# Databases and Database Users

Database -> essential to everyday life now   
traditional:

computerized library, banking, reservation, …  
new:  
-> social media -> proliferation of huge nontraditional database = big data/NOSQL systems  
-> cloud storage

Other applications of databases:  
-> Multimedia databases: Photo/Video/Audio cuz phones got so gud  
-> Geographic information systems: analyze maps/weather data/satellite  
-> Data warehouses & online analytical processing:  
-> RT-Active database: industrial/manufacturing control

## Introduction

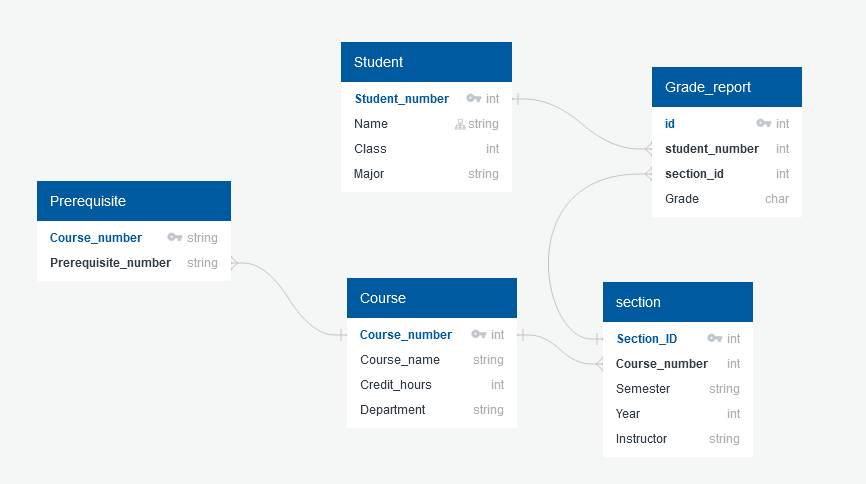
Database = collection of related data  
-> represents miniworld/universe of discourse  
-> logically coherent collection of data with inherent meaning  
-> specific purpose, intended group of users  
data = recorded facts with implicit meaning -> phonenumber/Name/…; can be manipulated  
physical example: library card catalog

Database management system (DMBS)  
= computerized system that enables users to create/maintain database

* Defining: def. datatypes/structure/constraints/information etc. = metadata
* Constructing: storing data
* Manipulating: queries; retrieve data, update database
* Sharing: multiple users at once
* Protection: system protection(crash/malfunction in SW/HW), security protection: unauth. Access
* Maintain: database evolves as requirements change

Application program used for that

-> complex systems  
database system = DB + DBMS



## Example

University:

Definition on the right: + Relationships!

Construction: Storing data of each student/course/grade/…

-> Example:

- Smith in Student\_file

- 2 records in Grade\_reoport  
- 2 courses taken relate to prerequisite\_file

Manipulation: querying/updating data  
Example: retrieve all courses smith did; Enter grade A for Smith for specific course

Next steps: requirement analysis -> conceptual design (+changes) -> physical design (=implementation)

## Characteristics of Database Approach

Database approach: single repository maintains data once defined; characteristics:

### - self describing nature:

-> complete definition of database structure + constraints (stored in meta-data); DMBS + users access that information  
Very important for other programs/databases: DBMS works universal  
Traditional file processing: data definition is typically part of application program  
DBMS can read that definition and use it for itself to extract DBs

In example: name of student referred + position and size of data item retrieved instead of each program used to open a specific location that stores the name of a student

### Insulation between Programs and Data & Data Abstraction

Traditional file processing: data files embedded in application program -> change to structure of file = changing all programs that access file

DBMS: structure of data files stored in DBMS catalog -> program-data-independence

Operation: interface(signature) = name + data types of args; implementation (method) -> specified separately & can be changed w/o affecting interface

= Program-operation independence

DBMS -> provides conceptual representation of data without details how data is stored

Data model = type of data abstraction -> provides conceptual representation  
-> uses logical concepts (objects & their properties/interrelationships) = hides storage and implementation details to DB-user; stored in the catalog

Object-oriented & object-relational database: Abstraction includes data structure + operations on data;  
= miniworld activities f.e. calculate GPA

### Support of multiple views of the data

Different users need different views of DB -> view= subset of DB/data derived from DB

### Sharing of Data and Multiuser Transaction Processing

DBMS -> concurrency control software = control process of multiple users updating same data -> updates must be correct. F.e. multiple users try to assign seat on airplane, only 1 agent per seat -> DMBS resolves that = “online transaction processing”

Transaction = executing program/process, reading/updating db records

Isolation -> each Transaction done weven though hundreds may be executed concurrently

Atomicity -> all operations in in Transaction executed or None

## Actors on the Scene

### DB-Admin

* Authorizes access to DB
* Coordinates and monitors it’s use
* Acquires software&hardware

### DB-Designer

* Identifies data to be stored in DB + chooses structures to represent it
* Understand requirements create design
* Develops views of DB + analyzes and integrates from other user groups

### End Users

Job dependent on access to DB for update/querying/generating reports

* Casual end user: occasional use of DB, different each time; ->managers
* Naïve/parametric end user: constantly querying and updating DB = canned transactions
  + Bank customers/tellers check account balances & stuff
  + Reservation agents
  + Employees at receiving stations for shipping
  + Social media user posts
* Sophisticated end users: engineers, scientists, analysts -> DBMS

## System Analysts and Application Programmers (Software Engineers)

System analysts: requirements of end users

Application programmers: implement specifications + test/debug/document/maintain -> familiar with capabilities of DBMS

## Workers behind the scene

DBMS system designers and implementers: create modules: query language, interface, accessing/buffering data,…

Tool Devs: design & implement tools -> software packages facilitate DB modeling & design (optional packages)

Operators and maintenance personnel: responsible for actual running & maintenance of hardware/software environment

## Advantages of using DBMS Approach

### Control redundancy

Traditional software dev with file processing:

* 1 update requires to update multiple dbs, everywhere where data is recorded
* Storage space wasted
* Inconsistency

DBMS:

* Views of different user groups integrated in DB design
* Data normalization = only one place in database
* Controlled redundancy: saving data in multiple spots = denormalization -> DBMS should prohibit inconsistencies among files

### Restricting Unauthorized Access

Some data is restricted/confidential like salaries -> DBMS provides security & authorization subsystem, DBA is managing it with privileged software

### Providing Persistent Storage for Program Objects

Object-oriented database systems: Provide persistent storage for program objects/data structure  
To save data from programs -> permanent files generated (structs/class) -> to access: need to be converted back to readable file (DBMS does this automatically cuz compatibilty) -> object becomes persistent = it survives termination of program execution

### Providing Storage Structures and Search Techniques for Efficient Processing

To efficiently execute queries and updates, databases use special data structures + indexes to improve efficiency  
-> buffering/Caching module that maintains parts of database in main memory buffers

Query processing and optimization module responsible for efficient query execution

### Providing Backup and Recovery

Backup & Recovery subsystem responsible for recovery

Computer system crashes/fails in middle of complex update transaction -> reset to previous state before update was made

### Providing multiple user interfaces

Multiple users -> different UIs needed = GUI

* Programming language UI
* Query language UI
* Apps for casual user
* …

### Representing Complex Relationships among data

Capability to represent complex RS among data

Can define new RS if needed

Retrieves & updates related data

### Enforcing Integrity Constraints

Integrity Constraint -> simplest form = data type definition.

More complex type of constraint = referential integrity (=one file must rely on another)

Others: uniqueness on data item values -> Key constraint

Derived from semantics of data and miniworld

Data item entry can be wrong, but satisfy condition of input -> no error, manual correction needed; if outside of def: automatic correction

### Permitting Inferencing and Actions Using Rules and Triggers

Database system can include deductive rules for inferences from new information on existing data

Example: complex ruleset for student on probation,  
-> declaratively as rules  
traditional DBMS: explicit procedural program code written to support applications; -> if miniworld rules change -> easier to change code than DB

Today: associate triggers with tables -> performs some additional operations to other table.

### Additional Implications of Using Database Approach

* Potential of Enforcing Standards -> DBA controls centralized database and can enforce standards easier than on individual file systems
* Reduced Application Dev Time -> new apps based on data easier to build rather than an entire new multiuser database
* Flexibility -> new user group may emerge that needs information not currently in database-> file can be added or created; or requirements of facility changes
* Availability of up-to-date Information: users update applies to the entire database; all users can see changes immediately
* Economies of scale: instead of investing in low-tier tech for each individual, investment in powerful switches