COMP207
Database Development

Lecture 17

Review of query processing

and introduction of Distributed Databases

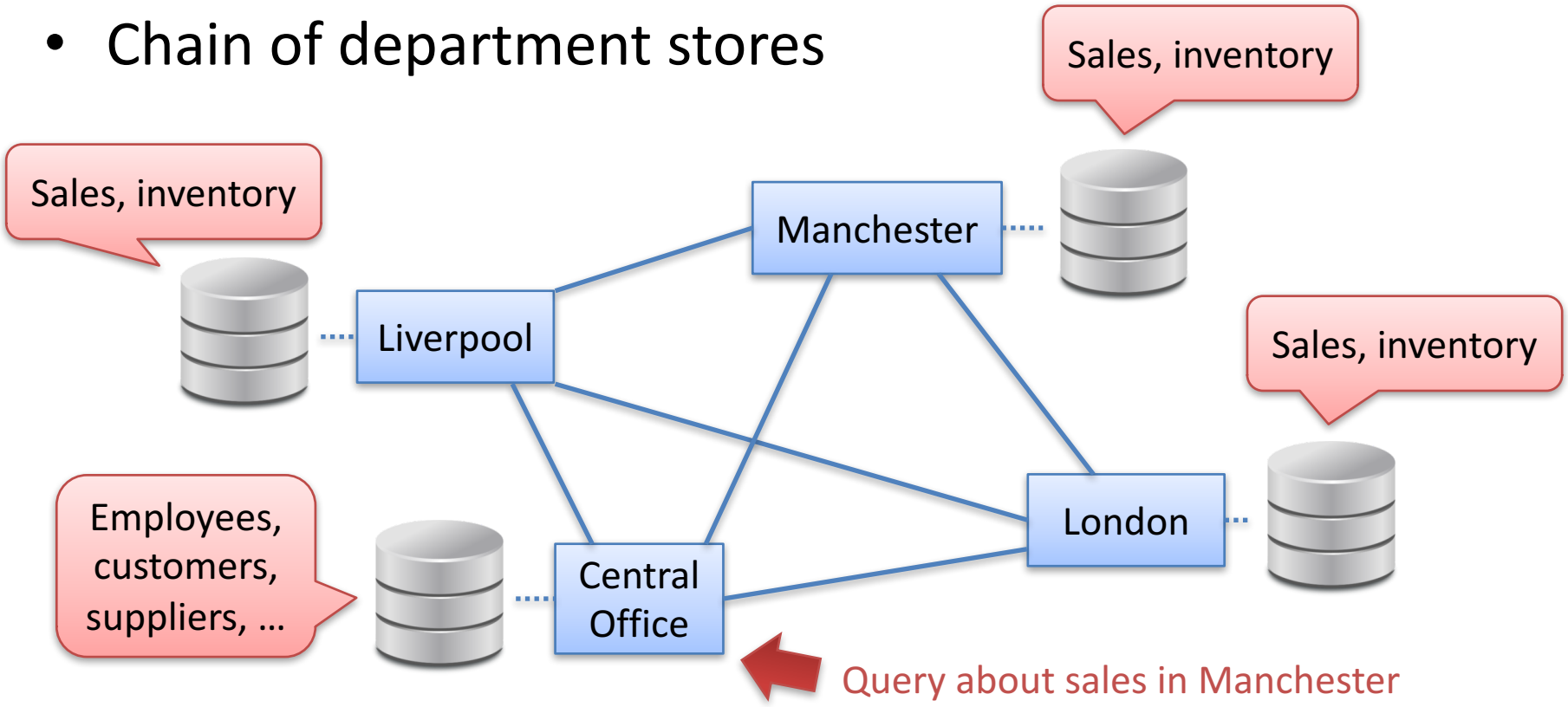
Chapter 4

- Extensions of the simple relational database setup
- **Distributed databases**
  - Databases connected through a network
  - More efficient by distributing tasks over several computers
  - + other desirable properties
- **Digression: Map-reduce**
  - Framework for parallel processing of data
  - Goes beyond relational data
- Next chapter: continues with non-relational data

What is a Distributed Database?

Motivation

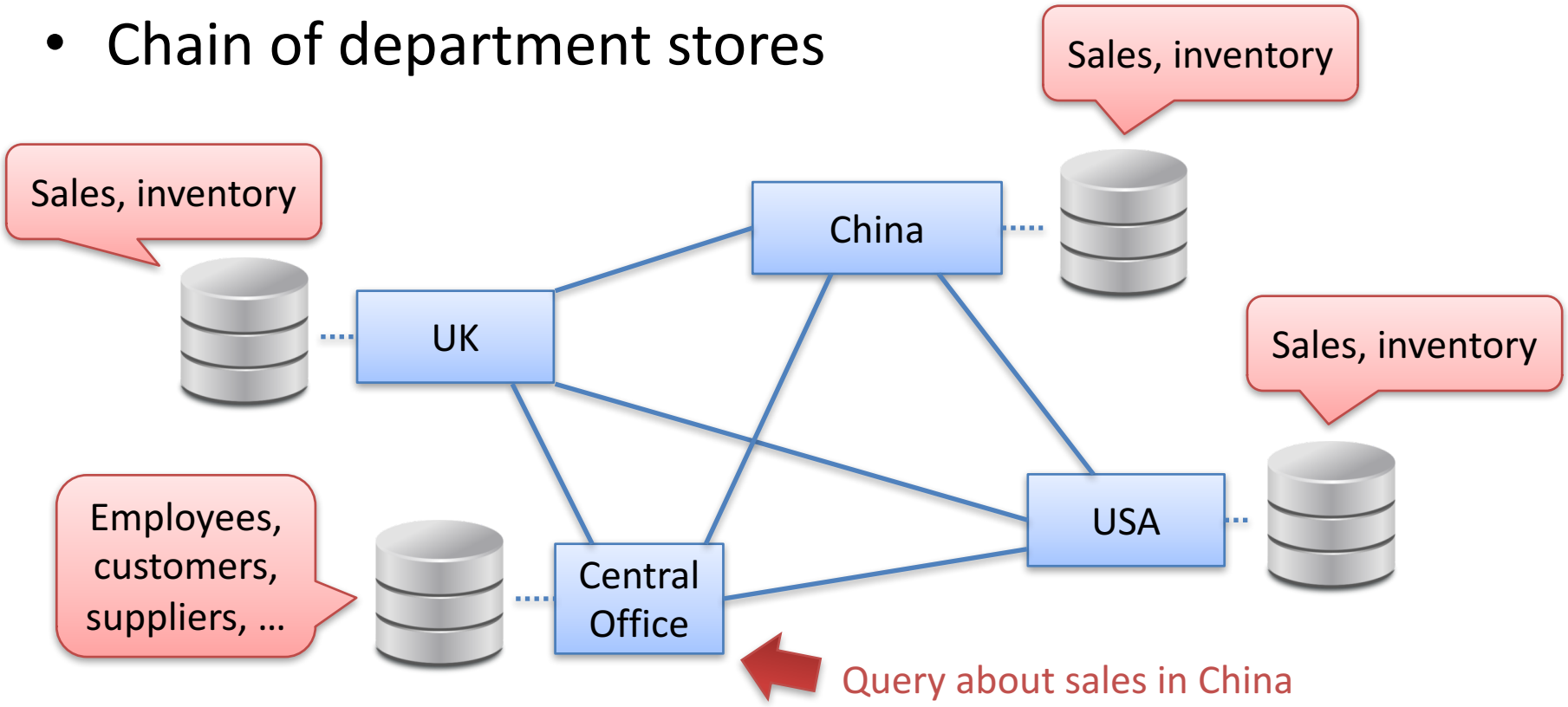
- Chain of department stores



- Each site stores only data primarily relevant to it
- Distributed DBMS provide access to data at all sites

Motivation

- Chain of department stores

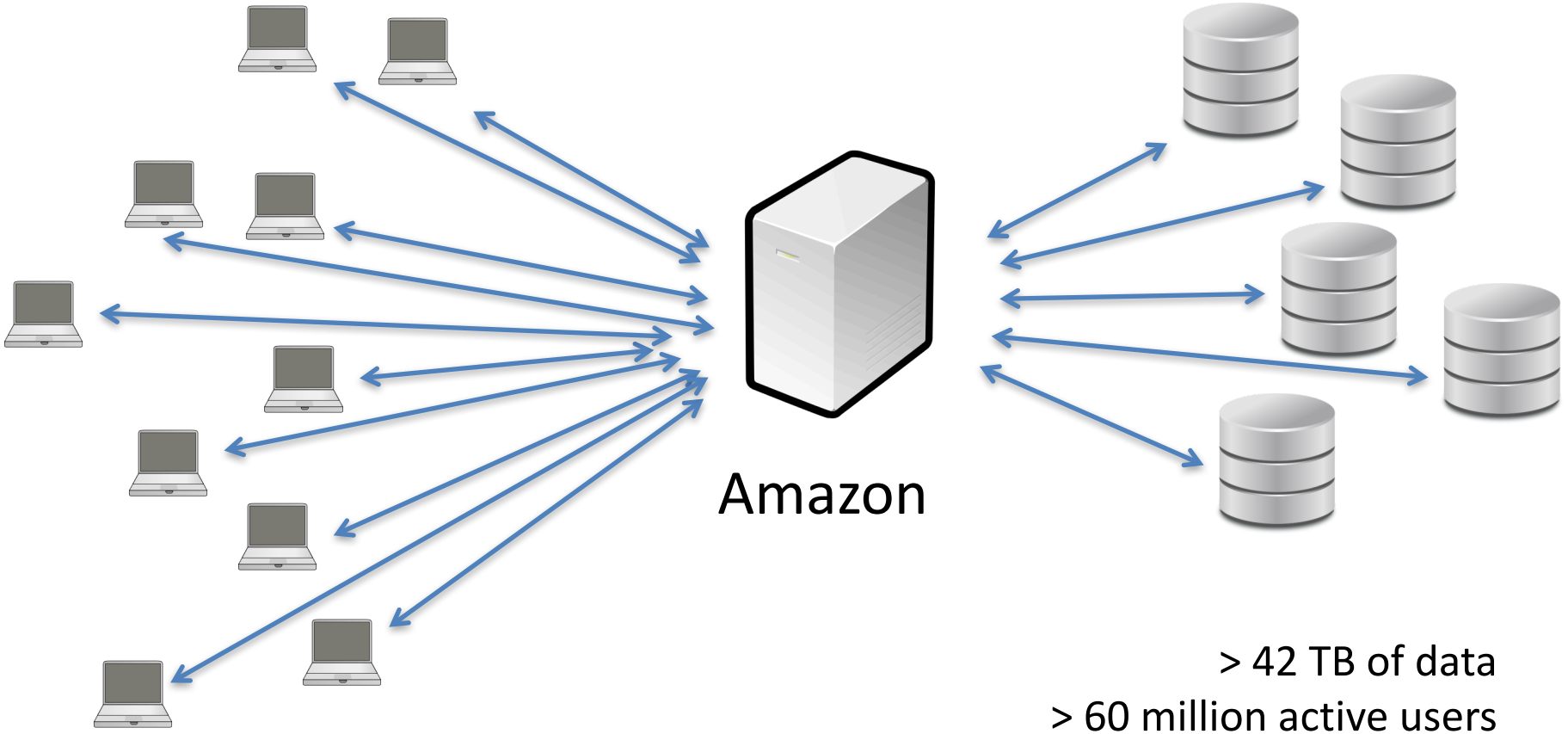


- Each site stores only data primarily relevant to it
- Distributed DBMS provide access to data at all sites

Other Applications

- More general: large organisations/companies
  - …with different branches or offices
  - …with different sub-companies
- Providing access to large datasets to many users (e.g., for an online store)
  - Distribute data over several computers
  - Computers could be at geographically separate locations
  - Possible advantages:
    - Balance workload & network traffic
    - Easier to extend capacity or scale to higher number of users

Recall Lecture 1…



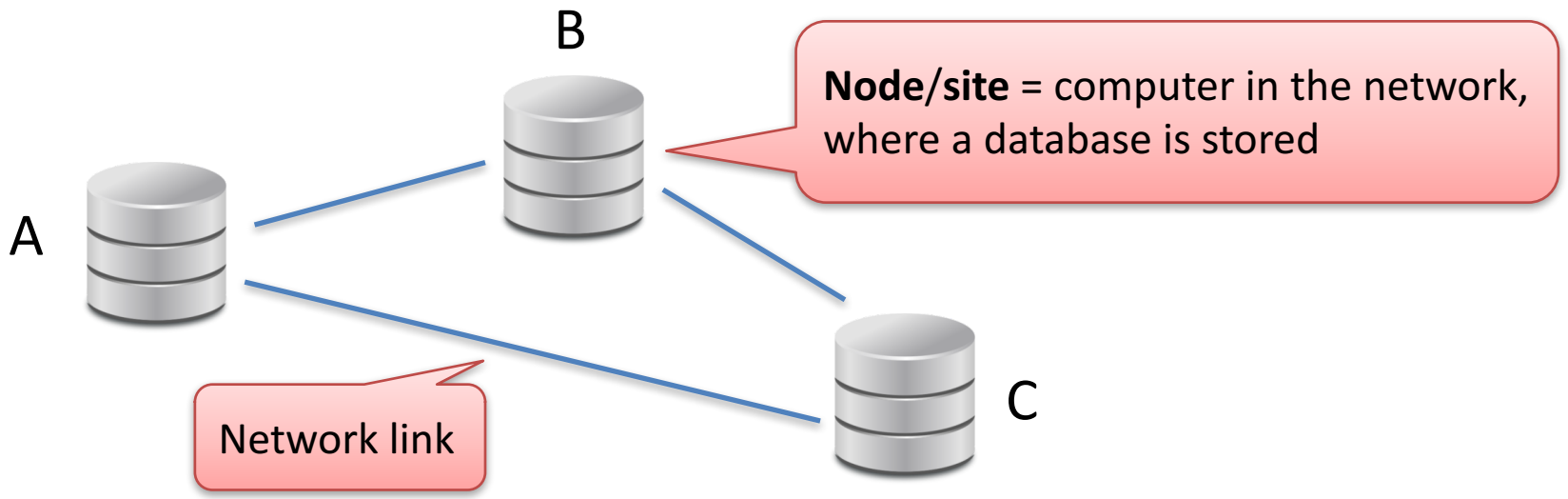
millions of users per day
many at the same time

> 42 TB of data
> 60 million active users
millions of products

100s of servers

Distributed Databases

- **Distributed Database**:
  - Collection of multiple **logically interrelated** databases
  - **Distributed** over a computer network



- **Distributed DBMS**: manages a distributed database

Advantages

- **Performance improvements**
  - Answer queries faster by distributing tasks over the nodes
  - Reduces CPU time, disk accesses, communication cost, … at individual nodes
- **Scalability**
  - Easier extension of the system: capacity, performance, …
  - Essentially just add a new node
- **Resilience**
  - Data can be replicated at geographically separate sites
  - Catastrophic failures don’t affect the entire system

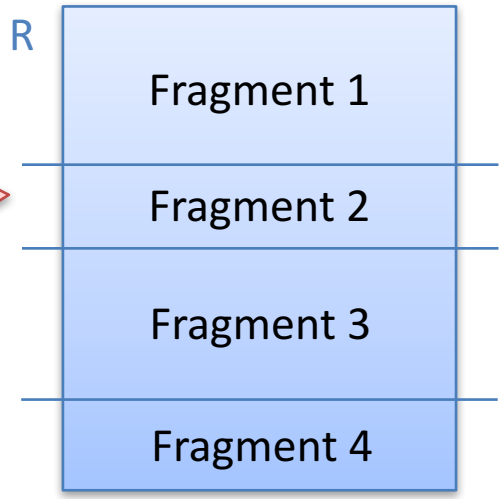
How to Distribute Data?

Fragmentation

- Split database into different parts that can then be stored at different nodes

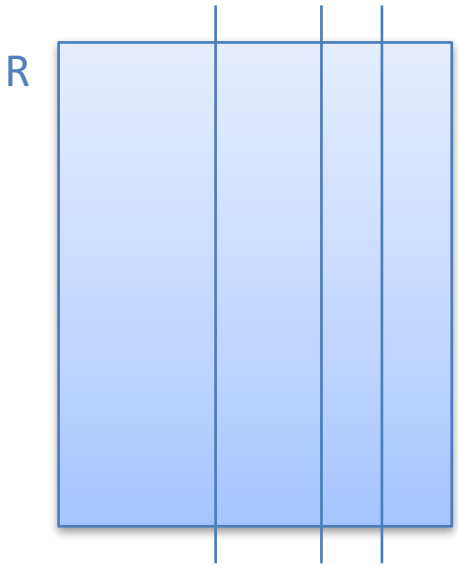
"Sharding"

Horizontal fragmentation



Fragments may overlap

Vertical fragmentation



- Users don't see fragments, just the full relations

"Fragmentation transparency"

Horizontal Fragmentation Example

- The chain of stores from our example might jointly store a relation

Does not exist physically!

item	date	price	purchaser	
Product A	17/10/2017	£29.99	Anna A.	← @Liverpool
Product B	19/10/2017	£199.99	Ben B.	← @Manchester
Product C	19/10/2017	£599.99	Chloe C.	← @Manchester
Product D	20/10/2017	£9.99	David D.	← @London
Product E	21/10/2017	£59.99	Emma E.	← @Liverpool
…	…	…	…	

- Each site stores a relation that contains a subset of the tuples
- Entire relation = union of relations at the different sites

Horizontal Fragmentation Example

- Typically, tuples stored at different sites can be distinguished
  - by the value of one or a few attributes; or
  - by other conditions that are easy to test

store	item	date	price	purchaser
Liverpool	Product A	17/10/2017	£29.99	Anna A.
Manchester	Product B	19/10/2017	£199.99	Ben B.
Manchester	Product C	19/10/2017	£599.99	Chloe C.
London	Product D	20/10/2017	£9.99	David D.
Liverpool	Product E	21/10/2017	£59.99	Emma E.
…	…	…	…	…

Vertical Fragmentation Example

- The stores might also jointly store a relation

Does not exist physically!

item	date	purchaser	last_payment
Product A	17/10/2017	Anna A.	30/09/2017
Product B	19/10/2017	Ben B.	30/09/2017
Product C	19/10/2017	Chloe C.	31/08/2017
…	…	…	…

@Central Office

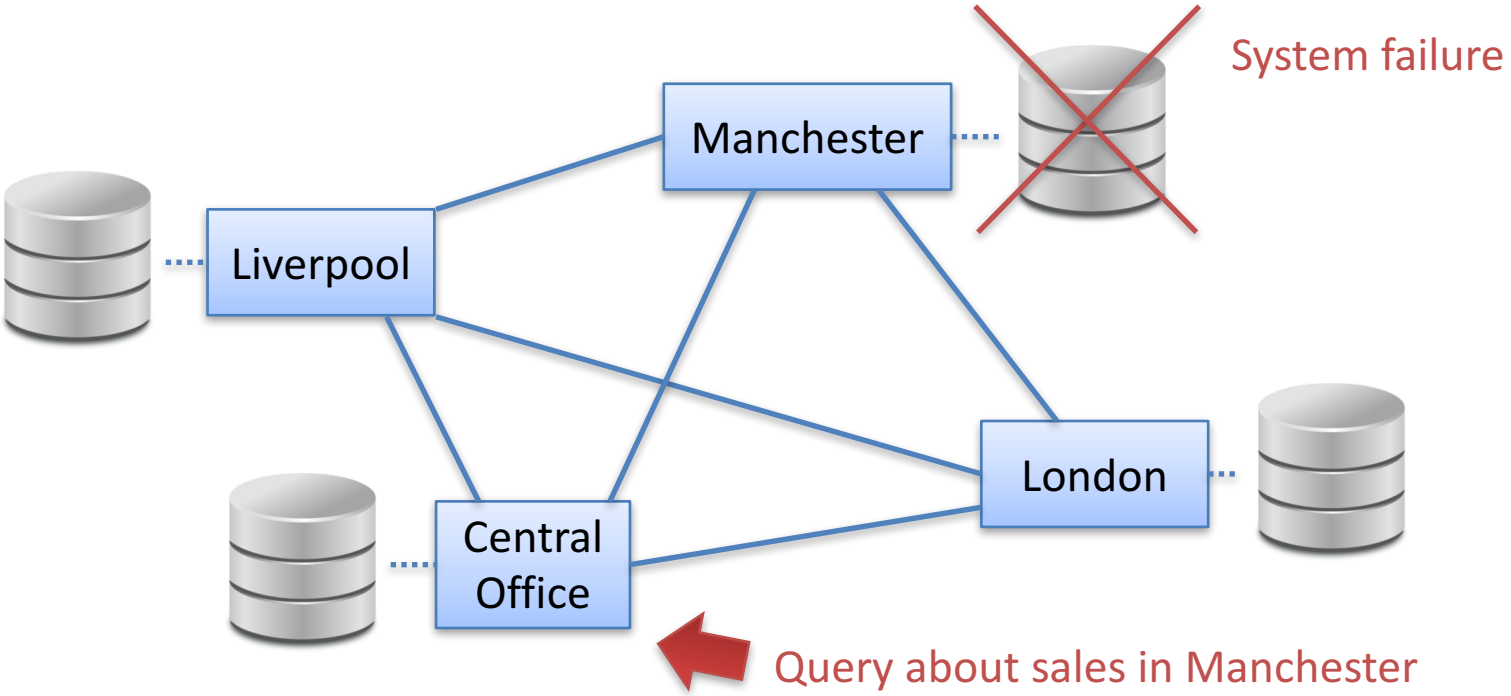
purchaser	last_payment
Anna A.	30/09/2017
Ben B.	30/09/2017
Chloe C.	31/08/2017
…	…

Stored at other sites

item	date	purchaser
Product A	17/10/2017	Anna A.
Product B	19/10/2017	Ben B.
Product C	19/10/2017	Chloe C.
…	…	…

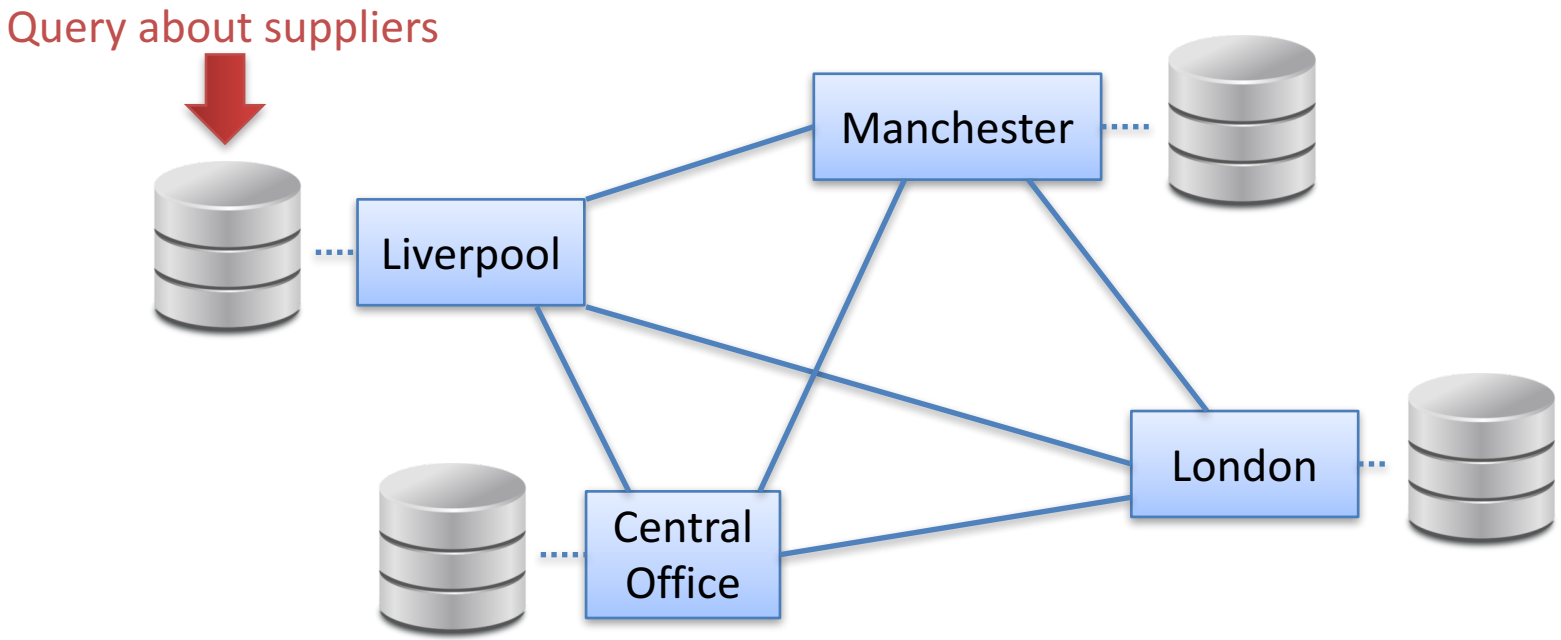
- Original relation ≈ join of the fragments

Redundancy Improves Resilience



- Other sites keep copies of fragment stored at Manchester
- Allows us to answer queries involving data from Manchester

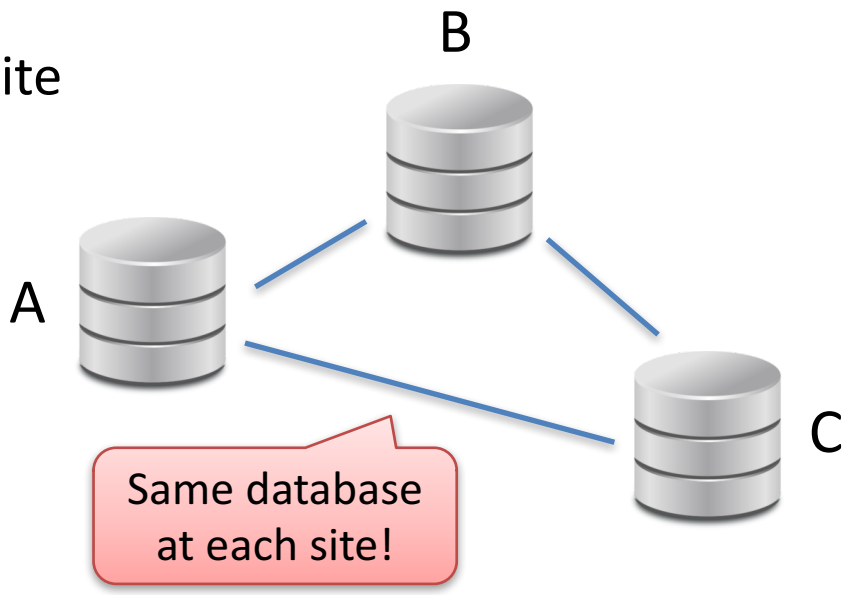
Redundancy Increases Efficiency



- Other sites keep copies of data about suppliers
- Allows stores to answer queries involving suppliers without establishing a connection to the central office

Replication

- Controls how many sites keep a copy of a fragment
- **Full replication**
  - Each fragment stored at every site (→ there are no fragments)
  - Faster query answering
  - Very slow updates: consider *every* copy
- **No replication**
  - Each fragment stored at a unique site



- Wide spectrum of **partial replication**
  - Limit number of copies of each fragment
  - Replicate only some fragments, …

Transparency

- DBMSs hide how data is distributed over sites
- Transparency at different levels
  - **Fragmentation transparency**
    - Fragmentation is invisible to users
    - Users pose queries against the entire database
    - The distributed DBMS translates this into a query plan that fetches the required information from appropriate nodes
  - **Replication transparency**
    - Ability to store copies of data items / fragments at different sites
    - Replication is invisible to users
  - **Distribution transparency**
  - **Naming transparency**

Summary

- **Distributed databases** are…
  - Collection of multiple logically interrelated databases
  - Distributed over a computer network
- Advantages: Improved performance, scalability, resilience against failures
- Query answering more challenging
  - Goal: minimise communication
  - With joins: a strategy using semijoins to prefilter tuples may yield better communication costs
- Next lecture: Transaction management is more challenging, too!