

## Database Case 5

### Chapman University

Problem: Develop an admissions database

Management Skills: Coordinate  
Decide

Access Skills: Queries  
Finding Records  
Sorting and Grouping  
Reporting

Data Table: CHAPMAN

The Graduate School of Business (GSB) at Chapman University began operations the mid-1960s. Since that time it has grown to become one of the most competitive and well respected schools of management in the United States. Chapman has particularly strong faculty in the Marketing and Information Systems areas.

Andrew Ulrich has been Dean of Admissions at the GSB for over thirty years and he feels it is time for a change. The business of running the admissions program has become so politicized, argumentative, and chaotic that he is seriously considering resignation. Before he resigns he has promised to himself and his boss Tom Davenport, the President of Chapman, that he would build a new admissions system. He could use your help.

The Office of Admissions gathers applications and supporting documentation from a number of sources. Each student submits a completed application form and an essay. Faculty who have worked with the students at their undergraduate institution submit references. The undergraduate university also submits transcript verification forms. In addition, the College Board submits data on GMAT scores. Miscellaneous information on student ethnicity, age and sex is also collected to meet various legal requirements and reporting standards.

The Alumni group has made it very clear that they want admissions to treat the children (though not other more distant relatives) of alumni with special care.

All of this information is collected by the Admissions Office staff and summarized on an Admissions Sheet. The Admissions Sheet is the first piece of paper in the bulging folders kept on each student. The essays and outside faculty references are graded on a 1-5 scale (with 5 being the highest score) by faculty committees.

This procedure was developed in the mid 1970's when the school had 600 or so applicants for 200 positions. Today the school has 5,000 applicants for about 400 positions. Although the Admissions Office (AO) staff has doubled in size (to 8 full time people), the workload is still heavy, and the process is breaking down as older, experienced personnel retire.

The procedure for processing the Admissions Sheet is one source of the problems. Because the data arrive from different sources at different times, the AO clerks have to go back to each student folder many times to enter and update the information.

Students, parents, faculty, institutions, and alumni call in frequently to see if certain information has been received. Sometimes clerks pull the sheet to answer queries and then the sheet is lost. New sheets have to be coded up in this case.

A number of data quality problems have emerged. For instance, more and more students are claiming that their GMAT scores are wrongly recorded. Changes from the University Board are submitted with growing frequency, but AO staff is often so pressed for time that sometimes these record changes are not made. Other data quality problems involve the undergraduate grade point average, address errors, and majors. Students move, change majors, and their GPAs change. The current system should be able to keep track of these changes.

The existing system does not appear to support the group decision-making process, which is at the very heart of admissions. A small group of ten faculty and three Admissions Officers make all the decisions on the Admissions Committee. Each member of the Admissions Committee takes a different cut at the data, wants the data organized differently, and feels frustrated when this is impossible. One result is that members of the committee spend too much time arguing over the decision criteria and weightings, and too little time searching for appropriate candidates.

For instance, some faculty are interested only in GPA and GMAT scores and want to see the applicant pool sorted first by GPA, and then by GMAT. Other faculty are more oriented towards GPA and references and do not even want to see the GMAT.

Administrators want to make sure that within accepted academic criteria, the alumni's children get a special hearing. There is little opportunity to find those candidates who do well on all the criteria.

The manual resorting of lists and list compilation is a very tedious process characterized by long delays, mistakes, and glitches.

All of these problems have produced a political dimension: no one is happy with the existing system and everyone blames the Admissions Office. As Dean of Admissions, Andrew Ulrich is now under attack from many sides. The AO staff are also unsatisfied, even a bit surly at times, because of the long hours they must spend each Spring compiling lists, up-dating files, and meeting last minute deadlines.

Rather than rely on the Administrative Computing Center (which is already extended beyond capacity), Andrew has decided to build an PC-based admissions database.

A sample of the admissions database (CHAPMAN) has been created for you on SOLVEIT.MDB as an Access table object. Create a new Access database and import the CHAPMAN table now.

**Tasks:** There are six tasks in this case:

1. There are two typical faculty requests which the system must provide: a list of applicants sorted by GPA and GMAT scores, and a different list for other professors who want the

---

students listed by GPA and references. Only include the essential fields in each report. For example, for the report sorted by GPA and GMAT scores, the professors only really need the student's name, school and scores. Create queries for the following:

(a) A listing of all the applicants for professors on the admissions committee who want to see the applicants listed by decreasing GPA scores. Students with the same GPA should be sorted by decreasing GMAT scores.

(b) A second listing showing the applicants sorted by decreasing GPA. Students with the same GPA should be sorted by decreasing reference scores. Now Print out the results of both listings.

2. The AO clerks are absolutely insistent on having rapid access to students' complete records on the basis of Last Name only. That is, they want to type the last name of the student into a PC and have the complete record pop up on the screen.

Because so many people call in checking on student applicant files, this capability would save an enormous amount of time. Using the CHAPMAN table, identify the Access sequence of commands that would enable the AO staff to do this.

3. Ulrich would like a report or listing of just the applicants who are children of alumni. He will use this list as a crib sheet in committee deliberations. Whenever a candidate is settled on, he will check his list of alumni applicants to see if that person is on the list. If not, he will suggest an equally well-qualified alumni applicant. Thus the alumni applicants should be sorted by decreasing GPA. Students with the same GPA should be sorted by decreasing GMAT scores.

It is advisable given that 5,000 applicants will be in the final database to make two lists here. One list is alphabetical permitting quick look-ups; the second list would arrange students by GPA and GMAT. (a) Create a query to display this information, and (b) print out the results at report level.

4. A small group of professors are concerned that the business school is loading up with Science majors. They would like to ensure that students from Liberal Arts and Social Science backgrounds are considered as well. The non-science students tend to have better GPA scores than GMAT scores (because of an alleged quantitative bias in the GMAT test) and they tend to have better recommendations.

This group of faculty would like a report, which plays to the strengths of the non-science majors. Create a report, which lists applicants by college major, showing name, GPA, and reference scores. Students of the same major should be sorted by GPA.

- \*5. There is an on-going dispute amongst Admissions Committee members about the comparative strengths of the Social Science (Business and Psychology), Liberal Arts (English and History), and Science majors (Math, Physics and Engineering). Filter out these three groups and calculate their average scores on all quantitative variables. Then compare the groups in a one-paragraph statement.

- \*6. There are number of ways this database system could be improved to better meet the needs of Chapman. List the ways the system could be improved. Pick one of these improvements and implement it in the database.

**Time Estimates (excluding tasks marked with \*):**

Expert: 1 hour

Intermediate: 1.5 hours

Novice: 3.5 hours

---

---

**Tutorial For Database Case 5 Using Access 2000****Indexing in Access**

If you often search a table, or frequently need to sort records by a certain field, you can speed up these operations by assigning indexes to your fields. Access uses indexes in a table in the same way as you would use an index in a book: to find data, it looks up the location of the data in the index.

In Access, you create an index on a single field by setting the Index property. The table below lists the possible settings for Indexed properties.

<b>Index property setting</b>	<b>Definition</b>
<b>No</b>	Do not create an index for this field (the default)
<b>Yes (duplicates OK)</b>	Create an index for this field
<b>Yes (no duplicates)</b>	Create a unique index for this field

If you create a unique index, Access will not allow a value that already exists in that field, to be entered in the same field for another record. Primary key fields (refer Tutorial in Chapter 4) are automatically indexed by Access, to help speed up the execution of queries and other operations. Up to 32 indexes can be assigned to a single table or query; each index can contain up to 10 fields. Fields with Memo or Counter data types cannot be indexed.

You want an index only if it speeds up the execution of queries, searching or sorting. A Last Name field is a good candidate for an index since the values stored in it vary greatly, and it is often used to find specific records.



1. Load the practice database FRIENDS.MDB. From the FRIENDS Database Window, click on the Tables object, and double click on the FRIENDS table. Switch to table design view by clicking the *Design View* toolbar button, or selecting VIEW/TABLE DESIGN from the menu. 
2. Add a new field before the TITLE field. The name for the new field will be "ID", and the data type will be AutoNumber. Right-click the ID field and select Primary Key from the menu.
2. In the FieldName column, click on the LastName field, and then click on the Indexed bar in the Field Properties box. Select option *Yes (Duplicates OK)* to set the index for the LastName field. Click on the Save toolbar button or select FILE/SAVE TABLE from the menu. Your table design window should look similar to Figure 5-59.

Figure 5-59

FRIENDS : Table			
	Field Name	Data Type	Description
	ID	AutoNumber	
	TITLE	Text	
	LAST_NAME	Text	
	FIRST_NAME	Text	
	STREET	Text	
	CITY	Text	
	STATE	Text	
	ZIP	Text	
	PHONE	Text	

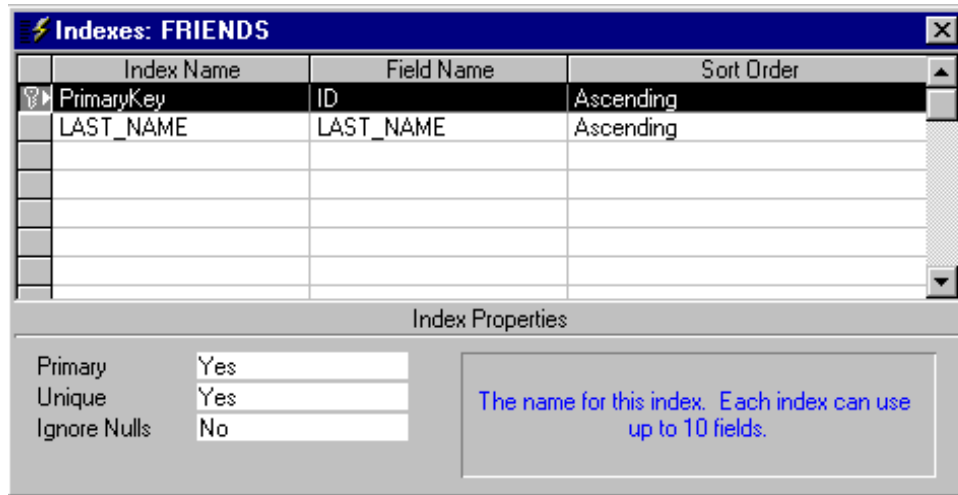
Field Properties	
General	Lookup
Format	
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Allow Zero Length	No
Unicode Compression	Yes

A field name can be up to 64 characters long including spaces. Press F1 for help on field names.

3. To view and/or edit existing indexes, open the Indexes window by clicking on the Indexes toolbar button or selecting VIEW/INDEXES from the menu (see Figure 5.60).



Figure 5-60



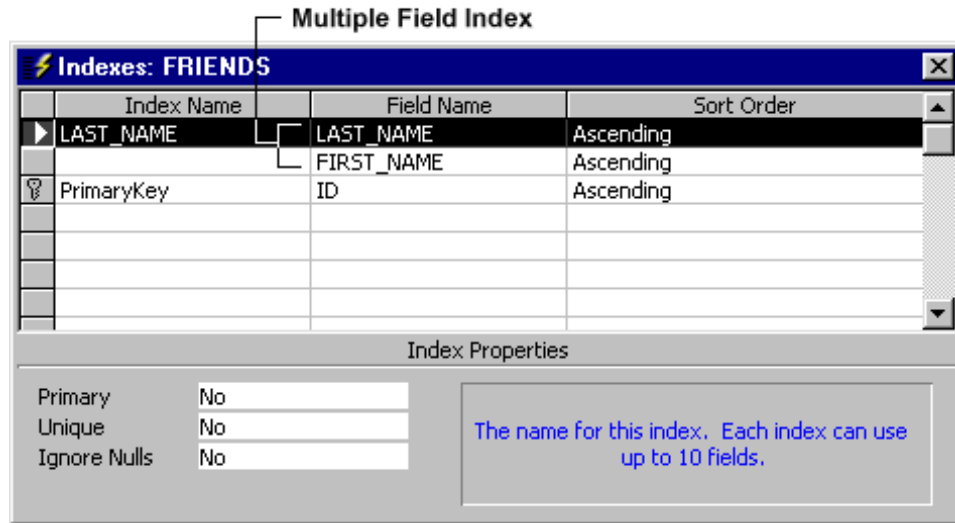
4. Change indexes or index properties as needed. To delete an index, highlight the intended row in the Indexes window and press the *Delete* key. This removes only the index, not the field itself. For Help on index properties, press F1 from the indexes window.
5. Close the Indexes window by clicking on the box in the top right hand corner.

### Creating Multiple-Field Indexes

If you often search or sort by two or more specific fields at the same time, you may need to create a multiple field index. For instance, if you often set criteria in the same query for LastName and FirstName fields, it makes sense to index on both fields. When you sort a table by a multiple field index, Access sorts firstly by the first field listed in the Indexes window. If there are records with duplicate values in this field, Access then sorts by the second field listed.

Multiple field indexes are created in the Indexes Window by including a row for each field in the index, but including the index name in the first row only (see Figure 5-61). Access treats all rows as part of the same index until it reaches a different Index Name. To insert a new row between existing indexes, click the row *below* the location of the row to be inserted. Then press the *Insert* key.

Figure 5-61



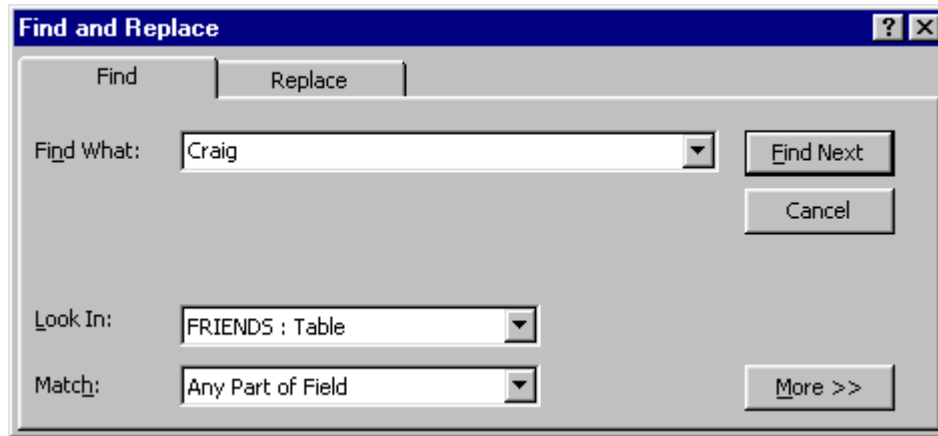
Save your changes, and press F11 to return to the Database Window.

### Forward and Backward Searching in Access

When you want to find a specific record or certain values within fields, you can use the *Find* command to go directly to a record. Find can be used with most Access objects. To use Find, select EDIT/FIND from the menu. This displays the Find and Replace dialog box (see Figure 5-62). The Find and Replace dialog box enables you to search and replace text within a field.

Type the wanted text into the Find What bar. For added flexibility, *Find* enables you to enter word stems, and wildcard symbols. Similar to DOS, a question mark (?) stands for any single character in the same position as the question mark. The asterisk (\*) stands for any number of characters in the same position as the asterisk. The hash (#) stands for a single numeric digit in the same position as the hash.

Choose between searching across fields or a single field, and whether to search forwards or the whole file. Practice using *Find* with your FRIENDS table.

**Figure 5-62**

## Tutorial For Database Case 5 Using Access 97

### Indexing in Access

If you often search a table, or frequently need to sort records by a certain field, you can speed up these operations by assigning indexes to your fields. Access uses indexes in a table in the same way as you would use an index in a book: to find data, it looks up the location of the data in the index.

In Access, you create an index on a single field by setting the Index property. The table below lists the possible settings for Indexed properties.

Index property setting	Definition
No	Do not create an index for this field (the default)
Yes (duplicates OK)	Create an index for this field
Yes (no duplicates)	Create a unique index for this field

If you create a unique index, Access will not allow a value that already exists in that field, to be entered in the same field for another record. Primary key fields (refer Tutorial in Chapter 4) are automatically indexed by Access, to help speed up the execution of queries and other operations. Up to 32 indexes can be assigned to a single table or query; each index can contain up to 10 fields. Fields with Memo or Counter data types cannot be indexed.

You want an index only if it speeds up the execution of queries, searching or sorting. A Last Name field is a good candidate for an index since the values stored in it vary greatly, and it is often used to find specific records.



1. Load the practice database FRIENDS.MDB. From the FRIENDS Database Window, click on the Table object, and double click on the FRIENDS table. Switch to table design view by clicking the *Design View* toolbar button, or selecting VIEW/TABLE DESIGN from the menu.



2. In the FieldName column, click on the LastName field, and then click on the Indexed bar in the Field Properties box. Select option *Yes (Duplicates OK)* to set the index for the LastName field. Click on the Save toolbar button or select FILE/SAVE TABLE from the menu. Your table design window should look similar to Figure 5-63.

Figure 5-63

FRIENDS - Table			
	Field Name	Data Type	Descriptor
	ID	AutoNumber	
	TITLE	text	
	LAST_NAME	text	
	FIRST_NAME	text	
	STREET	text	
	CITY	text	
	STATE	text	
	ZIP	text	
	PHONE	text	
	PROFESSION	Yes/No	
	PERSONAL	Yes/No	

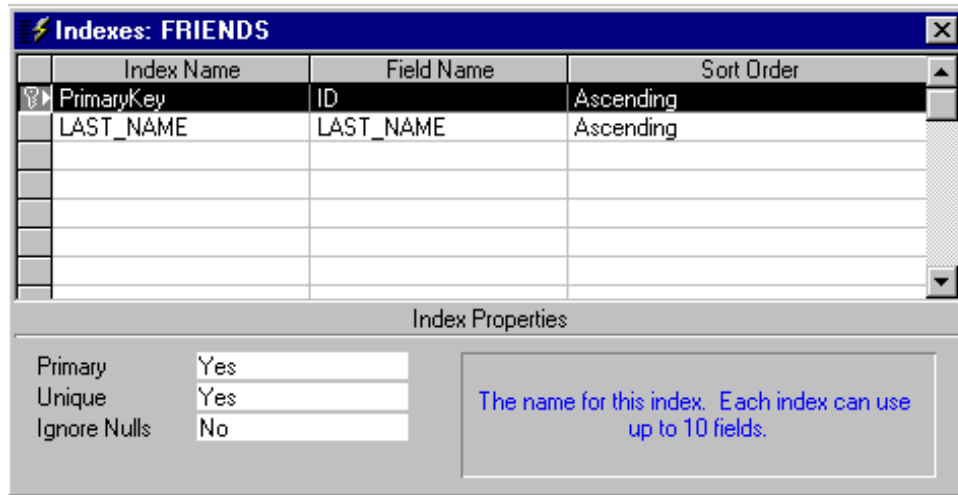
  

Field Properties	
General	Lookup
Field Size	255
Format	
Input Mask	
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Allow Zero Length	No
Indexed	Yes (Duplicates OK)

An index speeds up searches and sorting on the field, but updates. Selecting "Yes - No Duplicates" prohibits duplicates in the field. Press F1 for help on indexed fields.

3. To view and/or edit existing indexes, open the Indexes window by clicking on the Indexes toolbar button or selecting VIEW/INDEXES from the menu (see Figure 5.64).



**Figure 5-64**

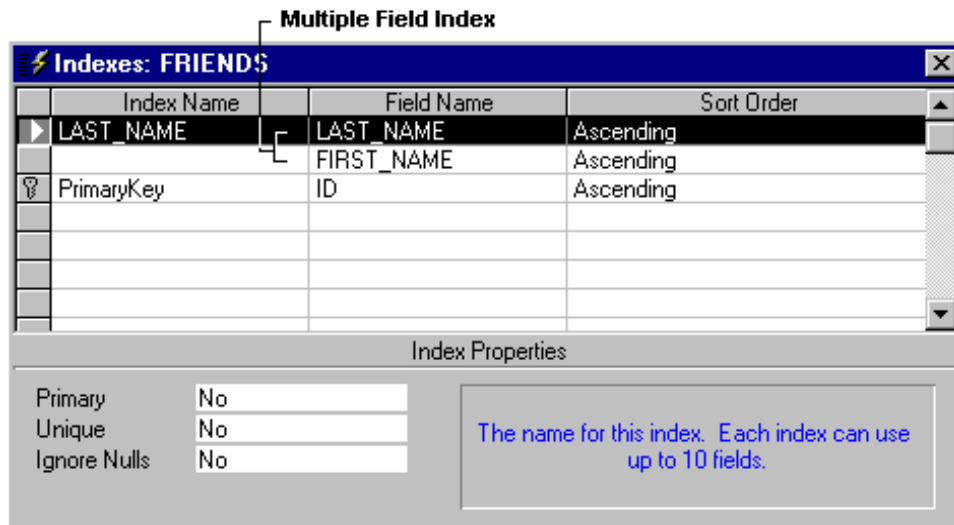
4. Change indexes or index properties as needed. To delete an index, highlight the intended row in the Indexes window and press the *Delete* key. This removes only the index, not the field itself. For Help on index properties, press F1 from the indexes window.
5. Close the Indexes window by clicking on the box in the top right hand corner.

### Creating Multiple-Field Indexes

If you often search or sort by two or more specific fields at the same time, you may need to create a multiple field index. For instance, if you often set criteria in the same query for LastName and FirstName fields, it makes sense to index on both fields. When you sort a table by a multiple field index, Access sorts firstly by the first field listed in the Indexes window. If there are records with duplicate values in this field, Access then sorts by the second field listed.

Multiple field indexes are created in the Indexes Window by including a row for each field in the index, but including the index name in the first row only (see Figure 5-65). Access treats all rows as part of the same index until it reaches a different Index Name. To insert a new row between existing indexes, click the row *below* the location of the row to be inserted. Then press the *Insert* key.

Figure 5-65



Save your changes, and press F11 to return to the Database Window.

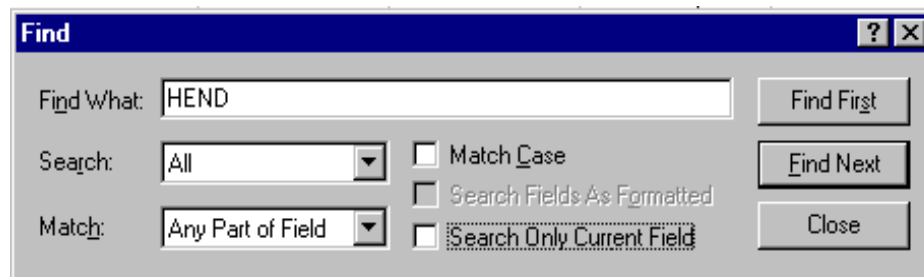
### Forward and Backward Searching in Access

When you want to find a specific record or certain values within fields, you can use the *Find* command to go directly to a record. Find can be used with most Access objects. To use Find, select EDIT/FIND from the menu. This displays the Find dialog box (see Figure 5-66).

Type the wanted text into the Find What bar. For added flexibility, *Find* enables you to enter word stems, and wildcard symbols. Similar to DOS, a question mark (?) stands for any single character in the same position as the question mark. The asterisk (\*) stands for any number of characters in the same position as the asterisk. The hash (#) stands for a single numeric digit in the same position as the hash.

Choose between searching across fields or a single field, and whether to search backwards, forwards, or the whole file. Practice using *Find* with your FRIENDS table.

Figure 5-66



Also explore the Access *Replace* command (select EDIT/REPLACE from the menu), which enables you to search and replace text within a field.