Artificial Intelligence and Digital Technology

Article

Utilize the Database Architecture to Enhance the Performance and Efficiency of Large-Scale Medical Data Processing

Xiangtian Hui [1,*]

[1] School of Professional Studies, New York University, New York, NY, 10012, USA
* Correspondence: Xiangtian Hui, School of Professional Studies, New York University, New York, NY, 10012, USA

Abstract: As the volume of medical data continues to grow, traditional database systems are increasingly challenged by demands for performance, scalability, and real-time responsiveness. Efficient database design is critical to meeting application needs in electronic medical record (EMR) systems, medical imaging storage, clinical decision support, and health data monitoring. This paper explores several architectural strategies to optimize database performance in large-scale medical environments. Techniques such as database sharding, table partitioning, index optimization, caching, and tiered storage of hot and cold data are shown to significantly improve system throughput, reduce latency, and enhance multi-threaded access efficiency. These methods collectively support the stable, secure, and scalable operation of modern healthcare information systems.

Keywords: medical data; database architecture; performance optimization; electronic medical records; caching strategies; database sharding; health informatics systems

Published: 13 December 2025

1. Introduction

The rapid advancement of hospital informatization has led to an exponential increase in the volume and complexity of medical data. Sources such as electronic medical records (EMRs), diagnostic imaging, monitoring equipment, and laboratory systems contribute to vast, heterogeneous datasets that require robust infrastructure for efficient management and processing. These datasets not only vary in structure but also demand high concurrency, rapid query response, and seamless scalability-requirements that often exceed the capabilities of traditional database systems. Inefficiencies in data retrieval, storage, and access can significantly hinder clinical workflows, delay decision-making, and compromise the real-time responsiveness of healthcare services. As a result, optimizing database architecture is critical for supporting high-performance, reliable, and secure healthcare information systems. This paper investigates how advanced database architectural strategies can enhance the performance and efficiency of medical data processing. It emphasizes the role of intelligent design in supporting diverse healthcare functions-including EMR storage, medical imaging management, clinical decision support systems (CDSS), and health monitoring platforms. Key optimization techniques discussed include database and table sharding, composite indexing, caching mechanisms, and cold-hot data separation-all aimed at improving throughput, minimizing latency, and ensuring system scalability under high data load conditions [1].

2. Basic Concepts of Databases

2.1. Composition of the Database System

A database system is composed of four core components: the database itself, the database management system (DBMS), application software, and end users. Each of these components plays an essential role in the operation and performance of medical data infrastructure. The database serves as the repository for both structured and unstructured medical data, encompassing patient records, clinical notes, laboratory reports, images, prescriptions, and more. The DBMS functions as the system's command center-responsible for organizing, managing, and securing the data. It provides standardized interfaces to handle operations such as data insertion, querying, updating, and deletion, while supporting essential system services including transaction control, logging, access permissions, backup, and recovery [2]. Application software interacts with the DBMS in accordance with specific healthcare workflows-such as clinical documentation, order entry, or test result review-and serves as the interface through which users engage with the system. Users, including physicians, nurses, and administrative staff, issue commands through graphical or terminal-based interfaces to execute data operations aligned with their respective responsibilities. Together, these components form a tightly integrated system characterized by high reliability, consistency, and concurrent processing capabilities. This architecture is especially suited to the demanding requirements of modern healthcare systems, where real-time access, data integrity, and operational stability are critical to safe and efficient clinical care.

2.2. Classification of Databases

Databases used in healthcare systems can be broadly classified into two categories: relational databases and non-relational (NoSQL) databases, each suited to different data types and application scenarios. Relational databases-such as MySQL, PostgreSQL, and Oracle-organize data in structured, tabular formats and are well-suited for managing transactional data with clearly defined schemas. These systems are commonly used for storing structured patient records, billing information, and standardized laboratory results. Their strengths include mature query capabilities, support for SQL, and robust data integrity mechanisms. In contrast, non-relational databases are designed to handle unstructured or semi-structured data, offering greater flexibility and scalability. These databases include Key-value stores (e.g., Redis) for caching and high-speed lookup. Document databases (e.g., MongoDB) for storing medical notes or imaging metadata. Columnar databases for analytics on large datasets. Graph databases for modeling complex relationships in biomedical research or patient networks. Additionally, time-series databases such as InfluxDB are optimized for storing and querying high-frequency, time-stamped data like vital signs, sensor readings, and real-time monitoring streams. Given the heterogeneous nature of medical data, modern healthcare systems often adopt hybrid database architectures, combining relational and non-relational technologies to accommodate diverse data types, operational loads, and performance requirements. This multi-model approach ensures flexibility, efficiency, and scalability in managing large-scale clinical data environments [3].

3. Application of Database Architecture in Medical Data Processing

3.1. Electronic Medical Record Management

The electronic Medical Record (EMR) system, as the core of medical informatization, the database design provides the basic service guarantee. The data of medical records contains structured data information such as basic patient information, visit information, medical order information, and drug record information, and can effectively construct the data structure by using the relational database. The database is divided into separate databases and tables by each department or visiting time period. Data is split according

to the demand to disperse the traffic of single-point access and optimize the concurrent processing capacity (See Table 1).

Table 1. Schematic Diagram of Sub-medical record Data Database Partitioning Design.

Sharding dimension	Sample value	Storage illustration
Department	Internal medicine, surgery, pediatrics	Be assigned to different database instances
Consultation time	Monthly or quarterly	Corresponding sub-table
Patient ID range	0001-9999.	Further table splitting to accelerate positioning

To improve the efficiency of retrieval, combined indexes can be generated in fields such as patient ID, ward, and date of visit, so that doctors can respond more quickly when using historical records or cross-table retrieval records. For some frequently retrieved and popular data, such as hospitalization information of the last three days and pharmacy lists, caching (for example, Redis) can be carried out to return results more quickly, thereby reducing the pressure on the main database. According to the patterns of busy and idle times, the query can be transferred to the read-only replica in combination with the shunting architecture, thereby maintaining a stable trend in the main database. Database design needs to be combined with access control to enable doctors of different levels to browse the corresponding data according to their permissions, thereby protecting medical history. From the perspective of the entire database, by adopting the strategy of regular repositories and regularly migrating historical data to backup repositories or cold storage repositories, the query speed is accelerated and the pressure on the main warehouse is reduced. After such design, the database structure meets the performance requirements of high efficiency, reliability and security of the EMR system [4].

3.2. Image Storage Optimization

Medical imaging systems generate a vast amount of high-resolution data from modalities such as CT, MRI, ultrasound, and X-ray. These images are typically large in size, unstructured in format, and frequently accessed for diagnosis and comparison-posing significant challenges for storage performance and retrieval speed. Rather than storing the full image files directly in the database, modern systems manage image metadata and access paths using a supporting database architecture. A document-oriented database such as MongoDB is well-suited for this purpose, offering flexibility to store metadata including image identifiers, patient IDs, acquisition timestamps, modality type, and departmental classification [5]. This structure enables efficient indexing and querying of non-uniform data across imaging modalities. To accelerate access to recently viewed or frequently referenced images, key-value caching systems like Redis can store access paths, reducing latency for repeated views and enhancing system responsiveness. Additional performance gains can be achieved by organizing images based on capture date or device source, enabling faster lookup by narrowing the search scope. A tiered storage strategy further improves efficiency by separating "hot" and "cold" data: Hot data (e.g., images from the past year) is stored on high-performance storage systems for rapid access. Cold data (e.g., older studies) is offloaded to cost-effective storage solutions such as cloud object storage, while maintaining index references in the primary database for traceability. For advanced search use cases-such as fuzzy keyword queries or annotation-based filtering-metadata tags and image labels enable clinicians to locate relevant images without manually scanning through archives. These optimizations collectively ensure that the imaging system can support real-time clinical demands while minimizing storage costs and maintaining accessibility.

3.3. Decision-making System Support

The Clinical Decision Support System (CDSS) requires high-quality and diverse medical information to provide services for it, and the storage format of the information is related to the service efficiency of CDSS. CDSS can retrieve the patient's past information, test reports, diagnosis and treatment information, etc. from the latest patient's data to assist in decision-making or issue warnings. To improve search efficiency and reduce the repetitive scanning of a large number of tables, patient data can be saved separately for different time periods or disease types. Alternatively, tables or entity views can be generated in advance for quick search of input to avoid duplicate calculations. Key information such as diagnostic codes, examinations, and medications is accelerated for rule matching and model query efficiency through multi-level indexing. Or, by means of the buffering mechanism, the calculation results of common models or the evaluation results of rules can be retained to further shorten the system response time. The database structure can also operate in conjunction with an event-driven system. When the values of important indicators change, the decision engine is immediately invoked in the form of triggers to enhance real-time promotion (see Figure 1).

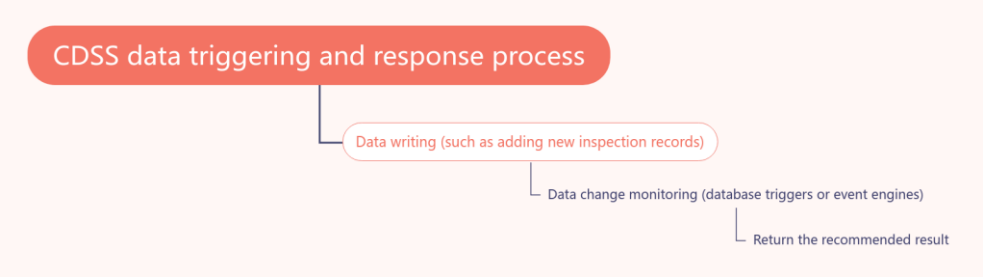

Figure 1. CDSS Data Triggering and Response Process Table.

In order to continuously optimize the system, it is also necessary to record the push results and the operation feedback of medical staff into the system to provide a basis for adjusting the decision-making model and rule logic in the later stage.

3.4. Health Monitoring and Management

In health management monitoring, the data structure should meet the processing requirements of a large amount of continuous data. The data sources include measurement devices for various vital signs such as heart rate, blood pressure, blood oxygen and body temperature. The data is continuously generated and updated, which will pose challenges to the input rate and the speed of data retrieval. Time series databases such as InfluxDB can efficiently store and compress time series data, accurately marking the time, number and status of each monitoring point. Combined with the relational database, it meets the requirements of information storage for patients' basic information and monitoring rules, as well as unified data retrieval and storage.

To provide anomalies and warnings, the database system needs to monitor the data flow in real time together with the stream processing engine. When unusual data is detected, the information can be transmitted to the foreground display screen for visualization by the event triggers in the database. To ensure that historical information can be traced back, this system can create tabular forms every day or every hour and improves the retrieval efficiency of history. Some key indicators can be generated automatically, such as average days and amplitude, to save system costs. In response to sudden peak demands, the database design scheme should adopt a master-slave synchronization approach to distribute the input and output pressure of data.

4. Optimization Strategies for Enhancing the Performance and Efficiency of Large-Scale Medical Data Processing by Leveraging Database Architecture

4.1. Database and Table Sharding Enhance Concurrent Processing Capabilities

As the requirements for generating a large amount of medical data are getting higher and higher, the access volume and access frequency also increase accordingly, resulting in bottlenecks for a single database structure in terms of concurrent access volume and response speed. This requires the adoption of sharding data according to the set rules, splitting the data into multiple independent databases or tables, reducing the pressure on a single point, and enhancing the overall system's usage efficiency. In most cases, horizontal segmentation can be carried out based on the uniqueness of the patient's identification, the time of visit, data type, etc., and the data can be stored separately physically or logically.

Information from different departments can be allocated to separate data entities respectively to avoid information interaction among departments and enhance the degree of information separation. Historical records can be stored in blocks or in a dispersed manner along the time axis, which is convenient for limiting the data volume of a single table, optimizing the search path and improving the search efficiency. The design of the database can also be combined with distributed database management tools to achieve automatic processing of data transmission and read/write allocation, which can ensure that the experience of visitors remains unchanged when the infrastructure changes.

In a read-write separation architecture, write operations occur on the primary database, while read operations take place on the secondary database. The replication mechanism is used to synchronize data to ensure data synchronization. Query pressure can also be horizontally scaled, and database nodes can be added according to the concurrent situation of the business to form an elastically scalable structure. When evaluating the processing capacity of a system, throughput (TPS) is commonly used as a measurement indicator, and its calculation formula is as follows:

$$\text{System throughput} = \frac{\text{Successfully processed the number of requests}}{\text{Unit time (second)}} \quad (1)$$

Enhancing the concurrent processing capacity of the database through sharding databases and tables can significantly improve the overall throughput efficiency of the system and meet the requirements for real-time performance and stability in high-traffic medical scenarios.

4.2. Establish a Combined Index to Accelerate Multi-Condition Queries

In the medical information system, there will be a large number of cross-field retrieval requests, such as querying medical cases, selecting test results, and conducting data statistics, which have high requirements for the retrieval efficiency of the database. The lack of an appropriate indexing scheme can lead to complex cross-conditional queries, causing a full table scan of the entire table and resulting in problems such as slow response and low system efficiency. Indexing is regarded as the main way to improve complex queries. Forming index paths with multiple fields optimizes the database to quickly locate the content.

Composite indexes are generally constructed starting from fields that are frequently queried and have a strong filtering effect, based on the order in which the fields are used, to avoid duplicate scans and invalid pairs. Common index creation methods accelerate search by using efficient algorithms such as B+ trees to achieve features like range scanning, prefix matching, and ordered query. Structured data, such as the patient's number, date of visit, and examination items, can significantly accelerate the filtering processing of operations like WHERE and JOIN by creating a multi-field joint index.

When the database executes SQL statements, it automatically selects the optimal index based on the index selector and further optimizes the program execution path with the help of the query optimizer, saving resource costs.

4.3. Introduce Caching to Reduce the Pressure of Database Access

For frequently used medical information systems, query operations are often carried out on similar data, so their databases often bear considerable pressure. For example, doctors retrieve medical record information, nurses view prescriptions, and the system loads test result reports, etc. Although these operations do not exert as much pressure on the database as inputting medical data, However, since it is completed in a high-concurrency operating environment, the database resources will be depleted rapidly, resulting in a relatively slow system response speed. To reduce the pressure on the host and improve the query efficiency of the system, it is advisable to introduce a caching mechanism into the system. By setting up an application-level buffer structure or an intermediate-level cache structure (Redis, Memcached), hot data can be migrated from the database to the memory, thereby enhancing the system's carrying capacity.

Cache generally uses key-value structures to build its own architecture. After the system's initial access to the database, it stores the results in the cache and retrieves them during subsequent accesses, thereby reducing the process of repetitive queries. Cache hit rate is an important indicator to measure its optimization effect and is defined as follows:

$$\text{Cache hit rate} = \frac{\text{Number of hits}}{\text{Total number of visits}} \times \text{One hundred percent} \quad (2)$$

When the cache hit rate is high, most requests do not need to be queried from the database, which can greatly reduce the system's latency and the pressure on the database. Reasonable cache designs, such as hot data identification, expiration policy configuration, and timed clearing, can effectively ensure the freshness of data and the stability of the system.

4.4. Hierarchical Storage Optimizes Cold and Hot Data Management

The data access patterns for different types of related medical information are also significantly different. For instance, data such as case summaries and test records are frequently retrieved and used within a short period of time, but information like past images and invalid test records is hardly used. By adopting a storage method of hierarchical management of cold and hot data, Frequently used information can be stored in media with higher performance, while information with less access probability can be stored in cheaper storage media. By taking both into account and complementing each other, it is possible to achieve both improved storage efficiency and cost reduction. Under normal circumstances, the data of the previous year is classified as "hot data" and stored in SSDS or high-end storage. The data after this period of time is positioned as "cold data" and is saved in cloud files or object storage, with indexes and access paths maintained by the database (see Table 2).

Table 2. Cold and Hot Data Storage Strategy Table.

Data type	Storage location	Update frequency	Respond to the demand	Sample data
Hot data	SSD, local disk	High frequency	second level	Recent examination results and hospitalization records
Cold data	Cloud object storage	Low frequency	Minute-level	Early images, hospitalization data from five years ago

By periodically storing the data in the database in the database archive, it is transformed into cold data; Use the cache layer and middleware to change the access path

to balance query performance and system reliability, and reduce the consumption of high-performance storage resources.

5. Conclusion

As healthcare systems increasingly rely on massive, diverse, and rapidly evolving datasets, database architecture has become a critical foundation for ensuring the performance, scalability, and reliability of medical information systems. This paper presents a comprehensive examination of architectural strategies designed to optimize large-scale medical data processing. By implementing database and table sharding, institutions can distribute data workloads and significantly enhance concurrent processing capacity. Composite indexing enables rapid execution of complex, multi-condition queries, while caching mechanisms alleviate pressure on core databases and reduce latency in high-frequency access scenarios. Furthermore, the adoption of tiered storage strategies-separating hot and cold data-supports efficient resource utilization and cost control without sacrificing accessibility or system responsiveness. Together, these architectural optimizations allow medical data systems to meet the rigorous demands of modern clinical environments, ensuring high availability, data integrity, and fast decision support. Tailoring database design to the specific needs of healthcare workflows promotes not only system performance, but also improves care delivery by enabling timely, informed, and data-driven clinical decisions. Moving forward, the integration of intelligent data orchestration, dynamic schema evolution, and machine learning-driven data management will further elevate the role of database architecture in digital healthcare transformation.

References

1. R. S. Nickerson, "How we know-and sometimes misjudge-what others know: Imputing one's own knowledge to others," *Psychological bulletin*, vol. 125, no. 6, p. 737, 1999.
2. D. J. Joel Devadass Daniel, and S. Ebenezer Juliet, "A secure data storage architecture for internet of medical things (iomt) using an adaptive Gaussian mutation based sine cosine optimization algorithm and fuzzy-based secure clustering," *Journal of Medical Imaging and Health Informatics*, vol. 11, no. 12, pp. 2883-2890, 2021. doi: 10.1166/jmihi.2021.3838
3. Y. Lina, and S. Wenlong, "Efficient Processing of Large-Scale Medical Data in IoT: A Hybrid Hadoop-Spark Approach for Health Status Prediction," *International Journal of Advanced Computer Science & Applications*, vol. 15, no. 1, 2024. doi: 10.14569/ijacsa.2024.0150108
4. R. Kapuscinski, "The other," *Verso Books*, 2018.
5. S. M. Andersen, S. Chen, and R. Miranda, "Significant others and the self," *Self and identity*, vol. 1, no. 2, pp. 159-168, 2002. doi: 10.1080/152988602317319348