INFO20003 Database Systems

Dr Renata Borovica-Gajic*

Lecture 20

Distributed Databases

Week 10

slides adopted from David Eccles

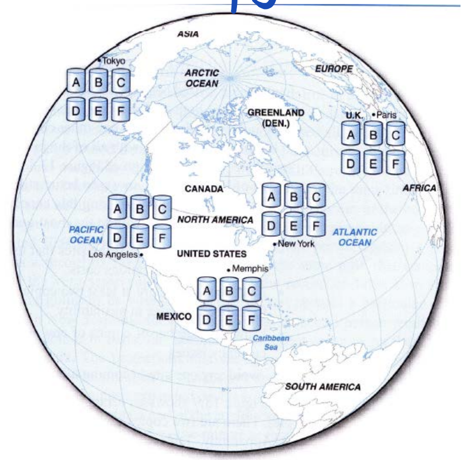
Today's lecture

- What is a distributed database?
- Why are they used, and how they work
- Pros and cons of different approaches

centralized database

distributed database

full copy



replicated database

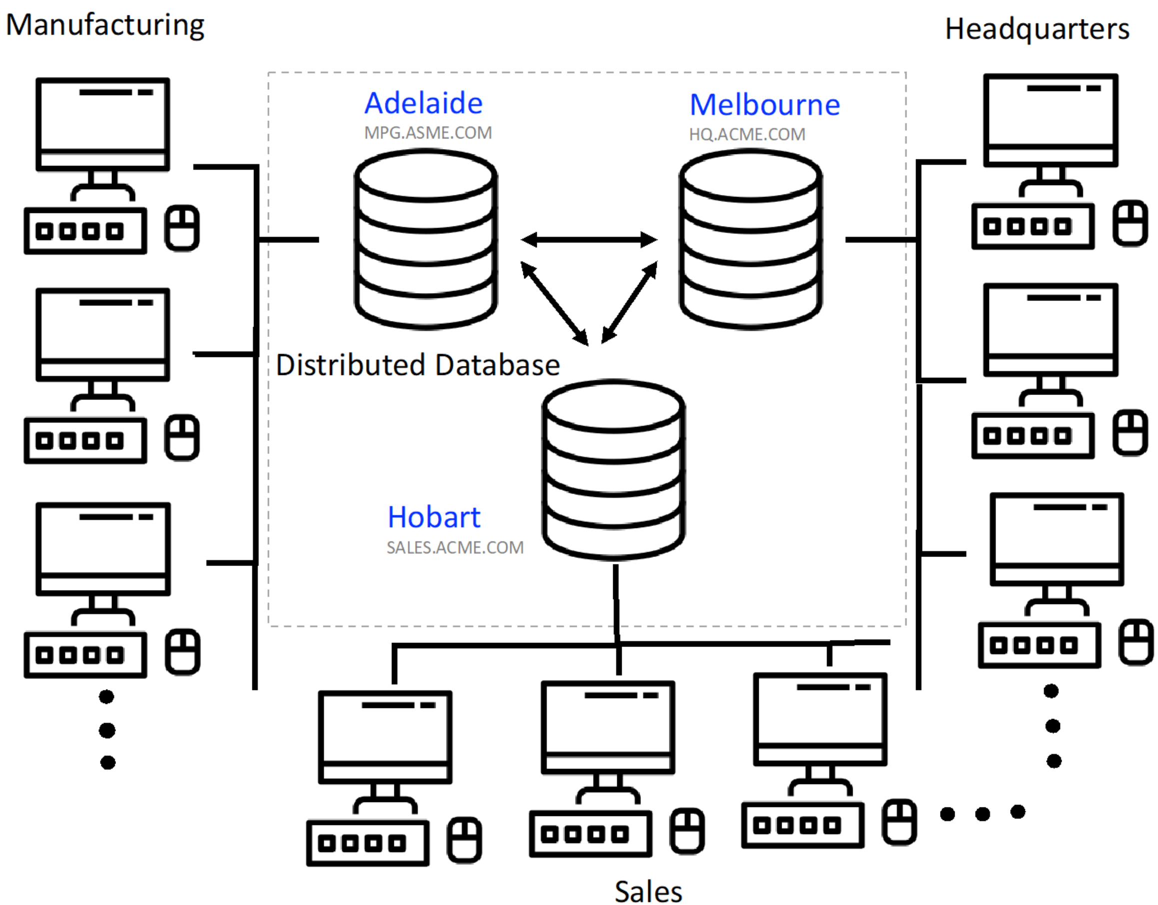
material in this lecture is drawn from Hoffer et al. (2013) Modern Database Management 11th edition, chapter 12, available online at http://wps.prenhall.com/bp_hoffer_mdm_11/230/58943/15089539.cw/index.html
pictures on this page are from Gillenson (2005) Fundamentals of Database Management Systems

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 though many databases*

- We are concerned with *distributed* databases

Example – distributed database



Advantages of distributed DBMS

- Good fit for geographically distributed organizations / users
  - Utilize the internet
- Data located near site with greatest demand
  - E.g. ESPN Weekend Sports Scores

AFL - Melbourne | EPL – London | NFL – New York | Hurling - Dublin

- Faster data access (to local data)
- Faster data processing
  - Workload split amongst physical servers

Workload
1 Server
(33,000
concurrent users)

(add more computational power to physical server)
⇒ buy memory, hard disk ...

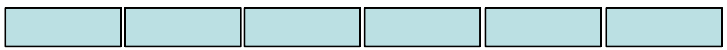
Workload
4 Servers
(33,000 users 9,000 + 12,000 + 7,000 + 5000)

(add more machines)

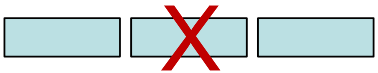
Vertical scaling VS. Horizontal scaling

Advantages of distributed DBMS

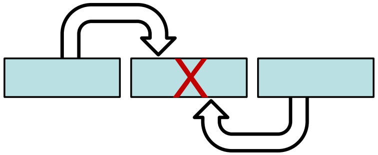
- Allows modular growth
  - add new servers as load increases (horizontal scalability)



- Increased reliability and availability
  - less danger of a single-point of failure (SPOF), IF data is replicated



- Supports database recovery
  - When data is replicated across multiple sites



THE UNIVERSITY OF MELBOURNE

Disadvantages of distributed DBMS

should satisfy data integrity
⇒ avoid anomaly

- Complexity of management and control
  - Database or/and application must stitch together data across sites
    - Who and where is the current version of the record (row & column)?
    - Who is waiting to update that information and where are they?
    - How does the logic display this to the web & 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 including network and storage infrastructure from both cyber & physical attacks

Disadvantages of distributed DBMS

- Lack of standards
  - Different Relational DDBMS vendors use different protocols
- Increased training & 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 of distributed DBMS

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

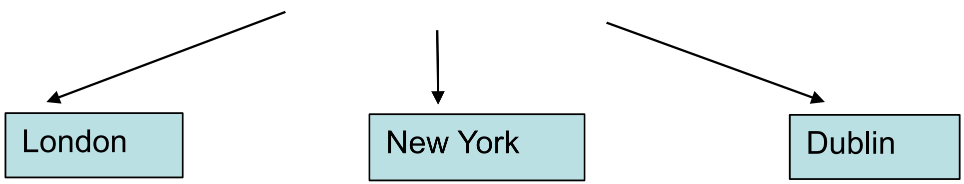
    doesn't care where to store data

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

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
- 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

- Being able to operate locally when connections to other databases fail

- Users can administer their local database → if connection is restored, database will synchronize and copy to other sites
  - control local data (e.g Hurling results)
  - administer security
  - log transactions
  - recover when local failures occur
  - provide full access to local data

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 optimization, failure recovery

Distribution options

- When distributing data around world – the data can be *partitioned* or *replicated*.
- **Data replication** is a process of *duplicating* data to different nodes.
- **Data partitioning** is the process of *partitioning* data into subsets that are shipped to different nodes.
- Many real-life systems use a combination of two (e.g. partition data and keep some replicas around -- usually 3)

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

when data is fully partition, when change happen, make sure whether it is unique

- Decoupled nodes don't affect data availability
  - Transactions proceed even if some nodes are down

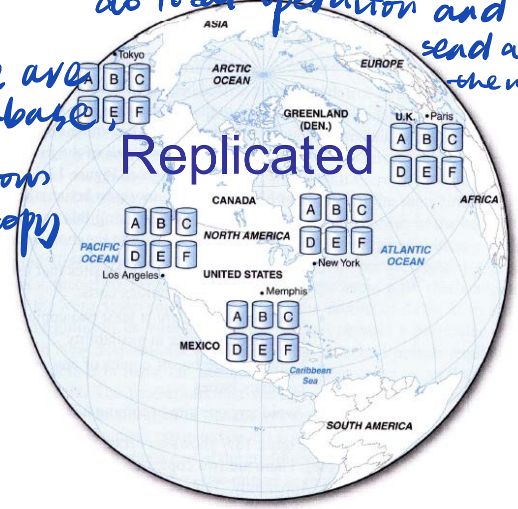
⇒ when fully copy, do local operation and send around the world

- Reduced network traffic at prime time
  - If updates can be delayed

when all people are access database,

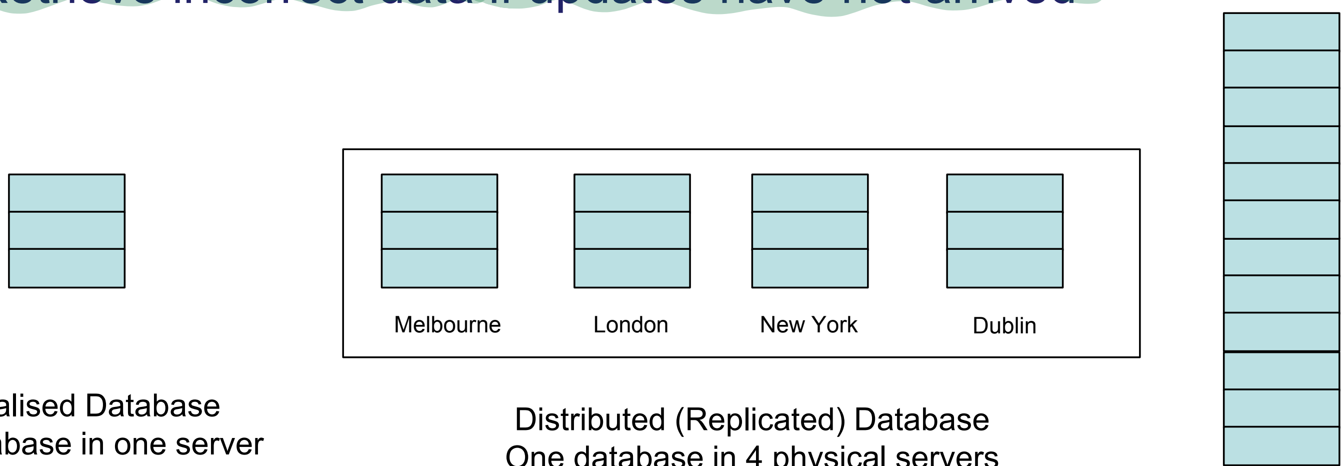
read from local copy

- This is currently popular as a way of achieving high availability for global systems
  - Most SQL & NoSQL databases offer replication



Replication - disadvantages

- Need more storage space
  - Each server stores a *copy* of the row
- Data Integrity:
  - High tolerance for out-of-date data may be required
  - Updates may cause performance problems for busy nodes
  - Retrieve incorrect data if updates have not arrived



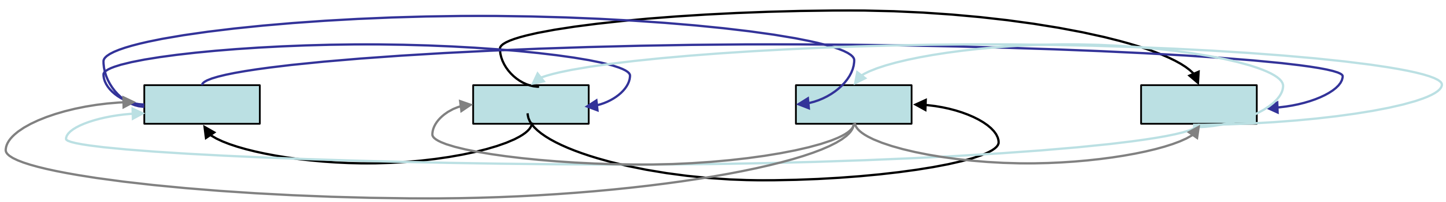
Centralised Database
One database in one server
(1 copy of data)

Distributed (Replicated) Database
One database in 4 physical servers
(4 copies of data)

Data Size

Replication - disadvantages

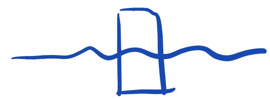
- Takes time for update operations → update need to propogated everywhere
  - High tolerance for out-of-date data may be required
  - Updates may cause performance problems for busy nodes



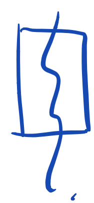
- Network communication capabilities
  - Updates can place heavy demand on telecommunications/networks
  - High speed networks are expensive ($$$$$)

Data partitioning

- Split data into chunks, store chunks in different nodes
- A chunk can be a set of rows or columns
- Thus, two types of partitioning: horizontal & vertical



- **Horizontal partitioning**
  - Table rows distributed across nodes (sides)
- **Vertical partitioning**
  - Table columns distributed across nodes (sides)



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)

eg. Melbourne node have AFL data

Team table

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

run a query will be efficient → (union of result, no rewrite)

→ send query to different node

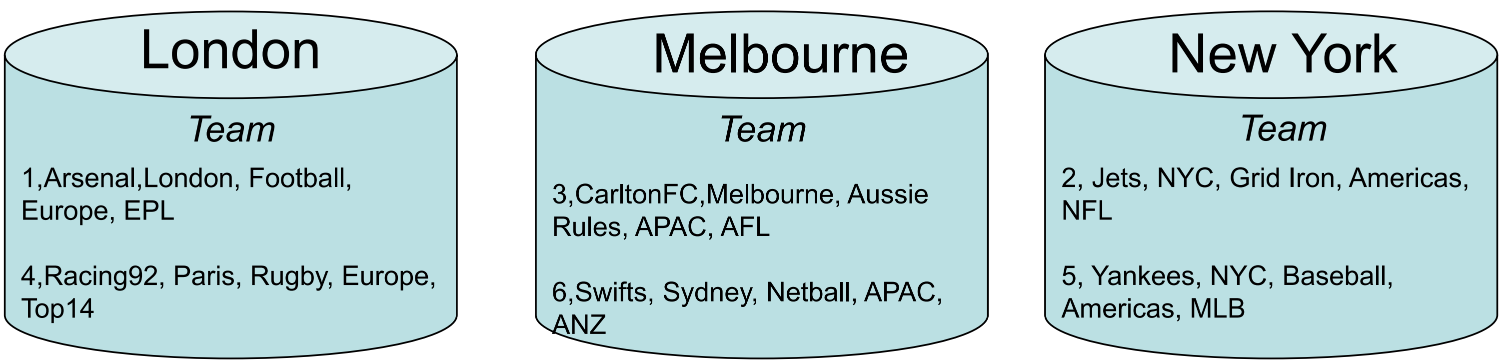
→ maybe one node busy

eg one site break down → combine with replica

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

Horizontal Partitioning based on Region



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)

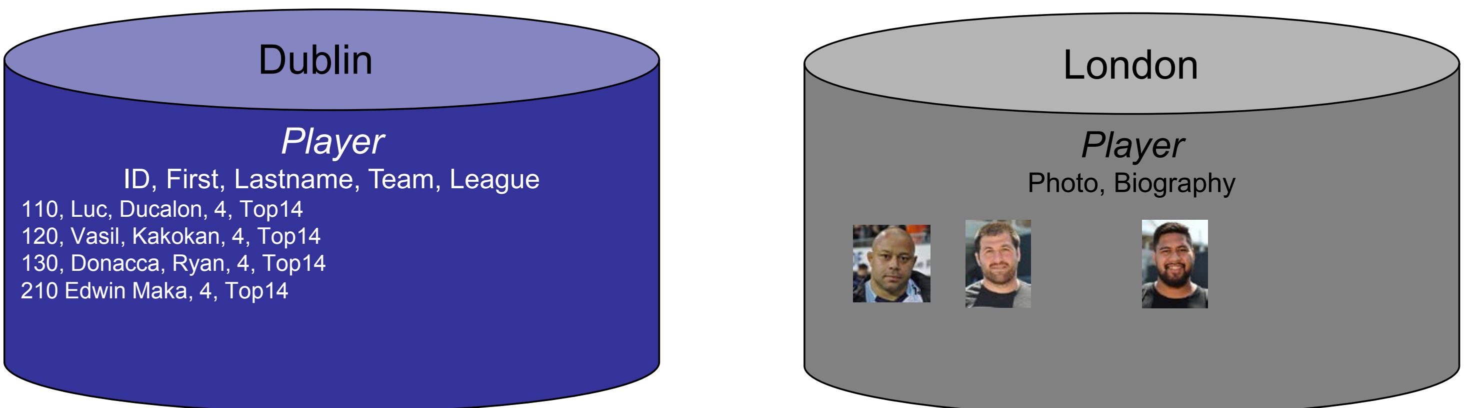
Player table

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



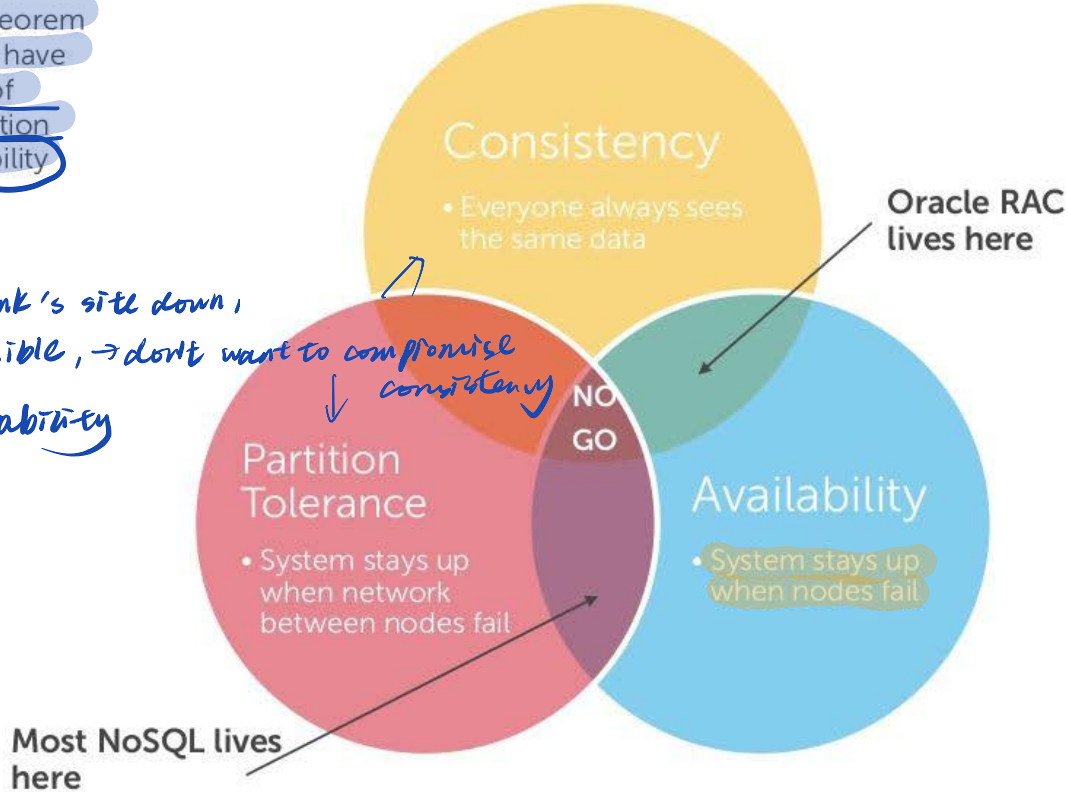
Trade-offs when dealing with DDBMS

- Trade-offs
  - Availability vs Consistency
    - The CAP theorem says we need to decide whether to make data always available OR always consistent
  - Synchronous vs Asynchronous updates
    - Are changes immediately visible everywhere (great BUT expensive) or later propagated (less expensive faster, but seeing stale data)?

The CAP theorem

CAP Theorem says something has to give

- CAP (Brewer's) Theorem says you can only have two out of three of Consistency, Partition Tolerance, Availability



when a bank's site down,
not responsible, → don't want to compromise consistency
↓
no availability

Synchronous updates

- Data is continuously kept up to date
  - users anywhere in the world 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 it is aborted
- Ensures data integrity and minimizes 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 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

synchronous asynchronous

THE UNIVERSITY OF MELBOURNE

What's examinable

- Advantages and disadvantages of DDBMS
- Distribution, partitioning and replication
- Synchronous vs asynchronous updates
- The CAP theorem

Next lecture

- NoSQL databases