**Introduction**

As database systems have become more popular over the years developers are finding new restrictions in the relational data model used in a majority of database today. The relational model has limited data types, thus, for years researches have been developing object-oriented database models which allow for complex data types. To deliver a database that is able to support the complex data types, developers are using a hybrid of the two models, an Object-Relational data model.

The focus of this project is on object-oriented relational database design. Two parts will be developed to accomplish this project, the first part will be the interface and its structure to pass data to the second part which will be the database itself. What is being attempted here is to provide support of object assignment, object identity, and object copying of data from an interface to a data management tool [1].

The desired outcome from this project will be to string together a query starting with an initial standard query. As the query delves deeper into the topic initial information will be passed on so that initial information does not need to be included. Some major features of the project are encapsulation which will contain parts of a procedure that is encapsulated into the data being passed and inheritance, which will pass data from a base table to an entry into another table.

**Programming Languages**

I decided to use Java as the primary programming language for this project. I chose Java because of its flexibility, once for all. There are a lot of different aspects to this project and I felt comfortable working with Java and connecting all the parts. For example, the JavaServer Pages (JSP) framework was used to build the pages that the user interacts with and the JSP Standard Tag Library was used to extend the use of the JSP framework. I used hyper-text markup language (HTML) to create the display pages along with Java, and I used cascading style sheets (CSS) to format pages being displayed.

**Integrated Development Environment (IDE)**

Because I chose to use Java as the primary programming language, I chose to use the integrated development environment (IDE) Eclipse over the other popular IDE, NetBeans. One thing that lead me to use Eclipse is that it has a project management framework called Maven, this helped me keep all the modules organized. Also, the Eclipse IDE has a lot of third party plugins that make development in this language efficient.

**Application Programming Interface (API)**

Tomcat was used as a server emulator in order to serve the pages to the web browser and as the translator for the web browsers interaction with the database. I used SQL Server 2012, Developers Edition, as the relational database management system (RDBMS). I chose this RDBMS because it is a fully functional version of SQL Server Enterprise. Along with the Eclipse IDE, I used SQL Server Management Studio to build the database and manage the data while creating the project.

**System Description**

The system was created using JavaServer Pages on the front-end to interact with the user and the database on the back-end. There are several pages links on the home page with options of adding a course, adding a teacher, and adding a student. Each of these links goes to their own JSP. The addCourse.jsp page will display the fields where the user will enter the information that will be entered into the database. On the addCourse.jsp page the primary key is ID this is auto-incremented so that it is always a distinct number this is not displayed on the input screen. The course name is a required field, as such, the user is required to enter a course name. The credits field is also a required field so the user will be required to enter a number. The teacher.jsp is a JavaServer Page that allows the user to enter a new teach. It also has an ID which is set to auto-increment so the user is not allowed to enter data into the database for this field. It also requires a last name and a designation for the teacher. The student.jsp page is where the user will enter student data that will be passed to the database. Again, the ID is set auto-increment so that the field is always distinct. The listCourse.jsp page will list all the course available from the Course table.

**Database Setup**

The fields in the Course table that the user will need to enter are:

* ID – this is the primary key and is set to auto-increment so that it will always be a distinct number.
* Name – this is the name of the course and is a required field.
* Credits – this is a required field that includes the number of credits received upon successful completion of the course.
* TeacherID – this is a foreign key that links the course table to the teacher table.

The fields in the teacher table that the user will need to enter are:

* ID – this is the primary key for the Teacher table. It is set to auto-increment so that it will always be a distinct number.
* LastName – this is a required field that needs the teacher’s last name as input.
* FirstName – this not a required field but if data is entered it is the teacher’s first name.
* Designation – this is a required field that designates the teacher’s status.

CREATE TABLE Teachers (id int PRIMARY KEY AUTO\_INCREMENT

, PersonTypeID as 1 persisted -- teacher

, Designation int not null

, foreign key (PersonID, PersonTypeID) references Person(PersonID, PersonTypeID)) [2]

The fields in the student table are:

* ID – this is the primary key for the student table. It is set to auto-increment so that it will always be a distinct number.
* LastName – this is a required field. This field holds the student’s last name.
* FirstName – this is not a required field. If data is entered it will be the student’s first name.
* State – this field inherits the data from the Person table. If it is different from the default the user is allowed to update the information.
* EnrolledSince – this is a required field. It inherits data from the Person table. The date defaults to the current year as this is part of the enrollment process.

create table Students (PersonID int primary key

, PersonTypeID as 2 persisted -- student

, EnrollmentDate datetime

, foreign key (PersonID, PersonTypeID) references People(PersonID, PersonTypeID)) [2]

There is an addition table in the Course Management database that is used as a base table. It contains information that is consistent with data that can be identified as belonging to a person. The Person table is a base table and it holds information that will be passed on to other tables when an option is selected from the main page. For example, there is a *Person* table which holds information such as the state of origin for a person which will default to the state in which the school is located or the date enrolled since field with will default to the current year.

The base table is the *Person* table. Once the person is created, you can add related row to the appropriate table, depending what type they are.

USE CourseManagement;

Drop Table dbo. Person;

GO

Create Table Person(ID Int Primary Key Not NULL Auto\_Increment

, State varchar(2)); [2]

The PersonType table is another base table that consists of the type of person being entered. For example; 1 = teacher, 2 = student, 3 = parent.

The *PersonType* table allows us to select the type of person we want to make.

INSET INTO TABLE PersonType

Select 1, ‘Teacher’ UNION ALL

Select 2, ‘Student’ UNION ALL

Select 3, ‘Parent’ [2]

With the *PersonType* we will add a *PersonType* column to our *Person* table.

CREATE TABLE Person (ID Int Primary Key Auto\_Increment

, PersonTypeID Int References PersonType(PersonTypeID) Not NULL

, Name varchar(10) Not NULL) [2]

When run the SQL will look something like this:

insert into People (PersonID, PersonTypeID, Name)

values (1,1,'Bob')

(1 row(s) affected)

insert into Students (PersonID, EnrollmentDate)

values (1,'2007-01-01')

**References**

1. Atkinson, M., DeWitt, D., Maier, D., Bancilhon, F., Dittrich, K., & Zdonik, S. (1990). The object oriented database system manifesto. In Deductive and object-oriented databases (pp. 223-240).

2. http://www.sqlteam.com/article/implementing-table-inheritance-in-sql-server