4 sec5 sec6 sec7 sec8 sec9 sec10 sec11 sec12 sec13 sec14 sec15

```
corsRecd1 corsRecd2 corsRecd3 corsRecd4 corsRecd5 corsRecd6 corsRecd7 corsRecd8
corsRecd9 corsRecd10 corsRecd11 corsRecd12 corsRecd13
univStudents |

"-------------------------------------------------------------------------------------------------------------------------"

pd1 ← PersonData new.
stud1 ← MasterStudent  new.
tr1 ← TRanscript new.
tr1CorsRecds ← CourseRecords new.
tr1CrntSecs ← Sections new.

pd2 ← PersonData new.
stud2 ← MasterStudent  new.
tr2 ← TRanscript new.
tr2CorsRecds ← CourseRecords new.
tr2CrntSecs ← Sections new.

pd3 ← PersonData new.
stud3 ← MasterStudent  new.
tr3 ← TRanscript new.
tr3CorsRecds ← CourseRecords new.
tr3CrntSecs ← Sections new.

pd4 ← PersonData new.
stud4 ← MasterStudent  new.
tr4 ← TRanscript new.
tr4CorsRecds ← CourseRecords new.
tr4CrntSecs ← Sections new.

pd5 ← PersonData new.
stud5 ← MasterStudent  new.
tr5 ← TRanscript new.
tr5CorsRecds ← CourseRecords new.
tr5CrntSecs ← Sections new.

pd6 ← PersonData new.
stud6 ← MasterStudent  new.
tr6 ← TRanscript new.
tr6CorsRecds ← CourseRecords new.
tr6CrntSecs ← Sections new.

pd7 ← PersonData new.
stud7 ← MasterStudent  new.
```

tr7 ← TRanscript new.
tr7CorsRecds ← CourseRecords new.
tr7CrntSecs ← Sections new.

pd8 ← PersonData new.
stud8 ← MasterStudent new.
tr8 ← TRanscript new.
tr8CorsRecds ← CourseRecords new.
tr8CrntSecs ← Sections new.

pd9 ← PersonData new.
stud9 ← MasterStudent new.
tr9 ← TRanscript new.
tr9CorsRecds ← CourseRecords new.
tr9CrntSecs ← Sections new.

pd10 ← PersonData new.
stud10 ← MasterStudent new.
tr10 ← TRanscript new.
tr10CorsRecds ← CourseRecords new.
tr10CrntSecs ← Sections new.

apPd1 ← PersonData new.
assProf1 ← AssiProfessor new.

apPd2 ← PersonData new.
assProf2 ← AssiProfessor new.

apPd3 ← PersonData new.
assProf3 ← AssiProfessor new.

fpPd1 ← PersonData new.
fullProf1 ← FullProfessor new.

fpPd2 ← PersonData new.
fullProf2 ← FullProfessor new.

cors1 ← Course new.
cors2 ← Course new.
cors3 ← Course new.
cors4 ← Course new.
cors5 ← Course new.
cors6 ← Course new.
cors7 ← Course new.

```
cors8 ← Course new.
cors9 ← Course new.
cors10 ← Course new.
sec1 ← Section new.
sec2 ← Section new.
sec3 ← Section new.
sec4 ← Section new.
sec5 ← Section new.
sec6 ← Section new.
sec7 ← Section new.
sec8 ← Section new.
sec9 ← Section new.
sec10 ← Section new.
sec11 ← Section new.
sec12 ← Section new.
sec13 ← Section new.
sec14 ← Section new.
sec15 ← Section new.

corsRecd1 ← CourseRecord new.
corsRecd2 ← CourseRecord new.
corsRecd3 ← CourseRecord new.
corsRecd4 ← CourseRecord new.
corsRecd5 ← CourseRecord new.
corsRecd6 ← CourseRecord new.
corsRecd7 ← CourseRecord new.
corsRecd8 ← CourseRecord new.
corsRecd9 ← CourseRecord new.
corsRecd10 ← CourseRecord new.
corsRecd11 ← CourseRecord new.
corsRecd12 ← CourseRecord new.
corsRecd13 ← CourseRecord new.



"-------------------------------------------------------------------------------------------------------------------"



pd1 name.first: 'Francis'; name.middle: 'Fedrick'; name.last: 'Wallace';
    sex: 'Male'; maritalStatus: 'Single';
    birthdate.day: 28; birthdate.month: 'Jan'; birthdate.year: 1965.

stud1  perData: pd1;
        studentId: 1000;
        project: 'unknown';
        thesis: 'unknown';
```

```
        transcript: tr1.

tr1 student: stud1; courseRecords: tr1CorsRecds; currentSections: tr1CrntSecs.

tr1CorsRecds purpose: 'Course Records for tr1'.
tr1CorsRecds setof add: corsRecd7; add: corsRecd9; add: corsRecd10;
add: corsRecd11.
tr1CorsRecds numCourseRecords: tr1CorsRecds setof size.
tr1CorsRecds  transcript: tr1.

tr1CrntSecs purpose: 'Current sections of tr1'.
tr1CrntSecs setof add: sec1; add: sec3; add: sec5; add: sec12.
tr1CrntSecs numSections: (tr1CrntSecs setof) size.


        "-------------------------------------------------------"


pd2 name.first: 'William'; name.middle: 'Francis'; name.last: 'Dodge'; name.extra: 'Bill';
    sex: 'Male'; maritalStatus: 'Single';
     birthdate.day: 8; birthdate.month: 'Jan'; birthdate.year: 1963.

stud2  perData: pd2;
        studentId: 2000;
        project: 'unknown';
        thesis: 'unknown';

        transcript: tr2.

tr2 student: stud2; courseRecords: tr2CorsRecds; currentSections: tr2CrntSecs.

tr2CorsRecds purpose: 'Course Records for tr2'.
tr2CorsRecds setof add: corsRecd5; add: corsRecd8; add: corsRecd9;                    add:
corsRecd10.
tr2CorsRecds numCourseRecords: tr2CorsRecds setof size.
tr2CorsRecds transcript: tr2.

tr2CrntSecs purpose: 'Current sections of tr2'.
tr2CrntSecs setof add: sec2; add: sec4; add: sec11; add: sec13.
tr2CrntSecs numSections: (tr2CrntSecs setof) size.

        "-------------------------------------------------------"

pd3 name.first: 'Kirk'; name.middle: 'Joe'; name.last: 'Christie';
    sex: 'Male'; maritalStatus: 'Single';
    birthdate.day: 8; birthdate.month: 'May'; birthdate.year: 1962.
```

```
stud3  perData: pd3;
        studentId: 3000;
        project: 'unknown';
        thesis: 'unknown';

        transcript: tr3.

tr3 student: stud3; courseRecords: tr3CorsRecds; currentSections: tr3CrntSecs.

tr3CorsRecds purpose: 'Course Records for tr3'.
tr3CorsRecds setof add: corsRecd3; add: corsRecd6; add: corsRecd7;                    add:
corsRecd9.
tr3CorsRecds numCourseRecords: tr3CorsRecds setof size.
tr3CorsRecds transcript: tr3.

tr3CrntSecs purpose: 'Current sections of tr3'.
tr3CrntSecs setof add: sec1; add: sec10; add: sec11; add: sec12.
tr3CrntSecs numSections: (tr3CrntSecs setof) size.

        "-------------------------------------------------------"

pd4 name.first: 'Peter'; name.middle: 'Joe'; name.last: 'OConnor';
    sex: 'Male'; maritalStatus: 'Single';
     birthdate.day: 1; birthdate.month: 'Sep'; birthdate.year: 1959.

stud4  perData: pd4;
        studentId: 4000;
        project: 'unknown';
        thesis: 'unknown';

        transcript: tr4.

tr4 student: stud4; courseRecords: tr4CorsRecds; currentSections: tr4CrntSecs.

tr4CorsRecds purpose: 'Course Records for tr3'.
tr4CorsRecds setof add: corsRecd1; add: corsRecd4; add: corsRecd5;                    add:
corsRecd8.
tr4CorsRecds numCourseRecords: tr4CorsRecds setof size.
tr4CorsRecds transcript: tr4.

tr4CrntSecs purpose: 'Current sections of tr4'.
tr4CrntSecs setof add: sec9; add: sec10; add: sec11; add: sec13.
tr4CrntSecs numSections: (tr4CrntSecs setof) size.

        "-------------------------------------------------------"
```

```
pd5 name.first: 'Pranav'; name.middle: 'Arun'; name.last: 'Patel';
    sex: 'Male'; maritalStatus: 'Single';
    birthdate.day: 28; birthdate.month: 'Jan'; birthdate.year: 1966.

stud5  perData: pd5;
        studentId: 5000;
        supervisor: assProf3;
        project: 'Simulation of a Token Ring LAN';
        thesis: 'not applicable';

        transcript: tr5.

tr5 student: stud5; courseRecords: tr5CorsRecds; currentSections: tr5CrntSecs.

tr5CorsRecds purpose: 'Course Records for tr5'.
tr5CorsRecds setof add: corsRecd2; add: corsRecd4; add: corsRecd6;
                    add: corsRecd8; add: corsRecd10; add: corsRecd11;                    add:
corsRecd12.
tr5CorsRecds numCourseRecords: (tr5CorsRecds setof) size.
tr5CorsRecds transcript: tr5.

tr5CrntSecs purpose: 'Current sections of tr5'.
tr5CrntSecs setof add: sec9; add: sec14.
tr5CrntSecs numSections: (tr5CrntSecs setof) size.

        "-------------------------------------------------------"

pd6 name.first: 'Nevil'; name.middle: 'Narendra'; name.last: 'Patel';
    sex: 'Male'; maritalStatus: 'Married';
    birthdate.day: 4; birthdate.month: 'Nov'; birthdate.year: 1965.

stud6  perData: pd6;
        studentId: 6000;
        supervisor: assProf2;
        project: 'Implementation of a University Database using C++';
        thesis: 'not applicable';

        transcript: tr6.

tr6 student: stud6; courseRecords: tr6CorsRecds; currentSections: tr6CrntSecs.

tr6CorsRecds purpose: 'Course Records for tr6'.
tr6CorsRecds setof add: corsRecd1; add: corsRecd3; add: corsRecd5;
                    add: corsRecd7; add: corsRecd9; add: corsRecd10;
 add: corsRecd11.
tr6CorsRecds numCourseRecords: (tr6CorsRecds setof) size.
tr6CorsRecds transcript: tr6.
```

```
tr6CrntSecs purpose: 'Current sections of tr6'.
tr6CrntSecs setof add: sec12; add: sec14.
tr6CrntSecs numSections: (tr6CrntSecs setof) size.

        "-------------------------------------------------------"

pd7 name.first: 'Rajesh'; name.middle: 'Babulal'; name.last: 'Parikh';
    sex: 'Male'; maritalStatus: 'Married';
    birthdate.day: 4; birthdate.month: 'Oct'; birthdate.year: 1964.


stud7  perData: pd7;
        studentId: 7000;
        supervisor: fullProf2;
        project: 'not applicable';
        thesis: 'Design and implementation of an Event/Trigger mechanism
                to enfore various constraints on CAD Databases.';

        transcript: tr7.

tr7 student: stud7; courseRecords: tr7CorsRecds; currentSections: tr7CrntSecs.

tr7CorsRecds purpose: 'Course Records for tr7'.
tr7CorsRecds setof add: corsRecd1; add: corsRecd4; add: corsRecd5;
                        add: corsRecd7; add: corsRecd9; add: corsRecd10;
 add: corsRecd11; add: corsRecd13.
tr7CorsRecds numCourseRecords: (tr7CorsRecds setof) size.
tr7CorsRecds transcript: tr7.

tr7CrntSecs purpose: 'Current sections of tr7'.
tr7CrntSecs setof add: sec15.
tr7CrntSecs numSections: (tr7CrntSecs setof) size.

        "-------------------------------------------------------"

pd8 name.first: 'Vivian'; name.middle: 'Watson'; name.last: 'Gomes';
    sex: 'Female'; maritalStatus: 'Single';
    birthdate.day: 12; birthdate.month: 'Dec'; birthdate.year: 1966.

stud8  perData: pd8;
        studentId: 8000;
        supervisor: fullProf1;
        project: 'not applicable';
        thesis: 'Design and implementation of a Frame-based Knowledge
                Base';
```

```
        transcript: tr8.

tr8 student: stud8; courseRecords: tr8CorsRecds; currentSections: tr8CrntSecs.

tr8CorsRecds purpose: 'Course Records for tr8'.
tr8CorsRecds setof add: corsRecd2; add: corsRecd3; add: corsRecd6;
                        add: corsRecd7; add: corsRecd9; add: corsRecd10;
 add: corsRecd11; add: corsRecd12.
tr8CorsRecds numCourseRecords: (tr8CorsRecds setof) size.
tr8CorsRecds transcript: tr8.

tr8CrntSecs purpose: 'Current sections of tr8'.
tr8CrntSecs setof add: sec15.
tr8CrntSecs numSections: (tr8CrntSecs setof) size.

        "-------------------------------------------------------"

pd9 name.first: 'Julia'; name.middle: 'Collins'; name.last: 'Richards';
    sex: 'Female'; maritalStatus: 'Single';
     birthdate.day: 1; birthdate.month: 'Mar'; birthdate.year: 1965.

stud9  perData: pd9;
        studentId: 9000;
        supervisor: fullProf1;
        project: 'not applicable';
        thesis: 'Design and execution of a Petri Net';

        transcript: tr9.

tr9 student: stud9; courseRecords: tr9CorsRecds; currentSections: tr9CrntSecs.

tr9CorsRecds purpose: 'Course Records for tr9'.
tr9CorsRecds setof add: corsRecd2; add: corsRecd3; add: corsRecd6;
                        add: corsRecd8; add: corsRecd9; add: corsRecd10;
 add: corsRecd11; add: corsRecd12.
tr9CorsRecds numCourseRecords: (tr9CorsRecds setof) size.
tr9CorsRecds transcript: tr9.

tr9CrntSecs purpose: 'Current sections of tr9'.
tr9CrntSecs setof add: sec15.
tr9CrntSecs numSections: (tr9CrntSecs setof) size.

        "-------------------------------------------------------"

pd10 name.first: 'Tanya'; name.middle: 'Fedrick'; name.last: 'Roberts';
    sex: 'Female'; maritalStatus: 'Married';
    birthdate.day: 18; birthdate.month: 'Jan'; birthdate.year: 1959.
```

```
stud10 perData: pd10;
        studentId: 9100;
        supervisor: assProf1;
        project: 'Developement of a user friendly interface for an
                existing NJIT Registration Database ';
        thesis: 'not applicable';

        transcript: tr10.

tr10 student: stud10; courseRecords: tr10CorsRecds; currentSections: tr10CrntSecs.

tr10CorsRecds purpose: 'Course Records for tr10'.
tr10CorsRecds setof add: corsRecd2; add: corsRecd4; add: corsRecd5;
  add: corsRecd7; add: corsRecd9; add: corsRecd10;                                    add:
corsRecd11; add: corsRecd13.

tr10CorsRecds numCourseRecords: (tr10CorsRecds setof) size.
tr10CorsRecds transcript: tr10.

tr10CrntSecs purpose: 'Current sections of tr10'.
tr10CrntSecs setof add: sec14.
tr10CrntSecs numSections: (tr1CrntSecs setof) size.

"-------------------------------------------------------------------------------------------------------------------"


apPd1 name.first: 'Rick'; name.middle: 'Collin'; name.last: 'Chen';
        sex: 'Male'; maritalStatus: 'Divorced';
        birthdate.day: 12; birthdate.month: 'Jun'; birthdate.year: 1945.

assProf1
    perData: apPd1;
    employeeNr: 9001.

        "-----------------------------------------------------"

apPd2 name.first: 'Peter'; name.middle: 'Ram'; name.last: 'Mohan';
        sex: 'Male'; maritalStatus: 'Married';
        birthdate.day: 31; birthdate.month: 'May'; birthdate.year: 1948.

assProf2
     perData: apPd2;
     employeeNr: 9002.
```

```
        "-------------------------------------------------------"

apPd3 name.first: 'Mike'; name.middle: 'Philip'; name.last: 'Romando';
        sex: 'Male'; maritalStatus: 'Single';
        birthdate.day: 2; birthdate.month: 'July'; birthdate.year: 1948.

assProf3
    perData: apPd3;
    employeeNr: 9003.

        "-------------------------------------------------------"

fpPd1 name.first: 'David'; name.middle: 'Henry'; name.last: 'Forsyth';
        sex: 'Male'; maritalStatus: 'Married';
        birthdate.day: 4; birthdate.month: 'Feb'; birthdate.year: 1940.

fullProf1
    perData: fpPd1;
    employeeNr: 9020.

        "-------------------------------------------------------"

fpPd2 name.first: 'John'; name.middle: 'Patrick'; name.last: 'Herman';
        sex: 'Male'; maritalStatus: 'Married';
        birthdate.day: 14; birthdate.month: 'Nov'; birthdate.year: 1940.

fullProf2
    perData: fpPd2;
    employeeNr: 9021.



"-------------------------------------------------------------------------------------------------------------------"


corsRecd1 grade: 3.0;
    semesterTaken: 'Fall 1989';
    course: cors1.

corsRecd2 grade: 4.0;
    semesterTaken: 'Spring 1989';
    course: cors1.

corsRecd3 grade: 3.0;
    semesterTaken: 'Fall 1989';
```

course: cors2.

corsRecd4 grade: 3.5;
semesterTaken: 'Spring 1989';
course: cors2.

corsRecd5 grade: 3.0;
semesterTaken: 'Spring 1989';
course: cors3.

corsRecd6 grade: 3.5;
semesterTaken: 'Spring 1990';
course: cors3.

corsRecd7 grade: 2.0;
semesterTaken: 'Fall 1989';
course: cors4.

corsRecd8 grade: 4.0;
semesterTaken: 'Spring 1989';
course: cors4.

corsRecd9 grade: 3.0;
semesterTaken: 'Fall 1989';
course: cors5.

corsRecd10 grade: 3.5;
semesterTaken: 'Spring 1989';
course: cors6.

corsRecd11 grade: 3.5;
semesterTaken: 'Spring 1989';
course: cors7.

corsRecd12 grade: 2.5;
semesterTaken: 'Fall 1989';
course: cors8.

corsRecd13 grade: 3.0;
semesterTaken: 'Spring 1989';
course: cors8.

"----------"

cors1 name: 'Computer Architecture';

```
    department: 'Computer Science';
    creditHours: 3;
    number: 'CIS610'.

cors2  name: 'Data Communications';
    department: 'Computer Science';
    creditHours: 3;
    number: 'CIS615'.

cors3  name: 'Data Structures and Algorithms';
    department: 'Computer Science';
    creditHours: 3;
    number: 'CIS620'.

cors4  name: 'Database Management System Design';
    department: 'Computer Science';
    creditHours: 3;
    number: 'CIS625'.

cors5  name: 'Operating System Design';
    department: 'Computer Science';
    creditHours: 3;
    number: 'CIS626'.

cors6  name: 'Interactive Graphics';
    department: 'Computer Science';
    creditHours: 3;
    number: 'CIS640'.

cors7  name: 'Software Engineering';
    department: 'Computer Science';
    creditHours: 3;
    number: 'CIS641'.

cors8  name: 'Management of Information Systems';
    department: 'Computer Science';
    creditHours: 3;
    number: 'CIS690'.

cors9  name: 'Masters Project';
    department: 'Computer Science';
    creditHours: 3;
    number: 'CIS700'.

cors10  name: 'Masters Thesis';
    department: 'Computer Science';
    creditHours: 3;
```

number: 'CIS711'.

"----------"

sec1 sectionNo: 'CIS610:100'; capacity: 35; instructor: assProf1.
sec2 sectionNo: 'CIS610:101'; capacity: 40; instructor: assProf1.
sec3 sectionNo: 'CIS615:101'; capacity: 35; instructor: assProf3.
sec4 sectionNo: 'CIS615:102'; capacity: 30; instructor: assProf3.
sec5 sectionNo: 'CIS620:100'; capacity: 35; instructor: assProf2.
sec6 sectionNo: 'CIS620:102'; capacity: 45; instructor: assProf2.

sec7 sectionNo: 'CIS625:100'; capacity: 25; instructor: fullProf2.
sec8 sectionNo: 'CIS625:101'; capacity: 35; instructor: fullProf2.
sec9 sectionNo: 'CIS626:100'; capacity: 25; instructor: assProf2.
sec10 sectionNo: 'CIS640:100'; capacity: 35; instructor: assProf1.
sec11 sectionNo: 'CIS641:102'; capacity: 35; instructor: assProf1.
sec12 sectionNo: 'CIS690:101'; capacity: 15; instructor: fullProf1.
sec13 sectionNo: 'CIS690:102'; capacity: 25; instructor: fullProf1.

sec14 sectionNo: 'CIS700:100'; capacity: 15.
sec15 sectionNo: 'CIS711:100'; capacity: 15.

"----------"

"*Above database is saved as follows:*"

univStudents ← Students new.

univStudents setof add: stud1; add: stud2; add: stud3; add: stud4; add: stud5;
add: stud6; add: stud7; add: stud8; add: stud9; add: stud10.
univStudents numStudents: univStudents setof size.

univStudents makePersistent surrogate.

"*got -> 329.*
old -> 163, 174"

"----------"

"

SOURCE CODE FOR CREATING THE RESUME DATABASE

This file contains Smalltalk code to create a database of employees and their resumes. The following instances have been created:

Employees:

assProf1	*AssiProfessor (1)*
fullProf1	*FullProfessor (1)*
aStaff1	*AdminStaff (1)*
univEmps	*Employees (1)*

This Smalltalk source code is in the file /OOUDB/impl/srcCodes/resDBsc

salil kulkarni
NJIT, New Jersey, USA.
Dec. 1990
"

"----------"

```
|
apPd1 assProf1
ap1Res
ap1ForEds ap1Bdeg ap1Mdeg ap1Pdeg
ap1Teach ap1Ateach ap1Acors2 cors2
ap1Exp ap1AcadApp
ap1Rgrants ap1Rgrant1
ap1Pubs ap1RCpapers ap1RCpaper1 ap1RJpapers ap1RJpaper1

fpPd1 fullProf1
fp1Res
fp1ForEds fp1Bdeg fp1Mdeg fp1Pdeg fp1IforEds IforEd1
fp1Teach fp1Ateach fp1NAteach fp1Acors1 cors1 fp1NAcors1
fp1Exp fp1AcadApp fp1Consult
fp1Rgrants fp1Rgrant1
fp1Hnrs fp1Hnr1
fp1Pubs fp1RCpapers fp1RCpaper1 fp1RCpaper2 fp1RCpaper3

aStaffPd1 aStaff1
aStaff1Res
aStaff1ForEds aStaff1Bdeg
aStaff1Exp aStaff1Exp1 aStaff1Exp2
```

univEmps
|

"----------"

apPd1 ← PersonData new.
assProf1 ← AssiProfessor new.
ap1Res ← Resume new.
ap1ForEds ← FormalEducations new.
ap1Bdeg ← BachelorDegree new.
ap1Mdeg ← MasterDegree new.
ap1Pdeg ← PhDDegree new.
ap1Teach ← TeachingActivities new.
ap1Ateach ← AcademicTeachings new.
ap1Acors2 ← AcademicTeaching new.
cors2 ← Course new.
ap1Exp ← Experiences new.
ap1AcadApp ← AcademicAppointment new.
ap1Rgrants ← ResearchGrants new.
ap1Rgrant1 ← ResearchGrant new.
ap1Pubs ← Publications new.
ap1RCpapers ← RefereedConferencePapers new.
ap1RCpaper1 ← RefereedConferencePaper new.
ap1RJpapers ← RefereedJournalPapers new.
ap1RJpaper1 ← RefereedJournalPaper new.

fpPd1 ← PersonData new.
fullProf1 ← FullProfessor new.
fp1Res ← Resume new.
fp1ForEds ← FormalEducations new.
fp1Bdeg ← BachelorDegree new.
fp1Mdeg ← MasterDegree new.
fp1Pdeg ← PhDDegree new.
fp1IforEds ← InFormalEducations new.
IforEd1 ← InFormalEducation new.
fp1Teach ← TeachingActivities new.
fp1Ateach ← AcademicTeachings new.
fp1NAteach ← NonAcademicTeachings new.
fp1Acors1 ← AcademicTeaching new.
cors1 ← Course new.
fp1NAcors1 ← NonAcademicTeaching new.
fp1Exp ← Experiences new.
fp1AcadApp ← AcademicAppointment new.
fp1Consult ← Consulting new.
fp1Rgrants ← ResearchGrants new.
fp1Rgrant1 ← ResearchGrant new.

```
fp1Hnrs ← Honors new.
fp1Hnr1 ← Honor new.
fp1Pubs ← Publications new.
fp1RCpapers ← RefereedConferencePapers new.
fp1RCpaper1 ← RefereedConferencePaper new.
fp1RCpaper2 ← RefereedConferencePaper new.
fp1RCpaper3 ← RefereedConferencePaper new.

aStaffPd1 ← PersonData new.
aStaff1 ← AdminStaff new.
aStaff1Res ← Resume new.
aStaff1ForEds ← FormalEducations new.
aStaff1Bdeg ← BachelorDegree new.
aStaff1Exp ← Experiences new.
aStaff1Exp1 ← NonAcademicAppointment new.
aStaff1Exp2 ← NonAcademicAppointment new.



"-------------------------------------------------------------------------------------------------------------------------------"

apPd1 name.first: 'Mary'; name.middle: 'Stephan'; name.last: 'Collins';
        sex: 'Female'; maritalStatus: 'Married';
        birthdate.day: 2; birthdate.month: 'Dec'; birthdate.year: 1945.

assProf1
    perData: apPd1;
    employeeNr: 9001;
    resume: ap1Res.

ap1Res
    jobTitle: 'Assistant Professor';
    belongsTo: assProf1;
    personalData: apPd1;
    formalEducation: ap1ForEds;
    teachingActivity: ap1Teach;
    experience: ap1Exp;
    grant: ap1Rgrants;
    publications: ap1Pubs.

ap1ForEds setof add: ap1Bdeg; add: ap1Mdeg; add: ap1Pdeg.

ap1Bdeg area: 'Computer Science'; yrGranted: 'Jan 1969';
          university: 'Case Western Reserve University';
          projectTitle: 'Design of a sub-set of a real time operating system'.

ap1Mdeg area: 'Computer Science'; yrGranted: 'Jan 1974';
```

university: 'California Technoloy Institute';
thesisTitle: 'A Distributed Query optimizer for a prototype Distributed Database '.

ap1Pdeg area: 'Computer Science'; yrGranted: 'Jan 1985';
university: 'University of San Jose';
dissertationTitle: 'Design of TDDBMS, The Distributed Database Management System'.

ap1Teach academic: ap1Ateach.

ap1Ateach setof add: ap1Acors2.
ap1Acors2 courseListing: cors2.

cors2 name: 'Principles of Distributed System Design';
number: 'CIS642';
department: 'Computer Science'.

ap1Exp setof add: ap1AcadApp.

ap1AcadApp
title: 'Special Lecturer';
employer: 'University of San Jose'.

ap1Rgrants setof add: ap1Rgrant1.

ap1Rgrant1
agency: 'Xerox Corporation';
amount: 20000.0.

ap1Pubs conferences: ap1RCpapers;
journals: ap1RJpapers.

ap1RCpapers setof add: ap1RCpaper1.

ap1RCpaper1
authors: 'Michelle Gilbert, Tom Hanks and Mary Collins';
title: 'An open architecture for a distributed database system';
year: 'Oct 1985';
conference: 'Fifth International Conference on Very Large Databases';
volume: 1;
pageNum: 13.

ap1RJpapers setof add: ap1RJpaper1.

ap1RJpaper1
authors: 'Mary Collins and Henry Stewart';
title: 'Query optimization rules for large distributed database systems';

```
    year: 'Sept 1988';
    journal: 'Third International Conference on Distributed Systems';
    volume: 2;
    pageNum: 313.


"-----------------------------------------------------------------------------------------------------------------------"

fpPd1 name.first: 'David'; name.middle: 'Henry'; name.last: 'Forsyth';
        sex: 'Male'; maritalStatus: 'Married';
        birthdate.day: 4; birthdate.month: 'Feb'; birthdate.year: 1940.

fullProf1
     perData: fpPd1;
     employeeNr: 9020;
     resume: fp1Res.

fp1Res
    jobTitle: 'Full Professor';
    belongsTo: fullProf1;
    personalData: fpPd1;
    formalEducation: fp1ForEds;
    informalEducation: fp1IforEds;
    teachingActivity: fp1Teach;
    experience: fp1Exp;
    grant: fp1Rgrants;
    honor: fp1Hnrs;
    publications: fp1Pubs.

fp1ForEds setof add: fp1Bdeg; add: fp1Mdeg; add: fp1Pdeg.

fp1Bdeg area: 'Electrical Engineering'; yrGranted: 'Jan 1962';
          university: 'University of California at San Diego';
          projectTitle: 'Simulation of a Token Ring Network'.

fp1Mdeg area: 'Computer Science'; yrGranted: 'Jan 1966';
          university: 'University of California at San Diego';
          thesisTitle: 'Efficient storage implementations for RDBMS'.

fp1Pdeg area: 'Computer Science'; yrGranted: 'Jan 1972';
          university: 'University of California at Los Angeles';
          dissertationTitle: 'Design of VODAK, an object oriented database
management system'.

fp1IforEds setof add: IforEd1.

IforEd1
```

```
    course: 'Design principles of open data models'.

fp1Teach academic: fp1Ateach; nonAcademic: fp1NAteach.

fp1Ateach setof add: fp1Acors1.
fp1NAteach setof add: fp1NAcors1.

fp1Acors1 courseListing: cors1.

cors1 name: 'Advanced Database Management Systems';
        number: 'CIS632';
        department: 'Computer Science'.

fp1NAcors1  course: 'Object oriented data models';
                periodOfTime: '6 months';
                institution: 'AT & T'.

fp1Exp setof add: fp1AcadApp; add: fp1Consult.

fp1AcadApp
    title: 'Assistant Professor';
    employer: 'University of California at San Diego'.

fp1Consult
    title: 'Software Consultant';
    employer: 'AT & T'.

fp1Rgrants setof add: fp1Rgrant1.

fp1Rgrant1
    agency: 'California Bell Laboratories';
    amount: 100000.0.

fp1Hnrs setof add: fp1Hnr1.

fp1Hnr1
    honor: 'Best visiting faculty at AT & T in the year 1988'.

fp1Pubs conferences: fp1RCpapers.

fp1RCpapers setof add: fp1RCpaper1;  add: fp1RCpaper2;
                        add: fp1RCpaper3.

fp1RCpaper1
    authors: 'Harry Goodman and David Forsyth';
    title: 'Metaclasses as tools for open data models';
    year: 'Oct 1986';
```

```
    conference: 'Fifth International Conference on Very Large Databases';
    volume: 2;
    pageNum: 233.

fp1RCpaper2
    authors: 'David Forsyth';
    title: 'Linear Hashing Algorithms';
    year: 'Sept 1981';
    conference: 'Second International Conference on Data Management';
    volume: 1;
    pageNum: 33.

fp1RCpaper3
    authors: 'David Forsyth, P. B. Gere and H. Chen';
    title: 'Algorithms for quick retrieval of complex objects from database';
    year: 'Oct 1986';
    conference: 'Third International Conference on Data Management';
    volume: 3;
    pageNum: 101.

"-----------------------------------------------------------------------------------------------------------------------"

aStaffPd1 name.first: 'Joe'; name.middle: 'Fred'; name.last: 'Bowie';
        sex: 'Male'; maritalStatus: 'Married';
        birthdate.day: 12; birthdate.month: 'Nov'; birthdate.year: 1955.

aStaff1
    perData: aStaffPd1;
    employeeNr: 1001;
    resume: aStaff1Res.

aStaff1Res
    jobTitle: 'Assistant Manager';
    belongsTo: aStaff1;
    personalData: aStaffPd1;
    formalEducation: aStaff1ForEds;
    experience: aStaff1Exp.

aStaff1ForEds setof add: aStaff1Bdeg.

aStaff1Bdeg area: 'Buisness Administration'; yrGranted: 'Jan 1979';
          university: 'State University of New York'.

aStaff1Exp setof add: aStaff1Exp1; add: aStaff1Exp2.

aStaff1Exp1 title: 'Junior Finanance Manager';
```

```
                employer: 'The Cable Corporation of San Jose'.

aStaff1Exp2 title: 'Junior Financial Consultant';
                employer: 'AICD Computers Inc.'.
```

"----------"

"The above database is saved as: "

```
    univEmps ← Employees new.
    univEmps setof add: fullProf1; add: assProf1; add: aStaff1.
    univEmps makePersistent surrogate.
```

"got 238.
old ones -> 235"

"----------"

"

SOURCE CODE FOR CREATING THE COMMITTEES DATABASE

This file contains Smalltalk code to create a database of committees, their members and chair persons. The following instances have been created:

Committees:

BCom	. . . *UniversityBCommittee (1)*
sabCom	. . . *UnivSabaticalCommittee (1)*
distProfCom	*UnivDistProfCommittee (1)*
gradCom	*GraduateCommittee (1)*
resCom	*ResourceCommittee (1)*
depExeCom	*DeptExecutiveCommittee (1)*
depPTCom	*DeptPTCommittee (1)*

Members & Chairs:

fp1 to fp5	*FullProfessor (1)*
dp1 to dp3	*DistProfessor (3)*
sl1 to sl3	*SpecialLecturer (3)*
gAdv	*MSAdvisor (1)*
rAdv	*ResourceAdvisor (1)*
dlChair	*DeptChairPerson (1)*

This Smalltalk source code is in the file **/OOUDB/impl/srcCodes/commDBsc**

salil kulkarni
NJIT, New Jersey, USA.
Dec. 1990

---"

"*decleration of temporary objects i.e. variables*"
|
BCom BComMem
sabCom sabComMem
distProfCom distProfComMem
gradCom gradComMem
resCom resComMem
depExeCom depExeComMem
depPTCom depPTComMem

fp1Pd fp2Pd fp3Pd fp4Pd fp5Pd
fp1Com fp2Com fp3Com fp4Com fp5Com
fp1 fp2 fp3 fp4 fp5

```
dp1Pd dp2Pd dp3Pd
dp1Com dp2Com dp3Com
dp1 dp2 dp3

sl1Pd sl2Pd sl3Pd
sl1Com sl2Com sl3Com
sl1 sl2 sl3

gAdvPd rAdvPd d1ChairPd
gAdvCom rAdvCom
gAdv rAdv d1Chair

 univComs |

"-------------------------------------------------------------------------------------------------------------------------------"

"creation of new instances"


fp1Pd ← PersonData new.
fp2Pd ← PersonData new.
fp3Pd ← PersonData new.
fp4Pd ← PersonData new.
fp5Pd ← PersonData new.

dp1Pd ← PersonData new.
dp2Pd ← PersonData new.
dp3Pd ← PersonData new.

sl1Pd ← PersonData new.
sl2Pd ← PersonData new.
sl3Pd ← PersonData new.

gAdvPd ← PersonData new.
rAdvPd ← PersonData new.


d1ChairPd ← PersonData new.

fp1 ← FullProfessor new.
fp2 ← FullProfessor new.
fp3 ← FullProfessor new.
fp4 ← FullProfessor new.
fp5 ← FullProfessor new.


dp1 ← DistProfessor new.
```

```
dp2 ← DistProfessor new.
dp3 ← DistProfessor new.

sl1 ← SpecialLecturer new.
sl2 ← SpecialLecturer new.
sl3 ← SpecialLecturer new.

gAdv ← MSAdvisor new.
rAdv ← ResourceAdvisor new.

d1Chair ← DeptChairPerson new.

fp1Com ←Committees new.
fp2Com ←Committees new.
fp3Com ←Committees new.
fp4Com ←Committees new.
fp5Com ←Committees new.

dp1Com ←Committees new.
dp2Com ←Committees new.
dp3Com ←Committees new.

sl1Com ← Committees new.
sl2Com ← Committees new.
sl3Com ← Committees new.

gAdvCom ← Committees new.
rAdvCom ← Committees new.

BCom  ← UniversityBCommittee new.
sabCom ← UnivSabaticalCommittee new.
distProfCom ← UnivDistProfCommittee new.
gradCom ← GraduateCommittee new.
resCom ← ResourceCommittee new.
depExeCom ← DeptExecutiveCommittee new.
depPTCom ← DeptPTCommittee new.

BComMem ← FullProfessors new.
sabComMem ← Professors new.
distProfComMem ← DistProfessors new.
gradComMem ← FacultyMembers new.
resComMem ← FacultyMembers new.
depExeComMem ← Advisors new.
depPTComMem ← FullProfessors new.

"-----------------------------------------------------------------------------------------------------------------------"
```

"giving values to the attributes and relationships of the instances created"

```
fp1Pd name.first: 'Rick'; name.middle: 'Collin'; name.last: 'Chen';
        sex: 'Male'; maritalStatus: 'Divorced';
        birthdate.day: 12; birthdate.month: 'Jun'; birthdate.year: 1940.

fp1Com setof add: BCom; add: gradCom; add: depPTCom.
fp1Com numCommittees: fp1Com setof size.

fp1
    perData: fp1Pd;
    employeeNr: 9001;
    position: 'Full Professor';
    committees: fp1Com.


fp2Pd name.first: 'Peter'; name.middle: 'Ram'; name.last: 'Mohan';
        sex: 'Male'; maritalStatus: 'Married';
        birthdate.day: 31; birthdate.month: 'May'; birthdate.year: 1938.

fp2Com setof add: BCom; add: sabCom; add: resCom; add: depPTCom.
fp2Com numCommittees: fp2Com setof size.


fp2
     perData: fp2Pd;
     employeeNr: 9002;
    position: 'Full Professor';
     committees: fp2Com.

fp3Pd name.first: 'Mike'; name.middle: 'Philip'; name.last: 'Romando';
        sex: 'Male'; maritalStatus: 'Single';
        birthdate.day: 2; birthdate.month: 'July'; birthdate.year: 1939.

fp3Com setof add: BCom; add: sabCom; add: resCom.
fp3Com numCommittees: fp3Com setof size.

fp3
     perData: fp3Pd;
     employeeNr: 9003;
    position: 'Full Professor';
     committees: fp3Com.


fp4Pd name.first: 'David'; name.middle: 'Henry'; name.last: 'Forsyth';
```

```
        sex: 'Male'; maritalStatus: 'Married';
        birthdate.day: 4; birthdate.month: 'Feb'; birthdate.year: 1940.

fp4Com setof add: BCom; add: gradCom; add: depPTCom.
fp4Com numCommittees: fp4Com setof size.

fp4
     perData: fp4Pd;
     employeeNr: 9004;
    position: 'Full Professor';
     committees: fp4Com.


fp5Pd name.first: 'John'; name.middle: 'Patrick'; name.last: 'Herman';
        sex: 'Male'; maritalStatus: 'Married';
        birthdate.day: 14; birthdate.month: 'Nov'; birthdate.year: 1941.

fp5Com setof add: BCom; add: gradCom; add: depPTCom.
fp5Com numCommittees: fp5Com setof size.

fp5
     perData: fp5Pd;
     employeeNr: 9005;
    position: 'Full Professor';
     committees: fp5Com.

        "-----------------------------------------------"

dp1Pd name.first: 'Alberto'; name.middle: 'Pat'; name.last: 'Hadley';
        sex: 'Male'; maritalStatus: 'Married';
        birthdate.day: 4; birthdate.month: 'Oct'; birthdate.year: 1935.

dp1Com setof add: distProfCom; add: resCom.
dp1Com numCommittees: dp1Com setof size.

dp1
    perData: dp1Pd;
    employeeNr: 8001;
    position: 'Distinctive Professor';
    committees: dp1Com.

dp2Pd name.first: 'Paul'; name.middle: 'Ralph'; name.last: 'Neuman';
        sex: 'Male'; maritalStatus: 'Divorced';
        birthdate.day: 1; birthdate.month: 'Apr'; birthdate.year: 1938.

dp2Com setof add: distProfCom; add: sabCom.
dp2Com numCommittees: dp2Com setof size.
```

```
dp2
     perData: dp2Pd;
     employeeNr: 8002;
     position: 'Distinctive Professor';
     committees: dp2Com.

dp3Pd name.first: 'Sherry'; name.middle: 'Philip'; name.last: 'Appleton';
         sex: 'Female'; maritalStatus: 'Single';
         birthdate.day: 2; birthdate.month: 'July'; birthdate.year: 1935.

dp3Com setof add: distProfCom; add: sabCom; add: resCom.
dp3Com numCommittees: dp3Com setof size.

dp3
     perData: dp3Pd;
     employeeNr: 8003;
     position: 'Distinctive Professor';
     committees: dp3Com.

         "------------------------------------------------"

sl1Pd name.first: 'Ahmed'; name.middle: 'Phil'; name.last: 'Akhtar';
         sex: 'Male'; maritalStatus: 'Married';
         birthdate.day: 24; birthdate.month: 'Nov'; birthdate.year: 1949.

sl1Com setof add: resCom.
sl1Com numCommittees: sl1Com setof size.

sl1
     perData: sl1Pd;
     employeeNr: 7001;
     position: 'Special Lecturer';
     committees: sl1Com.


sl2Pd name.first: 'Susan'; name.middle: 'Rick'; name.last: 'Sommers';
         sex: 'Female'; maritalStatus: 'Divorced';
         birthdate.day: 11; birthdate.month: 'Apr'; birthdate.year: 1948.

sl2Com setof add: gradCom.
sl2Com numCommittees: sl2Com setof size.

sl2
     perData: sl2Pd;
     employeeNr: 7002;
     position: 'Special Lecturer';
```

```
    committees: sl2Com.

sl3Pd name.first: 'Sam'; name.middle: 'Tom'; name.last: 'Salecka';
        sex: 'Male'; maritalStatus: 'Single';
        birthdate.day: 25; birthdate.month: 'Jun'; birthdate.year: 1945.

sl3Com setof add: gradCom; add: resCom.
sl3Com numCommittees: sl3Com setof size.

sl3
    perData: sl3Pd;
    employeeNr: 7003;
    position: 'Special Lecturer';
    committees: sl3Com.

        "------------------------------------------------"

gAdvPd name.first: 'Dinesh'; name.middle: 'Mahesh'; name.last: 'Roy';
        sex: 'Male'; maritalStatus: 'Married';
        birthdate.day: 2; birthdate.month: 'Jan'; birthdate.year: 1941.

gAdvCom setof add: depExeCom.
gAdvCom numCommittees: gAdvCom setof size.

gAdv
    perData: gAdvPd;
    employeeNr: 5001;
    position: 'Graduate Advisor';
    committees: gAdvCom;
    incharge: gradCom.


rAdvPd name.first: 'Pamela'; name.middle: 'Lochan'; name.last: 'Pallet';
        sex: 'Female'; maritalStatus: 'Married';
        birthdate.day: 3; birthdate.month: 'Mar'; birthdate.year: 1939.

rAdvCom setof add: depExeCom.
rAdvCom numCommittees: rAdvCom setof size.

rAdv
    perData: rAdvPd;
    employeeNr: 5002;
    position: 'Resource Advisor';
    committees: rAdvCom;
    incharge: resCom.

d1ChairPd name.first: 'Johnny'; name.middle: 'Jim'; name.last: 'Sellers';
```

```
        sex: 'Male'; maritalStatus: 'Single';
        birthdate.day: 7; birthdate.month: 'Jan'; birthdate.year: 1940.

d1Chair
    perData: d1ChairPd;
    employeeNr: 4001;
    position: 'CIS Department Chair Person';
    comChair: depExeCom.



"-----------------------------------------------------------------------------------------------------------------------"

BComMem setof add: fp1; add: fp3; add: fp4; add: fp5.
BComMem numFullProfs: BComMem setof size.
BCom
    name: 'UCLA CIS Department B Committee';
    code: 'B';
    members: BComMem;
    chair: fp2.

sabComMem setof add: fp2; add: fp3; add: dp3.
sabComMem numProfessors: sabComMem setof size.
sabCom
    name: 'UCLA CIS Department Sabatical Committee';
    code: 'SAB';
    members: sabComMem;
    chair: dp2.

distProfComMem setof add: dp1; add: dp2.
distProfComMem numDistProfs: distProfComMem setof size.
distProfCom
    name: 'UCLA CIS Department Distinctive Professors Committe';
    code: 'DP';
    members: distProfComMem;
    chair: dp3.

gradComMem setof add: fp1; add: fp4; add: fp5; add: sl2; add: sl3.
gradComMem numFacultyMembers: gradComMem setof size.
gradCom
    name: 'UCLA CIS Department Graduate Students Committee';
    code: 'G';
    members: gradComMem;
    chair: gAdv.

resComMem setof add: fp2; add: fp3; add: dp1; add: dp3;
                    add: sl1; add: sl3.
```

```
resComMem numFacultyMembers: resComMem setof size.
resCom
    name: 'UCLA CIS Department Resourse Committee';
    code: 'R';
    members: resComMem;
    chair: rAdv.

depExeComMem setof add: gAdv; add: rAdv.
depExeComMem numAdvisors: depExeComMem setof size.
depExeCom
    name: 'UCLA CIS Department Executive Committee';
    code: 'DE';
    members: depExeComMem;
    chair: d1Chair.

depPTComMem setof add: fp1; add: fp2; add: fp4; add: fp5.
depPTComMem numFullProfs: depPTComMem setof size.
depPTCom
    name: 'UCLA CIS Department PT Committee';
    code: 'DPT';
    members: depPTComMem;
    chair: d1Chair.

"-------------------------------------------------------------------------------------------------------------------------"

"The above database is saved as: "

univComs ← Committees new.
univComs setof add: BCom; add: sabCom; add: distProfCom;
                add: gradCom; add: resCom; add: depExeCom;
                add: depPTCom.

univComs numCommittees: univComs setof size.

univComs makePersistent surrogate.
"got  340.
 old ones -> 300,292,284,276,268,252,243"



"-------------------------------------------------------------------------------------------------------------------------"
```

APPENDIX D

THE IMPLEMENTATION DOCUMENT

* INTRODUCTION

What is VODAK/VML ?

VODAK is a object oriented database system currently being developed at GMD-IPSI, Darmstadt, FRG. VML is the data manipulation language of VODAK. Both of them are still under design, and what we have used in this implementation is a prototype system.

What is this application ?

A university database schema was modeled using the Dual model by V.Teli and H. chao at NJIT in 1990. Since the VODAK/VML prototype does not support Dual model, the database schema was modified to suit the implementation.

* APPLICATION DEVELOPEMENT STEPS

(A general guide line to develope any application database using VODAK/VML, given the database schema)

Definition of the Classes

All the data types and classes were defined in VML and compiled. Data types have been defined as regular Vml classes. In this application, 167 classes were defined. To compile all the class definitions in one shot:

- put a '.' after each class definition
- on the right button menu, select *compiler*. In compiler, select *vml2*
- high light all the class definitions and select *compile* on middle button menu
- if two classes have pointers to each other (i.e a mutual relationship), then while compiling the first class a pop-up menu will appear to declare the non existence of the second one. Select *undeclared class* in this menu and the compiler will automatically compile both the classes correctly

Compilation of a Vml Class creates a Smalltalk class having the name = Vml class name + 'Type', eg. a Vml class *Person* after compilation creates a Smalltalk class *PersonType*.

All the 167 classes were defined to be category of 'UNIV-Db' and all data types category of 'UNIV-Db-Types'. Using *categoryOf* clause in Vml definition of a class is purely for organization of classes so that their Smalltalk counterparts could easily be located on the Smalltalk Browser.

Identification and Implementation of Interfaces to Classes

Interface of a class is a set of messages to which any instance of that class can respond. Interface for each class was already identified by V.Teli and H.chao. Implementation of the interface means writing instance methods, one for each message identified.

Since Vml does not support methods, they were coded in Smalltalk. Writing methods in the appropriate class is supported by the Browser. Each method was tested after coding by passing messages to dummy instances. No queires were thought of at this stage.

Populating the Database

After much effort and time it was realized that with the limited RAM, it was not possible to create a database having instances of all 167 classes. So four seperate databases were created such that there was roughly one-to-one mapping between the databases and the pages of structural heirarchy figure prepared by V.Teli and H. chao. The databases created were *Transcript, Resume, University* and *Committees* database.

Database is the representation of application objects as instances of classes. For eg. TranscriptDB would need instances of class Student (or it's subclasses) to represent actual students in a university. Each student has a transcript (instances of class TRanscript are required) which keeps a record of the details of all the courses he/she has taken till now (instances of classes Course, Section. CourseRecord etc. required). Thus a database must have instances and these instances have to be given values for their properties (attributes and relationships).

The source code to create the databases is in Smalltalk. All the four databases created have a similar code structure, which is as following:

- first part of the code contains decleration of temporary variables which are used as instances
- in the next part, the instances declared are actually created from their respective classes using message *new*
- after all instances are created, each one of them is given values for selected properties

Once a database is created it resides in temporary variables. To save it, the instance of interest are collected in a set and then saved on a special file called VODAK.VSS, the details of which are in section 3.2. Instances of interest in case of TranscriptDB would be all the students. They are collected in a variable *univStudents* which is instance of class Students and univStudents is saved.

Now, if a query is to be run on a database, the database must reside on RAM. So one can write each query such that it first loads the appropriate database from VODAK.VSS file into RAM and then executes the method associated with the query. Instead, it is better to keep

the database in RAM till all queries are run. What is done is that a code is written to load each of the four databases from file into four global variables *TranscriptDB*, *ResumeDB*, *UniversityDB* and *CommitteesDB*. Now any query can be run as many times as required.

Identification and Implementation of Queries

A set of typical queries were identified for each database so that they would demonstrate the correctness of the databases created. All queries are basically messages sent to the appropriate database i.e. the appropriate global variable.

Since the global variables are nothing but instances of some class (eg. global vaiable TranscriptDB is instance of class Students), queries are implemented as instance methods of that class.

The existing set of queries is given below, though one can easily code more queries an add them to the existing set:

TRANSCRIPT DATABASE:

TranscriptDB	getStudents	
TranscriptDB	getNumOfStudents	
TranscriptDB	getTranscriptFor:	'Rajesh Parikh'
TranscriptDB	getCurrentSectionsFor:	1000
TranscriptDB	getStudentsIfGpa:	3.9

UNIVERSITY DATABASE:

UniversityDB	getAllDivDeans
UniversityDB	getAllColDeans
UniversityDB	getAllSchDeans
UniversityDB	getAllDepChairs

RESUME DATABASE:

ResumeDB	getAllEmployees	
ResumeDB	getHonorsFor:	9020
ResumeDB	getConsultingsFor:	'David Forsyth'

(c) database-namemessage1: parameter1
message2: parameter2

where database-name is a global variable storing the database,
message is the selector of a message(query is a message)
parameter is the variable passed with a message

step7: Before running any query, load all the databases in global variables by executing the code under label "loading the databases in global variables". To run a query, plug in the required parameters, select the whole query by dragging right button on mouse so that it is high lighted and then choose do it from middle button menu. Results are displayed on System Transcript Window. All error messages because of invalid parameters are also displayed on this window.

step8: To exit the demonstration, open main Smalltalk menu by clicking any button outside any of the three windows described above and select quit. This will open a window, where you select Quit, without saving.

A list of Files

Following is the list of files in which the University Database is distributed and stored. It also includes documentation files.

1) Class and data type definition files: **OOUDB/impl/classDefs**

**.objfiles* - definition of all classes
datatypes - definition of all data types

2) Class interface files: **OOUDB/impl/classInterface**

commDBif - implementation of interface of classes in Committees database
trnsDBif - implementation of interface of classes in Transcript database
resDBif - implementation of interface of classes in Resume database
univDBif - implementation of interface of classes in University database

3) Source codes for databases: **OOUDB/impl/srcCodes**

commDBsc - source code to create Committees database
trnsDBsc - source code to create Transcript database
resDBsc - source code to create Resume database
univDBsc - source code to create University database

4) Documentation files: OOUDB/doc

implDoc - this file
Queries - code in the Query Window used in the demonstration