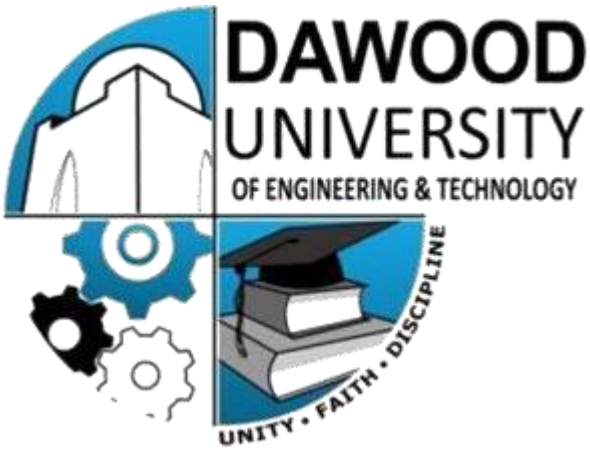


Project Report

Names of the Members:

Syed Huzaifa Nazim (23F – AI – 23)

Muhammad Basit Memon (23F – AI – 07)

Muhammad Unais (23F – AI – 65)

Subject: DBMS

Project: Case Studies from chapter 1 to 8

Book used : Modern Database

< Introduction >

Project Title: Mountain View Community Hospital

Overview

The **Mountain View Community Hospital** is a robust, database-driven solution designed to streamline the management of medical records, patient information, appointments, and doctor-patient interactions. This system addresses the growing need for efficient healthcare administration by providing a centralized platform for data storage, retrieval, and processing.

The project incorporates advanced database design principles and programming logic to ensure accuracy, reliability, and usability. By integrating modern technologies, the system offers functionalities such as patient registration, appointment scheduling, medical record management, and notification handling, all while adhering to stringent data validation rules.

Objectives

The primary objectives of the Mountain View Community Hospital are as follows:

1. **Simplify Healthcare Administration**: Provide a centralized platform for managing patients, doctors, appointments, and medical records.
2. **Enhance Data Accuracy**: Enforce strict validation rules for names, contact numbers, and email formats to ensure the reliability of data.
3. **Facilitate Communication**: Enable the seamless exchange of information between doctors, patients, and administrative staff.
4. **Improve Scalability**: Design a system capable of handling large volumes of data efficiently, catering to future expansion.

Problem Statement

Healthcare facilities often face challenges in managing large volumes of patient and medical data. The reliance on manual or semi-digital processes can lead to inefficiencies, errors, and data mismanagement. This project aims to address these challenges by developing a comprehensive system that ensures data consistency, automates routine tasks, and improves overall operational efficiency.

Scope of the Project

The Mountain View Community Hospital focuses on the following core modules:

1. **Patient Management**: Registration and maintenance of patient details, ensuring data integrity through validation checks.
2. **Doctor Management**: Tracking doctor details, including specialization and contact information.
3. **Appointment Scheduling**: Streamlining the scheduling process with unique constraints to prevent conflicts.
4. **Medical Records**: Storing and managing diagnostic and treatment records.
5. **Notifications**: Generating automated alerts for specific system events, such as updates to patient information.

Key Features

- **Data Validation**: Rules for name formats, contact number length, and email structure to ensure data accuracy.

- **Relational Database Design**: Efficient data organization through structured tables and well-defined relationships.
- **Custom Triggers**: Automated actions to enforce rules and provide user-friendly error messages.
- **User Roles**: Differentiated access for physicians, administrators, and patients.

Methodology

The project employs the following development phases:

1. **Requirement Analysis**: Identifying the needs of healthcare administrators and users.
2. **Database Design**: Structuring the schema with entities, attributes, and relationships.
3. **Implementation**: Writing SQL scripts for table creation, constraints, triggers, and stored procedures.
4. **Testing**: Validating the system with sample data to ensure compliance with functional requirements.

< Table of Contents >

Chapter 1: The Database Environment and Development Process

Case Study: Mountain View Community Hospital

Background:
Mountain View Community Hospital (MVCH) is a growing healthcare institution facing challenges with its current data management. The hospital needs an integrated database system to streamline operations, improve patient care, and support decision- making.

Case Questions:
1. The goal of Mountain View Community Hospital is to provide high-quality, cost-effective health-care services for the surrounding community in a compassionate, caring, and personalized manner. Give some examples of how the use of databases in the hospital might improve health-care quality or contain costs. How else could a well-managed database help the hospital achieve its mission?

Ans: Improving Health-Care Quality and Reducing Costs with Databases

- **Examples of Quality Improvement:**
  - Maintaining an accurate and updated patient record system to reduce medical errors.
  - Tracking inventory of medical/surgical items to ensure availability during emergencies.
  - Scheduling staff and physicians effectively based on patient needs.
- **Cost Containment:**
  - Reducing redundant tests by sharing patient information among departments.
  - Optimizing resource allocation, such as beds and medical equipment.
- **Supporting the Mission:**
  - Providing personalized care by storing and using patient preferences and medical history.

---

2. How can database technology be used to help Mountain View Community Hospital comply with the security standards of the Health Insurance Portability and Accountability Act of 1996 (HIPAA)? HIPAA requires health-care providers to maintain reasonable and appropriate administrative, technical, and physical safeguards to ensure that the integrity, confidentiality, and availability of electronic health information they collect, maintain, use, or transmit is protected. (For more details on HIPAA, visit www.hhs.gov/ocr/privacy.)

Ans: Using Databases for HIPAA Compliance

- **Administrative Safeguards:**
  - Role-based access control to ensure that only authorized personnel can view sensitive information.
- **Technical Safeguards:**
  - Encryption of data to protect electronic health information during transmission and storage.
- **Physical Safeguards:**
  - Maintaining secure server rooms and monitoring database access logs to detect and prevent unauthorized access.

---

3. What are some of the costs and risks of using databases that the hospital must manage carefully?

Ans: Costs and Risks of Using Databases

- **Costs:**
  - Initial investment in database software, hardware, and training.
  - Ongoing costs for maintenance, backups, and updates.
- **Risks:**
  - Data breaches that could compromise patient privacy.
  - System downtime affecting hospital operations and patient care.
  - Risk of data corruption or loss without proper backup procedures.

---

4. How critical are data quality requirements in the hospital environment? For which applications might quality requirements be more restrictive?

Ans: Criticality of Data Quality Requirements

- **High Importance of Quality:**
  - Accurate patient records are crucial for diagnosis and treatment.

- Incorrect or incomplete data could lead to medical errors or mismanagement of resources.
- **Restrictive Requirements:**
  - **Applications:** Diagnostic imaging, medication orders, and surgical records must meet stringent accuracy standards to ensure safety and compliance.

---

5. At present, Mountain View Community Hospital is using relational database technology. Although this technology isappropriate for structured data, such as patient or accounting data, it is less well-suited to unstructured data, such as graphical data and images. Can you think of some types of data maintained by a hospital that fit this latter category? What types of database technology rather than relational might be better suited to these data types?

Ans: Unstructured Data in Hospitals and Appropriate Database Technologies

- **Examples of Unstructured Data:**
  - X-ray and MRI images.
  - Notes from physicians or nurses.
  - Video data from surgical procedures.
- **Suitable Technologies:**
  - NoSQL databases for flexible and scalable storage of unstructured data.
  - Object-oriented databases for storing and managing multimedia content.

---

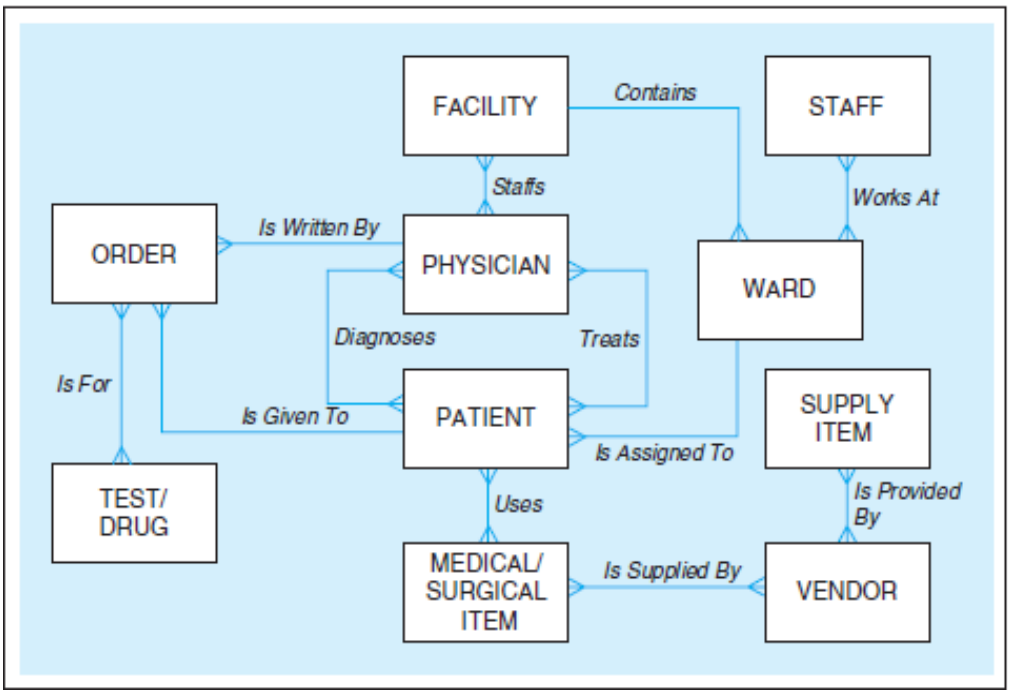
6. How could the hospital use Web-based applications? What are some of the benefits and risks associated with Webbased applications for the hospital?

Ans: Web-Based Applications in the Hospital

- **Use Cases:**
  - Patient portals for accessing test results, billing, and scheduling.
  - Telemedicine platforms for virtual consultations.
- **Benefits:**
  - Increased convenience for patients.
  - Faster access to medical records for physicians.
- **Risks:**
  - Potential cybersecurity vulnerabilities.
  - Risk of data exposure during transmission.

---

7. The case description lists 10 business rules. The study team used these rules to develop MVCH Figure 1-3. Are there any other business rules implied by or depicted in that figure? What are they?



MVCH FIGURE 1-3 **Preliminary enterprise data model**

Ans: Additional Business Rules Implied by MVCH Figure 1-3

- Physicians can diagnose multiple patients, but each diagnosis is tied to a single physician.
- Medical/surgical items used by patients must be supplied by a vendor.
- Supply items may be assigned to wards for general use.
- Staff members may work at multiple facilities, as implied by the "Works At" relationship.

Project Assignments:

- **P1. The study team's activities described in this case study are still in the very early stages of information system and database development. Outline the next steps that should be followed within the Information Systems unit to align current systems and databases to the future information systems needs of the hospital.**

Ans: Next Steps for Aligning Current Systems and Databases

1. **Needs Assessment and Requirements Gathering:**
- Conduct a thorough analysis of current processes, data requirements, and pain points.
- Engage stakeholders (physicians, nurses, administrators, and IT staff) to identify system gaps and improvements needed.
2. **Define System Objectives:**
- Align with goals such as cost reduction, improved patient care, HIPAA compliance, and operational efficiency.
3. **Data Modeling and Standardization:**
- Expand the enterprise data model to reflect current and future needs (see P3(a)).
- Ensure data consistency and interoperability by adhering to healthcare data standards (e.g., HL7, FHIR).
4. **Technology Assessment:**
- Evaluate current systems for scalability and integration potential.
- Investigate new technologies (e.g., cloud databases, NoSQL systems for unstructured data).
5. **Database and System Design:**
- Develop detailed database schema and application blueprints.
- Define rules for data storage, access, and security.
6. **Implementation and Integration:**
- Migrate current data to the new database system.
- Develop or procure software applications tailored to hospital workflows.
7. **Training and Support:**
- Train hospital staff on the new systems and processes.
- Set up ongoing technical support and maintenance.
8. **Evaluation and Continuous Improvement:**
- Monitor system performance, user satisfaction, and outcomes.
- Adjust systems to meet evolving hospital needs and technological advancements.
9. **Needs Assessment and Requirements Gathering:**
- Conduct a thorough analysis of current processes, data requirements, and pain points.
- Engage stakeholders (physicians, nurses, administrators, and IT staff) to identify system gaps and improvements needed.
10. **Define System Objectives:**
- Align with goals such as cost reduction, improved patient care, HIPAA compliance, and operational efficiency.
11. **Data Modeling and Standardization:**
- Expand the enterprise data model to reflect current and future needs (see P3(a)).
- Ensure data consistency and interoperability by adhering to healthcare data standards (e.g., HL7, FHIR).
12. **Technology Assessment:**
- Evaluate current systems for scalability and integration potential.
- Investigate new technologies (e.g., cloud databases, NoSQL systems for unstructured data).
13. **Database and System Design:**
- Develop detailed database schema and application blueprints.
- Define rules for data storage, access, and security.
14. **Implementation and Integration:**
- Migrate current data to the new database system.
- Develop or procure software applications tailored to hospital workflows.
15. **Training and Support:**
- Train hospital staff on the new systems and processes.

- Set up ongoing technical support and maintenance.

16. **Evaluation and Continuous Improvement:**

- Monitor system performance, user satisfaction, and outcomes.
- Adjust systems to meet evolving hospital needs and technological advancements.

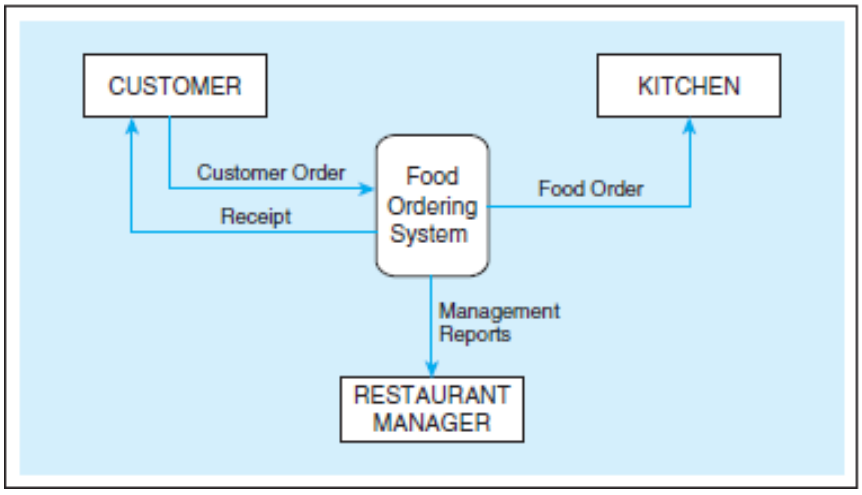
- **P2. The patient bill is an example of a view that would be of interest in a hospital environment. Identify and list other user views that could occur in a hospital environment.**

Ans: Other User Views in a Hospital Environment

- **Patient Admission and Discharge View:** Tracks patient admissions, transfers, and discharges.
- **Medical History View:** Displays patients' past and current medical records.
- **Inventory View:** Monitors medical/surgical supplies, drugs, and equipment.
- **Staff Scheduling View:** Tracks work schedules for physicians, nurses, and other staff.
- **Financial Reports View:** Summarizes revenue, expenses, and outstanding payments.
- **Compliance View:** Tracks activities related to regulatory compliance and audits.
- **Emergency Department View:** Monitors emergency cases, triage details, and resource availability.
- **Outpatient Services View:** Displays appointments, procedures, and lab results.

Volunteer Management View: **Tracks volunteer activities, schedules, and assignments.**

- **P3. Carefully read through the case description, exercises, and questions again and:**
  a. Modify the enterprise data model shown in MVCH Figure 1-3 to include any additional entities and relationships that you identify.
  b. Modify the list of business rules from the case description to include the additional entities and relationships you identified.
  c. Draw a context diagram of MVCH's improved information system similar to the one for a burger restaurant shown in MVCH Figure 1-6. A context diagram provides the highest-level view of a system and shows the system boundaries, external entities that interact with the system, and major information flows between the entities and the system.



MVCH FIGURE 1-6 Context diagram example

Source: Adapted from Hoffer et al. (2010).

Ans P3(a). Modifying the Enterprise Data Model (MVCH Figure 1-3)
Additional Entities:

- **Volunteer:** Tracks activities and roles of volunteers in the hospital.
- **Department:** Represents hospital departments like Cardiology, Neurology, and Pediatrics.
- **Appointment:** Links patients with physicians for outpatient consultations.
- **Equipment:** Tracks the use and maintenance of hospital equipment.

Additional Relationships:

- **Patient-Appointment:** A patient has one or more appointments with physicians.
- **Department-Staff:** Staff are assigned to specific departments.
- **Volunteer-Department:** Volunteers contribute to specific departments or roles.

- **Equipment-Ward:** Equipment is assigned to wards or departments for use.

Ans P3(b). Modifying Business Rules

- Each **volunteer** can contribute to multiple departments, and each department may have several volunteers.
- **Appointments** must be scheduled between patients and available physicians.
- Each **department** manages specific staff, equipment, and patients.
- Medical equipment used in treatments is **assigned** to specific departments or wards.

Ans P3(c). Context Diagram of MVCH's Improved Information System

To replicate the style of MVCH Figure 1-6 (food ordering system diagram), the **context diagram** for MVCH's improved system will have the following components:

- **External Entities:**
  - **Patients:** Provide personal details, make appointments, and receive bills.
  - **Physicians/Nurses:** Input diagnoses, treatments, and access patient records.
  - **Volunteers:** Assist with tasks and interact with the system for schedules and reports.
  - **Suppliers:** Provide medical supplies and drugs.
  - **Regulatory Bodies:** Access reports for compliance and auditing.
- **System:** MVCH Information System
- **Major Information Flows:**
  - Patients → Appointment Requests → System
  - System → Bills → Patients
  - Physicians → Diagnoses/Treatments → System
  - System → Reports → Physicians/Nurses
  - Suppliers → Supply Data → System
  - System → Compliance Reports → Regulatory Bodies

Chapter 2: Modeling Data in the Organization

Case Study: Mountain View Community Hospital

Background:

Mountain View Community Hospital is expanding its operations and requires a robust data model to ensure effective data organization and accessibility. The aim is to develop an Entity-Relationship (E-R) model that accurately represents the hospital's businessrulesand operations.

Case Questions:

1. Why would Mountain View Community Hospital want to use E-R modeling to understand its data requirements? What other ways might the hospital want to model its information requirements?

Ans: Reasons for Using E-R Modeling:

- **Clear Visualization:** E-R diagrams provide a clear, graphical representation of entities, attributes, and relationships, helping stakeholders understand the data structure.
- **Improved Communication:** It bridges the gap between technical and non-technical stakeholders by visually conveying how data is organized.
- **Accurate Data Design:** Helps in identifying the key entities, attributes, and relationships, ensuring the data model aligns with business processes.
- **Support for Database Development:** Acts as a blueprint for creating the hospital's database, ensuring efficiency and consistency.
- **Compliance and Auditing:** Ensures the data model adheres to regulatory standards (e.g., HIPAA), making audits and compliance checks easier.

Other Ways to Model Information Requirements:

- **Data Flow Diagrams (DFDs):** To model the flow of information within the hospital's systems.
- **Use Case Diagrams:** To illustrate the interactions between hospital staff, patients, and the system.
- **Business Process Modeling Notation (BPMN):** To represent hospital workflows and processes.
- **UML Class Diagrams:** For a more detailed representation of entities and their relationships.
- **Relational Schema Design:** After E-R modeling, a relational schema could be used to define the database tables and relationships.

---

2. Is Mountain View Community Hospital itself an entity type in the data model? Why or why not?
Ans: No, Mountain View Community Hospital itself is **not** an entity type in the data model.

Reason:

- The hospital is the organization or domain for which the data model is being created. It serves as the context within which other entities (like Patients, Physicians, Beds, etc.) exist.
- Entities in the E-R model represent objects or concepts (e.g., Patients, Items, Treatments) that the hospital must track, not the hospital itself.

---

3. Do there appear to be any of the following in the description of the Mountain View Community Hospital data requirements? If so, what are they?
a. weak entities
b. multivalued attributes
c. multiple relationships
Ans: a. Weak Entities:
Yes, **Bed** appears to be a weak entity.

- A bed's existence is dependent on the **Care Center** (via the Care Center ID). A bed cannot exist independently outside the care center context.

b. Multivalued Attributes:
No explicit multivalued attributes are mentioned. However, relationships such as an Employee working in multiple Care Centers might give the appearance of multivalued data, but these are better modeled as relationships rather than attributes.

c. Multiple Relationships:
Yes, there are multiple relationships:

- **Physician-Patient:** Includes relationships for referring and admitting patients.
- **Physician-Diagnosis:** Diagnoses conditions affecting patients.
- **Physician-Treatment:** Orders and performs treatments for patients.
- **Patient-Bed:** Some patients are assigned to beds, while others are not.
- **Employee-Care Center:** Employees are assigned to one or more care centers.

---

4. When developing an E-R diagram for Mountain View Community Hospital, what is the significance of the business rule that states that some patients are assigned to a bed, but outpatients are not assigned to a bed?
Ans: Significance:

- **Distinction Between Patient Types:** It highlights the distinction between inpatients (assigned to a bed) and outpatients (not assigned to a bed).
- **Optional Relationship:** The relationship between **Patient** and **Bed** is optional, reflecting real-world scenarios where not all patients require admission.
- **Data Modeling Impact:** Ensures that the database design accommodates both inpatients and outpatients without forcing unnecessary data entry for beds when dealing with outpatients.

---

5. Do you think that *Items* should be split into two separate entities, one for nonreusable and one for reusable items? Why or why not?
Ans: Yes, splitting Items into **nonreusable** and **reusable** entities is advisable.
Reasons:

- **Different Attributes:** Nonreusable items (e.g., syringes) might only need attributes like quantity and unit cost, whereas reusable items (e.g., surgical instruments) require attributes like maintenance schedule, usage history, and condition.
- **Data Integrity:** Separating these entities ensures that their unique characteristics are captured accurately, avoiding null values or irrelevant attributes.
- **Easier Queries:** Users can query and manage reusable and nonreusable items more efficiently without filtering through irrelevant data.
- **Scalability:** If the hospital expands its inventory system, having separate entities will simplify future enhancements.

---

6. What quality check(s) would you perform to determine whether the E-R model you developed can easily satisfy user requests for data and/or information?
Ans: Quality Checks:

1. **Entity Completeness:**
   - Ensure all critical entities (e.g., Patients, Beds, Items) are included and adequately described.

2. **Relationship Validation:**
   - Verify that all relationships between entities are defined correctly, including cardinality (e.g., one-to-one, one-to-many).
3. **Attribute Accuracy:**
   - Ensure each entity has appropriate attributes and that they are atomic (e.g., splitting full names into first and last names if needed).
4. **User Scenarios Testing:**
   - Simulate real-world user queries (e.g., finding all treatments performed by a physician on a specific date) to confirm the model supports them.
5. **Normalization:**
   - Check the data model for normalization (e.g., ensuring no redundant data) to optimize data integrity and storage.
6. **Compliance Check:**
   - Ensure the model adheres to industry and regulatory standards (e.g., HIPAA).
7. **Stakeholder Review:**
   - Present the model to hospital staff and administrators to confirm it aligns with their operational requirements.
8. **Data Flow Testing:**
   - Verify that the flow of data (e.g., from Patient to Treatment to Diagnosis) supports end-to-end hospital processes.

Chapter 3: The Enhanced E-R Model

Case Study: Mountain View Community Hospital

Background:
To handle complex data and relationships, Mountain View Community Hospital (MVCH) needs to enhance its data model by implementing advanced features like supertype/subtype relationships and specialization. These featureswill help the hospital organize data efficiently for better management and reporting.

Case Questions:

1. Is the ability to model supertype/subtype relationships important in a hospital environment such as MVCH? Why or why not?

Ans: Yes, the ability to model **supertype/subtype relationships** is essential in a hospital environment like MVCH due to the diverse categories of entities with shared and unique attributes. For example:

- **Persons**: Employees, patients, physicians, and volunteers all share common attributes such as Name, Address, and Phone. However, each has unique attributes (e.g., hire date for employees, DEA number for physicians, and emergency contact information for patients). Modeling these as subtypes of a "Person" supertype ensures consistency and avoids redundancy.
- **Employees**: Nurses, technicians, and staff share some common attributes but also have specific attributes such as nursing licenses for nurses and job classifications for staff. Supertype/subtype modeling simplifies the representation of such data while maintaining flexibility and clarity in managing shared and unique data for different subtypes.

---

2. Are there any weak entities, multivalued attributes, or multiple relationships in the description of the data requirements in this case segment? If so, what are they?

- **Weak Entities**:
  - The **Bed** entity can be considered a weak entity because its identifier (Bed ID) depends on the combination of Room# and Bed#. Additionally, its existence depends on its relationship with a **Care Center** and the assignment of a resident patient.
- **Multivalued Attributes**:
  - **Nurses** and **technicians** may have certifications in multiple fields (e.g., pediatrics, anesthesia, or specific equipment skills). These certifications are multivalued attributes.
  - **Patients** have emergency contacts and insurance details, which include multiple pieces of related data (e.g., contact name, relationship, address, and phone). These could be treated as multivalued attributes or modeled as separate entities.
- **Multiple Relationships**:

- There are multiple relationships involving **Persons** (e.g., employees may supervise volunteers, nurses may be assigned to care centers, and physicians may refer or admit patients).
- A **Patient** may be linked to multiple **Visits**, and a visit involves relationships with **Physicians**, **Treatments**, and **Care Centers**.

---

3. Can you think of any other business rules (other than the one explicitly described in the case) that are likely to be used in a hospital environment? Can these be represented on an EER diagram for MVCH?

- **Business Rules**:
  - Each patient must have an insurance policy, or the payment must be guaranteed by another responsible party.
  - Volunteers cannot work more than a certain number of hours per week (e.g., 20 hours).
  - Physicians can only admit patients to care centers where they have privileges.
  - Nurses with specific certifications can only work in care centers requiring those certifications.
  - A resident patient must have a physician responsible for their care during their stay.
  - Each bed assignment must record the date and time of assignment and discharge.
- **Representation in EER Diagram**:

These rules can be represented using constraints, relationships, and attributes in the EER diagram. For example:

- Cardinality constraints can enforce relationships like "each patient must have exactly one responsible physician."
- Attributes like "maximum hours worked" for volunteers can be added.
- Specialization hierarchies for nurses with specific certifications can enforce eligibility rules for care centers.

---

4. Are there any universal data models that can be reused as a starting point for modeling MVCH's data requirements? Would you recommend using such a model for the MVCH project? Why or why not?

- **Universal Data Models**:
  - Yes, there are universal data models available for healthcare organizations, such as those provided by industry frameworks like HL7 (Health Level Seven) or general healthcare data models for electronic health record (EHR) systems.
- **Recommendation**:
  - Using a universal data model as a starting point can be beneficial because:
    - It provides a standardized foundation with established relationships and attributes, saving time and effort.
    - It ensures compliance with healthcare industry standards and regulations (e.g., HIPAA).
    - It accommodates common use cases like patient management, physician-patient relationships, and billing systems.
  - However, customization would still be necessary to address MVCH's specific requirements (e.g., unique rules for volunteers and detailed technician skills). The balance between adopting a universal model and tailoring it to MVCH's needs should guide the decision.

Chapter 4: Logical Database Design and the Relational Model

Case Study: Mountain View Community Hospital

Case Questions:

1. Should MVCH continue to use relational technology for its systems development? Why or why not?

Yes, MVCH should continue to use relational technology for the following reasons:

- **Proven Reliability**: Relational databases are mature, well-understood, and widely used in industries like healthcare, offering stability and reliability.
- **Data Integrity**: Relational technology provides strong mechanisms for enforcing data integrity through constraints like primary keys and foreign keys, critical in a hospital setting where data accuracy is vital.
- **Query Language**: SQL, the standard query language for relational databases, is powerful, flexible, and widely supported.
- **Scalability**: Relational databases can scale to handle large volumes of data typical in healthcare, especially with proper indexing and optimization.
- **Integration**: Many existing hospital systems, such as EHRs and billing systems, integrate seamlessly with relational databases. However, MVCH should also explore newer technologies (e.g., object-oriented databases or NoSQL) for specific use cases like unstructured data storage or high-performance applications.

2. Should MVCH use normalization in designing its relational databases? Why or why not?
Yes, normalization should be used because:

- **Data Integrity**: Normalization reduces data redundancy and ensures consistency, preventing anomalies in insertion, update, and deletion.
- **Efficient Storage**: By eliminating duplicate data, normalization optimizes storage requirements.
- **Maintainability**: A well-normalized database is easier to maintain and adapt to changes. However, normalization must be balanced with performance needs. Over-normalization (e.g., reaching 4NF or 5NF) might complicate queries and slow down performance. A practical approach would be normalizing to 3NF and then denormalizing selectively for performance-critical areas.

3. Why are entity integrity constraints of importance to the hospital? Based on the case description from previous chapters, which attributes have you encountered that may be null?

- **Importance of Entity Integrity**:
  - Entity integrity ensures that every row in a table has a unique and non-null primary key, essential for uniquely identifying records (e.g., patients, employees, physicians).
  - In a hospital setting, this is critical to avoid misidentification, which could lead to incorrect treatments or billing errors.
- **Attributes That May Be Null**:
  - **Volunteers**: Skills and interests may be null if not provided.
  - **Physicians**: Pager numbers might not be applicable to all physicians.
  - **Patients**: Insurance information (e.g., policy number) could be null for uninsured patients.
  - **Beds**: Bed assignments may be null for care centers without assigned beds.

4. Why are referential integrity constraints of importance to the hospital?
Referential integrity constraints ensure that foreign key values in one table match primary key values in another, which is essential for:

- **Preventing Orphan Records**: For example, ensuring that each patient's assigned physician exists in the physician table.
- **Maintaining Relationships**: Referential integrity enforces relationships between entities, such as visits tied to patients or beds assigned to care centers.
- **Data Accuracy**: It helps avoid errors like referencing non-existent records, which is crucial in a hospital where incorrect data can lead to serious consequences.

5. Which attribute would you suggest using as the primary key for a PHYSICIAN relation? Why? What specific concerns are related to those attributes that you do not recommend be used?

- **Recommended Attribute**: PhysicianID (hospital-assigned identifier)
  - **Reasons**:
    - Unique and controlled by the hospital, avoiding external dependencies.
    - Immutable, unlike other identifiers that may change (e.g., DEA numbers or license numbers).
    - Simplifies internal record-keeping and avoids conflicts with external systems.
- **Concerns with Other Attributes**:
    - **Social Security Number**: Sensitive personal information that raises privacy and security concerns.
    - **License Number/DEA Registration Number**: Not unique across states or medical organizations and subject to changes if the physician moves or renews their license.
    - **PhysicianID** eliminates dependency on external entities and ensures uniqueness within the hospital.

6. Why is an enterprise key important in a hospital setting such as MVCH?
An **enterprise key** is important because it ensures uniqueness across the entire database, which is vital for:

- **Avoiding Duplication**: In a hospital setting, entities like patients, employees, and physicians may exist across multiple systems. An enterprise key guarantees uniqueness across these systems.
- **Interoperability**: Hospitals often integrate with external systems (e.g., insurance providers, government health programs). An enterprise key facilitates consistent record identification.
- **Future Scalability**: If MVCH expands to multiple locations or systems, an enterprise key ensures data consistency and uniqueness across all branches.

7. Why might you need to revisit and potentially modify the EER model during the logical design phase?
During the logical design phase, revisiting and modifying the EER model may be necessary because:

- **Normalization Requirements**: Relationships and attributes in the EER model may need adjustment to meet normalization standards.
- **Attribute Refinement**: Some attributes may need to be split (e.g., multivalued attributes like certifications for nurses).
- **Constraints**: Logical design requires defining primary keys, foreign keys, and constraints, which might highlight gaps in the EER model.
- **Performance Optimization**: Certain relationships or entities might be denormalized or simplified for better performance.
- **New Requirements**: Stakeholders may identify additional requirements or clarify ambiguities during the logical design phase, necessitating changes in the EER model.

Chapter 5: Physical Database Design and Performance

Case Study: Mountain View Community Hospital

Case Questions:

1. What additional kinds of information do you need for the physical database design of the MVCH database besides the 3NF relations you developed earlier for this case in Chapter 4?

For the physical database design, the following additional information is required:

- **Data Types**: Determine the appropriate data types for each attribute based on the DBMS (e.g., VARCHAR, INT, DATETIME).
- **Indexes**: Decide which attributes require indexing to optimize query performance.
- **Storage Requirements**: Understand data volume, growth rate, and storage constraints.
- **Constraints**: Specify primary keys, foreign keys, unique constraints, and default values.
- **Access Patterns**: Understand how users and applications will access and update the database to optimize performance.
- **Partitioning**: Identify whether horizontal or vertical partitioning is needed based on the size and access patterns.
- **Backup and Recovery**: Plan for data backup, recovery, and fault tolerance.
- **Security**: Define security measures, including encryption, user roles, and access control.

---

2. What different types or forms of clinical data are collected at a hospital such as MVCH? Can you identify data that may not be easily accommodated by the standard data types provided by a DBMS? How would you handle that?

- **Types of Clinical Data**:
  - Patient records: Demographics, medical history, insurance details.
  - Physician information: Licenses, certifications.
  - Treatments: Orders, results, and progress notes.
  - Laboratory tests: Test names, results, units.
  - Imaging: X-rays, MRIs, and other diagnostic images.
  - Billing: Insurance claims, invoices.
  - Scheduling: Appointments, room allocations.
- **Data Not Easily Accommodated**:
  - Medical imaging (e.g., X-rays, MRIs): These are large binary files (BLOBs).
  - Clinical notes: Textual data, possibly unstructured or semi-structured.
  - Sensor data: Continuous data streams from devices like heart monitors.
- **Handling Such Data**:
  - Use BLOBs or FILESTREAM for binary data storage.
  - Store unstructured or semi-structured data in JSON or XML columns supported by modern DBMSs.
  - For sensor data, consider integration with specialized systems or time-series databases.

---

3. Are there opportunities for horizontal or vertical partitioning of this database? If you are not sure, what other information would you need to answer this question with greater certainty?

- **Opportunities**:
  - **Horizontal Partitioning**: Based on time (e.g., patient records by year) or location (e.g., data for different hospital departments).

    - **Vertical Partitioning**: Splitting tables into frequently and infrequently accessed attributes (e.g., patient contact details vs. medical history).
- **Additional Information Needed**:
    - Volume and distribution of data.
    - Query patterns and access frequency.
    - Storage constraints and system load requirements.
    - Growth trends (e.g., how quickly new data accumulates).

---

4. Do you see an opportunity for using a join index for this database? Why or why not?
Yes, there is an opportunity to use a **join index** for optimizing frequently joined tables:

- For example, tables like PATIENT and TREATMENT or PHYSICIAN and TREATMENT could benefit from join indexes as they are likely to be queried together frequently.
- **Why Use Join Index**:
    - Reduces the cost of join operations by precomputing and storing results.
    - Improves query performance for common join scenarios in a hospital setting, such as finding treatments performed by specific physicians or treatments for specific patients.

---

5. Consider the following query against the MVCH database:
For each treatment ordered in the past two weeks, list by treatment ID and date (in reverse chronological order) the number of times a physician performed that treatment that day, sorted alphabetically by physician name.

a. Which secondary key indexes would you suggest to optimize the performance of this query? Why?

- **Recommended Indexes**:
    - Index on TreatmentDate (to filter treatments in the past two weeks efficiently).
    - Composite index on (PhysicianName, TreatmentID, TreatmentDate) to optimize sorting by physician name and grouping by treatment ID and date.
    - Index on PhysicianID in the TREATMENT table to speed up grouping by physicians.

b. Write the SQL statements to create these secondary key indexes.

```
-- Index on TreatmentDate
CREATE INDEX idx_treatment_date
ON TREATMENT (TreatmentDate);

-- Composite index on PhysicianName, TreatmentID, and TreatmentDate
CREATE INDEX idx_physician_treatment
ON TREATMENT (PhysicianName, TreatmentID, TreatmentDate);

-- Index on PhysicianID
CREATE INDEX idx_physician_id
ON TREATMENT (PhysicianID);
```

---

6. The 2002 Sarbanes-Oxley Act (SOX):

a. Can MVCH benefit from voluntarily complying with SOX?
Yes, MVCH can benefit from SOX compliance in the following ways:

- **Enhanced Data Accuracy**: Improves confidence in financial and operational data, reducing errors in patient billing and reporting.
- **Improved Audit Trails**: Strengthens accountability by ensuring all transactions and changes are logged.
- **Regulatory Readiness**: Positions MVCH to meet future regulatory requirements with minimal disruption.

b. How can proper physical database design help with compliance?

- **Improving Accuracy and Completeness of MVCH Data**:
    - Use constraints (e.g., primary keys, foreign keys) to ensure data integrity.
    - Implement checks for required fields to prevent incomplete records.
- **Eliminating Duplicates and Data Inconsistencies**:
    - Normalize data during logical and physical design phases.
    - Use unique constraints and indexes to avoid duplicate entries.
- **Improving Understandability of MVCH Data**:
    - Use clear, meaningful table and column names.
    - Maintain consistent documentation and metadata.
    - Create views and reports that aggregate data for easier interpretation.

Case Exercises:

1. In Case Exercise 2 in Chapter 4, you wrote CREATE TABLE commands for each relation of Dr. Z's small database, which was to be created in Microsoft Access. Since then, Dr. Z has decided to use Microsoft SQL Server, consistent with other databases at MVCH. Reconsider your previous CREATE TABLE commands in answering the following questions:

a. Would you choose different data types for any fields? Why?

Yes transitioning to Microsoft SQL Server requires selecting SQL Server supported data types:

- **Dates:** Use DATE or DATETIME2 instead of Access's DATE/TIME.
- **TEXT fields:** Use VARCHAR(n) for variablelength text and NVARCHAR(n) if Unicode support is needed.
  - **Large text:** Use TEXT or NVARCHAR(MAX) instead of Access's MEMO.
  - **Boolean:** Use BIT instead of Access's YES/NO.

b. Are any fields candidates for coding? What coding scheme?

Coding Fields:

- **Patient ID:** Use a numeric or alphanumeric scheme (e.g., PAT001).
- **Physician ID:** Use a coding scheme like PHY1001.
- **Treatment Type:** Use standard ICD9 or ICD10 codes for medical treatments and procedures.

c. Which fields require data values? Can any fields take nulls?

Fields that require values:

- Primary keys (e.g., PatientID, PhysicianID).
- Foreign keys to ensure relationships.

Fields that may take nulls:

- Optional fields like SocialWorkerID or ReasonForVisit.

d. Handling Missing Data

Procedures:

- Use default values where appropriate (e.g., "Unknown" for ReasonForVisit).
- For fields like SocialWorkerID, allow nulls and use left outer joins in queries to accommodate missing data.

2.

a. Opportunities for Userdefined Data Types?

Yes:

- **Phone Numbers:** Define as a CHAR(10) or VARCHAR(15).
- **ICD10 Codes:** Define as a CHAR(7) or VARCHAR(7) for uniformity.

b. Are any fields candidates for coding?

Same as above:

- Use standardized coding for patients, physicians, and treatments.

c. Fields that May Take Nulls?

Fields like SocialWorkerID or optional notes fields (VisitNotes) may take nulls.

d. Opportunities for Denormalization?

- Denormalize frequently queried fields such as patient demographics (e.g., include address in the Patient table).
- However, it depends on query patterns. If joins are not significantly impacting performance, keep the schema normalized.

e. Opportunities for Bitmap Index?

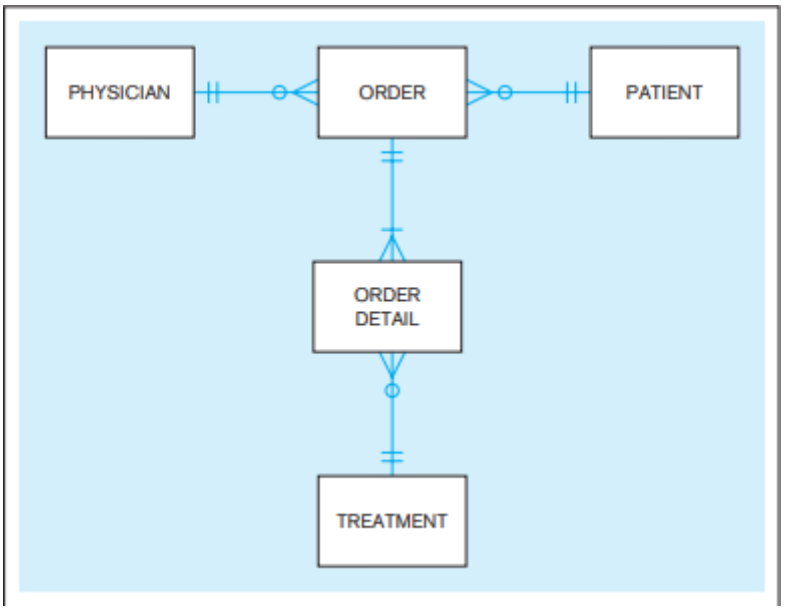
Bitmap indexes are ideal for lowcardinality fields such as:

- Gender or MaritalStatus in the Patient table.
- TreatmentType in the Treatment table.

f. Use of Join Index

A join index can optimize queries where Patient, Order, and Treatment are frequently join.

3. MVCH Figure 51 shows a portion of the data model for MVCH's database that represents a set of normalized relations based on the enterprise model shown in MVCH Figure 11 and additional business rules provided in the Chapter 2 case segment. Recall that TREATMENT refers to any test or procedure ordered by a physician for a patient and that ORDER refers to any order issued by a physician for treatment and/or services such as diagnostic tests (radiology, laboratory)



Using the information provided below regarding data volume and access frequencies, and following the example provided in Figure 51, modify the ER model shown in MVCH Figure 51 to create a preliminary composite usage map. a. Data volume analysis: • Recall from an earlier case segment that the hospital performs more than a million laboratory procedures and more than 110,000 radiology procedures annually. Add these two figures to arrive at the number of records for the ORDER DETAIL table. • There are approximately 250 PHYSICIANS, 20,000 PATIENTS, and 200,000 physician ORDERS in this database. • ICD9 procedure codes for treatments (lab procedures, radiology procedures, etc.) fall into approximately 3,500 major categories. Use this number to approximate the number of TREATMENT records. b. Data access frequencies per hour: • Across all applications that use the MVCH database, there are approximately 100 direct accesses to PHYSICIAN, 35 to ORDER, 200 to PATIENT, and 150 to TREATMENT. • Of the 200 accesses to PATIENT, 30 accesses then also require ORDER data, and of these 30, there are 20 subsequent accesses to PHYSICIAN, and 30 accesses to ORDER DETAIL. • Of the 35 direct accesses to ORDER, 10 accesses then also require PHYSICIAN data, and 20 require access to PATIENT data, ORDER DETAIL data, and TREATMENT data. • Of the 100 direct accesses to PHYSICIAN, 20 also access ORDER, ORDER DETAIL, and TREATMENT data. • Of the 150 direct accesses to TREATMENT, 10 also access ORDER DETAIL data and associated ORDER and PHYSICIAN data.

a. Data Volume Analysis

Records:

- **ORDER DETAIL:** More than 1,110,000 (sum of lab and radiology procedures annually).
- **PHYSICIAN**: 250 records.
- **PATIENT:** 20,000 records.
- **ORDER:** 200,000 records.
- **TREATMENT:** Approx. 3,500 records.

b. Data Access Frequencies

High access frequencies for Patient and Treatment tables suggest indexing on:

- PatientID in the Order table.
- TreatmentID in the OrderDetail table.

4. In Case Exercise 3, you created a composite usage map for part of the MVCH database, based on MVCH Figure 51. Referring to that composite usage map, do you see any opportunities for clustering rows from two or more tables? Why or why not? Is the concept of clustering tables supported in SQL Server? Does it differ from oracle implementation? If so, how?

a. Opportunities for Clustering Rows?

- Yes, clustering rows from Order and OrderDetail makes sense due to their frequent coaccess.
- Clustering rows can improve performance for readheavy queries.

b. Support for Clustering in SQL Server vs. Oracle

- **SQL Server:** Supports clustered indexes where the data is physically ordered based on the indexed column(s).
- **Oracle:** Uses IndexOrganized Tables (IOTs) for similar purposes but may differ in implementation details

ERD Diagram

The ERD includes the following entities:

- PHYSICIAN (physician_id, physician_name, specialty, phone_number).
- PATIENT (patient_id, patient_name, address, phone_number).

- ORDER (order_id, patient_id, order_date).
- ORDER_DETAIL (order_detail_id, order_id, treatment_id, quantity).
- TREATMENT (treatment_id, treatment_name, cost).

Relationships:

- PHYSICIAN 1--N ORDER
- ORDER N--1 PATIENT
- ORDER 1--N ORDER_DETAIL
- ORDER_DETAIL N--1 TREATMENT

Chapter 6: Introduction To SQL

Case Study: Mountain View Community Hospital

Case Questions:

1. What version of SQL and what RDBMS will you use to do the case exercises?

- **SQL Version**: I will use **T-SQL (Transact-SQL)**, which is specific to Microsoft SQL Server, as MVCH decided to implement their database using this RDBMS in Chapter 5.
- **RDBMS**: Microsoft SQL Server (e.g., SQL Server 2019 or later for optimal compatibility and features).

---

2. Which CASE tools are available for completing the case exercises? Can the CASE tool you are using generate the database schema from the physical data model(s) you created?

- **Available CASE Tools**:
  - **Microsoft SQL Server Management Studio (SSMS)**: Provides schema generation, query execution, and visualization.
  - **ER/Studio or ERwin**: For database modeling and schema generation.
  - **MySQL Workbench** (if using MySQL instead of SQL Server).
  - **dbForge Studio**: For schema generation and database design.
- **Schema Generation**:
  - Yes, most CASE tools (e.g., SSMS, ERwin) can generate the database schema directly from the physical data models using forward engineering. For instance, tables, relationships, and constraints can be auto-generated based on the logical or physical design.

---

3. Can you suggest an easy way to populate your tables if you want to create a large set of test data?

Yes, several methods can be used to populate the database tables with a large dataset:

1. **SQL Scripts**:

Write INSERT INTO statements manually or use loops in SQL (if supported) to generate test data.

```
INSERT INTO Patient (PatientID, Name, DOB, Gender)
VALUES (1, 'Basit', '1989-02-10', 'M');
```

2. **Data Generation Tools**:

- Use **SQL Server Data Generator** or third-party tools like:
  - **Mockaroo**: Generates mock data based on defined schemas.
  - **Redgate Data Generator**: A tool specifically designed to populate databases with realistic test data.

3. **Excel or CSV Import**:
   - Prepare test data in Excel or CSV and import it into SQL Server using the Import Wizard in SSMS.
4. **Custom Scripts in Python or Other Languages**:
   - Use libraries like pandas or Faker in Python to create large datasets programmatically and insert them into the database.
5. **Built-in Sample Data**:
   - If SQL Server includes built-in sample databases (e.g., AdventureWorks), these can be adapted for MVCH use cases.

---

4. How do the actual values you are using help you to test the functionality of your database?

- **Testing Database Integrity**:
  - Validate constraints such as **primary keys**, **foreign keys**, and **unique constraints** by trying to insert duplicate or null values.
- **Testing Query Accuracy**:
  - Use real-life-like data to ensure queries return correct and meaningful results (e.g., retrieving patient treatment histories).

- **Performance Testing**:
  - Populate tables with large datasets to evaluate query execution time, indexing efficiency, and response under load.
- **Scenario Testing**:

Use test data to simulate hospital scenarios:

  - Patient admissions, billing, and discharge.
  - Physician assignment to treatments.
  - Checking appointment schedules.
- **Validation of Data Relationships**:

Verify that relationships between tables (e.g., PATIENT and TREATMENT) work correctly by running queries that involve joins. By using varied and realistic test data, potential issues in design, queries, or performance can be identified and resolved before deploying the database.

Chapter 7: Advanced SQL

Case Study: Mountain View Community Hospital

Case Questions:

1. Does your SQL-based DBMS support dynamic SQL, functions, triggers, stored procedures, and UDTs?

- **Dynamic SQL**: Yes, Microsoft SQL Server supports dynamic SQL through the EXEC statement or the sp_executesql stored procedure, which allows you to execute dynamically constructed SQL queries at runtime.
  Example:
  ```
  DECLARE @query NVARCHAR(MAX) = 'SELECT * FROM Patient WHERE PatientID = 1';
  EXEC sp_executesql @query;
  ```
- **Functions**: SQL Server supports **scalar functions** (which return a single value) and **table-valued functions** (which return a table).
  Example of a scalar function:
  ```
  CREATE FUNCTION CalculateAge(@DOB DATE)
  RETURNS INT
  AS
  BEGIN
     RETURN DATEDIFF(YEAR, @DOB, GETDATE());
  END;
  ```
- **Triggers**: SQL Server supports **DDL (Data Definition Language)** and **DML (Data Manipulation Language)** triggers to execute custom logic in response to specific database events.
- **Stored Procedures**: SQL Server supports stored procedures for modularizing and reusing SQL logic.
  Example:
  ```
  CREATE PROCEDURE GetPatientByID(@PatientID INT)
  AS
  BEGIN
     SELECT * FROM Patient WHERE PatientID = @PatientID;
  END;
  ```
- **UDTs (User-Defined Data Types)**: SQL Server allows the creation of UDTs to define custom data types for specific columns.
  Example:
  ```
  CREATE TYPE PhoneNumber AS VARCHAR(15);
  ```

---

2. How can DDL triggers be used in support of HIPAA's mandate for audit controls?

DDL triggers can play a significant role in ensuring compliance with HIPAA's privacy and security rules by monitoring and logging changes to the database schema that involve **electronic protected health information (ePHI)**. Specifically, they help enforce audit controls by recording activities such as table creation, modification, or deletion.

- **Key Uses of DDL Triggers for HIPAA Compliance**:
  - **Track Schema Changes**: Log any alterations to tables containing ePHI to ensure no unauthorized changes are made.
  - **Prevent Unauthorized Actions**: Restrict certain users from performing DDL operations on sensitive tables.
  - **Audit Trail Maintenance**: Record the who, what, when, and how of DDL operations in an audit table for accountability.

- **Example Implementation**:
  A DDL trigger to log changes to tables:

```
CREATE TRIGGER AuditDDLChanges
ON DATABASE
FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE
AS
BEGIN
   INSERT INTO AuditLog (EventType, ObjectName, EventTime, UserName)
   VALUES (EVENTDATA().value('(/EVENT_INSTANCE/EventType)[1]', 'NVARCHAR(MAX)'),
        EVENTDATA().value('(/EVENT_INSTANCE/ObjectName)[1]', 'NVARCHAR(MAX)'),
        GETDATE(),
        SYSTEM_USER);
END;
```

- **Explanation**:
  - **EventType**: Captures the type of DDL operation (e.g., CREATE_TABLE).
  - **ObjectName**: Logs the name of the table affected.
  - **EventTime**: Records the time of the operation.
  - **UserName**: Identifies the user performing the action.
- **Advantages of DDL Triggers for HIPAA Compliance**:
  - Provides a detailed audit trail of schema changes.
  - Helps detect and prevent unauthorized access to sensitive data.
  - Ensures accountability by logging user activity.

This approach aligns with HIPAA's requirement to maintain robust audit controls for protecting sensitive patient information.

Chapter 8: Database Application Development

Case Study: Mountain View Community Hospital

Case Questions:

1. Should MVCH undertake the project of moving toward an integrated environment, or should it outsource the project?

MVCH IT staff, under Mr. Heller, could undertake the project because they have in-house expertise in managing their existing systems, which include relational databases and client/server technologies. However, outsourcing might be a more viable option due to the following reasons:

- **Advantages of In-House Development**:
  - Control over the project and system customization.
  - Internal staff will have better familiarity with existing systems.
- **Advantages of Outsourcing**:
  - Expertise in integration technologies such as Web services and portals.
  - Faster project completion due to specialized resources.
  - Reduced burden on in-house staff, allowing them to focus on operational IT needs.

Recommendation: A hybrid approach could be effective—outsourcing the integration while keeping critical elements like security and customization in-house.

---

2. Other approaches to integration besides Web-based solutions:

Alternatives to the Web-based approach include:

- **Middleware Integration**: Use middleware software like enterprise service buses (ESB) to connect disparate systems.
- **Database Consolidation**: Merge data from standalone systems into a single centralized database.
- **Cloud-Based Solutions**: Adopt cloud-based platforms for hosting shared applications and data.
- **Health Information Exchange (HIE)**: Participate in a regional or national HIE for easier exchange of patient data.

---

3. Extent and nature of security and privacy issues:

The committee should consider:

- **Access Controls**: Restrict access to patient data based on user roles (e.g., physicians, nurses).
- **Encryption**: Use encryption for data at rest and in transit to protect sensitive patient information.
- **Audit Trails**: Implement logging and monitoring to track data access and modifications.
- **Data Breach Response**: Develop a plan to address potential data breaches.

Integration is crucial for HIPAA compliance because it ensures consistent implementation of privacy and security rules across all systems.

---

4. Why has the health-care industry been slow to embrace Web services for integration?

- **Complexity**: Health-care systems are often older and harder to integrate with new technologies.
- **Regulatory Challenges**: Ensuring compliance with HIPAA and other laws can slow down adoption.
- **Cost Concerns**: Implementation can be expensive, especially for smaller hospitals.
- **Resistance to Change**: Physicians and staff may resist new workflows.

Critical Success Factors:

- Strong leadership support.
- User training and engagement.
- Incremental implementation.
- Robust privacy and security measures.

---

5. Is Web-based solutions a technology or a strategy issue?

It should be treated as a **strategic issue** because it impacts hospital operations, patient care, and long-term competitiveness. Technology is just a means to implement this strategy.

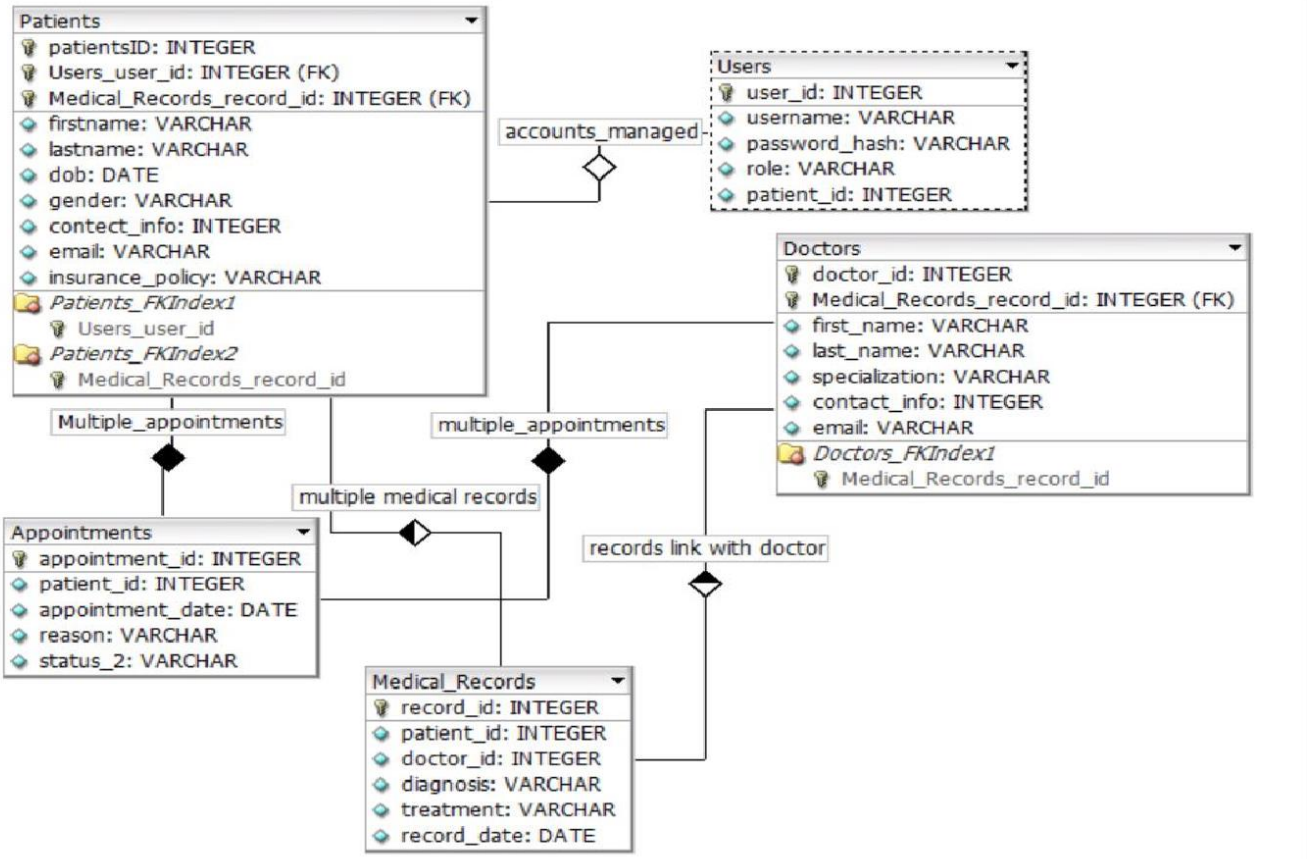
< Database Design >

ERD Explanation:

Relationships

1. **Patients ↔ Appointments**
   - A **patient** can have **multiple appointments**.
   - Each appointment must reference a specific **patient**.
2. **Doctors ↔ Appointments**
   - A **doctor** can have **multiple appointments**.
   - Each appointment must reference a specific **doctor**.
3. **Users ↔ Patients**
   - A **user** may optionally link to one **patient**.
   - This link is used for patient accounts managed via the users table.
4. **Medical Records ↔ Patients**
   - A **patient** can have **multiple medical records**.
   - Each medical record belongs to a specific **patient**.
5. **Medical Records ↔ Doctors**
   - A **doctor** can add **multiple medical records**.
   - Each medical record must be linked to a specific **doctor**.

Diagram:



Schema Of Mountain View Community Hospital (MVCH)

```
BEGIN
    EXECUTE IMMEDIATE 'DROP TABLE medical_records CASCADE CONSTRAINTS';
    EXECUTE IMMEDIATE 'DROP TABLE appointments CASCADE CONSTRAINTS';
    EXECUTE IMMEDIATE 'DROP TABLE users CASCADE CONSTRAINTS';
    EXECUTE IMMEDIATE 'DROP TABLE doctors CASCADE CONSTRAINTS';
    EXECUTE IMMEDIATE 'DROP TABLE patients CASCADE CONSTRAINTS';
    EXECUTE IMMEDIATE 'DROP TABLE notifications CASCADE CONSTRAINTS';
EXCEPTION
    WHEN OTHERS THEN NULL; -- Ignore errors if tables do not exist
END;
/
```

Table: Patients

```
CREATE TABLE patients (
    patient_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    first_name VARCHAR2(50) NOT NULL CHECK (REGEXP_LIKE(first_name, '^[A-Za-z]+$', 'i')), -- Restricts name to alphabets only
    last_name VARCHAR2(50) NOT NULL CHECK (REGEXP_LIKE(last_name, '^[A-Za-z]+$', 'i')), -- Restricts name to alphabets only
    dob DATE NOT NULL,
    gender CHAR(1) CHECK (gender IN ('M', 'F', 'O')),
    contact_info VARCHAR2(11) NOT NULL CHECK (REGEXP_LIKE(contact_info, '^[0-9]{11}$')), -- Exactly 11 digits, no alphabets
    email VARCHAR2(100) UNIQUE NOT NULL CHECK (REGEXP_LIKE(email, '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.com$', 'i')), -- Valid email format
    insurance_policy VARCHAR2(50) NULL
);
```

Table: Doctors

```
CREATE TABLE doctors (
    doctor_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    first_name VARCHAR2(50) NOT NULL CHECK (REGEXP_LIKE(first_name, '^[A-Za-z]+$', 'i')), -- Restricts name to alphabets only
    last_name VARCHAR2(50) NOT NULL CHECK (REGEXP_LIKE(last_name, '^[A-Za-z]+$', 'i')), -- Restricts name to alphabets only
```

```
    specialization VARCHAR2(100) NOT NULL,
    contact_info VARCHAR2(11) NOT NULL CHECK (REGEXP_LIKE(contact_info, '^[0-9]{11}$')), -- Exactly 11 digits, no alphabets
    email VARCHAR2(100) UNIQUE NOT NULL CHECK (REGEXP_LIKE(email, '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.com$', 'i')) -- Valid email format
);
```

Trigger: Patients

```
CREATE OR REPLACE TRIGGER trg_patients_contact_email
BEFORE INSERT OR UPDATE ON patients
FOR EACH ROW
BEGIN
    IF NOT REGEXP_LIKE(:NEW.contact_info, '^[0-9]{11}$') THEN
        RAISE_APPLICATION_ERROR(-20001, 'Contact number must be exactly 11 digits and contain no alphabets.');
    END IF;

    IF NOT REGEXP_LIKE(:NEW.email, '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.com$', 'i') THEN
        RAISE_APPLICATION_ERROR(-20002, 'Email must contain a valid format with @ and .com');
    END IF;
END;
/
```

Trigger: Doctors

```
CREATE OR REPLACE TRIGGER trg_doctors_contact_email
BEFORE INSERT OR UPDATE ON doctors
FOR EACH ROW
BEGIN
    IF NOT REGEXP_LIKE(:NEW.contact_info, '^[0-9]{11}$') THEN
        RAISE_APPLICATION_ERROR(-20003, 'Contact number must be exactly 11 digits and contain no alphabets.');
    END IF;

    IF NOT REGEXP_LIKE(:NEW.email, '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.com$', 'i') THEN
        RAISE_APPLICATION_ERROR(-20004, 'Email must contain a valid format with @ and .com');
    END IF;
```

```
END;
/
```

Table: Appointments

```
CREATE TABLE appointments (
    appointment_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    patient_id NUMBER NOT NULL REFERENCES patients(patient_id) ON DELETE CASCADE,
    doctor_id NUMBER NOT NULL REFERENCES doctors(doctor_id) ON DELETE SET NULL,
    appointment_date DATE NOT NULL,
    reason VARCHAR2(255),
    status VARCHAR2(20) CHECK (status IN ('Scheduled', 'Completed', 'Canceled')),
    CONSTRAINT appointment_unique UNIQUE (patient_id, doctor_id, appointment_date)
);
```

Table: Users

```
CREATE TABLE users (
    user_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    username VARCHAR2(50) UNIQUE NOT NULL,
    password_hash VARCHAR2(255) NOT NULL,
    role VARCHAR2(20) CHECK (role IN ('Physician', 'Admin', 'Patient')),
    linked_patient_id NUMBER NULL REFERENCES patients(patient_id) ON DELETE CASCADE
);
```

Table: Medical Records

```
CREATE TABLE medical_records (
    record_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    patient_id NUMBER NOT NULL REFERENCES patients(patient_id) ON DELETE CASCADE,
    doctor_id NUMBER NOT NULL REFERENCES doctors(doctor_id),
    diagnosis VARCHAR2(255) NOT NULL,
    treatment VARCHAR2(255) NOT NULL,
    record_date DATE DEFAULT SYSDATE
);
```

Table: Notifications

```
CREATE TABLE notifications (
    notification_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    message VARCHAR2(255) NOT NULL,
    notification_date DATE DEFAULT SYSDATE
```

```
);
CREATE INDEX idx_patient_last_name ON patients(last_name);
CREATE INDEX idx_doctor_specialization ON doctors(specialization);
CREATE INDEX idx_appointment_status ON appointments(status);

CREATE OR REPLACE TRIGGER insurance_update_notify
AFTER UPDATE OF insurance_policy ON patients
FOR EACH ROW
BEGIN
    INSERT INTO notifications (message, notification_date)
    VALUES ('Patient ' || :OLD.patient_id || ' updated insurance information.', SYSDATE);
END;
/
CREATE OR REPLACE PROCEDURE add_patient (
    p_first_name IN VARCHAR2,
    p_last_name IN VARCHAR2,
    p_dob IN DATE,
    p_gender IN CHAR,
    p_contact_info IN VARCHAR2,
    p_email IN VARCHAR2,
    p_insurance_policy IN VARCHAR2
) AS
BEGIN
    INSERT INTO patients (first_name, last_name, dob, gender, contact_info, email, insurance_policy)
    VALUES (p_first_name, p_last_name, p_dob, p_gender, p_contact_info, p_email, p_insurance_policy);
    COMMIT;
END;
/
CREATE OR REPLACE PROCEDURE schedule_appointment (
    p_patient_id IN NUMBER,
    p_doctor_id IN NUMBER,
    p_appointment_date IN DATE,
    p_reason IN VARCHAR2
) AS
```

```
BEGIN
    INSERT INTO appointments (patient_id, doctor_id, appointment_date, reason, status)
    VALUES (p_patient_id, p_doctor_id, p_appointment_date, p_reason, 'Scheduled');
    COMMIT;
END;
/
CREATE OR REPLACE PROCEDURE get_patient_details_as_xml (
    p_patient_id IN NUMBER,
    p_xml OUT CLOB
) AS
BEGIN
    SELECT XMLROOT(
            XMLELEMENT("Patient",
               XMLFOREST(patient_id, first_name, last_name, dob, gender, contact_info, email, insurance_policy)
            ), VERSION '1.0', STANDALONE YES
          )
    INTO p_xml FROM
    patients
    WHERE patient_id = p_patient_id; END;
/ BEGIN
    INSERT INTO patients (first_name, last_name, dob, gender, contact_info, email, insurance_policy)
    VALUES ('Farman', 'Afzal', TO_DATE('1980-05-15', 'YYYY-MM-DD'), 'M', '1234567890',
'farman@gmail.com', 'POL12345');


    INSERT INTO doctors (first_name, last_name, specialization, contact_info, email) VALUES
    ('Waqas', 'Ali', 'Cardiology', '987654321', 'waqas@gmail.com');


    INSERT INTO appointments (patient_id, doctor_id, appointment_date, reason, status)
    VALUES (1, 1, TO_DATE('2025-01-15', 'YYYY-MM-DD'), 'Routine Checkup', 'Scheduled');
    COMMIT;
END;
/
```