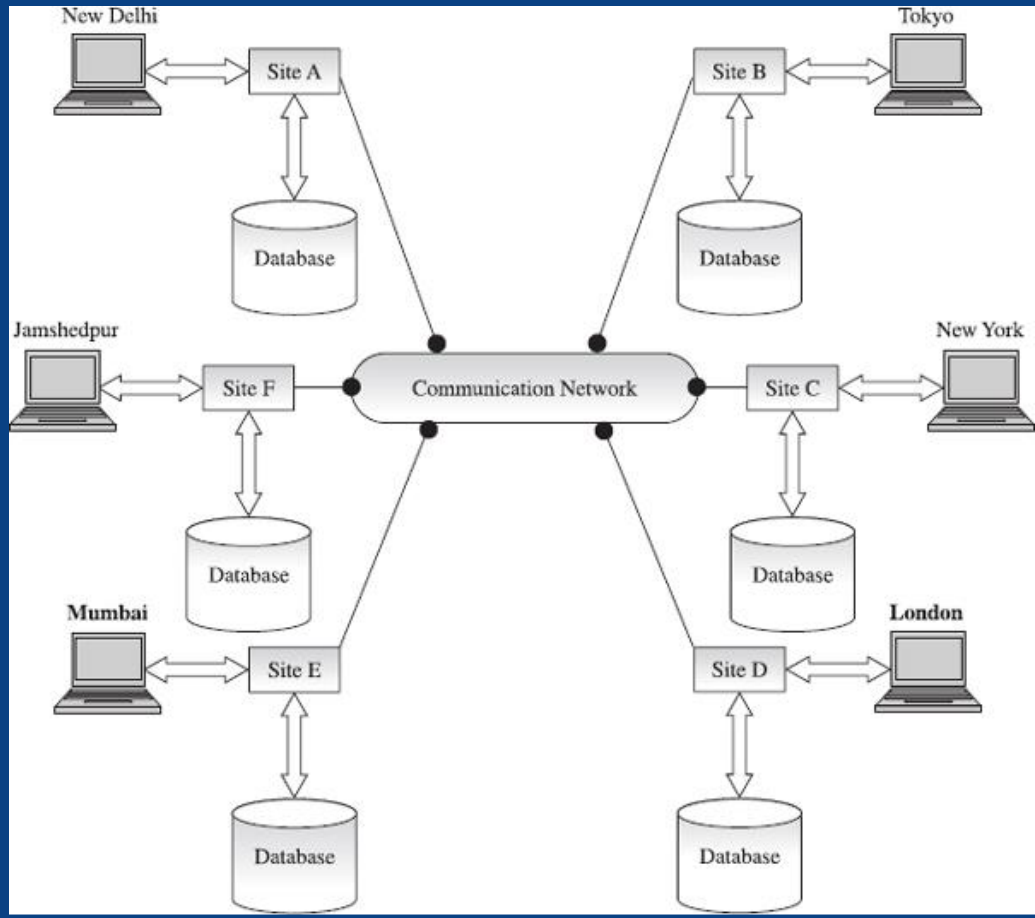
THE UNIVERSITY OF MELBOURNE

Distributed Databases

Database Systems & Information Modelling
INFO90002

Week 8-9 – Distributed Databases
Dr Tanya Linden
David Eccles



This lecture discusses...

so need distribution database

centralize server
↓ good for local company
but not good for global company

What is a distributed database?

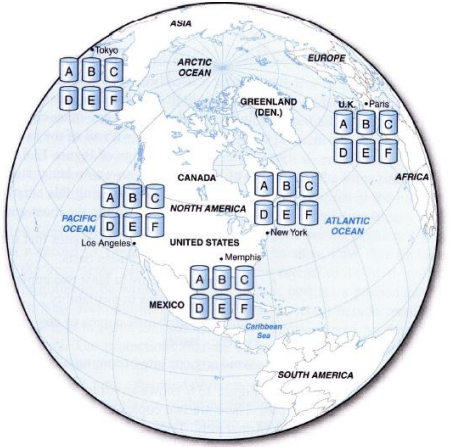
Why are they used, and how they work

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

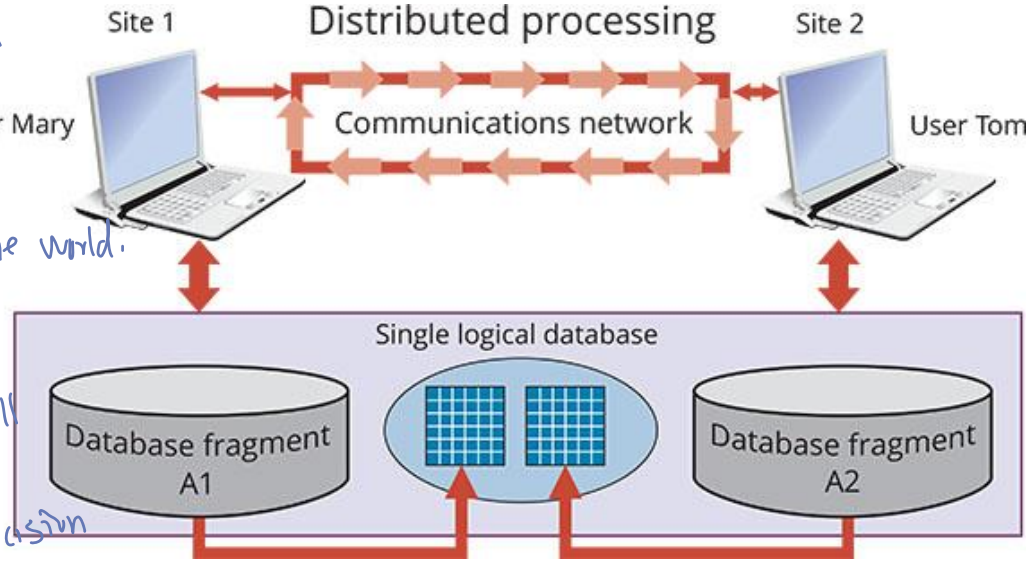


replicated database

Definitions

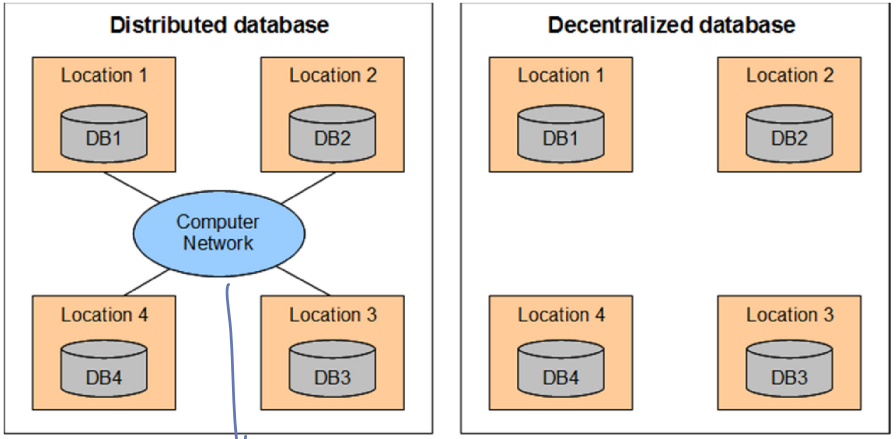
Distributed Database

- 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

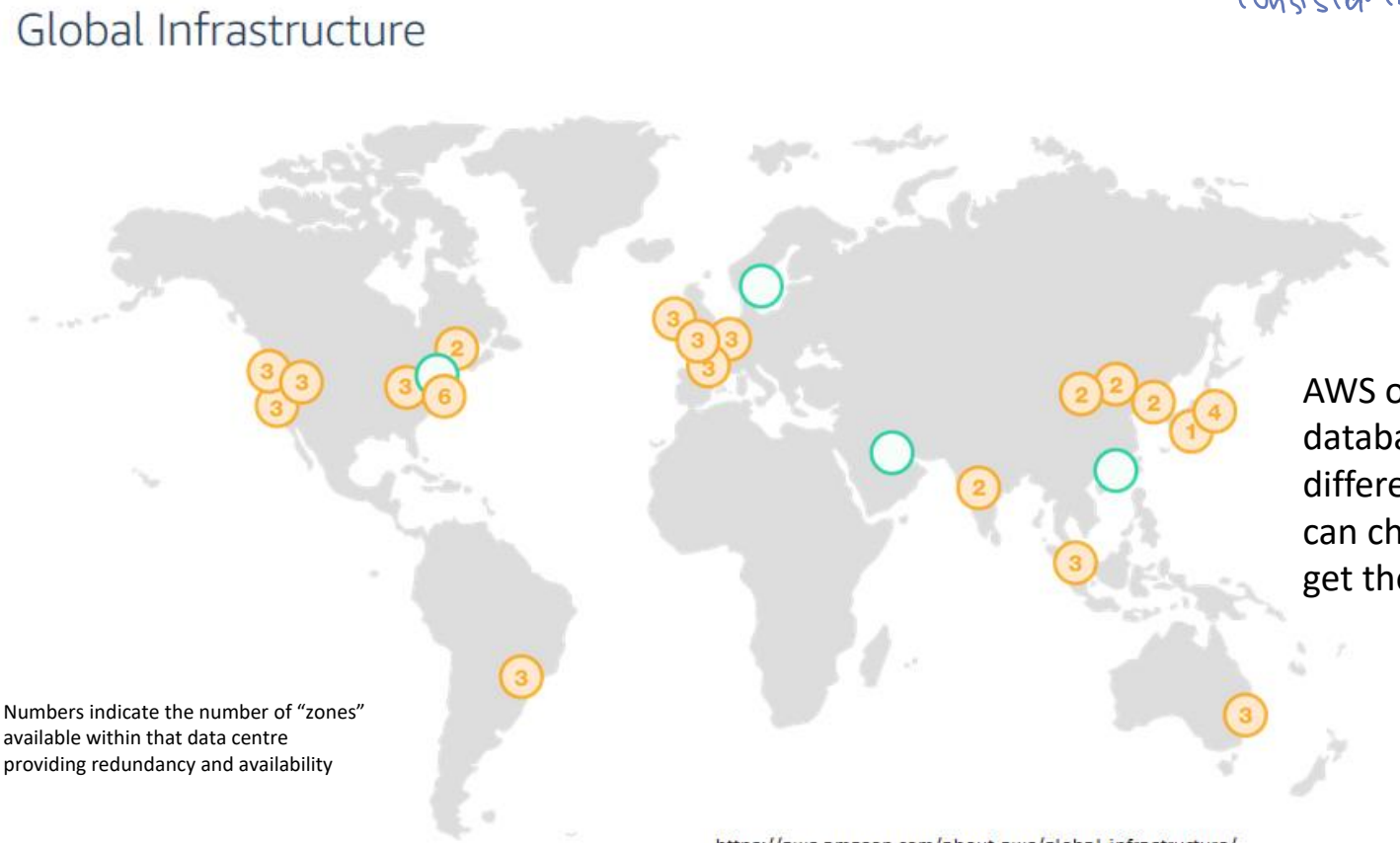
- a collection of independent databases which are not networked together as one logical database
- appears to users as many databases



We are concerned with *distributed* databases

for distribution database, the replication date has update. other replicate data will also be update. So the database is consistent.

Example – Amazon AWS



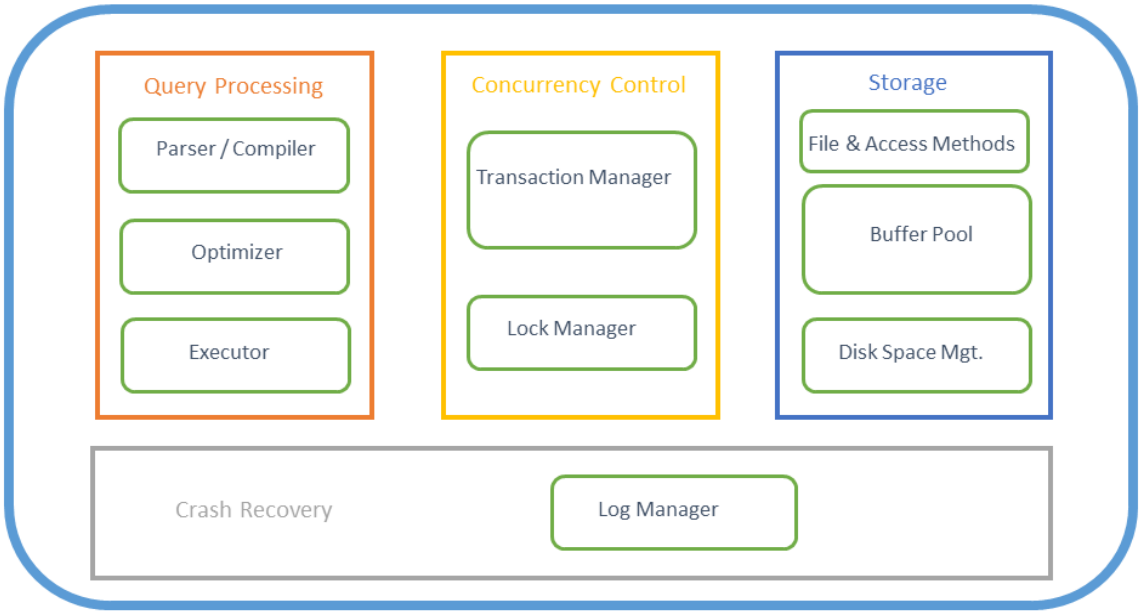
Numbers indicate the number of "zones" available within that data centre providing redundancy and availability

AWS offers the widest variety of databases that are purpose-built for different types of applications so you can choose the right tool for the job to get the best cost and performance.

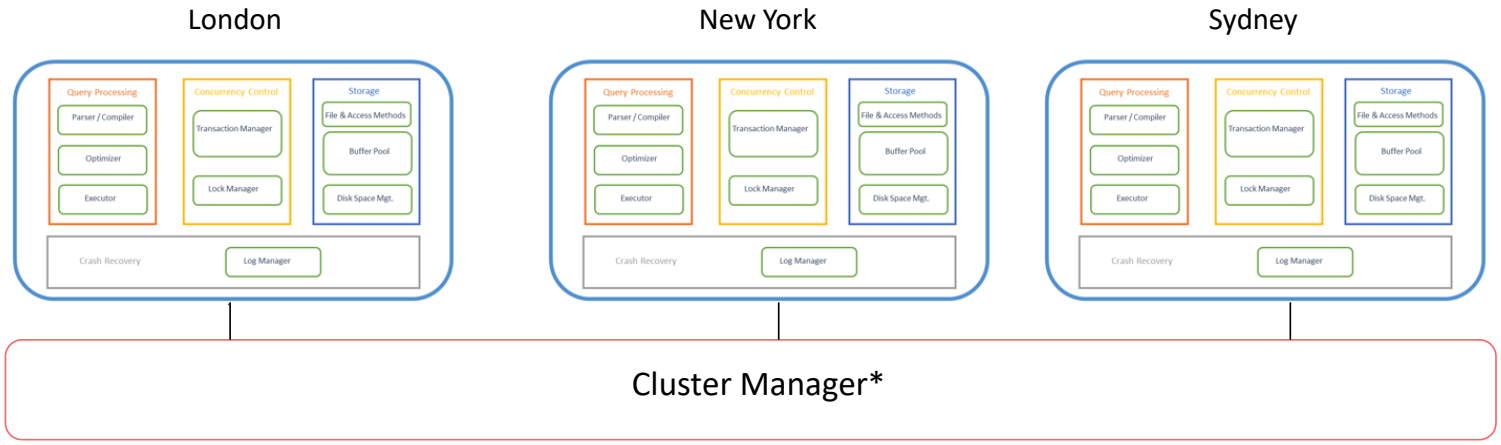
https://aws.amazon.com/about-aws/global-infrastructure/

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 including Query Processing areas, Multiple Concurrency Control, Storage, Crash Recovery

* May be called something else in different vendor databases

10:16

Distributed DBMS Advantages

Good fit for geographically distributed organisations / users

- Utilize the internet

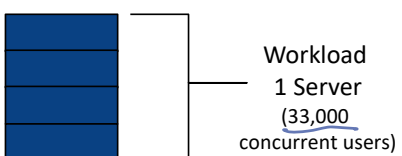
Data located near site with greatest demand

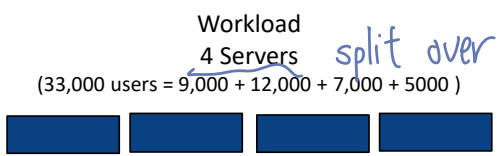
- ESPN Weekend Sports Scores

global → distributed

then we decide which data is important for which geographical area and put that data into server.

- Faster data access (to local data)
- Faster data processing — local server will process local data.
  - workload is shared between each physical server





split over four country or city

because it's distributed database, it's easy to add more servers.

Distributed DBMS Advantages (cont.)

Allows *modular growth*

- Add new servers as load increases

Increased *reliability* and *availability* of data → because it's a distributed database

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

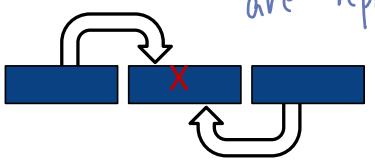
Even if you don't replicate data across all servers, you keep only relevant data on each server.

→ The meta data. the logs are everywhere. they are replicated, these files are small

Supports *database recovery*

- Data may be replicated across multiple sites
- Recovery Logs replicated

recovery logs are always replicated.



So. if London server crushed. you can get the data from (log file) Paris server. and restore London server.

Disadvantages

This means that your data, like response to every query, needs to be collected together from different servers from different sites.
↳ This might cause the problem: ① which one is the latest version.
↳ what if one server is update and another one hasn't yet.
② who is waiting, and where are they?
③ how do we tell the application that is sitting on top of database
→ data integrity.

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 (especially data replicated across servers)
  - 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

→ it decide who, when to get the access first.

↘ master is the server which hold the latest update
→ when the writting happen, it happens at the master server only
→ when someone need to read data (like create some report)
it can happen at any slave

is for update

Security

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

- Huwever. it could also mean that reading data from slaves are still dealing with the old version of record

→ So. we will need synchronous update. meaning that as soon as master is updated, it sends updates to all slaves.
But this means it has to freeze all slaves for responding to query while they're updating data.

- A synchronuous update means that master is updated. but it wait for slow down in traffic.
(less user)

disadvantage of distributed DBMS:

1. complexity of management and control
2. data integrity
3. security
4. lack of standard (chanllenge of interoperability)
5. increase frainning and mainfenance costs.

disavantage of replicated DBMS:

1. need more storage spare
2. data integrity
3. Takes time to update
4. network communication capability

advantage:

1. allow modular growth
2. increase reliability and availability
   - lower danger of single point of failure (SPOF)
3. supports database recovery
4. good fit for geographically distributed organisation/user
5. fast data access to local data
6. fast data process to local data

advantage:

1. high reliability
2. fast access
3. decoupled nodes don't affect data availability
4. reduced network traffic at prime time

Disadvantages

these day we often go into cloud for distributed database system.
↳ like outsource (cloud: Microsoft, Google)
example: when we want to change the all students Gmail in Google to Microsoft oracle, there might be a problem with compatibility

Lack of standards

- different Relational DDBMS vendors use different protocols
  - Challenge with interoperability 互操作性
  - Challenge to migration

Increased training and maintenance costs (money)

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

Objectives and Trade-offs

summary

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

(2) basically this cluster manager decides what is the nearest site where the data is available and looks after that request being proccessed.

(1) A user (or program) accessing data do not need to know the location of the data in the network of DBMSs

(2) 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

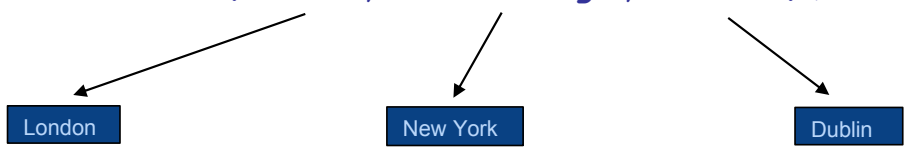
(3) All data in the network appears to users as a single logical database stored at one site

A single query can join data from tables in multiple sites

→ But in reality, a single query might from multiple sites. so it could be union or it could be join, depend how data distributed

(union is faster than join table)

```
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

It means that the database can be sitting on the servers and can operate without other servers.

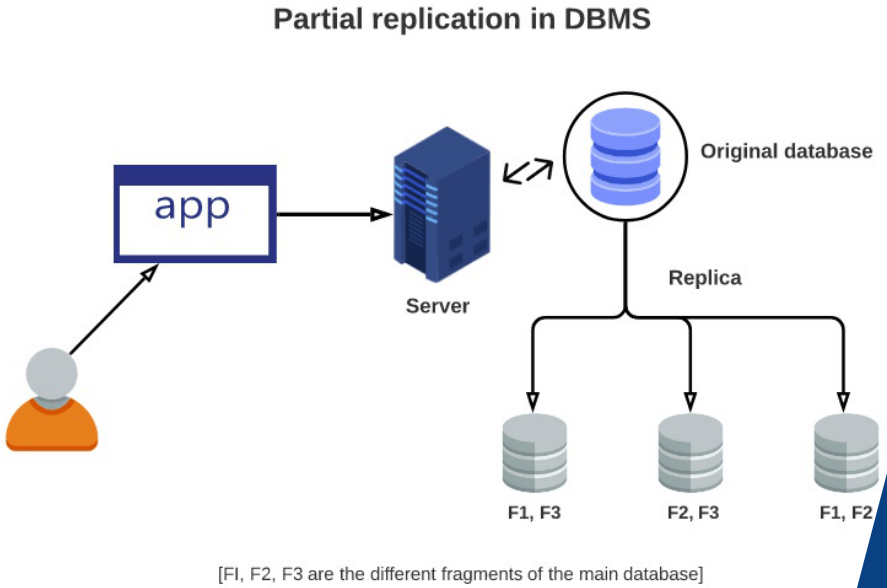
So if there is breakdown communications, like London lost connect with Paris server, London still operate using local data. And also be able to look after security, apply updates, use logs to restore whatever was lost during crashes and so on.

Trade-offs

consistency is about the same record on any server would be exactly the same.
But you have fully or partially replicated database, it's impossible to have every simultaneously.
For consistency. synchronous is the best one because as soon as I updated sth. the update will send all over the world. → But this mean, those server need to forget everything right now to do the update to ensure the consistency.

- Availability vs Consistency
  - The more data consistency enforced, the lower availability is
- Synchronous vs Asynchronous updates → To ensure everyone get the latest version of data, it slows down the servers.
  - Synchronous waits for all involved database nodes to update, before confirming with query requestor
  - Asynchronous writes to the master database node and then confirms with the query requestor, without the other nodes necessarily having been updated (they may be updated a while later)

Stocks exchange
social media definitely use asynchronuous approach, it's not in a hurry.



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

Distribution Options

Data Replication
Horizontal Partitioning
Vertical Partitioning

Distribution options

In partitioned, every server will have different data. but because it's a distributed database, the managing software know how to collect it. from different servers.

Replicated: everything is replicated

Most often: there is a combination.
important things → replication
others → partitioning

Data replication

- Data copied across sites

Horizontal partitioning by rows

- Table rows distributed across sites

Vertical partitioning by columns.

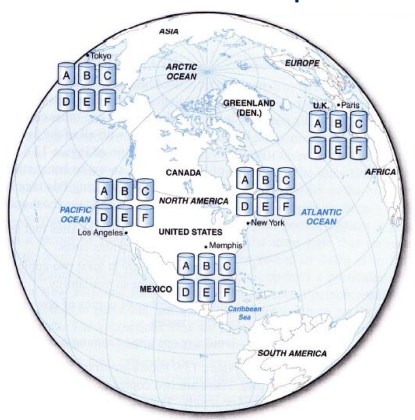
- Table columns distributed across sites

Combinations of the above

- E.g. partition a user table horizontally, and replicate a lookup table

Partitioned

Replicated



Horizontal partitioning

row to row to where it's needed most (ex: popular in London. store in London server)

Different rows of a table at different sites

Advantages

1. data stored close to where it is used
   - efficiency
2. local access optimization
   - better performance
3. only relevant data is stored locally
   - security

If the location is compromised, only part of things might be stolen. (we might lose some data, but not everything)

4. unions across partitions
   - ease of query

collect data across partitions → use union rows.

Disadvantages

1. accessing data across partitions
   - inconsistent access speed
2. no data replication (difficult to restore it)
   - backup vulnerability (SPOF)

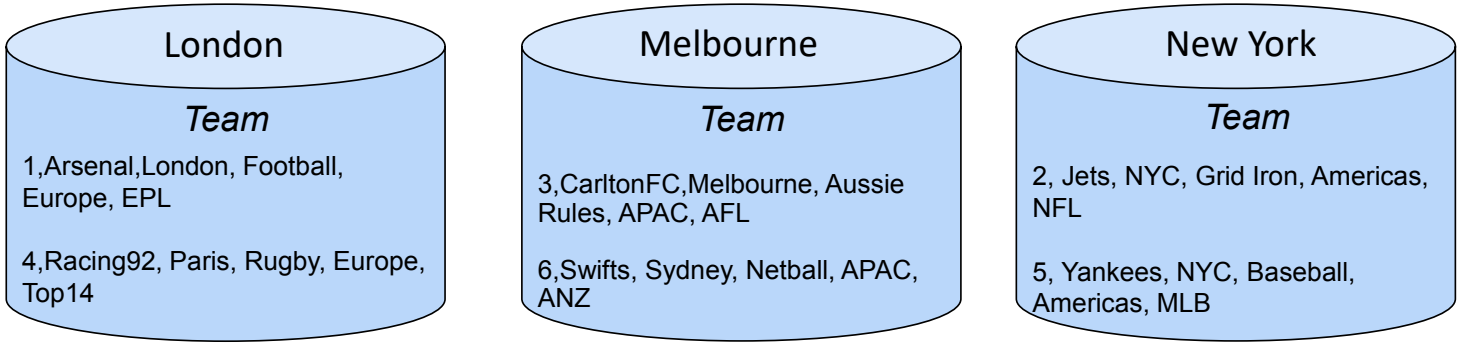
脆弱性

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



the partitioning is created base on the data needs, it can overlap.
like the same group of column can be stored on more than one servers.

Vertical partitioning

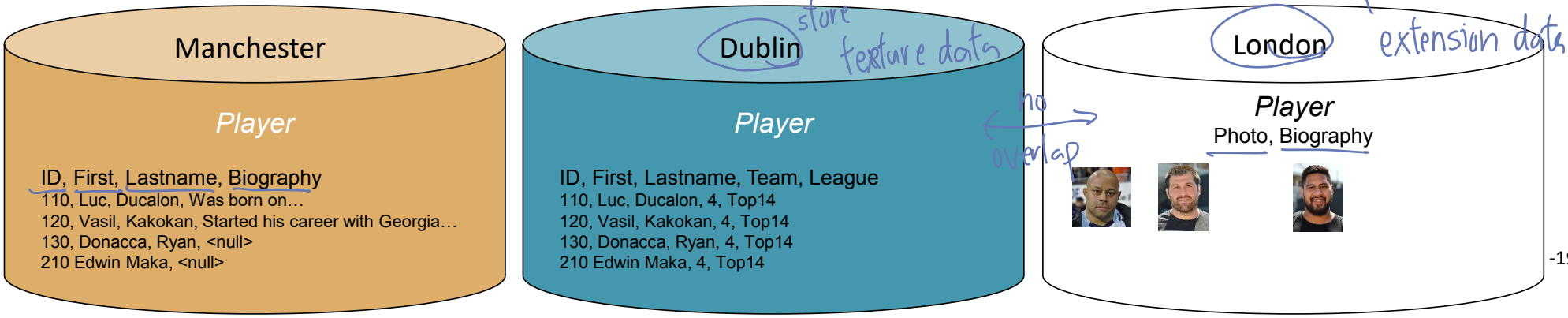
for aggregating function, vertical partitioning is more useful

Different columns of a table at different sites

Also possible to have all columns stored at some location

ID	Firstname	Lastname	Team	League	Photo	Biography
110	Luc	Ducalon	4	Top14		Was born on…
120	Vasil	Kakokan	4	Top14		Started his career with Georgia as back rower, as a 17-year-old
130	Donacca	Ryan	4	Top14	<null>	<null>
210	Edwin	Maka	4	Top14		<null>

Vertical Partitioning based on column requirements



Vertical partitioning

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)
  - advantage for aggregate column queries (vertical) vs queries that select entire row(s) (horizontal)

use in horizontal partitioning

faster

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		

can access is needed from more than one location

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 because we have local autonomy

- transactions proceed even if some nodes are down

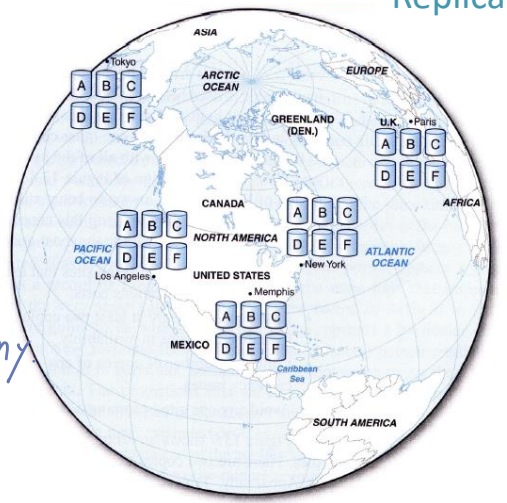
Reduced network traffic at prime time

- if updates can be delayed (only asynchronous)

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: the consistency that integrity can be violated.

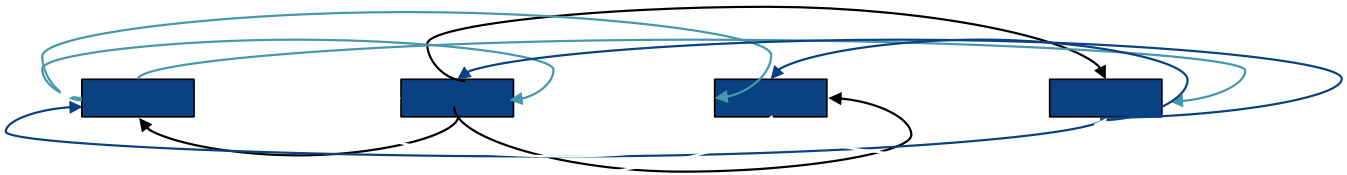
- retrieve incorrect data if updates have not arrived

Replication - disadvantages

If synchronous → everything slow down
If asynchronous → have to tolerate for out of date data.

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

acceptable delay in propagating data update

Asynchronous updates

Some delay in propagating data updates to remote databases

- Some degree of at least temporary inconsistency is tolerated
- May be acceptable 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

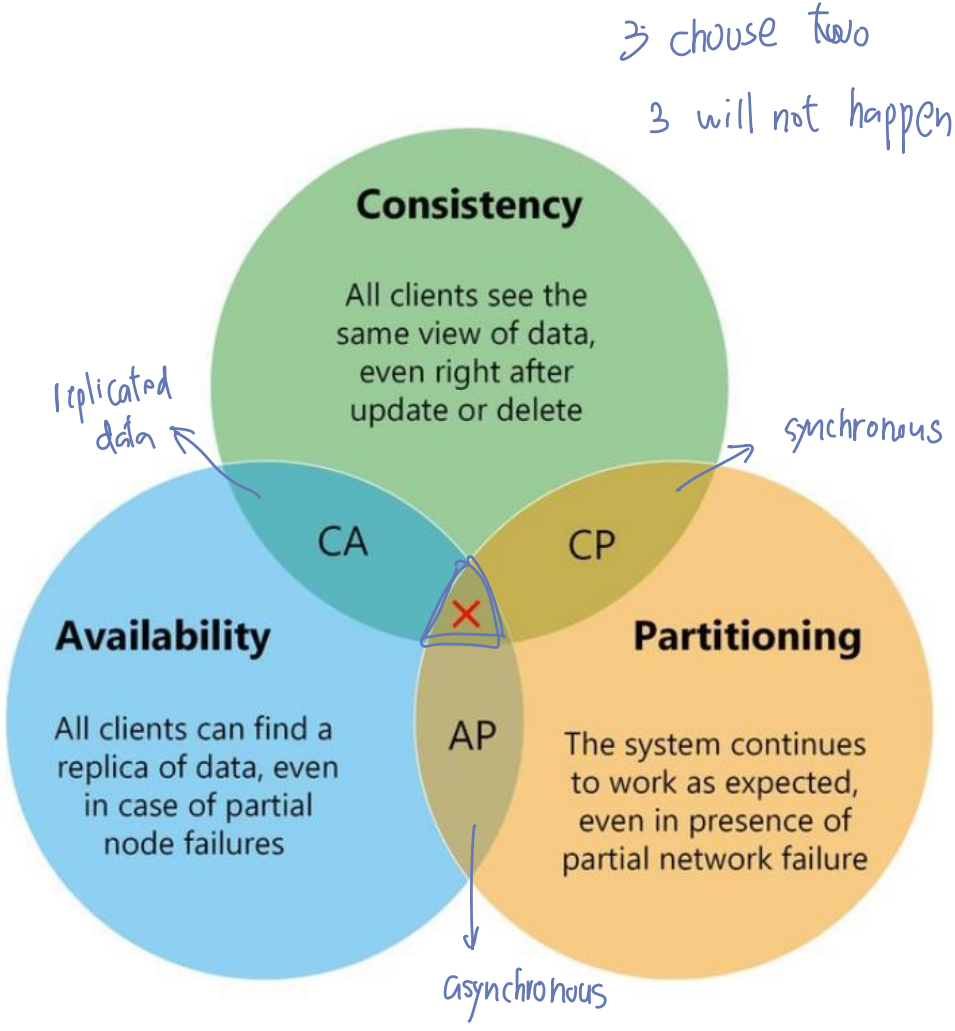
Suits some information systems more than others

- Compare commerce/finance systems with social media

CAP Theorem

The CAP theorem is a concept in distributed systems that states that it is impossible for a distributed database system to simultaneously provide more than two out of the following three guarantees ->

During a network partition, a distributed system must choose between consistency and availability.



Introduction to the CAP Theorem:
https://www.youtube.com/watch?v=gkg-FAEXIkY

Catalogue is a kind of group of files which contain metadata.

metadata is data about data, where data is and whether it's old version or updated version.

Distributed database catalogue

↘ It's smaller than actual data tables.

Distributed Catalog Management refers to the process of managing metadata about data objects that are distributed across multiple nodes or servers in a distributed database system.

This catalog contains information about the location and structure of the data objects, as well as the access rights and other attributes associated with each object.

Since catalogs are themselves databases that contain metadata about the distributed database system, they need to have a storage framework.

whatever changes are made, the metadata is written into catalogue

Three popular management schemes for distributed catalogs are:

- Centralised Catalogues → catalogue is on one server, and all server know where it is
- Fully Replicated Catalogues → metadata replicated on every server. So one server make update, it's urgently distributed to all outher server.
- Partially Replicated Catalogues ?

↳ It still more than one copy ↘ metadata about every object there will have more than one copy.

If one place fails, other place can give the catalogues.

A **distributed database catalog** is an essential component of a distributed database system that serves as a central repository containing metadata about the database. This catalog includes information about the database's structure, the location of its various parts, access paths, and other critical metadata required to manage, maintain, and query a distributed database effectively.

Key Features and Functions of a Distributed Database Catalog:

1. **Metadata Storage**: The catalog stores metadata, which includes:
    - **Schema Information**: Data about the database schemas, including tables, columns, data types, and relationships between tables.
    - **Location Transparency**: Information on where data is physically stored across the network. This feature allows the system to fetch and update data from multiple locations transparently, without requiring the user to know the data's actual physical location.
    - **Fragmentation Details**: Details if the data is fragmented (i.e., if different parts of a table are stored in different locations). This includes whether the data is horizontally or vertically fragmented.
    - **Replication Details**: Information about data replication across different sites, which helps in maintaining data consistency and availability.
    - **Security and Access Control**: Rules and policies regarding who can access or modify the data.
2. **Query Processing**: The catalog helps in query processing by providing necessary metadata that optimizes query execution. It directs queries to the appropriate locations and helps in combining data retrieved from multiple sites.
3. **Transaction Management**: For operations that involve multiple databases, the catalog provides necessary information to ensure transaction consistency and integrity across different sites. This includes details needed for implementing the two-phase commit protocol or other transaction coordination mechanisms.
4. **System Management**: The catalog aids in system management by keeping track of the health and status of different nodes in the distributed system, helping manage load balancing, and data redundancy.

Importance of Distributed Database Catalog:

- **Data Independence**: It provides logical data independence by separating the logical view (schema) of the database from its physical storage, thus allowing administrators to change the physical storage of data without affecting how the data is accessed by applications.
- **Efficiency and Optimization**: By storing details about data location and structure, the catalog allows the database system to optimize queries and reduce query response times by directing queries to the nearest or least-loaded nodes.
- **Scalability**: The catalog supports database scalability by managing distributed data transparently, allowing the system to add more nodes or change the data distribution without significant changes to the application logic.
- **Resilience and Fault Tolerance**: The information about data replication and distribution helps the system to continue functioning even if one or more database nodes fail.

Example Usage:

In a distributed database for a multinational corporation, the catalog would store information about employee data stored across servers in the US, Europe, and Asia. It would detail how employee records are partitioned by region and replicated across servers to ensure quick access and high availability. Queries asking for employees in a specific region would be routed to the appropriate server automatically, based on the metadata in the catalog.

Overall, the distributed database catalog is crucial for managing the complexity and enhancing the performance, scalability, and reliability of distributed database systems.

Comparing 5 configurations

1. Centralised database, distributed access
   - DB is at one location, and accessed from everywhere
2. Replication with periodic (asynchronous) snapshot update
   - Many locations, each data copy updated periodically
3. Replication with near real-time synchronization of updates
   - Many locations, each data copy updated in near real time
4. Partitioned, integrated, one logical database
   - Data partitioned across many sites, within a logical database, and a single DBMS
5. Partitioned, independent, non-integrated segments
   - Data partitioned across many sites.
   - Independent, non-integrated segments
   - Multiple DBMS, multiple computers

excellent: gray

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 inconsistency.

Legend:
the darker the colour
the better the feature

When we send the request, it locate data with distributed catalog through meta data.
Through that catalogue, it determines the location of the server is. Then it organise communication with server to bring the piece of data in.

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)

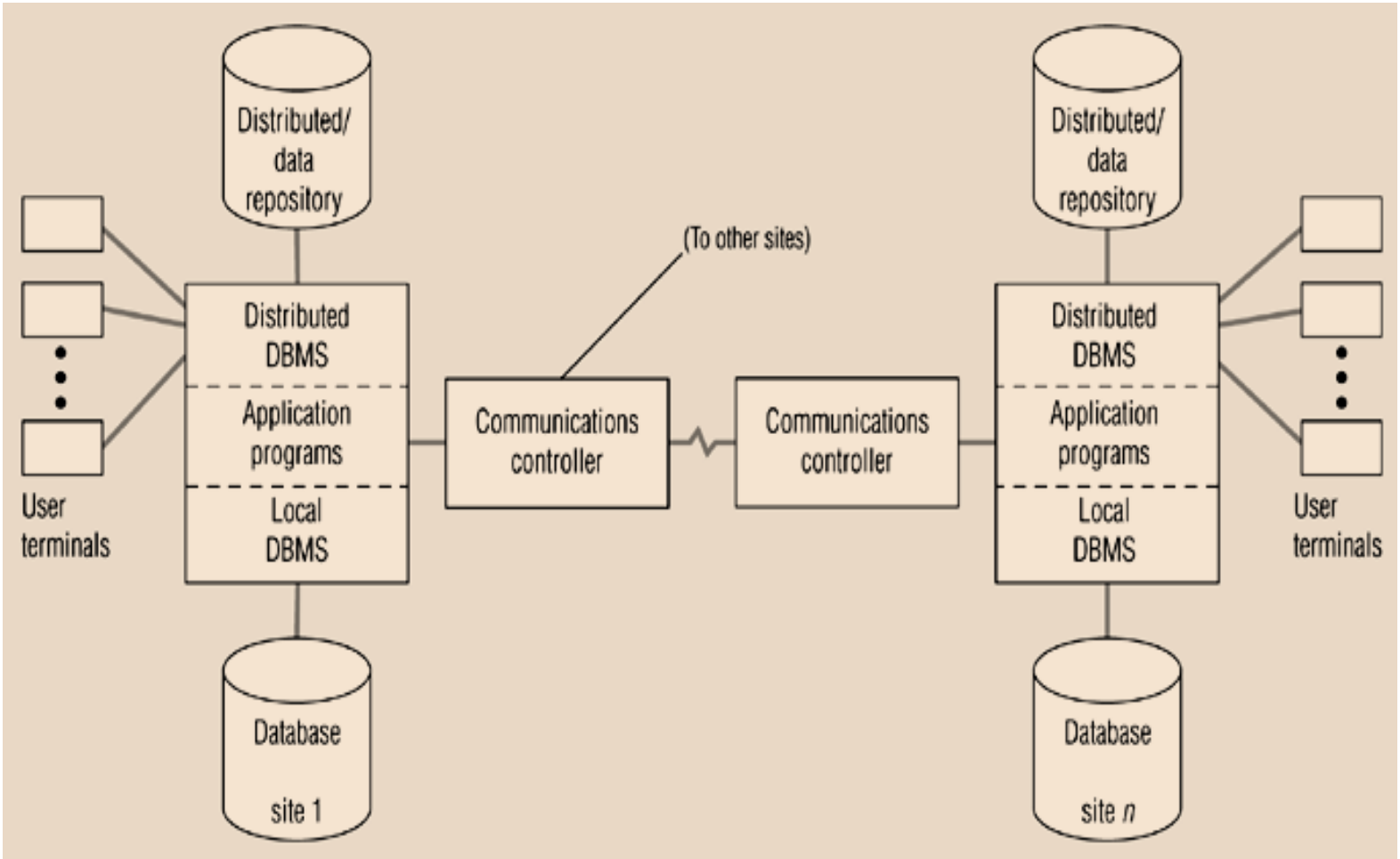
Data consistency (via multiphase commit protocols)

Global primary key control

Scalability

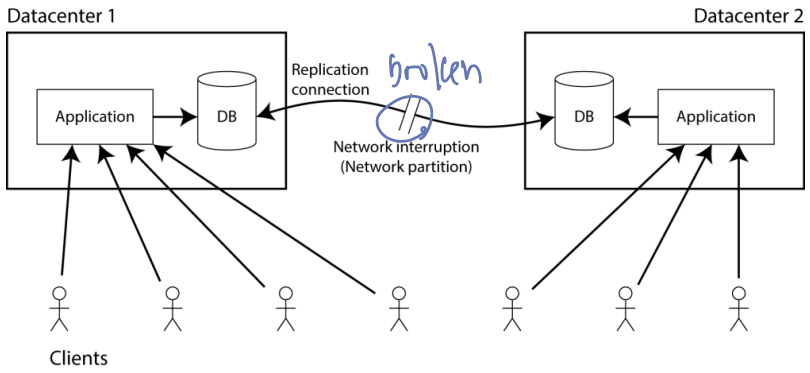
Security, concurrency, query optimisation, failure recovery

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) lose the update version.

(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