CGS 2545: Database Concepts
Fall 2011

Introduction to Database Systems

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
http://www.cs.ucf.edu/courses/cgs2545/fall2011

Department of Electrical Engineering and Computer Science
Computer Science Division
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, audio, 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 terabytes, 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.

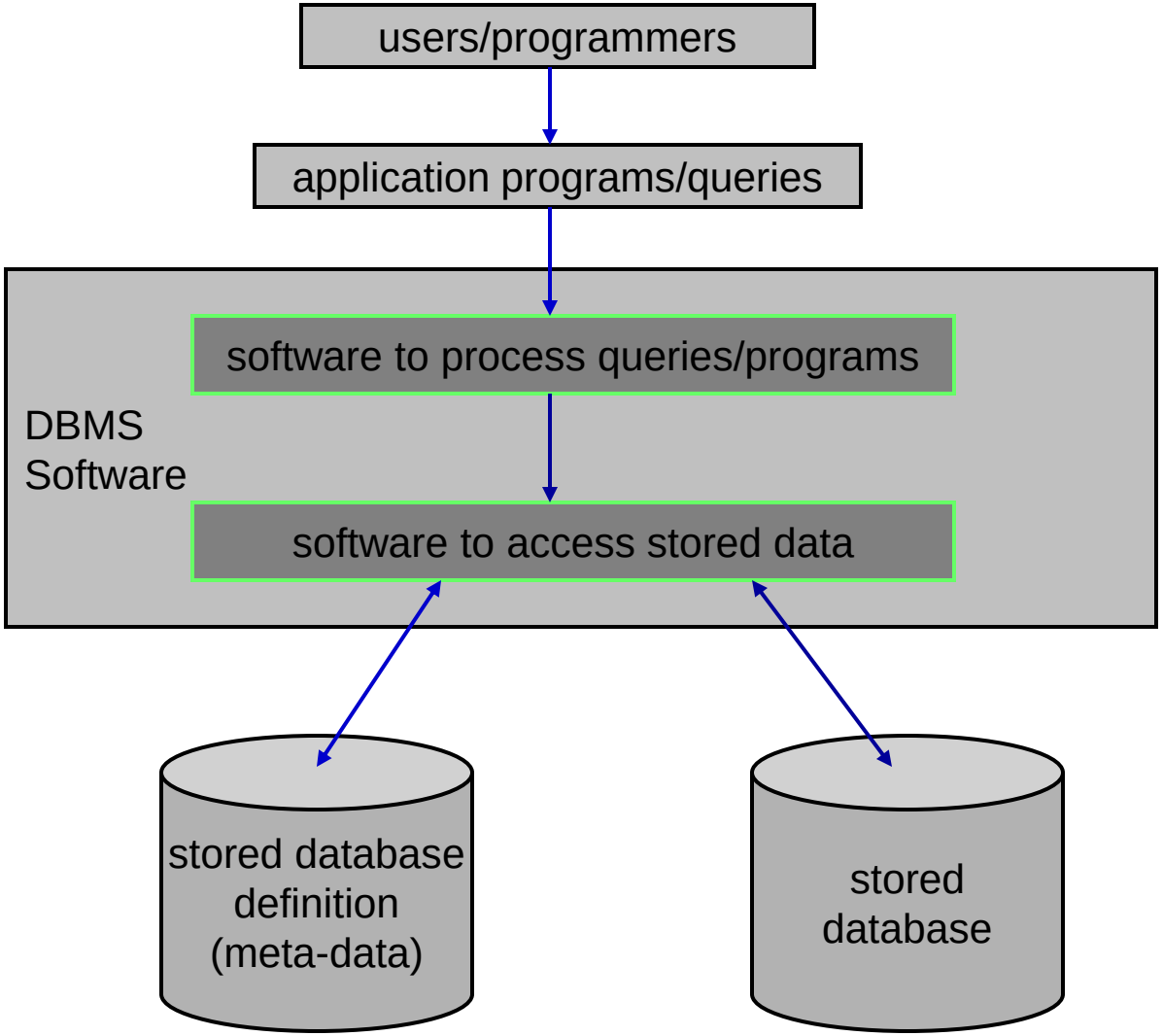
   – Tertiary devices involve transporting the object on which the data is stored to some reading device via a robotic conveyance of some sort. It is common to use CDs as the tertiary medium.

Trends Influencing Larger Databases

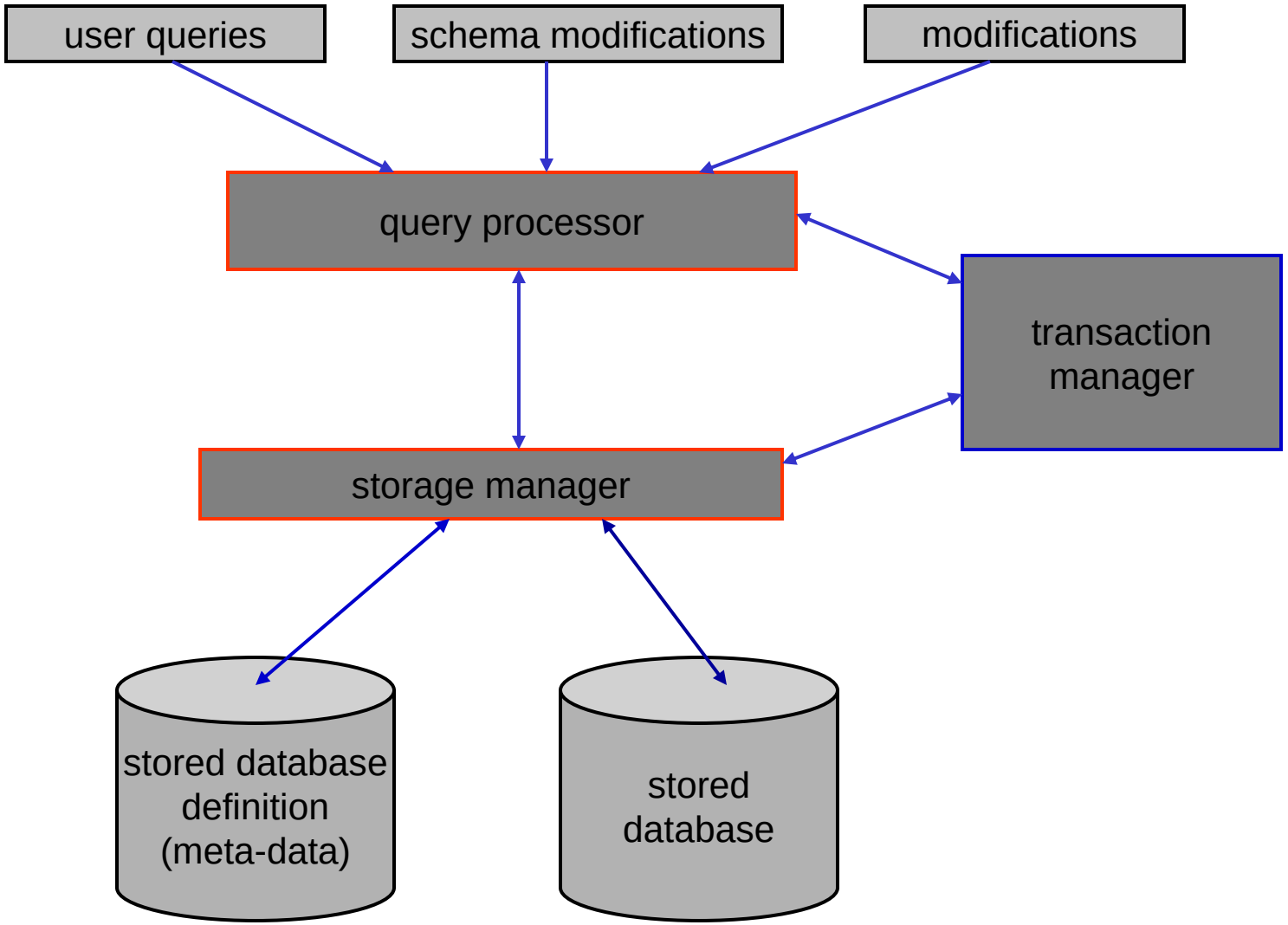
2. Parallel Computing: The ability to store enormous volumes of data is important, but it would be of little use if we could not access large amounts of that data quickly. Very large databases require speed enhancers. Speed enhancement is handled in many different fashions in modern databases including:

   - 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 page 14)). Now the data begins to take on more meaning.

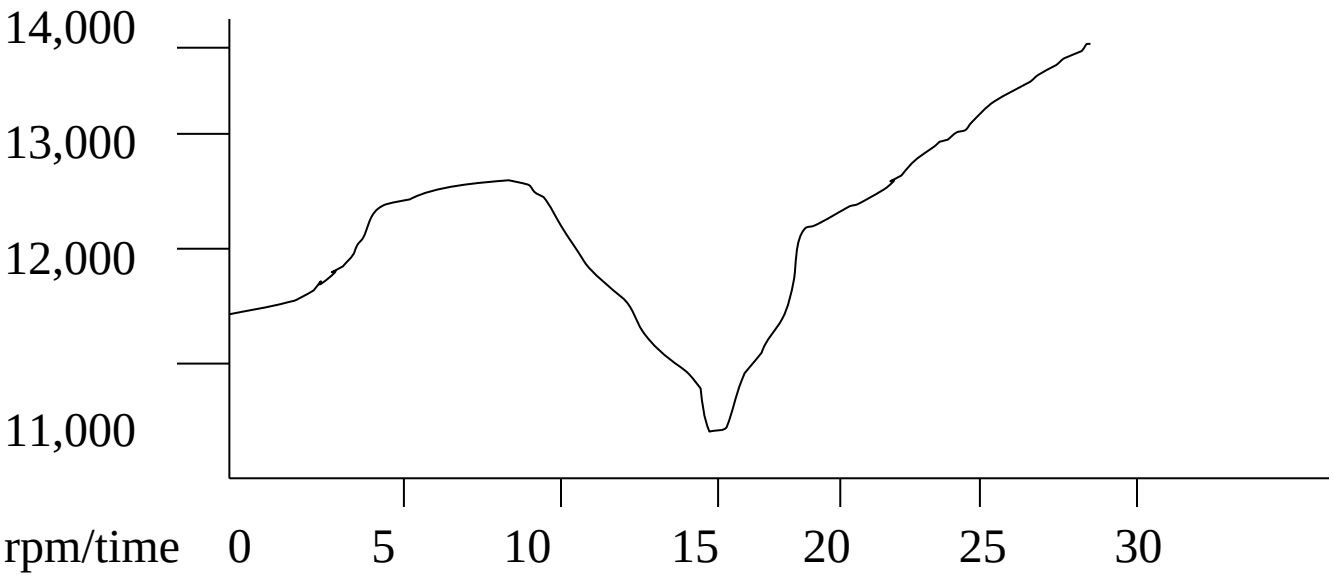
Information: Engine RPM Data: Roebling Road 10/22/2010 – Yamaha Heavy

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 10/22/2010 – Yamaha Heavy

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!

Database Design

- For the system to be acceptable to the end-users, the database design activity is crucial.
- A poorly designed database will generate error that may lead to bad decisions being made, which may have serious repercussions for the organization. On the other hand, a well-designed database produces, in an efficient way, a system that provides the correct information for the decision-making process to succeed.

Roles in the Database Environment

Data and Database Administrators

- The Data Administrator (DA) is responsible for the management of the data resource including database planning, development and maintenance of standards, policies and procedures, and conceptual/logical database design.
- The Database Administrator (DBA) is responsible for the physical realization of the database, including physical database design and implementation, security and integrity control, maintenance of the operational system, and ensuring satisfactory performance of the applications for users. The role of the DBA is more technically oriented than that of the DA.

Roles in the Database Environment (cont.)

Database Designers

- In large db design projects, we can distinguish between two types of designers: logical database designers and physical database designers.
  - Logical database designers are concerned with identifying the data (the entities and attributes), the relationships between the data, and the constraints on the data that will be stored in the database.
  - Physical database designers are highly dependent on the target DBMS, and there may be more than one way of implementing a mechanism. The physical db designer must be fully aware of the functionality of the target DBMS.

Roles in the Database Environment (cont.)

Application Developers

- Once the database has been implemented, the application programs that provide the required functionality for the end-users must be implemented. This is the responsibility of the application developers.

Roles in the Database Environment (cont.)

End Users

- End users are the "clients" for the database and can be broadly categorized into two groups based upon how they utilize the system.
  - Naïve users are typically unaware of the DBMS. They access the database through specially written application programs which attempt to make the operations as simple as possible. They typically know nothing about the database or the DBMS.
  - Sophisticated users are familiar with the structure of the database and the facilities offered by the DBMS. They will typically use a high-level query language like SQL to perform their required operations and may even write their own application programs.

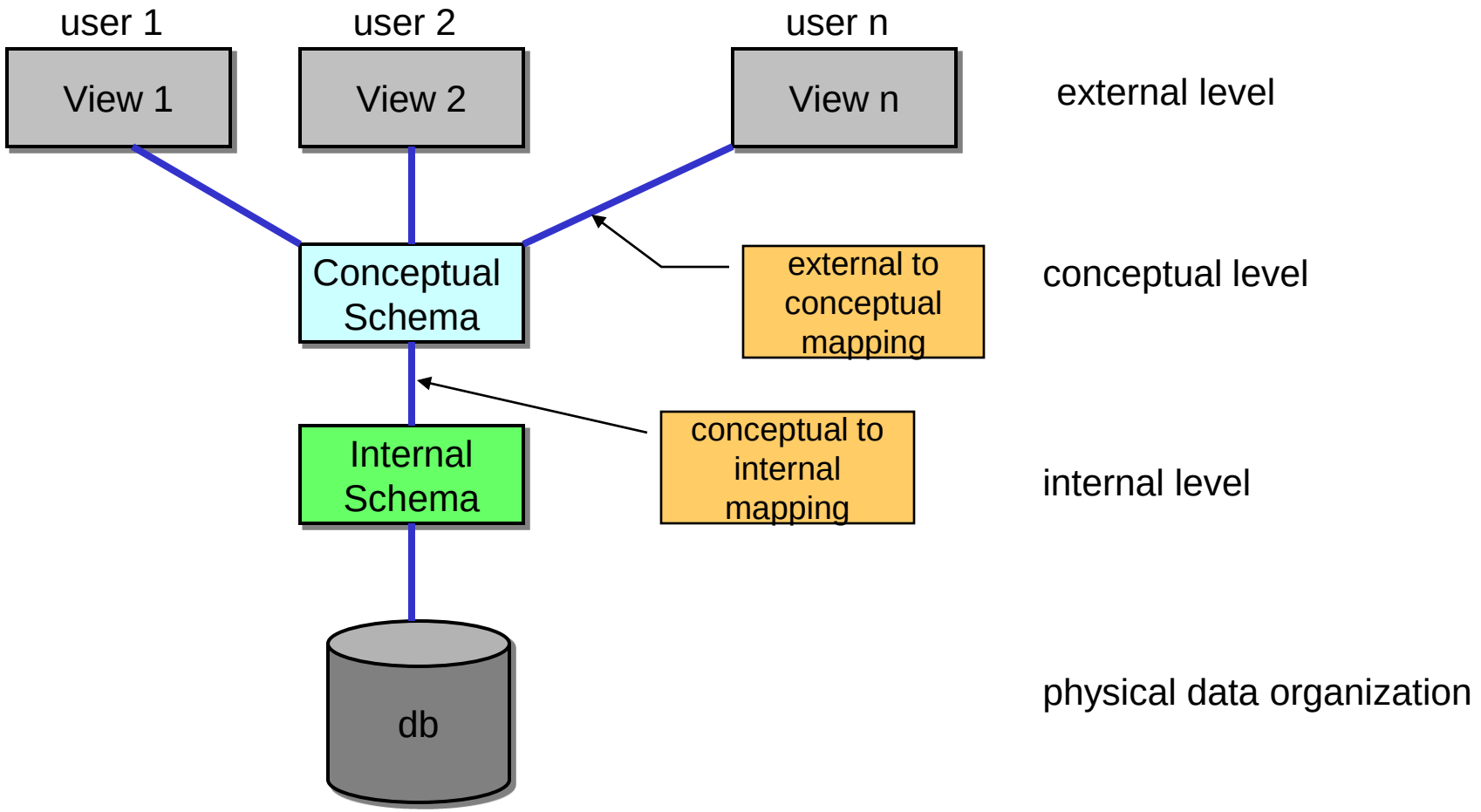
Advantages of DBMS

control of data redundancy	economy of scale
data consistency	balance of conflicting requirements
more information from same data	improved data accessibility
amount of data available	increased productivity
sharing of data	improved maintenance
improved data integrity	increased concurrency
improved data security	improved backup and recovery
enforcement of standards	improved responsiveness

Disadvantages of DBMS

complexity
size
cost of DBMSs
additional hardware costs
cost of conversion
performance (specific cases)
higher impact of failure

Three-Levels of Abstraction in a Database System



The External Level

- The external level is the user's view of the database.
- This level describes that part of the database which is relevant to each user.
- The external level consists of a number of different external views of the db. Each user has a view of the "real world" represented in a form that is familiar for that user.
- The external view includes only those entities, attributes, and relationships in the "real world" that the user is interested in. Other entities, attributes, and relationships may exist, but the user will be unaware that they even exist.

The External Level (cont.)

- It is often the case that different external views will have different representations of the same data.
  - Example: one view may represent dates in the form of (month, day, year) while another view may represent dates in the form of (day, month, year).
- Some views may include derived or calculated data. This is data that is not actually stored in the database as such, but created when needed.
  - Example: one view may need to see a person's age. However, this is probably not a stored value in the db since it would require daily updates. Rather, it is probably derived from stored data representing the person's date of birth and the current date.

The Conceptual Level

- The conceptual level is the community view of the database. This level describes *what* data is stored in the database and the relationships among the data.
- This is the level at which the logical structure of the entire database as seen by the DBA is contained. It represents a complete view of the data requirements of the organization that is independent of any storage considerations.
- The conceptual level supports each external view, in that any data available to a user must be contained in, or derivable from, the conceptual level.
- This level contains no storage-dependent details.
  - For example, an entity may be defined as represented by an integer data type at this level, but the number of bytes it occupies is not specified at this level.

The Internal Level

- The internal level represents the physical representation of the database on the computer. This level describes *how* the data is stored in the database.
- The internal level describes the physical implementation necessary to achieve optimal runtime performance and storage space utilization.
- It covers the data structures and file organizations used to store the data on the storage devices.
- It interfaces with the OS access methods (file management techniques for storing and retrieving data records) to place the data on the storage devices, build indexes, retrieve the data, and so on.

The Physical Level

- Below the internal level is the physical level that may be managed by the OS under the direction of the DBMS.
- The functions of the DBMS and the OS at the physical level are not clear cut and will vary from system to system.
- Some DBMSs take advantage of many of the OS access methods, while others will use only the most basic ones and create their own file organizations.
- The physical level below the DBMS consists of items only the OS knows, such as exactly how the sequencing is implemented and whether the fields of internal records are stored as contiguous bytes on the disk.

Schemas, Mappings, and Instances

- The overall description of the database is called the database schema.
- There are three different types of schema in the database and these are defined according to the levels of abstraction of the three-level architecture.
  - At the highest level, there are multiple external schema. Also called subschemas, that correspond to different views of the data.
  - At the conceptual level, there is one conceptual schema, which describes all the entities, attributes, and relationships along with their integrity constraints.
  - At the lowest level of abstraction, there is one internal schema, which is a complete description of the internal model, containing the definition of the stored records, methods of representation, etc..

Schemas, Mappings, and Instances (cont.)

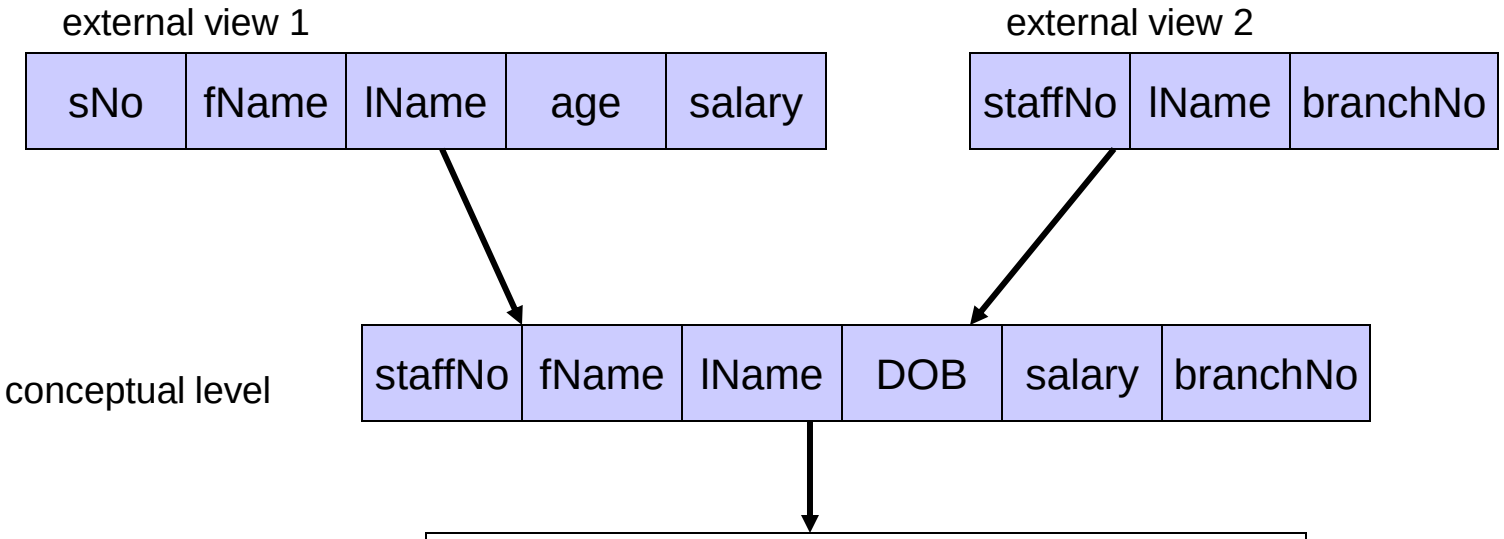
- The DBMS is responsible for mapping between these three types of schema. It must also check the schemas for consistency; in other words, the DBMS must check that each external schema is derivable from the conceptual schema, and it must use the information in the conceptual schema to map between each external schema and the internal schema.

- The conceptual schema is related to the internal schema through a conceptual/internal mapping. This enables the DBMS to find the actual record or combination of records in physical storage that constitute a logical record in the conceptual schema, together with any constraints to be enforced on the operations for that logical record.

Schemas, Mappings, and Instances (cont.)

- Each external schema is related to the conceptual schema by an external/conceptual mapping. This enables the DBMS to map names in the user's view on to the relevant part of the conceptual schema.

Schemas, Mappings, and Instances (cont.)



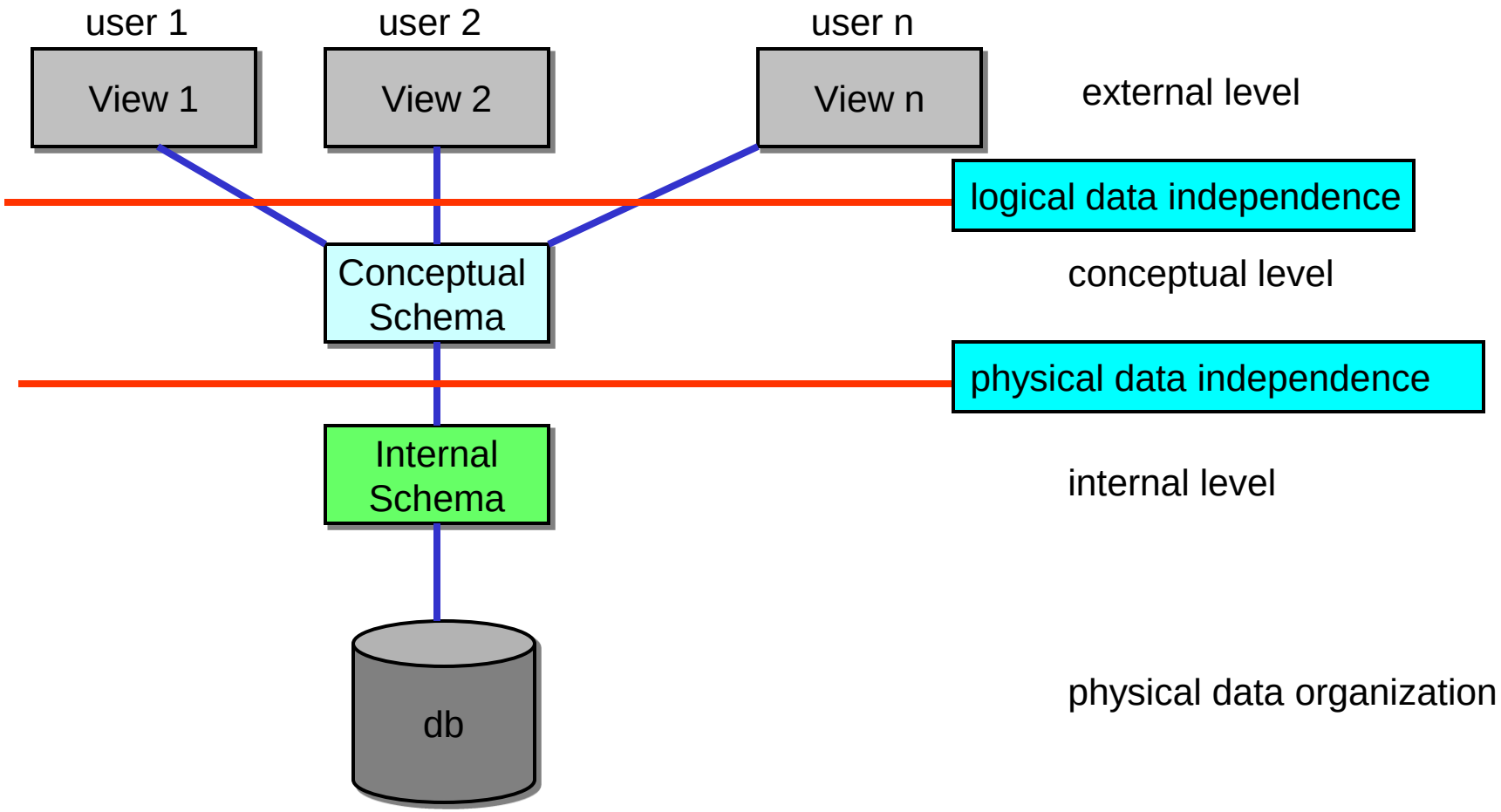
internal level

```
struct STAFF
    int staffNo;
    int branchNo;
    char fName[15];
    chaf lName[15];
    struct data dateofBirth;
    float salary;
    struct STAFF *next   /*ptr to next STAFF record */
};
index staffNo; index branchNo; /*define indices for STAFF */
```


Data Independence

- One of the major objectives of the three-level architecture is provide data independence, which means that the upper levels are unaffected by changes to lower levels.
- There are two types of data independence: logical and physical.
- **Logical data independence** refers to the immunity of the external schemas to changes in the conceptual schema.
- **Physical data independence** refers to the immunity of the conceptual schema to changes in the internal schema.

Data Independence (cont.)



Database Languages

- A data sublanguage consists of two parts: a Data Definition Language (DDL) and a Data Manipulation Language (DML).
- The DDL is used to specify the database schema and the DML is used to both read and update the database.
- These languages are called *data sublanguages* because they do not include constructs for all computing needs such as conditional or iterative statements, which are provided by the high-level programming languages.
- Most DBMSs have a facility for embedding the sublanguage in a high-level programming language such as COBOL, Pascal, C, C++, Java, or Visual Basic which is then called the *host language*.
- Most data sublanguages also provide a non-embedded or interactive version of the language to be input directly from a terminal.

The Data Definition Language (DDL)

- A Data Definition Language is a language that allows the DBA or user to describe and name the entities, attributes, and relationships required for the application, together with any associated integrity and security constraints.
- The result of the compilation/execution of the DDL statements is a set of tables stored in special files collectively referred to as the system catalog. The system catalog is also commonly referred to as the data dictionary or data directory.

The Data Manipulation Language (DML)

- A Data Manipulation Language is a language that provides a set of operations to support the basic data manipulation operations on the data held in the database.
- DML operations usually include the following:
  - insertion of new data into the database.
  - modification of data stored in the database.
  - retrieval of data contained in the database.
  - deletion of data from the database.
- The part of the DML that involves data retrieval is called a query language.

DMLs (cont.)

- DMLs are distinguished by their underlying retrieval constructs. We can distinguish two basic types of DMLs: procedural and non-procedural.
- Procedural DMLs are languages in which the user informs the system *what* data is required and exactly *how* to retrieve that data.
- Non-procedural DMLs are languages in which the user informs the system *only* of *what* data is required and leaves the how to retrieve the data entirely up to the system.
- It is common for procedural DMLs to be embedded in high-level programming languages.
- Procedural DMLs tend to be more focused on individual records while non-procedural DMLs tend to operate on sets of records.

Fourth Generation Languages

- There is no consensus as to what constitutes a 4GL. In essences it is a shorthand programming language. What requires several hundred lines of code in a 3GL will require only a few lines of code in a 4GL.
- 3GLs are procedural while 4GLs are non-procedural.
- 4GLs include spreadsheets and database languages.
- SQL and QBE are examples of 4GLs.

Data Models

- A data model is an integrated collection of concepts for describing and manipulating data, relationships between data, and constraints on the data in an organization.
- A model is a representation of "real world" objects and events, and their associations. It is an abstraction that concentrates on the essential, inherent aspects of an organization and ignores accidental properties.
- A data model must provide the basic concepts and notations that will allow database designers and end-users unambiguously and accurately to communicate their understanding of the organizational data.

Data Models (cont.)

- A data model can be thought of as comprising three components:

1. A structural part, consisting of a set of rules according to which databases can be constructed.
2. A manipulative part, defining the types of operations that are allowed on the data (this includes operations that are used for updating or retrieving data from the database and for changing the structure of the database).
3. Possibly a set of integrity rules, which ensures that the data is accurate.

Data Models (cont.)

- Looking at the three level architecture, we can identify three different, related data models.

1. An external data model to represent each user's view of the organization.
2. A conceptual data model to represent the logical (or community view) that is DBMS independent.
3. An internal data model to represent the conceptual schema in such a way that it can be understood by the DBMS.

Data Models (cont.)

- There have been many different data models which have been theorized, utilized, developed, and implemented over the years. They fall into three broad categories: object-based, record-based, and physical.
- There are three principle record-based models: the relational data model, the network data model, and the hierarchical data model. Our focus will be on the relational data model in this course.