Using Document-Based Databases for Medical Information Systems in Unreliable Environments

Oliver Schmitt
Department of Information Systems
University of Münster, Germany
research@schmitt.io

Tim A. Majchrzak
Department of Information Systems
University of Münster, Germany
tima@ercis.de

ABSTRACT

Healthcare and crisis management are pervaded by the usage of Information Systems (IS). Virtually all IS rely on data storage. Despite the document-oriented nature of medical datasets, the prevailing kind of database used are relational (RDBMS) ones. In order to find a more adequate solution in a development project for a patient-registry, we evaluated a document-based database incorporated into the data storage layer of a system. To foster the understanding of this technology, we present the background of form-originated data storage in healthcare, introduce document-based databases, and describe our scenario. Based on our findings, we generalize the results with a focus on crisis management. We found that document-based databases such as CouchDB are well-suited for IS in medical contexts and might be a feasible option for the future implementation of systems in various fields of healthcare, crisis response, and medical research.

Keywords

Medical information system, crisis management system, database, document, document-based, CouchDB

INTRODUCTION

Information systems (IS) in the fields of healthcare are routinely used. Running a hospital without a Hospital Information System (HIS) has become unthinkable and most physicians use Patient Information Systems (PIS). Moreover, IS form the interface between healthcare and patient-centered medical research. And also in cases of emergencies and crises, access to medical IS is vital. Almost all IS used in this context rely on a database (DB) that stores data about patients and related information such as diagnoses, treatment plans, and prescriptions. A high number of records is document-based; e.g. x-ray images or electroencephalography wave recordings.

The predominant kind of databases used in IS are relational (RDBMS) ones (Codd, 1983). Despite the strong focus on documents that can be observed in this context, this also applies to medical IS. The relational concept indisputably has some amenities and it is widely understood by IS developers. However, it also has drawbacks. We will not attempt to contribute to a discussion of pros and cons of relational databases. We rather deem checking alternative concepts worthwhile. Particularly, the document-based databases caught our interest due to the document-focus we observed in medical IS development projects we were involved in.

Document-based databases can be attributed to the *NoSQL* (not only SQL) databases (Lith and Mattsoon, 2010). These DBs do *not only* rely on the Structured Query Language, which is predominately used to access data from relational databases. The idea of using a document-centered database is not new; it has been incorporated into *Lotus Notes* as early as 1989 (Mohan, 1999). However, interest in document-based databases was revived.

We have used the document-oriented CouchDB (Anderson, Lehnardt and Slater, 2010; Holt, 2011a; Lennon, 2009) in a project based on work of colleagues from the field of medical informatics, who have designed and developed *DocFrame*, a Web-based documentation framework for electronic data capture for use in clinical care (Hartz, Verst, and Ueckert, 2009) as well as patient registries (Brüntrup, Lablans and Ückert, 2011b; König, Omran, Schmitt, Lablans and Ückert, 2012). They have also published a generic software library as a reference implementation for compliance with rigid data protection laws (Brüntrup, Lablans, and Ückert, 2011a).

The sophisticated patient registry is used in a multi-tier architecture that provides a Web interface; layers (e.g. business logic and data storage) are loosely coupled. Brüntrup et al. (2011b) decided to use CouchDB since they found a relational database to be inferior in concept to the document-based approach. Refraining from using an

RDBMS was preceded by an elaborate analysis; nevertheless, our work allows learning from using CouchDB and judging the general feasibility of it. At the same time, we want to check the applicability for crisis management. In contrast to a stationary setting, challenges such as a distributed environment, unclear conditions, potential loss of personal and technical problems have to be taken into account, each leading to additional requirements toward the database choice. A number of these problems with be addressed subsequentially.

Our work makes several contributions. Firstly, it summarizes the status quo of database usage for data typically entered via forms in the field of healthcare and comprehensively introduces document-oriented databases. Secondly, it illustrates why document-oriented databases are a viable option for medical IS. Thirdly, it exemplarily shows the usage of CouchDB. We have observed it in an actual scenario and give detailed hints how to use it in a RESTful scenario. Fourthly, it describes steps towards a generalization and a theorization of our findings.

This paper is structured as follows. The next section gives the background of form-originated data-storage in medical IS. We then introduce document-based databases. Based on this, we propose our approach towards using document-oriented databases for healthcare-related IS. We subsequently highlight implications for crisis management and discuss our findings. Finally, we draw a conclusion and highlight future work.

DATA-STORAGE IN MEDICAL INFORMATION SYSTEMS

Medical information systems store all kind of data like medical findings, measurements, and artifacts like x-ray images. This information is often collected with digital forms that accept text-based data and attachments such as images. Digital forms commonly support workflows that made use of paper-based forms and relied on analogue media in the past. Typical workflows that use the forms are standardized anamnesis procedures or clinical examinations but forms are also used for research purposes.

Using forms in such systems is divided into two phases: *design* and *usage*. In the design phase the form is created by setting up the visual appearance for screen and printer output. Additionally, data structures that store the form data are created. Besides the graphical representation, metadata are added for input verification. For instance *regular expressions* (Friedl, 2006) are employed. The graphical representation of the form, data structures for content storage, and metadata for input verification outline the so called *form definition*.

In the usage phase the information system uses the form definitions (Figure 1) to render the form on the screen (Figure 2). Aided by a more powerful form rendering engine, it is possible to derive other output formats from the form definition e.g. for printing or sending as PDF file via email.

```
{ _id: "fs_forms_medication_1",
  _rev: "C73B200B",
  form: {
          medication: {
                        type: "list",
                        items: [
                                  { type: "date-box",
                                    rule:  [ "mandatory" ],
                                    caption: "Date"
                                  },
                                  { type: "numerical-box",
                                     rule: [ "mandatory,
                                             "[1-10]"  ],
                                    caption: "Amount" }
                                ]
                      }
}
```

Figure 1. Declarative description of the input form as a JSON form definition

The data structures that store the form content are key-value pairs. For a given set of keys that can be used as captions while the form is displayed on the screen, the user enters values. A medical IS that handles form data must be able to render the form on the screen and display data that has been entered in the past. Additionally, it must be able to store new or changed data. Consequently, the information system has to support CRUD (Create, Read, Update, Delete) (Martin, 1983) operations on all forms.

As questions of liability are important in the medical environment, systems typically have to maintain an audit trail and to provide a mechanism for integrity verification. The audit trail consists of information about *what* was changed at *which* point of time by *which* user. This is usually achieved by logging user activities and creating revisions of form data (Anderson, 1996). Integrity verification is implemented by using cryptographic signatures over the logging data and form data revisions. From a juridical standpoint, it is essential that verification data cannot be altered.

The general concept of form usage in medical information systems as illustrated above is straightforward. However, the main challenge is that form definitions are changed frequently. This is caused by the fact that the requirements of forms are not fully know in advance during the design phase and that adjustments are needed in order to meet the demands of the users; reasons for change arise both from medical concerns and for usability issues. Additionally, the workflows that are supported by the forms might also be changed resulting in adjustments of form definitions. This is particularly relevant for crises, since consequences cannot be fully anticipated.

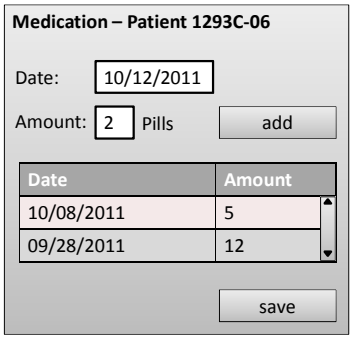

Figure 2. Form rendered by an information system based on the Form Definition

Coping with frequent changes in forms is not trivial. In fact, a well-though out concept is required for each IS that is subject to frequent changes of the way data is entered and of the kind of it. Altered form definitions have to be reflected in the data models both for documentation (and, therefore, maintenance) and consistency. Keeping existing datasets while changing the model poses problems particularly if data attributes are removed or their semantics are altered. Furthermore, it has to be pondered upon how to handle archived data revisions.

Some differences have to be noted regarding the DB requirements of standard medical IS and those used in the context of crises and emergencies. While the kind of data entered into medical IS might change rapidly, their setup is rather enduring. System landscapes only slowly change, new systems are only introduced occasionally, and the level of technical and technological dynamic is low. In particular, changes hardly happen during runtime. In crises settings, the setup is much less static. Faster response is required. Typically, a higher level of resilience and adaptability to a changing environment is required. This is challenging for databases and contrasts with the goal of securely and permanently saving data. To illustrate the differences you can think of two extreme cases. The first would be a system in a hospital whereas the server is hosted within the premises and clients are spread among wards. The second is a centrally served response network access by mobile client on disaster sites with instable network connections. In both cases, data access is vital but requirements are completely different.

DOCUMENT-BASED DATABASES

An approach is proposed as an alternative to relational databases. We introduce the document-based CouchDB, which offers features for resilient IS in distributed environments. Then, we have a look at related work.

Basic Concept and CouchDB

The basic idea behind a document-based database is that the smallest unit for storing data is a *document*. Documents can be structured data consisting of key-value pairs. The structure of the pair can be chosen freely; it is not required to be shared among documents. For structuring *JavaScript Object Notation* (JSON), *YAML Ain't Markup Language* (YAML), or *Extensible Markup Language* (XML) are used. JSON is a simple, text-based, human-readable data-interchange format based on a subset of JavaScript (Cockford, 2008) and typically used for RESTful Web service as a lightweight alternative to XML (Richardson and Ruby, 2007). YAML is a simplified, human-readable data serialization format ("YAML Spec", 2009) that can be seen as a superset of JSON.

The database system provides access to the documents and makes them available for accessing applications. In contrast to a RDBMS no schema is necessary within the document-based DB (Bhat and Jadhav, 2010). Although the idea of a database without schemes does not seem to be efficient, there are industry-proven products such as the above mentioned Lotus Notes that make use of such databases for the last two decades.

CouchDB is database system that uses the principle of document-based storage. CouchDB is an *Apache project* and available as *free* software. The aim of the developers is to provide a document-based database that offers high performance, good scaling behavior, and makes usage of the *MapReduce* algorithm (Dean and Ghemawat, 2008). This algorithm is the key to access data from large, typically network-distributed datasets (Holt, 2011b).

The server is accessible via a RESTful (Webber, Parastatidis and Robinson, 2010) JSON application programming interface (API). All documents are available in a *flat* address space. The database is queryable and idexable featuring a table-oriented reporting engine that uses JavaScript as the query language. CouchDB offers bi-directional incremental replication with integrated conflict detection and management (Bhat and Jadhav, 2010).

Existing Applications

Related work can be identified from two perspectives despite the fact that we know of no paper with the same approach. Firstly, work can be checked that addresses using NoSQL for IS in the field of healthcare. Secondly, it is a good idea to compare our work to cases of document-based DB usage in other fields. It has to be noted that there are alternatives to CouchDB such as MongoDB (Chodorow and Dirolf, 2010). However, general concepts introduced in this paper more or less apply to all document-oriented DBs. A comparison is out of scope.

Meissner, Luckenbach, Risse, Kiste, and Kircher (2002) present steps towards an "integrated disaster management communication and information system". They stress the importance of a "flexible information and work flow concept" and demand "distributed, redundant, and mobility adequate databases". While not explicitly naming NoSQL as a solution, assessing such databases can be seen as an implicit consequence of their work.

The idea of using not only SQL can be traced back to the 1960s. The Massachusetts General Hospital Utility Multi-Programming System (MUMPS) (Wasserman and Sherert, 1976) is still in use in hospital environments. However, it is conceptionally different to today's NoSQL DBs. The number of recent papers that deal with NoSQL in the medical domain is very low. A search on PubMed Central e.g. shows only seven results, which for instance briefly address NoSQL in the context of research methods (Russ, Ramakrishnan, Hovy, Bota and Burns, 2011) or large-scale data processing (Taylor, 2010).

BHOMA (2011) is a project to establish a medical information system in Zambia that uses CouchDB. While we know of no scientific assessment of the project, it underlines the feasibility of CouchDB in scenarios with high demands regarding replication but technical instability. As a second example, it is reported that the Danish government made a contract to use a NoSQL DB for its medical prescription card program ("Basho.com", 2011). Similar applications of NoSQL are discussed in Weblogs and Internet Forums. We will not cite these sources as they tend to report one-sided. Nevertheless, they document practitioners' interest.

There is an increasing number of articles on NoSQL in other fields but most of them have a technological focus. Thus, only very few approaches could be identified that can be related in concept. Cruz, Gomes, Oliveira, and Pereira (2011) present results from an application in telecom applications. They argue that NoSQL can be cost effective if a "change of how we reason about data as well as the data structures that support it" is accepted. Ruflin, Burkhart, and Rizzotti (2011) highlight the usage of NoSQL for storing social data. Although they identified amenities e.g. of the key-value-concept, they conclude that "finding the right storage system […] is still tricky" and demand further research. Hanlon, Dooley, Mock, Dahan, Nuthulapati and Hurley (2011) describe amenities of using CouchDB for "portals & science gateways". In particular, they identified speedups with regard to making data available. While the above examples are related to our work in general, they underline that not much research has been published on the implications of using NoSQL in actual projects, yet.

DOCFRAME'S APPROACH TOWARDS PROCESSING FORM-BASED DATA

In the following sections we first describe the scenario in which CouchDB was used by Brüntrup et al. (2011b). We then give details on how the document-based approach was used. We will analyze its appropriateness to tackle the challenges of form data in the medial context. Eventually, we draw intermediary results.

Scenario Description

Present medical IS typically are designed as multi-user applications that implement the client-server paradigm (Raghupathi and Tan, 2002) and use industry-proven RDBMS. The choice of relational databases is driven by the fact that the relational paradigm is widely taught in universities and is well-understood by today's software engineers. However, other paradigms appear to be more suitable for new applications with tasks such as handling large set of unstructured data or providing elastic scalability (Bhat and Jadhav, 2010). Therefore, we argue that such approaches should be discussed and evaluated. In the following, a medical IS is proposed that utilizes document-orientation. It is able to replicate data on multiple places without a permanently available network.

Our scenario is not artificial: Brüntrup et al. (2011b) and Hartz et al. (2009) have implemented novel IS that share three main requirements although being used in distinct fields. In both patient care and medical research

information is collected through forms; documenting changes in the data has to be possible, integrity protection is required, and high flexibility concerning form design and structural data is essential. Both systems are Web-based and use multiple document-based DBs and application servers to separate master data and medical information to comply with German data protection guidelines. Experiences are presented subsequently.

The Document-Based Idea

Document-based DBs use documents as atomic unit for storage. Hence, every dataset derived from a form is stored as a distinct document. Additionally, every related form definition that declares the appearance and behavior of a form is stored as a document. Unique identifiers are used to make documents accessible. Systematically declared identifiers enable a virtual folder structure. The identifier `fs_data_medication_1293C-06_2` for instance can be translated as a virtual folder structure delimited with underscores instead of slashes, where `fs` is the root folder and the folder `data` contains all records of medications that were applied to a group of patients. The identifier `1293C-06` points to the data of the patient with the respective id. The revision number `2` at the end of the document ID as well as the metadata and signatures are explained later.

```
{ _id: "fs_data_medication_1293C-06_2",
  _rev: "D1C946B7",
  data: {
         medication: {
                       [ { Date: "10/08/2011",
                           Amount: "5"
                         },
                         { Date: "09/28/2011",
                           Amount: "12"
                         },
                       ]
                     },
         }
  formdef: "fs_forms_medication_1",
  metadata: {
              changed-by: "f242-dc92",
              changed-at: "10/11/2011 - 11.23pm",
              prev-rev: "fs_data_medication_1293C-06_1"
            },
  signature: "d8e8ece39c407e515aa8997c1a1e94f1fd2a0e62"
}
```

Figure 1. CouchDB JSON document with metadata for logging and digital signature for integrity verification

For each dataset that was entered into a form, a respective document is created within the database. The document that contains the data of the form depicted in Figure 1 has a key `data` that stores data as a JSON object. As the medication may contain *0* to *n* entries, a *list* data structure is used that can be extended easily. The medical IS also adds the key `formdef` that contains the identifier of the form definition that was displayed to the user while he was entering the data. This key is important for rendering the data if the same form definition is requested later. To implement the audit trail, the systems adds the key `changed-by`, which contains the user's id, and the key `changed-at`, which contains the point of time when the data has been saved to the document. To ensure integrity, the application logic of the medical information system computes a cryptographic hash value over the values of the keys `data`, `formdef`, and `metadata` that is stored in the key `signature`.

Revisioning can be implemented without much effort. However, maintaining revisions has to be implemented into the IS logic. With CouchDB there are two options to realize this. The first option is to create *attachments* to documents. An attachment can be any binary content, which is represented as a *Base64* encoded string. If data was changed or added, the application logic of the medical IS appends the previous version of the document as an attachment to the current version of it. Any former revision of the document can then be accessed by tracking down the attachment structures. The second option is to extend the identifier of the documents holding the form data by a revision number. For instance, consider the identifier `fs_data_medication_1293C-06_2`. The medical IS determines the latest revision and then adds a new document with an incremented identifier.

It has to be noted that tasks often pursued by a RDBMS are shifted towards the application if using a document-based DB. Nevertheless, by focusing on additional important requirements, we will see that a database without any schema will be advantageous in this use case. As pointed out before, the appearance of forms and its functionality often change. With document-based DBs a new document is created for every form definition and every version of the form. If the system maintainers change a form definition, the new definition is saved as an own document with an incremented version number. As every form data maintains a reference to its form definition in the key *formdef*, the data can be displayed in the form that was valid at the point of time at which data

was entered. Data added newly is stored based on the newest versions of form definitions. This dramatically simplifies coping with changes from the database perspective. Although this principle saves time and effort, there are situations in which the newest form definition has to be applied to old data. Consider that a form is faulty. In this case additional application logic is required that iterates through the form documents, selects the affected form type, transforms the data determined by the revision number, and updates the reference to the newest form definition. This upgrade scenario requires custom-written code, which makes the upgrade procedure more complex at first hand. However, the amount of code to write is often small as data in documents is processed in batch. In comparison to a relational database, this approach allows reverting the upgrade by simply restoring the older revisions – a possibility that is often not given in relational databases once the data schema has been upgraded.

Accessing Documents can be done with the mechanism of *views* (Anderson et al., 2010). Views can be considered as a filter to include documents in a subset of the database. It is also helpful to generate indexes for finding documents by any key or value that is stored within documents. Views can moreover be used for different kinds of calculations on the document data. View functions in CouchDB are stated in JavaScript or, alternatively, in programming languages provided by third-party developers. To generate a view, the view function is applied to every document in the database. If a document meets the requirements that are stated by the function, it is added to the *document set* of the view. Document sets are persistently stored as B-Tree structures – temporary views have speed penalties (Lerner, 2010). As the view function is applied to all documents in the database, it takes some times until the new view is available (Anderson et al., 2010). Once it is ready, access to the included documents is fast[1] (Bhat and Jadhav, 2010). As a consequence, the focus of query execution is shifted from the calculation speed to storage utilization.

CouchDB uses a RESTful JSON API for data access. A basic HTTP-GET is used to access documents by a Uniform Resource Identifier (URI). For creating new documents, HTTP-POST is used; HTTP-PUT is used to update an existing document. IS rely on these simple operations for interaction with CouchDB. Access is possible locally and remotely. As requesting of URIs is implemented in most common programming languages and frameworks, little programming is necessary to implement the interface between application logic and database.

CouchDB comprises a built-in replication mechanism via basic HTTP operations, which can be triggered on demand (as batch) or continuously. In the latter case the target databases are kept up-to-date to the source database. Continuous replication is interesting to synchronize different databases over a network (Anderson et al., 2010). If a conflict occurs during replication, CouchDB marks the conflicted documents. The application explicitly is responsible to resolve conflicts (Anderson et al., 2010). This concept aligns with the earlier described shift of functionality from database to business logic, which contrasts RDBMS.

Besides the suitability for storing form-based data in the medical context, CouchDB is also interesting for medical IS specifically in the light of crisis management due to its replication mechanism. The peer-to-peer replication of CouchDB enables usage of medical data in environments in that network connections are not always active or unreliable. This allows breaking with the paradigm of client-server computing and enables an advantage to a peer-to-peer environment with little infrastructure overhead. This is particularly important in crisis situations. An example are electronic triage tags: they could rely on a remote database but would require data storage of their own for situations of network unavailability (Gao and White, 2006).

Conflict management of CouchDB's replication mechanism is *predictable*. Since resolution is a task of the application, additional error-correction can be implemented on the application layer. For instance, if a data medium such as a hard disk is corrupted, errors can be detected and compensated in the application if data is stored in multiple independent CouchDB nodes. Consequently, CouchDB is not only suitable for peer-to-peer scenarios but also for building resilient application as typically required in the field of crisis management.

Intermediary Results

As pointed out above, Brüntrup et al. (2011b) and Hartz et al. (2009) have developed two medical IS which make usage of CouchDB. *eKernPaeP 2.0* is a novel Web-based online and offline documentation system, which has been developed for pediatric palliative care-teams supporting patient documentation and communication among healthcare professionals (Brüntrup et al., 2011b). It was designed for reliability and enables fast and secure home care documentation (Hartz et al., 2009) eKernPaeP 2.0 is accessible online by registered users using a Web browser. It offers synchronization between the offline systems and the central database servers. Due to the demands of the involved physicians, nurses, and relatives of the patients form definitions have to be altered

[1] We cannot provide a figure; benchmark comparisons of SQL and NoSQL DBs are problematic (Cattell, 2011).

frequently. The flexible handling of form definitions is helpful for improving the documentation process and to improve the quality of living of patients. The other system that uses CouchDB is Nephreg, a Web-based documentation system for long term observation of patients suffering of the cystic kidney disease nephronophthisis (König, Omran, Kurlemann, Schmitt, Lablans, Ückert and Konrad, 2012).[2] To date it covers more than 60 patients from twelve different research centers in Germany. Data collection is the precondition for related research projects. Thus, form definitions have to be kept up-to-date to the current development in medical research. Moreover, demands of researches that run the patient registry can be easily reflected in the system without impairing existing data.

Both systems implement mechanisms for maintaining audit within CouchDB trails as they are suggested in this paper. In order to realize the feature of storing old revisions, binary attachments to CouchDB documents are used that contain older versions of the entire document. To create backups, only little effort is needed: there is the possibility to replicate the content of the database to a remote database instance. Additionally, the files containing the database content are backed up while the database is running. This is possible since CouchDB's file layout and commitment system features all Atomic Consistent Isolated Durable (ACID) properties (Anderson et al., 2010). Hence, the uptime of the system can be maximized as functional and complete backups of the database can be created without impairing the operation of the system and common file-based backup solutions can be used for additional safety. In the case of Nephreg the entire content of the database is additionally *dumped* with scheduled tasks during the night into JSON files with Base64 encoded binary attachments. These dumps are generated with Python scripts and result in a human-readable format. Both the dumps and the CouchDB database files are included in the reliable file-based backup mechanisms of the data center.

CONTRIBUTION TO CRISIS MANAGEMENT

Due to its strong feature set and design concept, CouchDB does not only allow building flexible distributed medical systems but it is also well suitable for information systems in crisis management. In the preceding sections, various hints to possible applications have been given. In order to show the suitability, we consider the information system architecture emergency services proposed by Meissner et al. (2002) and show how CouchDB fits perfectly into their requirements. The idea of the authors was to use distributed data sources in every involved organization and by its participants with distributed mobile information systems running on different system architectures. The architecture of their system is depicted in Figure 4.

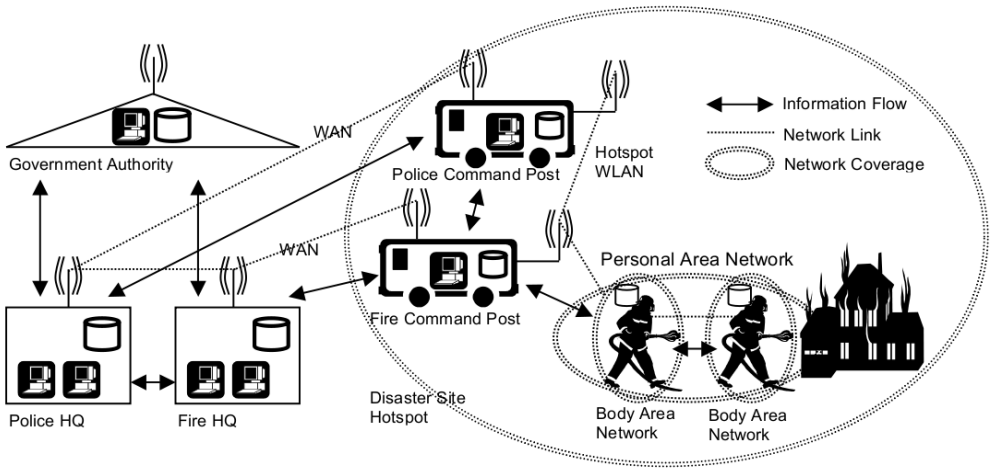

Figure 2. Architecture of a Disaster Management System (Meissner et al., 2002)

The idea of Meissner at al. was directed towards the future. No database that was able to handle the requirements properly without adding additional program components could be found among existing systems in 2002. Hence, we suggest a data storage solution for their architecture that enables the distribution of data with little available bandwidth and under unreliable network connections.

With CouchDB the architecture of Meissner et al. can be realized from the side of the database. Meissner et al. proposed XML as common data exchange format and set the requirements for the databases as resilient, distributed systems that work under uncertain network conditions with small amounts of data to be transferred. While XML was proposed, JSON is capable of storing structured data as well. The exchange of data between

[2] Nephronophthisis has been described in detail by König and Konrad (2010).

the involved organizations (*Fire HQ* and *Fire Command Post*) can be realized with CouchDB on the database level, which makes it transparent to every application that uses the database. During synchronization of two databases, CouchDB only submits the documents that have been changed. This principle is beneficial for saving network capacity and to allow synchronization over slow connections.

As CouchDB uses HTTP for data transfer, no permanent network connection is kept and the system can exchange data whenever it is possible. Hence, CouchDB takes advantage of automatic configurations mechanisms for ad-hoc networking such as DNS. In contrast to existing relational databases database access can be managed like any other HTTP application. Experiences in the application of disaster management with ad-hoc networks already exists (Asplund, de Lanerolle, Fei, Gautam, Morelli, Nadjm-Tehrani, and Nykvist, 2010). Another amenity is that HTTP allows building high-availability clusters with CouchDB that also allow load-balancing with existing HTTP-load balance solutions. Hence, the operation leaders can setup resilient database clusters and can scale the system easily. Additionally, power outages of the hardware do not endanger CouchDB's resilience since files on the disk remain consistent – assuming that hardware has been configured with care.

The portable information systems worn by the fire fighters in Figure 4 can be also supported. CouchDB qualifies itself for the usage in mobile device such as smartphones or Internet tablets (*pads*), as it has been ported to several mobile platforms. For programmers, this enables unified development of backend and frontend applications. As every crisis has its own requirements regarding the data that needs to be collected, the system allows flexible handling of submission forms. Hence, the headquarter can update data collection on remote information systems and adjust collect data without a full deployment of all updated form definitions. This core feature that has been presented in this paper is essential as setting up and adapting IS in catastrophe settings is time-critical.

DISCUSSION AND LIMITATIONS

Both theoretical considerations and the exemplary implementation of document-based DBs suggest that they are feasible for usage in medical IS in various contexts, including crisis management. In particular, CouchDB as an implementation of this paradigm has proven to be stable. Nevertheless, disadvantages have to be named.

In CouchDB ad-hoc queries are impossible. In order to retrieve information based on the whole document collection process, the system has to generate the view-specific B-Tree data structures. This generation takes time. For applications this property is commonly trouble-free: most of them use a fixed set of queries for inserting, updating and deleting data since all possible operations are known in advance. Hence, cases in which *parameterized SQL queries* would be used in relational scenarios, coverage by CouchDB is good. However, if data is needed outside the context of existing operations and cannot be extracted from pre-created views, it is necessary to wait until CouchDB has collected the data for the view. This e.g. applies to statistical analysis as used to generate reports. This issue can be attenuated by using additional program code that iterates through the database or existing views. Especially scripting languages like Python are a good choice for extracting data from large document sets ("CouchDB Python", 2011). Summing up, a performance and usage benefit can be expected for operations on data structures that can be well-represented as documents and that allow pre-defined operations. Performance penalties have to be expected in some scenarios of dynamic usage. Thus, a decision for a document-oriented database should rely on an examination of expected data structures and kind of usage.

Although documents in CouchDB are stored as JSON entities, it is no database for object storing. Despite some similar concepts and also being attributed to NoSQL systems, *object databases* have some inherent differences to document-based ones ("Databases", 2011). However, the paradigm and its usage of JSON documents make the handling of object easy. The document and the object-oriented paradigm of data storage in modern programming languages are closer related than the object-oriented the relational paradigm. The overhead to store objects in CouchDB is low. Sophisticated algorithms as used in object-relation-mapping (ORM) scenarios are not needed as object data structures can be serialized to JSON.

The realization of peer-to-peer scenarios with distributed CouchDB instances is an interesting mode for the usage in mobile medical applications. This makes it appropriate for home care and in crisis situations. The replication of databases is ideal for the usage of medical information system in disaster settings. Thus, document-based DBs are particularly suited for crisis management, particularly for the ad-hoc deployment of emergency IS.

Considerations about performance and technical suitability are only one aspect of choosing a technology. Acceptance, usability, and developers' skills are also vital. As argued before, relational databases are the natural choice for many programmers. If change is uncommon or even undesired, a paradigmatic shift to a document-based DB will be a poor choice. However, from our experience acquiring proficiency in using a document-based database is possible for a developer used to RDBMS without training. The learning-curve is rather steep if "thinking out of relations" is accepted. In this case, usability is perceived as good and the acceptance is high.

However, although we did not make this experience, we expect the introduction of CouchDB or similar systems to be set for failure if a change in development style is frowned upon.

CONCLUSIONS AND FUTURE WORK

We presented work on document-based databases – in particular on CouchDB – in the context of medical IS, and evaluated their feasibility for crisis management. We introduced the concepts of document-based DBs and CouchDB, and highlighted related work. We then described how CouchDB was used for the development of medical IS. Eventually, we discussed our findings and transferred them to the context of crisis response.

Work on using document-based database in the medical sector is not finished. We hope to encourage further research and expect to see case studies of additional applications to be published. Moreover, a future paper could be based on an artificial scenario that scrutinizes our approach's capabilities. It could use a framework for improvements regarding crisis events such as proposed by Mendonça and Wallace (2004). Several research topics can be derived. Despite the promising findings, evaluation of document-based DBs in medical IS and in crisis response projects has to be intensified. We hope to see case studies as well as quantitative papers to be published. Since experiences with document-based DBs are encouraging, additional concepts such as graph databases (cf. Angles and Gutierrez, 2008) could be checked for their capabilities in the medical context. Furthermore, theorization needs to be driven further. Ideally, future research would develop a decision framework that helps to build more adequate medical IS.

Our work is not finished. Our scenario for using CouchDB will be refined and customized for additional fields of application. We will observe progress and – in combination with insights from discussing the findings published in this paper – refine our approach. Moreover, we consider putting more emphasis on theorization. Thereby, we could become able to provide decision advice. Eventually, we hope to understand which data storage concept is the most adequate in a given situation and to put this knowledge into a framework. Also the suitability of CouchDB for crisis management can be assumed by aligning the requirement of crisis management IS and the features of the database as we have demonstrated above. However, there are no practical experiences in our research group due to the novelty of CouchDB, yet.

ACKNOWLEDGMENTS

The first author of the paper wrote his master thesis in cooperation with the department of medical informatics at the University of Münster, Germany, in the group of Frank Ückert. During his work he developed important parts of Nephreg. Most developments described in this paper are based on work of the department of medical informatics. We would like to thank Jens König for keeping us up-to-date regarding Nephreg.

REFERENCES

1. Anderson, J. C., Lehnardt, J. and Slater, N. (2010) CouchDB: The Definitive Guide, O'Reilly.
2. Anderson, R .J. (1996) A security policy model for clinical information systems, *Proc. IEEE Symposium on Security and Privacy*, 30-43.
3. Angles, R. and Gutierrez, C. (2008) Survey of graph database models, *ACM Comput. Surv,* 40, 1.
4. Asplund, M., de Lanerolle, T., Fei, C., Gautam, P., Morelli, R., Nadjm-Tehrani, S. and Nykvist, G. (2010) Wireless Ad Hoc Dissemination for Search and Rescue, *Proc. 7th Int. ISCRAM Conference*.
5. Bhat, U. and Jadhav, S. (2010) Moving Towards Non-Relational Databases, *IJCA*, 1, 13, 40-47.
6. Brüntrup, R., Lablans, M. and Ückert, F. (2011a) Eine generische Softwarebibliothek zur Web-Umsetzung des Datenschutzkonzepts A des TMF e.V., Retr. Feb. 05, 2012, from http://www.campus.uni-muenster.de/fileadmin/einrichtung/imfl/projekte/DSLib/Generische_Softwarebibliothek_zur_Web-Umsetzung_des_TMF-Datenschutzkonzepts_A.pdf
7. Brüntrup, R., Lablans, M. and Ückert, F. (2011b) Softwareframework zur effizienten Entwicklung medizinischer Dokumentationssysteme und Register, *56. Jahrestagung der GMDS*.
8. Cattell, R. (2011) Scalable SQL and NoSQL data stores, *SIGMOD Rec.*, 39, 4, 12-27.
9. Codd, E. F. (1983) A relational model of data for large shared data banks, *Commun. ACM*, 26, 1, 64-69.
10. Chodorow, K. and Dirolf, M. (2010) MongoDB: The Definitive Guide, O'Reilly.
11. Crockford, D. (2008) Javascript: The good parts, O'Reilly.

12. Cruz, F., Gomes, P. Oliveira, R. and Pereira, J. (2011) Assessing NoSQL Databases for Telecom Applications, *Proc. 13th Conf. on Commerce and Enterprise Computing (CEC '11)*, IEEE CS, 267-270.
13. Dean, J. and Ghemawat, S. (2008) MapReduce: simplified data processing on large clusters, *Commun. ACM*, 51, 1, 107-113.
14. Friedl, F. (2006) Mastering Regular Expressions, O'Reilly.
15. Gao, T. and White, D. (2006) A next generation electronic triage to aid mass casualty emergency medical response, *Proc. Engineering in Medicine and Biology Society (EMBS)*, IEEE, 6501-6504.
16. Hanlon, M. R., Dooley, R. Mock, S. Dahan, M., Nuthulapati, P. and Hurley, P. (2011) Benefits of NoSQL databases for portals & science gateways, Proc. TeraGrid Conference, ACM.
17. Hartz, T., Verst, H. and Ueckert, F. (2009) Kern-PaeP—a web-based pediatric palliative documentation system for home care, *Stud. Health Techno. Inform.*, 150, 337-341.
18. Holt, B. (2011a) Scaling CouchDB, O'Reilly.
19. Holt, B. (2011b) Writing and Querying MapReduce Views in CouchDB, O'Reilly.
20. König, J., Omran, H., Kurlemann, G., Schmitt, O., Lablans, M., Ückert, F. and Konrad. M. (2012) Nephronophthisis Registry, 38. Jahrest. Gesellsch. Neuropädiatrie, Gesellsch. für pädiatrische Nephrologie (GPN).
21. König, J. and Konrad, M. (2010) Nephronophthise, *Medizinische Genetik*, 22, 3, 339-344.
22. Lennon, J. (2009) Beginning CouchDB, Apress.
23. Lerner, R. M. (2010) At the forge: CouchDB views, Linux Journal, 196, August.
24. Lith, A. and Mattsson, J. (2010) Investigating storage solutions for large data, Department of Computer Science and Engineering, Chalmers University of Technology, Göteborg, Sweden.
25. Martin, J. (1983) Managing the Data-base Environment, Prentice-Hall.
26. Meissner, A., Luckenbach, T., Risse, T., Kiste, T., and Kircher, H. (2002) Challenges for an Integrated Disaster Management Communication and Information System, *First IEEE Workshop on DIREN*.
27. Mendonça, D., and Wallace, W. A. (2004) Studying organizationally-situated improvisation in response to extreme events, *Int. Journal of Mass Emergencies and Disasters*, 22, 5-30.
28. Moore, K. (1995) The Lotus Notes™ Storage System, *Proc. of the 1995 ACM SIGMOD international conference on Management of data* (SIGMOD '95).
29. Mohan, C. (1999), "A Database Perspective on Lotus Domino/Notes", Retr. Nov. 11, 2011, from http://www.almaden.ibm.com/u/mohan/domino_sigmod99.pdf
30. Raghupathi, W. and Tan, J. (2002) Strategic IT applications in health care, *Commun. ACM* 45, 12, 56-61.
31. Richardson, L. and Ruby, S. (2007), Restful web services, O'Reilly.
32. Ruflin, N., Burkhart, H., and Rizzotti, S. (2011) Social-data storage-systems, Proc. *Databases and Social Networks (DBSocial '11)*, ACM, 7-12.
33. Russ, T. A., Ramakrishnan, C., Hovy, E. H., Bota, M. and Burns, G.A.P. C. (2011) Knowledge engineering tools for reasoning with scientific observations and interpretations, *BMC Bioinformatics* 12, 351.
34. Schmitt, O., Brüntrup, R., Lablans M. and Ückert, F. (2011) Anwendung dokumentenbasierter Datenbanken in medizinischen Dokumentationssystemen, 56. Jahrestagung der GMDS.
35. Taylor, R. C. (2010) An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics, Proc. 11th Annual Bioinformatics Open Source Conference (BOSC).
36. Wasserman, A. I. and Sherertz, D. D. (1976) A balanced view of MUMPS, *SIGPLAN Not.*, 11, 4, 16-26.
37. Webber, J., Parastatidis, S. and Robinson, I. (2010), Rest in practice. O'Reilly.
38. "Basho.com, Danish Government Licenses Basho's Enterprise NoSQL Data Storage" Retr. Nov. 19, 2011, from http://basho.com/news/danish_government_licenses_riak/
39. "BHOMA" (2011), Retr. Nov. 19, 2011, from http://www.dimagi.com/bhoma/
40. "CouchDB Python Library" (2011) Retr. Nov. 19, 2011, from http://code.google.com/p/couchdb-python/
41. "Databases: relational vs object vs graph vs document" (2011) Retr. Nov. 19, 2011, from http://www.cbsolution.net/ontarget/databases_relational_vs_object_vs
42. "YAML™ Specification Index" (2009) Retr. Nov. 19, 2011, from http://www.yaml.org/spec/