CLINICAL TRIAL MANAGEMENT SYSTEM DESIGN AND DATA STRUCTURE

Boris Ivanov Bankov[1]

[1] *Chief Assist. Prof., PhD, Department of Informatics,*

University of Economics, Varna, Bulgaria.

E-mail: *boris.bankov@ue-varna.bg*

JEL: C88

Received: 30-03-2024

Accepted: 20-05-2024

Published: 28-06-2024

ABSTRACT

One of the industry areas where web and mobile technologies bestow impactful benefits for both researchers and users is the pharmaceutical sector, particularly in streamlining clinical trials, enhancing patient engagement, and improving data collection and analysis. This paper surveys the existing academic landscape regarding the challenges in clinical trial management. We provide insights into the practical implications of different database organizational strategies and their effect on system performance, particularly during intensive operations such as data exports and complex queries. Our discussion extends to the digitalization of clinical trial processes based on our experiences, though specifics of the drugs tested are not disclosed. We present an analysis of two different database design approaches – Conventional and Entity-Attribute-Value (EAV). Our goal is to assess how these architectures support the flexibility required for the rapid adaptation and deployment of new functionalities, a necessity given the short durations typical of clinical trials. This analysis aims to guide the optimization of Clinical Trial Management Systems (CTMS) for better operational efficiency in managing clinical trials.

Keywords: *clinical trial management systems, entity-attribute-value model, Laravel, database design*

Citation: *Bankov, B. (2024) Clinical trial management system design and data structure. Journal "Човешки ресурси & Технологии = HR & Technologies", Creative Space Association, 1, pp. 6 – 20.*

INTRODUCTION

Innovations in web and mobile smartphone applications have brought transformative changes and improvements across numerous fields, from education and finance to logistics and healthcare. Computing power is ever-so rapidly increasing and the possibilities to deliver scalable and high-performative solutions are endless. To state the obvious, sectors once bound by traditional processes are now embracing these technological advancements, setting new standards for optimizing costs, and enhancing interactivity and accessibility.

The landscape of medical and pharmaceutical software has rapidly evolved to accommodate the increasing complexity and demands of modern healthcare and drug development processes. As the digital revolution of the healthcare sector accelerates, a wide array of software applications have emerged, addressing everything from electronic health records and patient management to sophisticated pharmaceutical drug testing. These technological advancements have significantly improved the effectiveness of healthcare delivery and drug research, enabling better patient communication and feedback and streamlining operations across organizations.

Within this broad domain, specialized web applications designed for managing and conducting short clinical trials represent a crucial innovation, especially in the context of accelerating drug development timelines and improving adaptability in research methodologies. Short drug trials, often necessitated by urgent public health needs or rare medical conditions, require robust, agile, and compliant software solutions that can manage complex data, ensure regulatory compliance, and support dynamic collaboration across multidisciplinary teams. The traditional clinical trial process has often been time-consuming and resource-intensive, posing challenges in terms of participant recruitment, data collection, and real-time monitoring. Web applications in this niche not only enable real-time data capture and analysis but also enhance patient screening and engagement by providing user-friendly interfaces and remote access capabilities. This technological support is vital for shortening the duration of drug trials without compromising the quality and integrity of the research, thus paving the way for faster market entry of essential medications.

Clinical Trial Management Systems are software applications that enhance the processes in pharmaceutical rollout, while automating and simplifying data tracking, collection and reporting. When developing these systems different guidelines can be followed:

- "Guidance for Industry: Computerized Systems Used in Clinical Trials" from the U.S. Food and Drug Administration (FDA);

- "Guideline on the Computerised Systems and Electronic Data in Clinical Trials" from the European Medicines Agency (EMA).

These documents can provide a more in-depth academic as well as industry-level perspective of the proper CTMS design importance.

Available scientific literature suggests that there is a certain lack of software exploration and details about technology used for various Clinical Trial Management Systems. The goal of this paper is to present two approaches – Conventional and Entity-Attribute-Value model – for designing the database architecture and business logic of CTMS based on the same in-house codebase. To complement our practical experience, we offer insight and code examples on several tasks and configurations of the developed systems. As a result, developers of such systems may consider our tooling ecosystem. During the development lifecycle of each of the software systems different philosophies are used to balance code refactoring, performance and ease of ad-hoc feature introduction and alteration. An important factor when implementing such systems is the database design and structure. The key reasons include data integrity and accuracy, efficiency and performance, regulatory compliance as well as scalability and adaptability. Due to the short length of each trial, additional functionality needs to be readily available for deployment and software flexibility is key.

1. LITERATURE REVIEW

The topic of digitalizing clinical trials has been an interesting area of research since the early 2010s. The clear advantages of using more expert software like RealTime-CTMS, Medirio, TrialKit, Clinical Conductor, BSI or Bioclinica include reduction of human errors, greater reliability, archival and backup capabilities as well as data visualization and real-time monitoring.

Zhao et al. (2013) highlight the need for customized evaluation software in clinical trials to accurately assess the outcomes of pulmonary function tests (PFTs), which are commonly used to evaluate lung function. Traditional methods of PFT analysis often rely on manual measurements and subjective interpretations, leading to potential errors and inconsistencies. Key features include automated data processing, improved data visualization and reporting, extensibility, and flexibility.

More recent research papers of Eagleson et al., (2017) deal with privacy concerns in a web-based or mobile-based clinical trial research. The authors explain the unexpected delays in gaining ethics approval about collecting and managing personal data. As a result, the study

team needs to provide security and privacy for its participants as well as the ability to have the data audited at any time. Their research is based on a 12-month period examination – Smart Heart Trial is a single arm feasibility study that follows a yearlong health intervention.

Rodrigues et al. (2018) propose a standardized protocol for conducting software-assisted randomized clinical trials in healthcare treatments. They present a flowchart of 5 stages: formulation, criteria establishment, information extraction and data analysis, digitalization of physical approaches, dissemination of knowledge and meta-analysis.

Zhou et al. (2021) develop a web-based software platform designed to support various early-phase clinical trials by integrating multiple Bayesian optimal interval (BOIN) methods under a unified framework. It offers tools for single-agent and drug combination trials, accommodating scenarios such as late-onset toxicity and the determination of optimal biological doses. The software ensures a uniform and superior user experience by utilizing a consistent software architecture and a proven development standard operating procedure. Key functionalities include generating decision tables, simulating trial recommending doses, and estimating the maximum tolerated dose or optimal biological dose upon trial completion and preparing protocols.

There are also examples of researchers that have presented meta-analysis of scientific contributions in the software design of CTMS. Grayling and Wheeler (2020) review 31 journal articles to show availability of code and software used for Adaptive Design in clinical trials. To fall under the criteria of relevance to AD clinical trials certain keywords are looked for, e.g. adaptive randomization; Bayesian methods; Biomarker-based methods; Dose-modification/escalation; Group sequential; Sample size adjustment. Their discovery is that only 30% of all discussed solutions have code present in some form. They note that sharing computer code submissions can be subject to specific journal policies. It needs to be further discussed whether sharing discoveries in the healthcare sector is as unpopular as detailed intellectual property disclosure of software algorithms. In our expertise, NDAs on business logic and software implementations limit broader insight on the topic.

Similarly, others (Heesen and Roos, 2024) present a meta-analysis of clinical trial software aimed at recruitment management. This systematic review revealed that software implementations of complex statistical methods for recruitment prediction and monitoring are scarce and generally not available as free, open-source tools. This scarcity limits the ability of principal investigators and funders to use these methods routinely, which can affect the efficiency of recruitment processes in clinical trials.

The dynamic nature of conducting clinical trials in the digital age requires reliable and scalable software solutions. Business algorithms and data organisation as well as presentable and user-friendly layout all have their intricacies and are subject to modernization and unification. The reviewed studies show that there is a space to be explored that offers more insight into the technological aspects and software tools used in CTMS.

There are other numerous academic studies that present interesting challenges and their solutions in the field of Clinical Trial management. The majority of research concerns data algorithms, data safety and data validity. In the next chapter practical applications of database organization is presented as well as more broad approach to CTMS.

2. PROJECT DESIGN AND TOOLING

Building a CTMS requires both expertise in software development and knowledge in pharmaceutical sciences, ensuring that the platform efficiently manages clinical trials from inception to completion. Additionally, a deep understanding of regulatory compliance, data security, and user-centric design is crucial. In this section first we will describe the physical processes of conducting clinical trials and the digitalization of different stages from our experience, without going into details and specifics what drugs are tested.

Measuring the success of a clinical trial relies the validity and accuracy of the data acquired from researchers and participants. Case Report Form or CRF protocols are documents used for continuous and systematic data collection. These protocols follow all the steps in screening and accepting patients into the trial, as well as follow-up visits at specific dates after the drug administration has begun or has concluded. Typically, these protocols are represented by physical documents which each doctor must maintain and fill out for each patient while the trial is ongoing and at the end submit it to the drug manufacturer or the pharmaceutical board. CRF protocols for each step of the trial are prepared and it is key to have the data submitted at the time of the patient visit to provide valid and accurate information to reflect the condition and effects of the drug administration.

The process can be broken down into several stages. The first stage is the screening questionnaire. Screening helps the recruitment stage by filtering out patients which do not meet the criteria for being included in the trial. The screening forms can include questions about gender, age, height, weight, body mass index, alcohol consumption, tobacco and smoking habits, hereditary or other chronic conditions which are treated with medications that can skew

or affect the results of the trial. After the screening procedure patients can be randomized or selected manually for a specific therapy group.

Based on the type of recruitment and placement into the study groups there are several different designs:

- Single-blind: the patient does not know if he is taking the active ingredient drug or the placebo one;
- Double-blind: neither the patient nor the doctor know if the participant is getting the active drug or the placebo;
- Crossover: the patient receives both the treatment and the placebo at different times during the study;
- Open-label: both the researcher and the patient know which treatment is being administered.

Modern studies require an equal or near-equal number of participants both in the active and placebo treatments. In some cases, both groups can be divided into smaller groups, based on gender or age range. Furthermore, some studies can be multicentred, meaning researchers and patients from different cities can participate. In these cases, web and mobile-based CTMS can ensure that the randomization process of giving treatments does not favour one group more than the other, which as a result can cause young males to be given the active drug and older males to be given the placebo for example.

Once patients receive the treatment researchers need to schedule specific dates for follow-up checks and visits. Depending on the trial additional visits can be required with each having a CRF to track specific symptoms and prescribe additional therapies if the illness is not progressing favourably. Furthermore, participants can also use CTMS to submit their daily health status with these patient diaries or journals can last from anywhere from a week to couple of months or more. These can vary between different research teams as some diseases last longer than others or the doctors prescribe a different additional drug and administration period. Keeping track of a large number of patients with different visitation dates can be difficult for a physical CTMS but is a trivial task for modern software capabilities and solutions. Figure 1 shows a simplified lifecycle of the CTMS we built for two different drug treatments.

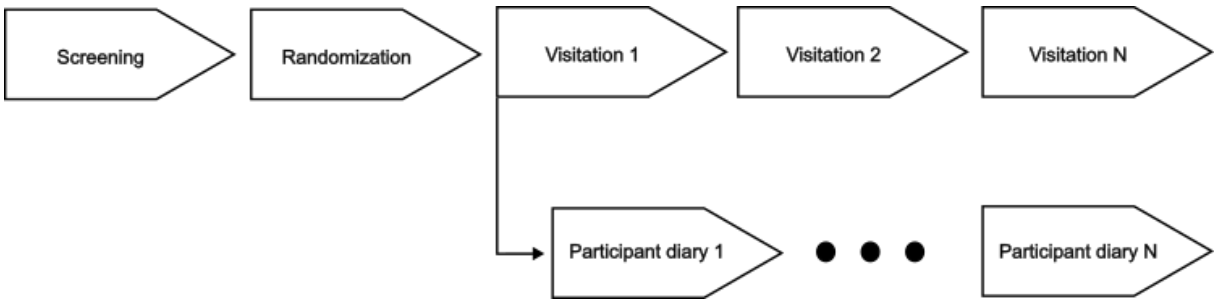

Figure 1. Simplified CTMS lifecycle

Source: Own Elaboration

To develop the CTMS we used Laravel. Laravel is a PHP-based open source back-end framework for building dynamic web applications. It implements the MVC (Model-View-Controller) architecture, which organizes the application structure into three interconnected components. By separating the business logic from the user interface and data handling, the development process can be focused on specific aspects of the application without affecting the rest of the project. Laravel is designed to facilitate the development of web applications by providing a rich set of features such as user interface layouts and templating, routing, authentication, sessions, security, email handling, and caching.

Initially released in 2011 Laravel has gone through eleven major releases, with available extensive official documentation from 4.2 up to 11.x. The framework has become one of the most popular and widely used PHP development kits due to its vast community, strong syntax, default features, wide range of modules and stellar performance.

The convergence of the object-oriented paradigm with relational database systems poses a series of technical challenges. Developers' expertise in Object-Oriented Programming (OOP) does not seamlessly translate to the implementation of data storage and updating operations such as selection, insertion, modification, and deletion within database environments. Object-based modelling of the real-life problems encompasses both the data and the logic for its processing. In the design of relational databases, the emphasis is placed exclusively on data management. The typology of data differs significantly, lacking a direct correspondence between the types utilized in programming languages (Common System Type) and those employed in database systems.

A key feature present in Laravel is the Eloquent ORM (Object-Relational Mapping). Each table in the database is represented by a "Model" class that is used as a template for table fields and relationships. Eloquent provides unified syntax for different database management systems and improves common tasks in data visualization and processing as well as

ActiveRecord implementation. Additionally, issues such as the N+1 Query Problem are resolved using an eager-loading mechanism to retrieve nested records from one-to-many relationships.

Although JavaScript has been more popular due to frameworks such as React and Angular, PHP is still more widely used on the account of the sheer number of existing web applications and PHP-stack tech resources. Laravel has cemented itself as the artisans` choice for building robust, scalable and high-performance web solutions.

3. IMPLEMENTATION

To ensure that all data collected is not altered between the researcher submitting participant data and what is saved in the database, any change to the database via the Laravel CTMS is recorded with an audit package (https://laravel-auditing.com/). To enable record tracking for each table, the Auditable trait must be included in each Model (MVC). If there is a database table that is not represented by a model, or if the model does not have the trait, any create, update or delete events would not be logged. The events are in a sequential order with an additional unique ID pertaining to the specific table entity that is being modified. They contain user, device and browser information in addition.

#example Model

```
class Researcher extends Model implements Auditable {
      use \OwenIt\Auditing\Auditable;
}…
```

Some events do not make changes to the database, like unsuccessful log-in attempts or downloading PDF or Excel reports. To log these, we use PHP magic constant __METHOD__ to record the name of the Controller method and who and when called it in each construction function.

#example Class and function

```
class UserController extends Controller {
      public function login(Request $request){
            $event->log_event($request->get('user'), __METHOD__);
      } …
}
```

A significant challenge we face is the implementation of the CRF protocols. For one of the projects, we use a standard table to view approach (Conventional database schema). For the screening stage we create a Model, a Controller and a View. The interface consists of 43

questions. Each question is represented by a single field in a database table. With this approach we only save answers or submitted values into the database. Some questions are set as Yes/No options, e.g. "Have you been a heavy smoker in the last 6 months (average more than a pack a day)?". Other questions ask participants about specific symptoms and offer answers on a scale from 1 to N. There are also blank single-line input fields for numeric values, like oxygen saturation or body temperature. As a result, the interface template file ends up at 600+ lines of code.

#example simplified form component for the screening researcher

```
<label> Oxygen Saturation (92-100%)</label>
<input type="number" placeholder="Enter a number">
```

We use the same approach to design the rest of the interface forms. Some code needs to be reused in the interface for follow-up visits. This is because specific parameters of the patient's health are monitored throughout the whole treatment. Using a conventional database schema means we either use another table in the database solely for visit number N, or a separate table entity specifically for those questions that are duplicate in subsequent visits with a user and visit identification number alongside the health values.

This straightforward conventional design is useful if there will be minimal changes to the interface after it is finished. It is neither easy nor too complex to maintain large interface files if new questions need to be inserted at specific positions or rearranged and modified. Running queries to extract specific data is also fast and effortless. A bigger challenge is setting up reports. To create an excel sheet with all 43 columns and populate them with data, we can use a Laravel exports. Creating column headers is time-consuming because the question text is not saved in the database but rather in the interface file. To generate a report, we query the database and place column labels separately. In this case these labels appear both in the interface component and in the report as plain text rather than variables.

example Export

```
class ScreenExport implements FromArray, WithHeadings {
      public function headings(): array {
            return [
             'Weight', 'Height', 'BMI', 'Oxygen Saturation', …
           ]
      }
      public function array(): array {
            …
            return ScreenModel::get ();
      }
}
```

If during the testing period changes need to be made to the interface, this will also reflect on the export, by adding, removing, or renaming columns headers. The trade-off is that this approach is relatively low-cost performance-wise, and the application can work smoothly and without delays as the database queries are simple.

For the second project we expected more dynamic environment and build the database in such a way that form questions, the type of the question (yes/no, number, free form text, etc), and the answers are saved in the database. The different stages of the treatment are sequentially numbered, and an order is created for which questions from the databank appear in specific interface during the treatment. Here we apply the **Entity-Attribute-Value** (EAV) design. This model is often used in scenarios where the number of attributes (properties, parameters) that can be stored about an entity (such as a person, product, etc.) is vast, but the number of attributes that will actually be used for any single entity might be relatively small. The EAV model is particularly useful in medical, CRM, and e-commerce systems where the schema is dynamic, and attributes of data can vary extensively from one entity to another. This model provides flexibility in terms of database schema, allowing the addition of new attributes without altering the overall database schema. Nguen et al. (2023) propose building a national platform for disability healthcare management using EAV model. Earlier research from Kim and Park (2012) show research into assessment and clinical decisions support with EAV database schema.

On Figure 2 a simplified database structure is shown, displaying the relationships between the key table entities. If changes are required to a specific visitation questionnaire only the database will need to be adjust. For all user interfaces and questionnaire exports the code is not modified even if questions and possible answers are added, removed or rearranged. Between the Screening phase and the last Visitation check-up all the data that is submitted is retrieved with the same algorithm even though there are differences in the number of monitored parameters. For example, on the last check-up there are fields about final condition and quantity of returned medication, which are two attributes that do not exist in the Screening phase. This solution help create really short interface components, e.g. the Screening form template was only 250 lines of code. And the programming segment can be reused more easily as the only difference will be in the visit number that is up next during the treatment.

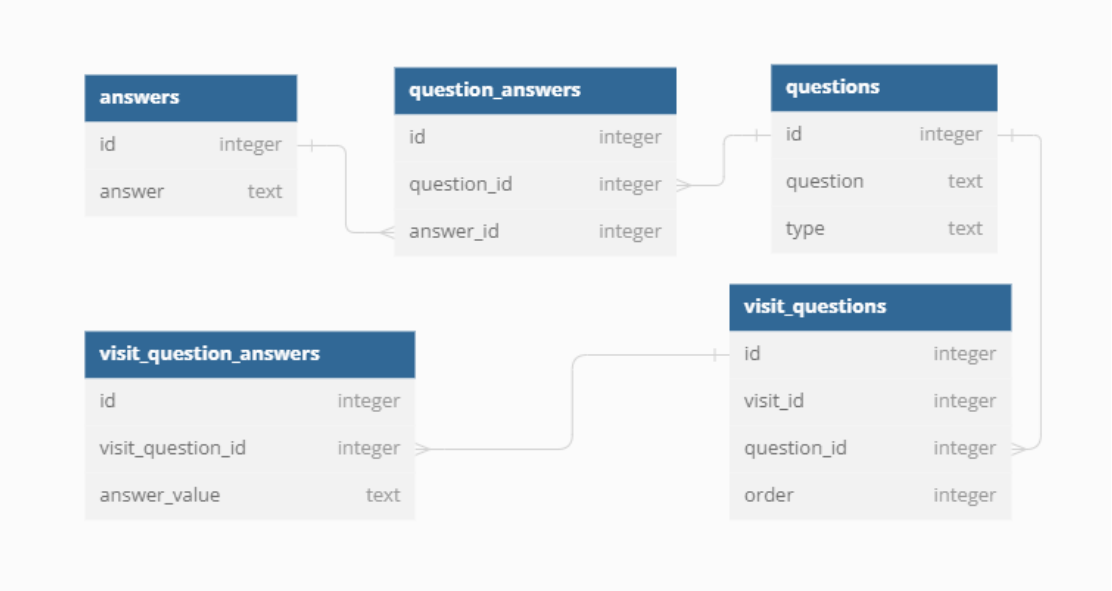

Figure 2. Database schema for question – answer EAV structure

Source: Own Elaboration

The interface component with all the fields can be loaded in a single for-loop with the help Laravel's Object Relational Mapping. Questions, Answers, Answers to questions and so on have separate class Models and methods that show the relationship between them, e.g. One-To-Many.

example of loading question labels and saved value

```
 @foreach($visit->questions->question_answers as $question_answers)
   <label> {{ $visit->questions->question }} </label>
   <input type="{{ $visit->questions->type }}" value="{{
$question_answers->answers->answer }}">
 @endforeach
```

The drawback of this approach is that when the export is being designed, the questions (column headers) are loaded in a single column, which forces us to transpose the first column into a row. The algorithm is significantly longer but general insight can be seen in the example below.

example Export with transposing and merging column headers and column data

```
public function headings(): array {
   $questions = $VisitQuestionModel::get();
   $transpose = $questions = array_map(null, ...($questions->toArray()));
   return $transpose;
}
public function array(): array {
   return array_merge($transpose, VisitQuestionanswerModel::get();
}
```

It is worth noting that this design is performance heavy only during complex exports and select/filters from the database. When we want to run a query on how many participants have not completed their diary for a specific date or a range of dates, the task itself is anything but expedient and often causes delays in the performance of the application. Although from a software engineer's perspective this solution is more elegant, it is also more sophisticated. Databases with fewer tables but more records might perform better for simple queries that require fewer joins, particularly in data warehousing or analytical contexts where large volumes of data are queried and aggregated. This is supported by the research of Kimball and Ross (2019) where the authors show the benefits of using a specific database scheme where there is a large fact table that is supported with smaller dimension tables. More tables with fewer records can enhance performance in transactional systems where data integrity and quick access to highly relational data are necessary. Normalization of tables is proven to be impactful for faster retrieval of specific records and efficient updates to individual tables according to Harrington (2016) and Date (2019).

DISCUSSION

The two CTs follow different treatments with variable parameters ranging from duration to participant number and drug intake period. Furthermore, this study can be improved by using database benchmark tests. Additional experiments need to be conducted with mirroring each database structure with the data for each clinical trial. Useful tools for this task are TPC benchmarks, BenchmarkSQL, SYSBench, and others. The Transaction Processing Performance Council (TPC) tests are appropriate and well-used standards that provide detailed and audited metrics on various aspects of database management and transaction processing performance. Benchmarking to show the differences between database schemas, particularly variations in table configurations and record distributions, involves a custom approach since typical benchmarks like TPC-C or TPC-H use standardized schemas and workloads.

The metrics that should be monitored include query performance, resource utilization and concurrency stability. We can easily observe decreased query execution speeds with the second database schema example on complex SELECT operations which due to how much data is spread in related tables require JOIN operations as well. More research can be analysed to improve and optimize the data structure to help specific complex tasks (Kuyumdzhiev, 2015), (Żabiński & Zielosko, 2020). The EAV model represents a significant deviation from

traditional relational database models, trading off schema rigidity for flexibility, which is beneficial in certain domains but can introduce complexity in data handling and analysis.

Implementing pure conventional relational database or EAV schema favour specific tasks. Some hybrid approaches exist that aim to combine the strengths of both models while mitigating their weaknesses. Here are some common hybrid approaches:

- Vertical Partitioning – with this structure, the core attributes are stored in relational tables. They are essential for the data presentation and are frequently used in queries. The highly variable attributes are stored in EAV tables to provide flexibility and accommodate schema changes.
- Horizontal Partitioning – Each entity type has its dedicated table with common attributes. This allows for efficient querying within that entity type. Attributes that are shared across multiple entity types are stored in a single EAV table. This reduces redundancy and simplifies data management for shared attributes.

Hybrid approaches can be more complex to implement and maintain than pure relational or EAV models. Careful design and indexing are essential to ensure optimal query performance across different storage models. Further studies can acknowledge and present the challenges of adopting hybrid database structures.

CONCLUSION

Advances in software and hardware capabilities help build a more stable and efficient environment for conducting drug and health treatment trials. Throughout the paper, we have discussed the importance of digitalizing clinical research and considering previous works in the field, described the design process from a software point of view. In the context of modern web and mobile applications, Laravel is suitable framework to implement a CTMS. Overall, the study highlights some key features of the MVC framework and how we approached the interface components in regard to maintaining readable and scalable code.

The use of an EAV database structure is not without challenges. Issues related to performance, data integrity, and complexity of query operations have been identified and addressed with innovative solutions, including hybrid approaches that combine the strengths of EAV with other database architectures to optimize both flexibility and performance.

Future research should focus on refining these hybrid models, improving the interpretability and ease of data retrieval, and integrating advanced data analytics capabilities directly into the CTMS. Such advancements will undoubtedly enhance the efficacy of clinical

trials by ensuring data is not only manageable but also readily analyzable, leading to faster and more informed decision-making processes in clinical research environments.

REFERENCES

1. Date, C.J., 2019. Database design and relational theory: normal forms and all that jazz. Second Edition. Apress.
2. Eagleson, R., Altamirano-Diaz, L., McInnis, A., Welisch, E., De Jesus, S., Prapavessis, H., Rombeek, M., Seabrook, J.A., Park, T. and Norozi, K., 2017. Implementation of clinical research trials using web-based and mobile devices: challenges and solutions. BMC medical research methodology, 17, pp. 1-8.
3. FDA, 2015, Guidance for Industry - COMPUTERIZED SYSTEMS USED IN CLINICAL TRIALS. [Online] Available at: https://www.fda.gov/inspections-compliance-enforcement-and-criminal-investigations/fda-bioresearch-monitoring-information/guidance-industry-computerized-systems-used-clinical-trials, [Accessed 2024].
4. EMA, 2023, Guideline on computerised systems and electronic data in clinical trials. [Online] Available at: https://www.ema.europa.eu/en/documents/regulatory-procedural-guideline/guideline-computerised-systems-and-electronic-data-clinical-trials_en.pdf, [Accessed 2024].
5. Grayling, M.J. and Wheeler, G.M., 2020. A review of available software for adaptive clinical trial design. Clinical Trials, 17(3), pp. 323-331.
6. Harrington, J.L., 2016. Relational database design and implementation. Fourth Edition. Morgan Kaufmann.
7. Heesen, P. and Roos, M., 2024. Freely accessible software for recruitment prediction and recruitment monitoring of clinical trials: a systematic review. Contemporary Clinical Trials Communications, p.101298.
8. Kim, H.Y. and Park, H.A., 2012. Development and evaluation of data entry templates based on the entity-attribute-value model for clinical decision support of pressure ulcer wound management. International journal of medical informatics, 81(7), pp. 485-492.
9. Kimball, R. and Ross, M., 2019. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, Ed. Wiley.
10. Kuyumdzhiev, I. 2015. Arhivirane i vazstanovyavane na bazi ot danni v MongoDB: vazmozhnosti, sastoyanie, problemi. Izv. na Suyuza na uchenite – Varna. Ser. Ikonomicheski nauki, pp. 125-133.

11. Nguyen, T.N., Nguyen, L.M., Pham, T., Dinh, M., Khwakhali, U.S. and Tran, Q., 2023, July. Big Data for Healthcare: using Entity-Attribute-Value (EAV) model to build a national platform for disability management. In 2023 International Conference on System Science and Engineering (ICSSE), pp. 554-558
12. Rodrigues, L.M.L., de Lima, I.B., dos Santos, L.R.A., Bollela, V.R., Cruz-Cunha, M.M., Rijo, R.P.C.L. and Alves, D., 2018. Towards a standardized protocol for conducting randomized clinical trial for software. Procedia computer science, 138, pp.125-130.
13. Żabiński, K. and Zielosko, B., 2020. Decision rules construction: Algorithm based on EAV model. Entropy, 23(1), p.14.
14. Zhao, Z., Möller, K., Miiller-Lisse, U., Vogt, B. and Frerichs, I., 2013. Customized evaluation software for clinical trials: An example on pulmonary function test with electrical impedance tomography. In 2013 ICME International Conference on Complex Medical Engineering pp. 128-133
15. Zhou, Y., Lin, R., Kuo, Y.W., Lee, J.J. and Yuan, Y., 2021. BOIN suite: A software platform to design and implement novel early-phase clinical trials. JCO Clinical Cancer Informatics, 5, pp. 91-101.