Project Report

James Kyle Harrison

Georgia Southern University

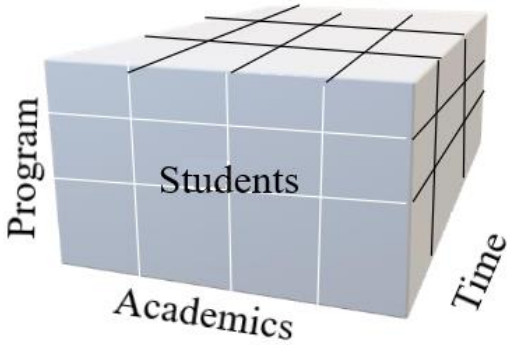
Data Model Design

Schema

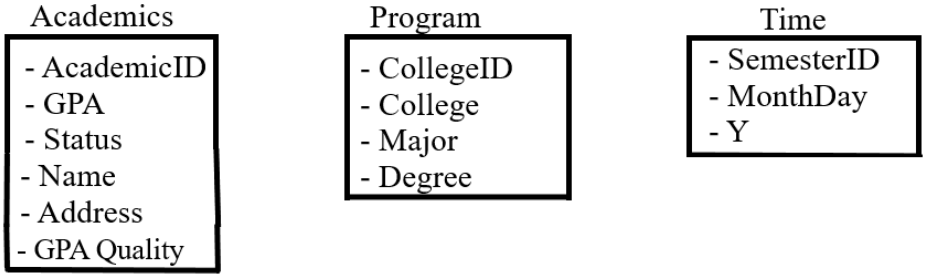
When it comes to designing a data warehouse for the University of Savannah at Richmond Hill (USRH), there is no denying that the subject of said data warehouse will be the students. With this and the requirements of having to study the students by major, degrees, colleges, GPAs, and semesters there are a variety of ways that this can be handled. For this particular implementation we will move forward with developing a data warehouse using the star schema as the dimensions (as explained below) do not have any duplicated attributes that can not be reduced even further.

Dimensions

This data warehouse has been elected to be a three-dimensional model to complete the requirements. The requirements can be broken down into the dimensions of academics (specifies student specific ID, GPA, enrollment status, name, and address), program (information on the college, major, and degree), and time (semester data).



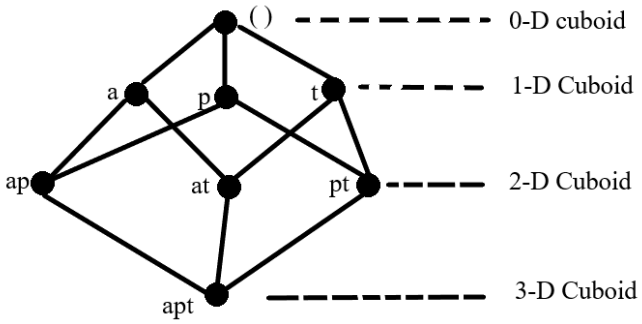
To further explain the justification of this design, let us look at the dimension tables for each assigned dimension.



The Academics dimension uses the AcademicID attribute (otherwise known as ID in the dataset collected by university authorities) as a primary key attribute for the subject to allow for the use of all of the student enrollment information. The simplest justification for this dimension and its layout is that it allows filtering around the ‘what’ axis in this subject and creates the way for analysis on the subject of a per-student basis. This dimension is made in mind for filtering

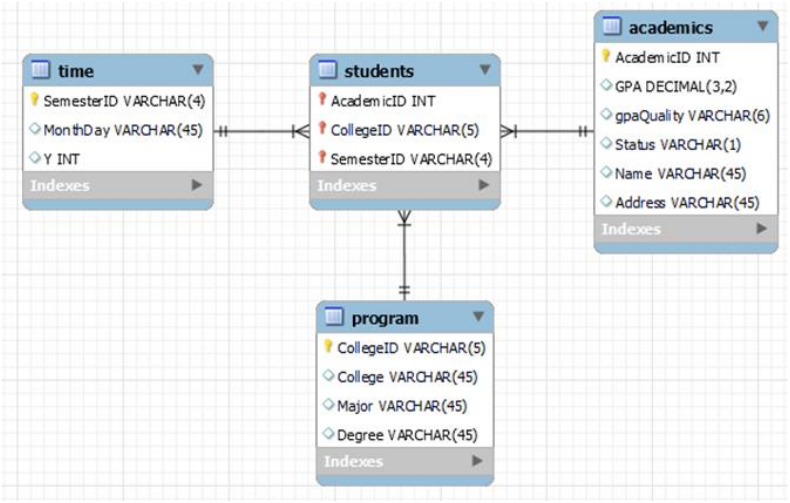
students on their GPA along with additional student personal information. The Program dimension uses the CollegeID attribute as its primary key for the subject relating to the school and degree. This ID will be used to filter for the subject's major, degree, and college and its layout allows for filtering around the 'where' axis in the subject and creates a way for analysis on the premises of a per-college basis. This allows for the pre-requisite filtering requirements on a college, degree, or major basis. Although the granularity of this dimension can be changed by creating a sub-dimension for the college attribute based off of the major, the data reduction of listing the college is not necessarily required. Finally, the Time dimension uses the SemesterID attribute as it's key for the subject of time to correctly filter using the semesters and time in general. This layout allows for filtering around the 'when' axis for the subject and will not need any changes to the granularity of time attributes as the analysis requirements only need to be met on a per-semester basis. This will meet the final requirement in the scope of analysis by allowing for filtering on the basis of semesters and year. With these three dimensions the following possible cuboids can be generated from analyses:

Three dimensions of Academics (a), Program (p), and Time (t):

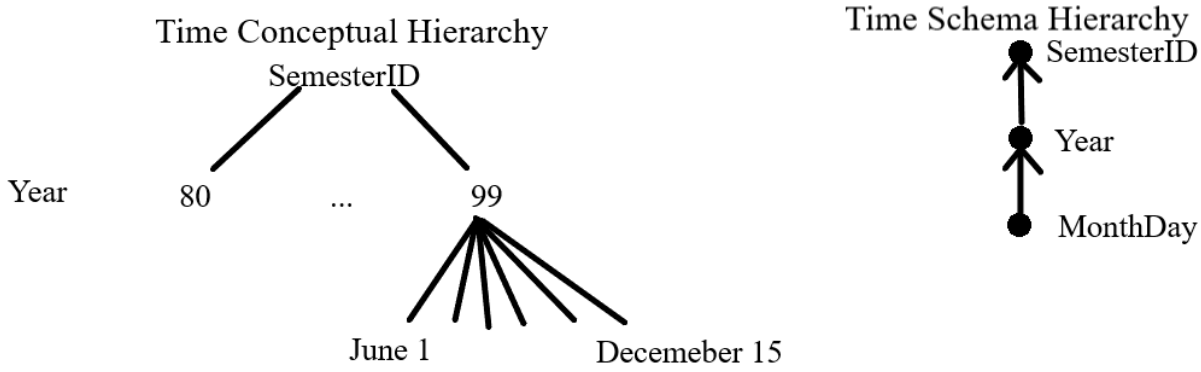


Fact Table and Hierarchies

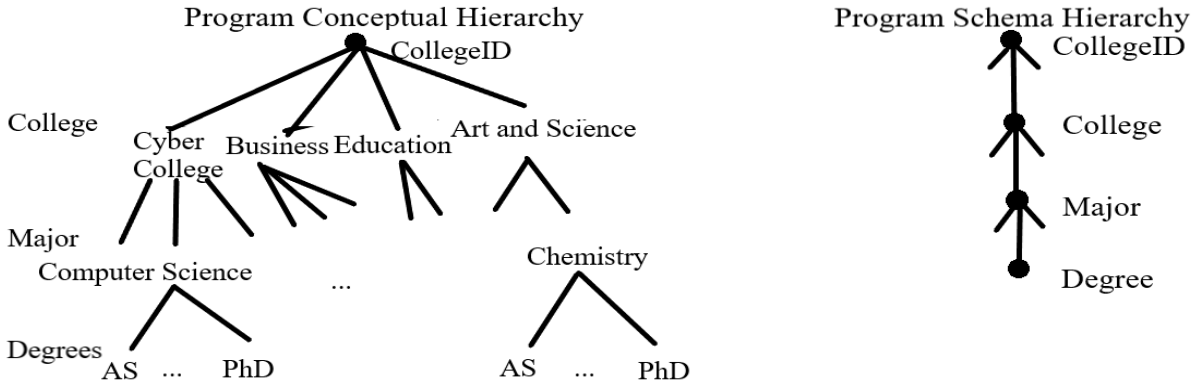
The fact table has a one-to-many relationship to the dimensions previously described, and thus the keys of these dimensions are within our fact table as this will allow for one to aggregate through some of the dimensions, but not all. Due to the lack of summable data within this data warehouse, the fact table will be just a set of keys to its dimensions and in result create a star schema with a fact table that will connect to the primary keys of each of the three dimensions by using them as foreign keys. This creates a fact table as described below:



Now that the fact table and its connections to the dimension tables have been established let's take a look at the concept and schema hierarchies for each of the following dimensions. Starting with the time dimensions hierarchies we can see a naturally total order of all concepts where the Semester ID when matched with the subjects will lead into identifying the year and then semester in the concept hierarchy, and the schema hierarchy will follow a hierarchy that moves from the lowest concepts to the highest concept in a simple manner.

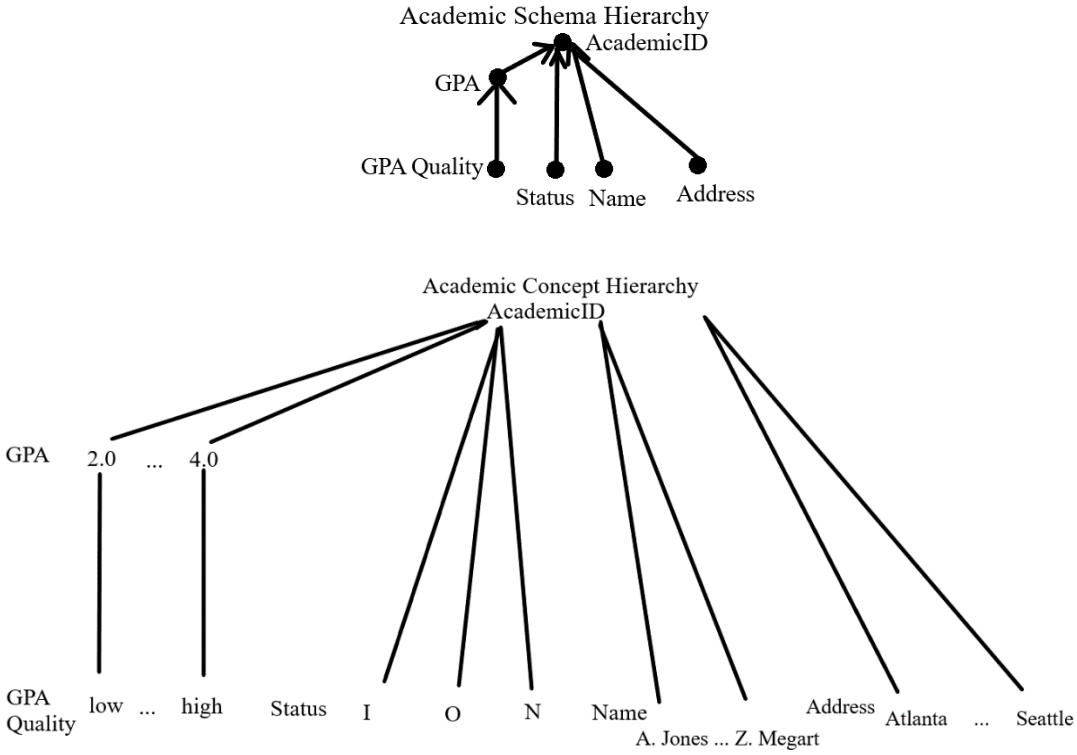


The program dimensions hierarchies will also be quite similar to the time dimensions hierarchies in that it too uses a naturally total order of all concepts. The CollegeID when compared to the subjects will find the college that the student is in. From here we can filter out what major they are by looking at the majors in the school, and finally be able to do the same with the degrees as not all degrees will apply to all majors (ex: you cannot get a PhD in the College of Education or MBA in Chemistry). This results in having another schema hierarchy where there exists an entire hierarchy that moves from the lowest concepts to the highest concept within the dimension.



Finally, there is the academics dimension, where the hierarchies are a bit more convoluted as this dimension is a partially ordered hierarchy, where only one set of footsteps is in a hierarchy. We are able to at least order GPA and the GPA Qualities such that this information can be derived from by declaring ranges for GPAs such that they can be categorized, however that does not seem to be the case for address and status based off of the provided data set. One would expect to be able to derive their student status of being an international or out-of-

state student based off of their address, but that does not appear to be the case, and thus they must remain out-of-hierarchy.



Implementation

Now that the modeling of the data warehouse has been completed it is time to being preparing to implement the model and the given data into the model. To prepare the data for the model we must first do a bit of extra processing on the data that we have been given. The data given comes from a few different sources that were able to successfully be merged into one file, however there are still some faults that can be found in this data. For example, the student IDs are not unique and there are a few entries that have the same ID; another issue that can be seen is also in the different ways that some of the student majors have been written (‘business admin’ vs ‘b administration’). Although the IDs can be fixed using a function to ensure that all IDs are made unique, creating a trained neural network to deal with the pedantic differences in major declarations would require significantly more time than it would to just fix the few instances of this happening manually, thus the students’ IDs will be fixed through automation and extreme outliers of the student majors will be changed manually.

The data is almost completely ready to be implemented into our data warehouse, however there are still attributes that must be addressed. Much of the data will fortunately be able to be converted directly from the dataset we were given, however there are a few things that must be derived and generated in order to properly follow the flow of this model. Starting with the simplest of the dimension tables, the Academics table will pull most of its data simply from the dataset with the exception being the GPA Quality. GPA Quality will be generated from the GPA

itself and always be stored as ‘high’, ‘medium’, or ‘low’ to describe what GPA the student has. The time dimension table and program dimension table are however a bit more complicated, as unlike the Academic dimension table where we are already given a student’s academic ID we will need to generate a SemesterID and CollegeID as the primary keys to describe the various combinations of the data within their tables. CollegeID will take the major and degree and combine them into an ID that is unique on a major and degree combination, and SemesterID shall do the same but with a combination of what semester and year combinations. The last part of data generation is in the Program dimension table, where we will derive what college a student is in by looking at their major according to the hierarchy that we defined earlier. With these values being generated, and the rest being pulled from our source data, we will finally be able to create the model in MySQL and begin loading the data into our model.

Academics Dimension Table	
Data Source Equivalent	**Table.ColumnName**
ID	academics.AcademicID
GPA	academics.GPA
Derived as GPA Quality: If GPA >= 3.0 THEN “high” If GPA > 2.0 AND GPA < 3.0 THEN “medium” ELSE THEN “low”	academics.gpaQual
Status	academics.Status
Name	academics.Name
Address	academics.Address

Time Dimension Table	
Data Source Equivalent	**Table.ColumnName**
LEFT(MonthDay, 0): IF = ‘Jul’ ||’Jun’ || ‘A’ THEN “S1”…”S4” IF = ‘D’ THEN “Fall” If = ‘M’ THEN “Spring” +Y	Time.SemesterID
MonthDay	Time.MonthDay
Y	Time.Y

Program Dimension Table	
Data Source Equivalent	**Table.ColumnName**
LEFT(Major, 2) + Degree	program.CollegeID
If Major = ‘Computer Sc’ || ‘Info Sc’ || ‘Applied Sc’ THEN “Cyber College” If Major = ‘Accounting’ || ‘B Admin’ || ‘Economics’ THEN “College of Business” If Major = “Elementary” || “ Secondary” THEN “College of Education” If Major = “Biology” || “Chemistry” THEN “College of Art and Science”	program.College
Major	program.major
Degree	program.degree

There is no denying between creating the model and loading the data that loading the data is a bit more advanced. With proper planning and correction of data types in order to create the primary key/foreign key connections between the three dimensional implementing the model is not too much of an issue, and as a result this model as able to be successfully implemented into MySQL such that we are able to recreate our star schema with our model that also has generational logic for the predescribed table attributes. With that being said, loading can finally begin to occur. There are various ETL tools to help with loading data quickly, however we will load the data without these tools simply by following the previously defined table/data source equivalents. Doing so will result in being able to query each table dimension to ensure that data has properly been generated as such:

SemesterID	MonthDay	Y
FA85	Dec-15	85
FA88	Dec-15	88
FA89	Dec-15	89
FA90	Dec-15	90

CollegeID	College	Major	Degree
AcBA	College of Business	Accounting	BA
ApMS	Cyber College	Applied Sc	MS
ApPhD	Cyber College	Applied Sc	PhD

AcademicID	GPA	gpaQuality	Status	Name	Address
100	3.10	high	I	A. Jones	Little Rock
101	3.20	high	N	A. Jones	Little Rock
107	3.90	high	N	M. Kochaal	Savannah
108	3.90	high	O	M. Big	Little Rock

Populating the fact table is a bit more complicated than just importing the data unlike the dimensional tables, but still quite feasible. To begin, we must create a staging table that will take our raw data and be inserted one-to-one directly into it. With this staging table we can use the raw data in conjunction with our defined dimensional tables data to finally populate the fact table using a query to compare the student id (primary key for Academic table), the time (MonthDay and Y that create primary key SemesterID for the Time table), and major and degree information (which creates the primary key of CollegeID for the Program table) in a query (query used described below). After using the query we are able to now confirm that the correct data has been applied to the fact table.

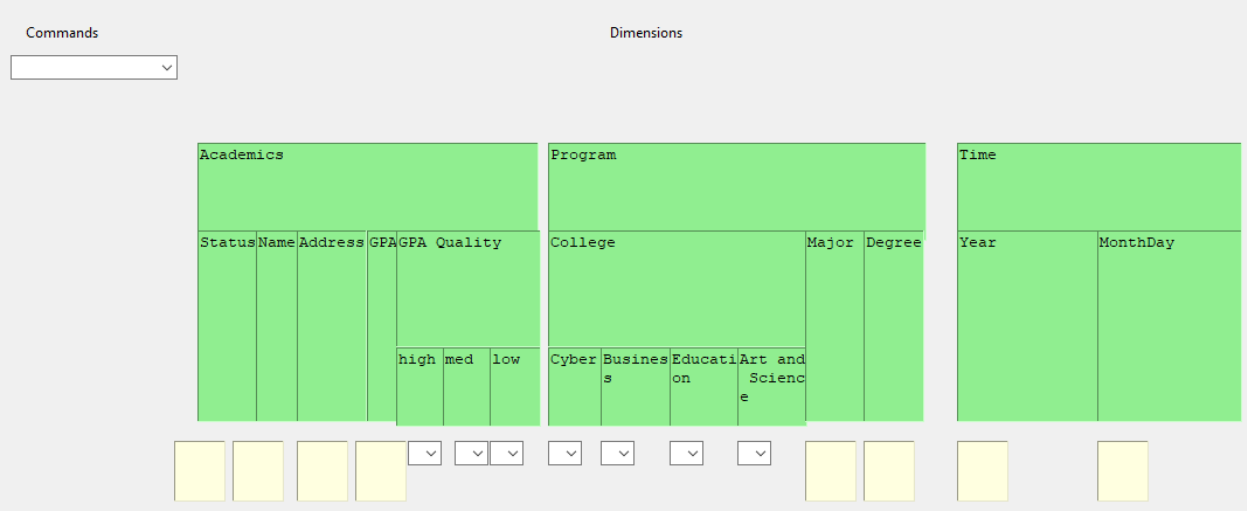
```
INSERT INTO students (AcademicID, CollegeID, SemesterID)
SELECT a.AcademicID, p.CollegeID, t.SemesterID
FROM academics a, program p, time t, staging s
WHERE a.AcademicID = s.ID
    AND p.Major = s.Major AND p.Degree = s.Degree
    AND t.MonthDay = s.MonthDay AND t.Y = s.Y;
```

AcademicID	CollegeID	SemesterID
100	CoBS	SP84
101	CoMS	SP87
107	ChBS	SP92
108	ChBS	SP92
109	ChAS	FA88
110	BiAS	FA88

Interface

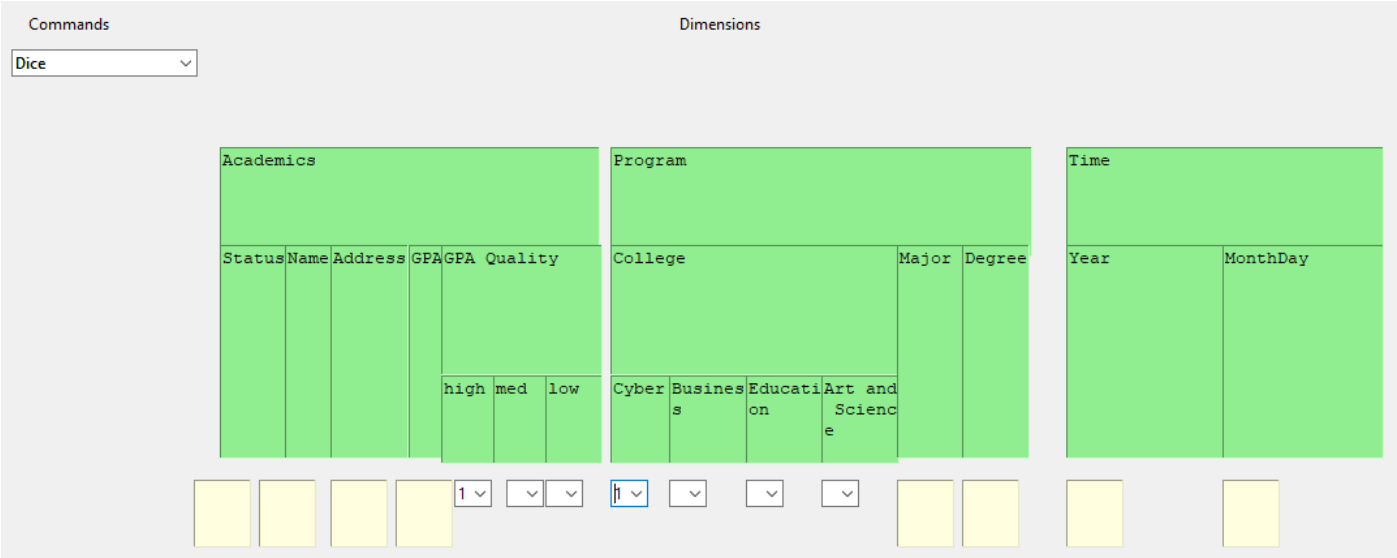
Now that the Data Warehouse has properly been implemented, we need some sort of user-friendly interface to allow the privileged staff of USRH to properly use the Data Warehouse. The interface will be a basic template of the generated data cube made in Python such that a user can simply click what options they would like to generate reports. This template is quite generic, but it allows for even the most basic user's to understand enough to a point that they can easily query without and knowledge of programming languages. First the user must choose their OLAP Operation on the Commands drop down box, then they must fill the data on the columns that they would like to use for querying the data. The options of selecting what to filter by can be determined by locating the properly labeled column that matches the criteria one would like to use, and then filling out the box below. For columns that have many of options, the user must fill them in by the proper names that are stored in the database, and if they want to filter by more than one item in the column they must add an ' OR ' between the two items in the same box. Some of these boxes are also replaced with dropdown menus that give the selection of '1' or '0' as these represent the lowest answers that can be found in the hierarchy of that

dimension. The user will select ‘1’ to filter by this option or a ‘0’ to not use it for querying; the user can also elect to not click on the box and leave it blank to also generate a query.



Once a user has successfully selected their required columns for a query they must simply press the ‘Submit’ button located at the bottom of the page to begin the creation and processing of the SQL query. This will be performed through the Mysql-connector-python driver, which creates the connection between the python interface and MySQL. With the connection being made the query results will be printed both in MySQL and also in Python.

This is not going to be in the final draft, however I added some images to show from current testing while I complete all of the components necessary to finalize Part 4:



```
SELECT s.AcademicID, s.CollegeID, s.SemesterID FROM students s, academics a, program p WHERE a.AcademicID = s.AcademicID AND a.gpaQuality
 'high' AND p.CollegeID = s.CollegeID AND p.College = 'Cyber College'
350, ApMS, S486
956, ApMS, FA89
175, ApPhD, S289
200, ApPhD, SP92
201, ApPhD, SP87
202, ApPhD, S287
352, ApPhD, S490
356, ApPhD, FA89
357, ApPhD, FA89
418, ApPhD, FA92
452, ApPhD, S488
524, ApPhD, FA92
653, ApPhD, FA91
681, ApPhD, SP91
702, ApPhD, SP87
703, ApPhD, S387
750, ApPhD, S284
132, CoAS, S492
232, CoAS, S491
100, CoBS, SP84
250, CoBS, S486
579, CoBS, SP84
687, CoBS, S284
739, CoBS, S184
101, CoMS, SP87
150, CoMS, S486
680, CoMS, SP90
986, CoMS, SP87
473, CoPhD, SP92
185, InBS, FA90
600, InBS, FA88
850, InBS, FA85
```