COSC 304
Introduction to Database Systems

Database Introduction

Dr. Ramon Lawrence
University of British Columbia Okanagan
ramon.lawrence@ubc.ca

What is a database?

A ***database*** is a collection of logically related data for a particular domain.

A ***database management system*** (***DBMS***) is software designed for the creation and management of databases.

- e.g. Oracle, DB2, Access, MySQL, SQL Server, MongoDB

Bottom line: A ***database*** is the *data* stored and a ***database system*** is the *software* that manages the data.

Databases in the Real-World

Databases are everywhere in the real-world even though you do not often interact with them directly.

- $25 billion dollar annual industry

Examples:

- Retailers manage their products and sales using a database.
  - Wal-Mart has one of the largest databases in the world!
- Online web sites such as Amazon, eBay, and Expedia track orders, shipments, and customers using databases.
- The university maintains all your registration information and marks in a database.

Can you think of other examples?

What data do ***you*** have?

Example Problem

Implement a system for managing products for a retailer.

- Data: Information on products (SKU, name, desc, inventory)
- Add new products, manage inventory of products

How would you do this without a database?

What types of challenges would you face?

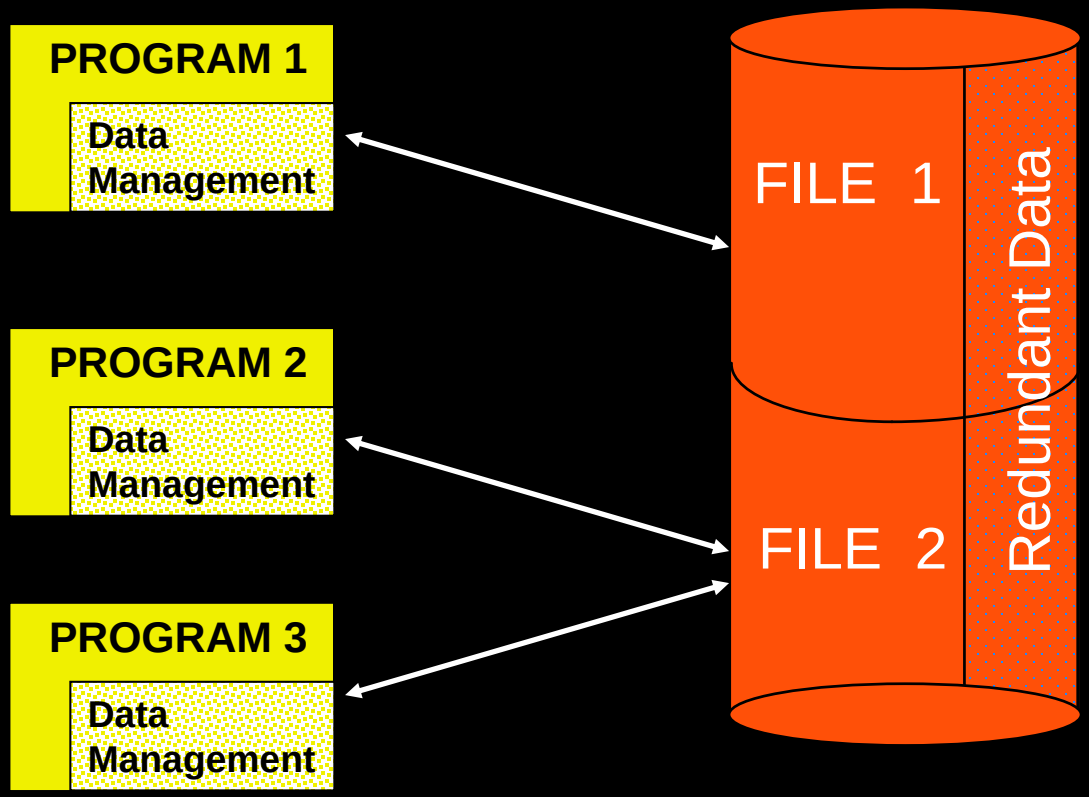
Why do we need databases?

Without a DBMS, your application must rely on files to store its data *persistently*. A ***file-based system*** is a set of applications that use files to store their data.

Each application in a file-based system contains its own code for accessing and manipulating files. This causes several problems:

- Code duplication of file access routines
- Data is usually highly redundant across files
- High maintenance costs
- Hard to support multi-user access to information
- Difficult to connect information present in different files
- Difficulty in developing new applications/handling data changes

File Processing Diagram



Data Independence and Abstraction

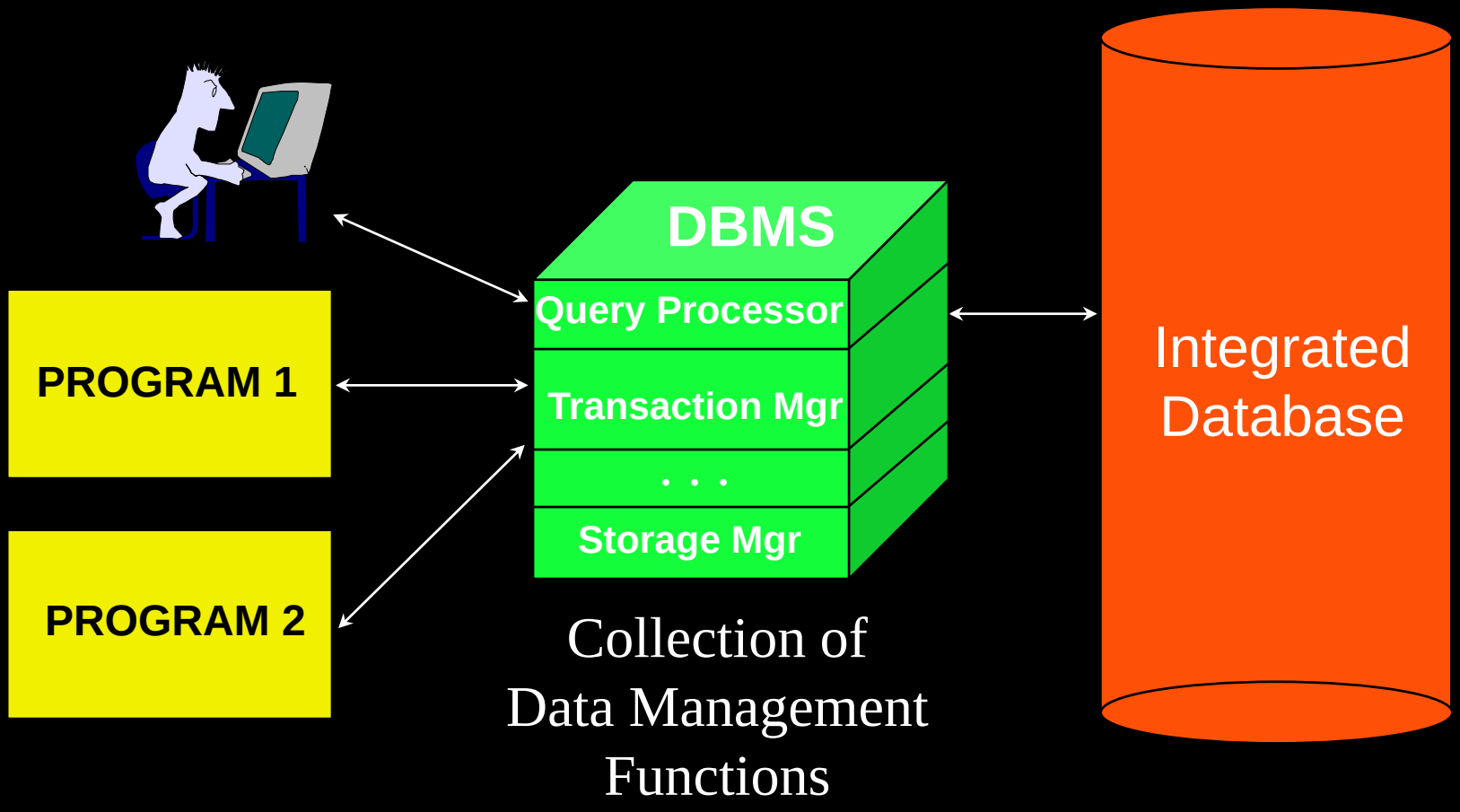
The major problem with developing applications based on files is that the application is dependent on the file structure.

That is, there is no ***program-data independence*** separating the application from the data it is manipulating.

- If the data file changes, the code that accesses the file must be changed in the application.

One of the major advantages of databases is they provide data abstraction. ***Data abstraction*** allows the internal definition of an object to change without affecting programs that use the object through an external definition.

Database System Approach



DBMS

A database management system provides *efficient*, *convenient*, and *safe multi-user* storage and access to *massive* amounts of *persistent* data.

Efficient - Able to handle large data sets and complex queries without searching all files and data items.

Convenient - Easy to write queries to retrieve data.

Safe - Protects data from system failures and hackers.

Massive - Database sizes in gigabytes/terabytes/petabytes.

Persistent - Data exists after program execution completes.

Multi-user - More than one user can access and update data at the same time while preserving consistency.

Data Definition Language

A DBMS achieves these goals by supporting data abstraction.

- The DBMS takes the description of the data and handles the low-level details of how to store it, retrieve it, and handle concurrent access to it.

The database is described to the DBMS using a ***Data Definition Language*** (***DDL***). The DDL allows the user to create data structures in the data model used by the database.

A ***data model*** is a collection of concepts that can be used to describe the structure of a database.

- In the relational model, data is represented as tables and fields.
- Examples: relational model, XML, graphs, object-oriented, JSON

Schemas

A database designer uses a DDL to define a schema for the database. The schema is maintained and stored in the ***system catalog***. The schema is one type of ***metadata***.

A ***schema*** is a description of the structure of the database.

- A schema contains structures, names, and types of data stored.
- For example, the data model for Access is a relational model. A relational model contains tables and fields as its model constructs. The following DDL creates a product table:

```
CREATE TABLE product(
    sku as VARCHAR(10) NOT NULL,
    name as VARCHAR(40),
    desc as VARCHAR(50),
    inventory as INTEGER,
    PRIMARY KEY (sku)
);
```

Data Manipulation Language

Once a database has been created using DDL, the user accesses data using a ***Data Manipulation Language*** (***DML***).

- The standard DML is SQL.
- The DML allows for the insertion, modification, retrieval, and deletion of data.

A DML provides data abstraction as user queries are specified using the names of data and not their physical representation.

- For example, in a file system storing 3 fields, you would have to provide the exact location of the field in the file. In a database, you would only have to specify it by name.
- The DBMS contains all the code for accessing the data, so the applications do not have to worry about those details any more.

SQL Examples

Retrieve all products in the database:

```
SELECT sku, name, desc, inventory FROM product;
```

Retrieve all products where inventory < 10:

```
SELECT name, inventory FROM product WHERE inventory < 10;
```

Insert a new product into the database:

```
INSERT INTO product VALUES ('1234','Soap','Ivory',100);
```

Delete a product from the database:

```
DELETE FROM product WHERE sku = '1234';
```

Database Properties Question

Question: **True or False**: The data in a database is lost when the power to the computer is turned off.

A) true

B) false

Database Abstraction Question

Question: Defining how data is stored using DDL is similar to what in object-oriented programming?

A) Objects

B) Classes

C) Inheritance

D) Polymorphism

DDL vs. DML Question

Question: If you are querying data in a database, which language are you using:

A) DML

B) DDL

C) schemas

D) Java

Components of a DBMS

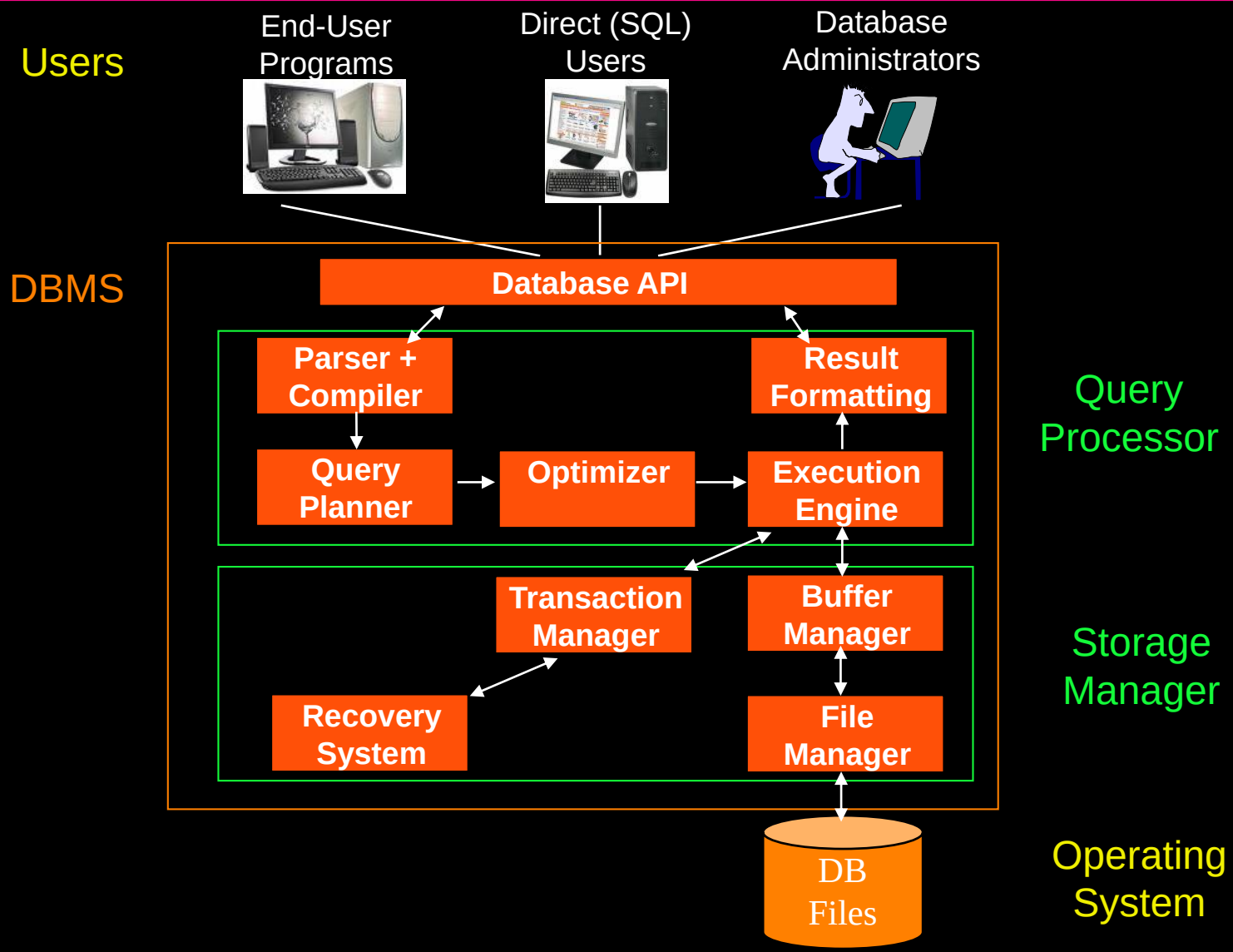
A DBMS is a complicated software system containing many components:

- ***Query processor*** - translates user/application queries into low-level data manipulation actions.
  - ⇨ Sub-components: query parser, query optimizer

- ***Storage manager*** - maintains storage information including memory allocation, buffer management, and file storage.
  - ⇨ Sub-components: buffer manager, file manager

- ***Transaction manager*** - performs scheduling of operations and implements concurrency control algorithms.

DBMS Architecture



Database Architectures

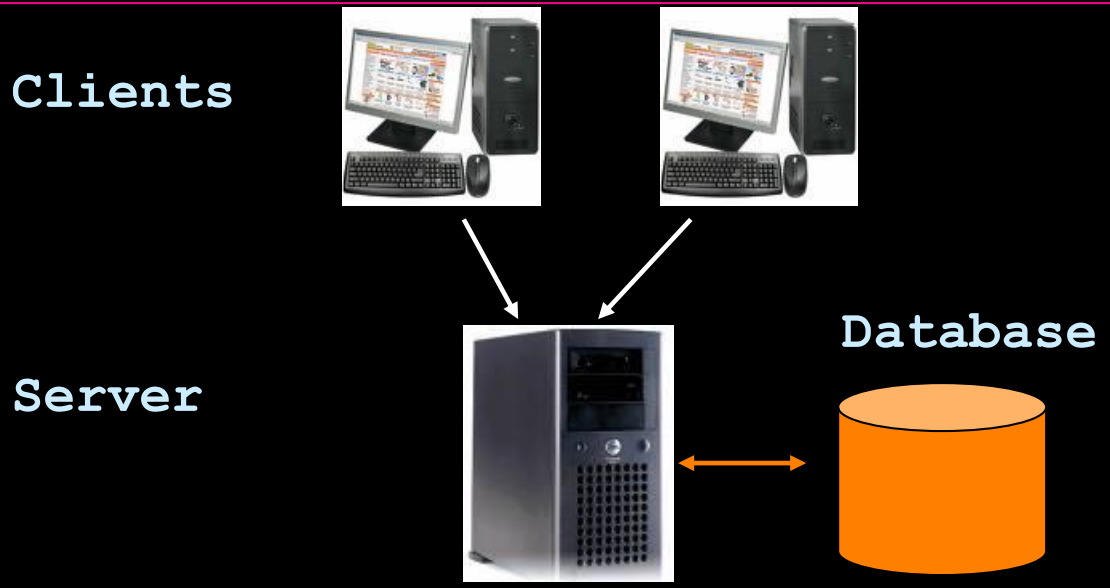
There are several different database architectures:

- **File-server (embedded) architecture** - files are shared but DBMS processing occurs at the clients (e.g. Microsoft Access or SQLite)

- **Two-Tier client-server architecture** - dedicated machine running DBMS accessed by clients (e.g. SQL Server)

- **Three-Tier client-server architecture** - DBMS is bottom tier, second tier is an application server containing business logic, top tier is clients (e.g. Web browser-Apache/Tomcat-Oracle)

Two-Tier Client-Server Architecture



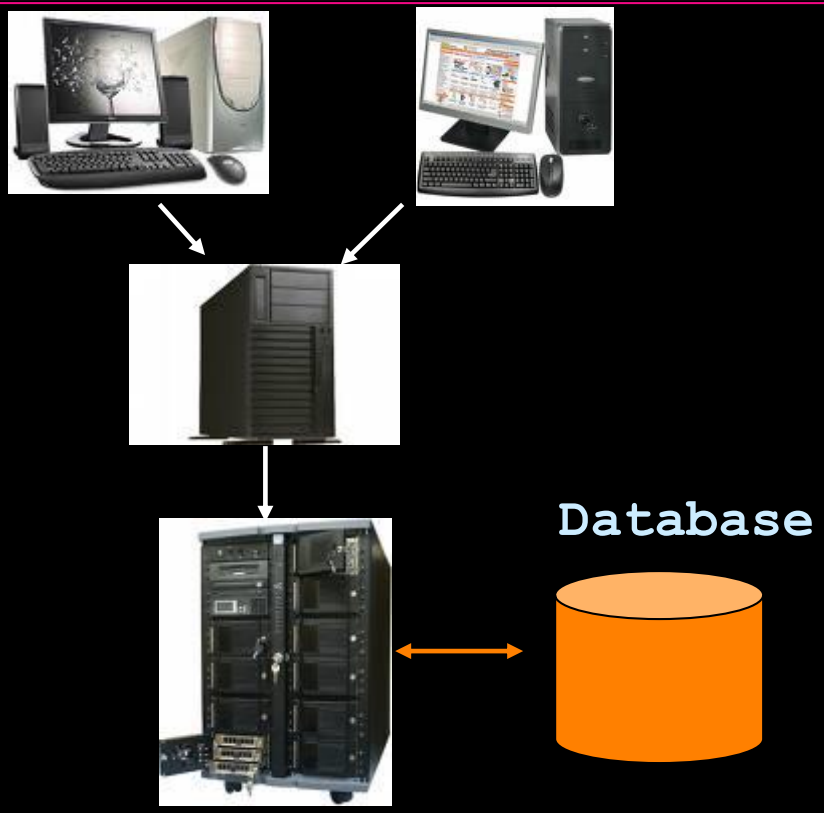
Tier 1: Client
- User Interface
- Business and Data Logic

Tier 2: Database Server
- Server-side validation
- Data storage/management

Advantages:

- Only one copy of DBMS software on dedicated machine.
- Increased performance.
- Reduced hardware and communication costs.
- Easier to maintain consistency and manage concurrency.

Three-Tier Client-Server Architecture



Tier 1: Client (Web/mobile)
- User Interface

Tier 2: Application Server
- Business logic
- Data processing logic

Tier 3: Database Server
- Data validation
- Data storage/management

Advantages:

- Reduced client administration and cost using thin web clients.
- Easy to scale architecture and perform load balancing.

Database People

There are several different types of database personnel:

- ***Database administrator*** (***DBA***) - responsible for installing, maintaining, and configuring the DBMS software.
- ***Data administrator*** (***DA***) - responsible for organizational policies on data creation, security, and planning.
- ***Database designer*** - defines and implements a schema for a database and associated applications.
  - ***Logical/Conceptual database designer*** - interacts with users to determine data requirements, constraints, and business rules.
  - ***Physical database designer*** - implements the logical design for a data model on a DBMS. Defines indexes, security, and constraints.
- ***DBMS developer*** - writes the DBMS software code.
- ***Application developer*** - writes code that uses the DBMS.
- ***User*** - uses the database directly or through applications.

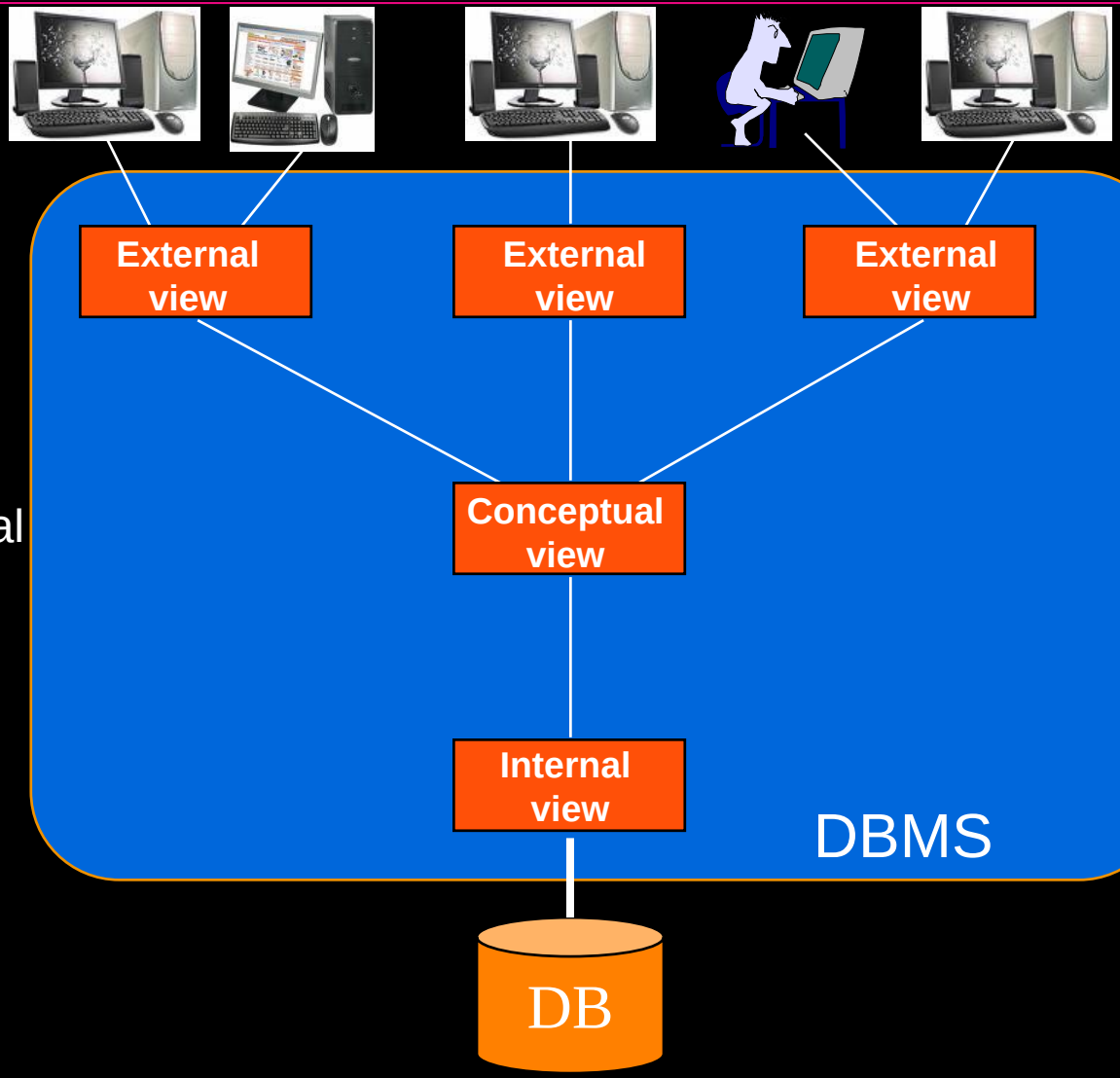
ANSI/SPARC Architecture

One of the major advantages of database systems is data abstraction. Data abstraction is achieved by defining different views of the data. Each view isolates higher-level views from data representation details.

The ANSI/SPARC architecture defined in 1975 consists of three views:

- ***Internal View*** - The physical representation of the database on the computer. *How* the data is stored.
- ***Conceptual View*** - The logical structure of the database that describes *what* data is stored and its relationships.
- ***External View*** - The *user's view* of the database that provides the part of the database relevant to the user.

ANSI/SPARC Architecture



3-schema architecture

Benefits of 3-Schema Architecture

External Level:

- Each user can access the data, but have their own view of the data independent of other users.
  - *Logical data independence* - conceptual schema changes do not affect external views.

Conceptual Level:

- Single shared data representation for all applications and users which is independent of physical data storage.
  - Users do not have to understand physical data representation details.
  - The DBA can change the storage structures without affecting users or applications. *Physical data independence* - conceptual schema not affected by physical changes such as adding indexes or distributing data.

Internal (Physical) Level:

- Provides standard facilities for interacting with operating system for space allocation and file manipulation.

Microsoft Access and the 3-Schema Architecture

External Level:

- Microsoft Access does not call them views, but you can store queries and use the results in other queries (like a view).
  - ⇨ External schema is the query (view) name and the attribute metadata.

Conceptual Level:

- All tables and field definitions are in the schema (accessible from the Tables tab).
  - ⇨ Note that conceptual **schema** is not the data but the metadata.

Physical Level:

- Access represents all data in a single file whose layout it controls.
- The system processes this raw data file by knowing locations and offsets of relations and fields.

ANSI/SPARC Architecture Three Levels of Views

Question: What are the three levels of views in the ANSI/SPARC architecture starting with the view closest to the user?

A) Physical, Conceptual, External

B) External, Physical, Conceptual

C) Physical, External, Conceptual

D) External, Conceptual, Physical

E) User, Logical, System

ANSI/SPARC Architecture Abstraction with Views

Question: Assume you have a Java program accessing data stored in a file. Select ***one*** true statement.

A) The file organization is changed. The physical view is where this change is made.

B) A field is added to the database. The conceptual view is changed.

C) A user account has restricted access to the file. The external view must be changed.

D) More than one of the above

Conclusion

A database is a collection of logically related data stored and managed by a database management system (DBMS).

A DBMS has advantages over traditional file systems as they support data independence and provide standard implementations for data management tasks.

- Data definition and manipulation languages (DDL and DML)
- System catalog maintains database description (schema) defined using the data model.

The 3-schema architecture consists of external, conceptual, and physical schemas. Each view provides data abstraction and isolates the layer above from certain data manipulation details.

Objectives

- Define: database, DBMS, database application/system
- Describe the features of a file-based system and some limitations inherent in that architecture.
- Define program-data independence and explain how it is achieved by databases but not by file systems.
- Define DDL and DML. What is the difference?
- List some modules of a DBMS.
- List different people associated with a DBMS and their roles.
- Explain how a schema differs from data.
- Draw a diagram of the 3-schema architecture and explain what each level provides. List benefits of the architecture.
- How does a schema provide data independence?
- Compare/contrast two-tier and three-tier architectures.