Relational databases have been fundamental to business systems for more than 25 years. Data normalization is a methodology that minimizes data duplication to safeguard databases against logical and structural problems, such as data anomalies. Relational database normalization continues to be taught in universities and practiced widely. Normalization was devised in the 1970s when the assumptions about computer systems were different from what they are today.

The first part of this 2-part series provides a historical review of record keeping inside and outside of computer systems. Based on this background, this article examines the problems associated with data normalization, such as complexity and the difficulty of mapping business records to normalized data in a changing world. The first part then discusses cases for denormalization and explains how the World Wide Web has impacted the creation and exchange of non-normalized business records. The second part of the series will discuss alternative data representations, such as XML, JSON, and RDF to overcome normalization issues or to introduce schema flexibility.

Introduction

This article describes the changing role of data normalization [30] in commercial systems. Since the 1970s, when data normalization was defined, computer technologies and systems and their applications have evolved considerably. In particular in the 1970s, data structures were stable, disk space was extremely limited, and business information was on paper. People and input-output devices were needed to translate the paper into a form that the computer could read, for example, punched cards. Large computers owned by major institutions such as banks had 512K of memory and cost nearly US$2,000,000. A large institution had 10 megabytes of disk space for all its computer systems and data. In the 1970s, the Internet infrastructure was just beginning to be created and the World Wide Web was over a decade away.
Because disk space was extremely limited, it was assumed that only the most current information would be stored and made available to applications. Normalization ensures that each piece of data, such as name, address, or order information, appears exactly once on disk to avoid data anomalies and preserve space. Typically, normalized data exists only in computer systems and does not match the original business representation of the data. In the 21st century, business data is almost always created in digital form, such as an order message in a web service request. Hence, normalization implies that an existing digital representation of a business record is broken up for storage in a database and later reconstructed for presentation and consumption of these business records.

In this article, the term "business record" is used to mean information that can be shared between two or more parties or components, such as a purchase order, a receipt, a debit note, a financial trade, a bank transfer, an insurance policy, an email, a patient record, a log record, a measurement, a recorded event, a mandated policy or edict and so on.

The assumptions that the relational model is based on have changed. Today, many systems and information structures are no longer simple and fixed but complex and they change rapidly. In today's world, normalization as a process can act as an inhibitor to both the delivery of agile systems and to the agile delivery of systems.

This article provides an introduction to record-keeping through history before and after the introduction of computers for commercial use. A key observation is that throughout history business records were stored "as is", and only the introduction of computers triggered the splitting of records into multiple pieces (normalization). This article then examines the motivation for the development of data normalization in the 1970s. Subsequently, it explains that some degree of data denormalization became a commonly applied compromise. Finally, this article discusses the effect of the web on business records, enabling them to be created in digital format. As a result, it became possible for the first time to store and process business records in computers in their original structure.

Record keeping through history

This section describes aspects of record keeping prior to the introduction of computers, which helps us understand significant changes that computers have introduced. These changes are described later in this article.

Collections of receipts have been found as far back as the 3rd millennium BC in ancient Sumeria \(^1\) in the form of clay tablets that were exchanged and then stored for record keeping. Babylonian loan records have been found from the 18th century BC \(^2\). The code of Hammurabi \(^3\) in Babylon (1792 BC) includes statements that govern the handling and keeping of records (clay tablets). For example:

- If anyone owes a debt for a loan, and a storm prostrates the grain, or the harvest fails, or the grain does not grow for lack of water; in that year he need not give his creditor any grain, he washes his debt-tablet in water and pays no rent for the year.
If anyone buys the field, garden, and house of a chieftain, man, or one subject to quit-rent, his contract tablet of sale shall be broken (declared invalid) and he loses his money. The field, garden, and house return to their owners.

Through history various mechanisms were introduced to record trading information, such as tally sticks \[4\][5], which were cheaper and more readily available than paper. In medieval Europe, a stick was marked with notches and then split lengthwise. The two halves bear the same notches and each party to the transaction received one half of the marked stick as proof. These sticks were then stored and maintained. The split tally stick was in continuous use by the UK government until 1826 for managing taxes. The tally stick store was ordered to be destroyed by burning in 1834 as more modern recording methods were introduced \[5\].

Paper, and in earlier times papyrus and vellum \[6\], were increasingly used through the centuries until the end of the 20th century to record business deals, bills of sale, contracts, and others important documents. Often, the records were signed and sometimes sealed in wax with the marks of the merchants involved. Methods such as double entry book-keeping were introduced in the 15th century. Clerks and scribes supported merchants as the paperwork increased. As computers came into commercial use in the 20th century, businesses began to computerize their systems – which required the conversion of the real-world paper records into a representation that the computer could understand \[7\].

Before computers were introduced, the main principle of record keeping was to capture and maintain an exact copy of the information that was exchanged among the parties involved in a transaction. Often the records were signed or marked in some way and stored "as-is" for future needs. Regulations that govern the storage and handling of business records and contracts have existed throughout history.

**Record keeping in computer systems**

This section describes the environment into which database systems were introduced in the second part of the twentieth century, and the primary purpose of those systems.

As digital computer systems were introduced to support commercial businesses in the 1950s and 1960s, the first records were stored on paper punched cards \[8\], which were often also used for input and output. Human operators typed the content of the paper records that represented the business transaction onto cards, so the computer could read and consume the information (Figure 1). The data stored and processed inside the computer system no longer matched the real business transaction on paper, although it might match how the data was input into the computer in the punched card era.
Figure 1. Data entry staff in the 1950s

Punched cards continued to be used for large quantities of data input and output into computer systems into the 1980s, but magnetic tape [9] and then disk storage [r10] soon replaced punched cards in large systems in the 1960s. With the advent of disk storage (Figure 2), the ability to access data directly and speedily became a possibility, as individual portions of a disk are addressable programmatically. Prior to the existence of disks, most processing took place in batches [r11] where data was processed in the order that it was stored in files on tape or on punched cards. Disks enabled random data access.

Figure 2. Shipping a 5MB IBM hard drive in 1956

In the 1960s, a number of database systems [r12] and direct access file systems [13] were developed to manage data stored on disk that could be accessed and updated by multiple people concurrently, taking advantage of the newly available disk storage. Two of the most common database structures used were the network model (CODASYL) [14] and the hierarchical model (IMS) [r15]. Prior to storing data in the database and implementing an application, a data design was performed by specialist staff (data analysts or database administrators) to de-construct the business data, still on paper in that era, into hierarchies or networks. The analysts produced two data design models, a logical design that maps the business records into hierarchies or networks that programmers access and manipulate, and a physical model that maps the hierarchies or networks to disks. Programmers learned the logical model and accessed the database through navigational programming interfaces (for example, get the next child within the parent) supplied along with the database system for popular programming languages of the time.

In the 1970s the hugely successful relational model [r30] was introduced, which continues to dominate business systems in the 21st century. It stores business data in tables. The relational
model removes the need for navigational access, but still requires data analysts to de-construct the business data into tables which programmers access through a declarative language (SQL). Business data was still paper based in the 1970s and 1980s and had to be transformed, often by scanners or operators re-typing forms. The de-construction of business data typically follows the principles of data normalization \[16\][17] which continues to be taught and used in the 21st century to minimize data duplication and anomalies.

At the time the concepts of relational databases were being defined, a popular disk storage device was the 3330 model 11 which held 200 megabytes and whose purchase price ranged from $74,000 to $87,000 (1970s dollars) \[19\]. When relational databases started taking off in the 1980’s, a very popular disk was the 3380. It was the size of a closet and contained 1.2 gigabytes of storage at the cost of over $200,000 \[20\]. Hence, 1MB of disk storage had a cost of over $160 (1970s dollars), the equivalent of thousands of dollars in 2010 \[21\].

Typically, relational database systems did not maintain the security information associated with signatures, and usually contained any piece of information exactly once – the latest version only, making it difficult to perform audits. It soon became apparent that it was necessary to store copies of the real world business records, for example to be able to audit insurance policies and the associated claims in case of disputes. The document systems were also needed to comply with regulations that require business data to be stored for a certain number of years. A new category of software, called Enterprise Document Management Systems \[23\], was developed in the late 1980s to store images of paper records. These systems were separate from the databases that stored the same data in relational tables. In the 21st century, Enterprise Document Management is called Enterprise Content Management \[24\].

The main principle of record keeping within computers in the twentieth century was to introduce a storage model that suits the way computers work, to store any given data item exactly once, minimizing storage. If an exact copy of the real world paper record was needed, separate systems were be built to do exactly that, causing the same data to be stored multiple times. Regulations to govern the storage and handling of records continued to increase.

**Data normalization process**

This section describes the purpose and effects of the data normalization process first introduced in 1970 with further normal forms introduced through the 1970s.

Data normalization is a methodology to devise a collection of tables that represent real world business records in a database, avoiding any data duplication as storage was expensive. Avoiding data duplication also means that update anomalies cannot happen. Data normalization is very well and widely documented \[18\]. It starts with a single large table that represents all the properties of a real world business record together with its main identifier (a key), then removes hierarchies (repeating groups) to simplify querying with a language such as SQL. Any duplicate data and functional dependencies in the resulting tables must then also be removed.

To achieve normalization, the single table with all the required properties is split into more tables that are linked through primary and foreign keys. The result of data normalization is that a single
business record can be represented in tens or sometimes hundreds of tables. Many artificial keys (and associated indexes) are introduced that do not exist in the real world, but are needed to reconstruct the real world business record. Storing multiple versions of a business record, for example, an order, and then any revisions made to the order, requires versioning all the tables involved which makes querying and maintaining the tables complex. An alternative approach, to conserve storage, is to store deltas only, instead of cascading full versions through the tables, introducing more complexity for programmers.

In 1980, the cost of two MB of storage was roughly equivalent to the cost of one week's work of a computer programmer in the US [19][22]. In 2010, even one GB of storage represents only a tiny fraction, not even minutes of a computer programmer's time, and the storage price continues to drop. Moreover, memory is becoming plentiful, and the cost (latency) of I/O operations continues to decrease as new kinds of storage –such as solid state disks- are being introduced. With the notable exception of relational databases, storage media is typically used to store non-normalized artifacts, for example in file servers, web servers, content repositories, application servers, and so on.

Contrast relational storage with the tablets, tally sticks, and paper records that were used for record keeping before computer systems were introduced and always stored "as is". For several reasons they were not broken up or converted to a different format for storage purposes. First, storage space was usually plentiful and did not have to be conserved. Second, any conversion (and reconstruction) of these artifacts would have been very expensive. And third, storing these records in their original form made it easy to use and understand them when they were retrieved from storage. The same reasons apply today for storing real world digital business records in a non-normalized form as discussed later in this article.

As the use of paper records increased rapidly in the 19th and 20th century, storage space became a problem for some libraries and archives. This triggered the invention of microfilm and microfiche to reduce the required storage space to between 0.25% and 3% of the original material [25]. However, this is merely a form of compression without representing the information in a conceptually different way. Similarly, digital compression can be applied today to reduce the storage consumption of non-normalized business records.

Driven by high storage costs, data normalization represents business records in computers by deconstructing the record into many parts, sometimes hundreds of parts, and reconstructing them again as necessary. Artificial keys and associated indexes are required to link the parts of a single record together. This is a strong contrast to earlier record keeping systems (tablets, tally sticks, paper etc) that store business records as-is. The normalized representations make business records much harder to understand and introduces additional costs for splitting and reassembling them.

**Denormalization process**

This section describes situations where denormalization has become common practice. Database schemas for data warehouses are one example, and new scalable data stores such as Google BigTable [47] and HBase [49] are another example.
Normalization has two inherent drawbacks. First, complex business records often lead to a large number of relational tables in a normalized database schema, which makes the data representation difficult to understand. As a result, writing queries can require many joins and becomes increasingly complicated [46]. Secondly, the potentially large number of joins is detrimental to the performance of data retrieval. Denormalizing normalized tables or using a non-normalized design directly can solve these problems.

**Denormalization in data warehouses**

As the capacity of computing and storage devices increased in the 1980s and 1990s while cost decreased, companies could afford accumulating and analyzing larger volumes of historical business data, such as sales records, in data warehouses. To gain insight into the business performance of a company, such warehouses are used by business personnel that need to run complex queries against an intuitive representation of the data. It was quickly discovered that "the use of normalized modeling in the data warehouse defeats the whole purpose of data warehousing, namely, intuitive and high-performance retrieval of data" [26].

As a result, denormalized star schemas became the most popular database schema for data warehousing. Since data warehouses typically add new data periodically instead of performing transactional updates, denormalization simplifies the schema and improves query performance with little risk of update anomalies.

A star schema consists of at least one fact table, such as "daily sales" containing sales records, and several dimension tables such as "store", "product", "date", and "customer". There is a one-to-many relationship between each dimension and the fact table. Each fact table row contains several measures, that is, numeric columns such as "quantity" or "price", as well foreign keys to all dimension tables to indicate which product was sold in which store to which customer on which date. This is an intuitive business view of the data and facilitates the analysis of (sales) facts according to relevant business dimensions.

The dimension tables are highly denormalized. For example, the "product" table may have columns such as "brand" and "category" where the same string values can appear redundantly for many products. Normalization would use INTEGER values as keys for brands and categories, plus separate tables where the name of each brand category occurs only once. This normalization of dimension tables is avoided, since it would lead to a snowflake schema that becomes harder to understand and introduces additional joins.

The success of star schemas in data warehousing has lead to the general understanding that denormalization has benefits for OLAP and decision support databases. For example, recommendations for data warehouse and OLAP databases in Oracle include "massive denormalization" and "widespread redundancy" [27]. Tests in Oracle 11g have shown that multi-dimensional queries can run 10x to 1000x faster on a denormalized than a normalized database schema [28]. Other studies have explained the performance benefit of denormalization theoretically, using relational algebra and query trees [29].
Despite the success of denormalization for decision support databases, normalization has typically been appropriate for update intensive OLTP applications. However, the need to normalize database schemas for OLTP applications is changing in the 21st century as more and more applications need to keep a complete history of all database rows. Therefore, many applications only perform inserts of new versions of a row and do not perform updates of existing rows [45] – reducing the risk of update anomalies in a denormalized schema, and reducing the need for normalization.

**Denormalization in Google's BigTable, HBase, and other systems**

Google's BigTable is a parallel shared-nothing database system that is implemented as a sparse, distributed, multi-dimensional, sorted map [47]. It is designed for scalability to very large data volumes (Petabytes) and for distribution over hundreds or thousands of computers. In each entry of the map, the BigTable maps a triple consisting of a row key, a column key, and a timestamp to a value. Additionally, column keys are grouped into column families, which form the basic unit of access control and compression.

One of the predominant principles in the design of databases and applications for BigTable is denormalization and data duplication [48]– a radical departure from traditional relational database theory. The intention is to optimize the database for efficient and scalable read access. Denormalization is typically used such that a single read operation can retrieve all fields that belong to a logical business record.

The denormalization in BigTable comes at the expense of more complex and less efficient updates, as potentially multiple copies of the same value must be updated programmatically. This sacrifice is accepted to gain high scalability for applications that exhibit a high read/update ratio. Additionally, the timestamp field in BigTable is used to facilitate versioning, that is, multiple copies of a customer’s address or multiple copies of a product description reflect the state of the world at a certain point in time.

Imagine a database that stores customers and orders, with a logical one-to-many relation between them. While relational database normalization would demand at least 2 tables, one for customers and one for orders, a typical BigTable design would repeat the customer information with each order. This represents the state of the customer information for each particular order. For example, a customer may or may not use the same address for each order. In this sense the denormalized representation resembles an actual purchase order form, that is, the original business record.

Other implementations similar to BigTable include HBase and Cassandra and also rely on a denormalized database design approach [49]. Similarly, other studies have shown that denormalization is a successful technique to build scalable web applications [50].

Since data warehouses are accessed by business users, the stored data needs to intuitively resemble the original structure of the business records, which is achieved by denormalization. Denormalization also improves performance by reducing the number of relational joins needed to evaluate the business records. Similar, denormalization is used in new data stores such as BigTable and HBase to provide simple and scalable data access.
The impact of the web

Starting in the mid 1990s, the digitization of business records coincided with the commercial success of the web. This section describes key technologies of the Web that caused a major change in how business records are represented in the early 21st century.

In 1989, many Internet-based projects were in development as more scientific, academic, government, and commercial institutions had access to the Internet infrastructure. One of the projects included the invention of HTML (Hypertext Markup Language) [31], HTTP (Hypertext Transfer Protocol) [32], and URLs (Universal Resource Locators) [33] that led to the creation of the World Wide Web [34]. HTTP defined a protocol to retrieve and modify HTML documents on the Internet by addressing them through a universal addressing scheme (the URL). Many general purpose viewers (browsers [35]) were built to access and navigate the documents and are used on many devices in the 21st century.

Scientific institutions were the initial users of the fledgling Web to share scientific documents. By 1995 the commercial community discovered the web. As many consumers were beginning to have access to the Internet from their workplaces and homes, there was a race to enable existing commercial systems to the web -- to create a web presence that provided access to the data held in relational databases, so consumers could track packages, or order services and goods. In the past, these activities would have been conducted in person, by phone, or letter. Infrastructures were developed to provide web access to relational database and convert their content to the ubiquitous HTML, and applications were created that used HTML for their user interface [36].

HTML became a huge success. It was described in a Standard Generalized Markup Language (SGML) [38] document definition which was at version 4.0 by 1997. SGML, which originated from IBM's Generalized Markup Language in the 1960s, is an ISO standard for defining markup. A simplification of SGML, called eXtensible Markup Language (XML) [37], was introduced in 1998 to ease parser implementations compared to full SGML parsers that HTML needed. An example of the simplifications in XML is that all start tags have to have end tags, which is not the case in SGML.

The introduction of XML encouraged the creation of many vocabularies beyond HTML to represent arbitrary data structures. As business records in the real world were created in digital form in the late 1990s, fueled by the Web as an input output medium, XML became the natural choice to represent business records in a non-normalized format, much like paper forms or stone tablets are non-normalized business records. Free open source XML software were derived from SGML processors, or brand new processors that could parse well-formed XML were created, removing the need for custom parsers.

More specifications were introduced to support XML, for example XML schemas which enabled institutions and consortia to specify exactly what constitutes acceptable content in a particular business record. Validating parsers became widely available. Namespaces enable a business record to contain data whose definitions are owned by different groups. Namespaces enable institutions to re-use, partition, or extend business record structures. Digital signatures can be
applied to XML, to ensure that it has not been tampered with, in a similar way that signatures and seals were used on vellum and paper.

Specifications were introduced that describe how XML records should be transmitted, including WSDL, SOAP, RSS and Atom. They make it possible to build general purpose frameworks around business record exchange, such as feed technologies and Service-Oriented Architectures (SOA).

The number of consortia that define business record standards in XML for their industry has grown. Corporations are beginning to use the standardized XML structures instead of defining their own XML business records. Examples include the Financial Products Markup Language (FpML) [52], the Financial Information eXchange Protocol (FIXML) [54], EML (Election Markup Language) [55], HL7 (Health Level 7) [56], HR-XML (Human Resources XML) [57], the OTA (Open Travel Alliance) [58], and the Open Applications Group Integration Specification (OAGIS) [53]. Formats such as IS20022 Universal financial industry message scheme [59] used in banking, and UBL (Universal Business Language) [60] are mandated, and subject to regulation, in parts of the world. In many industries, for the first time, it has become possible to store and manipulate business records directly, as in the eras prior to computers with tablets and paper. However, the practice of normalizing the (XML) business records for storage continued.

In the early 21st century, many business records are created and represented in XML. Business records in XML are exchanged between and within institutions through file transfer, HTTP, Web 2.0, and web services. They represent business objects or agreements between two or more parties. A common assumption is that processing XML is inefficient. Hence, many architects continue to design systems that convert business records to normalized relational tables and back, just as they did in 1995 converting between HTML and relational data, and in 1980 converting between paper records and relational data through scanners and human operators.

**Summary**

Up until the advent of commercial computing in the middle of the 20th century, business records were stored and processed the same form in which they were originally created. Examples include stone tablets, tally sticks, and paper forms. With the introduction of computing systems, data normalization was devised to organize business records such that each data item was stored exactly once to conserve storage and avoid update anomalies. Normalization was developed in the 1970s for compelling reasons at the time. Disk space was scarce and expensive, business records were not as complex as today, and only the latest version of each information was expected to be stored. Hence, the effort of converting business records to and from the normalized representation in computers was generally accepted.

With the emergence of data warehouses and business intelligence in the early 1990s, the drawbacks of normalization received attention. A normalized database schema is an unnatural representation of business records, very hard to understand for business users, and inefficient for the formulation and processing of analytical business queries. As result, denormalization was introduced to undo these shortcomings to a certain degree.
And the IT world continues to change in the 21st century. The cost per MB of digital storage space has dropped tremendously. Due to advances in storage density and compression, normalization is no longer required for the sake of saving space. Similarly, audit and compliance regulations require many applications today to retain a history of their data objects. As a result, inserts of new and immutable versions of data objects are often more common than updates of existing data, which reduces the risk of update anomalies. Hence, the need for data normalization is not as universally applicable anymore as it was in the 1970s.

Additionally, the success of the Web, Web Services, and Web 2.0 technologies has ensured that business records are created in digital form, mostly as XML. While client-side software has embraced XML and its derivatives, server-side software involving databases often continues to require significant customization to design, build, and evolve, due to the data transformations that normalization requires. One of the reasons is that data normalization has been taught as a database design methodology for 30 years, and continues to be taught as an integral part of systems design. However, since business records are now digital, more complex, and evolving, it is time to reconsider the use of normalization carefully.

The second part of this series discusses XML and other alternative data representation, and examines when and how they can alleviate common issues with normalization.

References


4. Trade and commerce in the Middle Ages: http://www.camelotintl.com/village/trade.html

5. Tally sticks: http://en.wikipedia.org/wiki/Tally_stick


36. World Wide Web timeline: http://www.w3.org/History.html

37. eXtensible Markup Language: http://www.w3.org/XML/
38. SGML and XML differences: http://www.w3.org/TR/NOTE-sgml-xml-971215


45. Helland, Pat: "Accountants Don't Use Erasers", June 2007


48. "How I Learned to Stop Worrying and Love Using a Lot of Disk Space to Scale"

49. Liu, Qingyan: "HBase Schema Design Case studies", http://www.slideshare.net/hmisty/20090713-hbase-schema-design-case-studies, 2009


56. HL7 (Health Level 7) specifications: [http://www.hl7.org/](http://www.hl7.org/)


62. RDF Reference – RDF Primer: [http://www.w3.org/TR/rdf-primer/](http://www.w3.org/TR/rdf-primer/)

64. JSON: [http://www.json.org/](http://www.json.org/)
Related topics

- **Evaluate IBM products** in the way that suits you best: Download a product trial, try a product online, use a product in a cloud environment, or spend a few hours in the **SOA Sandbox** learning how to implement Service Oriented Architecture efficiently.
- Download a free trial version of **IBM DB2 for Linux, UNIX, and Windows**.
- Now you can use DB2 for free. Download **DB2 Express-C**, a no-charge version of DB2 Express Edition for the community that offers the same core data features as DB2 Express Edition and provides a solid base to build and deploy applications.

© Copyright IBM Corporation 2011
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)