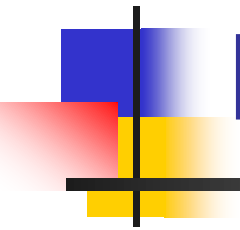
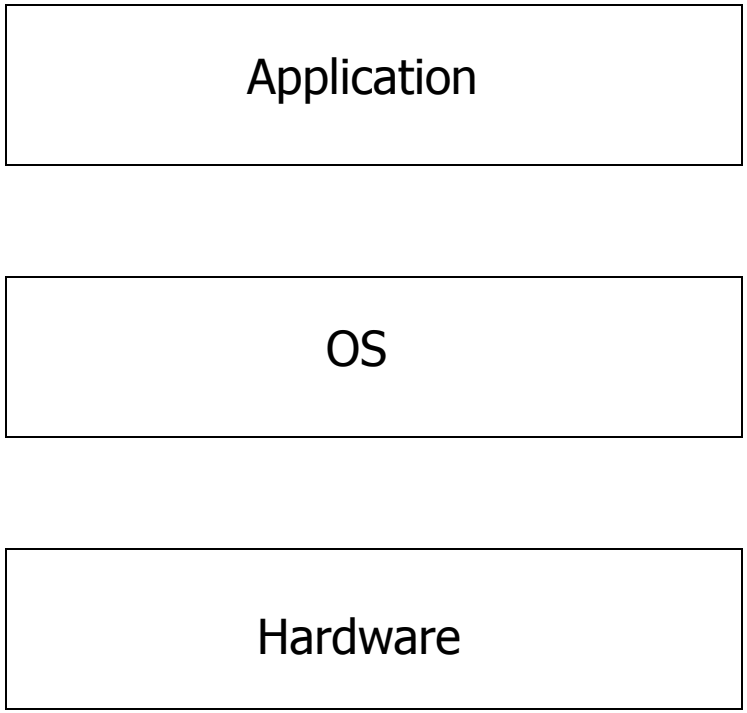
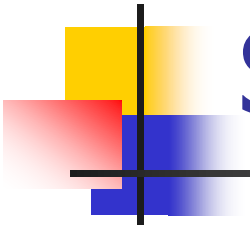


Operating System Support for Database Management

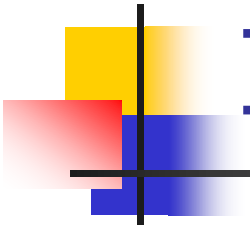
What is an operating system?





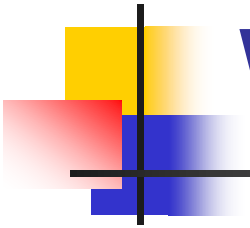
So what is the implication

- Design decisions affected
- Consider the following
  - Page replacements
  - File Systems
  - Scheduling
  - Process management & IPC
- Designed to work well in general



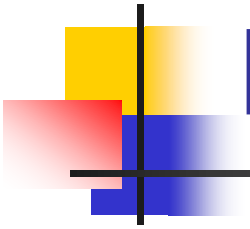
Is this enough?

- "Systems" does not mean Operating Systems
- Many other kinds of systems
  - Network Systems
  - Database Systems
  - Pub/Sub systems
- These also interface between hardware and users
- But here users have different requirements



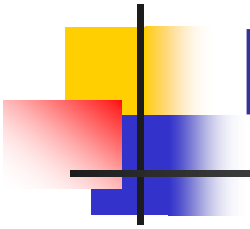
What's the insight?

- Many systems have specific properties
- General purpose operating systems principles work bad
- Why must I know this
  - Understand why general purpose operating systems are designed such
  - Understand that the principles might fail in many cases
  - Must understand the requirements of the system being designed
- Let us look at this from a database perspective



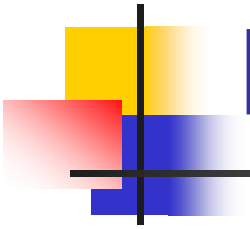
Lets take a detour

- What is a database system
  - Set of relations describe a schema
  - Each relation corresponds to a table on disc
  - Let us look at an example
    - Company X's employee database



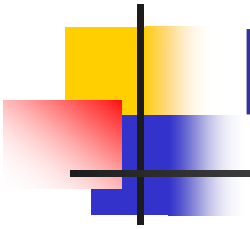
Example Contd….

- Two relations
  - Emp(id, name, age, sex)
  - Dept(id, dept_name, address)
- Each relation consists of a set of records
- These records are on disc
- Data size can be in the order of terabytes
- The two relations comprise the schema



How is this data used?

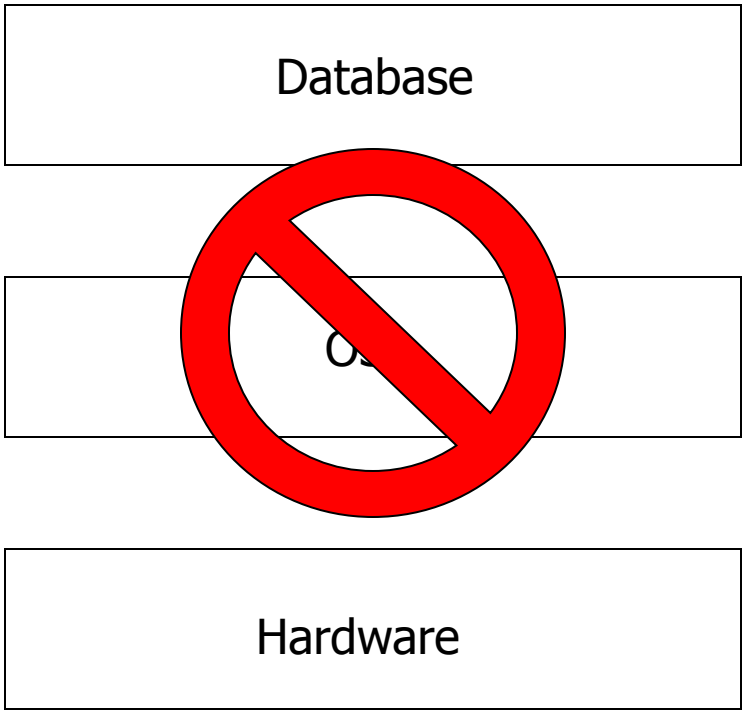
- Query the data
  - Select: Find all employees with age greater than 28
  - Project: Find the names of all the employees
  - Join: Find the departments in which John and Mary work

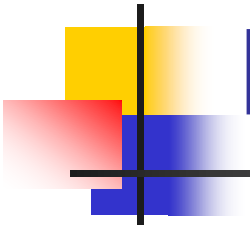


Database indexes

- Used to process queries efficiently
- Two kinds
  - B+ tree
  - Hash index
- Select: Use B+ index
- Project: Scan the relation sequentially
- Join: Loop over Dept for every id

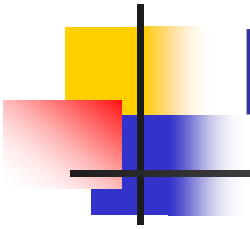
Ok so how do you build such a system





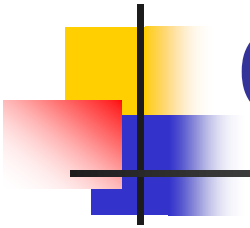
Buffer Pool Management

- Main memory caches in file systems
  - LRU
  - Prefetching
- Kinds of accesses databases see
  - Sequential with no rereference
  - Sequential access with cyclic rereference
  - Random access with no rereference
  - Random access with rereference



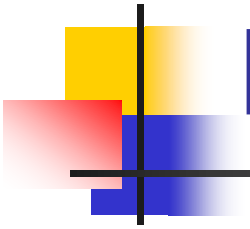
Design decisions

- LRU good only for case 4
- When you see a query you know the way it will access the relation
- Buffer manager follows policy according to query



Crash recovery

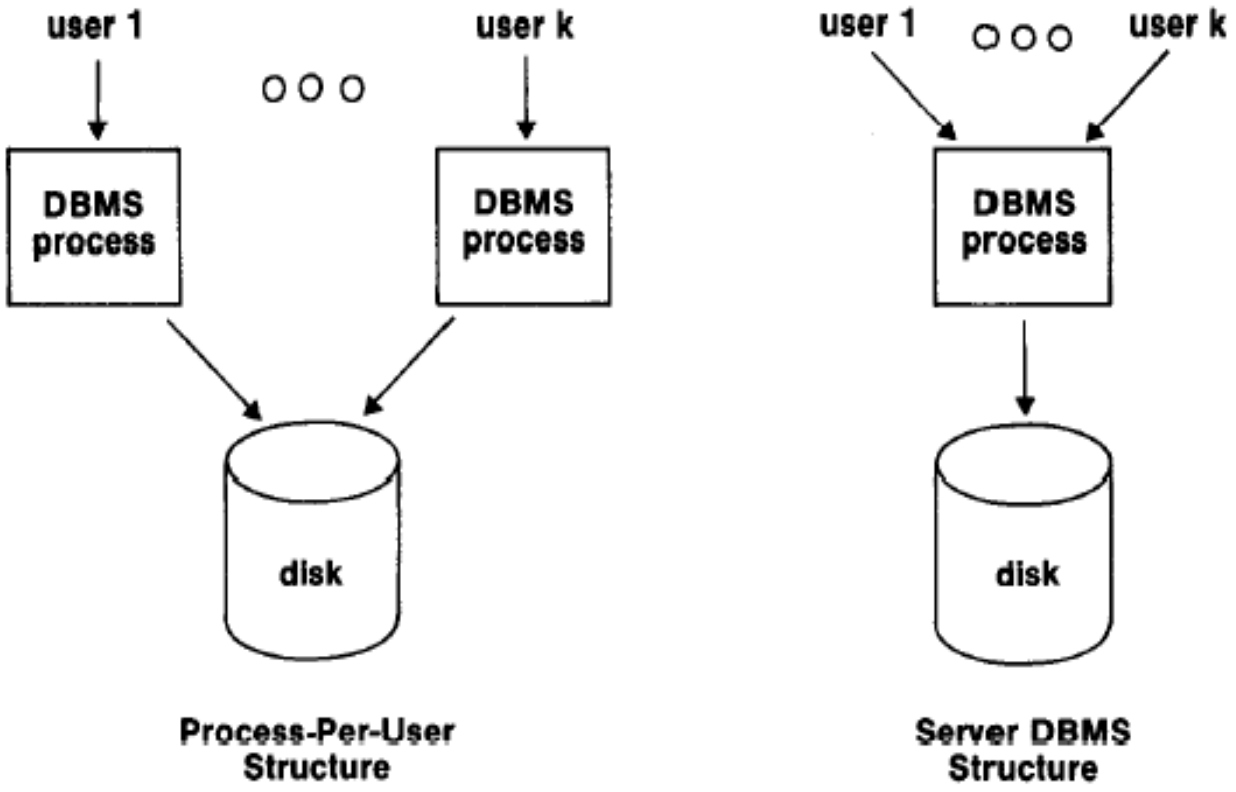
- Need to provide recovery from hard and soft failures
  - Finished transactions must be recovered
  - Unfinished must be undone
- What is required for this
  - Pages that finished transactions touch must be flushed to disc
  - Many OS's you cannot do that

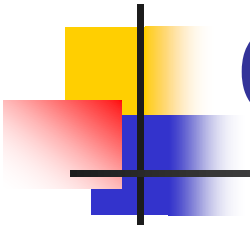


File System

- Tables stored as Unix files
- Files extend block at a time
  - Blocks scattered all over disc
  - Why is this bad for databases?
- File blocks stored as a tree in an inode
  - Don’t B+ trees do something similar
  - Can combining both give better efficiency

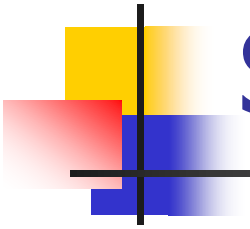
Two approaches for running a multi-user database system





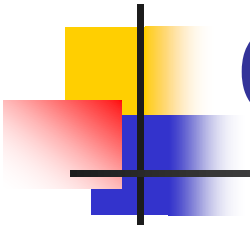
One process per user

- Easy to implement in Unix
- Problems
  - Buffer miss will cause a task switch
  - Round robin scheduling can remove a database process
  - Not good for performance as queries line up



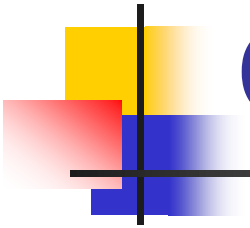
Server Model

- What facility must the OS provide?
  - n Processes send messages to one destination
- Even then server must take care of
  - Scheduling
  - Multitasking
- Duplication of OS services



Other issues

- Fine grained locking
  - Provided: Locking at level of files
  - Needed: Locking at level of records



Conclusion

- Operating system services inappropriate for database systems
- Before designing any system
  - Carefully study its requirements
  - Then decide how best to build it
- Welcome to the world of systems !!!