Comprehensive Database Design and Management for Enhanced Organizational Efficiency

Task 1

A database is a structured collection of data or information that is specifically designed to simplify the management, retrieval, and access processes. A Database Management System (DBMS) is a computer language used to efficiently organize and manipulate large amounts of data in an organized way. Users can employ this function to efficiently manage security and access regulations for the database, as well as perform tasks such as database construction, transformation, and querying. The DBMS has several objectives to achieve. "Databases are foundational tools in managing vast amounts of information effectively and securely" (Connolly & Begg, 2015).

The main purpose of an organizational database system is to collect the necessary information for running the company and providing access to individuals who have an appropriate requirement for it.The main objectives of databases are to guarantee the physical preservation of data and to provide users with control over the interpretation and organization of the data. The database ensures a reliable and accurate level of uniformity. The data should be easily accessible for retrieval and utilization by both present and future queries and applications. There are multiple objectives to improve the effectiveness and simplify the management of data within an organization. The main objectives are the organization and categorization of data, together with the assurance of data integrity and consistency. "A well-designed database system serves as the backbone of organizational data management" (Kroenke & Auer, 2013).

The database of a public library performs various functions, including preserving staff records, coordinating loan and return transactions, storing member information, calculating costs, and organizing book categorization. The goal is to improve user experience and optimize library operations by effectively managing and accessing data. For example, economic models use production and consumption data to distribute resources and strategize accordingly. Medical services depend on patient records, disease histories, and data on the effectiveness of treatment for problem classification. For instance, within the government sector, at each hierarchical level, there are documented records of individuals who fulfill their tax obligations and properties that are liable to be taxed. "Efficient data management in libraries can significantly enhance service delivery and operational efficiency" (Rob & Coronel, 2007).

DBMS efficiently organizes and manages vast quantities of data. However, these activities are essential for preserving data integrity, guaranteeing security, and enabling easy access. Here are several functions carried out by a Database Management System (DBMS): Management of data storage: It manages the storage of data on storage devices and improves the scheduling and distribution of space for data, as well as the arrangement of files, to optimize performance and data manipulation. It includes a range of data manipulation operations, including changing existing records, deleting records, and querying the database. A Database Management System (DBMS) primarily functions to construct the foundational structure of a database, encompassing its organization, interconnections, and tabular representations. "The core functions of a DBMS revolve around data organization, retrieval, and security management" (Elmasri & Navathe).Furthermore, it includes various data types, main keys, foreign keys, and extra attributes.

"Understanding the roles and needs of different stakeholders is crucial for successful database implementation" (Connolly & Begg, 2015).The stakeholders involved with the creation and implementation of a database for a public library include a diverse group of individuals and entities, each having unique interests and responsibilities: Library administration: It refers to an unique category of software systems that include functionalities for generating, regulating, storing, authorizing access to, and preserving digital artworks and collections. These systems are utilized by IT departments, library personnel, donors, and sponsors, among other individuals.

To protect sensitive staff and member information, it is crucial for the public library's database to maintain data security and integrity. "Data security and scalability are paramount in the design of modern database systems" (Kroenke & Auer, 2013).

Maintaining scalability is crucial for efficiently handling the increasing quantity of books, members, and transactions. To provide efficient storage and retrieval of information by employees, the system must have usability, characterized by a user-friendly interface. To achieve efficient data collection and modification, it is necessary to optimize queries for optimal performance. In addition, the database should have the ability to generate comprehensive reports that provide detailed information for management. Following to data protection rules is crucial for upholding legal and ethical norms. Together, these requirements allow for the efficient and organized execution of library tasks.

In recent years, healthcare organizations have increasingly turned to data analytics to optimize their operations and resource allocation. By analyzing healthcare datasets, hospitals can gain insights into patient needs, improve the efficiency of their operations, and allocate resources more effectively. "Data analytics in healthcare can lead to better patient outcomes and more efficient hospital operations" (Connolly & Begg, 2015).

"An effective Healthcare Management System is vital for the smooth functioning of healthcare facilities" (Wager, Lee, & Glaser, 2017).A Healthcare Management System is utilized to manage various aspects of healthcare facilities, including patient records, appointments, billing, inventory management, and doctor schedules. The database architecture for such a system is essential for maintaining data integrity, confidentiality, and efficiency in managing healthcare-related data.

First of all, the features in hospital operation optimization are determined as patient management, appointment management, billing management, inventory management and doctor management. Patient management includes information about the patient. These are could be disease history, age range, signs, symptoms and other health information. Appointment management is about when the patient comes, for what purpose, which doctors they apply and for what purpose. Billing management is about the patient's payment history from the moment they apply to the hospital and their health insurance, if any. Inventory management includes information about medical supplies and equipment. Doctor management includes information about the doctor. These are could be the doctor's areas of expertise, calendar and contact details.Also, the design of the database for such a system is essential for maintaining data integrity, confidentiality, and efficiency in managing healthcare-related data. "Comprehensive healthcare management systems ensure better coordination and care delivery" (Elmasri & Navathe, 2015).

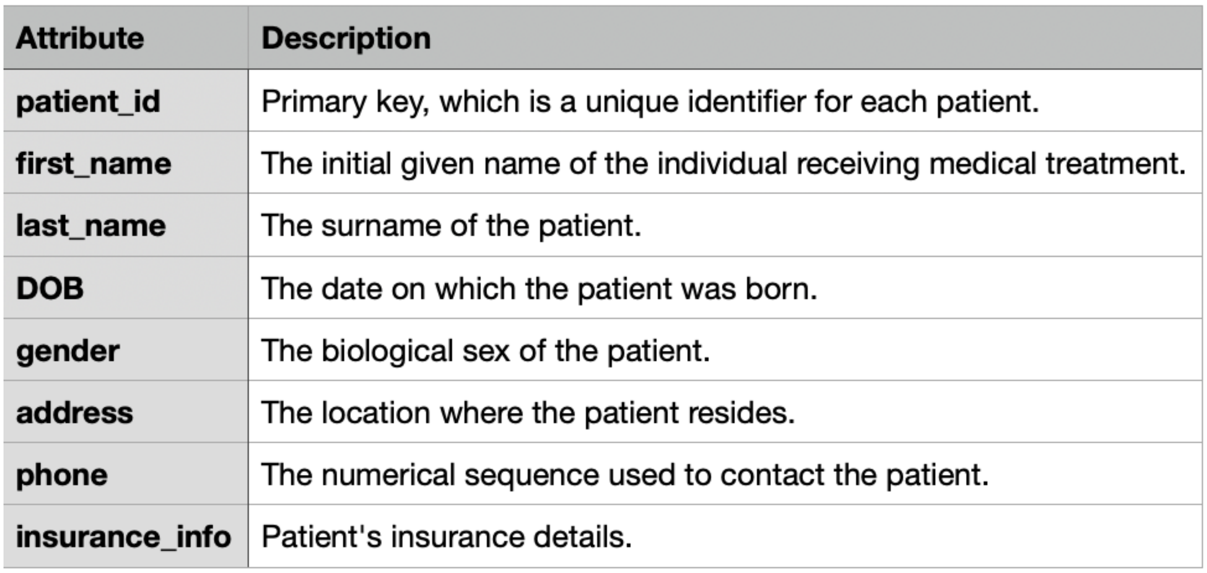
Features of a Healthcare Management System

Patient Management: Provide patient data, including medical records, demographic information, and insurance details.
Appointment Management: Organizing and managing appointments for patients to meet with doctors.
Billing Management: Manage the administration of billing and invoicing for healthcare services provided to patients.
Inventory Management: Manage the organization and control of medical supplies and equipment.
Doctor Management: Manage the organization and administration of doctor-related data, including areas such as specializations, timetables, and contact information.

Components and Characteristics of Healthcare Management System

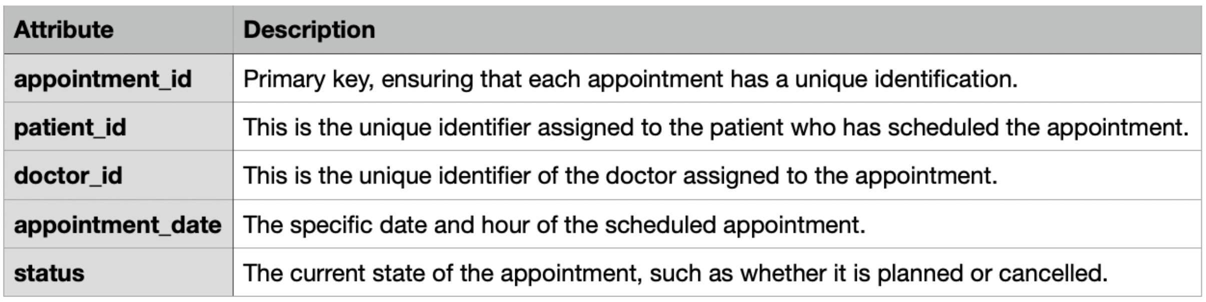
The database schema for a Healthcare Management System comprises elements such as Patient, Appointment, Billing, Inventory, and Doctor. These entities contain data regarding patients, their scheduled visits, financial records, inventory items, and facts about doctors.

Patient Table:



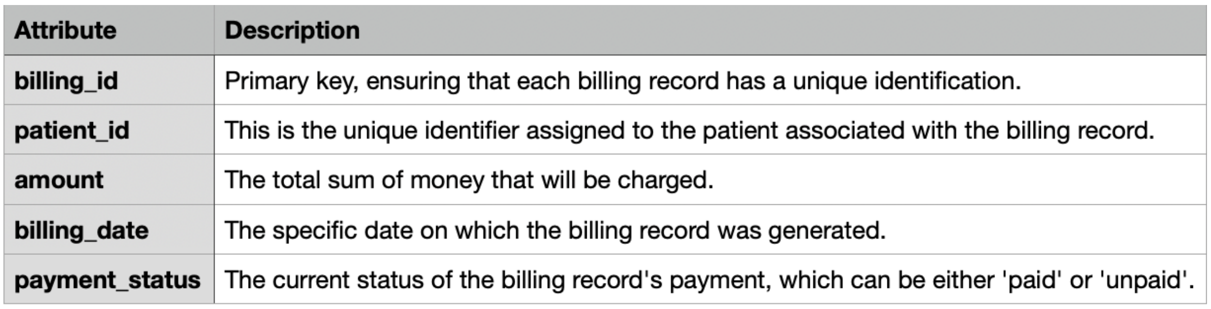
Attribute	Description
patient_id	Primary key, which is a unique identifier for each patient.
first_name	The initial given name of the individual receiving medical treatment.
last_name	The surname of the patient.
DOB	The date on which the patient was born.
gender	The biological sex of the patient.
address	The location where the patient resides.
phone	The numerical sequence used to contact the patient.
insurance_info	Patient's insurance details.

Appointment Table



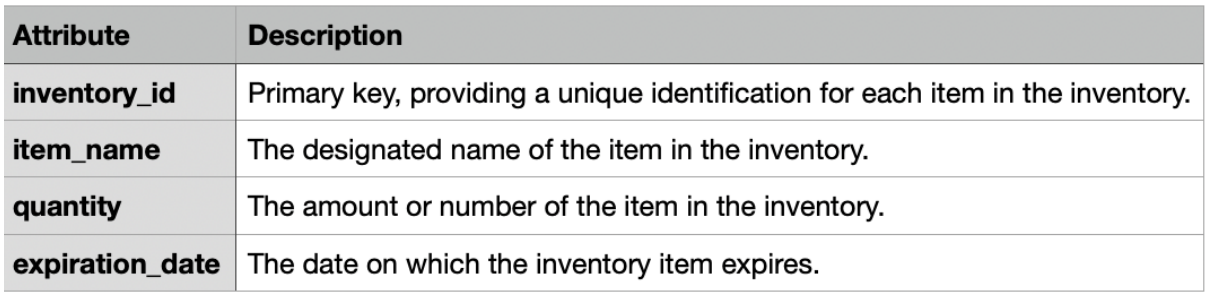
Attribute	Description
appointment_id	Primary key, ensuring that each appointment has a unique identification.
patient_id	This is the unique identifier assigned to the patient who has scheduled the appointment.
doctor_id	This is the unique identifier of the doctor assigned to the appointment.
appointment_date	The specific date and hour of the scheduled appointment.
status	The current state of the appointment, such as whether it is planned or cancelled.

Billing Table:



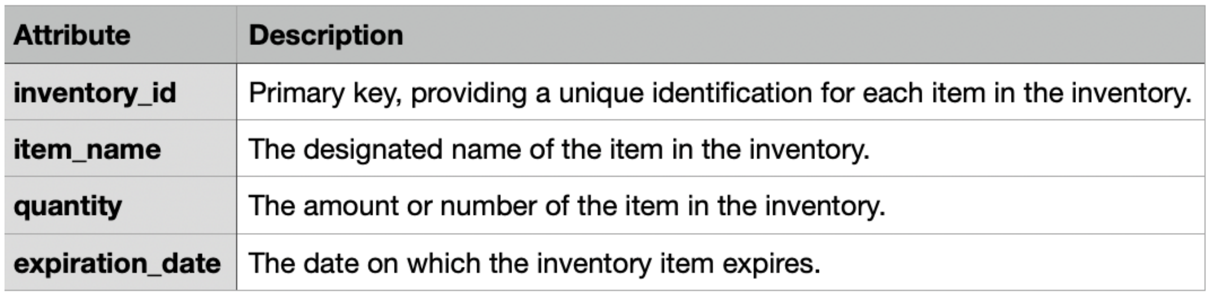
Attribute	Description
billing_id	Primary key, ensuring that each billing record has a unique identification.
patient_id	This is the unique identifier assigned to the patient associated with the billing record.
amount	The total sum of money that will be charged.
billing_date	The specific date on which the billing record was generated.
payment_status	The current status of the billing record's payment, which can be either 'paid' or 'unpaid'.

Inventory Table:



Attribute	Description
inventory_id	Primary key, providing a unique identification for each item in the inventory.
item_name	The designated name of the item in the inventory.
quantity	The amount or number of the item in the inventory.
expiration_date	The date on which the inventory item expires.

Doctor Table:



Attribute	Description
inventory_id	Primary key, providing a unique identification for each item in the inventory.
item_name	The designated name of the item in the inventory.
quantity	The amount or number of the item in the inventory.
expiration_date	The date on which the inventory item expires.

Task 2

The relationships Between These Entities

The connection between the Patient and Appointment tables enables the possibility for each patient to have many appointments. The Patient and Billing tables are related in a way that enables each patient to have several billing entries. The link between the Doctor and Appointment tables enables a doctor to have many appointments.

Patient – Appointment Relationship: The relationship between patients and appointments is a one-to-many relationship, meaning that each patient can have several appointments.
The Appointment table contains a foreign key called patient_id, which references the patient_id column in the Patient table.

The patient-billing relationship is characterized by a one-to-many relationship, meaning that each patient can have several billing records.
The foreign key "patient_id" in the Billing table references the "patient_id" in the Patient table.
The doctor-appointment relationship is a one-to-many relationship, meaning that each doctor can have several appointments.
The Appointment table contains a foreign key called doctor_id, which references the doctor_id column in the Doctor table.

Codes;

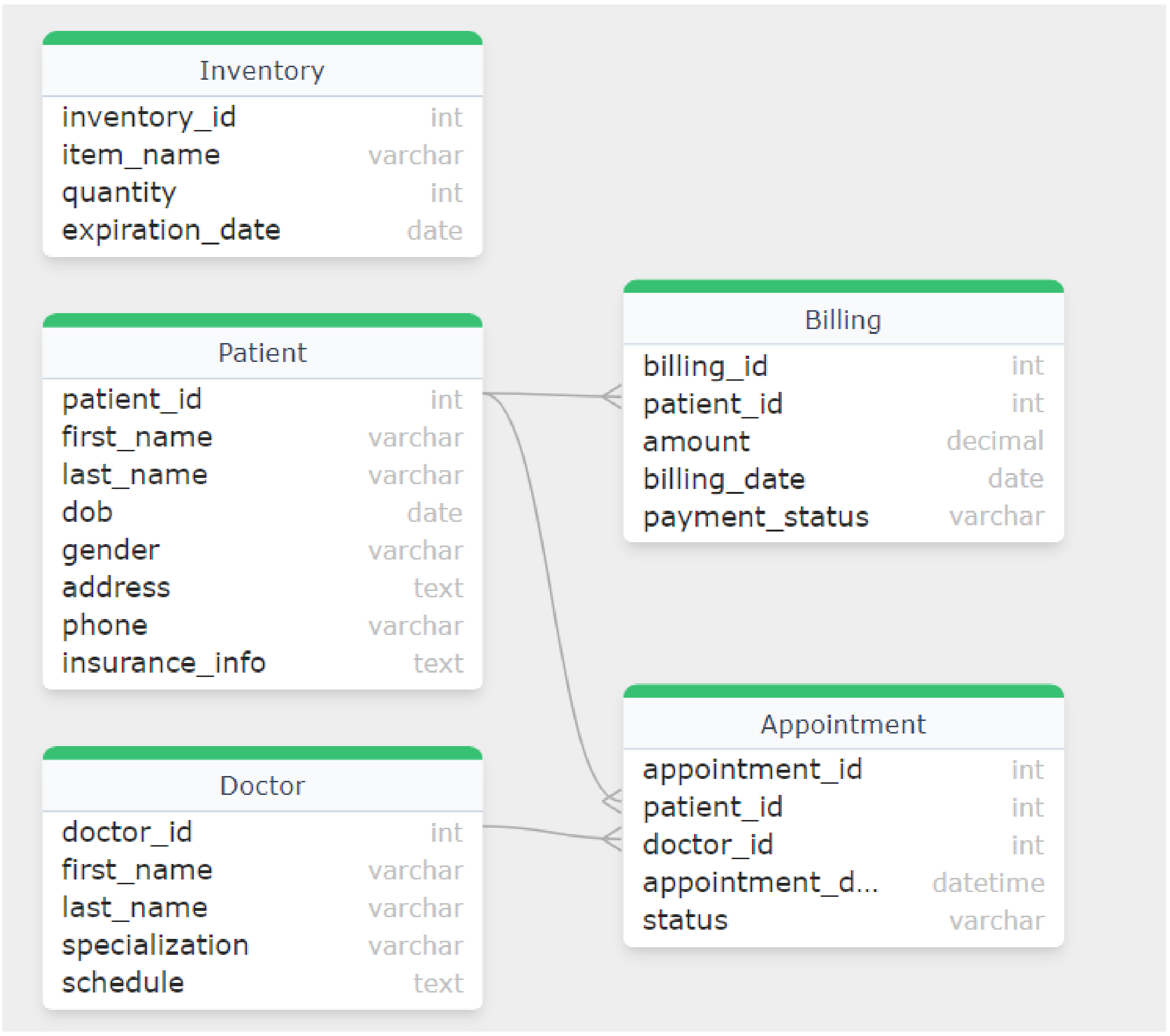
```
CREATE TABLE Patient (
    patient_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(30) NOT NULL,
    last_name VARCHAR(30) NOT NULL,
    dob DATE NOT NULL,
    gender VARCHAR(6) NOT NULL,
    address VARCHAR(100) NOT NULL,
    phone VARCHAR(15) NOT NULL,
    insurance_info VARCHAR(100)
);

CREATE TABLE Doctor (
    doctor_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(30) NOT NULL,
    last_name VARCHAR(30) NOT NULL,
    specialization VARCHAR(100) NOT NULL,
    schedule VARCHAR(24) NOT NULL
);

CREATE TABLE Appointment (
    appointment_id INT AUTO_INCREMENT PRIMARY KEY,
    patient_id INT,
    doctor_id INT,
    appointment_date DATETIME NOT NULL,
    status VARCHAR(10) NOT NULL,
    FOREIGN KEY (patient_id) REFERENCES Patient(patient_id),
    FOREIGN KEY (doctor_id) REFERENCES Doctor(doctor_id)
);

CREATE TABLE Billing (
    billing_id INT AUTO_INCREMENT PRIMARY KEY,
    patient_id INT,
    amount DECIMAL(10, 2) NOT NULL,
    billing_date DATE NOT NULL,
    payment_status VARCHAR(30) NOT NULL,
    FOREIGN KEY (patient_id) REFERENCES Patient(patient_id)
);

CREATE TABLE Inventory (
    inventory_id INT AUTO_INCREMENT PRIMARY KEY,
    item_name VARCHAR(100) NOT NULL,
    quantity INT NOT NULL,
    expiration_date DATE NOT NULL
);
```



Sample Data Codes;

```
patient_sample_data = {
    "patient_id": [1, 2],
    "first_name": ["Mustafa", "Ozan"],
    "last_name": ["Akgül", "Özgür"],
    "dob": ["1998-10-12", "1998-05-12"],
    "gender": ["Male", "Male"],
    "address": ["Alexander strase", "Revaler st."],
    "phone": ["10676", "10245"],
    "insurance_info": ["A-", "B+"]
}

doctor_sample_data = {
    "doctor_id": [1, 2],
    "first_name": ["Sumru", "Mustafa"],
    "last_name": ["Elden", "Akgül"],
    "specialization": ["Aesthetician", "Dermatology"],
    "schedule": ["Mon-Fri 9am-5pm", "Tue-Sat 10am-6pm"]
}

appointment_sample_data = {
    "appointment_id": [1, 2],
    "patient_id": [1, 2],
    "doctor_id": [1, 2],
    "appointment_date": ["2023-07-01 09:00:00", "2023-07-02 09:00:00"],
    "status": ["Scheduled", "Scheduled"]
}

billing_sample_data = {
    "billing_id": [1, 2],
    "patient_id": [1, 2],
    "amount": [150, 120],
    "billing_date": ["2023-07-01", "2023-07-02"],
    "payment_status": ["Paid", "Unpaid"]
}

inventory_sample_data = {
    "inventory_id": [1, 2],
    "item_name": ["Bandages", "Syringes"],
    "quantity": [100, 200],
    "expiration_date": ["2023-12-31", "2024-06-30"]
}

patient_df = pd.DataFrame(patient_sample_data)
doctor_df = pd.DataFrame(doctor_sample_data)
appointment_df = pd.DataFrame(appointment_sample_data)
```

```
billing_df = pd.DataFrame(billing_sample_data)
inventory_df = pd.DataFrame(inventory_sample_data)

import ace_tools as tools; tools.display_dataframe_to_user(name="Patient Table",
dataframe=patient_df)
tools.display_dataframe_to_user(name="Doctor Table", dataframe=doctor_df)
tools.display_dataframe_to_user(name="Appointment Table", dataframe=appointment_df)
tools.display_dataframe_to_user(name="Billing Table", dataframe=billing_df)
tools.display_dataframe_to_user(name="Inventory Table", dataframe=inventory_df)
```

Manipulation of Library System;

```
SELECT * FROM Patient;

SELECT a.appointment_id, a.appointment_date, a.status, d.first_name AS doctor_first_name,
d.last_name AS doctor_last_name
FROM Appointment a
JOIN Doctor d ON a.doctor_id = d.doctor_id
WHERE a.patient_id = 1;

SELECT * FROM Billing
WHERE patient_id = 1;

SELECT first_name, last_name, specialization FROM Doctor;

SELECT * FROM Inventory
WHERE quantity < 100;

SELECT a.appointment_id, p.first_name AS patient_first_name, p.last_name AS
patient_last_name, d.first_name AS doctor_first_name, d.last_name AS doctor_last_name,
a.appointment_date, a.status
```

```
FROM Appointment a
JOIN Patient p ON a.patient_id = p.patient_id
JOIN Doctor d ON a.doctor_id = d.doctor_id
WHERE a.status = 'Scheduled';


SELECT p.first_name, p.last_name, SUM(b.amount) AS total_billing
FROM Billing b
JOIN Patient p ON b.patient_id = p.patient_id
GROUP BY b.patient_id;


SELECT b.billing_id, p.first_name AS patient_first_name, p.last_name AS
patient_last_name, b.amount, b.billing_date, b.payment_status
FROM Billing b
JOIN Patient p ON b.patient_id = p.patient_id;


SELECT * FROM Inventory
WHERE expiration_date <= DATE_ADD(CURDATE(), INTERVAL 6 MONTH);
```

Task 3

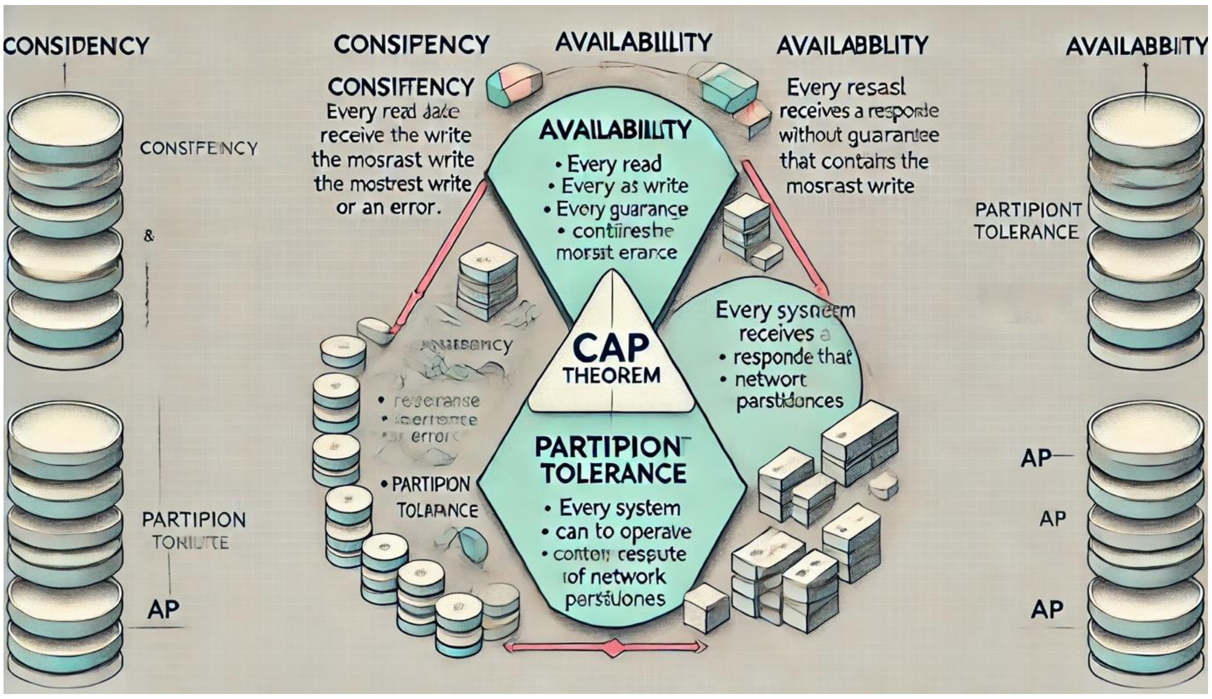
Normalization is a fundamental principle in database design that aims to reduce data redundancy and improve data integrity. By organizing data into separate tables for ‘Patient’ , ‘Doctor’ , ‘Appointment’ ext., aim to reduce data duplication and maintain consistent, accure information. Second principle is using of Primary and Foreign keys. Primary keys are used to uniquely identify and index each row within a single table. In this example, ‘PatientID’ in the ‘Patient’ table, ‘DoctorUD’ in the ‘Doctor’ table and ‘AppointmentID’ in the ‘Appointment’ table. In order to avoid a row from being created or modified in table_a unless the value in its foreign key column already exists in the appropriate column of table_b, foreign keys are used to link rows in two separate tables.For example, ‘PatientID’ and ‘DoctorID’ in the ‘Appointment’ table link each appointment to a specific patient and doctor,respectively. Moreover, ‘PatientID’ and ‘DoctorID’ in the ‘MedicalRecord’ table link medical records to the corresponding patient and doctor. The third principle is scalability and flexibility. As the healthcare organization grows new entities and relationships can be added thanks to the database design's flexibility and scalability. For instance, the current structure may handle changes such as the incorporation of new data types or healthcare providers without requiring a significant redesign.

TASK3

The 'CAP' in the CAP Theorem stands for consistency**,** availability and partition tolerance. In short, the theorem applies a simple logic that a distributed system can only deliver two of these characteristics. When designing a distributed application, it is essential to understand the CAP theorem and pick the data management system that delivers the most suitable two characteristics, depending on your needs, with the trade-off in the third term which should have a lower priority in our application.

The first component of the CAP theorem, consistency, states that in a distributed system, every read operation will either return the most recent write or an error. To put it another way, every node in the system displays the same data value at every moment. In applications like financial transactions or medical records, where data correctness is critical, achieving great consistency is essential.

Secondly, when a client requests data, availability suggests that they will receive a response even if one or more nodes are unavailable. An other way to describe it is this: every functioning node in the distributed system, without fail, responds to requests with a legitimate response. Thirdly, to understand partition tolerance, the semantic of "partition" must be clarified. A **partition** is a communications break within a distributed system (nodes remain up, but the network between some of them does not work). In other words, this is a delay or a drop in messages between nodes communicating over an asynchronous network. Partition tolerance, also called Robustness, means that the cluster continues to function even if there is a partition.



The CAP theorem states that it is difficult to achieve all three attributes simultaneously in a distributed system. Instead, you must prioritize two out of the three options, and the selection of which two options has an important effect on the behavior of the system:

a) CA or Consistency and Availability: By prioritizing both Consistency and Availability, the system ensures strong data consistency and rapid responsiveness, but it sacrifices Partition Tolerance. It functions effectively in consistent network settings, although it may encounter difficulties when network partitions occur.

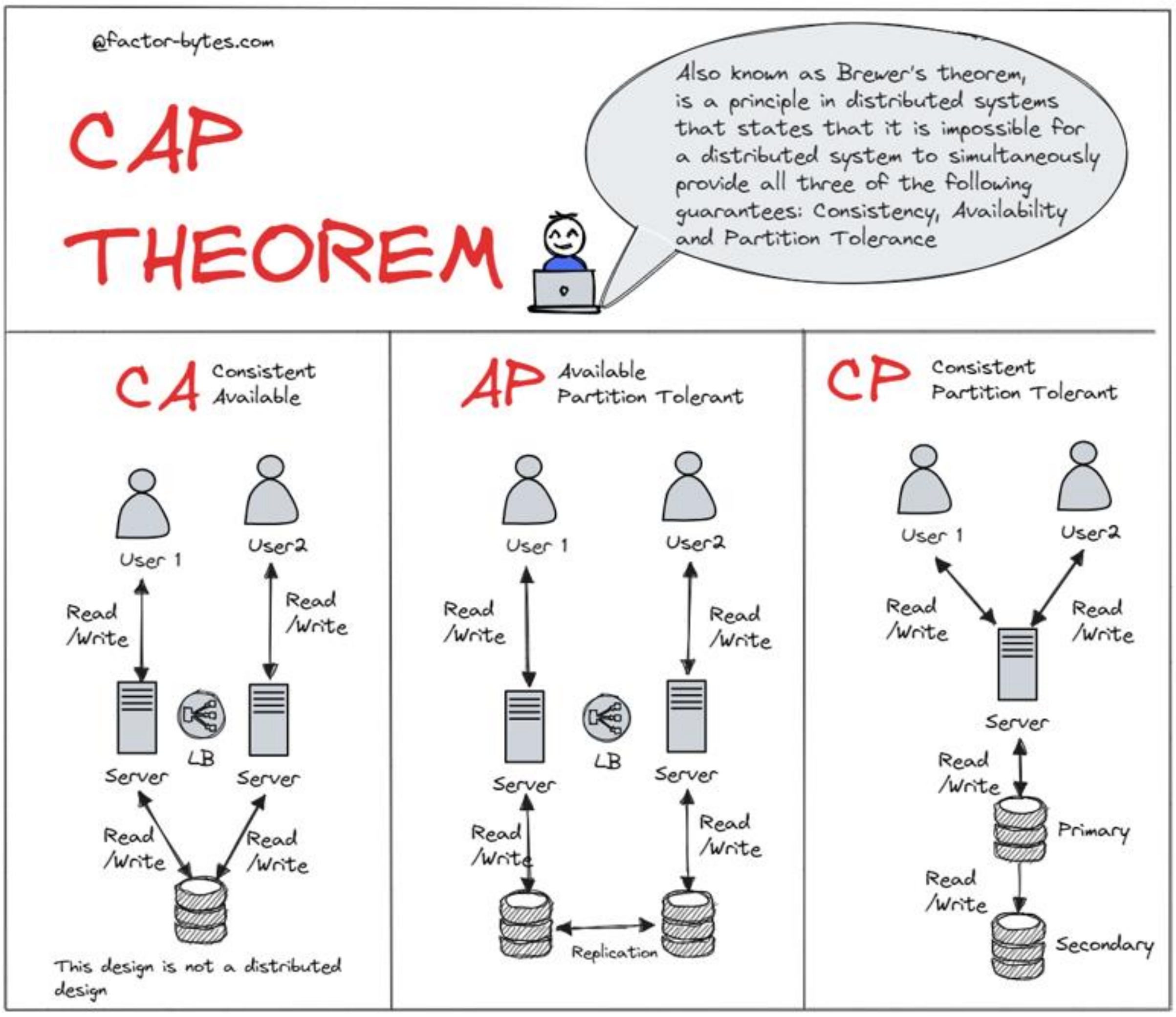
b) CP or Consistency and Partition Tolerance: Prioritizing Consistency and Partition Tolerance guarantees robust data consistency and the capability to endure network partitions, however it may lead to intermittent unavailability during partition occurrences.

c) AP or Availability and Partition Tolerance: The focus of AP is to achieve a high level of system availability and the capability to function even when there are network partitions. However, this trade-off may result in the relaxation of strong consistency, so enabling temporary inconsistencies.

For applications like real-time analytics and recommendation engines, a contract has been established where data is intertwined as long as it pertains to a current situation and cannot be temporarily addressed. This enhances the usefulness of the system.

Uninterrupted service is a crucial necessity for databases due to their handling of substantial workloads. Technologies like Hadoop or Spark often adhere to the AP paradigm, prioritizing availability and partition tolerance.

Take, for instance, any e-commerce platform. Consistency in this context guarantees that the product inventory of the e-commerce application is always up-to-date, preventing the sale of more products than are available and ensuring that orders are processed correctly. Regarding availability, clients have unrestricted access to the website at all times, allowing them to effortlessly browse products and make purchases without any disruptions or problems.


@factor-bytes.com
CAP THEOREM
Also known as Brewer's theorem, is a principle in distributed systems that states that it is impossible for a distributed system to simultaneously provide all three of the following guarantees: Consistency, Availability and Partition Tolerance
CA Consistent Available
User 1
User2
Read /Write
Server
LB
This design is not a distributed design
AP Available Partition Tolerant
Replication
CP Consistent Partition Tolerant
Primary
Secondary

References

1. Elmasri, R. and Navathe, S.B., 2015. *Fundamentals of Database Systems*. 7th ed. Pearson.
2. Connolly, T.M. and Begg, C.E., 2015. *Database Systems: A Practical Approach to Design, Implementation, and Management*. 6th ed. Pearson.
3. Date, C.J., 2003. *An Introduction to Database Systems*. 8th ed. Addison-Wesley.
4. Kroenke, D.M. and Auer, D.J., 2013. *Database Concepts*. 6th ed. Pearson.
5. Rob, P. and Coronel, C., 2007. *Database Systems: Design, Implementation, and Management*. 8th ed. Course Technology.
6. Wager, K.A., Lee, F.W. and Glaser, J.P., 2017. *Health Care Information Systems: A Practical Approach for Health Care Management*. 4th ed. Jossey-Bass.