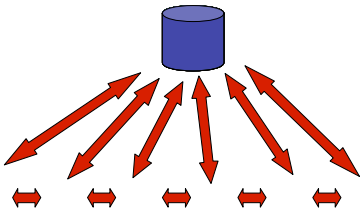
Parallel and distributed databases

R & G Chapter 22

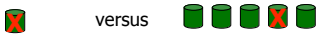
What is a distributed database?



Why distribute a database

- Scalability and performance

- Resilience to failures



Why distribute a database

- Data is already distributed
  - Or needs to be distributed

- Data is in multiple systems

amazon.com efollett.com iUniverse

BORDERS. BARNES&NOBLE

Why not distribute a database

You must earn your complexity!

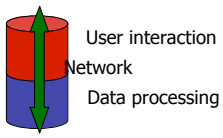
- Communication needed
  - Must build a complex infrastructure
  - Unpredictable latencies must be masked
- More types of failures
  - More components to fail
  - Network failures
  - Congestion, timeouts
- More complex planning
  - Communication cost plus I/O cost
- May have to deal with heterogeneity
  - Different types of systems
  - Different schemas, possibly incompatible
  - Different administrative domains

Types of distributed databases

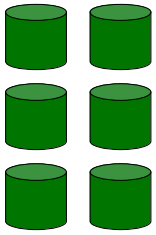
The old days: mainframes

Definitely not distributed!

Client-server

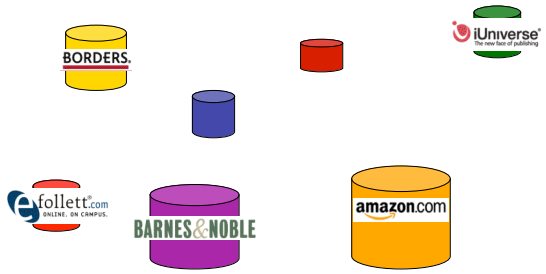


Parallel database



Primary/secondary

Multidatabase

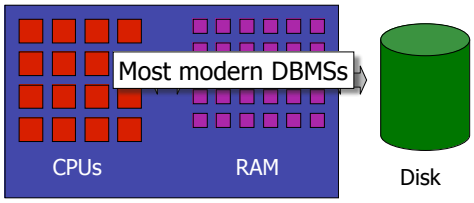


How do they work?

- What is shared?
- How to distribute the data?
- How to process the data?
- How to update the data?

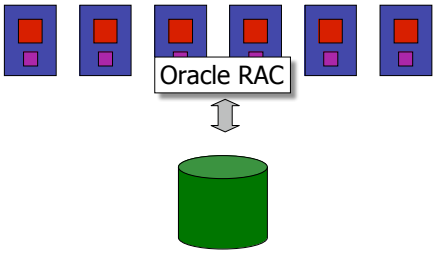
What is shared?

- Memory



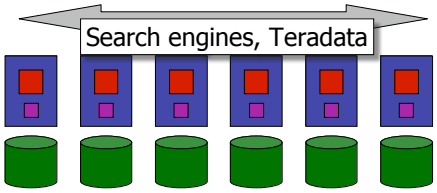
What is shared?

- Disk



What is shared?

- Nothing



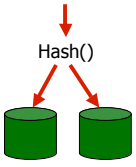
How to distribute the data?

6/1/07	424252	Couch	$570
6/1/07	256623	Car	$1123
6/2/07	636353	Bike	$86
6/5/07	662113	Chair	$10
6/7/07	121113	Lamp	$19
6/9/07	887734	Bike	$56
6/11/07	252111	Scooter	$18
6/11/07	116458	Hammer	$8000

How to distribute the data?

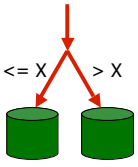
Hash partitioning

(key,value)



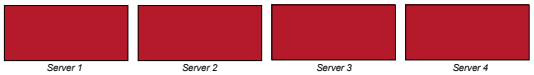
Range partitioning

(key,value)



How to distribute the data?

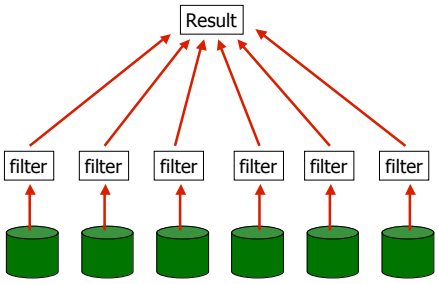
6/1/07	424252	Couch	$570
6/1/07	256623	Car	$1123
6/2/07	636353	Bike	$86
6/5/07	662113	Chair	$10
6/7/07	121113	Lamp	$19
6/9/07	887734	Bike	$56
6/11/07	252111	Scooter	$18
6/11/07	116458	Hammer	$8000



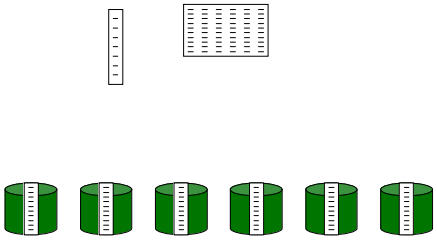
Query processing

- Intra-operator parallelism
- Inter-operator parallelism

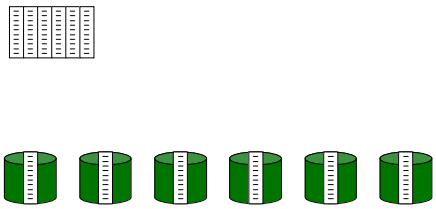
Parallel scanning



Sorting

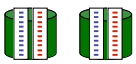


Sorting



Parallel hash join

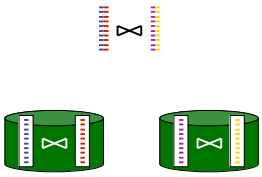
Hash()



Join

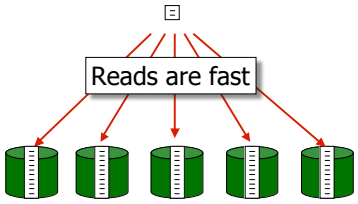
Semi-join

Inter-operator parallelism



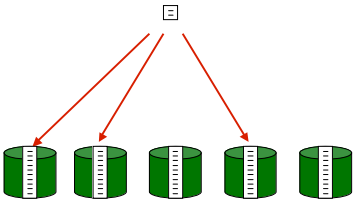
Updating distributed data

- Synchronous: read-any-write-all



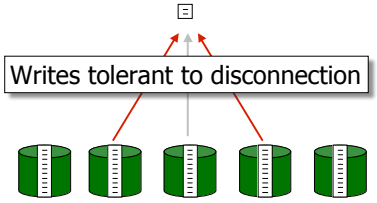
Updating distributed data

- Synchronous: voting



Updating distributed data

- Synchronous: voting

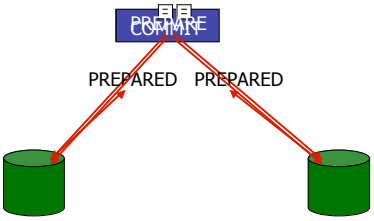


Consistency of distributed data

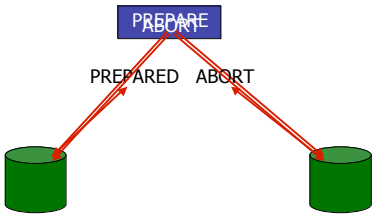
- Should provide ACID

Primary/secondary

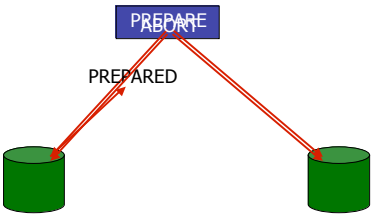
Two-phase commit



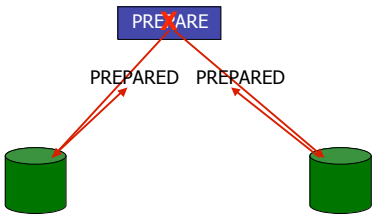
Two-phase commit



Two-phase commit



Two-phase commit



Conclusion

- Parallelism and distribution very useful
  - Performance
  - Fault tolerance
  - Scale
- But complex!
  - Rethink lots of aspects of the system
  - Must earn the complexity