

Lecture 1. Part 2.

Databases: Core Concepts

Databases: Core Concepts

1. The History and Origins of Databases.
2. File Systems as the Precursor to Databases and DBMS.

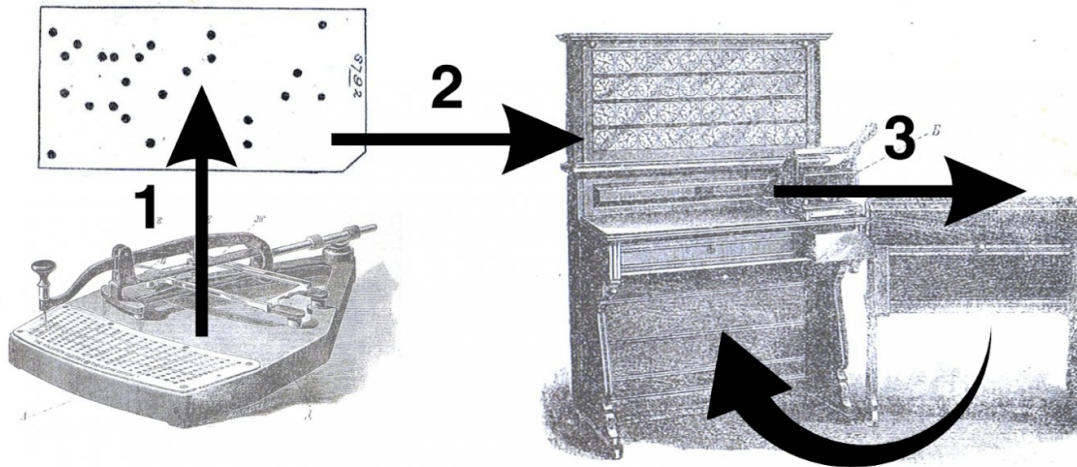
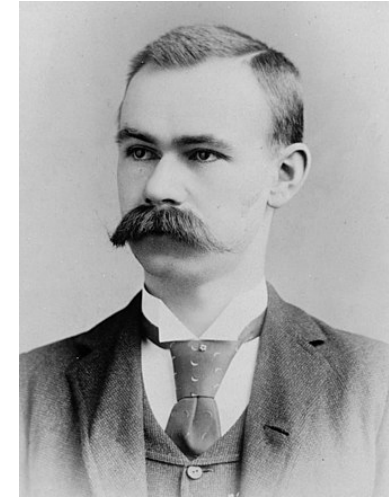
Data Storage and Processing Before Computers

Data before computers



Data before computers

In the 1880s, he invented the **tabulator**—an electromechanical machine capable of reading and sorting statistical records encoded on **punch cards**.



Herman Hollerith
(1860-1929),
American engineer and
inventor

These machines were used to process census data for the United States (1890, 1900), Canada, and Austria.

Data before computers

1896 – Hollerith founded the Tabulating Machine Company (**TMC**).

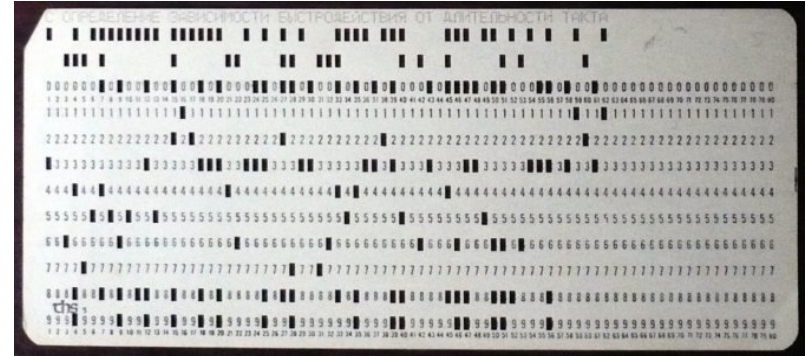
1911 – He sold the company; it became part of the Computing-Tabulating-Recording Company (**CTR**).

1924 – CTR was renamed **IBM** (International Business Machines Corporation).



*Herman Hollerith with his tabulator, 1894.
Columbia University.*

IBM Punch Cards and Tabulators

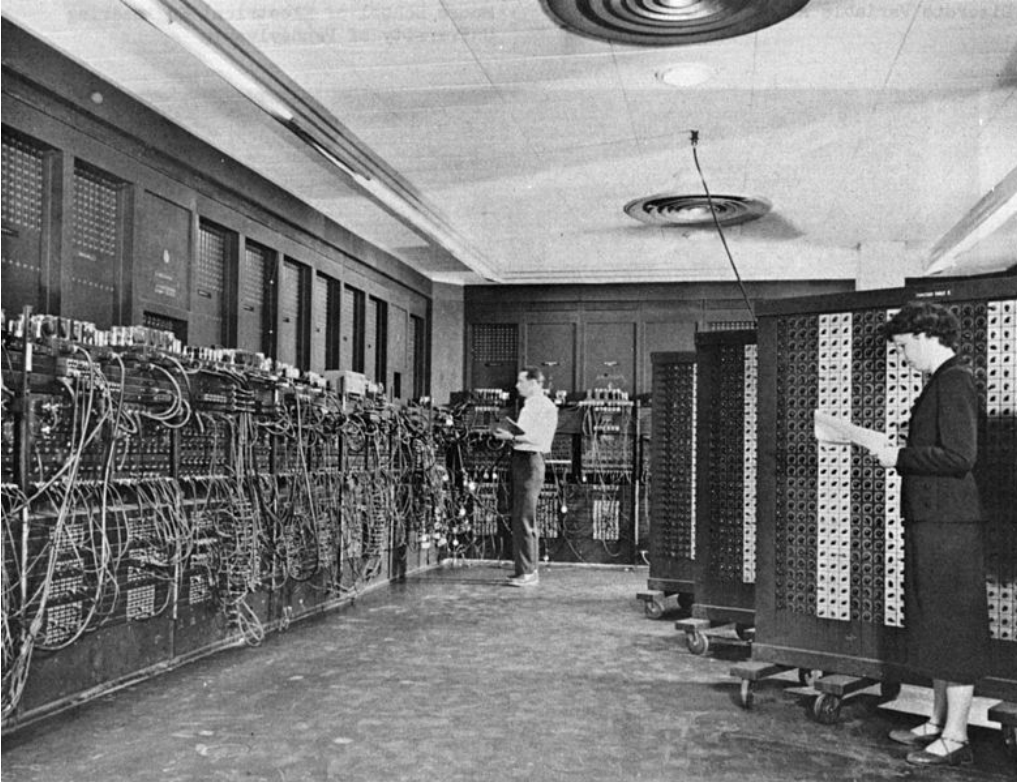


80 bytes

IBM Tabulators at the U.S. Social Security Administration, Baltimore, 1936

The Emergence of Computers (Electronic Computing Machines)

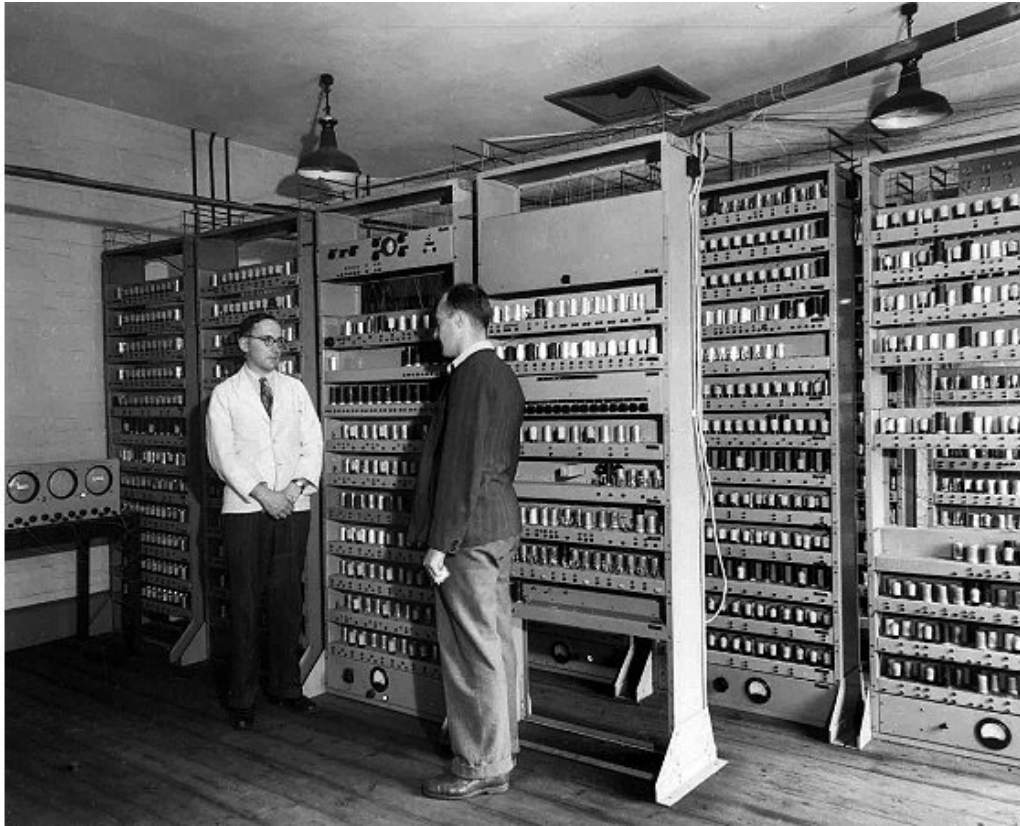
Early Computer Systems: Limited Instruction Sets and Programs



1945 – **ENIAC** (Electronic Numerical Integrator and Computer)

- Floor space: 167 m²
- Weight: 27 metric tons
- Used decimal arithmetic
- Input/Output: Punch cards
- Programming: Manual configuration via switches and cables
- Contained 18,000 vacuum tubes

The First Computer with von Neumann Architecture



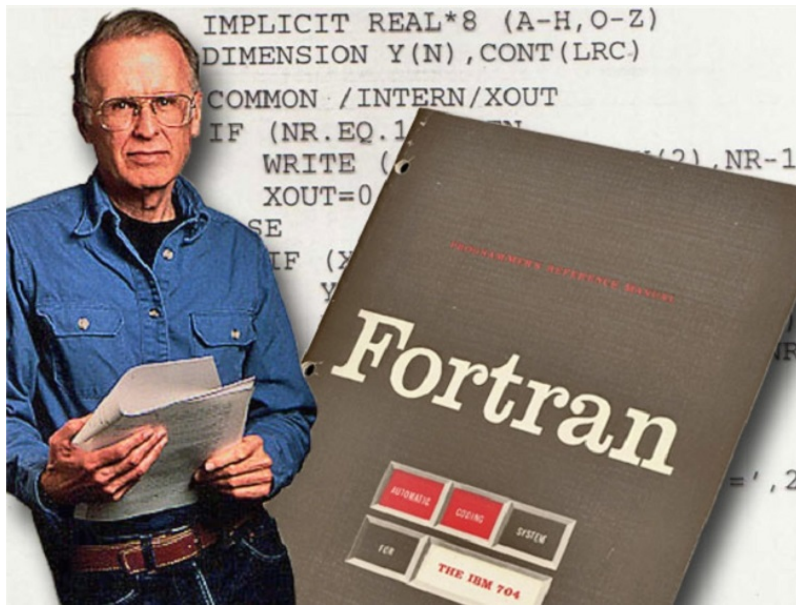
1949 – **EDSAC** (Electronic Delay Storage Automatic Computer)
University of Cambridge

- Floor space: 20 m²
- 3,000 vacuum tubes

Supported assembly language and subroutines

In 1952, the first computer game—"Tic-Tac-Toe"—was written on EDSAC

Fortran - the first high-level programming language



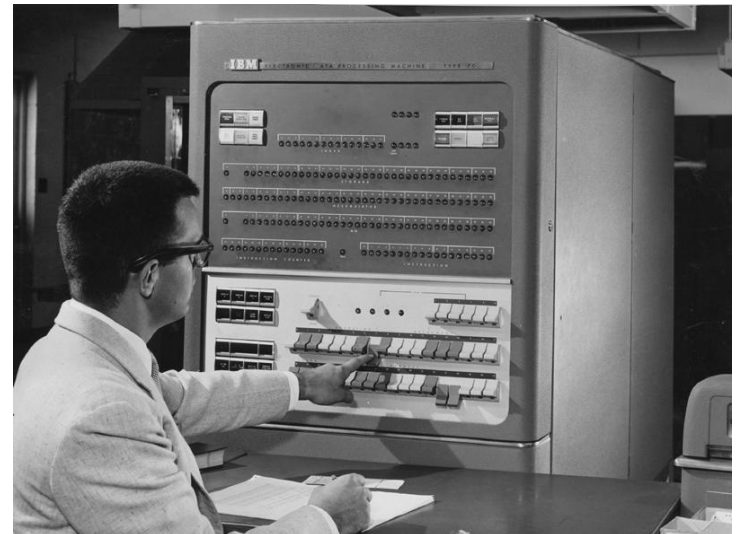
1954 год

John Backus (1924-2007)

Developed by IBM for the IBM 704 computer

Formula Translation

- Assignment operator
- Arrays
- DO loop statement



Data in Computer

Data in early computers





5 Mb of data

62 500 punched cards



Magnetic tapes

**The first HDD
IBM, 1956**

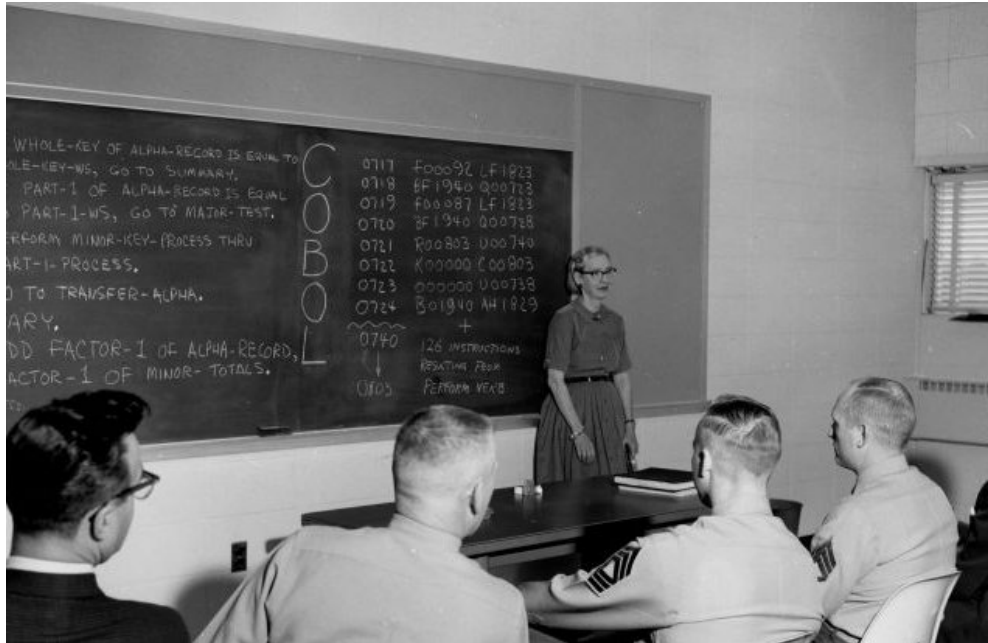
5 Mb, 1 ton



Classification of Tasks Solved on Computers

Type	Data representation	Computation Algorithm
Computational tasks	Simple	Complex
Traditional data processing tasks (non-computational)	Complex	Simple
Modern data processing tasks	Complex	Complex

COBOL - the first language for data processing



1959

Grace Hopper - project leader

Common **B**usines
Oriented **L**anguage

- Data structures (records)
- Files

```
PERFORM UNTIL EndOfStudentFile
    ADD 1 TO MonthCount(MOBirth)
    READ StudentFile
        AT END SET EndOfStudentFile TO TRUE
    END-READ
END-PERFORM
```

Evolution of data representation: from file systems to databases

1950s–60s: Data became separated from programs and was organized into standalone files.

The first commercial software applications were developed for accounting purposes (*file folder* = physical paper folder).

- **Computational tasks:** program code + internal data (variables, arrays)
- **Data processing tasks:** program code + external data (files)

Database concepts emerged as a natural evolution of file systems.

Logical and Physical File Structure

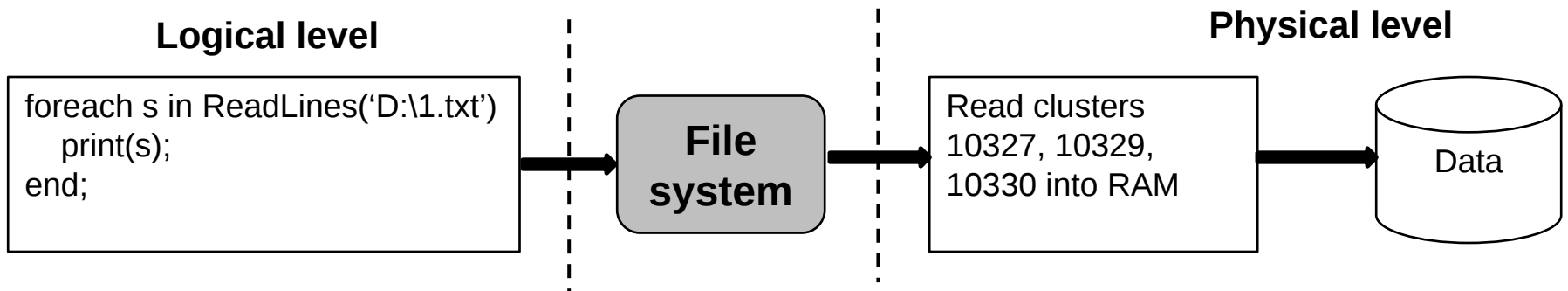
For an application program, a file is a **named region of external storage** that can be written to and read from.



On physical storage media, a file is represented as a chain of **clusters (physical records)**.

File System – for abstraction from data placement details

Users/programmers work only with logical data (a more convenient form) and do not deal with the details of actual low-level data placement.



C:\Documents\My file.doc

File directory

Дескриптор файла 1
...
Дескриптор файла N

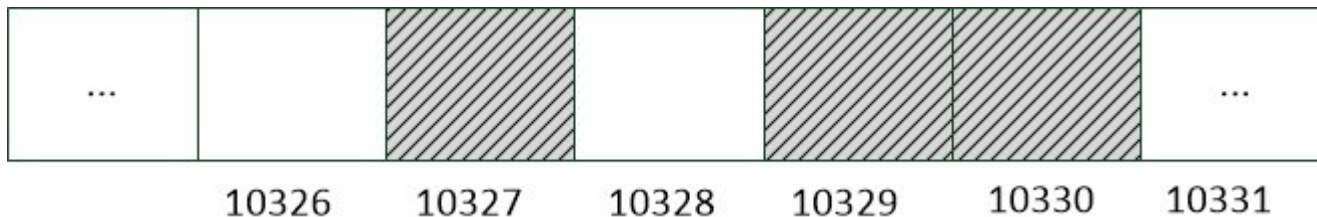
File descriptor

Name	<i>My file.doc</i>
Creation at	<i>01.02.2020</i>
Attributes	<i>Archive</i>
The first cluster	<i>10327</i>
Size	<i>9326</i>
...	...

File allocation table

№ кластера	Статус
10326	<i>Failed</i>
10327	<i>10329</i>
10328	<i>Free</i>
10329	<i>10330</i>
10330	<i>End of chain</i>
10331	<i>Free</i>
...	...

File Allocation Table (FAT)



Clusters on disk

File System – a component of the operating system that includes:

- The complete set of all files on disk along with their **physical organization**.
- File management data structures (file directories, file descriptors, file allocation tables, ...), i.e., the **logical organization of file structures**.
- A suite of system software tools that implement **file management** (creation, deletion, reading, writing, searching, and other file operations)

Core Functions of a File System

1. **Storage of Unstructured Data**

Files can contain any type of data—text, images, executables—without enforcing internal structure. The file system treats them as byte sequences.

2. **Hierarchical Directory Organization**

Files are grouped into folders (directories) arranged in a tree-like hierarchy, enabling logical categorization and easy navigation.

3. **Disk Space Management**

Tracks which disk blocks are free or allocated, and efficiently assigns space to new or growing files while minimizing fragmentation.

Core Functions of a File System

4. File Metadata Support

Maintains metadata for each file, such as name, size, creation/modification timestamps, ownership, and permissions.

5. Reliability and Recovery

Implements mechanisms (e.g., journaling, checksums) to protect data integrity and support recovery after crashes or power failures.

6. File/Directory-Level Security

Enforces access control through permissions (read/write/execute) for users and groups, ensuring data privacy and protection.

Core Functions of a File System

7. Flexible Data Access

Supports both **sequential access** (reading from start to end) and **random (direct) access** (jumping to any byte offset), enabling diverse application needs.

8. Abstraction from Physical Storage

Hides hardware-specific details (e.g., sectors, tracks) behind a simple logical interface—users and programs interact with files, not raw disks.

9. Support for Mounting and Diverse Storage Media

Allows integration of various storage devices (hard drives, SSDs, USB drives, network shares, optical discs) into a unified directory structure through **mounting**, enabling transparent access regardless of physical medium.

Let's write a console application for working with data in a file:

- Name, lastname, gender, age of students
- CRUD-operations

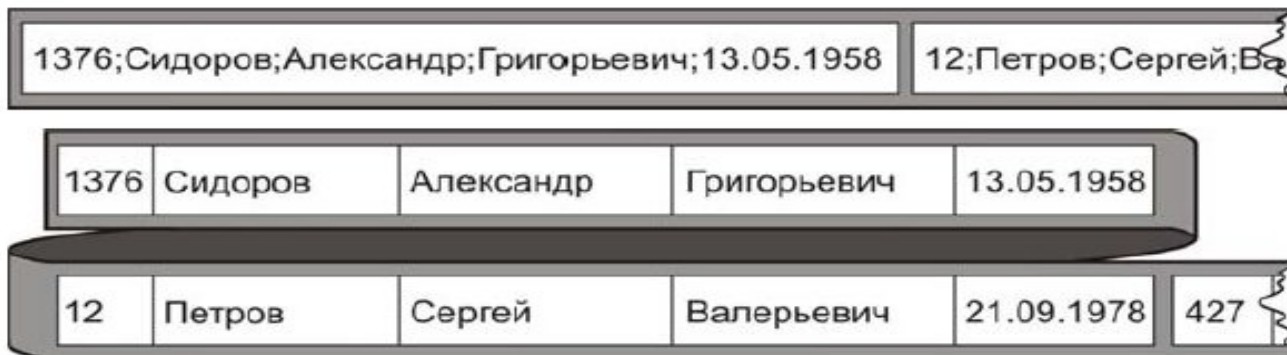
Create **R**ead **U**ppdate **D**elete

Logical records

id	surname	name	patronymic	birthday
1376	Сидоров	Александр	Григорьевич	13.05.1958
12	Петров	Сергей	Валерьевич	21.09.1978
...

Physical records

- A list of fields separated by delimiter characters.
- A list of fixed-length fields, where each field's length matches the width of the corresponding table column.



Structured Data - Language Support in Programming Languages

COBOL (Common Business-Oriented Language), 1959

Pascal allows defining composite data types (records), creating variables of those types, and storing them in a file.

```
Type Person = Record
  id: LongInt;
  surname: string[30];
  name: string[30];
  patronymic: string[30];
  birthday: string[10];
end;
var f: file of Person;
```

The file contains logical records composed of fields.

Let's write a console application for working with data in a file:

- Name, lastname, gender, age of students
- CRUD-operations

Free Pascal, 310 lines

Flat Structured Files

Flat files are the precursor to **databases**

- The file contains only raw data—no metadata about records is stored within the file itself.
- Record structure is defined and interpreted by the application program.

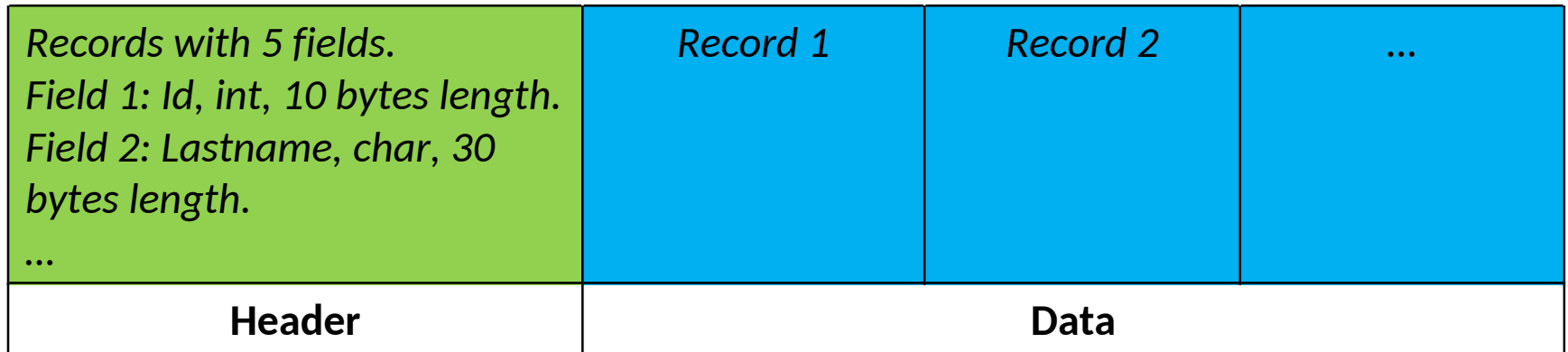
Disadvantages

- **Tight coupling** between the application and the physical file structure.
- Every new report or data requirement **requires changes to the application code**.
- **Security** must be implemented entirely in application logic.
- **Data integrity** constraints (e.g., validation, relationships) are enforced in code.
- **Administration tasks**—such as backups and recovery—must be handled programmatically.
- **Multi-user access** (concurrency, locking, etc.) must be manually implemented in the application.

Files with Metadata

A file consists of:

- A **header** containing metadata about the record structure (field names, sizes, and total number of fields);
- A **data area** storing records of fixed length.

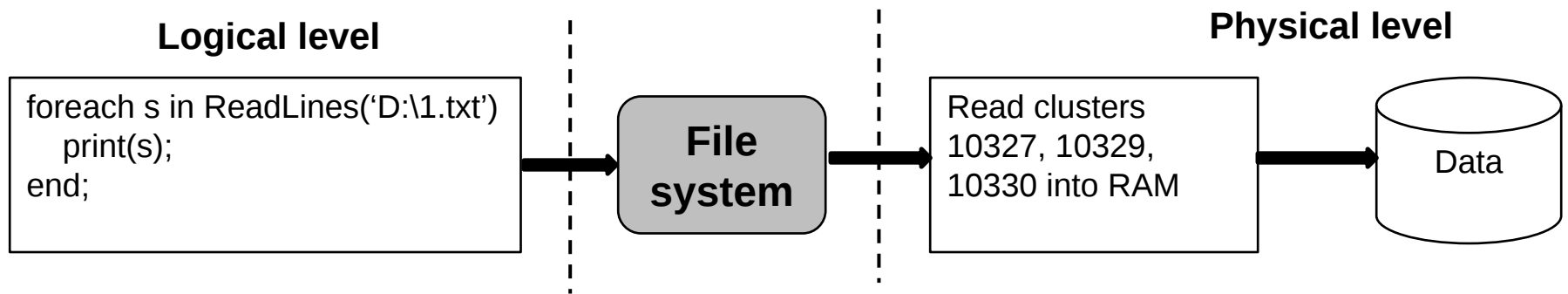


Advantage: The data structure is described **within the data file itself**, not hardcoded in application programs.

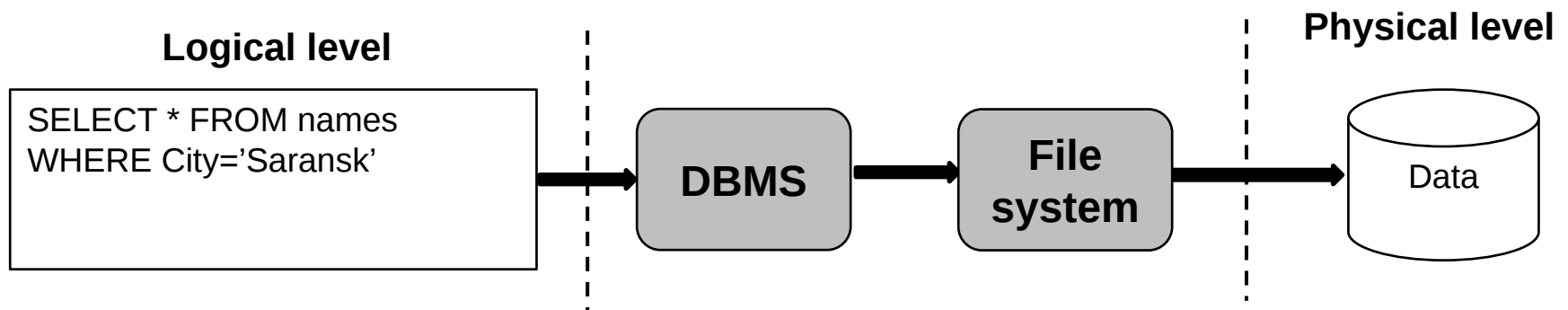
DBF files – a standard format used by early database systems on personal computers (e.g., dBASE, FoxPro).

A DBMS is the next step in abstracting the user/programmer from the data.

Working with a file via file system



Working with a database via DBMS



Let's rewrite our CRUD application using the SQLite DBMS.

Free Pascal, 287 строк

Data from Multiple Sources (Files)

Task 1. Personnel Records Management. File:

Lastname	Name	Surname	Birthday	Address	Position	Salary
Ivanov	Ivan	Ivanovich	01.02.1985	г. Саранск	Manager	30000

Task 2. Payroll Calculation. File:

Lastname	Name	Surname	Salary	Month	Days worked	Earnings
Ivanov	Ivan	Ivanovich	30000	February	10	15000

Task 3. Sick Leave Tracking. File:

Lastname	Name	Surname	Salary	Month	Sick Leave Days	Sick leave payment amount
Ivanov	Ivan	Ivanovich	30000	February	6	8500

Data from Multiple Sources (Files)

Task 1. Personnel Records Management. File:

Lastname	Name	Surname	Birthday	Address	Position	Salary
Ivanov	Ivan	Ivanovich	01.02.1985	г. Саранск	Manager	30000

Task 2. Payroll Calculation. File:

Lastname	Name	Surname	Salary	Month	Days worked	Earnings
Ivanov	Ivan	Ivanovich	30000	February	10	15000

Task 3. Sick Leave Tracking. File:

Lastname	Name	Surname	Salary	Month	Sick Leave Days	Sick leave payment amount
Ivanov	Ivan	Ivanovich	30000	February	6	8500

Information is duplicated – **this is bad!**

Data from Multiple Sources (Files)

Task 1. Personnel Records Management. File:

Lastname	Name	Surname	Birthday	Address	Position	Salary
Ivanov	Ivan	Ivanovich	01.02.1985	г. Саранск	Manager	30000

Task 2. Payroll Calculation. File:

Lastname	Name	Surname	Salary	Month	Days worked	Earnings
Ivanov	Ivan	Ivanovich	30000	February	10	15000

Task 3. Sick Leave Tracking. File:

Lastname	Name	Surname	Salary	Month	Sick Leave Days	Sick leave payment amount
Ivanov	Ivan	Ivanovich	30000	February	6	8500

Information is **uplicated** — data can become inconsistent.

Data should be **integrated** and stored in a single place.

Unified Information Repository

Option 1: Combine everything into a single file:

Names	Birthday	Address	Position	Salary	Month	Worked days	Sick Leave Days	Earnings	Sick leave payment
-------	----------	---------	----------	--------	-------	-------------	-----------------	----------	--------------------

Unified Information Repository

Option 1: Combine everything into a single file:

Names	Birthday	Address	Position	Salary	Month	Worked days	Sick Leave Days	Earnings	Sick leave payment
-------	----------	---------	----------	--------	-------	-------------	-----------------	----------	--------------------

Disadvantages:

- Data duplication will still remain within the file.
- The time required to solve Task 3 will increase significantly.

Unified Information Repository

Option 1. Combine everything into a single file:

Names	Birthday	Address	Position	Salary	Month	Worked days	Sick Leave Days	Earnings	Sick leave payment
-------	----------	---------	----------	--------	-------	-------------	-----------------	----------	--------------------

Option 2. Two files:

Lastname	Name	Surname	Birthdate	Saransk	Position	Salary
----------	------	---------	-----------	---------	----------	--------

Names	Salary	Month	Worked days	Sick Leave Days	Earnings	Sick leave payment
-------	--------	-------	-------------	-----------------	----------	--------------------

Option 3. Two files:

Id	Lastname	Name	Surname	Birtdate	Address	Position
----	----------	------	---------	----------	---------	----------

Id	Salary	Month	Worked days	Sick Leave Days	Earnings	Sick leave payment
----	--------	-------	-------------	-----------------	----------	--------------------

Database as a New Kind of Data

A database is a collection of interrelated data stored together with minimal redundancy, enabling optimal use by one or more applications.

ISO, 2015: A database is a collection of data organized according to a conceptual structure that describes the characteristics of the data and the relationships among them. Such a collection supports one or more application domains.

- Databases are essential when multiple tasks (or programs) need to share common data.
- The primary criterion for evaluating database performance is the response time of user queries.

Conclusions

- Databases are the result of the evolution of file systems.
- Information about data structure should be stored **in the database itself**, not in the application.
- Data duplication within a system is harmful, as it easily leads to violations of **logical integrity** (i.e., inconsistent or contradictory data).
Sick leave payment
- Data in a database is divided into interrelated parts. There are multiple ways to organize these relationships, and the goal is to minimize data redundancy.