COP 4710: Database Systems
Spring 2004

-Day 2 – January 7, 2004 –
Introduction to Database Systems

Instructor : Mark Llewellyn
markl@cs.ucf.edu
CC1 211, 823-2790
http://www.cs.ucf.edu/courses/cop4710/spr2004

School of Electrical Engineering and Computer Science
University of Central Florida

What is a Database?

- In the most general sense a database is simply a collection of related data.
  - This definition is too vague since we could consider this page of words to be a database under this definition.
- Note that the "data" in a database can encompass a wide variety of objects from numbers, text, graphics, video, etc.
- A more specific definition of a database consists of certain implicit characteristics which, when considered together, are assumed to define a database.

What is a Database? (cont.)

- A database represents some aspect of the real world. This abstraction of the real world is often referred to as the miniworld or the universe of discourse (UoD).
- A database is a logically coherent collection of data with some inherent meaning. Random data is not typically referred to as a database, although their are exceptions.
- A database is designed, built, populated, and utilized for some specific purpose. There is a set of intended users and specific applications in mind.

What is a Database? (cont.)

- A database is managed by a database management system (DBMS), typically referred to as a *database system*.
- A DBMS is expected to provide significant functionality including:

1. Allowing users to create new databases. This is done via data definition languages (DDLs).
2. Allow users to query the database via data manipulation languages (DMLs).
3. Support the storage of very large amounts of data. Typically gigabytes or more for very long periods of time. Maintaining its security and integrity in the process.
4. Control access to data from many users simultaneously.

Early Database Systems

- The first commercial database systems appeared in the late 1960's. They evolved from file systems which provide some of item (3) from the previous slide, however, they provide little or nothing from item (4).
- Furthermore, file systems do not provide direct support for the features of item (2), i.e., they don't support query languages per se.
- Neither do file systems directly support item (1), their support for schemas is limited to the creation of directory structures for files.
- Some of the more important early database systems were ones where the data was composed of many small items and many queries or modifications were made. Examples: airline reservation systems and banking systems.

Database Systems Evolved

- A famous paper written by Codd in 1970 had the effect of significantly changing database systems.
- Codd proposed that database systems should present the user with a view of data organized as tables called relations. Behind the scenes there might be a complex data structure that allowed rapid response to queries. But, unlike the user of the earlier database systems, the user of a relational system would not be concerned with the storage structure. Queries could then be expressed in a high-level language which greatly increased the efficiency of database programmers.

Reference:
Codd, E.F., "A relational model for large shared data banks", Communications of ACM, **13**:6, pp. 377-387.

Smaller and Smaller Systems

- Originally, DBMS's were large, expensive software systems running on large mainframe computers.
- The size was necessary, because to store a gigabyte of data required a large computer.
- Today, a gigabyte fits on a single disk, and it is quite feasible to run a DBMS on a personal computer.
- Relational DBMS based on the relation model are beginning to appear as a common tool for computer applications much as spreadsheets and word processors did before them.

Larger and Larger Systems

- On the other hand, a gigabyte isn't much data. Large databases contain hundreds of gigabytes (or much more).
- As storage becomes cheaper, people often find new reasons to store greater amounts of data. Retail chains often store terabytes (1 terabyte = 1000 gigabytes, or 10^{12} bytes) information recording the history of every sale over a long period of time.
- Data other than text and numbers, such as video and audio, often occupy huge amounts of space per item. An hour of video occupies about a gigabyte. Databases storing satellite imagery will hold many petabytes of data (1 petabyte = 1000 gigabytes, or 10^{15} bytes).

Larger and Larger Systems (cont.)

- Handling such large databases required several technological advances.
  - Modern databases of modest size are stored on arrays of disks (secondary storage devices).
  - Databases almost never operate with the assumption that the "data" will fit into main memory. Older systems typically only had secondary storage devices in the form of magnetic tape (linear technology).
- Two trends allow database systems to deal with larger amounts of data faster.

Trends Influencing Larger Databases

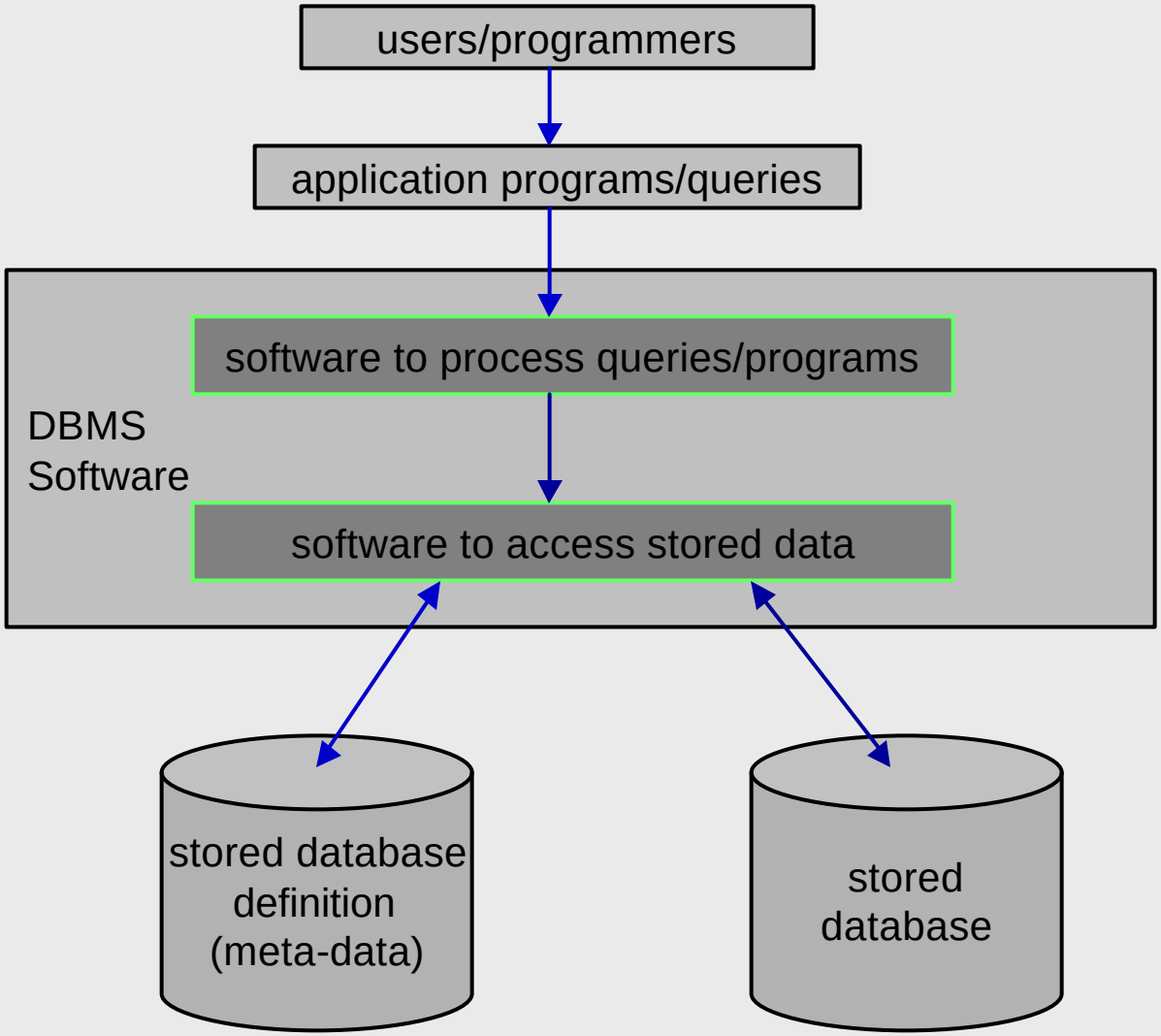
1. Tertiary Storage: The largest databases today require more than disks. Tertiary devices tend to store a terabyte each and have longer access times than do disks.
   - Typical disk access times are in the 10-20msec range. A typical tertiary device may take several seconds.
   - Tertiary devices involve transporting the object on which the data is stored to some reading device via a robotic conveyance of some sort. It is common to use CDs as the tertiary medium.

Trends Influencing Larger Databases

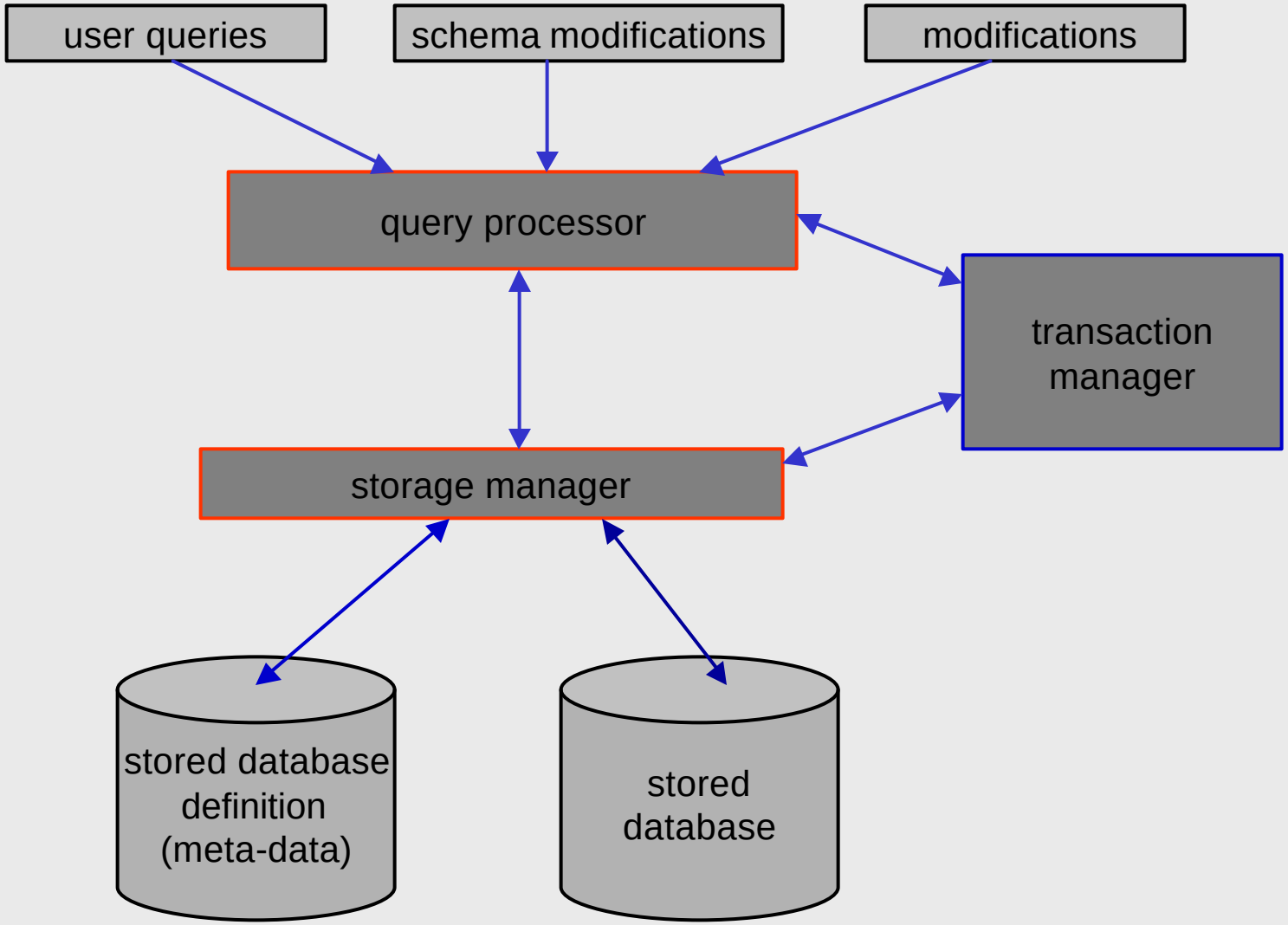
2. Parallel Computing: The ability to store enormous volumes of data is important, but it would be of little use if we could not access large amounts of that data quickly. Very large databases require speed enhancers. Speed enhancement is handled in many different fashions in modern databases including:

   - Indexing structures
   - Parallelism – both in terms of CPUs as well as in terms of the database itself. To some extent, distributed database systems can also be included as a speed enhancer, although in a slightly different manner, as we will see later in the term.

Components of a DBMS



Architecture of a DBMS



Overview of DBMS Components

- Stored Database and Meta-data: The stored database resides on secondary and tertiary devices. (At any given moment some portion of the database will also be mirrored in cache, but we will ignore this for the moment.)

- Meta-data is data about data. In this case the meta-data is a description of the data components of the database. Offsets of fields within records. Typing information. Schema information. Index information and so forth.

- For a given database, a DBMS may maintain many different indices designed to provide fast access to random data. Most indices are represented as B-trees in modern databases. B-trees tend to be short and fat resulting in fast access from root to leaves.

Overview of DBMS Components (cont.)

- Storage Manager: In a simple database system, the storage manager is nothing more than the file system of the underlying OS. In larger systems, for the purposes of efficiency, the DBMS's normally control storage on the disk directly.

- The storage manager consists of two basic components (1) the buffer manager, and (2) the file manager.

Overview of DBMS Components (cont.)

- File Manager: Keeps track of the location of files on the disks and obtains the block or blocks containing a file on request from the buffer manager. Disks are typically blocked into regions of contiguous space ranging between 2^{12} and 2^{14} bytes (between roughly 4000 to 16,000 bytes/block).

- Buffer Manager: Handles main memory. IT obtains blocks of data from the disk, via the file manager, and chooses a page of main memory in which to store than block. The paging algorithm will determine how long a page will remain in main memory. However, the transaction manager can also force a page in main memory to be returned to disk (we'll see the details of this later in the term as well).

Overview of DBMS Components (cont.)

- Query Manager: Turns a query or database manipulation, which may be expressed at a very high level (e.g., SQL) into a sequence of request for stored data such as specific tuples of a relation or parts of an index to a relation.
- Often the hardest part of query processing is query optimization, which involves the formulation of a good query execution strategy. We'll deal with query optimization in much greater detail later in the semester.

Overview of DBMS Components (cont.)

- Transaction Manager: There are certain guarantees that a DBMS must make when performing operations on a database. These guarantees are often referred to as the ACID properties.
  - Atomicity: all of a transaction is executed or none of it is executed.
  - Consistency: data cannot be in a inconsistent state.
  - Isolation: concurrent transactions must be isolated from each other both in effect and in visibility.
  - Durability: changes to the database caused by a transaction must not be lost even if the system fails immediately after the transaction completes.

Data vs. Information

In its “raw” form, data has little meaning. In this case it simply looks like a couple of lists of integer numbers. There is no context on which to base the data.

Data:	
0	11,500
5	12,300
10	12,800
15	10,455
20	12,200
25	13,900
30	14,220

Data vs. Information

By “processing” the data we have transformed it into something with more meaning. In this example, the processing consisted primarily of placing the data in context (which is usually done by adding more data! Although this additional data is really *metadata* (see below)). Now the data begins to take on more meaning.

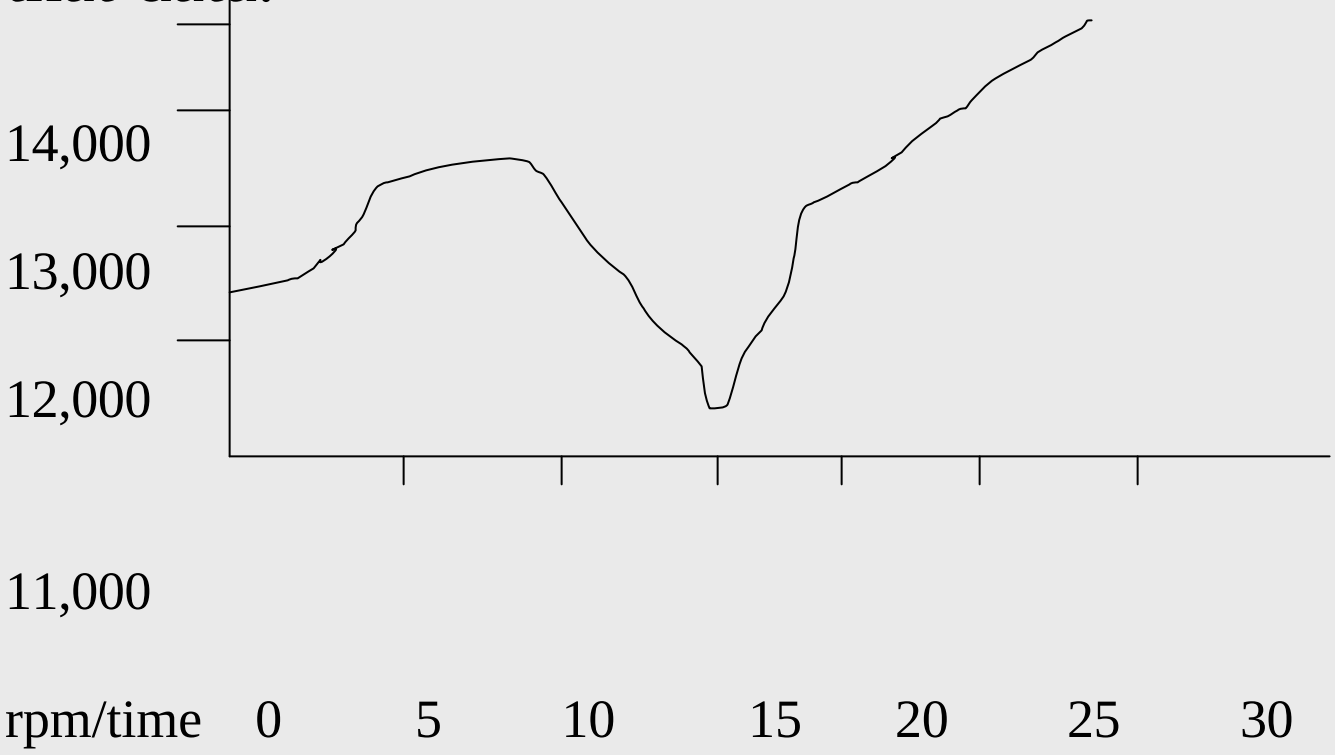
Information: Engine RPM Data: Roebling Road 10/4/2003 – Yamaha Heavy

Lap 12:

time	rpm
0	11,500
5	12,300
10	12,800
15	10,455
20	12,200
25	13,900
30	14,220

Data vs. Information

Considering the same data as was presented in the previous slide, consider the following processing of that data.



Graph: Partial Lap 12 – Roebling Road 10/4/2003 –

Derived Data vs. Physical Data

- Much of the data that appears in a database is there because it is modeling the characteristics of the enterprise which is represented in the database.
- For example, considering a database of students at UCF, we might represent the names, SSN, and major of each student along with a set of courses that they have taken accompanied by a grade in each of those courses. Somewhere along the line, someone with proper access to the database will have entered this data into the database in some fashion, typically either manually or electronically. If we now assume that this database also maintains, for each student, their GPA, then where does this GPA value come from? Is it input in some fashion by some user? Typically it isn't, but rather it is calculated by the DBMS (more specifically probably an application running on top of the DBMS, but we'll get to that later). Thus, a student's GPA value is derived from other data which is associated with that student. If the data on which the GPA is derived changes in some fashion, then so too will the derived value of the GPA.

Derived Data vs. Physical Data (cont.)

- Depending upon the level of sophistication of the application layer and/or DBMS, the amount of derived data which is resident in the database can be much larger than the amount of "actual data" or "physical data". Another reality that surrounds derived data is the question which concerns when or if it becomes "physical data" since there is no restriction that derived data ever be actually resident in the database!