

Foundations and Trends[®] in Databases
Vol. XX, No. XX (2017) 1–108
© 2017 T. Heinis and A. Ailamaki
DOI: 10.1561/XXXXXXXXXX



Data Infrastructure for Medical Research

Thomas Heinis
Imperial College London
t.heinis@imperial.ac.uk

Anastasia Ailamaki
École Polytechnique Fédérale de Lausanne
ailamaki@epfl.ch

Contents

1	Introduction	2
1.1	Medical Research Challenges	3
1.2	Data Management Challenges	5
1.3	Purpose of this Book	8
2	Data Privacy	10
2.1	Legal Context	11
2.2	Anonymization and Deidentification	14
2.3	Generalization & Differential Data Privacy	17
2.4	Secure Data Storage	23
3	Cleaning and Integrating Medical Data	25
3.1	Medical Standards & Ontologies	26
3.2	Data Extraction	29
3.3	Data Cleansing	30
3.4	Schema Mapping, Matching & Integration	34
4	In-Situ Data Analysis	41
4.1	Processing Data In-Situ	42
4.2	Traditional Databases	43
4.3	Accelerated In-Situ Analysis	46

5	Medical Data Processing	49
5.1	Workflow Systems for Medical Research	50
5.2	Medical Data Provenance	58
6	Medical Data Formats and Indexing	65
6.1	Genetic Data	65
6.2	Proteomic Data	69
6.3	Patient Data	73
6.4	Imaging Data	77
6.5	Other Types	80
7	Towards a Medical Data Lake	82
7.1	Centralized Medical Data Repositories and Infrastructures .	82
7.2	Distributed Medical Data Systems	84
8	Summary and Outlook	89
8.1	Current State	90
8.2	Open and Future Challenges	91
	References	95

Abstract

While we are witnessing rapid growth in data across the sciences and in many applications, this growth is particularly remarkable in the medical domain, be it because of higher resolution instruments and diagnostic tools (e.g. MRI), new sources of structured data like activity trackers, the wide-spread use of electronic health records and many others.

The sheer volume of the data is not, however, the only challenge to be faced when using medical data for research. Other crucial challenges include data heterogeneity, data quality, data privacy and so on. In this article, we review solutions addressing these challenges by discussing the current state of the art in the areas of data integration, data cleaning, data privacy, scalable data access and processing in the context of medical data. The techniques and tools we present will give practitioners — computer scientists and medical researchers alike — a starting point to understand the challenges and solutions and ultimately to analyse medical data and gain better and quicker insights.

1

Introduction

Our ability to collect data in the medical sciences is rapidly increasing, which leaves us with an enormous deluge of data that can only be analyzed with difficulty. An example from genomics illustrates the growth very well: next-generation sequencing has led to a rise in the number of human genomes sequenced every year by a factor of 10 Costa [2014], Marx [2013], far outpacing the development in data analysis capacity.

The growth cannot be expected to slow down anytime soon for several reasons. First, sequencing hardware has become so cheap that entire farms of devices are sequencing DNA faster and more precise in parallel. To increase the coverage of DNA sequencing, only more hardware needs to be used. Second, sequencing is only an example. Data today is recorded with more and more devices (e.g., MRI and other medical devices and even with activity trackers) and there are substantial efforts to move medical records to an electronic form to facilitate exchange and analysis. Third, the instruments used are becoming increasingly precise, i.e., have increasingly resolution resulting in a massive growth of the data output Metzker [2010].

The size of the data, however, is not the only challenge of working with medical data. The data produced is becoming more heterogeneous

(originating from different devices, software, manufacturers, etc.) as well as dirtier (incomplete, incorrect, inaccurate, irrelevant records) at the same time and is thus, in dire need of reconciliation, integration, and cleansing.

Efficient integration of the data, however, is pivotal as medical data is moving more and more at the center of medical research to develop novel diagnostic methods and treatments Gluud and Gluud [2005], Shah and Tenenbaum [2012]. As purely symptom-based diagnostic methods are slowly showing their limitations, the shift to data-driven medicine which makes the efficient ability to integrate, process and analyze medical data become key in the new era of data-driven medicine Ohno-Machado [2012].

While recent large-scale medical research initiatives like the Human Brain Project Markram et al. [2011], the BRAIN initiative Insel et al. [2013], the Precision Medicine initiative EA [2015] or the recent BD2K big data in genomics initiative, have helped considerably in addressing some of the big data challenges in medical research and in healthcare, i.e., harmonizing data formats or producing scalable analytics methods, many challenges remain.

1.1 Medical Research Challenges

The traditional approach to diagnosing and treating disease based on symptoms alone is no longer sufficient in the face of increasingly complex disease patterns. Entirely dissimilar diseases with substantially different underlying causes and interactions can exhibit similar symptoms, making their diagnosis and the proper treatment based solely on symptoms very challenging.

Famously, several of the first patients diagnosed with Alzheimer disease by Alois Alzheimer himself — based on symptoms alone — later had to be reclassified as suffering from dementia related to syphilis. Furthermore, diseases that impair cognitive abilities (like dementia, Alzheimer's but also others) are particularly difficult to diagnose: the account of symptoms may be imprecise due to cognitive impairment.

To overcome the limitations of identification and thus, treatment

of disease based on symptoms alone, medical scientists are in the process of developing new methods to understand the precise interactions (pathways and regulatory networks), at different biochemical and biophysical levels, that lead from local malfunction to disease, or how psychological, nutritional, therapeutic, or other environmental factors modulate these interactions.

Given the complexity of diseases, algorithms to process and analyze medical data efficiently and scalable are a crucial component of these new methods: data of different sources and modalities (medical records, genetics, imaging, blood tests, etc.) needs to be integrated and used to discover meaningful correlations within and between etiological, diagnostic, pathogenic, treatment, and prognostic parameters. By finding patients with the same disease into constellations of parameterised biological, anatomical, physiological and clinical variables that define homogeneous populations, signatures of disease (shared attributes of patients suffering from the same disease) can be identified. Disease signatures are crucial to (a) discriminate disease for the purpose of developing more robust and reliable diagnostic methods and (b) for an understanding of the disease itself (with the potential to develop new treatments).

On this quest to data-centric medical research moving beyond classic symptom-based diagnostics and treatments, medicine is facing major challenges.

First, many challenges related to heterogeneous and dirty data are not directly related to the underlying technology. The quality of the data captured directly depends on how well the acquisition process is designed. A key aspect of using medical data, therefore, is the design of the capturing process as well as the training of the medical professionals involved in the process. Only by using controlled vocabulary early in the data acquisition process at hospitals and in research studies can the quality of the data be ensured on a reasonable level. Solely if the data is initially collected at a reasonable quality, can data cleansing compensate for errors and omissions?

Second, whilst the fundamental analytics algorithms may not be developed by medical professionals, the key will be their use in building

the models of disease. Albeit there is certain expectation that through the analysis of data alone insights can be gained, fundamental domain knowledge will always be needed. It is, therefore, key that on the medical side of the problem the models are developed that are not only informed by data but also by an understanding of the domain, i.e., models that clearly understand and connect cause and effect (even if the connection has first to be established through the use of data). Further, understanding the strengths and weaknesses of different analysis approaches by medical researchers, as well as designing combinations for an improved analysis is important in general, but particularly in the context of medical data.

1.2 Data Management Challenges

The major data management challenges revolve around the heterogeneity and quality of the data as well as data privacy. Heterogeneity of the data incurs considerable overhead for integration and combination of data but also complexity for design access methods and indexes for a plethora of different file formats. Data privacy, on the other hand, requires the careful adherence to ethical guidelines and laws but, as we will discuss, also makes architectural considerations necessary as not all data can be made accessible freely and can be moved between systems, sites, and countries.

A fundamental challenge with designing infrastructure for the processing, storage, and analysis of medical data is the heterogeneity of the data. On a practical level, this means that a plethora of tools needs to be used to access, analyze and visualize the different data formats, leading to a complex software integration challenge.

On a more technical level, however, this also means that data completely different data formats need to be organized for efficient and scalable access. Even if the data can be reorganized — it may not be possible for various reasons — different modalities need a different organization. Some of the data is of very high dimension; some are spatial, some — as is frequent in medical studies — is longitudinal while other data are only one dimensional. With different dimensionality also come

different access patterns or types of queries which then again require a different organization of the data on disk or on the system. Extensive research has been done for most different types of data, i.e., different dimensionality and solutions are readily available. Accessing the different types of combined queries, however, requires different data layouts on disk as well. More precisely, as different types of analyses require different data layouts on disk, mixing queries and their predicates also requires trade-offs, i.e., mixed layouts on disk. The fact that the organization of data of different dimensionality is fundamentally different makes finding an ideal organization of the data virtually unattainable.

Additional complexity is introduced to accelerate analysis of different types of data through auxiliary data structures like indexes. Multitudes of different indexes exist to accelerate the queries/analyses on different types of medical data. Still, using them can become challenging, particularly if queries and analyses use different predicates, making the use of different indexes and auxiliary structures necessary to accelerate the analysis. As with the fundamental problem of organizing the data layout, once predicates are combined, the underlying data layout is mixed and optimal solutions cannot be found, and one needs to settle for trade-offs. Indexes for different types of data and particularly dimensionality have different selectivity and different efficiency, thereby also raising the question how, i.e., in what order, to use them to accelerate the queries.

A substantial share of medical data originates from processes where humans are heavily involved in capturing it (e.g., electronic health records). Lack of quality control and enforcement of correct values quickly leads to considerable errors to the point where the data may be impossible to use. Even if not strictly acquired through human intervention, misconfiguration of devices (e.g., wrong units, missing or faulty calibration and others) can lead to erroneous data. Although faulty, a lot of the data can still be salvaged. Solely through checking constraints on datatypes, e.g., values in a particular column must be of a particular datatype, on ranges, e.g., typically, numbers or dates should fall within a certain range (e.g., body temperature), etc. can faulty data be identified and flagged accordingly. Knowing a particular

value is not correct in itself is immensely important for analysis. In some cases, the values may even be corrected.

Quality, however, is only one aspect. Data originating from different sources is bound to have discrepancies. This can be due to different naming conventions used or because of imprecise descriptions. Even just using different descriptions for the symptoms or diagnoses can render data virtually unusable as queries attempting to find data for a particular illness may not find all occurrences. The problem is to some degree addressed by using structured *controlled vocabulary*, i.e., ontologies, to describe symptoms, diagnoses and phenomena in general. There are, however, different ontologies and vocabularies available as well and so mappings must be defined between them to use datasets structured and named after one and the other together, i.e., to integrate them, so they will be consistent with other similar data sets.

Data privacy has two fundamental sides to it. First, clearly available guidelines, standards, and laws need to be followed. In the simplest case, only data from one source or organization is involved, and local laws and rules need to be followed, e.g., certain information in the data needs to be removed, areas in images need to be blacked out, and our personal information needs to be perturbed.

Privacy, however, becomes substantially more challenging when different organizations and data providers are involved, even in the same country as different ethics boards and committees may set different requirements. Moving data between countries, i.e., building infrastructure to analyze data across countries again makes the problem more challenging as in some instances no data can leave the country — a limitation that needs to be factored in when designing infrastructure.

Generally, medical data to be processed and treated on such infrastructure is ideally already completely anonymized or de-identified, i.e., all personal identifiers and means of identification removed from it in accordance with the ethics committee of the data provider, before it is used by any infrastructure.

Privacy, however, has significantly more far-reaching consequences that not only affect how the data items are treated, but the entire infrastructure needed to process medical data. Even if completely de-

identified or anonymized, medical data frequently is not allowed to be moved outside the country, the organization or even just the system where it resides. The architecture of any data infrastructure needs to be designed accordingly. For example, if data must be kept in the system where it resides, all data access must be designed to enable to work efficiently on the physical layout of the system. All indexes and auxiliary data structures need to be optimized for a physical layout that cannot be influenced in any particular way. If on the other hand, the data cannot be moved outside a country as a whole, the architecture needs to be designed such that samples, anonymized subsets; aggregates can still be exchanged (taking into account the ethical guidelines, etc.) on a level which still serves the overarching analytical goals.

1.3 Purpose of this Book

The purpose of this book is to serve as a starting point for medical and computer science professionals designing and understanding data infrastructure for medical research.

The book focuses primarily on the data management and computer aspects of the problem. As has been briefly touched upon before, developing and deploying research infrastructure for medical data also involves technical and organizational challenges for the medical science and not only for data management. Still, we focus primarily on the data management aspects. Nevertheless, the book is intended for both audiences, technical/computer scientists in the process of designing infrastructure and systems for the analysis of medical data as well as a medical professional who want to understand the challenges involved.

The book touches on the most common and most challenging topics in the context of developing infrastructure storing, processing and analyzing medical data. It surveys state-of-the-art methods, current research efforts of the most important topics around the treatment of medical data as well as the most relevant up and coming research efforts in the context of medical data.

The topics, as well as the discussion of each topic, are by no means exhaustive. Data infrastructures designed for medical research is a hot

research topic and a moving target. New hardware, new instruments, new analysis techniques and, probably most importantly, upcoming privacy standards and legislation shape and influence the field considerably. The book thus needs to be understood as the initial point needed to study the state-of-the-art for the purpose of understanding the challenges involved, providing a comprehensive background needed and serving as a starting point for further readings on the subject.

2

Data Privacy

Medical data as such is inherently personal — hardly any other piece of information can reveal more personal information about us. Yet, we willingly give this vital information up in exchange for medical treatments. The problem of balancing the interest in the protection of personal information and using the data for the greater good of society — for making medical data available for research potentially leading to new treatments — has been understood by society and lawmakers alike. It is no surprise that legislation protecting medical data has been passed or is in preparation in several countries.

In the following chapter, we discuss the legal context and the mandated necessity for de-identification and anonymization of medical data as well as the available methods to do so. As we will see, there is an inherent trade-off between data utility and data anonymization — the more a piece of information is detached from an individual, the less information it contains and thus, the less useful it is for research. Furthermore, de-identification and anonymization can have a considerable impact on the architecture of a medical information system as we will discuss.

2.1 Legal Context

The importance of protecting private information has been well understood for years. Only recently, however, did medical researchers turn their attention to mining these masses of data to improve diagnostic techniques as well as treatments. Ever more data being collected in general and in medical treatments, in particular, has enabled a true revolution moving from symptom-based to evidence-based medicine based on data mining. Increased interest and pressure from medical research to make data available is met with new ethical guidelines as well as legislation.

The key to working with medical data or, more precisely, with patient data is to understand the legal and ethical context. To be able to use and work with medical data, legal and ethical standards need to be met. Ethical guidelines typically depend on the institution or hospital where the data is collected and stored. They are very specific to the institution — or more specifically to the ethics board — and there is little point in attempting to cover them in this book. It is important to know, however, that ethical guidelines are not a superset of the legal requirements: from complying with ethics, one cannot assume to be automatically on safe, legal grounds.

The legal situation, on the other hand, depends on the country. In the US, the requirements are very well defined with HIPAA Rothstein [2013]. HIPAA, the Health Insurance Portability and Accountability Act Title II (Title I addresses health insurance cover not relevant in this context) defines national standards for electronic health care transactions and national identifiers for providers, health insurance plans, and employers.

Crucial for the use of medical data in research is HIPAA's privacy rule which specifies precisely what data needs to be removed from the dataset for it to be de-identified and for the medical researcher to be legally safe. What needs to be removed is a rather small set of clearly defined pieces of data: names, geographic subdivisions smaller than states, dates (except years) that are directly related to a subject, telephone/fax numbers, serial numbers, E-mail addresses, social security numbers, Internet protocol addresses, full face photographs, biomet-

ric identifiers and a limited number of similar unique identifiers of a subject.

The provisions of HIPAA's privacy rule are rather well defined and do not unduly limit research on the data. Only removing fields like the exact geographic location, day and month from dates specifically relating to people will limit research somewhat. Still, the anonymization, or rather de-identification defined with HIPAA allows for broad research and is clearly defined, providing researchers with easy to follow rules for a legal, safe harbor.

HIPAA is not undisputed and draws heavy criticism for the cost of compliance Kilbridge [2003], Nass et al. [2009], Withrow [2010]. Indeed, to comply with both titles of it, healthcare providers require substantial investment. For example, essentially all employees dealing with health records of healthcare providers must be trained in the protection of patients' privacy Wolf and Bennett [2006]. The training focuses on practices for handling protected health information: for example, avoiding discussing patients within hearing distance of others, locating fax machines and printers in secure areas, and obtaining only the information about a patient that is required. To ease implementation of HIPAA for smaller healthcare providers, commercial, cloud-based platforms have been developed Datica, Aptible.

While HIPAA may indeed incur considerable cost for healthcare providers, this does not apply to medical research. Typically, to allow for the use of patient data in research, the data needs to be exported from hospital systems once at which time it can be made HIPAA compliant by stripping identifiers/fields. Specifying what fields and identifiers need to be removed is a manual process, but subsequently, the export can be performed automated.

The recently introduced HITECH Act (Health Information Technology for Economic and Clinical Health Act) K [2010] has greatly simplified adoption of electronic health records as well as HIPAA. The HITECH Act initially provided a stimulus and subsidies for health care providers and insurances to accelerate adoption of electronic healthcare records in the US. The subsidies have considerably helped in offsetting the additional cost of compliance with HIPAA and have therefore in-

deed helped to increase the use of electronic health records. As of 2015, HITECH uses financial penalties to further encourage adoption.

The Data Protection Directive of the European Union is an important component of the EU privacy and human rights law. It has been controversially discussed for a while, and several different proposals, each with different implications have been put forward. Particularly, while the text proposed by the EU commission made broad exemptions for research on the issue of privacy, the European parliament removed many exemptions again Castro-Edwards [2013].

A crucial question for the use of the Data Protection Directive is *consent* of the patient/subject. The initial proposal by the Commission foresaw a complete exemption for research to require the explicit consent of subjects (which is very difficult to obtain in practice unless in the context of a well-defined medical study). The parliament, on the other hand, removed all such provisions, such that the use of personal data in research without specific consent would be prohibited or become impossible in practice, despite the fact that this research is subject to ethical approval and strict confidentiality safeguards, and that the identity of individuals is often masked.

Ultimately in the final negotiations, the Regulation follows the approach of the current Directive and includes a special set of rules for research. This waives some regulatory requirements in recognition of the benefits that research offers to society. For example, the rules facilitate the re-use of data for research, even where the data was collected for another purpose. This provision relaxes the need for explicit consent (it cannot be given if the data is collected in a different context) which opens up masses of already collected patient data in hospitals for use in medical research.

With the issue of consent addressed, the remaining question is how data has to be treated/anonymized so it can be used in medical research. The approach followed is to use the concept of pseudonymization as defined by International Standardization Organization (ISO) published Technical Specification (TS) No. 25237 which defines the basic concept of pseudonymization and includes technical and organizational aspects of de-identification as well as guidelines for re-

identification risk assessment. Pseudonymization means transposing identifiers (like names and date of birth etc.) into a new designation, preferably by encryption, so that the recipient of the information cannot identify the data subject.

In the context of the EU's legislation, it is, however, crucial to understand that EU-wide legislation defines a minimum standard from which countries can deviate. Particularly in the case of the European Data Protection, a Directive agreement was difficult to reach, and so the legislators agreed that for many aspects member states would be left to work out the details for themselves. Research is one such area, and within limits, the regulation allows member states to develop their own system of safeguards and exemptions from subject data rights for research. This flexibility creates opportunities for member states to adapt the rules to fit their existing arrangements and make them more relevant to their own society and culture. On the flip side, this makes it very hard to develop data processing infrastructure for medical data that works across countries.

2.2 Anonymization and Deidentification

It is common misbelief that anonymization is already accomplished by decoupling a subject's identity from the dataset or, in this case, from the electronic health record Fung et al. [2007]. Indeed, complete anonymization as based on its definition is only achieved, if no inference can be made from the anonymized health record, i.e., if the anonymized record does not allow anyone to narrow down the potential set of subjects/patients it relates to. Releasing any type of information about the health record will always allow ruling out even just a few patients, thereby narrowing down the potential subjects. Put more formally, if the release of the statistic S makes it possible to determine the value more accurately than without access to S , disclosure has taken place Mohammed et al. [2009a]. If the statistical summary, or the attributes, released do not allow narrowing down potential subjects, it is questionable if the attributes have any value to begin with. Even just knowing minimal attributes about the subject like, for example, the

geographic location allows ruling out some subjects.

Complete anonymization of the data is thus unattainable unless all information is removed which of course renders the data useless. There clearly is a direct relationship between the utility of the data as well as anonymization. Only datasets which have not been anonymized and thus contain all information have maximum utility while completely anonymized datasets contain no information and thus have no utility. Striking the right balance to enable medical research while protecting the personal data of patients/subjects, is therefore key.

Besides complete anonymization, a further concept, the one of de-identification, has been proposed Gellman [2010]. De-identification of data refers to the process of removing or obscuring personally identifiable information from individual records to minimize the risk of unintended disclosure of the identity of individuals and information about them Gellman [2010]. Specific steps and methods used to de-identify information may vary depending on the requirements but also on the data at hand (format, semantics, etc.), but should be appropriate to protect the confidentiality of the individuals. While it may not be possible to remove the disclosure risk completely, de-identification is successful when there is no reasonable basis to believe that the remaining information in the records can be used to identify a subject McGraw [2010].

An important aspect of medical research, particularly when it is used as the basis for novel treatment or diagnostic tools, is reproducibility or provenance to track back how the results were obtained (as well as for incidental findings). De-identification, in this case, is problematic as it may sever the connection between data and identity of the subject.

A further concept, pseudonymization has thus been developed Neubauer and Heurix [2011]. Pseudonymization is a procedure by which the most identifying fields within a data record are replaced by one or more artificial identifiers or pseudonyms. There can be a single pseudonym for a collection of replaced fields or a pseudonym per replaced field. The purpose is to render the data record less identifying and therefore lower customer or patient objections to its use. Data in

this form is suitable for extensive analytics and processing.

The pseudonym allows tracking back of data to its origins, which distinguishes pseudonymization from anonymization, where all person-related data that could allow backtracking has been purged.

One approach is to simply replace personal information with surrogates that can later be used to look up the real values (for example, create keys as found with randomization or blinded studies). Alternatively, one can simply drop the columns (descriptions used in clinical trials) or recode the variables (age or age range instead of date of birth).

The application of pseudonymization to medical data intends to preserve the patient's privacy and data confidentiality. It allows the use of medical records by authorized medical researchers. However, plain pseudonymization for privacy preservation often reaches its limits when genetic data is involved. Due to the identifying nature of genetic data, depersonalization is often not sufficient to hide the corresponding person. Potential solutions are the combination of pseudonymization with fragmentation and encryption. Even in this case, however, it is very likely inferences about the patient can be made.

The choice of which data fields are to be pseudonymized is partly subjective but should include all fields that are highly selective. Less selective fields, such as date of birth, postcodes are often also included because they are usually available from other sources and therefore make a record easier to identify. Pseudonymizing these less identifying fields removes most of their analytic value and should, therefore, be accompanied by the introduction of new derived and less identifying forms (similar to the generalization approach in k-anonymity).

Data fields that are less identifying, such as date of attendance, are usually not pseudonymized. It is important to realize that this is because too much statistical utility is lost in doing so, not because the data cannot be identified. For example, given prior knowledge of a few attendance dates, it is easy to identify someone's data in a pseudonymized dataset by selecting only those people with that pattern of dates.

2.3 Generalization & Differential Data Privacy

Based on the understanding that data can never completely be anonymized or de-identified — particularly when using genetic information but also based on much more basic information — without completely losing its utility, several approaches to partially anonymize data while providing statistical guarantees have been developed.

In the following we discuss two approaches to anonymization: generalization and differential privacy. Generalization lends itself particularly well when data sets are released (privacy-preserving publishing of data). Differential privacy, on the other hand, was developed only recently and is considered the state of the art in the research community. Still, generalization is broadly used for data releases. Differential privacy, however, is particularly useful in cases where the data resides with the organisation collecting it (e.g., a hospital) and a set of pre-defined queries can be executed remotely with only the result returned.

The research community working on data privacy shifted its focus from generalization to differential privacy in recent years. The major drawback of generalization is the absence of a clearly defined mathematical notion of information disclosure and it is thus challenging to quantify how much information is disclosed with a release of the anonymized data. Differential privacy overcomes this issue with a sound definition of information disclosure. While the research community has shifted focus to differential privacy, generalization is still broadly used in practice for its simplicity and because it is typically deemed by ethical committees to provide sufficient anonymization for data releases. Consequently, we discuss both approaches.

2.3.1 Generalization

A fundamental approach is to generalize some of the attributes of patients to make it more difficult to identify individuals. A first approach is k -anonymity which in itself does not define an approach or algorithm but rather defines a property of anonymized data Sweeney [2002]. Anonymized patient data is considered k -anonymous if statistical guarantees can be given that the individuals who are the subjects

of the data cannot be re-identified while the data remains useful in practice. More precisely, anonymized patient data is said to have the k -anonymity property if the information for each person contained in the release cannot be distinguished from at least $k-1$ individuals whose information also appears in the dataset.

k -anonymity can generally be provided through two mechanisms, suppression and generalization. Using suppression, certain values of the attributes are dropped, so the entire record becomes less distinguishable. All or some values of a column may be dropped. For example, attributes like name, religion or others that are very specific to a particular subject are dropped. With generalization, individual values of attributes are replaced with a broader category. For example, instead of specifying a precise age, an age bandwidth is given. Generalization is particularly useful if by dropping an attribute not too much information is lost. Figure 2.1 shows an example of a table with several of the attributes generalized (zipcode and age). In this table, at least three individuals have the same combination of the generalized attributes, i.e., $k = 3$.

A general problem of k -anonymity is that sensitive attributes may be homogeneous. The example in Figure 2.1 illustrates this: although we do not know which tuple a patient with zipcode 47645 and age 29 refers to — it could be any of the three — we still can infer that the patient suffers from heart disease.

l -diversity [Machanavajjhala et al., 2006] attempts to tackle this issue of an homogeneity attack. It essentially ensures that sensitive fields or attributes have diversity, i.e., guarantees are made that sensitive attributes

Zipcode	Age	Disease
476**	2*	Heart Disease
476**	2*	Heart Disease
476**	2*	Heart Disease
4790*	≥ 40	Flu
4790*	≥ 40	Heart Disease
4790*	≥ 40	Cancer
476**	3*	Heart Disease
476**	3*	Cancer
476**	3*	Cancer

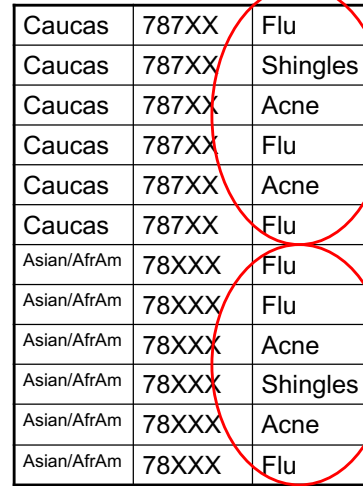
Figure 2.1: Example of a table providing 3-anonymity.

take a minimum number of different values.

More precisely, l -diversity is a form of group-based anonymization that is used to preserve privacy in data sets by reducing the granularity of a data representation. This reduction is a trade-off that results in some loss of information (or utility) in order to maintain some privacy. The l -diversity model addresses some of the weaknesses in the k -anonymity model where protected identities to the level of k -individuals is not equivalent to protecting the corresponding sensitive values that were generalized or suppressed, especially when the sensitive values within a group exhibit homogeneity (i.e., they are virtually all the same and thus easy to guess). The l -diversity model adds the promotion of diversity within a group for sensitive values in the anonymization mechanism. Figure 2.2 shows an example where the sensitive attribute (disease) is diverse.

Three approaches to make a dataset l -diverse have been proposed so far. Distinct l -diversity is the simplest approach which ensures that at least l distinct values for the sensitive field in a group. Entropy l -diversity is defined such that the entropy of a group is required to be bigger than $\log(l)$. Recursive $(c-l)$ -diversity, on the other hand, simplifies the definition so as to ensure that the most common value does not appear too often while less common values are ensured not to appear too infrequently, thereby ensuring that the frequency of a value's attribute does not reveal too much information.

t -closeness is a further refinement of l -diversity group-based



Caucas	787XX	Flu
Caucas	787XX	Shingles
Caucas	787XX	Acne
Caucas	787XX	Flu
Caucas	787XX	Acne
Caucas	787XX	Flu
Asian/AfrAm	78XXX	Flu
Asian/AfrAm	78XXX	Flu
Asian/AfrAm	78XXX	Acne
Asian/AfrAm	78XXX	Shingles
Asian/AfrAm	78XXX	Acne
Asian/AfrAm	78XXX	Flu

Figure 2.2: An example table providing l -diversity: sensitive attributes are ensured not to be homogeneous.

anonymization which additionally uses k-anonymity’s idea of reducing the granularity of the data representation Chaytor and Wang [2010]. It is a trade-off that results in a controlled loss of utility of the data while statistically guaranteeing additional privacy. The t-closeness model extends the l-diversity model through treating the values of an attribute distinctly by taking into account the distribution of data values for this attribute.

t-closeness is particularly useful in real datasets where attribute values may be skewed or semantically similar. Taking into account value distributions, however, may cause difficulty in creating feasible l-diverse representations. The l-diversity technique is useful as it impedes an attacker exploiting the global distribution of an attribute’s data values in order to infer information about sensitive data values. l-diversity may consequently be difficult and unnecessary to achieve when protecting against attribute disclosure Evfimievski et al. [2003].

Data perturbation Kargupta et al. [2003], as well as a generalization as discussed before, has been successfully used to mask or anonymize electronic health records. It has been hailed as one of the most effective data protection techniques, whilst being relatively simple to implement. In an example from neuroscience, medical data is anonymized on two levels best explained with the two perturbation methods of k-anonymity. First, sensitive attributes (broadly in accordance with HIPAA) are suppressed or dropped from the data, and second, the data is generalized through aggregation, i.e., single values are replaced by aggregate values Venetis et al. [2015], to anonymize data (in accordance with ethics committees).

The main threat to data published based on generalization, however, is still the availability of background knowledge (or the composition attack Ganta et al. [2008]). Using additional data sources (or potentially the same data source but with different attributes generalized), individuals in the anonymized dataset may still be de-identified, despite the use of more advanced generalization techniques like t-closeness. Preventing composition attacks is challenging as it requires different providers of even remotely related datasets (e.g., hospital discharge dataset and voter registry) to collaborate and agree on what

attributes to generalize how. What makes avoiding this attack particularly hard, is that it is difficult for an organization wishing to release data to know (a) which datasets could be combined with their data as they may seemingly be unrelated, (b) which other (potentially related) datasets are available and (c) which datasets will be available in the future. Generalization, however, still is the the most broadly used method for anonymising medical data prior to release for it is simple and is typically deemed sufficient by ethical committees governing data release.

2.3.2 Differential Privacy

Given the major challenge of generalization — the absence of a well-defined model of information disclosure, making it difficult to quantify how much information has been disclosed — has lead to the development of the concept of differential privacy Dwork [2008], Gehrke [2010]. Differential privacy covers different techniques to anonymize data whilst providing statistical guarantees and essentially is a measure which, intuitively, captures the increased risk to privacy incurred by a tuple participating in a database. Put differently, given a database, differential privacy is the risk to privacy, i.e., the information learned, by asking aggregate queries on different subsets of the database.

For example, an adversary can execute two aggregate queries on database D . The adversary may engineer the queries such that one is executed on $D_1 \subset D$ and the other on $D_2 \subset D$ where D_1 and D_2 only differ in one tuple t , e.g., $D_1 + t = D_2$. When comparing the results of both queries, the adversary can learn information about t . If the queries aggregate over a numeric attribute, the value of t 's attribute can be inferred. Even in case of different types of attributes, however, the adversary can learn statistical properties of t 's attribute (and can learn more through repeated queries).

The information inferred depends on the dataset, the aggregate queries and the number of queries executed. The more the result of a query varies, depending on the participating tuples in the query execution, the more sensitive the query is, and the more information about the dataset can be learned. Also, the more queries are executed, the

more can be learned about a database and the individual tuples.

Differential privacy mechanisms aim to reduce the information learned about individuals (or individual tuples) while still providing useful information. They do so by adding controlled noise to query results with low sensitivity Dwork [2008], Dwork et al. [2006].

Intuitively, this means that for any two datasets that are similar, a given differentially private algorithm will behave approximately the same on both datasets. The definition gives a strong guarantee that presence or absence of an individual tuple will not affect the final output of the algorithm significantly. Put differently, the differential privacy algorithm will add noise to the results such that only limited information about an individual tuple can be learned.

More formally, an algorithm A provides ϵ -differential privacy for datasets D_1 and D_2 that differ on a single tuple if $\Pr[A(D_1) = x] \leq e^\epsilon \times \Pr[A(D_2) = x]$ for all x . This definition gives the guarantee that presence or absence of a single tuple does not affect the final output of the algorithm significantly. To provide ϵ -differential private answers, A needs to add statistically bounded random noise to the results. The choice of differential privacy mechanism, i.e., the noise added to the query result, depends on the query or the aggregation function used Kotsogiannis et al. [2017]. Multiple differential privacy approaches have been developed for specific applications like recommender systems McSherry and Mironov, mining frequent patterns Bhaskar et al. [2010], clustering Feldman et al. [2009], Rastogi and Nath [2010] and others.

The precise method used to add noise to the query result directly depends on the query asked. In scenarios where the queries asked are not known a priori, for example, in case where datasets are released, differential privacy is therefore to use. Indeed, whether differential privacy mechanisms can be used for releasing data is a subject of ongoing research Mohammed et al. [2009b]. The most broadly used approach to anonymization of released medical data to date in practice consequently still is generalization, partly due to its simple implementation and due to its acceptance by ethical committees.

A further challenge is the choice of the parameter ϵ . While for

anonymization based on generalization, the impact of generalizing attributes can easily be calculated (e.g., in case of k -anonymity, the k can be determined), it is much harder to understand the impact of the choice of ϵ . ϵ does not limit what is disclosed about an individual tuple but instead limits the impact of a tuple on the result. Setting ϵ thus is a challenge.

2.4 Secure Data Storage

While querying the data without releasing personal information is vital, storing it securely is crucial as well. Recent developments have been proposed to store the data encrypted and indeed to execute queries on the encrypted data. CryptDB Popa et al. [2011], for example, stores data encrypted using efficient SQL-aware encryption schemes and can then answer SQL queries directly on the encrypted data without ever decrypting the data. CryptDB has recently been successfully used for storing and querying electronic health records Shahzad et al. [2015]. Executing queries on the encrypted data did not incur an undue performance penalty. Figure 2.3 illustrates how encrypted queries are passed to the database which executes them on encrypted data. The encrypted results are sent back to the application without query, data or result ever being decrypted outside the application.

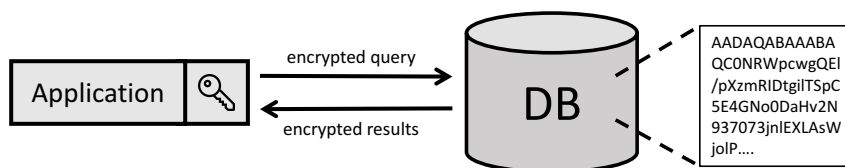


Figure 2.3: CryptDB never decrypts query, data or results outside the application but only works on encrypted data.

The ideas of CryptDB have led to a number of similar research projects focusing on executing queries on encrypted data, e.g., TrustedDB Bajaj and Sion [2014], Cipherbase Arasu et al. [2013] and others. Many of these projects are driven by the need to outsource and

move large amounts of data into the cloud. More importantly, however, the concepts of CryptDB have been integrated into multiple commercial products like Microsoft's *Always Encrypted SQL Server*, SAP's *SEED* or Google's *Encrypted BigQuery* which can easily be used for production deployments.

What helps adoption of these systems are new CPUs with specific instruction sets like the Intel Advanced Encryption Standard New Instructions Gueron [2010] to efficiently support encryption and decryption in hardware. Similarly, Intel Software Guarded Extensions (and similar proposals), is a set of new instructions from Intel that allows user-level code to allocate private regions of memory. These regions are inaccessible by other processes and can thus be used for processing medical data securely.

3

Cleaning and Integrating Medical Data

Medical data almost always originates from diverse sources, be it different instruments or systems used for its capture. Integrating the data is consequently challenging for two different reasons. First, the data almost always contains errors that need to be corrected before it can be used. Errors may be missing values in manual or automatic capture or erroneous values due to malfunction of instruments, faulty calibration or due to other reasons. No matter the reason for the faulty or missing values, they need to be fixed wherever possible before the data can be used. Possible fixes include to exclude faulty values, fix values (e.g., in the case of calibration errors, some errors can be fixed a posteriori) and so on. Whatever approach is used, the provenance of the changing of the values needs to be tracked.

The second challenge is the vocabulary used to describe events and observations. When combining data from different studies, the same observation may be described differently, making integration challenging. To address the issue, either standardized controlled vocabularies need to be used when designing a medical study and when capturing the data or mappings between the vocabularies of different studies need to be defined.

3.1 Medical Standards & Ontologies

LOINC (Logical Observation Identifiers Names and Codes) is a database and universal standard for labeling medical laboratory observations and is, therefore, crucial to name observations in medical studies Dugas et al. [2009], McDonald et al. [2003]. LOINC applies universal code names and identifiers to medical terminology related to electronic health records. The purpose is to assist in the electronic exchange and integrating of clinical results. Several standards, such as HL7, are based on LOINC to enable electronic transfer of results between different health care systems. LOINC has two main parts: laboratory LOINC and clinical LOINC (clinical reports and documents).

Mapping each entity-specific code to its corresponding universal code can represent a significant investment of both human and financial capital. Using a universal code system, a priori enable the integration of data from different studies instantly, without major overheads.

LOINC essentially defines the vocabulary needed to describe observations and events in medical studies. It uses a formal, distinct and unique 6-part name that is given to each term to test or identify an observation. LOINC currently defines 71,000 observation terms that can be accessed and understood universally. Each database record includes six fields for the unique specification of each identified single test, observation, or measurement. It defines what is measured or observed, the characteristics of the measurement, the time of measurement, the instruments used, the scale and unit of the measurement as well as the method used.

A unique code is assigned to each entry upon registration. Other database fields include status and mapping information for database change management and provenance tracking, synonyms, related terms, substance information (e.g. molar mass), choices of answers for nominal scales and translations.

LOINC covers naming and terminology in medical studies broadly with an increasing international adoption which improves exchange and integration between different countries. A number of efforts have been undertaken to translate the LOINC documents and terms into various languages, such as Simplified Chinese, German, Spanish and others

Whilst LOINC primarily covers controlled vocabulary for medical studies, i.e., to name observations (description, measurement, unit and so on), it does not cover diagnostic classification. For the purpose of uniquely identifying and classifying disease, the International Statistical Classification of Diseases and Related Health Problems (ICD) was developed World Health Organization [1992].

ICD-10 is the current version of the ICD developed by the World Health Organization (WHO). It encompasses codes for diseases, signs and symptoms, abnormal findings, complaints, social circumstances, and external causes of injury or diseases.

The code set allows more than 14,400 different codes and permits the tracking of many new diagnoses. The codes can be expanded to over 16,000 codes by using sub-classifications.

ICD-10 classifies diseases and health related problems into a hierarchy. On the top level, it groups disease into 22 groups ranging from parasitic diseases, neoplasms to diseases of the circulatory system, the respiratory system and so on. Each group (or block) is further structured on multiple levels until a disease can unequivocally be identified.

ICD-10 also has several national modifications that frequently include much more detail, and sometimes have separate sections for procedures (e.g., the US ICD-10 Clinical Modification, for instance, has 68,000 additional codes). National modifications, as well as slow adoption of different revisions in different countries and organizations, make integration difficult as studies may be using different revisions. However, WHO publishes the differences of consecutive revisions so studies in different revisions can be mapped to each other without any gaps.

The Systematized Nomenclature of Medicine (SNOMED) broadens the scope and is a systematic collection of medical terms, in human and veterinary medicine spanning anatomy, diseases, findings, procedures, microorganisms, substances, etc. SNOMED Lee et al. [2013] can be used to describe all medical documents, be it in the context of medical studies or for electronic health records. It allows a consistent way to index, store, retrieve, and aggregate medical data across specialties and sites of care. SNOMED was started in the United States by the College of American Pathologists in the 1970s and has become an

internationally accepted standard.

SNOMED was designed from its inception with complex concepts defined in terms of simpler ones. For example, a disease can be defined in terms of its abnormal anatomy, abnormal functions, and morphology. In some cases, the etiology of the disease is known and can be attributed to an infectious agent, a physical trauma or a chemical or pharmaceutical agent.

Initial versions of SNOMED used a multiaxial, hierarchical classification system where a disease was first located in the anatomy which results in a code in the topography axis and may lead to morphological alterations represented by a morphology code. More recent versions, i.e., SNOMED CT, use a subtype hierarchy, supported by defining relationships based on description logic.

The current design of SNOMED CT centers around the *concept* and uses four primary core components: concept codes (numeric code to identify clinical terms), their descriptions, their relationships (between related concept codes) as well as reference sets used to group concepts and descriptions. *Concepts* are units that categorize all the things that characterize health care processes and need to be recorded. The latest SNOMED CT revision included 311'000 concepts. All SNOMED CT concepts are organized into *acyclic is-a* hierarchies. For example, viral pneumonia is-a infectious pneumonia which is-a pneumonia and so on until one arrives at a top-level concept. The taxonomic structure allows data to be recorded and later accessed at different levels of aggregation.

Both, SNOMED CT as well as ICD-10, use standardized definitions and form a common medical language used within electronic health record (EHR) systems. SNOMED CT enables information input into an EHR system during the course of patient care, while ICD facilitates information retrieval, or output, for secondary data purposes.

To support communication about clinical drugs and supporting interoperability between drug vocabularies, the National Library of Medicine has developed RxNorm Nelson et al. [2011]. RxNorm is released on a monthly basis and entails normalized names for clinical drugs on different levels of abstraction. It can be understood as ontology of the drugs in the US market.

The Unified Medical Language System (UMLS Bodenreider [2004]) is an attempt by the National Institutes of Health in integrating the breadth of ontologies and controlled vocabularies available in the context of medicine. It integrates the aforementioned standards like ICD-10, SNOMED, LOINC, and RxNorm but also others like Gene Ontology and others.

An excellent example of the use of multiple ontologies and standards is their integration into the common data model of PCORNet Fleurence et al. [2014]. PCORNet integrates electronic health records of multiple healthcare providers and medical research facilities for the purpose of research.

3.2 Data Extraction

Whilst medical data frequently is well-structured because it is collected using well-defined protocols and graphical user interfaces, there also is a lot of unstructured or semi-structured data available. Two prominent examples are doctor's notes which typically follow little form as well as medical publications (which can contain classification information and can thus be understood as semi-structured). Considerable research Sarawagi [2008] has been pursued in the automated information extraction for the purpose of name entity recognition, relationship extraction, terminology extraction and others from general, unstructured documents. An increasing number of techniques and approaches are based on natural language processing to understand relation and semantics between terms used Califf and Mooney [1999], Soderland [1999].

Several techniques have been developed to specifically extract knowledge or information from scientific publications. DeepDive De Sa et al. [2016], for example, attempts to transform unstructured data into a well-structured, relational model using machine learning techniques. It has been successfully used in applications like pharmacogenomics Whirl-Carrillo et al. [2012] to extract information about the interaction of small chemicals or drugs from scientific literature and to annotate a pharmacogenomics database with it. Similarly, it has been

used to extract information for building OMIM Hamosh et al. [2000], a catalog of human genes and genetic disorders. Other techniques have been developed in the Blue Brain project to extract information about brain connectomics from neuroscience literature Richardet et al. [2015].

Similarly, efforts have been undertaken to extract information from doctor's notes, i.e., unstructured medical records. IBM's Watson for Healthcare Chen et al. [2016], for example, uses natural language processing as well as machine learning to extract information from doctor's and nurse's notes (among many other sources of information) to form and test hypothesis and assist diagnostics. Additional research has also used natural language processing for extracting information from notes. The more focused the application, e.g., extraction of medication information, the more accurate the result Uzuner et al. [2010], nlp [2009].

3.3 Data Cleansing

Errors in data may originate from a whole array of sources like data entry errors (particularly when capturing the data manually in forms — a common procedure in electronic health records), measurement errors (due to faulty sensors/instruments, instrument design, miscalibration or interference), distillation errors (errors in processing before the data is entered into a database) or data integration errors (errors introduced through reconciling data from different sources) Hellerstein [2008].

While general methods to clean relational data Ilyas and Chu [2015] can be used to clean medical data, more specific approaches are ideally used. The quality of medical data can improve using several different approaches. First, although typically too late when dealing with medical data already collected but still good practice to review before data entry, data entry interface design has a tremendous impact in improving data quality. Errors in data entry mostly are due to human errors filling in the forms. Solid design which already at data entry ensures integrity makes random errors substantially harder and thus rare. Second, organizational management can also help to address the issue of entering faulty data. Although, typically also too late when dealing with already captured medical data, proper training of medical profes-

sionals to enter data correctly improves data quality immensely. Third, automated auditing and cleaning improves the quality of the data post hoc. Several techniques exist to fix erroneous data automated. Typically they involve recomputing distillation errors, fixing measurement (particularly calibration) errors through recomputation or excluding implausible values or outliers. Fourth, in many if not most instances, data can only be cleaned using human intervention. There is consequently typically an interaction between data cleaning tools and data visualization systems, i.e., the data is visualized to identify outliers and implausible values which can subsequently be fixed.

The approach to cleaning data depends to a large degree on the type of data. Postal addresses, for example, have both structure and intrinsic redundancy. Using specific software as well as databases makes it fairly easy to improve the quality of postal addresses. Still, ensuring consistency and deduplication of data entries is not straightforward. Similarly, errors in categorical data (e.g., gender, occupation, colors, brands, etc.) can be fixed by comparing the values against the allowed value domain (stored in a database). Doing so allows ruling out values which are not allowed and, if appropriate rules are defined, also enables us to fix erroneous values. Improving the quality of quantitative data (integer, float values, etc.), on the other hand, is difficult as correctness cannot be judged by the value itself. Only with additional information or value constraints can quantitative data be identified as erroneous or fixed. An example where domain knowledge can help to identify faults is a field storing the body temperature which of course cannot exceed a given threshold. Even without particular domain knowledge, assumptions about the distribution of values can be made, and outliers that do not conform to the distribution can be removed. Finally, identifiers are a special case of categorical data, and correctness can also be checked by comparing the values of all allowed identifiers in a list, i.e., in a database.

Specific approaches have, for example, been developed for genetic data. Pedigree genotypes are one example where clean data is particularly vital as they capture animal breeding graphs based on DNA samples from subjects (more precisely, they are represented by SNPs

— Single Nucleotide Polymorphisms which express the difference from the base alphabet as a pair of letters A, C, G, T). Any error in the SNP on a higher level in the pedigree tree will propagate down and affect multiple SNPs in the lower levels (as Mendelian inheritance affects SNPs downstream, i.e., each SNP downstream inherits at least one base pair from its predecessor in the tree).

For this reason, an existing pedigree tree visualization tool VIPER Rahm and Do [2000] has been extended to support visual cleaning of pedigree trees. More specifically, VIPER’s interface has been equipped with visual cleaning functionality for removal of suspect genotypes, markers, and individuals, and — probably not generally applicable to the cleaning of genomic data — for breaking troublesome pedigree relationships, so that it becomes an interactive data cleaning application rather than merely an error viewing interface.

The key cleaning operation supported in extended VIPER is masking of potential errors. The basic idea is to manipulate data by marking certain data points as having unknown values, so explicit discrepancies between parents and offspring are removed. The genotype-checking algorithm then treats these user-defined unknowns identically to missing values in the raw data: it infers the complete set of acceptable possible values based on analyzing the closest individuals in the pedigree that have present and correct values for the same marker. Thus, as well as discovering where errors occur, the algorithm uses structural information to help fill in the blanks.

Missing values have also been studied in the context of clinical trials Kennedy et al. [2013]. The question to be answered is what should missing values be replaced with. The most used technique for dealing with missing values is imputation, i.e., replacing each unknown by a value estimated from the data available. Imputation can be point based or distribution based. In the latter case, the (conditional) distribution of the missing value is calculated, and predictions are based on this estimated distribution. Multiple (or repeated) imputation generates complete versions of the data that are combined for final inference in a statistical setting.

Past work has primarily focused on identifying relevant features

in existing data to infer missing data using classification tree models Prather et al. [1997], Sariyar et al. [2011]. This representation of a hierarchical structure is achieved by inducing a classification tree on labeled training data, i.e., typically a manually selected subset of the existing data. Once built, the edges of the classification tree represent the conditions according to which the objects in parent nodes are allocated to their child nodes. The leaves represent those classes to which the objects in these leaves are assigned.

Although computationally rather costly – at least more costly than random imputation of values – unique value imputation appears to work best and the pragmatic approach of replacing NA by 0 exhibits results that are not worse than the computational far more costly reduced-model classification. This is unusual for most other settings in which unique value imputations are distinctly worse to more sophisticated approaches. A major exception is an imputation with 0.5 which is preferable when two conditions hold: missingness depends on the values of the class variable, and this dependence is present both in training and in the test data.

Study validity has in the past been discussed predominantly with regard to study design, general protocol compliance, and the integrity and experience of the investigator. Data handling, although having an equal potential to affect the quality of study results, has received little attention. As a result, even though the importance of data-handling procedures is being underlined in good clinical practice and data management guidelines, there are important gaps in knowledge about optimal data-handling methodologies and standards of data quality.

Data cleaning has long been viewed as a suspect activity, bordering on data manipulation Van Den Broeck et al. [2005]. Even today, whenever discussing data cleaning, it is still felt to be appropriate to start by saying that data cleaning can never be a cure for poor study design or study conduct. Concerns about where to draw the line between data manipulation and responsible data editing are legitimate. Yet all studies, no matter how well designed and implemented, have to deal with errors from different sources and their effects on study results. This problem occurs as much to experimental as to observational research

and clinical trials. Clearly, in all studies, the thorough description of the data cleaning procedures needs to be reported. Exactly what to report and under what circumstances remains mostly unanswered. In practice, it remains rare to find any statements about data cleaning methods or error rates in medical publications.

Although certain aspects of data cleaning such as statistical outlier detection and handling of missing data have received separate attention, the process of cleaning medical data, as a whole, with all its conceptual, organizational and statistical aspects, has not been described or studied comprehensively in the context of medical data and considerable work remains to be done. An important aspect to study for example particularly in medical research is the question when a value is a statistical outlier and when it is an interesting medical phenomena. Ruling out all values that do not conform to the expectation may readily confirm the initial hypothesis of the experiment but may remove conflicting data and may indeed result in data manipulation.

3.4 Schema Mapping, Matching & Integration

Research on medical data frequently makes it necessary to combine data from several different sources, be it to combine data from different hospitals, different medical studies or a combination thereof. Where all sources complying with the same ontology or controlled vocabulary, integration would be straightforward. Different sources, however, typically follow different naming conventions, making a schema matching step necessary before the data can be integrated.

A distinction between schema matching and mapping and the alignment of ontologies has to be made. Reconciling ontologies refers to matching ontology concepts and can play a vital role in, for example, aligning ontologies like SNOMED CT and ICD-10. Alignment of the ontologies can be helpful when matching different schemas based on ontologies. The particular instance of a schema, however, almost always deviates from the ontology even if the data was collected with a particular ontology in mind and relying on the ontology alignment thus, rarely is sufficient. Still, ontology alignment information can cer-

tainly help in the schema matching process, and schemas in themselves also contain additional information that can be used for the matching process.

Schema matching is generally referred to as the process to identify semantic equality between two fields of different schemas (e.g., fields surname and last name could be matched) while mapping refers to the process of transforming values between schemas (e.g., conversion from different scales or units like Celsius and Fahrenheit for temperature).

The challenges in automated schema mapping Lenzerini [2002] typically are due to the heterogeneity of the data on multiple levels. Different schema can vary due to differences in the language used for representing the elements, resulting in (syntactic heterogeneity), differences in the types, structures of the elements (structural heterogeneity), differences in the underlying models (model heterogeneity) or where the same real world entity is represented using different terms (semantic heterogeneity).

Given the differences, reconciliation or mapping is done in several steps Bernstein et al. [2011]. First, during preintegration, the differing schemas are analyzed to decide on an integration policy which defines what schemas to match as well as the order of matching. Second, the schemas are compared to detect correspondence as well as potential conflicts between concepts. Third, conflicts between concepts are resolved as far as is possible and finally, the schemas are merged into an intermediate schema.

Over the years, several different approaches have been developed to match schemas. One research direction relies on artificial intelligence and semantic information to match the schemas, i.e., to find corresponding (and also conflicting) concepts in the schemas. The results, however, are rather esoteric as there is little indication that schema mapping can be done without human involvement.

More traditional schema matching approaches work on different levels in that they consider different pieces of information for the matching Rahm and Do [2000]. Schema-level matchers only consider schema information, i.e., only use the information of the schemas. The information available in the schemas includes the usual properties of elements,

such as name, types, relationship types, constraints, and the structure. Working at the element (atomic elements like attributes, i.e., rows or fields) or structure level (matching combinations of elements that appear together), these properties are used to identify matching elements in two schemas. Language-based or linguistic matchers use names and text (i.e., words or sentences) to find semantically similar schema elements. Constraint-based matchers additionally exploit constraints often contained in schemas. It is, for example, precisely constraints where schemas offer substantially more information for matching than ontologies. Such constraints are used to define data types and value ranges, uniqueness, optionality, relationship types, and cardinalities, etc. Constraints in two input schemas are matched to determine the similarity of the schema elements.

Instance-level matchers use the actual data to gather important information about the contents and meaning of the schema elements. These are typically used in addition to schema level matchers in order to increase the confidence in match results, more so when the information available at the schema level is insufficient. Matchers at this level use linguistic and constraint-based characterization of instances. For example, using linguistic techniques, it might be possible to look at the actual values of an attribute denoted last name, as well as an attribute, denoted surname to infer that they are likely the same field.

Hybrid matchers directly combine several matching approaches to determine match candidates based on multiple criteria or information sources. Most of these techniques also use additional information such as dictionaries as well as user-provided match or mismatch information.

A complimentary approach is to reuse matching information as auxiliary information for future matching tasks. The motivation for this work is that structures or substructures often repeat, particularly in the reconciliation of medical data from different studies. Such a reuse of previous matches, however, needs to be a careful choice as it may only make sense for some part of a new schema or only in some domains.

Once the schemas are matched, the data needs to be integrated on a physical level. In the simplest of all cases, the data sources can simply be copied together in a single data warehouse with a number

of scripts performing the matching and mapping information. In many cases, however, physically copying the data is not feasible, be it for reasons of size, due to legal reasons or for other reasons. In this case, the data is typically left in its original source, i.e., database, to maintain physical independence and after the schemas of the different sources are matched into a global schema, queries are asked on the global schema but translated into the local ones. Figure 3.1 illustrates the idea: queries are asked against a global/mediated schema and are then translated to the schemas of each local data source to query them individually.

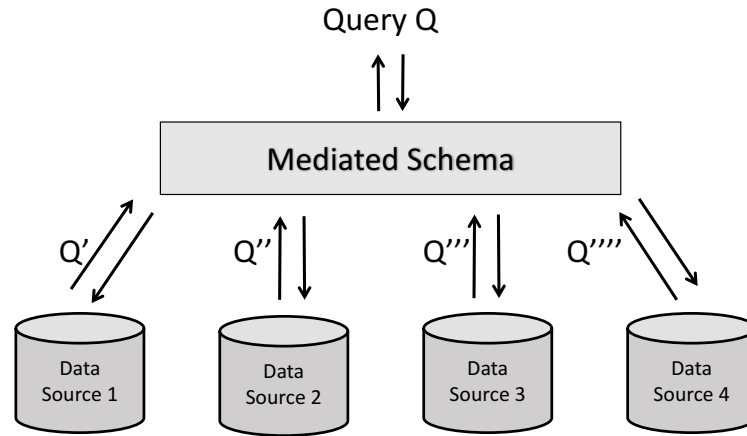


Figure 3.1: A global schema is used to mediate between data sources.

The process of data integration involves combining data residing in different sources and providing users with a unified view of these data sources which is mainly achieved through schema matching. At the core of any data integration attempt is the mediated schema which is the schema emerged from matching all schemas of data sources. More formally, data integration between several different data sources requires a triplet of the mediated schema, the set of heterogeneous source schema (the local data sources) as well as the mapping between global and local schemas. Given this triplet, a mediator will accept incoming queries, translate them as queries for the local schemas, execute them on the local data sources, collect the result (transforming them to the global

schema) and finally return the full result.

Two popular ways to implement data integration based on the triplet exist Lenzerini [2002] : Global as View (GAV) and Local as View (LAV) (see Halevy [2001] for a comprehensive review of methods). GAV systems model the global database as a set of views over the local schemas. In this case, the mapping associates to each element of the global schema a query over the local schemas. Query processing becomes a straightforward operation due to the well-defined associations between the global schema and the local schemas. The burden of complexity falls on implementing mediator code instructing the data integration system exactly how to retrieve elements from the source databases. If any new sources join the system, considerable effort may be necessary to update the mediator. Thus the GAV approach appears preferable when the sources seem unlikely to change.

In LAV, on the other hand, the source database is modeled as a set of views over the global schema. In this case the mapping associates to each element of the local schemas a query over the global schema. In this case, the exact associations between the global schema and the local schema are no longer well-defined. As is illustrated in the next section, the burden of determining how to retrieve elements from the sources is placed on the query processor. The benefit of an LAV modeling is that new sources can be added with far less work than in a GAV system. Thus, the LAV approach should be favored in cases where the mediated schema is less stable or likely to change.

On a practical level, multiple systems and approaches have been proposed. An interesting approach for general purpose data integration for web data, Web Tables Cafarella et al. [2008] (now maintained by Google) has been developed to integrate abundantly available tables with semi-structured data on the Web. With Web Tables, multiple tables in HTML documents can easily be extracted and integrated. Fusion Tables Gonzalez et al. [2010] is an extension of Web Tables and, in addition to the integration of structured web data, also allows its collaborative integration and visualization with different tools.

Several application or domain-specific systems have been developed for the purpose of integrating data in the sciences. One example is

InterMine Louie et al. [2007] which integrates the heterogeneous biology datasets. Following from the previous discussion of data integration, InterMine can be understood as a data warehouse which essentially copies all data into a single location: all data is imported from its respective raw data files and is mapped to objects which are in turn mapped to a relation schema. The global schema is fixed and covers all the file format and information the system can import.

The InterMine database is equipped with data integration modules that load data from common biological formats (e.g. GFF3, FASTA, OBO, BioPAX, GAF, and others). Custom sources can be added by writing data parsers or by creating XML files in the appropriate format. InterMine's core model is based on the sequence ontology and can be extended by editing an XML file. Model-based aspects of the system are automatically derived from the data model, ensuring that also UI elements are easily extensible at least to some degree.

A major challenge particularly of biological databases is the reliance on external identifiers (especially in the context of protein information). External identifiers are, for example, used to reference and thus, loosely integrate information in external databases. Identifiers, however, may change breaking the reference to other external databases. InterMine addresses this with an identifier resolver system that automatically replaces outdated identifiers with current ones when loading the dataset into InterMine. This is achieved using a map of current identifiers and synonyms from relevant databases.

The query approach of EHR4CR Marco-Sola et al. [2012], on the other hand, is a true data integration approach with a mediator translating queries to be executed on different data sources. EHR4CR enables the selection of patients for trials to form different data repositories. Its approach essentially can be understood as local as a view: each one of heterogeneous data sources provides roughly the same information (heterogeneous patient information from different studies and health records). A mediator (called broker in EHR4CR) forwards incoming queries to the query module implementation which uses a specific adapter to access a data source. To integrate a new data source, a new adapter has to be developed.

A particular challenge in integrating medical data is EHR record linkage. To study healthcare data in its entirety, EHR's for the same patient stored at different deliverers of care need to be integrated. Unfortunately, linkage is not as simple as just matching identifiers. Identifiers are frequently missing, or inaccurate and probabilistic linkage thus needs to be used. Probabilistic linkage assigns different fields in the records different weights, e.g., name has more weight than gender, and a record is linked to a record with the best overall agreement, i.e., the highest cumulative weight. Similarly, fuzzy matching of EHR record identifiers can be used Hollar [2009].

Weighted or fuzzy matching, however, is not straightforward. Even small errors in linkage can lead to biased results mis [1965] or if unlinked records are excluded from studies, can lead to biased results Schmidlin et al. [2013]. Important biases also arise if linkage success differs between groups, with differential linkage by ethnic group producing biased mortality ratios. Similarly, variations in data quality between hospitals can affect linkage.

4

In-Situ Data Analysis

When handling data in medical use applications, the data oftentimes has to be processed and analyzed in-situ for two main reasons. First, data in the medical sciences is available on an unprecedented scale, and the amounts are steadily growing. Importing this data into a database before it can be processed or analyzed is a major challenge and introduces a considerable delay. Second, the data can oftentimes not be moved or copied (see Chapter 2 on privacy) outside the institution. Even just reformatting the data within the system collecting it may not be acceptable Venetis et al. [2015].

In this chapter, we discuss the approaches available to query and analyse massive amounts of data — be it structured or unstructured — in its place and in its format without having to import it into a database in a time-consuming process. We discuss how traditional databases support querying raw structured files and optimizations thereof and also discuss how to query and analyze unstructured data.

4.1 Processing Data In-Situ

The problem of processing massive amounts of unstructured data, i.e., data that does not lend itself easily to be imported into relational databases, has recently been identified and several systems have been developed to address the issue. Examples of abundantly available and unstructured data are electronic health records, imaging data (e.g., CT scans), genomic data and others. In the following, we discuss applications of the most prominent approaches used to process and analyze medical data.

The trend to store more data in recent years does not only increase the amount of data stored but also the variety of data formats. This is particularly true in the medical sciences where the formats vary considerably, e.g., even the format of MRI images can differ substantially between manufacturers of different hardware.

Still, while the data can be integrated, it is frequently questionable if the overhead of the data import is amortized over the analyses. Frequently, analyses are ad-hoc and importing all data into structured data repositories like relational databases is not worth the effort as it takes a considerable amount of time. As a result, the research community developed several platforms in recent years Babu and Herodotou [2013] to process data ad-hoc, without integration. The most prominent approach arguably is MapReduce Dean and Ghemawat [2010, 2004] which Google developed to process and analyze the unstructured data for its search functionality. MapReduce essentially defines a programming model and a system to execute analyses on unstructured data. While MapReduce is proprietary code and is not publicly available, Hadoop is an open source alternative offering the same functionality.

MapReduce's (and Hadoop's) programming model simplifies formulating a problem such that it can be run in parallel on a large number of commodity computers. MapReduce's system at the same time provides the infrastructure to easily run these computations at scale. Both are key to enable analyzing massive amounts of data on an equally big infrastructure. MapReduce has consequently become popular to solve many different types of big data problems for unstructured but also structured data (e.g., through using databases implemented on

MapReduce Thusoo et al. [2010]) where scalability is a challenge. The medical research community has developed toolkits and tools based on MapReduce for medical images Markonis et al. [2012], for vast amounts of genomic data McKenna et al. [2010], to link large numbers of electronic health records Bonner et al. [2015], mine health records Wang et al. [2010], analyze clinical data Mohammed et al. [2014] and others.

To address MapReduce’s limited programming model as well as its limited performance (because of the storage bottleneck), researchers at the University of California in Berkeley developed Spark Zaharia et al. [2012]. Spark first uses a more complex programming model, thereby enabling modeling of more interesting program flows. Second, Spark avoids the problem of persisting data to disk by simply keeping it in main memory. Spark, along with tools and abstractions McKinney [2011] built on top of it, has replaced MapReduce in practice. Genomic applications, for example, successfully use Spark in the context of processing medical data. Its more expressive programming model — compared to MapReduce — allows for the implementation of genomic pipelines which run by a factor of at least 2.8X Nothaft et al. [2015] faster.

4.2 Traditional Databases

Fully loading data sets is oftentimes preventive: it is at times legally challenging and there is also a significant initialization cost in loading data and preparing the database system for queries. For example, a medical scientist needs to quickly examine Terabytes of new data, e.g., imaging data, metadata accompanying the images, electronic health records or others, in search of particular phenomena. Even though only a few attributes might be relevant for the task, the entire data must first be loaded into the database. For large amounts of data, this means a few hours of delay, even with parallel loading across multiple machines. Besides being a significant time investment, it is also important to consider the extra computing resources required for the complete loading.

Instead of using traditional database systems, medical applications

oftentimes rely on custom solutions that usually abandon important and useful database features. For instance, declarative queries, schema evolution and complete isolation from the internal representation of data are rarely present and quite often data layout and query execution and very closely tied together, making changes in either very challenging.

Traditional databases are of course designed to efficiently query and analyze large amounts of data efficiently. The parsing and loading of all information into relational tables, however, is still a requirement. Import of the data does not only represent overhead in terms of parsing and writing it into the database but also makes it necessary to design the database schema.

Clearly, the data is oftentimes already available in structured files amenable to querying directly. Querying the file directly has several advantages as it (a) saves the overhead of importing, (b) makes the data instantly available for querying and (c) avoids duplication/replication of data (and thus may make updates more efficient, i.e., if the base file is updated, the changes are immediately available for querying).

The benefits of querying structure files directly have been understood by database developers. As a consequence, MySQL, as well as Oracle, support querying data stored in files without importing it.

MySQL natively already supports different storage engines to allow the query engine to read from different sources (e.g., from disk, from memory and so on). A recent and straightforward extension of MySQL, therefore, is the CSV storage engine which can read and store data in CSV files Schwartz et al. [2012].

CSV tables simply have to be stored in a configurable directory and can be moved in and out freely. With this ease of adding (or removing data files), however, also comes overhead: files, more specifically, CSV tables have to be registered with the system before they can be queried. In MySQL, this is done through creating a CSV table on the underlying file. Whilst creating the table the types of the columns need to be specified such that MySQL can split the lines of the file (using the comma as the separator) and convert the string values into the data types of the database (such that relational operators defined on the

types can be used). Once done, the table can be used to read data from the file if it exists or to write to the file using insert statements.

The drawback of this flexibility (of moving CSV files in and out of the directory) is that indexing is not supported as the file underlying a table can easily be moved away without notifying the engine. Also, transactions are not supported on CSV tables for the same reason.

CSV tables in MySQL support the most common data types like numeric types, strings, dates and so on.

Oracle provides users with external tables Loney and Osborne [2004], a concept similar to CSV tables. External tables complement SQL Loader which the Oracle database already provides to efficiently load structured data files like CSV. Similar to CSV tables in MySQL, external tables in Oracle also support querying as well as manipulating raw data files outside the database.

External tables in Oracle, however, have more options to be configured. Similarly to MySQL, the table has to be defined or created like a regular SQL table, as illustrated in Figure 4.1. Beyond the basic configuration, however, external tables can also be configured to read the data from a specific location (or file). Furthermore, particular value delimiters or formatting instructions can be specified to parse files more complex than CSV files. Parallel loaders contribute to the efficiency of Oracle external tables: if defined during the creation of the external table, the external files can be manipulated in parallel.

Distributed data analytics or warehousing solutions also support querying over external files. Amazon Redshift Spectrum Spectrum, for example, can run queries on structured files distributed in Amazon S3 storage if they are stored in CSV format, as text files, sequence files or in many other formats. Similarly, Apache Kudu Lipcon et al. [2015] based on Hadoop allows SQL queries over well-structured, distributed CSV files.

Besides Oracle and MySQL as well as Amazon Redshift Spectrum and Apache Kudu in the distributed case, several modern database systems offer the ability to query structure data files directly with SQL. External files, however, can only access structured data with no support for advanced database features such as data manipulation oper-

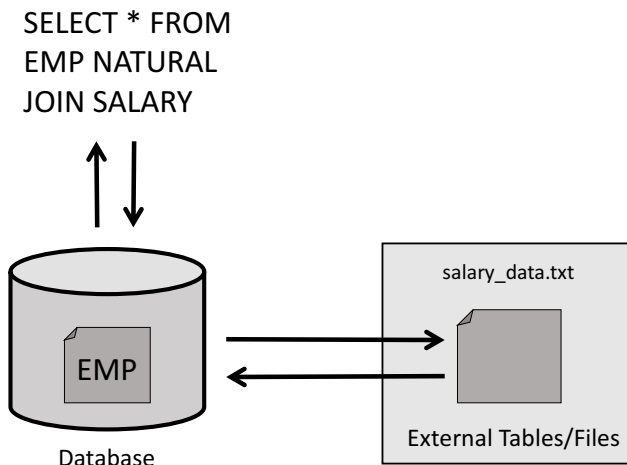


Figure 4.1: The database treats files as external tables.

ations, indexes, transactions or statistics. External files thus require every query to access the entire data file, as if no other query did so in the past. Each row (and typically the entire file) has to be reread and, more crucially, be parsed again. In fact, this functionality is provided mainly to facilitate data loading tasks or to avoid it and not for querying Alagiannis et al. [2012].

4.3 Accelerated In-Situ Analysis

With traditional databases supporting structured external files as tables, the problem is somewhat alleviated. Absence of advanced external data structures to accelerate access to the data, however, puts an undue performance penalty on the query execution on raw data.

The recently proposed NoDB philosophy Alagiannis et al. [2012] aims to provide in situ access to query processing performance that is competitive with a database system operating over previously loaded data. Data therefore does not need to be loaded — to avoid legal and performance challenges — but can instead reside in place without changes. In other words, the vision is to completely shed the loading

costs, while achieving or improving the query processing performance of a traditional DBMS. The proposed design takes significant steps in eliminating or greatly minimizing initialization and query processing costs that are unique for in-situ systems. Even though individual queries may take longer to respond than in a traditional system, the data-to-query time is reduced as there is no need to load and prepare data in advance. In addition, performance improves gradually as a function of the number of queries processed.

The main performance overhead of accessing raw data is parsing and tokenizing the data repeatedly — an overhead avoided in traditional, relational databases through initially parsing all data and transforming and storing it in a suitable format. To provide competitive performance, NoDB uses incremental caching as well as positional maps, a mechanism similar to indexing, on CSV files.

NoDB reduces the overhead by using selective parsing and tokenizing: parsing tuples is aborted as soon as the required attributes for a query have been found. Of those attributes, only the ones needed for the query are parsed (into their binary representation) to further reduce the overhead. The parsed values are additionally cached.

If values are no longer cached, reparsing and retokenizing is necessary. NoDB further reduces the overhead of either by using positional maps which maintain low-level metadata information on the structure of the flat file, i.e., the position of attributes in the raw file. For example, if a query needs an attribute X that is not loaded, NoDB exploits this metadata information that describes the position of X in the raw file and jumps directly to the correct position without having to perform expensive tokenizing steps to find. Positional maps are created on-the-fly during query processing.

The ViDa approach Karpathiotakis et al. [2015] takes the idea of optimizing access or querying raw data a step further and optimizes the query engine for access to structured data. To avoid any undue performance penalty due to operating on structured data files, the ViDa approach first generates and compiles access paths and query operators aggressively optimized for the formats data is stored in, and thus avoids the overhead of query interpretation. Second, through adapt-

ing and changing the engine’s internal data layout, ViDa can adapt to different queries that are best answered based on different data layouts. Finally, ViDa uses an intermediate language to decouple query language from data layout and thus enables users to formulate queries in the language best suited for their problem (e.g., SPARQL for graph queries) independent of how data is physically stored.

ViDa generates its internal structures and query operators on demand, so it can execute workloads more efficiently, depending on the workload characteristics. Decisions concerning query execution are postponed until runtime when as much useful information for the query invocation as possible has been gathered. Influenced from applications of code generation Alagiannis et al. [2014], Karpathiotakis et al. [2014], Krikellas et al. [2010], Neumann [2011], ViDa generates its query operators just-in-time. By creating all operator implementations on the fly, it can flush out optimal, special-purpose operators satisfying the needs of a very specific query.

Approaches based on ViDa and NoDB are at the core of the integration of hospital data Venetis et al. [2015], i.e., electronic health records as well as imaging data, in the medical informatics platform of the Human Brain project Markram et al. [2011]. Access to raw hospital data in the platform is pivotal to develop a better understanding of brain disease.

5

Medical Data Processing

Workflow systems have been developed over decades to support the efficient and distributed processing of massive amounts of data at scale. Initially developed to support the execution of the business process with complex control flows, they found their way into supporting scientific applications through the orchestration of programs and pieces of code via control and (in some instances only) data flow. Figure 5.1 illustrates a genome sequencing workflow orchestrating multiple tools. Each box represents one software tool which passes the results (along the arrows) as input to the next software tool. Workflow systems are extensively used in drug design and discovery Scannell and Bosley [2016], PandeLab based on publicly available databases Bairoch et al. [2005], Irwin and Shoichet [2005] and have also found their way into commercial products supporting scientific applications Numerate, Atomwise.

Several workflow engines and systems are now broadly used for scientific and medical applications. Specific support for medical/scientific data as well as commonly used pieces of code (for the parsing, processing, visualization, etc. of common file formats in medicine) is what makes the workflow engines particularly well suited for medical or scientific applications.

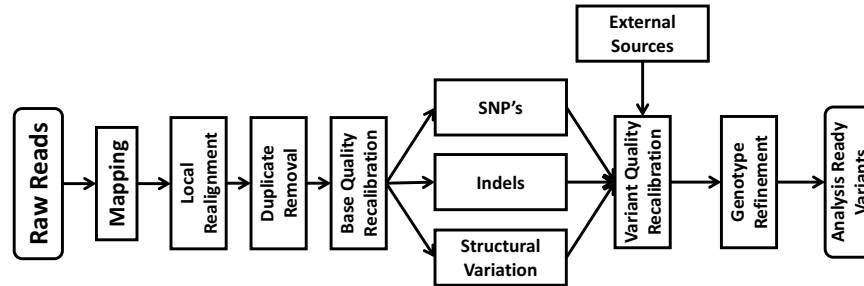


Figure 5.1: Illustration of a genome sequencing workflow.

A crucial aspect related to processing data in general and medical data, in particular, is to keep track precisely how data products were derived and what, i.e., input data software (and versions thereof), was involved in producing it. Capturing, storing and enabling querying of the data provenance or lineage is thus crucial but is also well supported in the context of workflow systems.

In the following, we discuss the two major aspects of medical data processing. First, we give an overview of the available workflow systems to process medical data distributed. Second, we discuss state of the art in capturing provenance originating from the execution of medical workflows.

5.1 Workflow Systems for Medical Research

Several different research workflow systems have been developed in recent years in support of large-scale data processing. In the following, we focus on giving an overview over primarily the distributed ones which can process massive amounts of data at scale and which have been used in the context of scientific and biological applications in general and for medical use cases in particular.

Initially designed for the myGrid project, Taverna Oinn et al. [2004] is an open-source workflow engine now part of the Apache ecosystem. Primarily designed to orchestrate and execute Web services, functionality to execute additional components like small Java snippets, R statistical services, local Java code, external tools on local and remote

machines (via ssh), XPath as well as other text manipulation and others. Individual services, particularly Web services can be defined by importing their service description (WSDL or also as REST API's) or can be searched on BioCatalogue. Additionally, workflows can be searched on myExperiment and directly be integrated.

The execution of Web services, as well as the connection to external machines (via ssh) and the execution of arbitrary applications on them, is crucial to enable scalability through Taverna. Taverna is primarily driven by the data flow with limited support for arbitrary control flow primitives. Although initially designed with a simple model where the complete result is produced before it is passed on, Taverna has been extended to also support a streaming model where single data items, e.g., items on a list or similar, can be passed on by the upstream service to the next service downstream without having to wait for the complete result.

What makes Taverna particularly appealing for its use in scientific workflows, and medical applications is the wide range of available services for biological and medical data. Through its use in several biology-driven projects, Web services from SADI, BioMart, BioMoby and SoapLab can be integrated without any overhead.

The Taverna Workbench also includes the ability to monitor the running of a workflow and to track the provenance of the data produced. The provenance is captured as a W3C PROV-O RDF provenance graph which can be analyzed and visualized with tools for the PROV standard.

Specifically developed to analyze genomics data, Galaxy Giardine et al. [2005], Afgan et al. [2010] provides a powerful, yet simple user interface. The user interface enables users to graphically define the workflow, i.e., to orchestrate the execution of tools (picked from a library of genomics tools) working on datasets (also picked from a library). Workflows are executed on dedicated clusters and, as of recently, also the cloud. While little limits Galaxy for use with other tools and datasets, it is currently exclusively used to implement genomics pipelines.

Initially developed as a workflow management system in support of data exploration and particularly visualization, VisTrails Callahan

et al. [2006] supports the development, execution, and management of arbitrary workflows. Similar to workflow systems, it allows the combination of loosely coupled components, specialized libraries and grid and Web services. Similar to Taverna, and driven by the needs of visualization, workflows in VisTrails are primarily defined with the data flow, i.e., processing components of a data item upstream are executed before the consumer of the same data downstream. Also similar to Taverna, scalability of the execution is accomplished by executing services and pieces of code remotely and not via scaling the workflow engine itself. Still, execution of the workflow is scalable and with proper infrastructure to execute the services (and to exchange data between them) massive amounts of data can be processed.

VisTrails puts strong emphasis on exploration of the parameter space and the visualization of the different results as well as to manage exploratory processes in which computational tasks evolve over time as a user iteratively formulates and tests hypotheses. A key feature for VisTrails therefore also is the ability to visualize the difference not only between runs of workflows (the historic view of them) but also between versions of the workflows themselves. It also supports the ability to create workflows with only minor differences, e.g., parameters, efficiently.

Although crucial in all workflow management and execution systems for the purpose of reproducibility and thus typically a feature, provenance tracking is of particular importance in VisTrails, and the engine is designed accordingly. The importance of provenance tracking stems from the fact that finding a good result will inherently mean that one has to go back and determine what parameters and versions of software components were used to produce it. Similarly, given the support for the fast-paced changes to workflows and hence versioning, keeping provenance traces is crucial so that results can be reproduced.

Visualizations generated as the result of a workflow are shown similar to a spreadsheet, allowing multiple visualizations from different runs or from versions of a workflow to be viewed and compared at the same time. Key to VisTrails is its integration with VTK, a broadly used library for scientific visualizations, particularly for large-scale simulations. Integration with VTK, the VTK plugin for VisTrails, however,

is not open source and only available commercially.

Initially developed to support gravitational wave experiments, Triana Majithia et al. [2004] has been developed into a complete workflow management system designed for scientific applications. Triana models workflows as a directed acyclic graph with vertices being the applications also called components — like other engines, Triana supports primarily Web services but also other, arbitrary snippets of code needed and used to transform the data — and edges being the data dependencies. Workflows in Triana consequently are also a data-driven with little support for control mechanisms: all control flow conditions need to be implemented as part of the components themselves. By using the latter mechanism, i.e., implementing the control flow as part of components, however, enables the implementation of loops and other arbitrary control flow mechanisms.

To address the scalability issue, i.e., to enable processing of massive amounts of data, Triana moves beyond only supporting scale-out through calling of remote services, be it Web services or applications remotely. Crucial for scaling workflows with considerable local computation, e.g., Java snippets executed locally, scripts to support control flow, etc., is also the replication of the workflow engine itself. Triana supports this through the use of the Triana controlling service: while several instances of the engine are running, this service coordinates parallel execution of components of the same workflow (essentially those components that can be parallelized).

There are 600 local components available for use in the core distribution of Triana, which span many domains, such as signal, image, text as well as audio processing. Others can be easily integrated. Triana particularly supports Web services, Grid services, P2P services and other but also legacy code or Java objects.

YAWL (Yet Another Workflow Language) van der Aalst et al. [2004] is a workflow language driven by a comprehensive list of workflow patterns supporting by far the most patterns beyond BPEL. The language is supported by a software system that includes an execution engine, a graphical editor, and a worklist handler.

Based on an exhaustive analysis of existing workflow languages,

YAWL was developed. The new language is based on Petri nets, a well-established concurrency theory with a graphical representation and on well-known workflow patterns — a benchmark used to evaluate and compare differing workflow languages and engines. Petri nets can capture a considerable number of the identified control-flow patterns, but they lack support for the multiple instance patterns, the cancellation patterns, and the generalized OR-join. YAWL, therefore, extends Petri nets with dedicated constructs to deal with these patterns. As opposed to the other workflow engines discussed so far, YAWL supports the explicit definition of a control flow and implements an extensive set of control constructs. The data flow has to be specified explicitly.

An important point of extensibility of the YAWL system is its support for connecting to external components. This enables running workflow instances and external applications to interact with each other in order to delegate work. Custom YAWL services are external applications that interact with the YAWL engine through XML/HTTP messages via defined formats, some located on the YAWL engine side and others on the service side. Integrating a service other than a Web service, therefore, means wrapping the legacy code or the application to interface with the proprietary XML interface.

Although YAWL has been used extensively for hospital applications, the support for out of box components is very limited. The focus on supporting complex control flow patterns has had a detrimental impact on the support for more components. Still, only Web services or code/application wrapped in the proprietary XML/HTTP can be integrated into YAWL workflows.

Kepler Ludäscher et al. [2006] is a comprehensive workflow tool for designing, executing, evolving, and sharing scientific workflows. Kepler provides facilities for process and data monitoring as well as provenance information. As with most other workflow engines, workflows are defined as directed graphs where the vertices represent computational components, and the edges define the exchange of data and results between paths along which data and results flow. As such, Kepler thus also is data flow driven with no support for control flow constructs.

To scale, Kepler features the ability to call services remotely. This

includes rather stateless Web services as well as stateful Grid services where arbitrary jobs can be submitted. It is important particularly to scale local computations. However, Kepler can also be replicated across different machines. With this it supports single sign-on and, importantly, automated and efficient staging of data and models to nodes as needed, efficient scheduling of resources, monitoring of the progress of execution, and job control and so on for submitting workflow tasks and components to be distributed among compute nodes in a computer cluster.

As other systems do, Kepler also comes with many predefined components. Standard components include reading local files of different types, code snippets to format data (read arrays and so on), etc. More interesting to scale out the computation are components to call Web services, submit Grid jobs and others. Crucial in the context of processing scientific and medical data are available components for statistical analysis, computational chemistry tools as well as several components specifically developed for scientific projects where Kepler has been used but which can also be used in general.

A particularly interesting aspect of the processing of scientific and medical data is that Kepler provides direct access archives for scientific data. For example, Kepler provides access to data stored in the Knowledge Network for Biocomplexity (KNB) Metacat server and described using Ecological Metadata Language. Additional data sources that are supported include data accessible using the DiGIR protocol, the OPeNDAP protocol, GridFTP, JDBC, SRB, and others.

The LONI Pipeline Rex et al. [2003] is a distributed system for designing, executing, monitoring and sharing scientific workflows used particularly in medical research, e.g., genomics, neuroimaging, and biomedical informatics. Developers can orchestrate arbitrary different tools, and run and monitor the execution as well as visualize the results.

Like most other workflow engines, Pipeline also follows the data flow pattern: input data or results are connected to the next computational unit/software tools to define the order of execution. Clearly, applications in a Pipeline definition without a dependency/path between them in the resulting directed acyclic graph can be executed in

parallel.

The LONI Pipeline is designed as a lightweight middleware. While this makes it very simple to set it up and run it, it also means that no simple computations can be executed locally, i.e., a set of internal core libraries, filters, and processes for rudimentary image processing (e.g., alignment). All processing has to be done remotely through Web services, grid services, and others. While this appears somewhat limiting, it also means that execution on a single instance scales very well. In the absence of control flow patterns, executing the workflows involves very little overhead as all processing is done remotely. Doing so, however, means that considerable amounts of data need to be moved between applications run remotely.

What makes the LONI Pipeline interesting for medical research, particularly for research in the context of imaging is that through its use in several projects, a wide array of pre-built software applications are available. Examples include the FMRIB Software Library, a software library for image analysis and statistical tools for functional and structural MRI brain imaging data, Analysis of Functional NeuroImages for processing functional MRI data, Freesurfer for MRI image analysis in general. Most of the applications are readily available to be integrated into workflows.

SOARIAN Haux et al. [2003] is a workflow management system specifically developed by Siemens for the use in hospital and the treatment of electronic health records therein. As opposed to other workflow engines discussed, it is only available commercially and is not designed for research directly. Given that it is specifically designed for processing electronic health records and is used broadly in hospitals, interfacing with it for medical research is key.

The engine is specifically designed to support the business process of handling electronic health records and document treatments. Whilst new process/workflows can be designed; they can typically only use a limited set of applications found in the libraries coming with SOARIAN. Designed to efficiently acquire patient and treatment data, it only supports processes based on simple data flows. Strong emphasis is put on featuring visual interfaces which efficiently support the acquisition

of data by medical staff.

The extensibility of the system, therefore, is limited as only prepackaged software components can be used. At the same time, SOARIAN includes a number of analytical tools that can be used to perform analyses on the patient data directly. The analytics focus on the analysis of key metrics in hospitals, e.g., entry application can be measured by the turn-around time, but can also be used for more sophisticated statistical analysis about the effectiveness of treatments. An analytical database with process metrics and clinical and/or financial patient data serves as a base to export the data to external applications.

Given that SOARIAN is primarily used in the data acquisition process, its scalability primarily depends on the underlying data persistence infrastructure (any type of SQL database like SQL server can be used). Examples of large-scale implementations based on Windows 2000 and SQL Server these products include an OLTP-System with 930 concurrent users on a 1.3 TB database.

Still, SOARIAN's engine can be replicated across different machines to improve scalability, but it is primarily done to reduce the response time which is key to the data acquisition process based on the interaction with medical staff.

Swift/T Stef-Praun et al. [2007] focuses entirely on high performance and scalability. Workflows expressed in the Swift language can be translated into MPI programs which in turn can be executed on massively parallel infrastructure like clusters or supercomputers. The Swift language is defined as a concurrent language with C-like syntax. It is primarily used to manage calls to external functions written in C, C++, Fortran, Python, R, Tcl, Julia, Qt Script, or already compiled programs. Given that the Swift language essentially is C, the workflow specification features control and data flow definition giving workflow developers full expressive power. Complete expressiveness, however, comes at the price of ease of use: to develop a workflow programming skills are needed as no graphical user interface is available.

Swift/T scales easily to hundreds and thousands of computational nodes. Its applications are compiled with most snippets (e.g., C, C++, etc.) directly into the executable or are called from the compiled code

(e.g., existing applications, etc.) and so workflow execution does not have to call external components but can execute them locally. With its focus on MPI, exchange of data can be accomplished very efficiently – an important cornerstone for processing massive amounts of data.

Given the absence of any requirement for integrating computational components – most functions can simply be linked via the respective library while others can be executed as applications – Swift/T can be used for arbitrary applications. All existing code bases can easily be integrated into workflows without the need for wrappers or similar infrastructure. It has, however, particularly broadly been used in the context of processing MRI images. To process MRI images of high resolution particularly fast data transfer is needed and Swift/T offers this as it is designed for MPI and thus cluster and supercomputers with substantial bandwidth.

5.2 Medical Data Provenance

Capturing provenance information is crucial in the delivery of care or in the context of electronic health records is crucial to enable auditing access to electronic health records as HIPAA requires. Indeed, inspecting the provenance and access information automatically recorded by HIPAA compliant systems allows to find unauthorized access Perez and Moreau [2008].

Capturing the provenance of data processing, however, is also crucial in workflow systems processing medical data. When processing medical data as reproducibility is key when developing diagnostics or treatments for legal and ethical reasons. What makes tracking the provenance simpler in this context, is that processing is done with a well-defined control flow and not with an ad-hoc definition, e.g., scripts, and tracking has to be set up and tested only once and can then be reused repeatedly without any additional overhead.

The need for provenance or lineage tracking for data processing (and hence workflow systems) is well-understood. Most workflow systems discussed before support capturing the provenance automatically to some degree. Taverna even goes as far as to produce an archive

including everything — depending on the configuration — to reproduce an experiment: workflow definition, input data, intermediate data products as well as the provenance trace (connecting the data with the computational units as well as storing versions of the software used). Clearly, this incurs considerable overhead, particularly when processing massive amounts of data and so storing intermediate products is rarely done in practice (as they can typically be reproduced with input data and workflow). Figure 5.2 shows an example of a lineage graph of a proteomics pipeline/workflow. The node pointed to depends on the node point from, e.g., a peptide depends on a spectrum as well as a DB entry while a protein depends on multiple peptides.

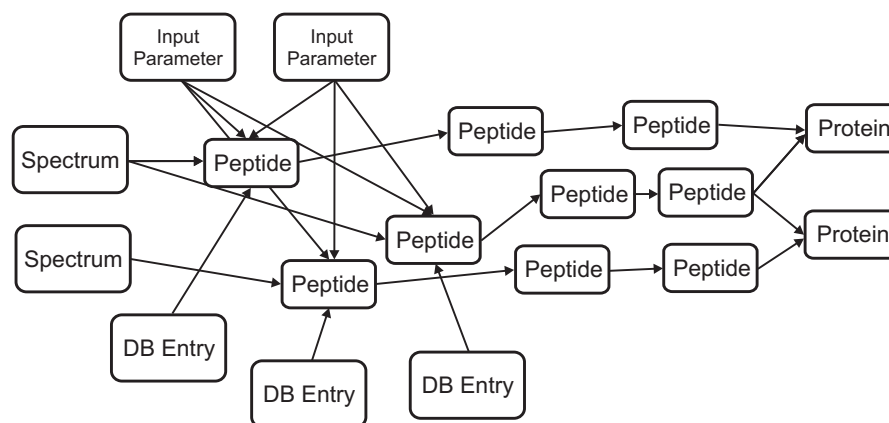


Figure 5.2: Lineage graph of a proteomics workflow.

Scientists rarely process data using just using one workflow system, and processing oftentimes is done in several systems and also with other toolsets. In practice, the workflow system or processing pipeline is primarily chosen by the tools that it features and is thus chosen for the particular application. If different workflow systems are the best fit for a particular piece of work needed to be done, multiple different systems will be used. Crucial to tracking provenance thus is the interoperability between the provenance tracked and also a uniform representation of the provenance information. To this end, a provenance standard has been defined such that the provenance information can be exchanged/combined between different workflow systems and so that

it can be analyzed with visualization tools that are independent of the workflow system.

The PROV provenance standard Gil et al. [2012] evolved from the need to standardize how provenance information is captured in general and from workflow systems in particular. Over a series of workshops, the open provenance model was devised by comparing and testing interoperability between the provenance produced by different workflow systems. The open provenance model has since become the W3C PROV standard.

Borrowing several concepts from agent-based computing, the PROV model defines provenance information about entities, activities, and people involved in producing a data product, which is used to reproduce the piece of data and to assess quality and trustworthiness. PROV fundamentally defines a generic model useful to define how anything came to be, not just data products. In describing it, we will, however, focus on provenance for data.

PROV defines a family of standards defining a basic/core model of provenance as well as standards on how to store it (XML), the ontologies used, and constraints to validate correctness and others. The basic provenance model defines agents, entities, and activities. At the core, entities correspond to the data products used and generated by activities. Fundamentally, the agent can be a person, an organization, but in our context is probably best described as a software artifact, i.e., a piece of code. The relation or difference between agent and activity is not straightforward, but mapped to our problem can be considered the agent being the software/application and the activity is an execution of the software. Consequently, generation of a data product does not necessarily occur at the end of an activity but it may be generated throughout it.

Crucial to the model are the relationships between entity, activity, and agent. As already mentioned, a data product is generated by an activity (or derived from it) but is also used by an activity (potentially the same). An agent is associated with activity, and an entity is attributed to an agent (although this relationship can also be inferred through the activity alone).

Key to PROV is that entities are mutable. A single data product, therefore, is an entity, and all versions of it are generated through edits. To connect different versions of a data product together, the PROV model uses specialization: the data product represents an entity, and all versions of it are specialized entities. The relationship between activity, agent, and entity already defines a partial order over the resulting directed acyclic graph and time is not strictly needed. For the versioning, however, time is a crucial component and should be stored along with each entity (the time an entity was generated).

A further concept in PROV is the subtype. The subtype is used to introduce a hierarchy of provenance information. As several workflow engines support the recursive definition of workflows, i.e., a workflow can execute other workflows (including itself), the provenance information needs to provide similar capabilities. In the PROV model, this is accomplished through subtypes that essentially can encapsulate parts of the provenance. The mechanism can be used to simplify the provenance graph (parts of it can be summarized to reduce complexity) or to provide privacy, i.e., not all provenance information is available to everybody.

The PROV family of standards also defines other aspects beyond the core model. It defines how to serialize the provenance graph into XML for storage, an ontology to map it to RDF (and to Dublin Core terminology) as well as annotations to make it readable for humans. It also defines how to use standard Web protocols to obtain information about the provenance of resources on the Web as well as how to query it. Finally, it also defines the means/constraints needed to test for the correctness and validity of provenance traces.

While the PROV model is able to comprehensively capture essential provenance information, it does have shortcomings. A challenge is the definition of granularity of the provenance information or moving from a more fine-grained representation to a more coarse-grained through collapsing several nodes of a graph into one (or vice versa). As discussed previously, this is a crucial operation to reduce the complexity of a provenance graph for the purpose of visualization and to hide details for reasons of privacy.

What makes the PROV model difficult to use is that the operation of collapsing several nodes into one is underspecified (as is the reverse operation): it is not clear how to map the edges from a fine-grained on a coarse-grained representation. For example, if several nodes have outgoing edges, it is not clear how they can be collapsed into one outgoing edge. Even if this can be defined, the provenance graph defined according to the PROV model does not capture sufficient information to reverse the operation. The problem has already been identified, and formalisms to address have been defined Buneman et al. [2016].

As discussed, the PROV standard also defines how provenance information is stored (e.g. in XML format). How it needs to be organized on disk, however, remains undefined but is crucial as it has a considerable impact on query performance.

Most provenance queries reduce to queries on the directed acyclic graph representing the provenance information. Most provenance models represent the information tracked as directed acyclic graph (as the PROV model does) where nodes represent software tools or data artifacts and the edges between them model a dependency. Most interesting queries are graph queries on the provenance graph, e.g., what software input data was involved in producing a particular result, translates directly into a reachability query in the provenance graph.

Unfortunately, graph queries are rather difficult to execute on a relational database. Graph information is typically serialized or flattened into a relation model where, for example, a table stores all the edges including the two endpoints (other variations are also possible). Executing a reachability query then means to compute the transitive closure which results in a recursive execution of the queries, rendering the execution inefficient as recursive queries are typically not efficiently executed on database management systems.

To address the the problem, labeling schemes have been conceived Heinis and Alonso [2008]. With them, a total order over all nodes in the graph can be defined (at least for a subset of all directed acyclic graphs), and the labels over the total order can be stored in a relational table, thereby turning a graph query into a simple range query on a relational table. While not all graphs can be encoded or

labeled with this approach, given the control flow patterns in workflows, most directed acyclic provenance graphs resulting from workflow execution can be encoded.

A crucial aspect in the context of provenance is privacy. While the definition of the workflow may not reveal any sensitive information, the provenance, i.e., what data is passed through the workflow, may show information that is best-protected Davidson et al. [2011].

Privacy of provenance information can pertain to three different areas: intermediate data, behavior of a module or the structure of a workflow. Privacy with respect to data means that certain intermediate results must be protected from unauthorized access. An example of an intermediate result may be the genetic disorders the patient is susceptible to which should not be revealed to users without the required access privilege. Module privacy, on the other hand, means that an unauthorized user should not be able to infer from the input to a module its output. Put differently, a user may not understand the module, i.e., it is still treated as a black box, but given the input, the adversarial user should not be able to infer the output. While this may not appear to be crucial, from a data owner’s perspective, this is important because they may not want someone who has access to some of their data be able to infer the result of the computation of the module. Similarly, the developer of the module may not want others to be able to simulate their model: knowing what input leads to what output, the behavior can easily be simulated without implementation. Structural privacy refers to hiding structure of the control and data flow in the given execution. Structural privacy is probably best described as the inability of unauthorized users being able to know what variables/data artifacts contributed to the result of particular computations. For example, whether or not a variable lifestyle was used to infer the disorder.

Data privacy probably is the simplest case of the three and may simply be achieved by blanking or removing intermediate results from the provenance trace. Sensitive intermediate results should only be accessible by authorized personnel.

If the information about intermediate data is repeatedly fed into the execution of the workflow or a module, the behavior of the module can

be partially or even completely inferred. To address this issue some of the intermediate results need to be hidden, i.e., only a well-chosen subset will be made available in the provenance trace. What precise subset should be chosen needs to be balanced: on the one hand some information needs to be hidden to conceal the functionality of the module, but at the same time the remaining data needs to have enough utility for a user of the provenance trace, i.e., it should be sufficient to establish trust. Clearly, a level of privacy needs to be set initially, and the utility of the subset chosen needs to be maximized accordingly as not all intermediate data may have the same utility and neither have the same impact on privacy. Intermediate data items thus have a weight in terms of privacy and utility and meeting the privacy goal while maximizing the utility is an interesting optimization problem.

Structural privacy refers to hiding that a particular module contributes to an intermediate or final data item. In terms of the provenance graph, structural privacy essentially means that edges should be removed such that it is no longer obvious what modules or intermediate data have an impact on other modules/data items. By removing the edges, however, we may lose more information than intended: we may wish to hide what information is passed on, but not that information is exchanged. One approach to address the issue thus is to hierarchically cluster or aggregate components C into composite modules and the edges going into C are collapsed into one as are the ones going out (similar to how PROV aggregates modules). By doing so, we can hide the precise dependencies between modules but can still make the dependency explicit, thus limiting unnecessary loss of information.

6

Medical Data Formats and Indexing

In this chapter, we will review the current state-of-the-art in indexing medical data for faster analysis. We distinguish between genetic data, proteomic data, patient data, imaging data and other potential types of medical data. We will review the standards for each of these types of data and discuss how they can be indexed.

6.1 Genetic Data

Although generally encompassed in the CDISC (Clinical Data Interchange Standards Consortium) Kuchinke, Aerts, Semler [2014] standard which standardizes a wide array of aspects of medical or patient data (XML representation, study/trial tabulation and data models for laboratories, operations, and others), the representation of genetic data is not standardized per se.

Genetic data, however, is frequently acquired through the use of DNA microarrays which are a collection of microscopic DNA spots attached to a solid surface. Scientists use DNA microarrays to measure the expression levels of large numbers of genes simultaneously or to genotype multiple regions of a genome. DNA microarrays are typically

arranged as rasters or arrays, and raster or array databases consequently can be used to accelerate access to them for a more efficient analysis. Figure 6.1 shows an example microarray with its array structure.

One particular instance of an array database which is broadly used in general but in particular for imaging data from astronomy (e.g., raster telescope) and, most importantly in this context, for DNA microarrays, is SciDB Brown [2010]. For these type of applications, arrays or rasters are an ideal data model. Relational databases do not have a notion of position per se but instead are using a data model based on sets of items without order. In images and microarrays, however, the notion of neighbor or adjacency is important and can simplify querying and analysis. Clearly, positional information can also be modeled in relational databases but incurs overhead (already to simply store the additional information).

In addition to a more apt data model, SciDB also claims to outperform traditional relational database systems by a factor of 100 for querying and analyzing array data.

Modeling uncertainty is vital not only for its use in genetics and astronomy but also for other applications. Some recordings, be it of pixel from a telescope or an expression level of a gene may be inherently uncertain. Crucially, SciDB models this by providing the ability to attach error bars or ranges to the data. The uncertainty can then also be used as an additional predicate when querying the data.

SciDB adds several operators for arrays that are not common in



Figure 6.1: Example of a DNA microarray used to detect DNA.

relational systems. The operators are based on the notion of position information, adjacency, and neighborhood. More particularly, SciDB introduces operators to slice arrays, retrieve subsets and samples of them. SciDB's query language furthermore supports efficient execution of operators on entire slices and subsets, i.e., on all fields in the subsets without the need to resort to procedural languages.

SciDB is designed for massive amounts of data as they can be seen in astronomy and medicine (genetics). To this end, SciDB is designed to scale out on multiple nodes. Given that most data in the sciences is only written once and rarely (if ever) updated, SciDB does not support updates. Avoiding any updates makes a shared nothing architecture Stonebraker [1986] used for SciDB very well scalable.

The data is distributed across nodes in contiguous chunks of neighboring array fields. To accelerate operators which require access to neighboring fields of the array to compute a local result, the chunks overlap. If they did not overlap, different chunks on different nodes would have to be accessed to compute the result. Overlap or replication is typically a challenge for updates but not for SciDB as it does not support updates.

Developed for arbitrary raster or array data, SciDB does not offer support for operations particular to genetic data or, more particularly, to microarray data. The NCI/CIT MicroArray Database (mAdb) System Greene et al. [2003], on the other hand, has specifically been designed for microarray data and supports no data from other applications.

As opposed to SciDB, mAdb is not a database in the classical sense but an entire system where users can upload gene expression array data. As such it is not designed to manage and index vast amounts of data but rather to bundle the necessary tools and make them available for the analysis of reasonably sized datasets through the Web.

Users can filter gene expression spots (the fields of the array) based on quality parameters to create new data files which can be further filtered by applying additional filters and analysis tools. Doing so reduces the size of the files used and thus accelerates subsequent analyses.

A major advantage of mAdb is the integration of open source tools,

for example, the R statistical language to perform statistical analyses directly on the array data (k-means, principal component analysis and others), ImageMagick to reformat and visualize the output (interactive scatter plot, multiple array graphical viewer) or Perl enabling scripting of analyses. Data sets can be retrieved and formatted for specific applications suites. The integration of external data source for annotation like Unigene, GenBank, and others means that reports can automatically be enriched with information from other genomic and pathway databases.

A different approach for the analysis of gene expression or microarray data is taken by providing websites for its analysis. Advantages of this approach clearly are the ease of use, scalability oblivious to the user as well as the absence of any overhead to set up. Whereas these platforms come with a lot of tools already set up, the downside is that they are not easily extensible. Strong emphasis on these platforms is also put on sharing data. While this can enable collaboration, it may not be perceived as advantageous by organizations concerned about intellectual property.

One such database is ArrayExpress Brazma et al. [2006] which allows for the deposition of microarray data. Based on a relational database management system for the storage backend and a Web server for the UI, ArrayExpress provides the basic functionality to analyze single studies but allows for the comparison and analyses across studies. An additional tool, the Expression Profiler allows users to develop analysis pipelines using predefined components like data analysis components for gene expression data preprocessing, missing value imputation, filtering, clustering methods, visualization, significant gene finding, between-group analysis and other statistical components.

A further example is the Genex bioinformatics analysis software for archiving and analyzing microarray data. It provides a framework for documentation and analysis of microarray experiments and data to support individual research laboratories and large companies.

Another example of a public data deposition is ChipDB which supports analyses of single experiments as well as experiment comparisons using hierarchical clustering and multidimensional scaling. It also sup-

ports set comparisons between experiments or between experiments and categorisations of genes. The results of analyses are stored in the database for search and further analysis using R as the language for statistical analyses.

A more comprehensive overview of databases for the analysis of genetic information can be found here Gardiner-Garden and Littlejohn [2001].

6.2 Proteomic Data

Standards for proteomic data, mostly mass spectrometry data, were completely absent in the early years of the field. Various different formats were consequently developed but with almost complete lack of interoperability. Although standards for mass spectrometry data were recently defined, even today most hardware produces proprietary data formats, but with recent standardization efforts, conversions between different formats and standards have been defined.

The first attempt at standardization stems from the Human Proteome Organization-Proteomics Standards Initiative (HUPO-PSI) Mayer et al. [2013], Omenn [2007] which defined several standards over recent years. The standards are broadly adopted to harmonize data from different mass spectrometry hardware.

mzData was the first attempt by the HUPO-PSI to create a standardized format for mass spectrometry data. mzData is an XML format for representing mass spectrometry data in such a way as to completely capture the hardware aspects (hardware as well as configuration used) of an experiment/analysis. The key feature of the format is the use of external controlled vocabularies (also defined by HUPO-PSI) to allow data from new instruments and new experimental designs to be shared in a common format.

mzData files essentially contain data on the use of a mass spectrometer, the data generated, and the initial processing of that data (to the level of the peak list — processed data from a mass spectrometry experiment, i.e., the peaks of the mass distribution of the sample). There can be multiple peak lists in a mzData file, which might be related via

a separation, or just in sequence from an automated run.

The data contained is essentially described in three parts: a reference to the controlled vocabulary used as well as a description and the spectra's (the peak lists — essentially the mass spectrometry data). The reference to the controlled vocabulary serves as a way to structure the descriptions of the experiments. Put differently, the description of a mzData file uses the controlled vocabulary to describe the experiment.

The description of the experiment contains all aspects of it including a description of the hardware used, the input file names of the raw data and generally all aspects to reproduce the experiment (and thus the provenance of the experiment).

Finally, spectra's are a list composed of an array of acquisitions which are either represented as base64 encoded binary (single or double precision) or arrays of simple data types. All arrays used to describe a single spectrum are the same length, with the same indexing, making access simple and very efficient.

Whilst this format is now deprecated and replaced by mzML, it is still broadly used in legacy software.

mzXML Perez-Riverol et al. [2015] has been developed as a standard in parallel to mzData at the Seattle Proteome Center/Institute for Systems Biology and is still in use in the proteomics community. The mzXML schema stores the most important information about an experiment. Similar to mzData, it essentially stores the provenance, i.e., all raw data files used to generate the mzXML file as well as a comprehensive specification of the MS instrument (e.g., resolution, manufacturer, model, ionization type, mass analyzer type, detector type) and acquisition software used to generate the data. A generic element can be used to describe modifications to the instrument used. Figure 6.2 shows an (abbreviated) example mzXML file with elements storing information about the instrument used (`msInstrument`) as well as as the actual data (the scans and peaks of mass in them).

To completely capture the provenance, mzXML also tracks any type of data processing performed during the creation of the document. The description will include the name and version of the program used and a list of the input parameters.

```

<mzXML>
.
.
  <msInstrument>
    <msManufacturer category="msManufacturer" value="ThermoFinnigan"/>
    <msModel category="msModel" value="LCQ Deca"/>
    <msIonisation category="msIonisation" value="ESI"/>
    <msMassAnalyzer category="msMassAnalyzer" value="Ion Trap"/>
    <msDetector category="msDetector" value="EMT"/>
    <software type="acquisition" name="Xcalibur" version="1.3 alpha 8"/>
  </msInstrument>
.
.
  <scan num="20" msLevel="2" peaksCount="43" polarity="+" scanType="Full" retentionTime="PT356.68S"
  collisionEnergy="35" lowMz="223.089" highMz="531.078" basePeakMz="428.905"
  basePeakIntensity="301045" totIonCurrent="764637">
    <precursorMz precursorIntensity="120053">
      445.3466797
    </precursorMz>
    <peaks precision="32" contentType="m/z-int" compressionType="zlib" compressedLen="55">
      eJw7wAABDQzYAbR4ARzyB9DoBhw0.....
    </peaks>
  </scan>
.
.
</mzXML>

```

Figure 6.2: Example mzXML file.

mzXML is based on XML which unfortunately cannot directly incorporate binary data and the conversion to a human readable, clear text representation is not possible without a significant size increase. The mzXML standard addresses this issue by encoding the spectra, i.e., the mass/charge (m/z) intensity binary pairs in base64. Base64-encoded binary data are larger than a binary representation but not considerably and by removing all peaks with an intensity equal to zero the file size becomes manageable.

mzXML has been criticized for not capturing the raw data or experiment design and for not being sufficient for regulatory submission. Combined with the fact that two formats for representing the same information is an undesirable state, a joint effort was set by HUPO-PSI, the SPC/ISB and instrument vendors to create a unified standard borrowing the best aspects of both mzData and mzXML, and intended to replace them.

While the two formats essentially describe the same information, the major difference is that mzXML is rather strict on the vocabulary use while mzData is rather flexible as different controlled vocabularies can be used. The challenges to be resolved are consequently not of a

technical nature but of a philosophical one.

In the design of mzML Deutsch [2008, 2010] the best aspects of both mzXML and mzData were used. The central design aspects were to cover the entire features of both mzXML as well as mzData and also to keep the format simple, i.e., not to add any features beyond the two base formats.

It was consequently deemed best to retain the flexibility of the mzData approach and keeping the controlled vocabulary flexible. In its current form, also mzML references a controlled vocabulary instance. This makes it possible to extend the vocabulary without changing existing XML files and gives considerable flexibility to change the vocabulary in future revisions of the standard. Taking from mzXML, the peaks are represented by binary format data encoded into base 64 strings, rather than human-readable ASCII text for enhanced fidelity and efficiency when dealing with profile data. Although this can increase the file size considerably, it was deemed that readily available compression algorithms can be used to compress the mzML files before transport. Given the high redundancy of information — repetition of tags — any compression algorithm should readily achieve a considerable reduction of file size. Furthermore, in order to enable fast access to the file, mzML was designed with a standardized index for efficient random access, in the same way as mzXML. This enables programs to directly locate a specific spectrum within the file during processing, rather than having to read the file sequentially.

Both aspects of mzML, controlled vocabulary as well as base64 strings, make it straightforward to accelerate access/analysis through indexing. The XML file can be flattened (if the structure/hierarchy is not needed) or shredded with known XML shredding approaches (e.g., Tatarinov et al. [2002]) to store them in a relational database. The attributes can then be indexed with standard database indexes available in most relational database systems.

An alternative to improve read performance (and use of space) but not necessarily indexed access is mz5 Race et al. [2012], Wilhelm et al. [2012] which is based on the features of mzXML and mzML. Although based on very useful and broad controlled vocabularies, relying

on XML for the straightforward implementation of mzData, mzXML, and mzML introduces a major efficiency bottleneck. XML was designed to be a human readable, textual data format with considerable inherent verbosity and redundancy. XML was not designed for efficient bulk data storage and generally requires reading complete files to construct the XML parse tree. The mzXML and mzML formats work around these limitations by using base-64 encoding, compression of the raw data and indexing for random access. Despite the improvements gained from these efforts, proprietary formats generally outperform mzXML and mzML in terms of space requirements as well as in I/O efficiency.

mz5, is a novel data representation that combines the merits of the mzML ontology with the efficiency of the Hierarchical Data Format (HDF5). HDF5 is an industrial standard for efficient storage and retrieval of large amounts of complex data and is based on a portable binary representation. HDF5 was specifically designed for large data sets; the key features used in mz5 are its native support for compression and cached partial read and write access.

HDF5 library implementations are available for a range of languages and computational platforms. Introduced more than 20 years ago, HDF5 is adopted as a prime format for intensive data applications in fields like astronomy, geology, remote sensing, and avionics, and will thus be used and maintained in the long term.

mz5 outperforms the XML-based representations of the mzXML and mzML formats in terms of both space requirements and I/O speed.

6.3 Patient Data

openEHR Kalra et al. [2005] is an open standard for describing the management and storage and exchange of health data in electronic health records (EHRs). Crucially, openEHR describes how all information about a patient is stored in one vendor-independent electronic health record (EHR). openEHR includes the EHR Extract specification but are primarily concerned with the structure of patient data rather than the exchange of data between EHR systems which is the focus of other standards (e.g., HL7). The openEHR specifications are

maintained by the not for profit openEHR foundation which supports research, development, and implementation of open EHRs.

At the core of openEHR is a fairly generic information model that is left deliberately independent of the specification of clinical information. Decoupling the information model and the content clinicians and patients need to record is key in the face of very large, growing, and ever-changing set of information types in health care and informatics.

openEHR defines two distinct layers to describe clinical content: archetypes and templates. Archetypes define groups of reusable data points that may be used in different contexts. Blood pressure measurements, for example, may be used in the diagnostic process of different diseases. An archetype for blood pressure consequently defines all attributes of a blood pressure measurement and can be used in different contexts. Templates, on the other hand, group archetypes and are used to logically represent a use case-specific dataset like, for example, a radiology report which contains different groups of measurements, i.e., archetypes.

Archetypes are usually predefined whereas templates are use case specific. More specifically, templates may be defined on the basis of an organization and are typically defined for GUI screen forms, message definitions and document definitions and thus correspond to operational content definitions.

A template is constructed by referencing relevant items from a number of archetypes. A template might only require one or two data points or groups from each archetype, but they are not allowed to violate the semantics of the archetypes from which they are constructed (i.e., cannot only choose to include a subset of data points).

openEHR only defines the structure of the data but not how it is stored. Several systems implement openEHR, either as part of a larger application or solely to support storage of patient data. Given the absence of any particular specification of the storage (an XML-like format is defined but does not specify how to store it) each implementation stores the information differently Blobel and Pharow [2009], Rinner et al. [2013]. Some implementations use the specified XML representation and rely on a relational database for efficient storage whereas oth-

ers use the relational database directly, i.e., with a table per archetype and per template (with the template using the primary key of the archetype as a foreign key).

Storing the EHRs natively in relations enables the use of indexes on the data. Given the flat data model (only two levels), most queries can be expressed as relational queries and basic indexes can be used to accelerate most predicates. Some implementations of openEHR also use inverted indexes to index all keywords used in EHR's and thereby manage to accelerate full-text searches on all EHR's.

Health Level 7 or HL7 Dolin et al. [2001] is a comprehensive standard to simplify the exchange of electronic health records between software applications of different healthcare providers. HL7 is defined by an international standards organization and are adopted by standards issuing bodies like ANSI and OSI.

Health Level 7 primarily defines the exchange of messages between hospital systems, i.e., it defines the content of the messages as well as their format. For example, it defines the clinical document architecture for exchanging documents, formats for exchanging health summaries, data types which can be used across data and messages and others.

Crucially, by defining messages exchanged between systems as well as the data types used in them, HL7 also defines electronic health records. Whilst it does not necessarily specify how the data is stored explicitly, it defines the message format which is typically also the format in which the data is stored.

HL7 has been heavily criticized for over complicating dealing with health care information. FHIR (Fast Healthcare Interoperability Resources) FHIR has been developed as a response. Instead of attempting to cover all possible use cases and exceptions, FHIR instead only focuses on the most common 80% use cases and simplifies the implementation (e.g., by using JSON and a RESTful approach). It furthermore extends the standard to cover new application areas like mobile healthcare and the integration of new medical device. Figure 6.3 shows an (abbreviated) example of a FHIR document storing patient information.

HL7 and, in the latest version, also FHIR are at the core of multiple open-source and commercial systems like OpenMRS, VistaA and

openEMR de Abajo and Ballesterio [2012] among others. The latter is an open source system for electronic practice management software which has become increasingly popular and thus is also facilitating the adoption of EHR standards for smaller healthcare providers. Similarly, Mirth Connect Bortis [2008] (or the open-source alternative Mirth Open) can be used by bigger healthcare providers to integrate multiple HL7-based systems using a variety of protocols and databases. The core of FHIR is also implemented on top of Postgres with FHIRBase FHIR-Base and as mobile platform Waghlikar et al. [2017].

Most important for medical research arguably is the Clinical Document Architecture (CDA) standard Al-Enazi and El-Masri [2013]. CDA defines the structure of documents including EHR's.

A CDA document has a header and a body. The header conveys the context in which the document was created, and the body contains the informational statements that make up the actual content of the document. The purpose of the header is to make clinical document exchange possible and to support the compilation of an individual patient's clinical documents into an EHR.

The header has four components: (1)

```
<entry>
  <resource>
    <Patient>
      <id value="555-44-4444"/>
      <identifier>
        <use value="official"/>
        <system value="http://ghh.org/patient"/>
        <value value="555-44-4444"/>
      </identifier>
      <name>
        <use value="official"/>
        <family value="Everywoman"/>
        <given value="Eve E."/>
      </name>
      <telecom>
        <system value="phone"/>
        <value value="(206)3345232"/>
        <use value="home"/>
      </telecom>
      <gender value="female"/>
      <birthDate value="1962-03-20"/>
      <address>
        <line value="153 Fernwood Dr."/>
        <city value="Statesville"/>
      </address>
      <active value="true"/>
    </Patient>
  </resource>
</entry>
```

Figure 6.3: Example of an FHIR document.

document information identifies the document and defines the confidentiality, (2) encounter data which describes time location and so on in which an encounter occurred (3) service actors, i.e., those who authenticate the document, those intended to receive a copy of the document, as well as health care providers who participated in the medical services provided and finally, (4) service targets which include the patient and relatives.

The CDA body comprises sections, paragraphs, lists, and tables. These structural components have captions, can nest (so that, for instance, a section can contain a section), and can contain character data, multimedia, and codes drawn from standard terminologies.

6.4 Imaging Data

Digital Imaging and Communications in Medicine (DICOM) Yam et al. [2004], Mildenerger et al. [2002] is a standard for handling, storing and transmitting information in medical imaging. It includes a file format definition and, not necessarily crucial for medical research, a network communications protocol. The communication protocol is an application protocol over TCP/IP to exchange imaging data between systems. The standard is not open, but the National Electrical Manufacturers Association (NEMA) holds the copyright to this standard. Still, many developers of imaging hardware implement the standard.

DICOM enables the integration of scanners, servers, workstations from multiple manufacturers into a picture archiving and communication system (PACS). Each device clearly defines what DICOM standard it complies with, thereby facilitating the exchange and integration. DICOM has been widely adopted by hospitals.

The DICOM standard essentially groups the imaging information with metadata. To strengthen the link between image(s) and the metadata — and to avoid that metadata and image are separated by accident — some of the metadata is embedded into the imaging information. While this indeed ties metadata and image together, it makes anonymization of the imaging information a challenge.

A DICOM data object consists of a number of attributes, including

items such as name, ID, etc., and also one special attribute containing the image pixel data. A single DICOM object can have only one attribute containing pixel data. For many different types of imaging data, this means that only a single image can be used in a DICOM object. To store multiple images. However, the standard defines the use of frames such that multi-frame data can be stored. Another example is nuclear imaging data, where a nuclear medicine image (tomography and others), by definition, is a multi-dimensional multi-frame image. In these cases, three- or four-dimensional data can be encapsulated in a single DICOM object. DICOM allows using compression of the imaging data. Standard method including JPEG, Lossless JPEG, JPEG 2000, and Run-length encoding (RLE) can be used for the imaging data while LZW (zip) compression can be used for the whole data set.

The DICOM standard is very flexible in that many fields of the standard are optional. While this can be useful to cater for many different use cases, this is also the main criticism of it: in many cases, the metadata is underspecified and/or not uniform, making integration difficult.

As such, the metadata used for DICOM can be understood as an ontology or controlled vocabulary to describe the imaging data. DICOM metadata includes very basic fields used to describe the imaging data (format, alignment, resolution and others), information about the hardware (manufacturer, version, firmware etc.), the compression used (algorithm and parameters), patient information (name, identifier, date of birth, gender, ethnicity, weight, address and others) as well as fields in other areas (e.g., more detailed information on the imaging method etc.).

Accelerating access to DICOM files through indexing has to be done through two approaches: through indexing, the metadata as well as through indexing the imaging data. Indexing the metadata is straightforward: the metadata fields can be readily mapped to relation tables and indexes over frequently used attributes/fields can be defined to accelerate access. To accelerate access to DICOM files via the age of patients, for example, a B-Tree index on the date of birth field can be defined.

An alternative approach is to use a full-text search as in Dicoogle. Dicoogle uses Apache Lucene as the full-text indexing engine and offers to index (and ultimately search) DICOM images through a full-text search of the metadata. It also offers an inverted index which maintains a file with all words found across documents and pointers to the DICOM file.

Dicoogle supports different types of queries. In the very simplest case, users can simply execute a full-text search on the metadata without any prior knowledge of the fields and what they store. Additionally, however, it is possible to query by keywords that must occur in the context of specific structural parts of the DICOM standard as well to obtain more targeted results. For instance, it could also be possible to query for keywords in a specific DICOM tag element. Second, it is possible to query content only, as in a common information retrieval system. Using the two previous search types, one can build mixed queries where content and the DICOM structure are combined.

Indexing the imaging data is more challenging and essentially can be reduced to extracting features from the images or image similarity search. Both heavily depend on the particular use case, e.g., feature extraction for brain images is considerably different than for other body parts. A generic approach is the use of Octrees Jackins and Tanimoto [1980] on raster images (e.g., JPEG as is used in DICOM). Octrees divide space uniformly and can thus be used as a hierarchical representation of similar cluster in the raster image. By hierarchically clustering the image, it can be compressed and, more importantly, compared with other hierarchical representations.

While most imaging today is stored in the DICOM format, a substantial share is still kept in DICOM's precursor ACR-NEMA Spilker [1989]. ACR-NEMA is made up of a number of parts. The first describes a common image data format, which can be understood by equipment that complies with the standard. Commonly, each imaging system within a hospital generates images in a slightly or vastly different format than the others, regardless of modality or manufacturer. The image format portion of the standard will allow manufacturers to understand images generated by each other's systems. Only conversion

of each manufacturer's internal image format.

ACR-NEMA also defines a set of metadata fields that can be used to describe the imaging data. The fields primarily revolve around the image features such as geometry, orientation, resolution and other aspects and, compared to DICOM, do not allow to store other aspects of patient information and so on.

A less important part from the perspective of the analysis of imaging data is the specification of the physical connector used to retrieve the imaging. While still a large fraction of imaging data is stored in ACR-NEMA, the hardware producing the old format is no longer used, and this part of the specification is deprecated.

6.5 Other Types

While the previously discussed standard capture one particular aspect of medical analyses, CDISC Kuchinke, Aerts, Semler [2014] attempts to structure and define lab results in general. This is particularly useful for medical studies which touch on several different aspects, i.e., include imaging data, health records and so on.

The Clinical Data Interchange Standards Consortium (CDISC) is working towards the development of standards to streamline data collection primarily in medical research but also in healthcare. The CDISC suite of standards supports medical research of any type from protocol through analysis and reporting of results. The CDISC standards are harmonized through a model that is now not only a CDISC standard but also an HL7 standard.

CDISC defines several aspects of studies. For example, it defines data models for studies, for analyses, for lab data, as well as operational data. Beyond data models, it also specifies a standard to store the data in XML as well as message formats to exchange the data between systems.

Most relevant in the family of CDISC standards for medical research are the standard defining the study data tabulation model (SDTM). This standard governs how the data about longitudinal studies is structured. It is built around the concept of observations collected from

subjects who participated in a clinical study. Each observation can be described by a series of variables, corresponding to a row in a dataset or table. Each variable can be classified according to its *role*. A role determines the type of information conveyed by the variable about each distinct observation and how it can be used. Variables can be classified into identifiers (identifiers for the study, study subject, domain, etc.), topics (description of study and other text attributes), timing (start and end of observation) as well as qualifiers which ultimately are the variables storing information about observations.

Most observations collected during a study are divided among three general observation classes: interventions, events, or findings.

Interventions are treatments, i.e., therapeutic treatments and surgical procedures, that are administered to patients as specified by the study protocol coincident with the study assessment period or other substances self-administered by the patient. Events are occurrences or incidents independent of planned study evaluations occurring during the study or prior to the trial (e.g., the medical history) of the patient). Finally, findings are probably the most important aspect SDTM as they capture the observations resulting from planned evaluations to address specific questions such as observations made during a physical examination, laboratory tests, ECG testing, etc.

SDTM is already structured as a tabulation model. It thus follows directly that the model can easily be translated/flattened into a relation table. CDISC's SDTM defines not only the structure but also the vocabulary, for example, of the findings, events, and treatments, making the translation into a relation table even easier. Based on the relational model the known indexes can be used to accelerate access and analyses.

7

Towards a Medical Data Lake

Many systems have been developed to integrate heterogeneous medical data stored federated. In the following, we first discuss centralized examples (e.g., data warehouses) as well as distributed systems and infrastructure.

7.1 Centralized Medical Data Repositories and Infrastructures

The neuGRID Redolfi et al. [2009] infrastructure is designed to support storage and analysis of clinical data/images collected from subjects specifically studied for neuGRID, previously collected in different research projects that will be archived in neuGRID as well as previously collected in different research projects that will be analyzed but not archived in neuGRID.

As the name implies neuGRID builds on a grid, i.e., a distributed compute infrastructure which, however, resides in one location (and is thus not federated). The user-facing services include applications for visualization, image analysis, browsing of neuroscience images. These services are built on a second level infrastructure which includes work-

flow, provenance, querying services. The latter make the entire platform extensible as new services can be easily built on the second level infrastructure. At the bottom is the basic infrastructure building on the distributed grid platform. The bottom layer infrastructure entails data access, execution as well as storage management.

NeuGRID's workflow and pipeline service is a key element in providing generic services for research analysis. Neuroimaging pipelines allow neuroscientists and clinicians to apply series of automated transformations and processes on brain images for analysis. These processes are very computer intensive and deal with large amounts of data. This service enables scientists to create and design workflows in a user-friendly fashion, calling and combining automatically different algorithms (e.g., FMRIB Software Library, BrainSuite, FreeSurfer, MNI tools, Automated Image Registration, Analysis of Functional NeuroImages and many other programs) in a single environment.

To achieve interoperability as well as extensibility, the pipeline service is designed to allow multiple workflow authoring tools to define standardized workflow specifications including the LONI pipeline workflow management system and other scientific workflow applications (such as Kepler). The developed workflows generated will be transformed into a common format that can be executed in multiple execution environments on the lowest level infrastructure.

Users can extend and modify workflows themselves by downloading them to their workspace and edit them by modifying algorithms, changing input data sources and other settings. From within the portal service, the user has the opportunity to submit the workflow for execution and visualize the results. NeuGRID final users will browse for available workflows through the querying service. Finally, to execute the user-defined pipeline in a grid environment and to fully exploit the grid capabilities, the pipeline will be transformed to be executed on the grid.

The Alzheimer's Disease Neuroimaging Initiative (ADNI) Jack et al. [2008] is a research effort to provide a clinical data infrastructure for the study of the principle, prevention, and treatment of Alzheimer's disease. Multiple research groups contribute their findings of the bio-

logical markers to the understanding of the progression of Alzheimer's disease in the human brain.

A cornerstone of the initiative is the commitment by all groups to publish their data and findings as soon as possible without waiting for the completion of their research, thereby making it available to others earlier than in traditional research collaborations. Collaborators also agree to cede any patent opportunities.

ADNI itself does not necessarily provide infrastructure or services to the neuroimaging or Alzheimer's community but ensures that data capture is uniform. ADNI captures and stores data across multiple sites and to keep the data consistent across sites (e.g., image quality, acquisition time and others). To this end, ADNI provides the tools for electronically capturing the data. More precisely, it provides the protocol and interfaces for a consistent capturing of the medical data.

7.2 Distributed Medical Data Systems

Many federated and distributed systems have been developed to analyze medical data. In the following, we will discuss the most prominent of them.

The Shared Health Research Information Network (SHRINE) Weber et al. [2009] is a prototypic implementation of federated systems enabling investigators to search the electronic health records of patients across multiple independent Harvard hospitals in real-time.

The prototype SHRINE architecture consists of a query aggregator hosted on a central server and SHRINE adapters located at each hospital. The query aggregator receives user queries and broadcasts them to each of the adapters and receives the results back from each health center. The purpose of the adapters is to translate the query into a format that is compatible with each hospital's local source databases, thus hiding the complexity of the local databases from the rest of the SHRINE. The prototype implemented so far includes three adapters, though in theory any number of other health centers could be added, potentially requiring additional adapters.

The SHRINE architecture does not make any restrictions on the

schemas at the local institutions. Although adapters must accept the standardized query definition format, a hospital may make any customizations needed to make the adapter work on the local database schema. This can be done in one of two ways: the adapter can be modified to work with the local databases, or the local databases can setup to work with the adapter. The latter approach was followed for the prototype but puts severe restrictions on the local hospitals. More particularly, the local databases were set up using an i2b2 schema, meaning that the adapters only required minimal changes.

Besides the technical aspects of SHRINE, what makes the system interesting is also the question of complying with ethics guidelines set out by the hospitals. SHRINE addresses the issues of data privacy by simply only allowing for counts to be returned from individual sites. While this does not allow for very interesting query results, it prevents the release of data about individuals and is what made SHRINE pass ethics committees (as well as complying with legal requirements).

tranSMART Athey et al. [2013], Schumacher et al. [2014] is a knowledge management platform that enables scientists to develop and refine research hypotheses by investigating correlations between genetic and phenotypic data, and assessing their analytical results in the context of published literature and other work.

The initial tranSMART platform integrates 'omics' data in a single database backend. It provides tools for loading different data types, chemical, biological and clinical data, from a range of sources. The platform focuses on integration unification of this data: all data imported is mapped on the CDISC and ICD-10 standards which are then used as the schema made available to the scientists through a web interface. The tranSMART platform implements a set of data models, shared datasets, data transformation utilities, and analytical web applications all centered on the patient, to enable discoveries by creating a standardized and semantically integrated data warehouse.

tranSMART features an API which allows plugging in of external components for analysis and visualization. Currently tranSMART interfaces with external packages for extended data viewing (Genome Browser, etc.), statistical analysis (R) and visualization and analytics

(Spotfire). These may be part of the distributed open source platform or distributed privately by vendors who are using our open API.

Initially designed as a data warehouse means that an inherent assumption of the design is that no ethical or legal requirements must be met. Put differently; it is assumed that all data is already anonymized or consent of its use has already been given. tranSMART thus does not address the issue of data privacy directly.

Only recently did a federated version of tranSMART become available. The new version is able to integrate different databases in the backend. The architecture is straightforward: all databases must use the same schema and are connected using standard mechanisms, i.e., JDBC. Although now diverse data sources located at different organizations can be integrated, no provisions are made to address data privacy issues. It is still assumed that all participating organizations agree to the use of the data.

The EU has significantly invested into infrastructure providing researchers from all fields with state-of-the-art instruments and services, thus enabling research which produces massive amounts of data in recent years. With this development came the need to also provide infrastructure to store, maintain and make the data available to researchers.

To this end, the EUDAT Lecarpentier et al. [2013] initiative was launched in 2011. Although research communities from different disciplines typically have different approaches and requirements particularly with respect to data organization and content, they also share many basic service requirements. Based on this insight, EUDAT has started to develop data services to enable collaborative data infrastructure across disciplines.

With the input of several different disciplines, EUDAT has developed four generic services: geographic data replication (to prevent data loss in long-term archiving and preservation, to optimize access for users from different regions and to bring data closer to computers), data staging to computer facilities (to easily move large amounts of data between EUDAT storage and high-performance computing resources), metadata (catalogue to find datasets) and easy storage. A number of enabling services, such as distributed authentication and authoriza-

tion, persistent identifiers, hosting of services, workspaces and center registry, have also been developed.

EUDAT is currently used for a number of EU projects in the fields of linguistics, solid earth sciences, climate sciences, environmental sciences, biological and medical sciences.

The Integrated Rule-Oriented Data System (iRODS) Rajasekar et al. [2010] similarly provides the software infrastructure to integrate distributed or federated data collections. More precisely, it facilitates storing, searching, organizing, and sharing files and datasets that are large and complex. iRODS can be used for small, single machine set ups but, originating from the supercomputing environment, it also scales to large-scale, massively distributed deployments over hundreds and thousands of machines where it integrates massive numbers of files over geographically distributed sites.

iRODS fundamentally provides the infrastructure to integrate data or files on multiple or thousands of data sources (more precisely, servers). It provides an overlay over a network of storage sources featuring metadata servers. All files are registered (along with a description of them) in metadata directories, making them discoverable. With this iRODS provides a logical file system on top of existing file systems on each of the storage servers, thus presenting users with a single global logical namespace.

A crucial element of iRODS is rules. iRODS uses a rule engine to evaluate rules specified in an iRODS specific language featuring many well-known programming language constructs like loops, conditional statements and so on. The rules are used to express policies. iRODS features policies for access control to limit who can access particular subsets of the entire data at what time, data migration and backup to replicate and backup subsets of data, data conversion and preparation so data can be moved between different formats as well as simple preparation tasks like eliminating duplicate data, chunking or removing null values.

iRODS generally works on the abstraction level of files in that it provides a logical file system on top of other filesystems. Rules, however, can also be used to extract metadata and so files can also be made

discoverable by defining rules on how to extract interesting features of the file contents and registering it in the metadata directories.

8

Summary and Outlook

The increasingly automated recording of health records, the abundantly available high-precision diagnostic devices (producing massive amounts of data) as well as the increasing public availability of medical trials or studies by pharmaceutical companies lead to massive amounts of diverse data available to medical researchers. It is, however, not only the availability of data which makes this the right time to enable a medical revolution through moving from symptom- to evidence-based medical diagnosis (and treatment). The major factor enabling this paradigm shift is the available tools, infrastructure, and standards. Surely, a lot of work remains to be done, but a lot of what is needed has been put in place in recent years.

Tools, algorithms, and standards we have today surely need to be improved, but major advances have already been made, giving us a solid basis to start from. We will summarize state of the art as well as discuss the areas of improvement in the following.

8.1 Current State

The technical challenges in mining medical data are well understood, and a lot of work has been done to tackle them in recent years.

The need for data privacy particularly in the context of medical data is well understood. Legislation and guidelines have been implemented in recent years. The requirements to legally use medical data for research are well defined in the US. Laws regulating the use of data have also recently been passed in the EU but only mandate a minimum standard such that each country can still implement stricter laws. However, as discussed, designing the architecture of distributed systems integrating medical data properly, i.e., only allowing the exchange of aggregates between sites, allows to address the issue of varying local legislation.

Medical data is inherently inconsistent, be it because it comes from different instruments or because it is captured manually. The problem of data cleaning as well as integration, however, is well understood in general and many tools can be readily used for medical data as well. Ontologies and standards defining controlled vocabulary for most areas in medicine (diagnostics and treatments) as well as medical studies are available and facilitate the integration of medical data from different sources. With the available standards (as well as the mappings already defined between them) a considerable share of existing data can be reused and can readily be integrated. Clearly, integration is only possible if everybody uses standards for labeling it (or at least can map to them with little effort).

Standardization efforts have also helped to homogenize file formats. Over the years different hardware manufacturers, organizations and sometimes also individual labs have developed their own formats or at least own extensions. Integration with so many different formats is a challenge, but the efforts of standardization bodies have paid dividend and tools have to be developed (or updated) for only few file formats today.

Despite the efforts, lack of structure and standards for file formats in the medical sciences has long been a problem making the ingestion of large amounts of data into relational databases challenging. The heterogeneity made the import very difficult. With the increasing prevalence

of systems to analyze data in-situ, in its place and in its original format, analyzing heterogeneous data has become considerably simpler. Only little code has to be written to analyze the medical data in-situ. An additional benefit is the scalability of the approach: most available systems can easily scale out on a cluster of machines.

More complex processing of medical data is already well supported by sophisticated workflow execution engines. Workflow engines have been developed over decades and have matured considerably. Many have particularly been developed in support of the sciences and thus come with many tools developed for the analysis of datasets particularly found in biology and medical sciences that can be used readily.

Many of the technologies, tools, and standards developed have been used to develop new systems for the integration and analysis of medical data. Most systems available today have been developed for a particular purpose. While this, unfortunately, means they only support a limited number of analysis approaches and file formats with little potential for extension, it also means they are readily available and can be used instantly with little effort (particularly because they are typically available as web-based platforms).

8.2 Open and Future Challenges

While a lot has been accomplished in the past, several major challenges remain. In the following, we discuss open and future challenges centering on data harmonization, data privacy, and systems available.

One major issue remains the harmonization of ontologies and standards. Several high-level standards like SNOMED CT or ICD-10 have been developed by international organizations and are based on a broad consensus. Their broad adoption across healthcare systems and providers in different countries makes the matching and integration between schemas of different countries considerably simpler particularly because the interfaces and relationships between different standards are well understood, and the mappings are to a large degree well defined.

Still, rarely are hospitals (and pharmaceutical companies even less so) required to adopt particular standards to describe patient data or

clinical studies. While many hospitals use standardized ontologies — usually in the absence of any better solution — they typically amend them with local extensions, turning them into dialects. Dialects do have considerable overlap with the underlying standard, making the mapping simpler. They do, however, also have a minor part which is different in every local instance, requiring a considerable effort to define this many-to-many mapping.

A related challenge is the rapid change with respect to instruments and file formats. New techniques, an excellent example is mass spectrometry, pop up quickly, making new file formats, standards and ontologies necessary. In the first couple of years after a new analysis technology emerges, a vast number of new file formats, definitions, and ontologies become available. This is not only due to different manufacturers but also due to research labs and institutions, each of which develops a new format or standard. Particularly in the first few years the formats are unstable and go through considerable change very rapidly. The effort to make sense of this data and to integrate it is substantial, yet it is precisely the data produced in the first few years in which a new analysis technique is available which is of most interest. Early consensus of what new techniques will be key is crucial and will enable putting effort into harmonization, thereby curbing the effort of harmonizing between a plethora of formats early on.

It is vital to protect privacy of data pertaining to individual patients or subjects in studies. What is crucial from the medical research perspective is that the regulations are clearly defined and that the balance between enabling medical research as well as protecting privacy is found. After all, medical research, particularly at public universities, is rarely done for the sake of itself but typically at least with some aspiration to help the greater good.

Today, we lack both, clear rules as well as a good balance. Anonymization and de-identification requirements are well defined in the United States (and other countries) where HIPAA clearly defines the requirements which de-identification has to follow. Still, a case for the usefulness of the research proposed has to be made in front of the ethics committee of the data owner, but once it is, the HIPAA rules

provide a safe harbor which when strictly followed will keep the medical researchers on safe, legal grounds. Although HiPAA is criticized for incurring considerable cost for the adopting health care providers, the de-identification rules are not associated with this criticism and can generally be used independently of the standard.

The legal situation in the rest of the world and particularly in the European Union is much less clear. While the EU is defining legal standards for its member states, they only define a minimal standard, and they are also a moving target. Concerning the first point, the EU is only defining the minimal standards which need to be followed by member states. Each member can implement more restrictive national laws, thereby rendering the EU guidelines of limited use from a medical research perspective. Laws in France, for example, mandate that no data is allowed to leave the country whereas local laws in Germany make removing of particular fields in the data necessary. The latter makes data integration particularly difficult and challenges integration of data across borders. Furthermore, the legal requirements in the EU are currently not yet clearly defined — several proposals of varying degrees of limitations are discussed — and are putting medical researchers who work with medical data in an uncertain situation.

Finally, the current proposals put forward by the European Union are considered too strict and impede medical research. Several universities, non-profit and charitable organizations, research labs and pharmaceutical companies are openly critical of current proposals for data privacy guidelines of the EU. It appears as if the EU is heeding their input, but we will not know for sure until the guidelines are agreed upon by the member states.

An open challenge is the many tools and systems that keep popping up. Many of the systems made available have a very narrow focus in terms of data/information used as well as analytics offered. They are typically developed for a single format and a single purpose and cannot be used for anything else. The tight connection to a few data formats means that they are quickly out of date and cannot be used any longer. Furthermore, the tools typically have no clearly defined interfaces and cannot be made to work together to exchange data and so on.

The major problem is the poor use of established software engineering as well as database management techniques. For example, the data model is rarely clearly separated or decoupled from the application logic, making it very challenging to change either and particularly the data model: updating to a new version of a file standard has far-reaching implications for the application code and sometimes even for the user interface.

Most tools and software packages and tools developed today are web-based applications that allow to import data files and to perform analyses. Most tools integrate a few existing tools to provide users with analysis capabilities. Useful as this is, the problem typically is that the analysis capabilities cannot be extended as the web interface is not modular. Updating already integrated tools or integrating new ones becomes a major challenge and is not easily possible, rendering the web application of limited use a few years after initial development.

Finally, novel applications and tools introduce new challenges for data management. Sensor-based data generated by patients through using mobile devices like Apple's HealthKits, CareKit, Google Fit Terry [2015] and others will generate large amounts of data for years to come. While integrating and mapping the data to known standards will not entail major challenges (known standards can easily be extended, new ones created) and commercial solutions for integration into EHRs already exist Validic, Redox, its quality as well as finding patterns in streaming data will challenge current methods. Although considerable work in both data cleansing and time series analytics Hamilton [1994] (finding interesting patterns in time-based measurements), the amounts of data collected as well as their limited quality — as they are typically produced on cheap sensors and transmitted using lossy compression due to limited bandwidth — mean new methods need to be developed.

References

- The Effect of Mismatching on the Measurement of Response Error. *Journal of the American Statistical Association*, 60(312), 1965.
- Assessment of Commercial NLP Engines for Medication Information Extraction from Dictated Clinical Notes. *International Journal of Medical Informatics*, 78(4), 2009.
- Enis Afgan, Dannon Baker, Nate Coraor, Brad Chapman, Anton Nekrutenko, and James Taylor. Galaxy CloudMan: Delivering Cloud Compute Clusters. *BMC Bioinformatics*, 11(Suppl 12), 2010.
- Thamer Al-Enazi and Samir El-Masri. HL7 Engine Module for Healthcare Information Systems. *Journal of Medical Systems*, 37(6), 2013.
- Ioannis Alagiannis, Renata Borovica, Miguel Branco, Stratos Idreos, and Anastasia Ailamaki. NoDB: Efficient Query Execution on Raw Data Files. In *International Conference on Management of Data (SIGMOD)*, 2012.
- Ioannis Alagiannis, Stratos Idreos, and Anastasia Ailamaki. H2O: a Hands-free Adaptive Store. In *International Conference on Management of Data (SIGMOD)*, 2014.
- Aptible. Aptible. URL <http://www.aptible.com/>.
- Arvind Arasu, Spyros Blanas, Ken Eguro, Raghav Kaushik, Donald Kossmann, Ravishankar Ramamurthy, and Ramarathnam Venkatesan. Orthogonal Security with Cipherbase. In *Biennial Conference on Innovative Data Systems Research (CIDR)*, 2013.

- Brian D Athey, Michael Braxenthaler, Magali Haas, and Yike Guo. transMART: An Open Source and Community-Driven Informatics and Data Sharing Platform for Clinical and Translational Research. *AMIA Summit on Translational Science*, 2013.
- Atomwise. Atomwise. URL <http://www.atomwise.com//>.
- Shivnath Babu and Herodotos Herodotou. Massively Parallel Databases and MapReduce Systems. *Foundations and Trends® in Databases*, 5(1), 2013.
- Amos Bairoch, Rolf Apweiler, Cathy H. Wu, Winona C. Barker, Brigitte Boeckmann, Serenella Ferro, Elisabeth Gasteiger, Hongzhan Huang, Rodrigo Lopez, Michele Magrane, Maria Jesus Martin, Darren A. Natale, Claire O'Donovan, Nicole Redaschi, and Lai-Su L. Yeh. The Universal Protein Resource (UniProt). *Nucleic Acids Research*, 33:154–159, 2005.
- Sumeet Bajaj and Radu Sion. TrustedDB: A Trusted Hardware-based Database with Privacy and Data Confidentiality. *IEEE Transactions on Knowledge and Data Engineering*, 26(3), 2014.
- Philip a Bernstein, Jayant Madhavan, and Erhard Rahm. Generic Schema Matching , Ten Years Later. *Very Large Database Conference (VLDB)*, 4 (11), 2011.
- Raghav Bhaskar, Srivatsan Laxman, Adam Smith, and Abhradeep Thakurta. Discovering Frequent Patterns in Sensitive Data. In *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2010.
- Bernd Blobel and Peter Pharow. Analysis and Evaluation of EHR Approaches. *Methods of Information in Medicine*, 48, 2009.
- Olivier Bodenreider. The Unified Medical Language System (UMLS): Integrating Biomedical Terminology. *Nucleic Acids Research*, 32, 2004.
- S. Bonner, A. S. McGough, I. Kureshi, J. Brennan, G. Theodoropoulos, L. Moss, D. Corsar, and G. Antoniou. Data Quality Assessment and Anomaly Detection via Map/Reduce and Linked Data: A Case Study in the Medical Domain. In *International Conference on Big Data (Big Data)*, 2015.
- Gerald Bortis. Experiences with Mirth: An Open Source Health Care Integration Engine. In *International Conference on Software Engineering (ICSE)*, 2008.
- Alvis Brazma, Misha Kapushesky, Helen Parkinson, Ugis Sarkans, and Mohammad Shojatalab. Data Storage and Analysis in ArrayExpress. *Methods in Enzymology*, 411, 2006.
- Paul G. Brown. Overview of SciDB. *International Conference on Management of Data (SIGMOD)*, 2010.

- Peter Buneman, Adra Gascon, and Dave Murray-Rust. Composition and Substitution in Provenance and Workflows. *USENIX Workshop on the Theory and Practice of Provenance (TaPP)*, 2016.
- Michael J. Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. WebTables: Exploring the Power of Tables on the Web. In *International Conference on Very Large Databases (VLDB)*, 2008.
- Mary Elaine Califf and Raymond J. Mooney. Relational learning of pattern-match rules for information extraction. In *National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*, AAAI/IAAI, 1999.
- S. P Callahan, J Freire, E Santos, C. E Scheidegger, T Vo, and H. T Silva. VisTrails : Visualization Meets Data Management. In *International Conference on Management of Data (SIGMOD)*, 2006.
- James Castro-Edwards. The Proposed European Data Protection Regulation. *Journal of Internet Law*, 17(3), 2013.
- Rhonda Chaytor and Ke Wang. Small Domain Randomization: Same Privacy, More Utility. *International Conference on Very Large Databases (VLDB)*, 3(1), 2010.
- Ying Chen, JD Elenee Argentinis, and Griff Weber. IBM Watson: How Cognitive Computing Can Be Applied to Big Data Challenges in Life Sciences Research. *Clinical Therapeutics*, 38(4), 2016.
- Fabricio F. Costa. Big Data in Biomedicine. *Drug Discovery Today*, 19(4), 2014.
- Datica. Datica. URL <http://www.datica.com/>.
- Susan B. Davidson, Sanjeev Khanna, Sudeepa Roy, Julia Stoyanovich, Val Tannen, and Yi Chen. On Provenance and Privacy. In *International Conference on Database Theory (ICDT)*, 2011.
- Beatriz Sainz de Abajo and Agustín Llamas Ballester. Overview of the most important Open Source Software: Analysis of the Benefits of OpenMRS, OpenEMR, and VistA. In *Telemedicine and E-Health Services, Policies, and Applications: Advancements and Developments*. IGI Global, 2012.
- Christopher De Sa, Alex Ratner, Christopher Ré, Jaeho Shin, Feiran Wang, Sen Wu, and Ce Zhang. DeepDive: Declarative Knowledge Base Construction. *SIGMOD Record*, 45(1), 2016.
- Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *International Conference on Symposium on Operating Systems Design & Implementation (SOSP)*, 2004.

- Jeffrey Dean and Sanjay Ghemawat. MapReduce. 53(1):72, 2010.
- Eric Deutsch. mzML: A single, Unifying Data Format for Mass Spectrometer Output. *Proteomics*, 8, 2008.
- Eric W Deutsch. Mass Spectrometer Output File Format mzML. *Methods in Molecular Biology*, 604, 2010.
- Robert H. Dolin, Liora Alschuler, Calvin Beebe, Paul V. Biron, Sandra Lee Boyer, Daniel Essin, Elliot Kimber, Tom Lincoln, and John E. Mattison. The HL7 Clinical Document Architecture. *Journal of the American Medical Informatics Association*, 8(6), 2001.
- Martin Dugas, Sylvia Thun, Thomas Frankewitsch, and Kai U. Heitmann. LOINC Codes for Hospital Information Systems Documents: A Case Study. *Journal of the American Medical Informatics Association*, 16(3), 2009.
- Cynthia Dwork. Differential Privacy: A Survey of Results. In *Theory and Applications of Models of Computation*, volume 4978. 2008.
- Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography: Third Theory of Cryptography Conference*, 2006.
- Ashley EA. The Precision Medicine Initiative: A New National Effort. *JAMA*, 313(21), 2015.
- Alexandre Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. Limiting Privacy Breaches in Privacy Preserving Data Mining. In *International Conference on Principle of Database Design (PODS)*, 2003.
- Dan Feldman, Amos Fiat, Haim Kaplan, and Kobbi Nissim. Private Coresets. In *Symposium on Theory of Computing (STOC)*, 2009.
- FHIR. FHIR: Fast Healthcare Interoperability Resources. URL <http://hl7.org/implement/standards/fhir/>.
- FHIRBase. FHIRBase. <http://fhirbase.github.io/>.
- Rachael L Fleurence, Lesley H Curtis, Robert M Califf, Richard Platt, Joe V Selby, and Jeffrey S Brown. Launching PCORnet, a National Patient-centered Clinical Research Network. *Journal of the American Medical Informatics Association*, 21(4):578, 2014.
- B.C.M. Fung, K. Wang, and P.S. Yu. Anonymizing Classification Data for Privacy Preservation. *IEEE Transactions on Knowledge and Data Engineering*, 19(5), 2007.

- Srivatsava Ranjit Ganta, Shiva Prasad Kasiviswanathan, and Adam Smith. Composition Attacks and Auxiliary Information in Data Privacy. In *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2008.
- Margaret Gardiner-Garden and Timothy Littlejohn. A Comparison of Microarray Databases. *Briefings in Bioinformatics*, 2(2), 2001.
- Johannes Gehrke. Programming with Differential Privacy. *Communications of the ACM*, 53(9), 2010.
- Robert Gellman. Why Deidentification Fails Research Subjects and Researchers. *The American Journal of Bioethics : AJOB*, 10(9), 2010.
- Belinda Giardine, Cathy Riemer, Ross C. Hardison, Richard Burhans, Laura Elnitski, Prachi Shah, Yi Zhang, Daniel Blankenberg, Istvan Albert, James Taylor, Webb Miller, W. James Kent, and Anton Nekrutenko. Galaxy: A Platform for Interactive Large-scale Genome Analysis. *Genome Research*, 15(10), 2005.
- Yolanda Gil, Simon Miles, Khalid Belhajjame, Helena Deus, Daniel Garijo, Graham Klyne, Paolo Missier, Stian Soiland-Reyes, and Stephen Zednik. Prov model primer. Technical report, W3C, 2012. URL <http://www.w3.org/TR/prov-primer/>.
- Christian Gluud and Lise Lotte Gluud. Evidence Based Diagnostics. *BMJ : British Medical Journal*, 330(7493), 2005.
- Hector Gonzalez, Alon Y. Halevy, Christian S. Jensen, Anno Langen, Jayant Madhavan, Rebecca Shapley, Warren Shen, and Jonathan Goldberg-Kidon. Google Fusion Tables: Web-centered Data Management and Collaboration. In *International Conference on Management of Data (SIGMOD)*, 2010.
- J M Greene, E Asaki, X Bian, C Bock, S Castillo, G Chandramouli, R Martell, K Meyer, T Ruppert, S Sundaram, J Tomlin, L Yang, and J Powell. The NCI/CIT microArray Database (mAdb) System - Bioinformatics for the Management and Analysis of Affymetrix and Spotted Gene Expression Microarrays. *AMIA Annual Symposium*, 2003.
- Shay Gueron. Intel® Advanced Encryption Standard (AES) New Instructions Set. *Intel Corporation*, 2010.
- Alon Y. Halevy. Answering Queries Using Views: A Survey. *The VLDB Journal*, 10(4):270–294, 2001.
- James Douglas Hamilton. *Time Series Analysis*, volume 2. Princeton University Press, 1994.

- Ada Hamosh, Alan F. Scott, Joanna Amberger, David Valle, and Victor A. McKusick. Online mendelian inheritance in man (omim). *Human Mutation*, 15(1), 2000. ISSN 1059-7794.
- R Haux, C Seggewies, W Baldauf-Sobez, P Kullmann, H Reichert, L Luedecke, and H Seibold. Soarian - Workflow Management Applied for Healthcare. *Methods of Information in Medicine*, 42(1), 2003.
- Thomas Heinis and Gustavo Alonso. Efficient Lineage Tracking for Scientific Workflows. In *International Conference on Management of Data (SIGMOD)*, 2008.
- Joseph M Hellerstein. Quantitative Data Cleaning for Large Databases. *United Nations Economic Commission for Europe*, 2008.
- David W. Hollar. Progress Along Developmental Tracks for Electronic Health Records Implementation in the United States. *Health Research Policy and Systems*, 7(1), 2009.
- Ihab F. Ilyas and Xu Chu. Trends in cleaning relational data: Consistency and deduplication. *Foundations and Trends® in Databases*, 5(4):281–393, 2015. ISSN 1931-7883. .
- Thomas R Insel, Story C Landis, and Francis S Collins. The NIH Brain Initiative. *Science*, 340(6133):687–688, 2013.
- John J. Irwin and Brian K. Shoichet. ZINC - A Free Database of Commercially Available Compounds for Virtual Screening. *Journal of Chemical Information and Modeling*, 45(1), 2005. .
- Clifford R Jack, Matt A Bernstein, Nick C Fox, Paul Thompson, Gene Alexander, Danielle Harvey, Bret Borowski, Paula J Britson, Jennifer L Whitwell, Chadwick Ward, Anders M Dale, Joel P Felmlee, Jeffrey L Gunter, Derek L G Hill, Ron Killiany, Norbert Schuff, Sabrina Fox-Bosetti, Chen Lin, Colin Studholme, Charles S DeCarli, Gunnar Krueger, Heidi A Ward, Gregory J Metzger, Katherine T Scott, Richard Mallozzi, Daniel Blezek, Joshua Levy, Josef P Debbins, Adam S Fleisher, Marilyn Albert, Robert Green, George Bartzokis, Gary Glover, John Mugler, and Michael W Weiner. The Alzheimer’s Disease Neuroimaging Initiative (ADNI): MRI Methods. *Journal of Magnetic Resonance Imaging : JMRI*, 27(4), 2008.
- Chris L. Jackins and Steven L. Tanimoto. Oct-trees and Their Use in Representing Three-dimensional Objects. *Computer Graphics and Image Processing*, 14(3), 1980.
- Jha Ashish K. Meaningful Use of Electronic Health Records: The Road Ahead. *JAMA*, 304(15), 2010.

- Dipak Kalra, Thomas Beale, and Sam Heard. The openEHR Foundation. *Studies in Health Technology and Informatics*, 115(28), 2005.
- Hillol Kargupta, Souptik Datta, Qi Wang, and Krishnamoorthy Sivakumar. On the Privacy Preserving Properties of Random Data Perturbation Techniques. In *International Conference on Data Mining (ICDM)*, 2003.
- Manos Karpathiotakis, Miguel Branco, Ioannis Alagiannis, and Anastasia Ailamaki. Adaptive Query Processing on RAW Data. *PVLDB*, 7(12), 2014.
- Manos Karpathiotakis, Ioannis Alagiannis, Thomas Heinis, Miguel Branco, and Anastasia Ailamaki. Just-In-Time Data Virtualization: Lightweight Data Management with ViDa. In *Biennial Conference on Innovative Data Systems Research (CIDR)*, 2015.
- Jessie Kennedy, Martin Graham, Trevor Paterson, and Andy Law. Visual Cleaning of Genotype Data. *Symposium on Biological Data Visualization (BioVis)*, 2013.
- Peter Kilbridge. The Cost of HIPAA Compliance. *The New England Journal of Medicine*, 348, 2003.
- Ios Kotsogiannis, Michael Hay, and Gerome Miklau. Pythia: Data Dependent Differentially Private Algorithm Selection. In *International Conference on Management of Data (SIGMOD)*, 2017.
- Konstantinos Krikellas, Stratis Viglas, and Marcelo Cintra. Generating Code for Holistic Query Evaluation. In *International Conference on Data Engineering (ICDE)*, 2010.
- Ohmann Kuchinke, Aerts, Semler. CDISC Standard-based Electronic Archiving of Clinical Trials. *Methods of Information Medicine*, 48(5), 2014.
- Damien Lecarpentier, Peter Wittenburg, Willem Elbers, Alberto Michelini, Riam Kanso, Peter Coveney, and Rob Baxter. EUDAT: A New Cross-Disciplinary Data Infrastructure for Science. *International Journal of Digital Curation*, 8(1), 2013.
- Dennis Lee, Ronald Cornet, Francis Lau, and Nicolette de Keizer. A survey of SNOMED CT Implementations. *Journal of Biomedical Informatics*, 46(1), 2013.
- Maurizio Lenzerini. Data Integration: A Theoretical Perspective. In *International Symposium on Principles of Database Systems (PODS)*, 2002.
- Todd Lipcon, David Alves, Dan Burkert, Jean-Daniel Cryans, Adar Dembo, Mike Percy, Silvius Rus, Dave Wang, Matteo Bertozzi, Colin Patrick McCabe, and Andrew Wang. Kudu: Storage for Fast Analytics on Fast Data. Technical report, Cloudera, 2015. URL <https://kudu.apache.org/kudu.pdf>.

- Kevin Loney and McGraw-Hill Osborne. *Oracle Database 10g : The Complete Reference*, volume 33. 2004.
- Brenton Louie, Peter Mork, Fernando Martin-Sanchez, Alon Halevy, and Peter Tarczy-Hornoch. Data Integration and Genomic Medicine. *Journal of Biomedical Informatics*, 40(1), 2007.
- Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward a. Lee, Jing Tao, and Yang Zhao. Scientific Workflow Management and the Kepler System. *Concurrency and Computation: Practice & Experience - Workflow in Grid Systems*, 18(10), 2006.
- Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. l-Diversity: Privacy Beyond k-Anonymity. *International Conference on Data Engineering (ICDE)*, 2006.
- S. Majithia, M. Shields, I. Taylor, and I. Wang. Triana: a Graphical Web service Composition and Execution Toolkit. In *International Conference on Web Services (ICWS)*, 2004.
- Santiago Marco-Sola, Michael Sammeth, Roderic Guigó, and Paolo Ribeca. The GEM Mapper: Fast, Accurate and Versatile Alignment by Filtration. *Nature Methods*, 9(12), 2012.
- D. Markonis, R. Schaer, I. Eggel, H. Muller, and A. Depeursinge. Using MapReduce for Large-Scale Medical Image Analysis. In *Second International Conference on Healthcare Informatics, Imaging and Systems Biology*, 2012.
- Henry Markram et al. Introducing the Human Brain Project. In *Procedia CS*, volume 7, pages 39–42, 2011.
- Vivien Marx. Drilling into Big Cancer - Genome Data. *Nature Methods*, 10(4), 2013.
- Gerhard Mayer, Luisa Montecchi-Palazzi, David Ovelleiro, Andrew R Jones, Pierre-Alain Binz, Eric W Deutsch, Matthew Chambers, Marius Kallhardt, Fredrik Levander, James Shofstahl, Sandra Orchard, Juan Antonio Vizcaíno, Henning Hermjakob, Christian Stephan, Helmut E Meyer, and Martin Eisenacher. The HUPO Proteomics Standards Initiative - Mass Spectrometry Controlled Vocabulary. *Database : the Journal of Biological Databases and Curation*, 2013.
- Clement J. McDonald, Stanley M. Huff, Jeffrey G. Suico, Gilbert Hill, Dennis Leavelle, Raymond Aller, Arden Forrey, Kathy Mercer, Georges DeMoor, John Hook, Warren Williams, James Case, and Pat Maloney. LOINC, a Universal Standard for Identifying Laboratory Observations: A 5-year Update. *Clinical Chemistry*, 49, 2003.

- Deven McGraw. Data Identifiability and Privacy. *The American Journal of Bioethics : AJOB*, 10(9), 2010.
- Aaron McKenna, Matthew Hanna, Eric Banks, Andrey Sivachenko, Kristian Cibulskis, Andrew Kernytsky, Kiran Garimella, David Altshuler, Stacey Gabriel, Mark Daly, and Mark A. DePristo. The Genome Analysis Toolkit: A MapReduce Framework for Analyzing Next-generation DNA Sequencing Data. *Genome Research*, 20(9), 2010.
- Wes McKinney. Pandas: a Foundational Python Library for Data Analysis and Statistics. *Python for High Performance and Scientific Computing*, 2011.
- Frank McSherry and Ilya Mironov. differentially private recommender systems: Building privacy into the net.
- Michael L Metzker. Sequencing Technologies - The Next Generation. *Nature Reviews Genetics*, 11(1), 2010.
- Peter Mildenerberger, Marco Eichelberg, and Eric Martin. Introduction to the DICOM Standard. *European Radiology*, 12, 2002.
- Emad A. Mohammed, Behrouz H. Far, and Christopher Naugler. Applications of the MapReduce Programming Framework to Clinical Big Data Analysis: Current Landscape and Future Trends. *BioData Mining*, 7(1), 2014.
- Noman Mohammed, Benjamin C.M. Fung, Patrick C.K. Hung, and Cheuk-kwong Lee. Anonymizing Healthcare Data: a Case Study on the Blood Transfusion Service. *International Conference on Knowledge Discovery and Data Mining (ICKM)*, 2009a.
- Noman Mohammed, Benjamin C.M. Fung, Patrick C.K. Hung, and Cheuk-kwong Lee. Anonymizing Healthcare Data: A Case Study on the Blood Transfusion Service. In *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2009b.
- Sharyl J Nass, Laura A Levit, and Lawrence O Gostin. Beyond the HIPAA Privacy Rule : Enhancing Privacy , Improving Health Through Research. 2009.
- Stuart J Nelson, Kelly Zeng, John Kilbourne, Tammy Powell, and Robin Moore. Normalized Names for Clinical Drugs: RxNorm at 6 years. *Journal of the American Medical Informatics Association*, 18(4), 2011.
- Thomas Neubauer and Johannes Heurix. A Methodology for the Pseudonymization of Medical Data. *International Journal of Medical Informatics*, 80(3), 2011.
- Thomas Neumann. Efficiently Compiling Efficient Query Plans for Modern Hardware. *PVLDB*, 4(9), 2011.

- Frank Austin Nothaft, Matt Massie, Timothy Danford, Zhao Zhang, Uri Laserson, Carl Yeksigian, Jey Kottalam, Arun Ahuja, Jeff Hammerbacher, Michael Linderman, Michael J. Franklin, Anthony D. Joseph, and David A. Patterson. Rethinking Data-Intensive Science Using Scalable Analytics Systems. In *International Conference on Management of Data (SIGMOD)*, 2015.
- Numerate. Numerate. URL <http://www.numerate.com/>.
- L. Ohno-Machado. Informatics Research to Enable Clinically Relevant, Personalized Genomic Medicine. *Journal of the American Medical Informatics Association*, 19(2), 2012.
- Tom Oinn, Matthew Addis, Justin Ferris, Darren Marvin, Martin Senger, Mark Greenwood, Tim Carver, Kevin Glover, Matthew R. Pocock, Anil Wipat, and Peter Li. Taverna: A Tool for the Composition and Enactment of Bioinformatics Workflows. *Bioinformatics*, 20(17), 2004.
- Gilbert S. Omenn. THE HUPO Human Plasma Proteome Project. *Proteomics - Clinical Applications*, 1, 2007.
- PandeLab. Pande Lab. URL <http://pande.stanford.edu/>.
- Rocio Aldeco Perez and Luc Moreau. Provenance-based Auditing of Private Data Use. August 2008.
- Y. Perez-Riverol, J. Uszkoreit, A. Sanchez, T. Ternent, N. del Toro, H. Hermjakob, J. a. Vizcaino, and R. Wang. Ms-Data-Core-API: an Open-Source, Metadata-Oriented Library for Computational Proteomics. *Bioinformatics*, 2015.
- Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. CryptDB: Protecting Confidentiality with Encrypted Query Processing. In *Symposium on Operating Systems Principles (SOSP)*, 2011.
- J C Prather, D F Lobach, L K Goodwin, J W Hales, M L Hage, and W E Hammond. Medical Data Mining: Knowledge Discovery in a Clinical Data Warehouse. *American Medical Informatics Association Annual Fall Symposium*, 1997.
- Alan M. Race, Iain B. Styles, and Josephine Bunch. Inclusive Sharing of Mass Spectrometry Imaging Data Requires a Converter for All. *Journal of Proteomics*, 75(16), 2012.
- E Rahm and Hh Do. Data Cleaning: Problems and Current Approaches. *IEEE Data Engineering Bulletin*, 23(4), 2000.

- Arcot Rajasekar, Reagan Moore, Chien-yi Hou, Christopher A. Lee, Richard Marciano, Antoine de Torcy, Michael Wan, Wayne Schroeder, Sheau-Yen Chen, Lucas Gilbert, Paul Tooby, and Bing Zhu. *iRODS Primer: Integrated Rule-Oriented Data System*. 2010.
- Vibhor Rastogi and Suman Nath. Differentially Private Aggregation of Distributed Time-series with Transformation and Encryption. In *International Conference on Management of Data (SIGMOD)*, 2010.
- Alberto Redolfi, Richard McClatchey, Ashiq Anjum, Alex Zijdenbos, David Manset, Frederik Barkhof, Christian Spenger, Yannik Legré, Lars-Olof Wahlund, Chiara Barattieri di San Pietro, and Giovanni B Frisoni. Grid Infrastructures for Computational Neuroscience: the NeuGRID Example. *Future Neurology*, 4(6), 2009.
- Redox. Redox. URL <http://www.redoxengine.com/>.
- David E. Rex, Jeffrey Q. Ma, and Arthur W. Toga. The LONI Pipeline Processing Environment. *NeuroImage*, 19(3), 2003.
- Renaud Richardet, Jean-CÃldric Chappelier, Martin Telefont, and Sean Hill. Large-scale Extraction of Brain Connectivity from the Neuroscientific Literature. *Bioinformatics*, 31(10), 2015.
- Christoph Rinner, Michael Kohler, Samrend Saboor, Gudrun Huebner-Bloder, Elske Ammenwerth, and Georg Duftschmid. Searching for Document Contents in an IHE-XDS EHR Architecture via Archetype-based Indexing of Document Types. *Studies in Health Technology and Informatics*, 192, 2013.
- Mark A Rothstein. HIPAA Privacy Rule 2.0. *Journal of Law, Medicine, and Ethics*, 2013.
- Sunita Sarawagi. Information extraction. *Foundations and Trends® in Databases*, 1(3), 2008.
- M Sariyar, A Borg, and K Pommerening. Missing Values in Deduplication of Electronic Patient Data. *Journal of the American Medical Informatics Association (JAMIA)*, 2011.
- Jack W. Scannell and Jim Bosley. When Quality Beats Quantity: Decision Theory, Drug Discovery, and the Reproducibility Crisis. *PLOS ONE*, 11(2), 02 2016.
- Kurt Schmidlin, Kerri M. Clough-Gorr, Adrian Spoerri, Matthias Egger, and Marcel Zwahlen. Impact of Unlinked Deaths and Coding Changes on Mortality Trends in the Swiss National Cohort. *BMC Medical Informatics and Decision Making*, 13(1), 2013.

- Axel Schumacher, Tamas Rujan, and Jens Hoefkens. A Collaborative Approach to Develop a Multi-omics Data Analytics Platform for Translational Research. *Applied & Translational Genomics*, 3(4), 2014.
- B Schwartz, P Zaitsev, and V Tkachenko. *High Performance MySQL: Optimization, Backups, and Replication*. 2012.
- N. H. Shah and J. D. Tenenbaum. The Coming Age of Data-driven Medicine: Translational Bioinformatics' next Frontier. *Journal of the American Medical Informatics Association*, 19(1), 2012.
- F. Shahzad, W. Iqbal, and F. S. Bokhari. On the Use of CryptDB for Securing Electronic Health Data in the Cloud: A Performance Study. In *International Conference on E-health Networking, Application Services (HealthCom)*, 2015.
- Stephen Soderland. Learning Information Extraction Rules for Semi-Structured and Free Text. *Machine Learning*, 34(1), 1999.
- Amazon Redshift Spectrum. Amazon Redshift Spectrum. URL <https://aws.amazon.com/redshift/spectrum/>.
- C Spilker. The ACR-NEMA Digital Imaging and Communications Standard: a Nontechnical Description. *Journal on Digital Imaging*, 2(3), 1989.
- Tiberiu Stef-Praun, Benjamin Clifford, Ian Foster, Uri Hasson, Mihael Hategan, Steven L Small, Michael Wilde, and Yong Zhao. Accelerating Medical Research using the Swift Workflow System. *Studies in Health Technology and Informatics*, 126, 2007.
- Michael Stonebraker. The Case for Shared Nothing. *Database Engineering*, 9, 1986.
- Latanya Sweeney. k-Anonymity: A Model for Protecting Privacy. *International Journal Uncertainty, Fuzziness and Knowledge-Based Systems*, 10, 2002.
- Igor Tatarinov, Stratis D. Viglas, Kevin Beyer, Jayavel Shanmugasundaram, Eugene Shekita, and Chun Zhang. Storing and Querying Ordered XML Using a Relational Database System. In *International Conference on Management of Data (SIGMOD)*, 2002.
- Nicolas P. Terry. Mobile Health: Assessing the Barriers. *Chest*, 147(5), 2015.
- Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Ning Zhang, Suresh Antony, Hao Liu, and Raghotham Murthy. Hive - A Petabyte Scale Data Warehouse Using Hadoop. *International Conference on Data Engineering (ICDE)*, 2010.

- Ozlem Uzuner, Imre Solti, and Eithon Cadag. Extracting Medication Information from Clinical Text. *Journal of the American Medical Informatics Association*, 17(5), 2010.
- Validic. Validic. URL <http://www.validic.com/>.
- Jan Van Den Broeck, Solveig Argeseanu Cunningham, Roger Eeckels, and Kobus Herbst. Data Cleaning: Detecting, Diagnosing, and Editing Data Abnormalities. *PLoS Medicine*, 2(10), 2005.
- Wil M P van der Aalst, Lachlan Aldred, Marlon Dumas, and Arthur H M ter Hofstede. Design and Implementation of the YAWL System. *Advanced Information Systems Engineering*, 2004.
- Tassos Venetis, Anastasia Ailamaki, Thomas Heinis, Manos Karpathiotakis, Ferath Kherif, Alexis Mitelpunkt, and Vasilis Vassalos. Towards the Identification of Disease Signatures. In *International Conference on Brain Informatics and Health (BIH)*, 2015.
- Kavishwar B Waghlikar, Joshua C Mandel, Jeffery G Klann, Nich Wattanasin, Michael Mendis, Christopher G Chute, Kenneth D Mandl, and Shawn N Murphy. SMART-on-FHIR implemented over i2b2. *Journal of the American Medical Informatics Association*, 24(2), 2017.
- Fei Wang, Vuk Ercegovic, Tanveer Syeda-Mahmood, Akintayo Holder, Eugene Shekita, David Beymer, and Lin Hao Xu. Large-scale Multimodal Mining for Healthcare with MapReduce. In *International Health Informatics Symposium (IHI)*, 2010.
- Griffin M Weber, Shawn N Murphy, Andrew J McMurry, Douglas MacFadden, Daniel J Nigrin, Susanne Churchill, and Isaac S Kohane. The Shared Health Research Information Network (SHRINE): A Prototype Federated Query Tool for Clinical Data Repositories. *Journal of the American Medical Informatics Association*, 16(5), February 2009.
- M Whirl-Carrillo, E M McDonagh, J M Hebert, L Gong, K Sangkuhl, C F Thorn, R B Altman, and T E Klein. Pharmacogenomics knowledge for personalized medicine. *Clinical Pharmacology & Therapeutics*, 92(4), 2012.
- M. Wilhelm, M. Kirchner, J. a. J. Steen, and H. Steen. mz5: Space- and Time-efficient Storage of Mass Spectrometry Data Sets. *Molecular & Cellular Proteomics*, 11(1), 2012.
- Scott C Withrow. How to Avoid a HIPAA Horror Story. *Healthcare Financial Management*, 64(8), 2010.
- Michael S Wolf and Charles L Bennett. Local Perspective of the Impact of the HIPAA Privacy Rule on Research. *Cancer*, 106(2), 2006.

- World Health Organization. The ICD-10 Classification of Mental and Behavioural Disorders. *International Classification*, 10, 1992.
- Chun Shan Yam, Jonathan Kruskal, Arkadiusz Sitek, and Michael Larson. A Web-based ACR Index for Radiological Diagnoses. *American Journal of Roentgenology*, 183(5), 2004.
- Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, and Ankur Dave. Fast and Interactive Analytics over Hadoop Data with Spark. *USENIX*, 37(4), 2012.