

Institutional Research Database Study

The Office of Institutional Research uses data provided by Administrative Computing to perform reporting requirements to SCHEV and other state government agencies. The data that is used is provided in a flat file format and imported into Microsoft Access for query and reporting purposes.

Objective:

To determine if changes to the database design would provide enough benefit to justify any cost incurred in implementing the design.

Possible Benefits:

- Storage space savings
- Increased speed of query execution
- Improved reporting.

Possible Cost:

- Monetary costs in hardware or software
- Labor costs in database redesign and conversion
- Labor costs in report and query creation

Current Status

We understand the current situation in the Office of Institutional Research to be as follows:

1. The Office of Institutional Research periodically requests certain types of data from the Administrative Computing center. This data is provided in the form of a *flat file*.
2. This data is used by the Office of Institutional Research to provide reporting information to SCHEV and to other organizational units of the university.
3. The Office of Institutional Research will review the contents of this data file to find and correct certain types of data errors. These errors are corrected on the data used by the Office of Institutional Research for purposes of reporting but they are not necessarily corrected in the data kept by the administrative computing center.
4. The Office of Institutional Research will periodically receive new (refreshed) copies of this data.
5. The data provided by the Office of Institutional Research to SCHEV is provided in a flat file format to be consistent with formats required by SPSS or SAS.

Background Concepts

There are two distinctly different design alternatives that need to be considered. The first alternative deals with a *database* while the second alternative deals with a *data warehouse*.

Database: Databases are used in an environment where there will be a large number of users sharing the same data. These users will typically be making changes to the data in the form of insertions of new data, deletions of existing data, and modification of existing data. Since this data is normally very volatile (changes frequently) the design approach is to normalize the tables to remove anomalies that occur during updating. When tables are normalized, duplicated data is removed and larger tables are broken into several smaller

tables. Relationships are established between the smaller tables. These relationships allow you to rebuild the larger tables in a virtual form without duplicating the data.

The first advantage of the database approach is the reduction in the amount of data stored, which should result in less file space required. The second advantage comes in updating the data. Since there is very little data duplication, updating of the data is simplified.

Disadvantages of the database approach occur in its processing speeds. Since the smaller normalized tables must occasionally be rebuilt (virtually) to form the original larger table, many queries will require more execution time. To decrease the execution time, indexes can be built. The indexes require storage space thus canceling some of the advantages we get in storage space savings.

The database approach is best used when an organization uses the data for its normal transaction processing systems or the systems that are required to perform the organization's day-to-day business.

Data Warehouse: The data warehousing approach is used in an environment where the data is very static (seldom changes). In a data warehouse, most tables are not normalized, thus they store a large amount of duplicated data.

This duplicated data carries the expense of extra storage space thus making a warehouse much larger than a database. The total amount of additional space required depends upon the amount of duplicated data being carried.

Since the data is rarely updated we do not encounter the anomalies that occur when updating duplicated data. Since tables are not normalized, they do not need to be rejoined. This will normally allow queries on a data warehouse to execute a query faster than a query made on a database.

Currently, the databases used by the Office of Institutional Research are organized in a manner that more closely resembles a data warehouse. Based upon the way the Office of Institutional Research uses the data, this would appear to be the proper organizational approach.

Process

To determine if changes to the approach used by the OIR would provide any benefit we decided to take a sample database in its current form and normalize the table structures to at least Third Normal Form. This would provide a structure that more closely resembles a database. Once we have the sample database in the two different formats we can compare the two in terms of speed and efficiency. We would expect to see the following efficiencies by normalizing the tables:

1. File space savings: Since normalization will reduce the amount of duplicated data that is held in a database, it is expected that the total amount of space required by a normalized database will be less than that required by a database that is not normalized.
2. Reduction of anomalies: Anomalies are errors or inconsistencies that can occur when making changes to the data within the database. When a database is properly normalized, the probability of an anomaly existing is greatly reduced.
3. Decreased time required to execute certain types of queries: Since some of the data will reside in smaller tables, certain types of queries will execute faster. It should also be true

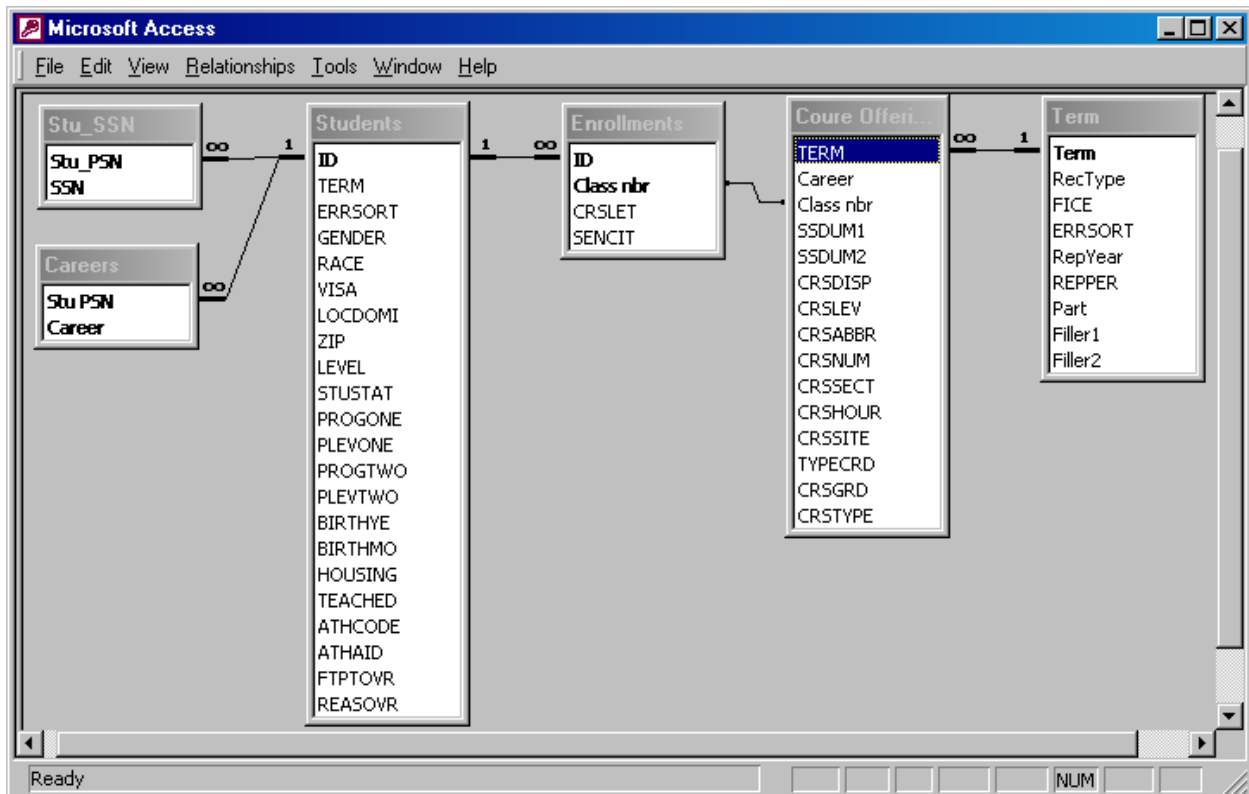
that some queries would execute slower since tables may need to be joined before a query can be executed.

Problem areas and other considerations

1. In the current database, there are a number of students that appear as two different students. This may happen when a student has multiple career indicators. It also happens when students have multiple IDs. This happens primarily with international students and occurs because of the way identifiers are assigned.
2. SCHEV uses Social Security numbers for identifiers. JMU uses PeopleSoft Numbers.
3. Periodically the OIR will receive refreshed copies of the data from Administrative computing. This data is in a flat file or un-normalized format. This would require that the refreshed copies be normalized each time they are received.
4. The data sent to SCHEV must be provided in a flat file format compatible for SPSS and SAS. If normalized tables are used these tables would need to be un-normalized when sent to SCHEV.

Database Design

The sample database was examined for data duplication and normalized based on several logical considerations. The following represents the design of the normalized database.



1. Social Security Numbers were extracted from the student table. This allows a student with multiple Social Security numbers to be represented only once. This change saved minimal space. It eliminated the two social security columns in the student tables but created a new table with two columns.
2. Careers were extracted from the student table. This allows students with multiple careers to be represented only once. This change also saved minimal space. It eliminated four records from the student table, reducing it from 14,179 students to 14,175 students.
3. Student ID numbers were removed from the course offerings table and a relationship was established between the student data and the course offerings table. By having the student ID in the course offerings table, all course data needed to be duplicated for each student enrolled in a course. This change created the greatest amount of space savings. The Course Offering table was reduced from 68,910 rows to 4,645 rows. This change reduced the size of the database by almost 50%.

There is a problem creating a course identifier when the table is normalized this way. Since some courses are listed with different career paths the course number is not unique. Also, sometimes a course is listed with multiple CRSGRD values. If the database were to be normalized there would need to be a better technique for determining a course offering identifier.

4. Term data was removed from the course offerings table and a relationship was established. This eliminated some duplicated data and columns from the course offering table. The space savings was insignificant with this change.

Comparisons of file sizes for the different databases are shown below.

| | Un-normalized Database | Normalized without Indexes | Normalized with indexes |
|------------------|-------------------------------|-----------------------------------|--------------------------------|
| File Size | 21.5 Megabytes | 8 Megabytes | 11.4 Megabytes |

The next step was to compare the time it takes to execute a query using the different designs. Several different queries were run and there were no significant differences in the time required to execute the queries using the different designs.

Results

The size of the normalized database was reduced by approximately 50%.
The speed of query execution for most queries was not measurable.

Recommendations

1. Continue using the same processes you are currently using. This recommendation is based upon the following conclusions:
 - a. The data provided by Administrative Computing is in the format you are currently using. By continuing with this format, there is no additional time and labor required to restructure the database tables. Additionally, fresh copies of the data are provided periodically. These would also need to be restructured.

- b. The data that must be provided to SCHEV is in a flat file format similar to the data you are currently using. This should make it simpler to provide data to SCHEV.
 - c. The majority of the savings in a redesign would be realized in space savings. Assuming that sample database is representative of other years' databases, the savings amounts to approximately 10 megabytes per year. This amounts to approximately 500 megabytes over a period of 50 years. In today's terms, we would argue that this saving is not worth the cost of the redesign.
 - d. The current limit on a Microsoft Access database is 2 gigabytes. The sample database, un-normalized, is only 22 megabytes. This is less than 2 percent of the maximum allowable by Microsoft Access. Even when multiple years are combined, the database does not approach the maximum allowed.
2. Keep archive copies of previous years compressed in ZIP files. The sample database, uncompressed, is 21.5 megabytes. Compressed, the database occupies 2.68 megabytes. This compression reduces the storage requirements by almost 90%.
3. The regular use of a database requires both considerable processing power and quick disk access to be efficient. We recommend that staff members who execute database queries regularly be equipped with new, powerful computers on a regular basis to maintain productivity. As the processing demands increase, hardware capabilities must be increased concomitantly.