**Academic Student DB**

**INST 733 – Database Design**

**Spring 2015**

**Project by Bhoopesh and Anuj Shah**

**Academic Student DB**

**Group: Bhoopesh and Anuj Shah**

# INTRODUCTION

The goal of the project is to develop an academic program database for students who wish to pursue graduate and post-graduate studies. It is a one-stop all necessary database of the students and their standardized test scores, sectional scores in the tests, VISA details and the programs and degrees enrolled by them. This data base will assist in performing data analysis and decision making regarding the degrees and program offered and enrolled by the students, the test scores and the sectional cut offs accepted for the same. Relation between different tables and possible activities or operations of the database have been performed to deduce inferences from the database. We have tried to emulate the activities performed by students who wish to apply for higher studies. Database design contains information about students who have attended a particular university before, the range of GRE and TOEFL, students who have attended a particular university before, information of students who have got admits in particular university.

# TARGET AUDIENCE

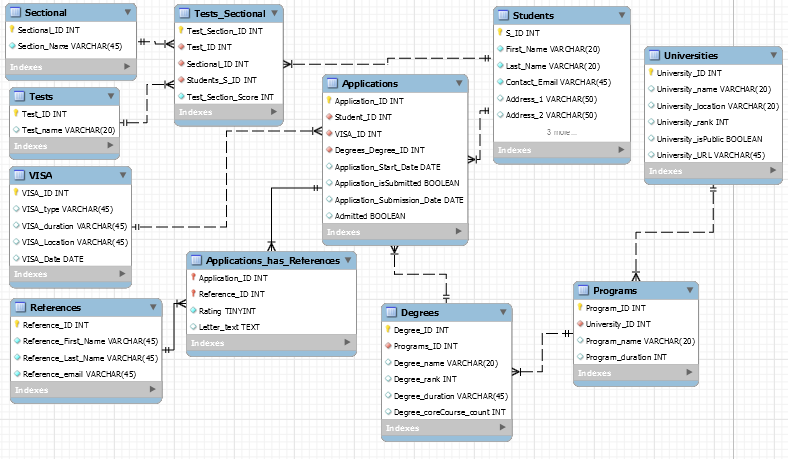
1. Students who wish to pursue Masters, PhD and other certificate courses
2. Students (alumni) who got admits/attending feed in information which will be used for showing up to other students (aspiring)
3. Staff who want to access the details of the batch

**LOGICAL DESIGN**

The process of logical designing involves arranging data into a series of logical relationships called entities and attributes. An entity represents a chunk of information and maps to a table. An attribute is a component of an entity and helps define the uniqueness of the entity. It is an abstraction of the data flows, inputs, and outputs. Logical design basically represents the steps to create a conceptual model.

As part of our logical design process we created E-R Diagram. It helped us to get a better understanding of what our physical design would be like.

**ER DIAGRAM:**

ER model consists of the following tables:

1. Students
2. Tests
3. Programs
4. Universities
5. Sectional
6. Applications
7. VISA
8. Degrees
9. Test\_Sectional
10. References
11. Application\_has\_References

The logical design process consisted of:

* Identifying entities
* Identifying attributes. This also included deciding on the primary key constraints.
* Deciding on the relationship between various entities. It was important to decide on the type of relationship (cardinality).

**PHYICAL DESIGN**

We used the forward engineering option of workbench to create the physical tables. It was important for us to have relevant datatypes added in the ER diagram.

By definition, **forward engineering** is a traditional process of moving from high-level

abstraction and logical, implementation-independent designs to the physical implementation of

a system. After the physical design was created we began populating the data using <http://www.databasetestdata.com/> and other similar data generatorsto generate random data to populate the database. This website shall be used to generate ID’s, names, addresses, numeric

# SOME ISSUES FACED AND LESSONS LEARNT

* Foreign key of a particular table was not properly referenced. Foreign key should be having same data type as the parent table.

For example: In Students table and Applications table, with Student\_ID and Application\_ID has Student ID as Foreign key. Student\_ID is data type ‘Int’. This Application\_ID has ‘TinyInt’ as data type but since type is different, it will not allow you to populate DB.

* While creating schema and ER Diagram, University name had to have appropriate Varchar memory or character limit.

**CRUD OPERATIONS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **View Name** | **Requirement A** | **Requirement B** | **Requirement C** | **Requirement D** | **Requirement E** |
| number\_students\_degree | ✓ | ✓ | ✓ | ✗ | ✗ |
| number\_students\_university | ✓ | ✓ | ✓ | ✗ | ✗ |
| program\_number\_enrolled | ✓ | ✓ | ✓ | ✗ | ✗ |
| reference\_details\_for\_students | ✓ | ✓ | ✗ | ✗ | ✓ |
| student\_university\_attending | ✓ | ✓ | ✗ | ✗ | ✗ |
| top\_3\_toefl | ✓ | ✓ | ✓ | ✗ | ✗ |
| top\_3\_gre | ✓ | ✓ | ✓ | ✗ | ✗ |
| top\_3\_verbal\_scores | ✓ | ✓ | ✗ | ✗ | ✗ |
| visa\_count\_location | ✗ | ✓ | ✓ | ✗ | ✗ |
| visa\_student\_date | ✓ | ✓ | ✗ | ✗ | ✗ |
| Average\_reference\_rating\_for\_students | ✓ | ✓ | ✓ | ✗ | ✗ |
| DEGREE\_CORE\_COURSES | ✓ | ✗ | ✓ | ✓ | ✗ |
| DEGREE\_DURATION | ✓ | ✗ | ✓ | ✓ | ✗ |

# Query 1: Select u.University\_name, concat(s.first\_name, space(1), s.last\_name) as Student\_Name, s.Contact\_Email as Email\_ID,

CASE a.Admitted

WHEN 0 THEN 'Rejected'

WHEN 1 THEN 'Accepted'

END

as Admitted\_Status,

round(avg(ar.Rating),2) as Average\_Rating

from applications a, applications\_has\_references ar, students s, universities u, programs p, degrees d

where a.Application\_ID = ar.Application\_ID

and s.S\_ID = a.Student\_ID

and u.University\_ID = p.University\_ID

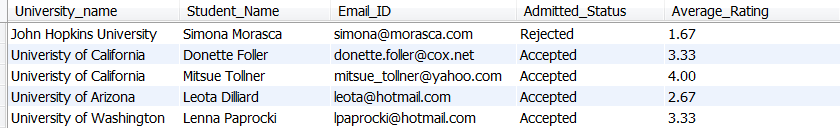
and p.Program\_ID = d.Programs\_ID

and d.Degree\_ID = a.Degrees\_Degree\_ID

group by a.admitted, a.Application\_ID

order by u.University\_name;

**Output:**



All the queries are written in the same way as “Views” and stored in the database.