This is a list of topics that we have covered since midterm exam 2. The final will be cumulative, that is, it will cover material from the entire quarter, but it will focus primarily on the last third during which we discussed the topics covered here. To be fully prepared you should go back and read the previous two review sheets (linked from the calendar page) and be sure you are comfortable with that material too.

This review is not all inclusive of every detail and there may be items on the exam that are not explicitly listed here, but these are the primary topics of interest from the last third of the quarter.

Numbers and Information

Computers represent information with numbers. The same numeric value can represent many different pieces of information depending on the context.

There are numerous standards in use for representing common quantities. Text characters are often represented by relatively small numbers (0 to 255) with the ASCII encoding. However, this scheme cannot be used to represent all the languages that people use and so other schemes have been invented, including Unicode that uses a larger range of numbers to represent more characters.

Colors are another quantity that is often represented numerically. In a simple monochromatic system (black, gray, white) the brightness of a particular pixel (picture element) is often represented with a single value in the range 0 to 255. In a more complex system, colors are usually represented using three brightness values, one each for Red, Green, and Blue, with each value selected from the range 0 to 255.

Any information that can be characterized numerically (by "digitizing" it, ie, converting it to discrete values) can be represented in the memory of a computer. This is the basis of image and audio processing. As long as the program that stored the data and the program that reads the data agree on the organization of the numbers and what they represent, we can store whatever we want.

One of the fundamental ideas associated with numbers is the concept of positional notation. For a particular "number base", each position in a number (digit) can represent as many different values as the base. In other words, for a base 2 number, each position can represent one of two values. For a base 10 number, each position can represent one of ten different values. And for a base 16 number, each position can represent one of 16 different values. Consequently, in order to represent numbers in base two, only two different symbols are needed. Traditionally the symbols used are 0 and 1, although any two distinct symbols will suffice. In order to represent numbers in base 10, we traditionally use 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. For base 16, we use the same 0 through 9, then add A, B, C, D, E, and F to represent 10_{10}, 11_{10}, 12_{10}, 13_{10}, 14_{10}, and 15_{10}. Each column in a numeric value represents one more multiplication by the base value as the number of columns increases.

Inside the computer itself, numeric values are generally represented in base 2 binary numbers, since most computers operate on binary signals (either on or off, true or false, high or low, etc). However, a value is the same no matter what base it is expressed in. So, for example, if a computer stores the number 00100011_2 in its memory, we can say that it is storing 23_{16} or 35_{10} without contradiction or loss of information. (By the way, this is the ASCII code for the number sign character "#".)

You should be able to convert a small binary (base 2) number or hexadecimal (base 16) number to base 10, and also be able to convert easily between base 2 and base 16 directly. Remember that four binary positions (bits) convert directly to one hexadecimal digit, and vice versa.

Flat File Database

Know that the term "flat file database" describes a very simple database model, where all the information is stored in plain text files, one database record per line. Each record is divided into fields using delimiters or at fixed column positions. The data is "flat", as in a sheet of paper, as compared to a more complex model such as a relational database.

Flat file databases are very handy when every record contains the same attributes with simple values. This is true of simple tables such as a list of names and phone numbers or tables of scientific data with one key attribute and one or more simple descriptive attributes such as the example of chromosome data from UW-FHCRC (http://pga.gs.washington.edu) that I gave in class.

The main problem with flat file databases is that they don't handle redundancy very well. If one or more of the entities in the database have the same value for an attribute, and that value is a complex value like "publisher information" or "course information", then the complex value must be repeated for each entity in the database. When the same information is repeated, it is very likely that it will contain small (or large) differences each time it is entered, and so the database will be inconsistent. Inconsistent data is a bad thing!

Relational Database

One solution to the redundancy issue is the relational database. In this design, the data is split apart into tables of basic information. Tables store information about entities. Entities have characteristics called attributes. Each row in a table represents a single entity: each row is a set of attribute values and every row must be unique, identified by a primary key (an attribute or set of attributes whose values are unique for every row in the table). Relationships among the attributes in the tables are stored and can be used to reconstitute the original data. Tables have names, attributes, and rows. Each table represents a set of entities. The schema for a table describes the name and type of each attribute in the table.

There is an extensive formal definition of relational algebra that is the basis for building queries that extract and combine information from tables. The operations that form the basis of our work are select (pick rows of a table based on some criterion like age<18), project (pick columns of a table), product (produce a new table containing all combinations of two original tables), union (produce a new table by combining two tables with the same attributes), and difference (produce a new table by removing the entities contained in one table from another table with the same attributes).

In actual practice, we use a database management system (DBMS) to manipulate our database. The DBMS defines additional operations based on the basic algebra. An important operator is the join operator that combines rows from two tables if the value of a common attribute matches.

Generally, each table will have an attribute or set of attributes that is unique for every possible row in the table. This attribute set is identified as the primary key. If the table has a relationship with another table, then the attribute (or set of attributes) that identifies the associated record in the other table is called the foreign key. The possible values of the foreign key in one table are chosen from the values of the primary key in the other table. The join operator matches the value of the foreign key given in one table with the value of the primary key in the other table in order to find the rows that should be combined.

An entity-relationship diagram is often used to describe the logical architecture of a database. Each entity name is shown, representing the table that contains one or more instances of that entity type. The attributes of each entity are shown. In the tables, these attributes will be columns of the tables. Relationships are shown as connecting lines between entities, with a description of the relationship shown in a diamond. In the tables, these relationships will be implemented as attributes (columns) that contain common values in the two related tables.

Relationships can express several different types of connections. In a one-to-one relationship, an entity in one table is associated with one and only one entity in another table. For example, this sort of relationship might be used to associate a student with a transcript: each student has one transcript, and a particular transcript is associated with only one student.

In a one-to-many relationship, an entity in one table can be associated with one or more entities in another table, but each entity in the second table is only associated with one entity in the first table. For example, this sort of relationship might be used to associate a table of artists with a table of artworks: each art work has one artist that created it, but an artist may have created more than one artwork.

In a many-to-many relationship, an entity in either table can be associated with one or more entities in the other table. For example, this sort of relationship might be used to associate a table of students with a table of classes: a particular student is probably taking several classes, and a particular class includes several students. A many-to-many relationship is implemented using an intermediate cross-reference table that allows the DBMS to look up all occurrences of a key from one table (eg, the student id) and find all occurrences of the foreign key for the second table (eg, the course id). This would implement the command "show all courses that this student is taking." Similarly, the DBMS can look up all occurrences of the course id and find all occurrences of the student id. This would implement the command "show all students in this course."

The actual work of getting data out of a database is done with queries. Generally speaking, a query will involve a join of two or more tables to produce a larger "virtual" table, then a selection to pick out particular rows of the table, and a projection to pick out particular columns of the table.

Access Database

The database program that we used is Access. You should be able to read screen shots of Access windows and answer questions about the tables and queries that are shown. You should be familiar with design view and datasheet view as they apply to tables. You should be familiar with design view (or Query by Example), datasheet view, and SQL view as they apply to queries. You should be familiar with the layout and meaning of the entries in the relationships tool window. You should be familiar with the idea that forms and reports are ways to organize and present data from the database tables and queries.

The actual command language that is used behind the scenes to implement queries is the Structured Query Language (SQL). We did not study SQL explicitly in any detail and I will not ask you to write any SQL from scratch. However, you should know that it exists and its relationship to the Query By Example windows that we use in Access to build the queries. Given two views of a simple query (the design view and the SQL view), you should be able to map between parts of the SQL statement and items in a screen shot of the associated Query By Example.

Privacy

With the growing speed and reach of the modern publishing technology (ie, the Internet and the Web) it is becoming more and more difficult to control where our information goes. There are obvious issues of privacy involving limits on publishing personal records such as school records, medical information, and financial information. There are less obvious issues associated with publishing seemingly smaller bits of information about us such as photos of people in public places, live video of people at public events, archived text and images that can be searched years after the event to which they are related. More recently, there are issues associated with being able to correlate a set of very innocuous (ie, seemingly unimportant) facts with each other to be able to derive significant facts about a person and their activities.

I gave examples in class of using third party cookies as a common attribute to join (in the database sense) two or more separate sets of information about a customer and thus derive extensive knowledge about their web browsing activities. I also gave the example of using “preferred shopper” cards to build a purchasing profile of a customer and then further correlating buying activities with any other information that the company can acquire about the customer. This correlation can be used to provide useful information to the customer; it can also be used to provide startling invasions of privacy and to limit the options that a person has for financial services, medical assistance, and other services.

There are efforts under way to define and enforce so-called Fair Information Practices. However, there are many competing interests involved, and significant amounts of money can be made if the practices are not very restrictive, and so progress has been slow. The Organization of Economic Cooperation and Development (OECD) codified an expanded set of principles related to privacy. These principles are followed more closely in some nations than others. The “correct” interpretation of these principles and enforcement are difficult challenges as there are always competing interests pushing us towards more or less restrictions on data gathering and analysis.