Database Vs. File Systems Approaches

- Abstraction
  - Data
  - Execution
- **Reliability**
- Efficiency/Performance

Reliability

- Enforcing *integrity constraints*
  - E.g., data type, relationship between values
  - Data in DB must satisfy the integrity constraints
  - Transactions are committed if they do not violate any integrity constraint
  - Integrity constraints are stored in the catalog
- Ensuring *data integrity* despite failures
  - Data are not lost when the system or a transaction fails for whatever reason

Reliability…

Ensuring *data integrity* …

- Security
  - Encryption & Private Information Retrieval
  - Authentication
  - Data Domains, Compartmentalization
- Access control
  - *Who* (user/role), *what* (data), *how* (operations)
  - Views and access permissions in the catalog

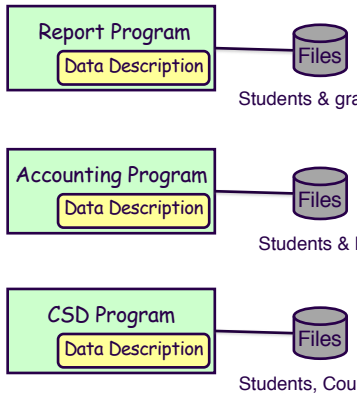
Example: Database Access ☺

"On Facebook, 273 people know I'm a dog.
The rest can only see my limited profile."

Database Vs. File Systems Approaches

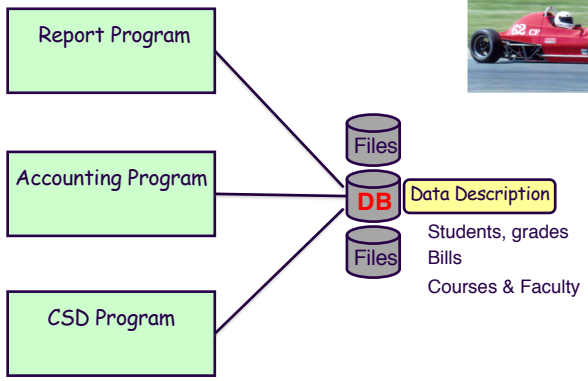
- Abstraction
  - Data
  - Execution
- Reliability
- **Efficiency/Performance**

Performance Problems with Files

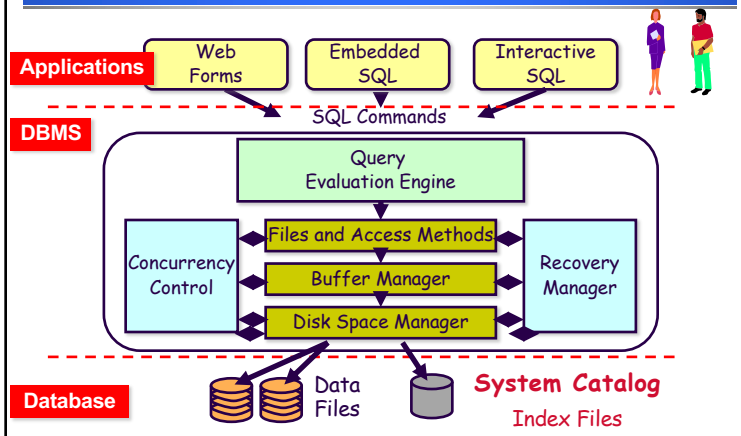


- **Redundancy**
  - waste of space
- **Waste of effort**
  - Same change in multiple places
- **Need for coordination**
  - inconsistent data
  - independent updates
  - E.g., Jan-19-1974 vs. Jan-29-1974

No Performance Problems with DBs



Database Management System (DBMS)



Efficiency and Performance

- Space efficiency:
  - minimizes data redundancy by storing data only once
- Time efficiency (response time):
  - eliminates the need for multiple updates to keep the replicas consistent and up-to-date
  - Enhances query performance by means of ***optimizations*** and ***access methods***
  - Allows many users (transactions) to access and share the database ***concurrently***

Performance Requirements

- Abstraction
  - Data abstraction
  - Execution abstraction
- Reliability
  - High availability: recovery time is *short*
  - Trusted/Quality data
- Efficiency/Performance
  - High throughput (Committed transactions per unit time)
  - Short or bounded response time
  - Energy Efficiency

When an SQL-DBMS is Inappropriate?

- Disadvantages:
  - Price to buy (DBMS & Hardware)
  - additional expertise (SQL/DBA)

- Hence, it is ***over-kill*** when
  - the database has simple structure and/or its size is small
  - the application is simple, special purpose and is not expected to change
  - Concurrent, multiple-user access is not required
  - Can tolerate failures
  - Monitor applications, high volume of updates

When a DBMS is needed?

- To integrate efficiently and correctly large volumes of related data
  - Central control, tuning for better performance, security
- Building of large applications
  - New applications using a DBMS is estimated 1/6 to ¼ of the time of application using a file system
  - Economy of scale
- Expandability/Flexibility
  - Supports evolution without affecting existing applications
- Supports sharing and discovering of information
  - Data mining, deductive databases, active databases
- DBMS ensure true QoS