Why Databases? – Roadmap

- what happens to data over its lifetime?
  → the data life cycle
- what goes wrong without a DBMS?
  → file-based system problems
- what does a DBMS actually give us?
  → DBMS advantages
- is a DBMS always the right answer?
  → tradeoffs/disadvantages

The Data Life Cycle

- creation
  – order placed at checkout
- processing
  – payment validated, inventory updated
- review/reporting
  – included in monthly sales reports
- retention/retrieval
  – customer looks up order status weeks later
- destruction
  – order purged per data retention policy after *N* years
- while this is often presented (and thought of) linearly, it is a cycle
  – data might be archived, reactivated, regenerated
- a database system exists to support this entire life cycle
  – not just “store data”

Before Databases – Managing Data With Files

Consider a small business with –
- a customers.txt file
- an orders.txt file
- an inventory.xlsx spreadsheet
- different departments maintaining their own copies

What can go wrong?
- redundancy and inconsistency
- integrity problems
- concurrent access anomalies
- isolation of data (hard to access/combine)
- security gaps

Before Databases – Managing Data With Files

What can go wrong?
- redundancy and inconsistency
  – same data stored in multiple places (e.g. customer address in both orders.txt and customers.txt)
  – update one copy, forget to update the other → data goes out of sync
- integrity problems
  – no built-in way to enforce rules like “an order must reference a real customer” or “inventory count can’t be negative”
  – nothing stops invalid or contradictory data from being written
- concurrent access anomalies
  – two employees editing inventory.xlsx at the same time – last save wins and the other person’s changes vanish
  – no coordination between simultaneous users

Before Databases – Managing Data With Files

What can go wrong?

- isolation of data (hard to access/combine)
  - data scattered across many separate files
    - even simple questions can require manually cross-referencing multiple files to retrieve the necessary information
  - no shared, structured way to query across data –
    - asking a question of your data
    - text files vs spreadsheets, different file formats even just for the text files
- security gaps
  - file permissions are coarse – read/write the whole file, or not
    - lacking finer-grained control like “Alice can see customer names but not credit card names”
  - no audit trail of who accessed or changed what

DBMS to the Rescue

A DBMS is built to directly solve each of these problems –

- redundancy and inconsistency
  - DBMS→ centralized, normalized data
- integrity problems
  - DBMS → constraints (keys, checks)
- concurrent access anomalies
  - DBMS → transactions, locking
- isolation of data (hard to access/combine)
  - DBMS → shared query language (SQL)
- security gaps
  - DBMS → fine-grained access control, auditing

DBMS Advantages

- reduced data redundancy
- improved data consistency and integrity
- concurrent access without anomalies
- efficient, standardized access (query language)
- centralized security and access control
- crash recovery
- data independence

Tradeoffs

- added complexity (installation, configuration, administration)
- cost (licensing, hardware, specialized staff – DBAs)
- overhead for very simple use cases
- performance overhead vs a hand-tuned custom file format, in narrow cases