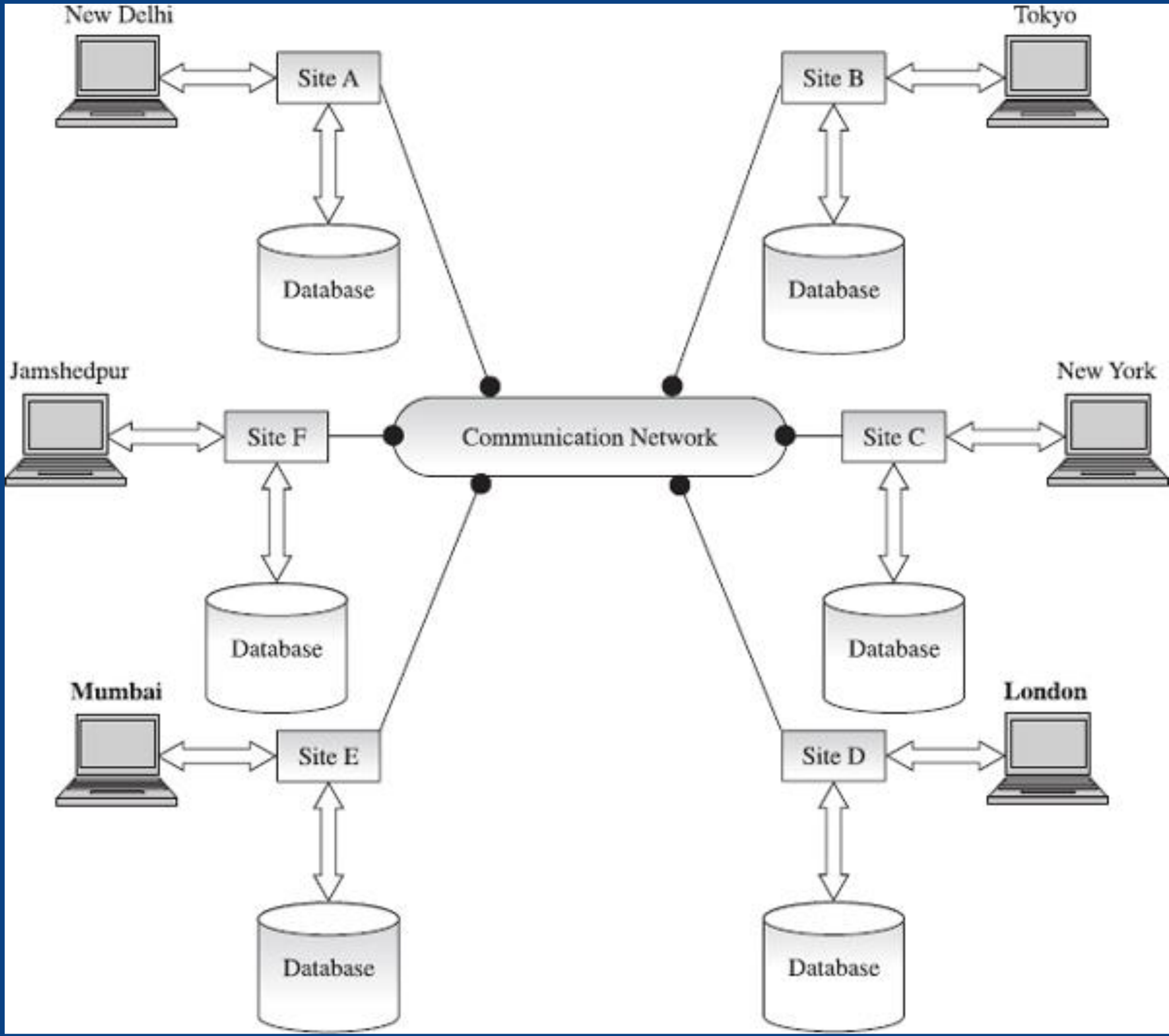
THE UNIVERSITY OF MELBOURNE

Distributed Databases

Database Systems & Information Modelling INFO90002

Week 10 – Distributed Databases
Dr Tanya Linden
David Eccles



This lecture discusses…

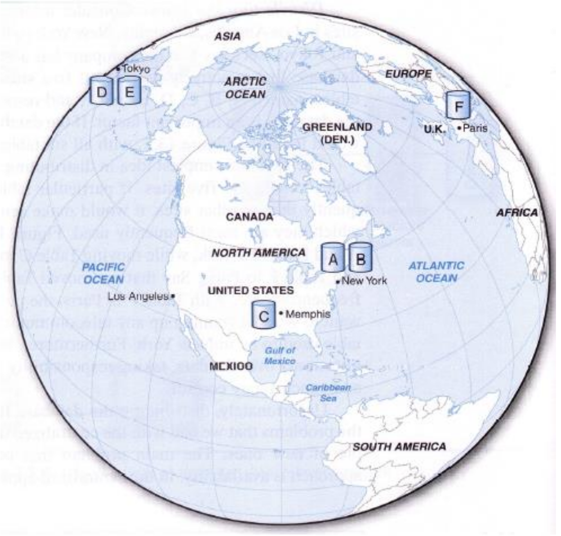
What is a distributed database?

Why are they used, and how they work

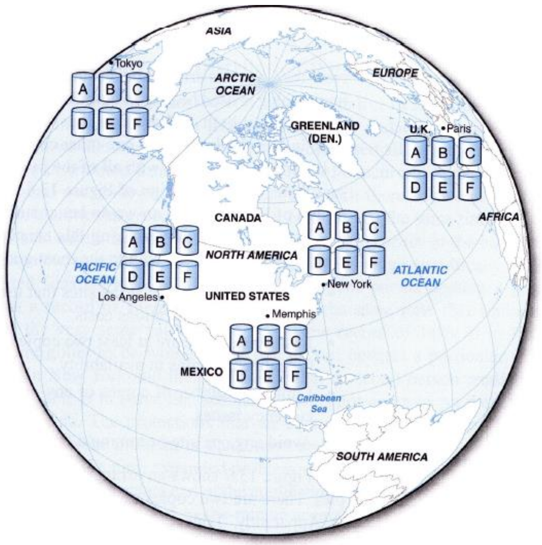
Pros and cons of different approaches

Material in this lecture is drawn from Hoffer et al. (2013) *Modern Database Management* 11th edition, chapter 12, available online at http://wps.prenhall.com/bp_hoffer_mdm_11/230/58943/15089539.cw/index.html

Images on this page are from Gillenson (2005) *Fundamentals of Database Management Systems*



distributed database



replicated database

Definitions

Distributed Database

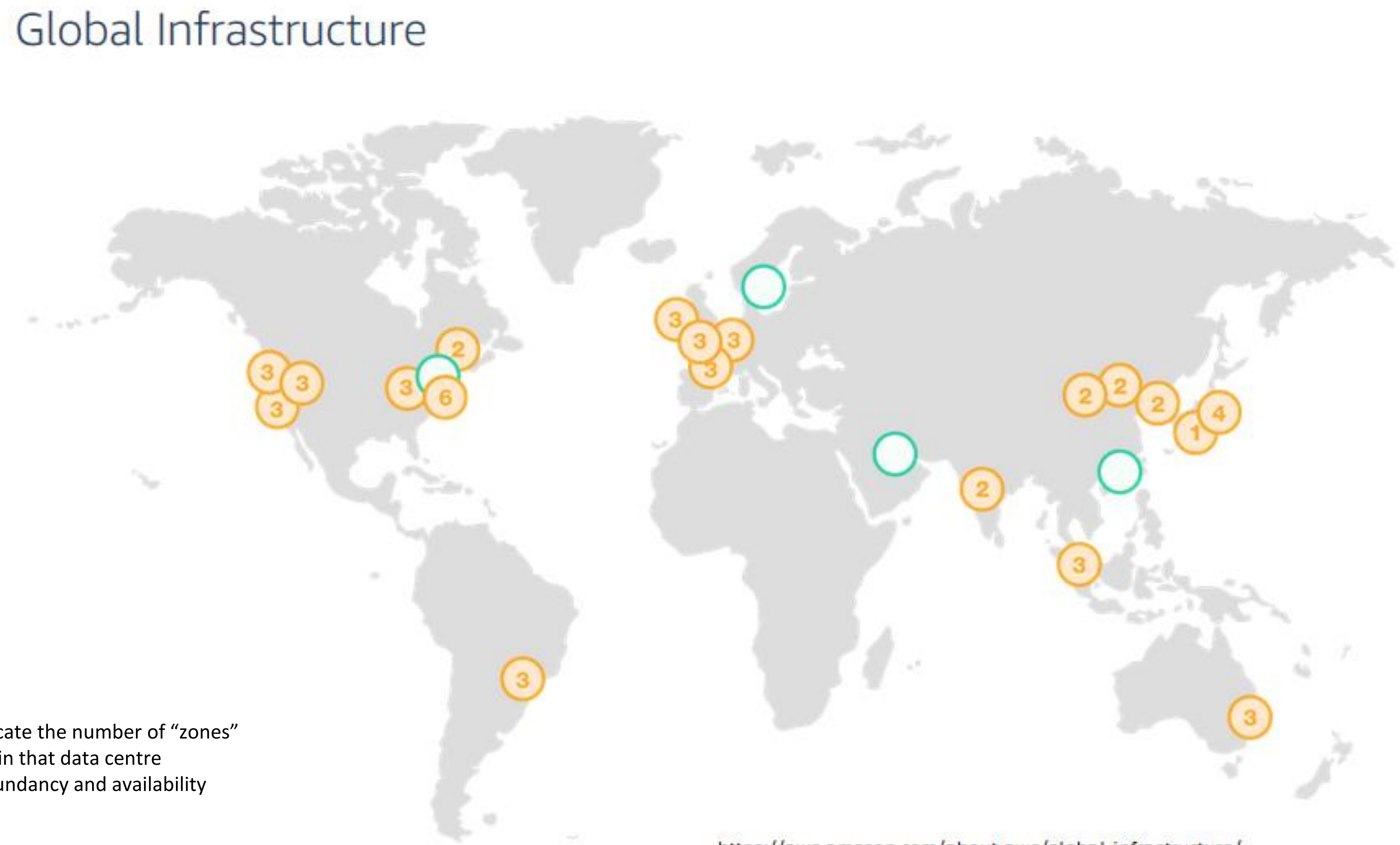
- a single logical database physically spread across multiple computers in multiple locations that are connected by a data communications link
- *appears to users as though it is one database*

Decentralized Database

- a collection of independent databases which are not networked together as one logical database
- appears to users as many databases

We are concerned with *distributed* databases

Example – Amazon AWS

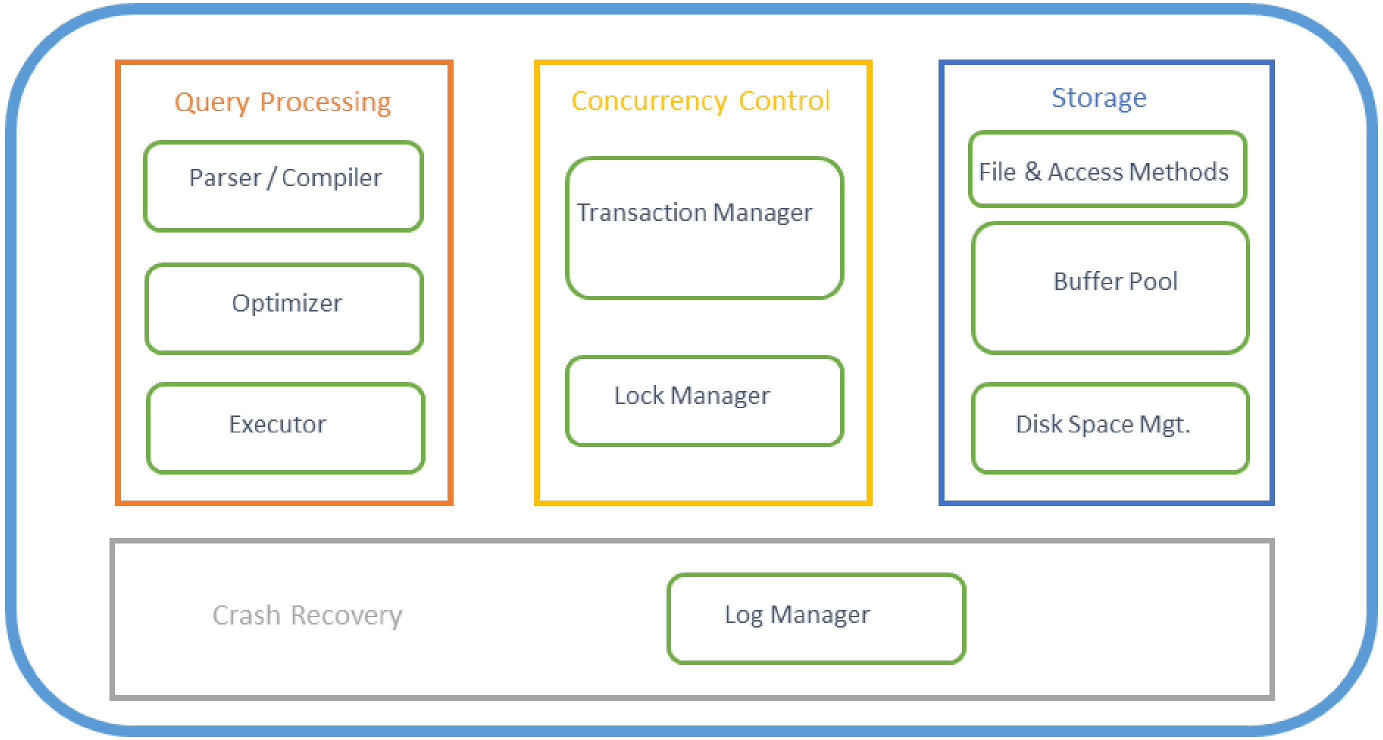


Numbers indicate the number of “zones” available within that data centre providing redundancy and availability

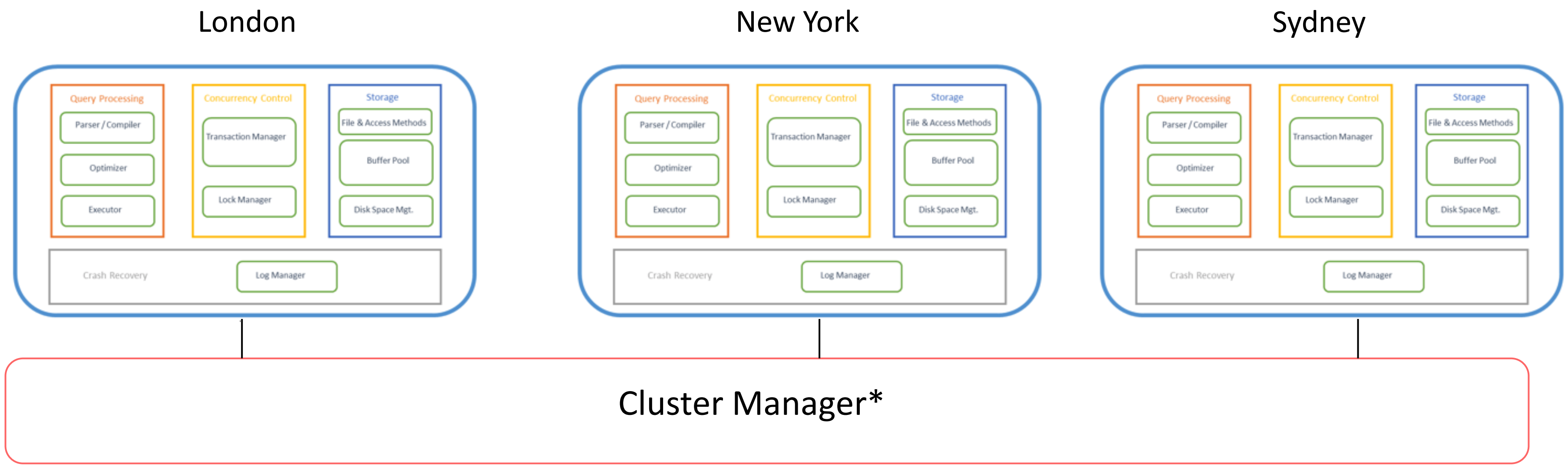
https://aws.amazon.com/about-aws/global-infrastructure/

Database Memory Structure (1 Server)

Remember this?



Distributed Memory Structures



Each Physical Server has one of these memory structures

Often accessing their own and shared physical storage between all physical servers

Cluster Manager coordinates communication between physical servers

* May be called something else in different vendor databases

Distributed DBMS Advantages

Good fit for geographically distributed organisations / users

- Utilize the internet

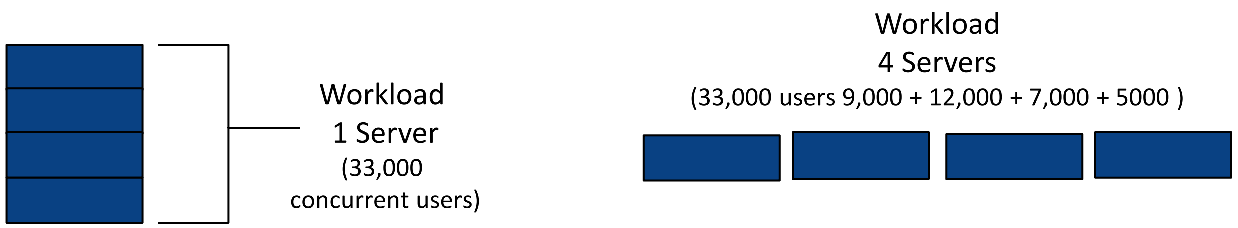
Data located near site with greatest demand

- ESPN Weekend Sports Scores

Faster data access (to local data)

Faster data processing

- workload is shared between each physical server



Distributed DBMS Advantages (cont.)

Allows *modular growth*

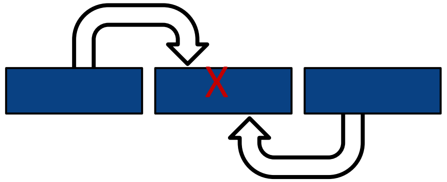
- Add new servers as load increases

Increased *reliability* and *availability*

- Lower danger of a single-point of failure (SPOF)

Supports *database recovery*

- Data is replicated across multiple sites
- Recovery Logs replicated



Disadvantages

Complexity of management and control

- database or/and application must stitch together data across sites
  - What is the current version of the record (row and column) and where is it?
  - Who is waiting to update that information and where are they?
  - How does the logic display this to the web and application server?

Data integrity

- additional exposure to improper updating
  - If two users in two locations update the record at the exact same time who decides which statement should “win”?
  - Solution: Transaction Manager or Master-slave design

Security

- many server sites -> higher chance of breach
  - Multiple access sites require protection from both cyber and physical attacks (including protection of network and storage infrastructure)

Disadvantages

Lack of standards

- different Relational DDBMS vendors use different protocols

Increased training and maintenance costs

- more complex IT infrastructure
  - Increased Disk storage ($)
  - Fast intra and inter network infrastructure ($$$)
  - Clustering software ($$$$)
  - Network Speed ($$$$$)

Increased storage requirements

- Replication model

Objectives and Trade-offs

Location transparency

- a user does not need to know where particular data are stored

Local autonomy

- a node can continue to function for local users if connectivity to the network is lost

Trade-offs

- Availability vs Consistency
- Synchronous vs Asynchronous updates

Location Transparency

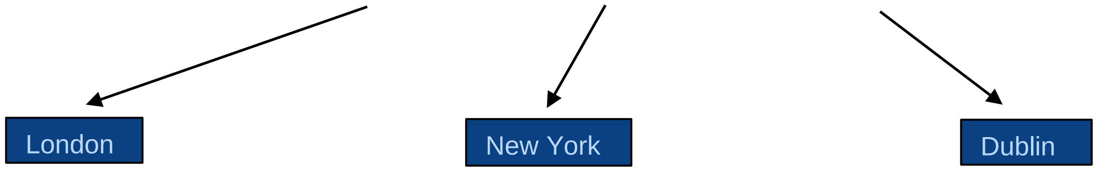
A user (or program) accessing data do not need to know the location of the data in the network of DBMS's

Requests to retrieve or update data from any site are automatically forwarded by the system to the site or sites related to the processing request

All data in the network appears to users as a single logical database stored at one site

A single query can join data from tables in multiple sites

```
SELECT hometeam, homescore, awayteam, awayscore
FROM results INNER JOIN codes
ON results.codeid = codes.codeid
WHERE sportscode in ('NFL', 'Hurling', 'EPL');
```



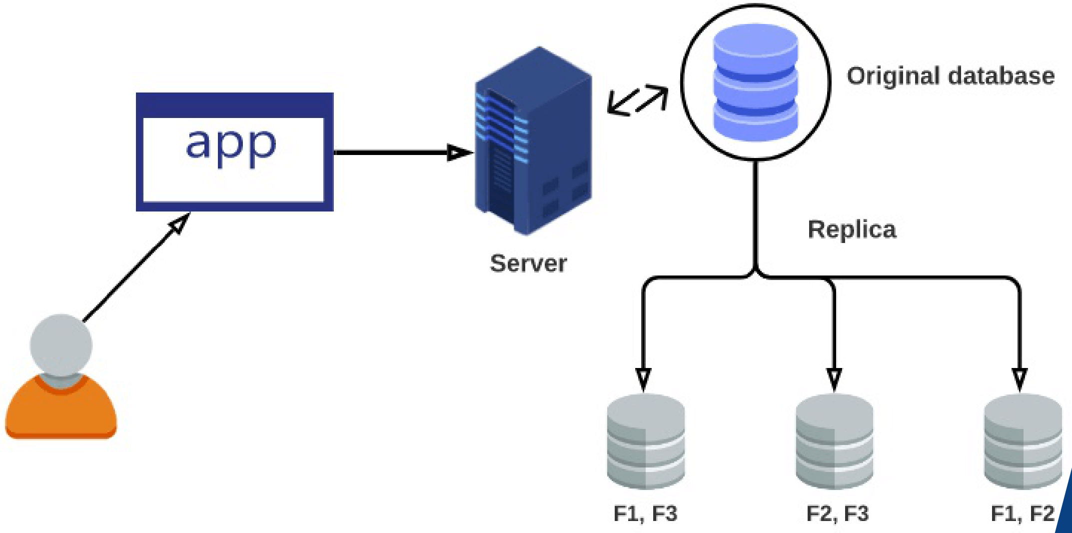
Local Autonomy

Users can administer their local database

- control local data
- administer security
- log transactions
- recover when local failures occur
- provide full access to local data

Being able to operate locally when connections to other databases fail

Partial replication in DBMS



[FI, F2, F3 are the different fragments of the main database]

Distributed Options

Data Replication
Horizontal Partitioning
Vertical Partitioning

Distribution options

Data replication

- Data copied across sites

Horizontal partitioning

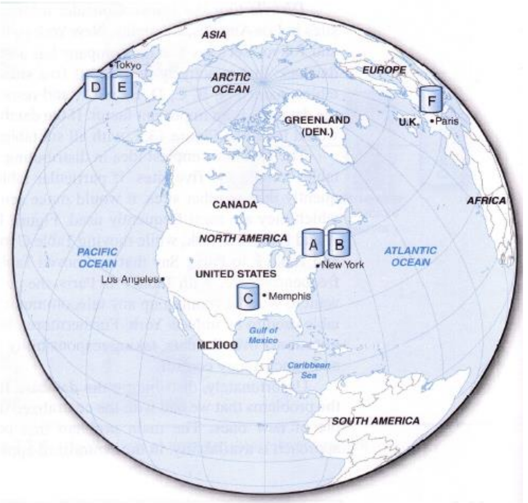
- Table rows distributed across sites

Vertical partitioning

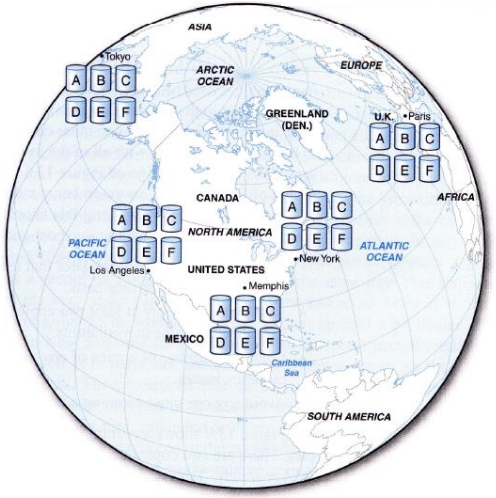
- Table columns distributed across sites

Combinations of the above

Partitioned



Replicated



Replication - advantages

High reliability due to redundant copies of data

Fast access to data at the location where it is most accessed

May avoid complicated distributed integrity routines

- replicated data is refreshed at scheduled intervals

Decoupled nodes don't affect data availability

- transactions proceed even if some nodes are down

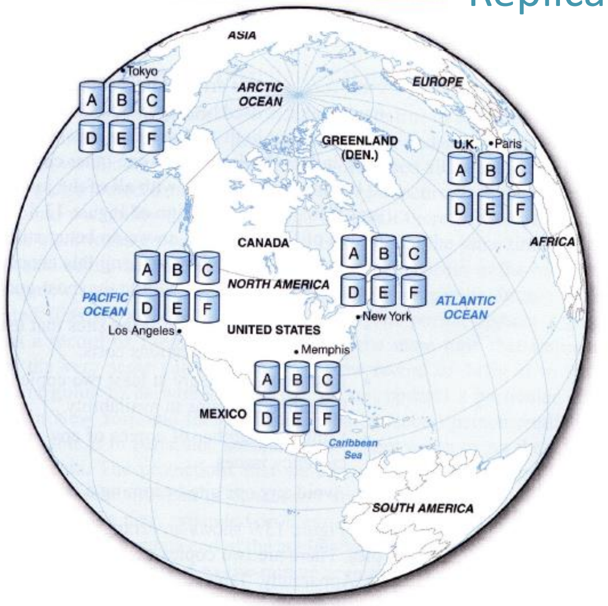
Reduced network traffic at prime time

- if updates can be delayed

This is currently popular as a way of achieving high availability for global systems.

- Most SQL and NoSQL databases offer replication

Replicated



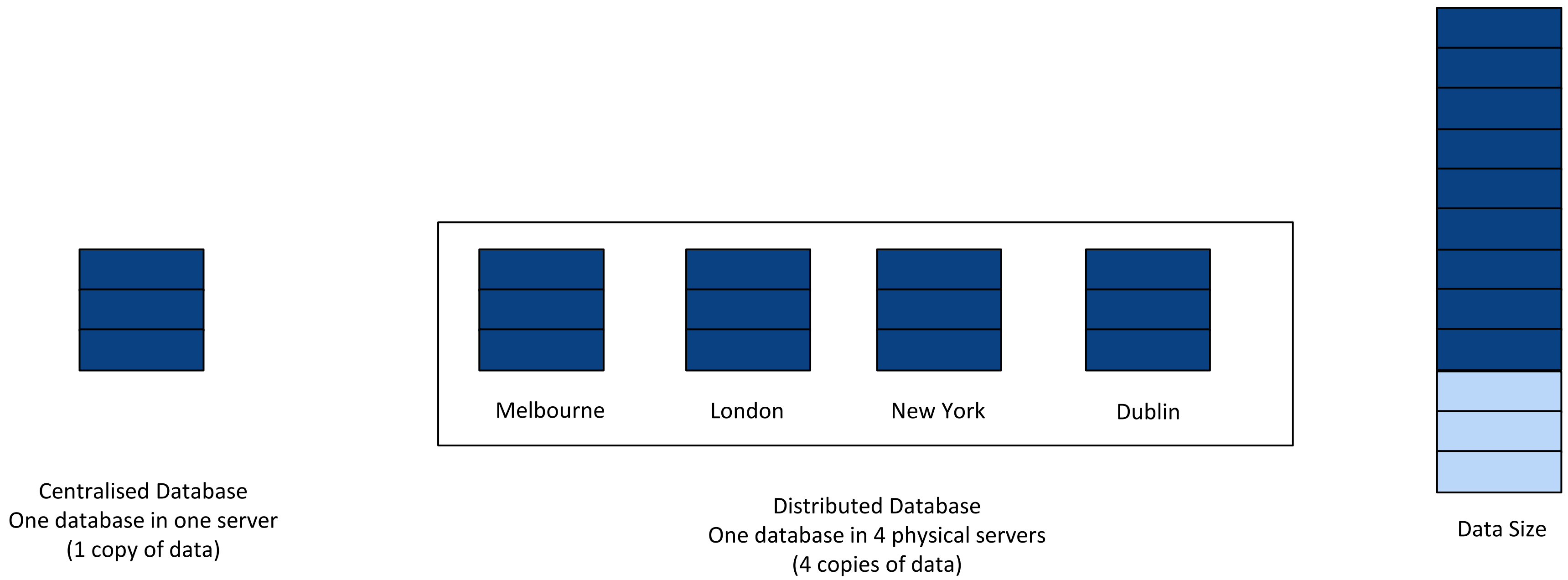
Replication - disadvantages

Need more storage space

- Each server stores a copy of the row

Data Integrity:

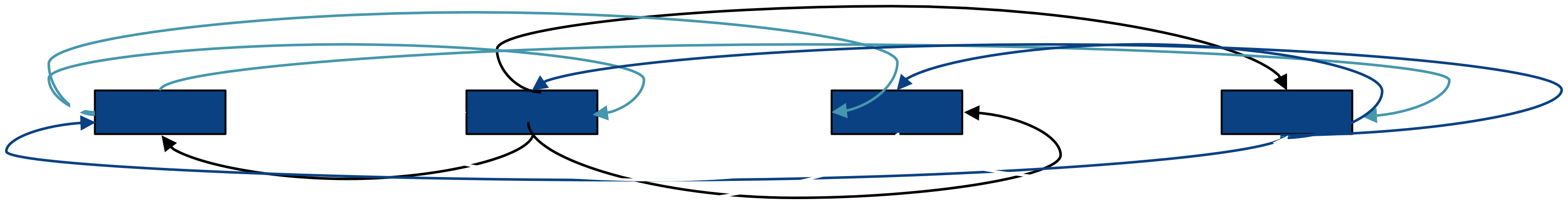
- retrieve incorrect data if updates have not arrived



Replication - disadvantages

Takes time for update operations

- High tolerance for out-of-date data may be required
- Updates may cause performance problems for busy nodes



Network communication capabilities

- Updates can place heavy demand on telecommunications/networks
- Cost high speed networks are expensive ($$$$$)

Synchronous updates

Data is continuously kept up to date

- Users anywhere can access data and get the same answer.

If any copy of a data item is updated anywhere on the network, the same update is immediately applied to all other copies or the update is aborted.

Ensures data integrity and minimises the complexity of knowing where the most recent copy of data is located.

Can result in slow response time and high network usage

- The DDBMS spends time checking that an update is accurately and completely propagated across the network.
- The committed updated record must be identical in all servers

Asynchronous updates

Some delay in propagating data updates to remote databases

- Some degree of at least temporary inconsistency is tolerated
- May be ok if it is temporary and well managed

Acceptable response time

- Updates happen locally and data replicas are synchronized in batches and predetermined intervals

May be more complex to plan and design

- Need to ensure the right level of data integrity and consistency

Suits some information systems more than others

- Compare commerce/finance systems with social media

Horizontal partitioning

Different rows of a table at different sites

Advantages

- data stored close to where it is used
  - efficiency
- local access optimization
  - better performance
- only relevant data is stored locally
  - security
- unions across partitions
  - ease of query

Disadvantages

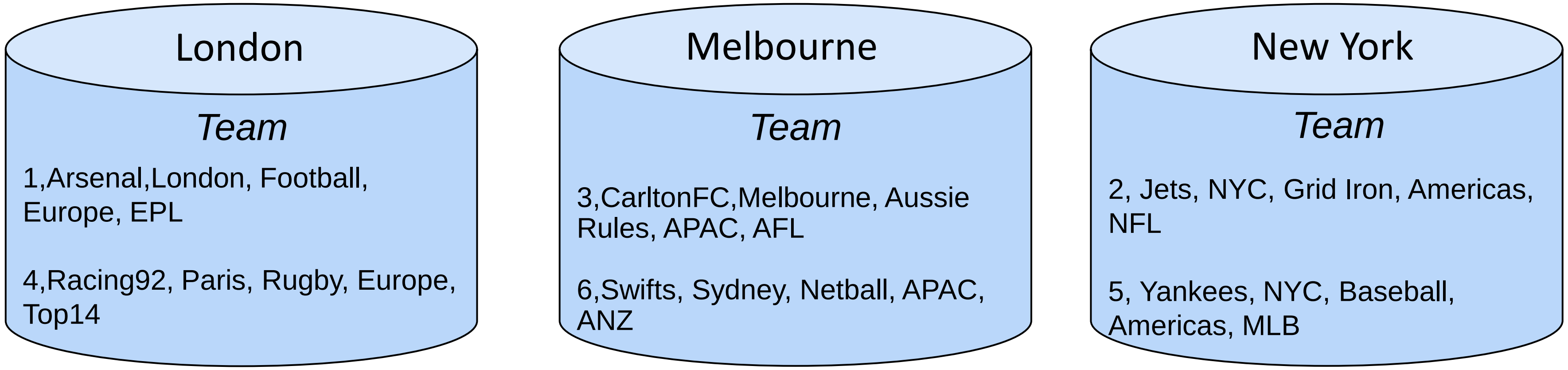
- accessing data across partitions
  - inconsistent access speed
- no data replication
  - backup vulnerability (SPOF)

ID	Team	City	Code	Region	League
1	Arsenal	London	Football	Europe	EPL
2	Jets	NYC	Grid Iron	Americas	NFL
3	Carlton FC	Melbourne	Aussie Rules	APAC	AFL
4	Racing92	Paris	Rugby	Europe	Top14
5	Yankees	NYC	Baseball	Americas	MLB
6	Swifts	Sydney	Netball	APAC	ANZ

Team table

Example horizontal partitioning

ID	Team	City	Code	Region	League
1	Arsenal	London	Football	Europe	EPL
2	Jets	NYC	Grid Iron	Americas	NFL
3	Carlton FC	Melbourne	Aussie Rules	APAC	AFL
4	Racing92	Paris	Rugby	Europe	Top14
5	Yankees	NYC	Baseball	Americas	MLB
6	Swifts	Sydney	Netball	APAC	ANZ



Vertical partitioning

Different columns of a table at different sites

Advantages and disadvantages are the same as for horizontal partitioning

- except
  - combining data across partitions is more difficult because it requires joins (instead of unions)

Player table

ID	Firstname	Lastname	Team	League	Photo	Biography
110	Luc	Ducalon	4	Top14		Ipso locum
120	Vasil	Kakokan	4	Top14		Ipso locum est
130	Donacca	Ryan	4	Top14	<null>	
210	Edwin	Maka	4	Top14		

Example vertical partitioning

ID	Firstname	Lastname	Team	League	Photo	Biography
110	Luc	Ducalon	4	Top14		Ipso locum
120	Vasil	Kakokan	4	Top14		Ipso locum est
130	Donacca	Ryan	4	Top14	<null>	
210	Edwin	Maka	4	Top14		

Vertical Partitioning based on column requirements

Dublin

Player

ID, First, Lastname, Team, League
110, Luc, Ducalon, 4, Top14
120, Vasil, Kakokan, 4, Top14
130, Donacca, Ryan, 4, Top14
210 Edwin Maka, 4, Top14

London

Player
Photo, Biography

Comparing 5 configurations

Centralised database, distributed access

- DB is at one location, and accessed from everywhere

Replication with periodic (asynchronous) snapshot update

- Many locations, each data copy updated periodically

Replication with near real-time synchronization of updates

- Many locations, each data copy updated in near real time

Partitioned, integrated, one logical database

- Data partitioned across many sites, within a logical database, and a single DBMS

Partitioned, independent, non-integrated segments

- Data partitioned across many sites.
- Independent, non-integrated segments
- Multiple DBMS, multiple computers

Comparing Configurations

	Reliability	Expandability	Communication Overhead	Management	Data Consistency
Centralised	POOR Depends on central server.	POOR Single Server is limited by memory & storage maximums.	VERY HIGH Traffic heads to one centralised location.	EXCELLENT One very large site is easier to manage.	EXCELLENT All users always see the same data.
Replicated with Snapshots	GOOD Redundancy and tolerated delays in data synch.	VERY GOOD Cheap to scale up with new servers.	LOW to MEDIUM Intermittent bursts of network traffic (but not constant flooding of network).	VERY GOOD Each copy is alike.	MEDIUM Update delays are tolerable with snapshot catch ups for data consistency.
Synchronised Replication	EXCELLENT Redundancy and minimal delays.	VERY GOOD Low cost and only linear growth in synchronisation.	MEDIUM Constant messages to maintain synchronisation.	MEDIUM Data collusions need to be resolved and need good design and management.	VERY GOOD Close to precise consistency.
Integrated Partitions	GOOD Effective use of partitioning and redundancy.	VERY GOOD New nodes only get the data they need and no need to change DB design.	LOW to MEDIUM Most queries are local, but global queries to create temporary comms load.	DIFFICULT Distributed table updates require tight precise coordination.	VERY POOR Requires considerable effort and inconsistencies are not tolerated.
Decentralised Independent Partitions	GOOD Depends on local DB availability.	GOOD New sites are independent of all other sites.	LOW Little or no traffic needs to be communicated across the network.	VERY GOOD Easy – as each site is independent of the other sites and minimal need to share data.	LOW No guarantee of consistency – therefore high chance of consistency.

Legend:
the darker the colour
the better the feature

Functions of a distributed DBMS

Locate data with a distributed catalog (meta data)

Determine location from which to retrieve data and process query components

DBMS translation between nodes with different local DBMSs (using middleware)

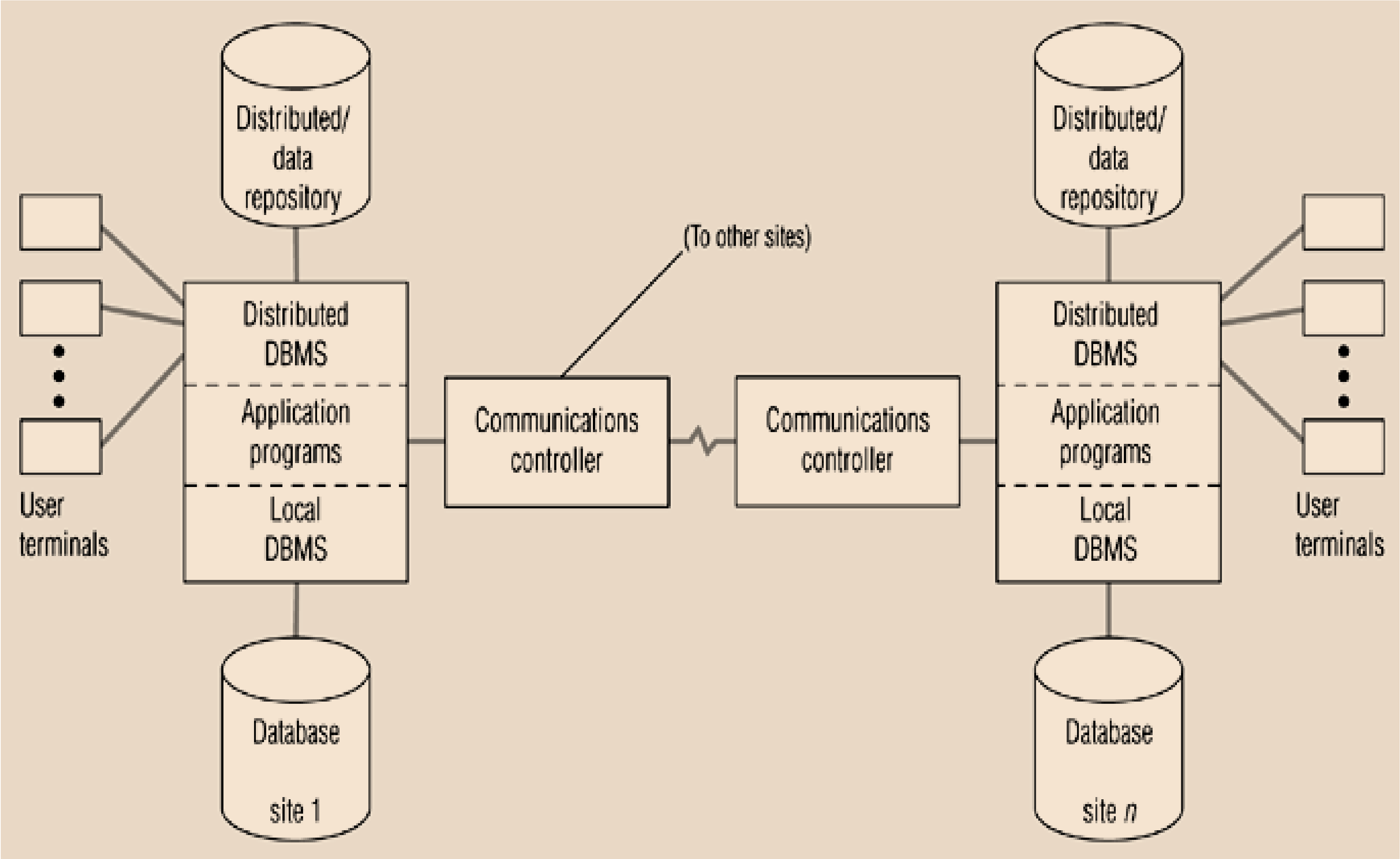
Data consistency (via multiphase commit protocols)

Global primary key control

Scalability

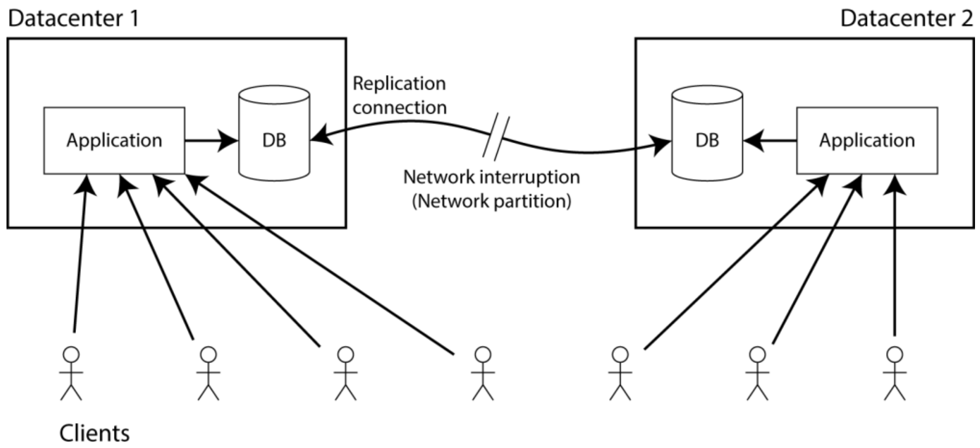
Security, concurrency, query optimisation, failure recovery

Distributed DBMS architecture



Network partitions

- Imagine you have a synchronously-updating, replicated database
- Now imagine that the link between 2 nodes is interrupted



- What are your choices?
  - shut down the system (to avoid inconsistency)
  - keep it available to users (and accept inconsistency)

(Diagram from https://martin.kleppmann.com/2015/05/11/please-stop-calling-databases-cp-or-ap.html)

What's examinable

Distributed Database

Advantages and Disadvantages

Replicated Databases

Advantages and Disadvantages

Synchronous vs Asynchronous

Difference between

Advantages and Disadvantages

Partitioning Options

Vertical, Physical, Vertical and Physical

The five configurations

Advantages and Disadvantages

*** All material is examinable – these are the suggested key skills you would need to demonstrate in an exam scenario**

THE UNIVERSITY OF
MELBOURNE

Thank you