Databases Are Everywhere

- Database = a large (?) collection of related data
- Classically, a DB models a real-world organisation (e.g., enterprise, university)
  - Entities (e.g., students, courses)
  - Relationships (e.g., “Martin is taking IDS in 2010/11”)
- Changes in the organisation = changes in the database
- Examples:
  - personnel records
  - banking
  - airline reservations

Scientific Databases (Examples)

- Biology:

  e.g., DNA sequences of genes, amino-acid sequences of proteins, genes expressed in tissues

  (up to several Gigabytes)

- Astronomy:

  e.g., location and spectra of astronomic objects

  (up to several Terabytes)

- Physics:

  e.g., sensor measurements in particle physics experiments

  (up to several Petabytes)

DB Tendencies

- Sensors record data
  - ➔ DBs grow in size
  - ➔ DBs become more widespread
  - ➔ date may be less reliable, i.e., uncertain
- Multimedia data
  - ➔ Requirements for larger storage
  - ➔ New query operations
    (e.g., find a song by humming the melody,
    find pictures with a given face)
- Data on the Web
  - ➔ Accessed/changed by many people (Facebook,…)
  - ➔ Speed up access, loosen consistency (NoSQL)

Operations with Databases

- **Design**
  - *Define* structure and types of data
- **Construction**
  - *Create* data structures of DB, *populate* DB with data
- **Manipulation of Data**
  - *Insert*, *delete*, *update*
  - *Query*: “Which department pays the highest salary?”
  - Create *reports*:

    “List monthly salaries of employees, organised by department, with average salary and total sum of salaries for each dept”

An Ideal DB Implementation Should Support:

- Structure
  - data types
  - data behaviour
- Persistence
  - store data on secondary storage
- Retrieval
  - a declarative query language
  - a procedural database programming language
- Performance
  - retrieve and store data quickly
- Data Integrity
- Sharing
  - concurrency
- Reliability and resilience
- Large data volumes

Database Management System (DBMS)

- A DBMS is a software package designed to *store* and *manage* databases

- A DBMS provides *generic functionality* (see previous slide) that otherwise would have to be implemented over and over again

  ➔ *Reduced application development time*

- Several brands, e.g.,
  - Oracle *X*i/Yg (Oracle), DB2 (IBM), SQL Server, Access (Microsoft), MySQL, PostgreSQL, HSQLDB, SQLite (open source)

Database Actors

Database Designers

Application Programmers

Database Administrator (DBA)

End Users
- **sophisticated**
- **casual**
- **'parametric' or 'canned' transactions**

Database

"on the scenes"

DBMS developers

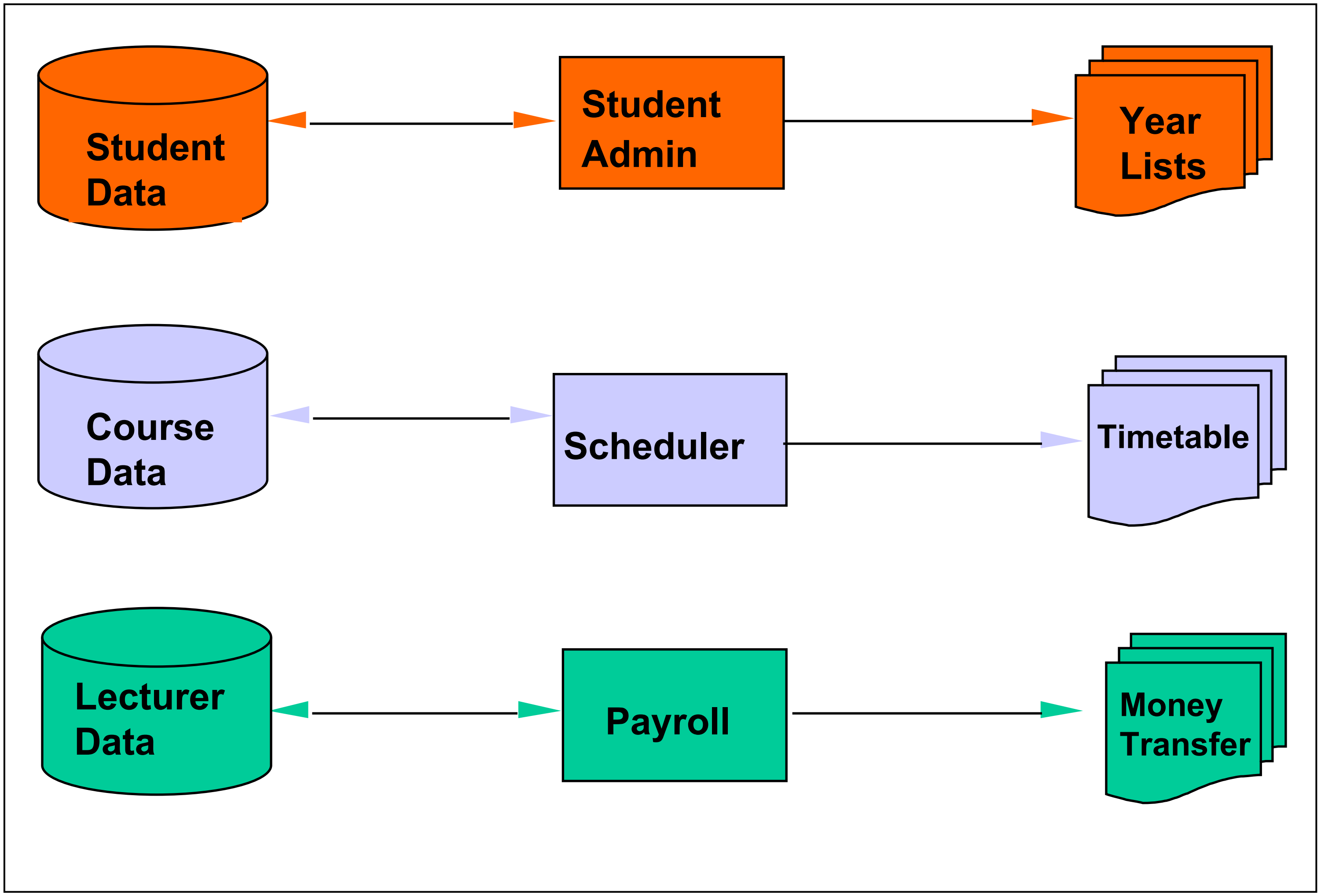
Tool Developers

Operators and Maintenance Personnel

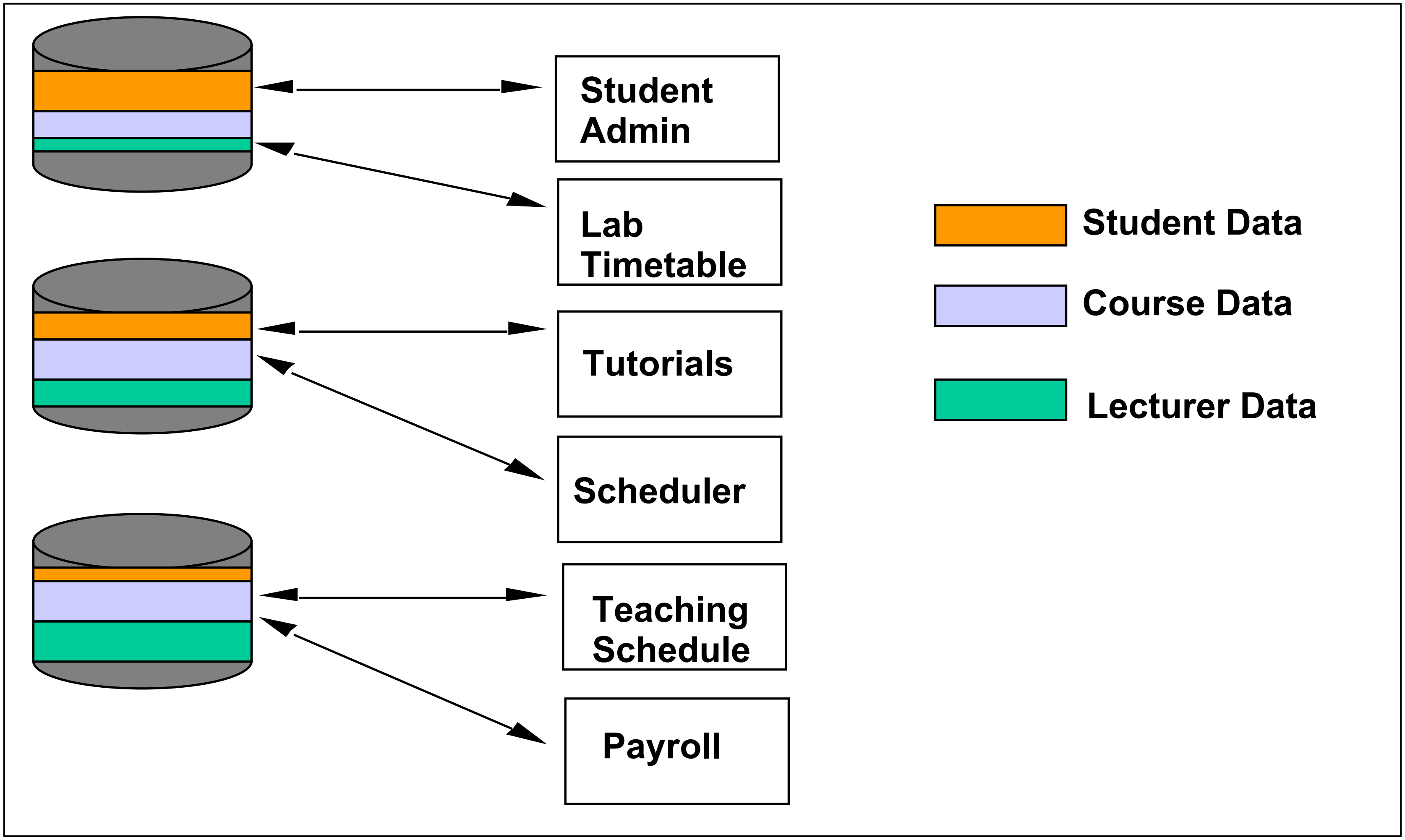
Database Management System

"behind the scenes"

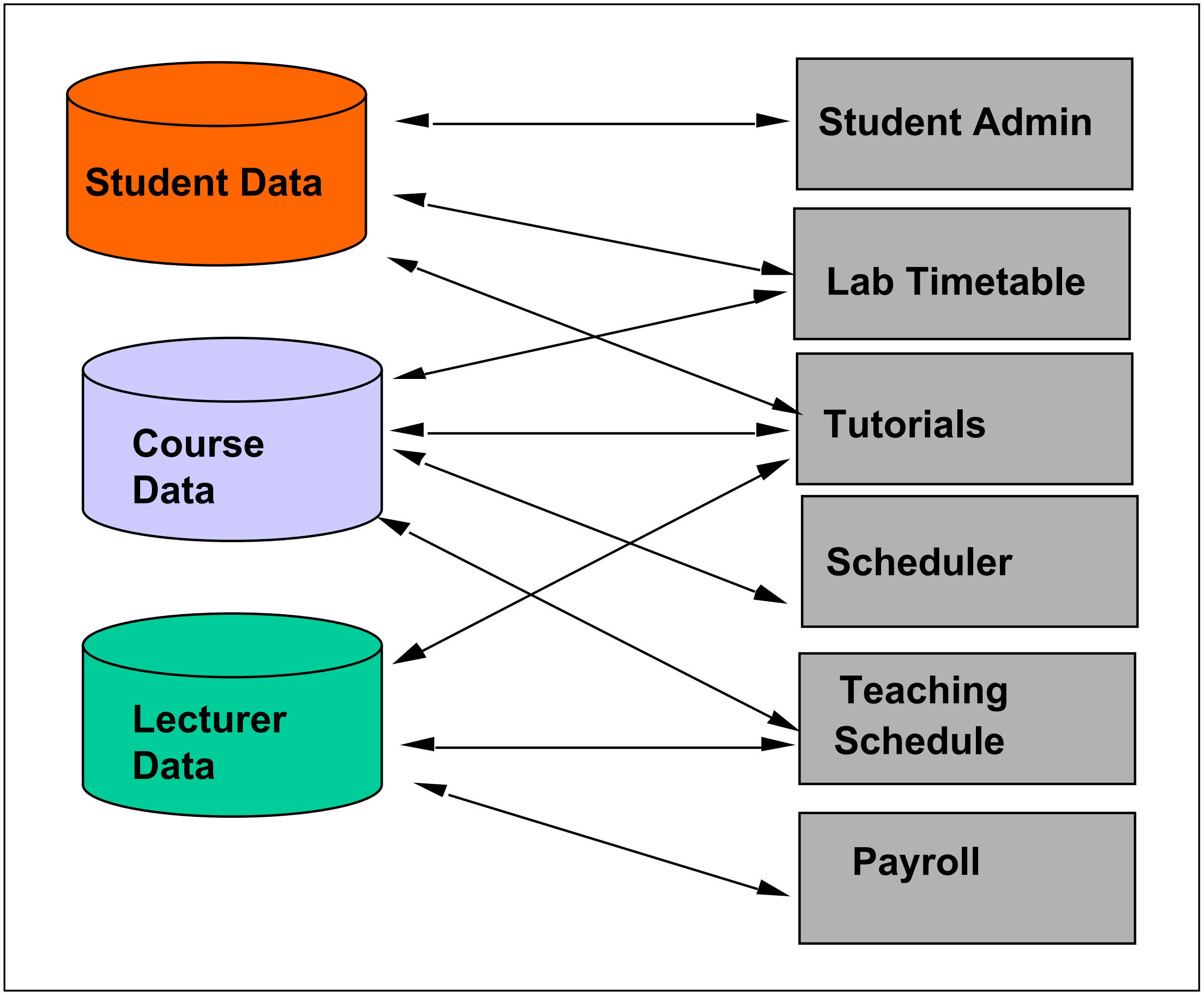
File System: A Physical Interface



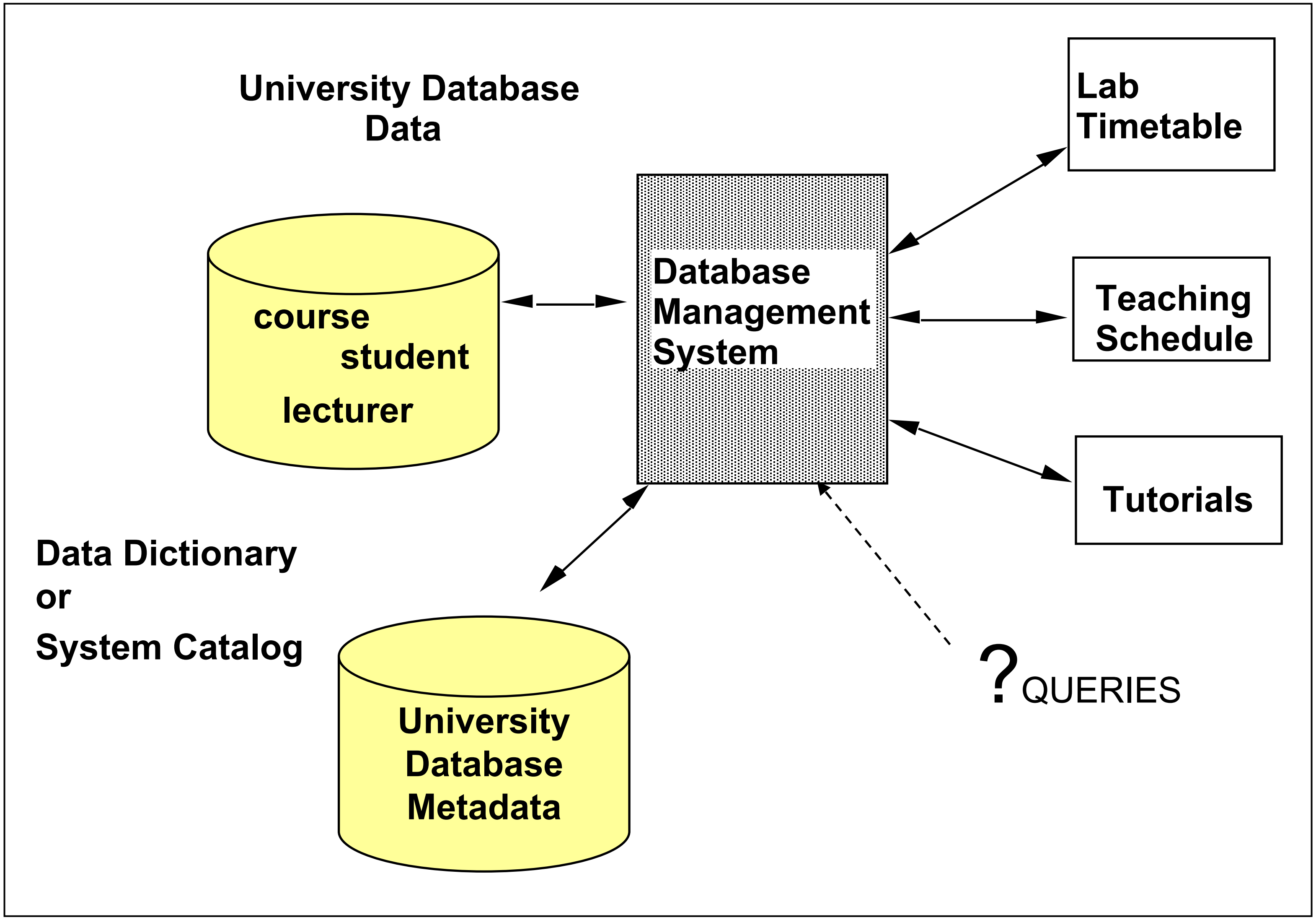
Sharing Data: Replication ➔ Redundancy



Sharing Data and Operations



DBMS: A Logical Interface



File System Approach

- Uncontrolled redundancy
- Inconsistent data
- Inflexibility
- Limited data sharing
- Poor enforcement of standards
- Low programmer productivity
- Excessive program maintenance
- Excessive data maintenance

DBMS Approach

- Controlled redundancy
  - consistency of data & integrity constraints
- Integration of data
  - self-contained
  - represents semantics of application
- Data and operation sharing
  - multiple interfaces
- Services & controls
  - security & privacy controls
  - backup & recovery
  - enforcement of standards
- Flexibility
  - data independence
  - data accessibility
  - reduced program maintenance
- Ease of application development

However....

If an application is

- simple
- stringent real-time
- single user
- static,

files are the option of choice

DBMS downside:

- more expensive
- more complex
- general

Summary:

- In a file system, data is *physically accessed* and *not integrated*
- In a DBMS, data is *logically accessed* and *integrated*:
  - query language
  - data dictionary