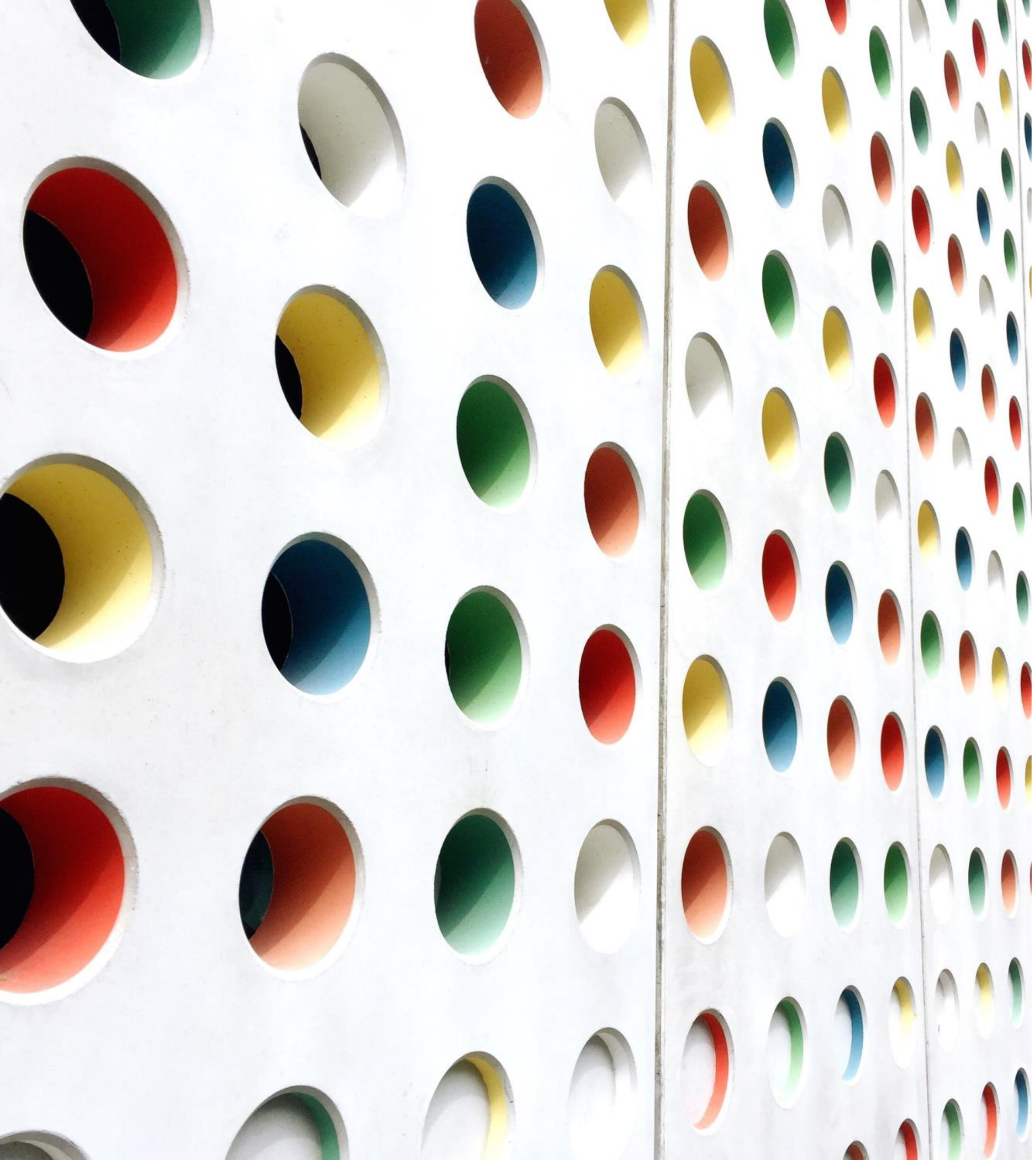
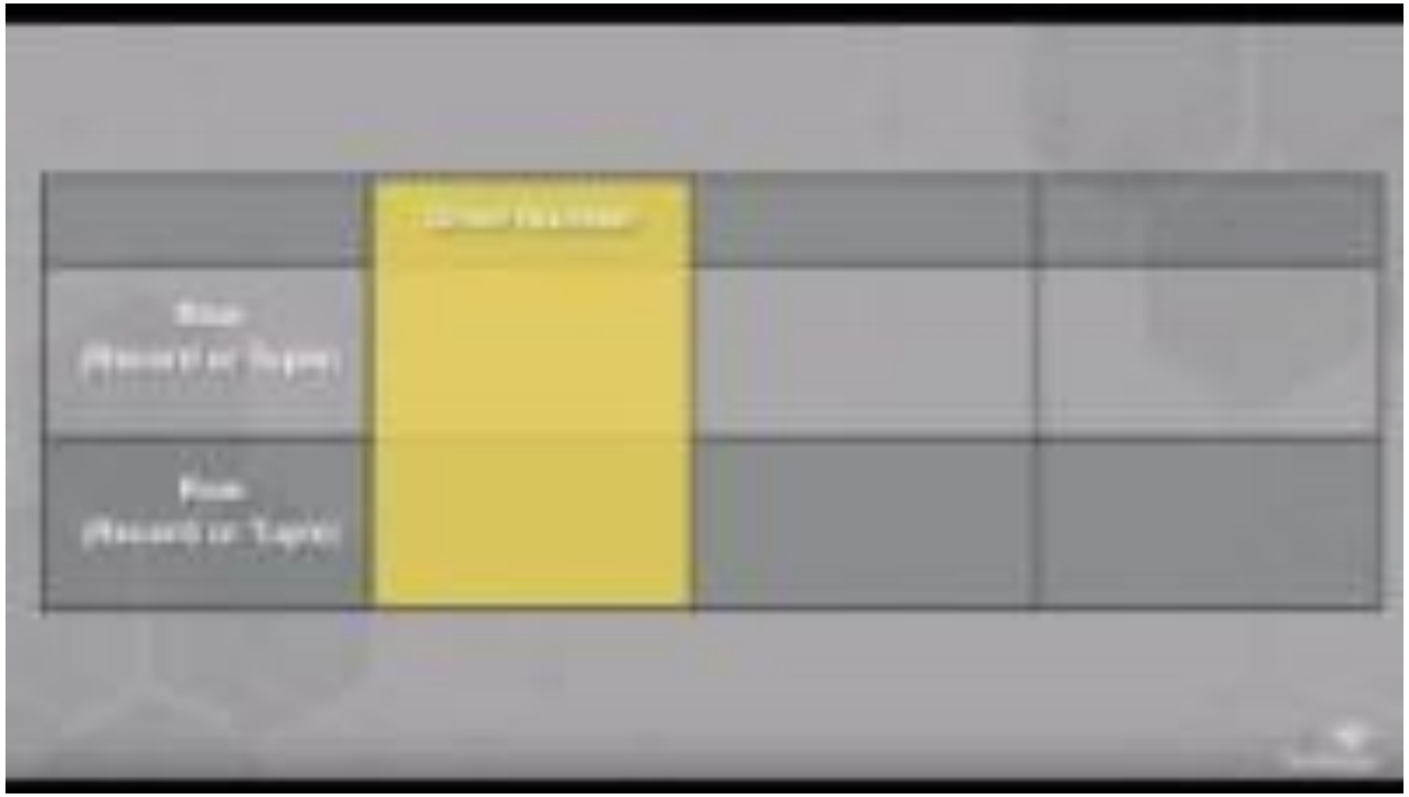


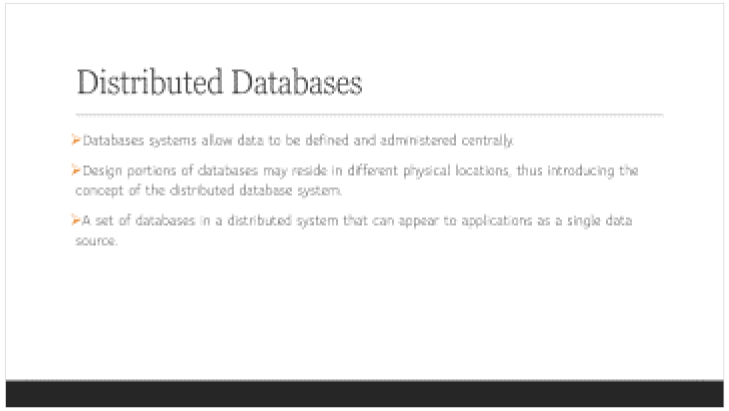
CS685 – DISTRIBUTED DATABASES

WEEK 1 – DISTRIBUTED DATABASES

Relational Database Review

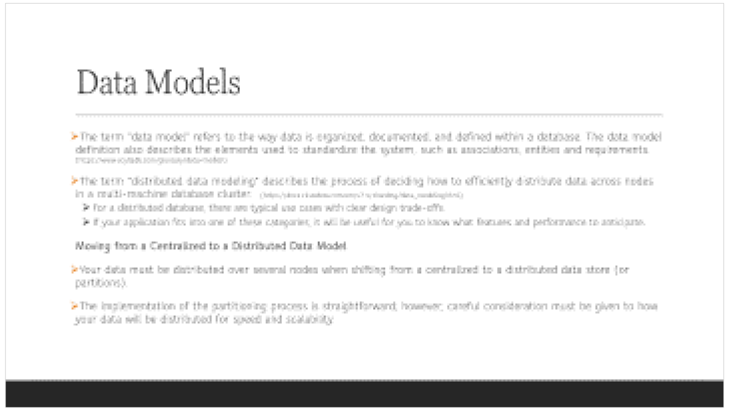


Topics



Distributed Databases

- Databases systems allow data to be defined and administered centrally.
- Design portions of databases may reside in different physical locations, thus introducing the concept of the distributed database system.
- A set of databases in a distributed system that can appear to applications as a single data source.

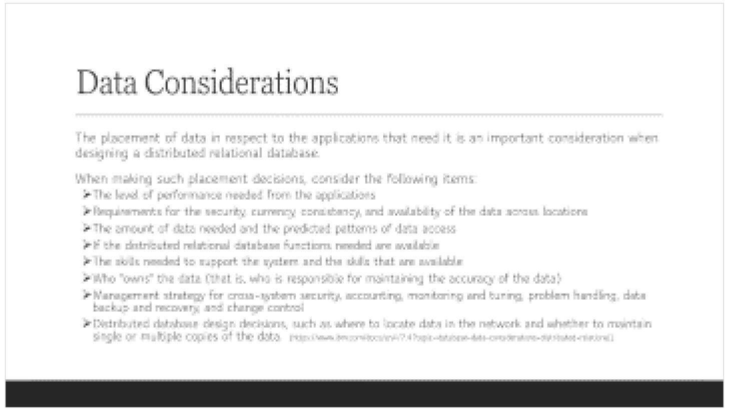


Data Models

- The term "data model" refers to the way data is organized, documented, and defined within a database. The data model definition also describes the elements used to standardize the system, such as associations, entities and requirements. [illegible]
- The term "distributed data modeling" describes the process of deciding how to efficiently distribute data across nodes in a multi-machine database cluster. [illegible]
  - For a distributed database, there are typical use cases with clear design trade-offs.
  - If your application fits into one of these categories, it will be useful for you to know what features and performance to anticipate.

Moving from a Centralized to a Distributed Data Model

- Your data must be distributed over several nodes when shifting from a centralized to a distributed data store (or partitions).
- The implementation of the partitioning process is straightforward, however, careful consideration must be given to how your data will be distributed for speed and scalability.



Data Considerations

The placement of data in respect to the applications that need it is an important consideration when designing a distributed relational database.

When making such placement decisions, consider the following items:
- The level of performance needed from the applications
- Requirements for the security, currency, consistency, and availability of the data across locations
- The amount of data needed and the predicted patterns of data access
- If the distributed relational database functions needed are available
- The skills needed to support the system and the skills that are available
- Who "owns" the data (that is, who is responsible for maintaining the accuracy of the data)
- Management strategy for cross-system security, accounting, monitoring and tuning, problem handling, data backup and recovery, and change control
- Distributed database design decisions, such as where to locate data in the network and whether to maintain single or multiple copies of the data. [illegible]

DISTRIBUTED DATABASES

DATA MODELS

DATA REQUIREMENTS

Distributed Databases

- Databases systems allow data to be defined and administered centrally.
- Design portions of databases may reside in different physical locations, thus introducing the concept of the distributed database system.
- A set of databases in a distributed system that can appear to applications as a single data source.

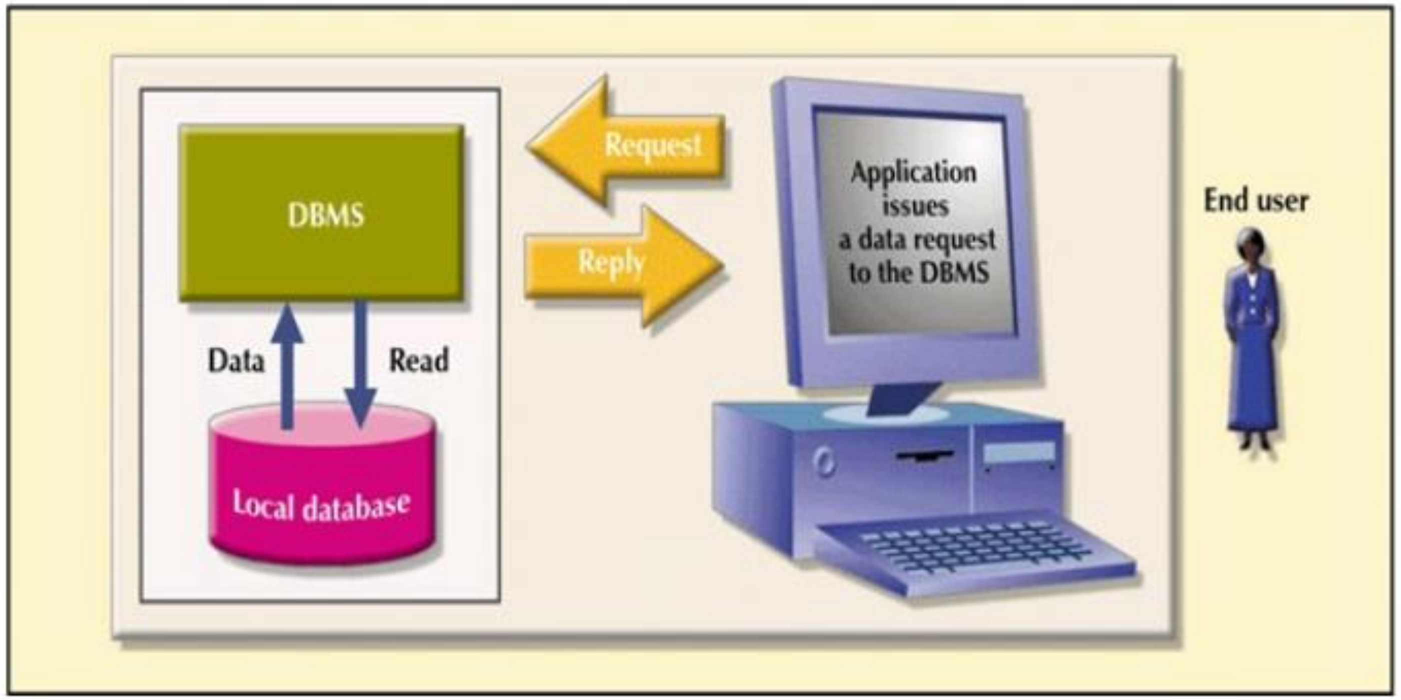
What does a DDBMS have?

To be considered a distributed database management system the DBMS must have:

- An application interface
- Validation to analyze data requests
- Transformation to identify distributed transactions from local ones
- Query optimization
- Mapping to identify data location and remote fragments
- I/O interface to read and write data from the permanent data store.
- Formatting to prepare data for presentation.
- Security to provide data privacy at both local and remote databases.
- Backup and Recovery to ensure availability and recoverability of the database.
- DB administration for the database administrator.
- Concurrency control to manage simultaneous data access
- Transaction management to ensure the data moves from one consistent state to another.

Centralized Database Management System

This image shows a typical example of a centralized database management system.



Database Systems: Design, Implementation, & Management, 6th Edition, Rob & Coronel

(Image source: https://slideplayer.com/slide/5702889/)

Advantages/Disadvantages of a DDBMS

- Advantages of DDBMS
  - Data can be distributed to match business processes.
  - Users only work with a subset of the companies' data, thus making access faster.
  - Allows for the processing of data at several sites.
  - New sites can be added without affecting the operations at other sites.
  - Local sites foster better communications.

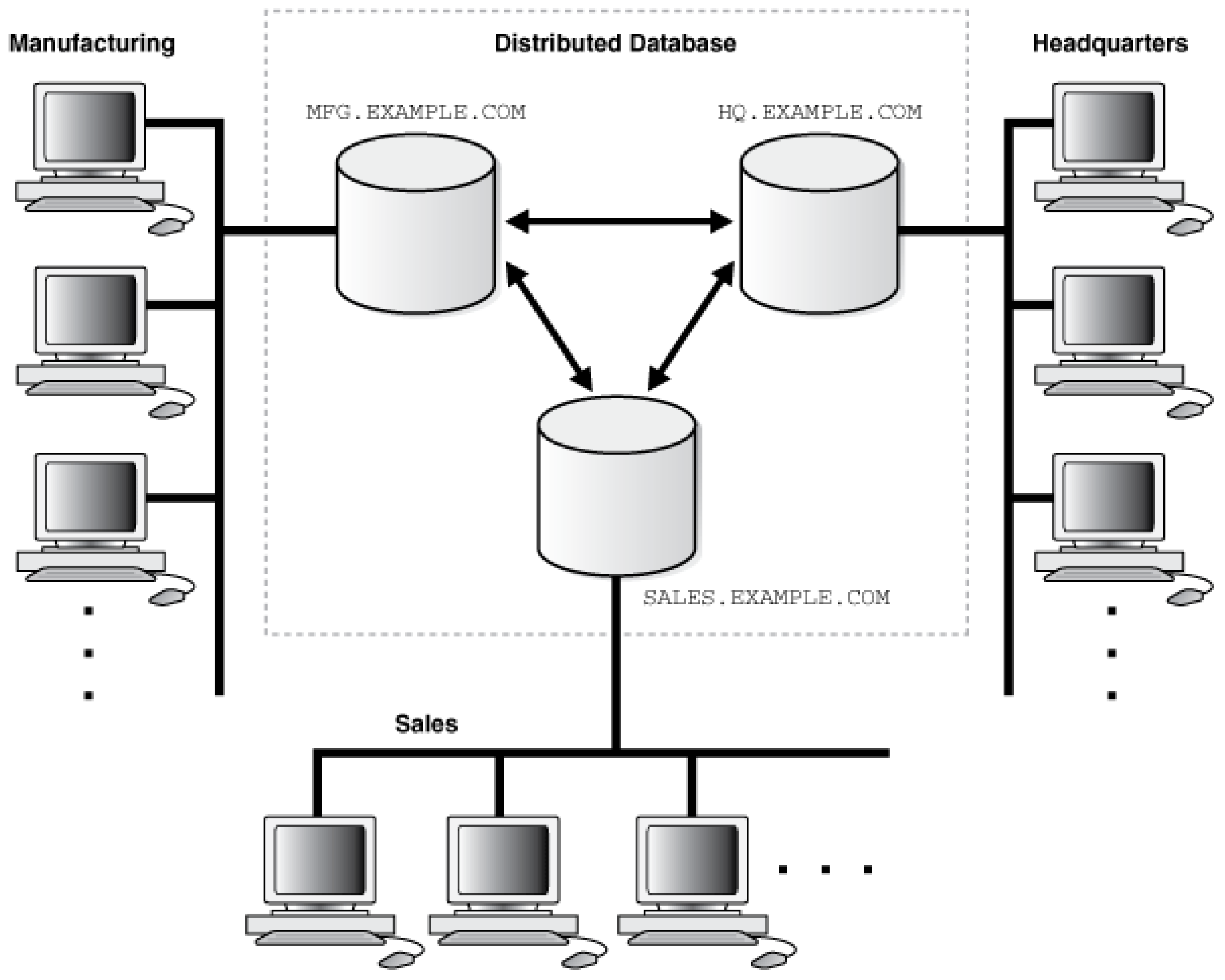
Advantages/Disadvantages of a DDBMS (continued)

- More effective to add workstations to the network, thus reducing operating cost.
- Provides a user-friendly graphical user interface.
- Less danger of a single point of failure.
- Provides processor independence since the users are accessing a local copy of the data.

- Problems faced by DDBMS
  - Management and control are very complex.
  - Probability of security lapses given the distributed nature of the data.
  - There is no standard communication protocol in place.
  - Data replication requires increases storage requirements.

Distributed Database Distribution

This is an example of what a distributed database system may look like.



(Image Source: https://docs.oracle.com/cd/E11882_01/server.112/e25494/ds_concepts.htm#ADMIN12075)

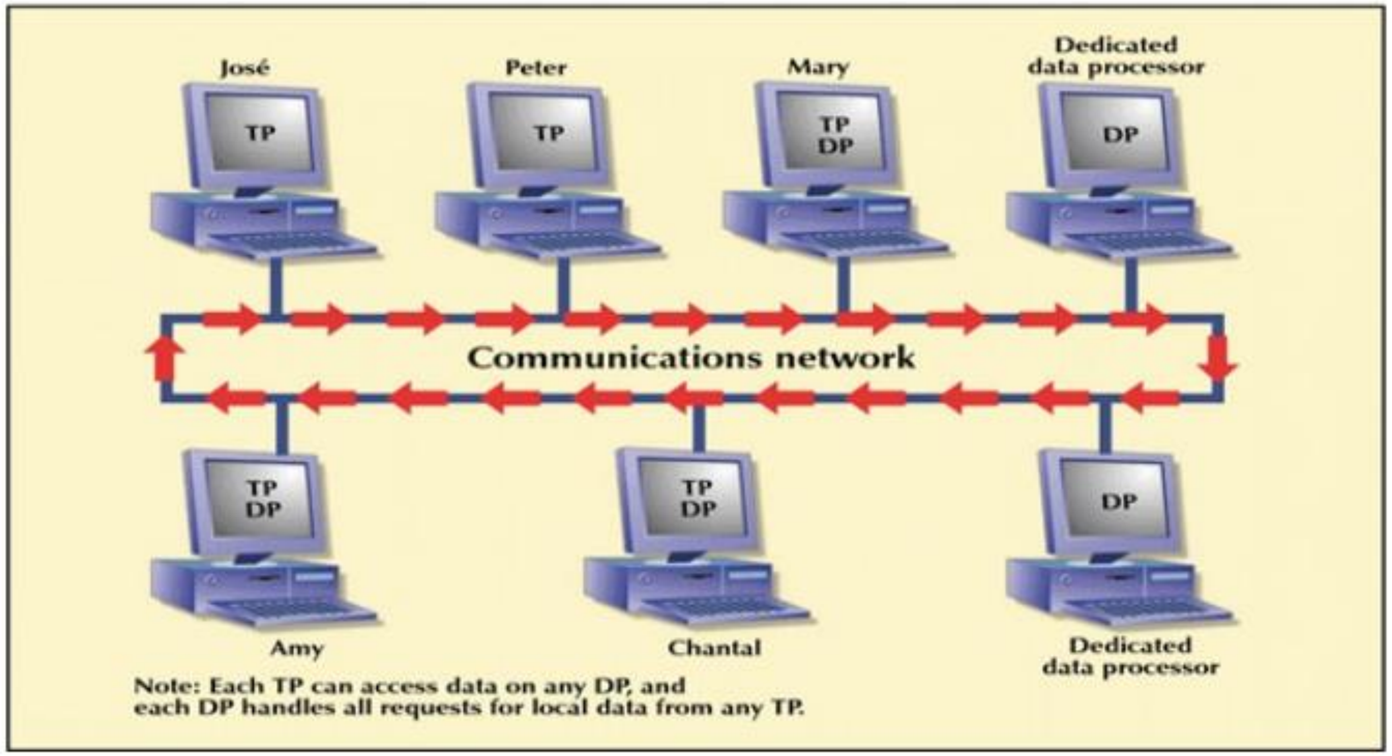
DDBMS Components

The distributed database management system must include at least the following components:

- Computer workstations
- Network hardware and software
- Communication media
- The transaction processor (tp) (also known as the application processor)
- The data processor (dp) (also known as the data manager (dm)

DDBMS Components

This slide shows the placement and interaction between the different components.



Database Systems: Design, Implementation, & Management, 6th Edition, Rob & Coronel

(Image source: https://slideplayer.com/slide/5702889/)

Data Models

- The term “data model” refers to the way data is organized, documented, and defined within a database. The data model definition also describes the elements used to standardize the system, such as associations, entities and requirements. (https://www.scylladb.com/glossary/data-model/)
- The term "distributed data modeling" describes the process of deciding how to efficiently distribute data across nodes in a multi-machine database cluster. (https://docs.citusdata.com/en/v7.1/sharding/data_modeling.html)
  - For a distributed database, there are typical use cases with clear design trade-offs.
  - If your application fits into one of these categories, it will be useful for you to know what features and performance to anticipate.

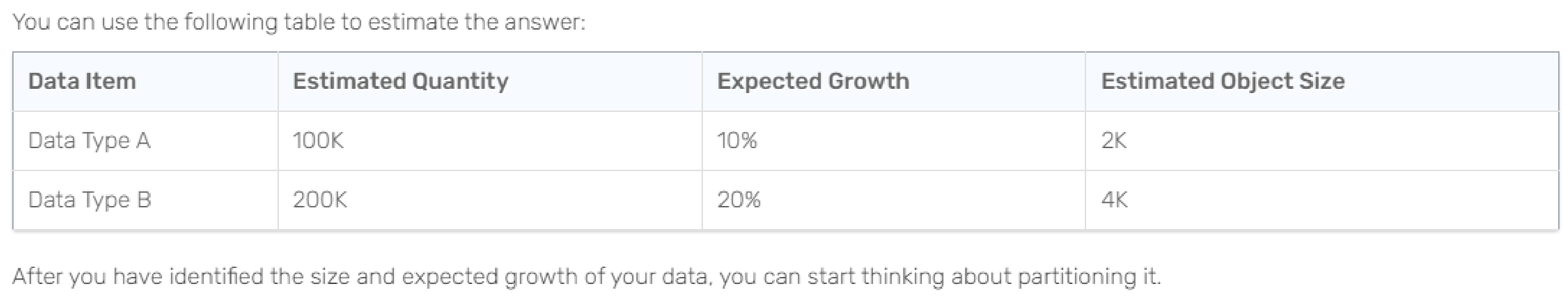
Moving from a Centralized to a Distributed Data Model

- Your data must be distributed over several nodes when shifting from a centralized to a distributed data store (or partitions).
- The implementation of the partitioning process is straightforward; however, careful consideration must be given to how your data will be distributed for speed and scalability.

Data Models (continued)

There are several questions that should be answered when planning your data partitioning:

- *What information I should store in memory?*
  - The answer to this question is not technical and should not be confused with the structure of the data. This is in essence a business question; how much the data will it grow over time, and for how long should it be kept?



You can use the following table to estimate the answer:

Data Item	Estimated Quantity	Expected Growth	Estimated Object Size
Data Type A	100K	10%	2K
Data Type B	200K	20%	4K

After you have identified the size and expected growth of your data, you can start thinking about partitioning it.

Data Models (continued)

What are my application's use cases?

- Modeling data based on the logical relationships between the data components is a frequent strategy. However, a different strategy is required for distributed data.
  - As a general guideline, avoid cross-cluster interactions since they can result in cross-cluster queries and updates that are typically significantly slower and less scalable than their local counterparts.
- When dealing with dispersed data, it is misleading to think of relationships as "one to one," "one to many," and "many to many."
  - How many distinct associations each entity has should be the first thing to consider.
  - An entity cannot be embedded into another entity if it is linked to several containers (parent entities).
  - The entity may not be able to be stored with all its components.

Types of Data Models

Conceptual Data Model

- ➢A conceptual data model identifies the entities that describe the data and relationships between them. Conceptual data models only show the highest-level relationships between entities, not attributes or primary keys within the data model.

Physical Data Model

- ➢A physical data model identifies the table structures that will be built in the database, including all tables, columns, primary keys and foreign keys used to identify the relationships between tables.

Relational Data Model

- ➢A relational data model is the basis for SQL databases. Relational data models have a fixed schema and deal with structured data. In a relational database management system, or RDBMS, the database is the outermost container that has data associated with an application.

Types of Data Models (continued)

Non-relational Data Model

- ➢ A non-relational data model offers a flexible schema design and can handle unstructured data—its storage model can be optimized to meet the requirements of the type of data being stored. Non-relational data models are used in non-relational databases, also called NoSQL databases.

Dimensional Data Model

- ➢ A dimensional data model is used in data warehouse design. Dimensional data models analyze numeric information (such as balances or values) in a data warehouse. By contrast, relational data models update, add or delete data in real-time information systems.

Enterprise Data Model

- ➢ An enterprise data model incorporates an industry perspective to give an unbiased view of how data is stored, sourced and used across an organization. Enterprise data models are useful for addressing the specific business needs of an enterprise.

Data Modelling Techniques

While data modelling techniques will vary depending on the type of database your organization uses, there are a few data modeling best practices to keep in mind in the data modelling process:

- ➢ Start with the data modelling basics: ask business teams what results they need from the data, and organize the data model around those requirements
- ➢ Build a draft data model with entities and relations, and test the model with best-case and worst-case scenarios
- ➢ Take database queries into account: you should know what your data looks like and what it contains, but also how you intend to query it
- ➢ Assess hardware requirements since servers working with huge datasets can soon run into problems of computer memory and input-output speed.
- ➢ Validate the data model: verify each action (such as your choice of primary key) before moving on to the next step (https://www.scylladb.com/glossary/data-model/)

Data Considerations

The placement of data in respect to the applications that need it is an important consideration when designing a distributed relational database.

When making such placement decisions, consider the following items:

- The level of performance needed from the applications
- Requirements for the security, currency, consistency, and availability of the data across locations
- The amount of data needed and the predicted patterns of data access
- If the distributed relational database functions needed are available
- The skills needed to support the system and the skills that are available
- Who "owns" the data (that is, who is responsible for maintaining the accuracy of the data)
- Management strategy for cross-system security, accounting, monitoring and tuning, problem handling, data backup and recovery, and change control
- Distributed database design decisions, such as where to locate data in the network and whether to maintain single or multiple copies of the data. (https://www.ibm.com/docs/en/i/7.4?topic=database-data-considerations-distributed-relational)

Architecture

- A **homogeneous** distributed database has identical software and hardware running all databases instances and may appear through a single interface as if it were a single database.
- A **heterogeneous** distributed database may have different hardware, operating systems, database management systems, and even data models for different databases.

(https://www.networxsecurity.org/members-area/glossary/d/distributed-database.html)

Architecture Approaches

There are two different approaches that can be used when developing a DDBMS.

Top-Down Approach:

1. Most used approach when implementing a DDBMS from the beginning.
2. Starts with requirements analysis, including an analysis of companies' problems and constraints.
3. Next is the conceptual and view design.
    1. Conceptual design deals with the entity relationship modeling and normalization. Here we determine the entity types and relationships among those entity types.
    2. View design deals with defining the interfaces for the end users.

Architecture Approaches

Bottom-Up Approach:

1. Used when the distributed database is already existing, and we have to add another database in an existing environment.
2. We can enhance existing features in the existing database during this design phase.
3. Important here is to determine the process for integrating multiple databases together.

(Hiremath and Kishor, 2016)

Homogeneous DDBMS

In **homogeneous distributed database** all sites have identical software and are aware of each other and agree to cooperate in processing user requests. Each site surrenders part of its autonomy in terms of right to change schema or software. A homogeneous DDBMS appears to the user as a single system. The homogeneous system is **much easier to design and manage**.

The following conditions must be satisfied for **homogeneous** database:

- ➢The operating system is used, at each location must be same or compatible.
- ➢The data structures used at each location must be same or compatible.
- ➢The database application (or DBMS) used at each location must be same or compatible.

Heterogeneous DDBMS

- In a **heterogeneous distributed database**, different sites may use different schema and software.
- Difference in schema is a major problem for query processing and transaction processing.
- Sites may not be aware of each other and may provide only limited facilities for cooperation in transaction processing. In heterogeneous systems, different nodes may have different hardware & software and data structures at various nodes or locations are also incompatible.
- Different computers and operating systems, database applications or data models may be used at each of the locations. For example, one location may have the latest relational database management technology, while another location may store data using conventional files or old version of database management system.

Heterogeneous DDBMS

- Similarly, one location may have the Windows NT operating system, while another may have UNIX. Heterogeneous systems are usually used when individual sites use their own hardware and software. On heterogeneous system, translations are required to allow communication between different sites (or DBMS). In this system, the users must be able to make requests in a database language at their local sites.
- Usually, the SQL database language is used for this purpose. If the hardware is different, then the translation is straightforward, in which computer codes and word-length is changed.
- The **heterogeneous** system is often not technically or economically feasible. In this system, a user at one location may be able to read but not update the data at another location.

Distribution of Data

In a distributed database system access to data at remote location is done in one of two ways:

Fragmentation: consists of breaking a relation into smaller relations or fragments and storing those fragments, possibly at different sites.

1. Major advantages of this process include:
   1. Efficiency
   2. Local optimization
   3. Easy of querying
2. Two types of fragmentation exists, and they are:
   1. Horizontal fragmentation – each fragment consist of a subset of rows of the original relation.
   2. Vertical fragmentation – each fragment consist of a subset of columns of the original relation.

Distribution of Data

Replication: several copies of the relation are stored at different site.

1. This helps in increasing reliability, locality and performance.
2. Advantages of this model include:
    1. Reliability
    2. Fast responses
    3. Possible avoidance of complicated distributed transaction integrity routines
    4. Node coupling
    5. Reduced network traffic at prime time.
3. Two types of replication include:
    1. Synchronous Replication – all copies of modified relation (fragment) must be updated before commit.
    2. Asynchronous Replication – allows for different copies of the same object to have different values for short periods of time. Data is updated at a predefined interval of time.

(Tomar and Megha, 2014)

Individual Project – Week 1

You have been asked to create a Distributed Database Implementation Plan for a local organization. The organization can be your choice of either a hypothetical example or an existing organization.
The document should be in the following format:

- **Distributed Database Implementation Plan**
  - Use Word
  - Title Page
    - Course number and name
    - Project name
    - Your name
    - Date
  - Table of Contents (TOC)
    - Use the autogenerated TOC.
    - Make it a maximum of 3 levels deep.
    - Be sure to update the fields of the TOC so it is up-to-date before submitting your project.
  - Section Headings (Create each heading on a new page with TBD as the content except for sections listed under New Content each week.)
    - **Distributed Database Needs Assessment (Week 1 IP)**
    - Distributed Query Processing Plan (Week 2 IP)
    - Data Distribution Transaction Management Plan (Week 3 IP)
    - Distributed Database Recovery Plan (Week 4 IP)
    - Distributed Database Performance and Security Plan (Week 5 IP)

Each week, **you will add a section to this Distributed Database Implementation Plan** and submit it for grading. Be sure that you support your information each week with scholarly sources and that you cite each source both in-text and in the References, section using APA format.

Individual Project – Week 1

In 4–5 pages, write a Distributed Database Needs Assessment that includes the following:

•Describe the general business environment for the case study organization, ensuring that all geographic locations and functional units are included.
•Describe the collection of logically related shared data used by the various geographic locations of the organization.
•Describe how the geographic locations are linked in a communications network.
•Describe the data needs for local and global applications for each geographic location based on the collection of logically related shared data.

Name the document "**yourname_CS685_IP1.doc**."

Submit the document for grading.

Please submit your assignment.

For assistance with your assignment, please use your text, Web resources, and all course materials. Find resources on how to write academically and use APA citations, including an example of Masters-level writing**, in the** Writing Style Guide for Master's Students**.**

References

Hiremath, D., & Kishor, S. (2016). Distributed database problem areas and approaches. IOSR Journal of Computer Engineering (IOSR-JCE), 2, 15-18.

Ozsu, M. T., & Valduriez, P. (2020). Principles of Distributed Database Systems (4 ed.). https://doi.org/10.1007/978-3-030-26253-2

Rob, P., & Coronel, C. (2000). Database Systems: Design, Implementation and Management (4th ed.). Course Technology.

Tomar, P., & Megha. (2014). An Overview of Distributed Databases. International Journal of Information and Computation Technology, 4(2), 207-214. https://www.ripublication.com/irph/ijict_spl/ijictv4n2spl_15.pdf