Distributed Systems

What is a database ?

A **database** is like a digital filing system where you store and organize information so you can easily find and use it later. Imagine you have a collection of notebooks where you write down all your friends' contact details—like their names, phone numbers, and addresses. Each notebook represents a **table** in a database, where each page in the notebook is a **record** (or **row**) that contains the information about one friend. The details on the page—like the name, phone number, and address—are the **fields** (or **columns**) in the table.

Here's a simple example:

- **Notebook (Database Table)**: Friends List
- **Page (Record/Row)**:
  - **Name (Field/Column)**: John Doe
  - **Phone Number (Field/Column)**: 123-456-7890
  - **Address (Field/Column)**: 123 Maple Street

In a database, you can quickly search through all the notebooks (tables) to find the information you need, like all the phone numbers in your Friends List. Instead of flipping through pages by hand, the database does it for you, making it fast and efficient to manage large amounts of information.

So, a database is like an organized collection of information that you can access, update, and manage easily, just like looking through your notebooks but much faster and more powerful.

Centralized Database

A **centralized database** is like a big, single library where all the books and information are stored in one place. Imagine a school where all the students' records—like their grades, attendance, and personal details—are kept in a single office. This office is the only place where the information is stored and managed. Whenever a teacher or a student needs information, they must go to this one office to get it.

In this scenario:

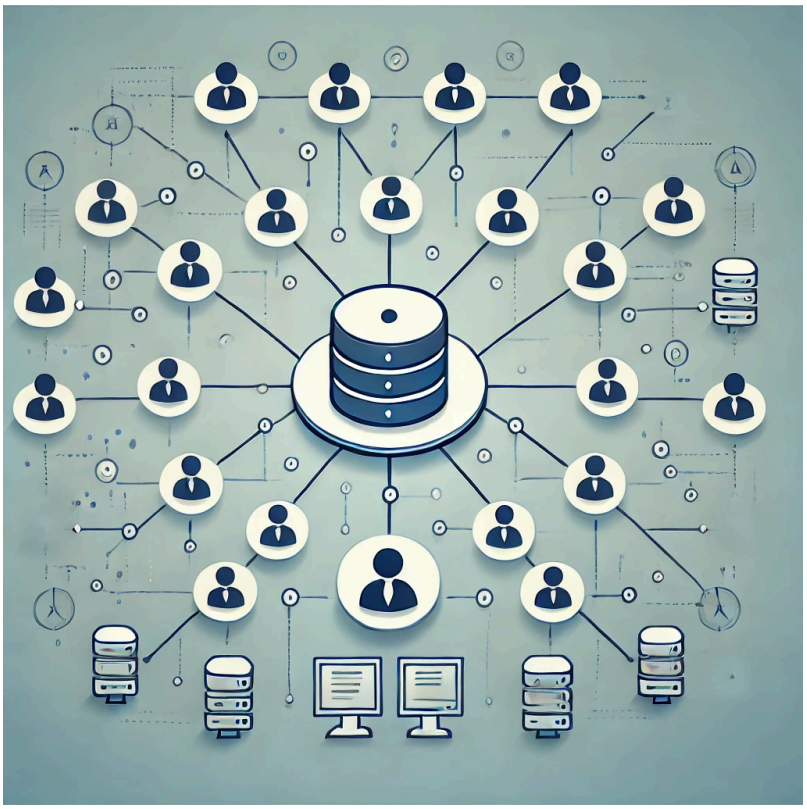
- **Library (Centralized Database)**: The office where all student records are kept.
- **Books (Data)**: The student records, including grades, attendance, etc.
- **Librarian (Database Administrator)**: The person managing and updating the records in the office.
- **Visitors (Users)**: Teachers or students who need to access the information.

Since everything is stored in one central location, it's easy to manage and ensure that the information is accurate and consistent. However, if something happens to this office (like it gets locked, or the files are lost), no one can access the information, which could be a downside.

In [ ]:

In [6]:
```
from PIL import Image

# Open the .webp image
im = Image.open('cd.webp')

# Convert it to .png format
im.save('cd.png')

# Now you can display the .png image
from IPython.display import Image as DisplayImage

DisplayImage(filename='cd.png', width=400, height=300)
```

Out[6]:



Disadvantages of a Centralized System:

1. **Single Point of Failure**:
    - If the central system fails (like a power outage or server crash), the entire network or organization can lose access to important data.
    - Example: If the main server in a company's IT system crashes, employees can't access their files.
2. **Performance Bottlenecks**:

- High traffic or demand on the central system can slow down performance for everyone.
- Example: During peak hours, students might struggle to access their online grades if the school's centralized system is overloaded.

3. **Limited Scalability**:
   - Expanding the system to accommodate more users or data can be challenging and costly.
   - Example: A small business with a centralized database may face difficulties as it grows and needs more storage and processing power.
4. **Security Risks**:
   - If the central system is hacked or breached, all the data is at risk.
   - Example: If hackers gain access to the central server of an online retailer, they could steal all customer information.
5. **Dependence on Network Connectivity**:
   - Users need a reliable network connection to access the centralized system; if the network goes down, so does access.
   - Example: Employees can't access a company's centralized cloud-based storage if the internet is down.

These points highlight the potential drawbacks of relying on a centralized system.

why we prefer distributed systems over centralized systems

1. **Reliability and Fault Tolerance**:
   - Distributed systems are more reliable because if one part fails, the rest can continue functioning.
   - Example: In a distributed database, if one server goes down, others can still provide access to the data.
2. **Scalability**:
   - Distributed systems can easily scale by adding more machines or resources without overloading a single point.
   - Example: Large websites like Amazon use distributed systems to handle millions of users simultaneously by distributing the load across many servers.
3. **Improved Performance**:
   - By distributing tasks across multiple machines, distributed systems can handle more work faster than a single centralized system.
   - Example: In a distributed computing system, complex calculations are divided among multiple computers, speeding up the processing time.
4. **Reduced Bottlenecks**:
   - Since tasks and data are spread across different nodes, there's less chance of one part becoming a performance bottleneck.

- Example: In a distributed content delivery network (CDN), data is served from multiple locations, reducing the load on any single server and improving access speed.

5. **Geographical Distribution**:
   - Distributed systems can operate across different locations, providing localized access to data and services, which can reduce latency.
   - Example: Cloud services like Google Drive use distributed systems to allow users worldwide to access their files quickly from nearby data centers.
6. **Cost Efficiency**:
   - Instead of investing in one large, expensive system, distributed systems can use multiple smaller, cost-effective machines.
   - Example: A startup might use a distributed system of smaller servers to handle their needs instead of buying a single powerful server.
7. **Flexibility and Modularity**:
   - Distributed systems can be more flexible, allowing different parts to be updated, replaced, or expanded without affecting the whole system.
   - Example: A distributed microservices architecture allows developers to update individual services without disrupting the entire application.

These advantages make distributed systems more suitable for many modern applications, particularly those requiring high availability, scalability, and performance.

In [ ]:

Distributed system

A **distributed system** is like a group of friends working together on a big project. Instead of one person doing all the work, each friend takes on a part of the task. They might be in different places, but they communicate and share their work to complete the project together.

Simple Example:

Imagine you and your friends are planning a school event. Instead of one person organizing everything, you divide the tasks:

- **Friend 1**: Handles the invitations.
- **Friend 2**: Organizes the decorations.
- **Friend 3**: Takes care of food and drinks.
- **You**: Manage the schedule and activities.

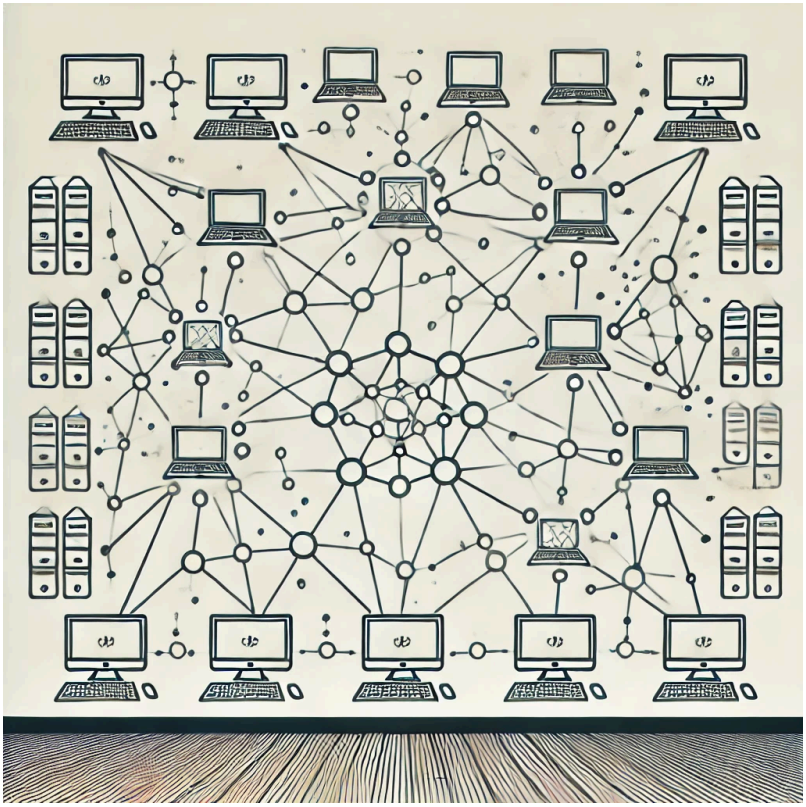
Each friend works on their part separately, but together, you make the event happen. If one friend can't complete their task, the others can step in to help, so the event doesn't fail. You all communicate to make sure everything is on track.

In this example, the **event** is like a **distributed system**, where the tasks (data and processes) are shared among multiple people (computers or servers). Each person (server) is responsible for a specific part, and by working together, they accomplish the goal efficiently and reliably.

In [ ]:

In [9]:
```
from PIL import Image

# Open the .webp image
im = Image.open('dd.webp')

# Convert it to .png format
im.save('dd.png')

# Now you can display the .png image
from IPython.display import Image as DisplayImage

DisplayImage(filename='dd.png', width=400, height=300)
```

Out[9]:



Advantages and Dis-Advantages of Distributed Systems

Advantages of Distributed Systems:

1. **Reliability**:
   - If one part of the system fails, the rest can keep working.
   - Example: If one server in a network goes down, others can continue to operate.
2. **Scalability**:
   - You can easily add more computers or servers to handle more work.

- Example: As a website grows, you can add more servers to manage increased traffic.

3. **Flexibility**:
   - Parts of the system can be updated or changed without affecting the whole system.
   - Example: You can update one part of an app without shutting down the entire service.
4. **Improved Performance**:
   - Tasks are divided among multiple machines, making things faster.
   - Example: Complex calculations are completed quicker when split across several computers.
5. **Geographical Distribution**:
   - Services can be accessed from different locations, reducing delays.
   - Example: A user in India can access a server in Asia, while a user in the U.S. can access a server in North America.

Disadvantages of Distributed Systems:

1. **Complexity**:
   - Managing and coordinating multiple systems can be complicated.
   - Example: Ensuring all parts of a system communicate properly can be challenging.
2. **Security Risks**:
   - More points of access can mean more security vulnerabilities.
   - Example: Hackers might target weaker parts of a distributed network.
3. **Consistency Issues**:
   - Keeping data consistent across different locations can be difficult.
   - Example: Making sure all servers have the same updated information can be tricky.
4. **Cost**:
   - Setting up and maintaining a distributed system can be expensive.
   - Example: More servers and networking equipment might be needed, increasing costs.
5. **Network Dependence**:
   - The system relies heavily on network connections; if the network fails, the system can be disrupted.
   - Example: A poor internet connection can slow down communication between parts of the system.

Overview of database and computer network concepts

Databases

A **database** is an organized collection of data that is stored and managed in a way that makes it easy to access, update, and manage. Here are some basic concepts related to databases:

1. **Data**: Raw facts and figures that can be processed to produce information. For example, a list of names and phone numbers.
2. **Database Management System (DBMS)**: Software that manages databases. It allows users to create, read, update, and delete data in a database. Examples include MySQL, Oracle, and Microsoft SQL Server.
3. **Tables**: In a relational database, data is organized into tables. A table consists of rows (records) and columns (fields). For example, a table for storing information about students might have columns for ID, name, age, and grade.
4. **Primary Key**: A unique identifier for each record in a table. No two rows can have the same primary key value.
5. **Foreign Key**: A field in one table that links to the primary key of another table, establishing a relationship between the two tables.
6. **SQL (Structured Query Language)**: A programming language used to interact with a database. It allows users to perform tasks like retrieving specific data, adding new records, and updating or deleting existing records.
7. **Normalization**: The process of organizing data in a database to reduce redundancy and improve data integrity. This involves breaking down tables into smaller, related tables.
8. **Index**: A database index improves the speed of data retrieval operations on a table by providing quick access to the rows.
9. **Transaction**: A sequence of operations performed as a single logical unit of work. Transactions ensure data integrity by following the ACID properties (Atomicity, Consistency, Isolation, Durability).

Computer Networks

A **computer network** is a group of interconnected computers that communicate with each other to share resources like files, printers, and internet connections. Here are some basic concepts related to computer networks:

1. **Network**: A collection of computers and devices connected together to share resources and information.
2. **Nodes**: Devices on a network, such as computers, printers, and routers, are referred to as nodes.
3. **LAN (Local Area Network)**: A network that covers a small geographical area, like a single building or campus. It connects computers and devices within that area.
4. **WAN (Wide Area Network)**: A network that covers a large geographical area, such as a city, country, or even the globe. The internet is the largest example of a WAN.
5. **Internet**: A global network that connects millions of private, public, academic, business, and government networks.
6. **IP Address**: A unique identifier assigned to each device connected to a network. It allows devices to locate and communicate with each other over the network.
7. **Router**: A device that forwards data packets between computer networks, directing the data along the most efficient route.
8. **Switch**: A network device that connects multiple devices on a LAN and uses MAC addresses to forward data to the correct destination.
9. **Protocol**: A set of rules that govern how data is transmitted over a network. The most common protocol for communication on the internet is TCP/IP (Transmission Control Protocol/Internet Protocol).

10. **Bandwidth**: The maximum rate of data transfer across a network. It's usually measured in bits per second (bps).
11. **Firewall**: A security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules, helping to protect the network from unauthorized access.
12. **Network Topology**: The arrangement or layout of devices in a network. Common topologies include star, ring, bus, and mesh.
13. **DNS (Domain Name System)**: A system that translates domain names (like www.example.com (http://www.example.com)) into IP addresses, allowing users to access websites using easy-to-remember names instead of numerical IP addresses.

Summary

- **Databases** help in storing and managing data efficiently, allowing easy access, updates, and management through software like DBMS.
- **Computer Networks** connect multiple devices, allowing them to communicate and share resources. They use various technologies and protocols to manage data transfer and ensure security.

In [ ]: