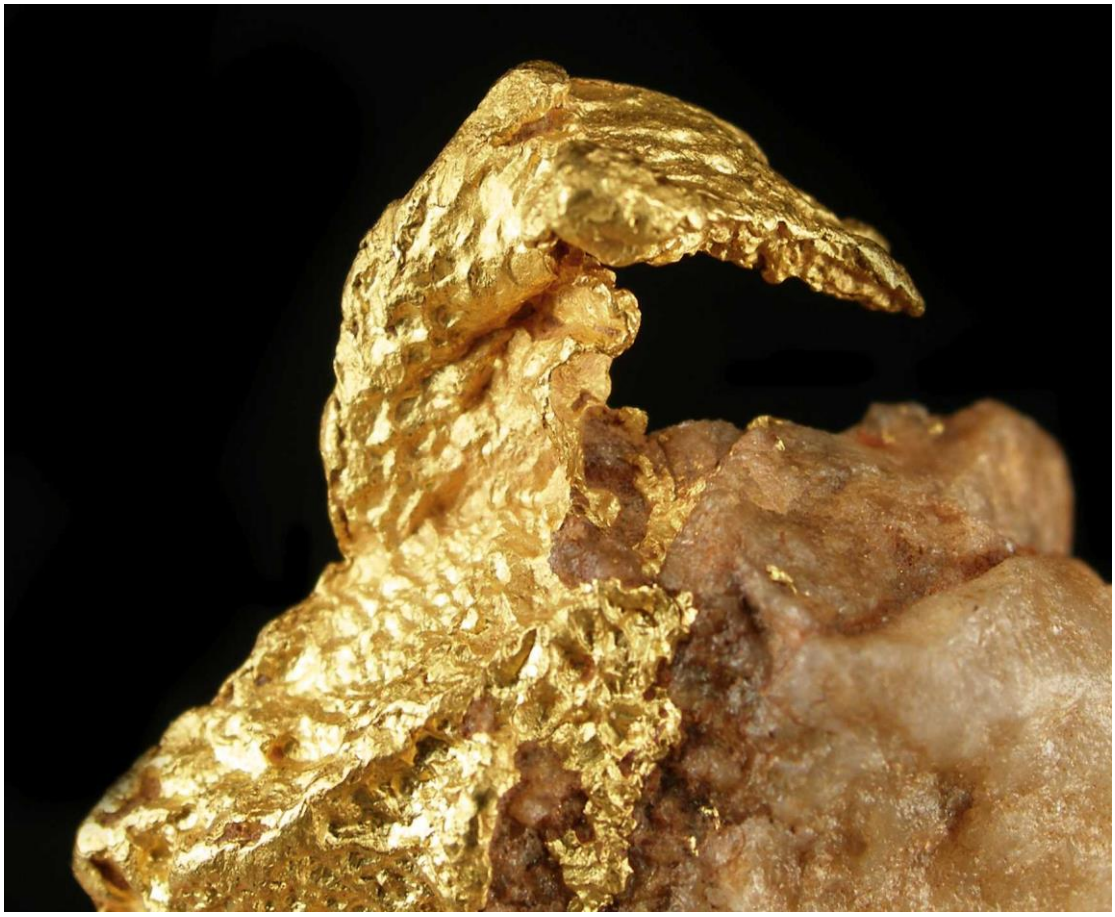
The Big Data Gold Rush



www.plainprocess.com

gary.mohan@plainprocess.com

This ebook can be downloaded for free in PDF format at:

http://www.plainprocess.com/bigdata.html

Table of Contents

Introduction

This ebook is for users needing to understand how Big Data can be used strategically. Deliberately, it has been kept concise, briefing users who need to understand Big Data quickly.

It starts with a “popular science” explanation of relational databases, Big Data and why Big Data really is the most important innovation in database technology since 1970. How Big Data is being used right now in various industries is then noted.

This ebook finishes with how Big Data can be used strategically. For example, data-oriented barriers to entry can be circumvented. Mergers and acquisitions previously deemed infeasible can become feasible. Brands can be repositioned, quality improved and outsourcing reversed.

Relational Databases and Big Data

What follows is a short description of how relational databases emerged in the late 20th century, automating business practices that had become commonplace in the middle of the 19th century. In the 21st century, relational databases are reaching their limits, hence the need for Big Data.

A Brief History of Relational Databases

Relational databases underpin most corporate and public sector IT systems. These range from large-scale driver's license systems to tiny implementations used by less than ten users. The fundamentals of relational databases were devised in two steps, one in 1970, the other in 1974.

In 1970, Edgar Codd published a paper (whilst working for IBM) proposing that the mathematical concept of a "relation" could be applied to managing data within an IT system [1]. In mathematics, "relation" refers to the mapping of one set of data onto another set, where the mapping is a "graph" describing the relationship between the two sets. In 1974, Donald Chamberlin and Raymond Boyce (also IBM) proposed that data related together in this way could be queried using a new simplified programming language called SEQUEL (Structured English Query Language), later abbreviated to SQL [2].

The work of Codd, Chamberlin and Boyce on relational databases was one of the most important innovations of the late 20th century, impacting the lives of billions of people. The late 1990s Internet boom led to an even greater adoption of this already successful approach. To understand why relational databases are so important, it is necessary to look at the 19th century boom in filling in paper-based forms, the problems created by the volume of this paperwork and how Codd's approach addressed complexity around this.

Applications for Gold areas at

No. of Application.	Date of Application and payment.	Applicant's name.	By whom money paid.	Amount paid.	Description of Area.

Figure 1. A form appearing in *An Act Relating To The Gold Fields Of Nova Scotia 1862*.

Figure 1 above shows a template form that appeared in Canadian legislation, requiring the Chief Gold Commissioner of Nova Scotia to record certain information in a "Book of Record" which was to be "uniformly ruled",

concerning gold claims in Nova Scotia [3]. The legislation was needed after chaos created by a confirmed discovery of gold in Nova Scotia in 1861.

Whilst record keeping of business transactions dates back millennia, 19th century economic booms necessitated the creation of new paper forms, with the volume of recorded transactions increasing dramatically. At the same time, the printing industry became more efficient at producing massive numbers of blank forms. Literacy improvements allowed businesses to ask people to fill in forms themselves.

In the 21st century, governments around the world have built temperature controlled storage facilities in an attempt to preserve the billions of pages of "uniformly ruled" records collected in the 19th and 20th centuries. In a sense, these buildings are "databases". Digitization of this material will take decades.

As data volumes grew, it was obvious that 20th century computing could be adapted to store and search through this information. A model was adopted, where hand-written paper forms would be manually keyed into IT systems, with information recorded in an underlying structure similar to the 19th century uniformly ruled Book of Record. Data was aggregated together, allowing management information reports to be created. For a time, this was highly successful.

The problem Codd was trying to solve was that of relating information in one Book of Record to another. Specifically, he introduced something called normalization in order to protect users from "disruptive changes in data representation" [1]. Also, he sought to maintain consistency so as to solve the "serious practical problem" of "more and more different types of data" being "integrated together into common data banks", an issue still with us today.

In Figure 1, it is unclear how the Chief Gold Commissioner is supposed to record the description of the area concerned. The date of "Application and Payment" looks like two pieces of information being recorded in one column. An applicant may have applied and paid on different dates. The way this information was recorded could have varied greatly over time. To help explain normalization, an outline of one attempt at normalizing this data structure is given below:

ID	Number	Application Date	Payment Date	Applicant Name	Payee Name	Amount Paid	Area ID
1	123456	01/01/1863	07/01/1863	John Smith	John Smith	1500	1
2	123457	03/01/1863	03/01/1863	Joe Smith	Joe Smith	1500	2

Figure 2. The Application form as a database table.

ID	Name	Attribute Data	Spatial Data
1	The Old Farm	Attributes (e.g., soil acidity).	Information describing exact geographical points.
2	The New Farm	Ditto.	Ditto.

Figure 3. Area information as a separate database table.

In Figure 2, application and payment dates have been put into separate columns. Each application is given an official application number (the Number column). A “physical” ID has been introduced (the first column, ID). In practice, most enterprise systems use these automatically generated database record IDs (called “primary keys”) to uniquely identify each table record.

Figure 3 shows information about the area “split” out as a separate database table. Records in the area table are linked back to the application table by the use of a “foreign” key. Here, the column Area ID will contain the ID value of an entry in the Area table. This allows information about the area to be recorded once and then linked to multiple applications. An ownership history of an area can then be determined by looking at the applications linked to it. Figure 4 below shows how the relationship between the two tables can be represented in a database design tool.

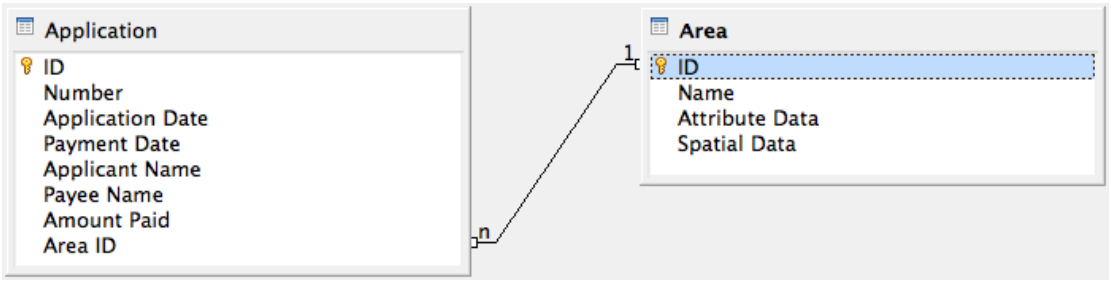


Figure 4. The relationship between the Application and Area tables in a database design tool.

The data structure in Figure 4 is more “normal” than that implied in the paper form. This use of foreign keys (“ID” in the Area table to “Area ID” in the Application table) is the essence of normalization. Information about applications and areas can change independently of one another. It is also more obvious that from the “point of view” of the area, it can have multiple (“n”) applications (a one-to-many relationship). From the “point of view” of the application process, many applications can refer to one area (a many-to-one relationship). Normalization can be extended to record highly complex relationships between data, with great precision. For simplicity, the highly complex data structure that would be needed to accurately record attribute and spatial information for each area has not been shown.

To help query normalized data, Chamberlin and Boyce created the “declarative” programming language SEQUEL (or SQL) to simplify data extraction. SQL is “declarative” in the sense that it declares *what* information should be returned, rather than *how* the database should retrieve it. Three basic keywords are found in most SQL statements:

- *SELECT*. The columns to be returned in the results. The character * can be used to indicate that all columns should be returned.
- *FROM*. The table or tables to be queried.
- *WHERE*. Conditions constraining the query (i.e., the records we really want).

For example, to select the records from the Application table where the applicant is Joe Smith, the query would be:

```
SELECT *

FROM Application

WHERE "Applicant Name" = 'Joe Smith';
```

If we only wanted Joe Smith's application number and amount paid:

```
SELECT Number, "Amount Paid"

FROM Application

WHERE "Applicant Name" = 'Joe Smith';
```

In the above two examples, column names containing a space were put in double quotes. The text specifying Joe Smith is in single quotes. In large enterprise systems, database administrators often adopt a convention of naming columns in upper case (with underscore characters substituting spaces) or in "CamelCase". "Applicant Name" would become APPLICANT_NAME or ApplicantName. Although very common, these conventions are not required by SQL. The current official SQL standard only states that a table cannot have two columns with the same name.

If we want to find information from both the Application and Area tables **based on their relationship**, we can introduce a "join":

```
SELECT "Application"."Number", "Application"."Applicant Name",
"Area"."Name"

FROM "Application", "Area"

WHERE "Application"."Area ID" = "Area"."ID";
```

The FROM clause now names both tables and the SELECT clause prefixes table names before column names. The WHERE clause states that the Area ID column in the Application table is a "foreign key" to the ID table in the Area table, using the equals sign, =, as an "operator".

At first glance, it may not be obvious that this technique for "joining" data from within a normalized table structure really was one of the most important innovations of the late 20th century. In the early 21st century, billions of human beings are in regular contact with systems that use it.

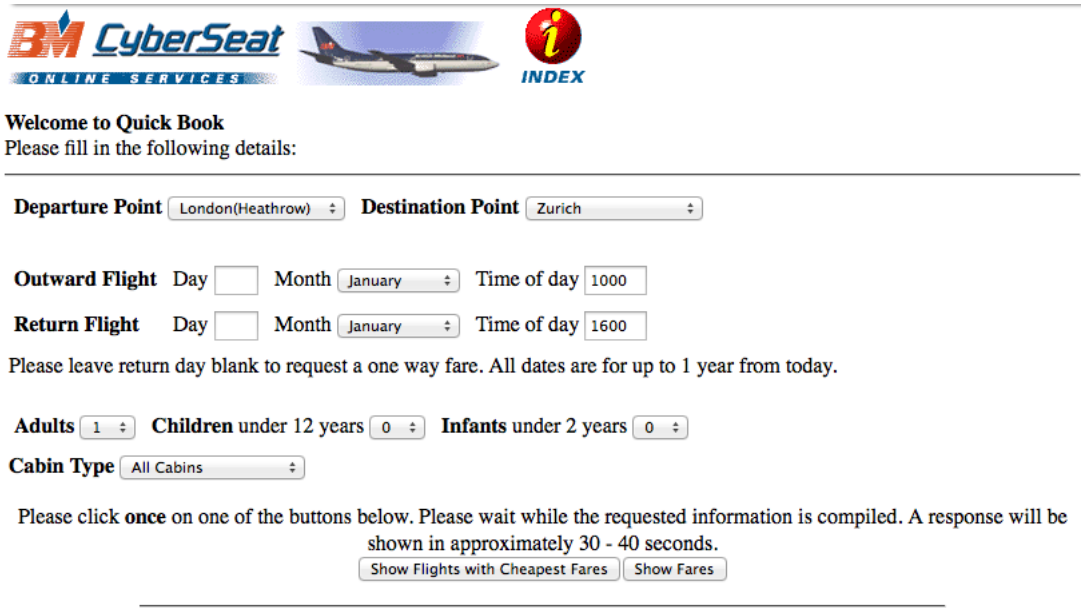


Figure 5. CyberSeat (launched in November 1995).

Figure 5 shows a screen from CyberSeat, a 1995 application that was one of the world's first online air ticket booking systems. 1990s online shopping environments stuck rigidly to capturing the same information that would be used in a paper order form or as in the case of airlines, the information requested by travel agents for use with underlying booking systems. Almost 20 years later, this model still dominates, with businesses being far more conservative and slow at innovating than they think they are. The 19th century form-filling idea has had remarkable longevity.

Again, at first glance it may not be obvious that database normalization and SQL really were two of the most important innovations of the late 20th century. Even less obvious is that if it is so good, why do we now need Big Data? The answer is that since the late 1990s Internet boom we now have enormous volumes of data stored in formats that either look nothing like paper-based forms or could not possibly be captured using a form-based user interface.

Database normalization and SQL are reaching the complexity limits of what they can achieve. Database vendors seem to be reaching the limits of how fast they can run highly complex SQL queries (though SQL systems are likely to still be around for a very long time). Free, open-source SQL systems (like MySQL) are catching up in performance with expensive enterprise SQL offerings.

An example of an unexpected consequence of all of this is that hacking of encrypted passwords is becoming easier. Hackers can pre-compile databases containing all possible results for particular encryption schemes and then run a simple query (or join) to find the real password. The days when hackers needed to have a program running through all possible combinations are disappearing. Despite this, encryption system vendors still sell enterprise cryptography solutions with quotes like "it would take a

standard laptop 100 years to break this scheme". This sales pitch is easier when client procurement managers do not understand how relational databases work.

A fundamental risk that most organizations face when using relational databases is that there is an implicit assumption in using SQL that the underlying data structure has been normalized. This may seem obvious based on the history of how relational database technology developed.

As you are reading this text, somewhere in the world an incompetent software engineer will be insisting on using a denormalized structure, despite Codd writing in 1970 that he knew of "no application which would require any relaxation of these conditions" [1]. This leads to a problem referred to in data warehousing as "dirty" data, a constant source of frustration to users who imagine their enterprise systems to be more sophisticated than they actually are.

Denormalization introduces ambiguities into a data structure that then need algorithmic code to cope, pointlessly increasing software complexity. These systems then require expensive extra testing. It really is the case that it takes just one incompetent engineer to undo a decade of careful work building an enterprise data model for a particular organization. In the period of "cost saving" pressure since the 2008 crash, the risk of this fragility hurting users has become even worse.

Big Data and MapReduce

Big Data uses a programming model called MapReduce. This was innovated by Google in 2003 in order to address the issue of efficiently searching data that did not originate from form filling or from within a relational database [4]. Examples of non-relational data include:

- Implied information within web pages. For example, determining how many public Internet pages link to a particular target page.
- Image data. In principle, images can be stored within a relational database. In practice, attempting to query this data quickly using SQL can be very frustrating.
- Geographical Information Systems data. Often, these are 3D applications, needing to take into account the curvature of the Earth (including that it is not a perfect sphere).
- Automatically generated log information. This can range from attempting to monitor how long individual employees spend on social networking sites to parsing security logs to find intrusions.

Strictly, log information could be stored in a relational database. Often, the time taken to load this data into a relational database is much slower than running a Big Data query directly on the log files. This frustration with load and query times is helping to drive adoption of Big Data [5].

MapReduce is a two-stage process:

- A “mapping” stage parses the data set to find elements containing a specific characteristic. For example, all log entries showing an employee access to FaceBook.
- A “reduction” stage that aggregates together the results of various mapping processes that had been running in parallel.

Superficially, MapReduce looks like standard file parsing taught to Computer Science undergraduates, where a text file is searched line-by-line to find lines with specific text. The difference is in how it uses parallel processing to run the operation simultaneously over various “commodity” servers. These can be put together into clusters, allowing the approach to “scale up”. Google have stated that their clusters run 100,000 MapReduce jobs per day and that their software engineers find this approach easy to use [5].

Attempts at “debunking” MapReduce have appeared in Computer Science journals, to little effect. The people submitting these papers either work for a relational database vendor or a company that supplies some type of value-added service for relational implementations.

Like SQL, MapReduce can suffer from poorly designed queries. This is a symptom of bad software engineering, not a problem with the underlying approach. MapReduce offers these benefits:

- Data that would be difficult or impossible to search using SQL can now be queried.
- The format in which data is stored is no longer a limitation on its usefulness.
- Simultaneous MapReduce jobs can run over the same data, without bottlenecks.
- MapReduce jobs can be chained together, allowing extremely complex queries to be built up.

MapReduce is missing some of the features typical in an enterprise relational database. Importantly, as it does not define a fixed, normalized data structure for stored data, a programmer has to interpret the data structure on a job-by-job basis. Also, it is missing a feature of relational databases called “transaction management”, where the database referees simultaneous attempts to update a particular record. It does not come with an SQL-like query language. Gradually, new frameworks have emerged around MapReduce, addressing some of these missing features.

Hadoop

Created by Yahoo, Hadoop is a popular implementation of MapReduce. It adds to the MapReduce model by providing the Hadoop Distributed File System (HDFS). Based on the UNIX file system, HDFS offers these features:

- A central NameNode stores information about files in a cluster of multiple servers (DataNodes).
- The actual files are stored on DataNode servers, with each file replicated onto multiple nodes (usually three).
- Reliable fault tolerance, even when using hundreds of DataNode servers.
- Performance optimization that assumes data will be written infrequently and read intensively.

By design, MapReduce jobs access HDFS using a client "library". In Linux, it is possible to access a HDFS file system directly using a tool called FUSE. HDFS's design allows cheap servers to be used for implementing a Big Data cluster. In principle, it can be deployed to a Cloud environment. However, some users have found it far more cost-effective to build their own in-house clusters, given the non-competitive pricing of many Cloud services.

Other services provided by Hadoop include:

- *HBase*: A column-oriented table service, providing indexing.
- *HIVE*: Data warehouse infrastructure, which can convert SQL statements in to a series of MapReduce jobs.
- *Pig & "Pig Latin"*: A high-level environment for creating MapReduce jobs, as an alternative to SQL.
- *Chukwa*: Log file analysis.

Pig Latin

Some Hadoop services provide SQL-like query languages. For example, Hive offers HiveQL. Pig offers a language called Pig Latin. These query languages automatically create and run MapReduce jobs, based on the query.

Pig Latin mixes SQL-like declarative statements with "imperative" style assignments (assigning a value to a variable). For example:

```
A = load 'data1' as (x, y);

B = load 'data2' as (x, y, z);

C = join A by x, B by x;

D = foreach C generate B::y;
```

The first and second lines read two files data1 and data2, which are already in a column structure. The first file has only two columns and these are given the names "x" and "y". The second file has three columns, given the names "x", "y" and "z".

The first file is assigned to variable "A" and the second to variable "B", using the = operator. The third line performs an operation on A and B, by joining them together using the first column x, assigning the result to variable C. The

fourth line then assigns to variable D the value of column y from the data2 file (“foreach C” means “read through every record in C”).

Superficially, it may seem pointless to create a language like Pig Latin to perform operations currently supported by SQL. The data files might contain billions of records, necessitating lengthy load times if they were to be stored in a relational database. The files may have been generated by incompatible systems at different times. The query may only need to ever be run once, with the output assigned to variable D needing to go through further, more complex steps. Query languages like Pig Latin do a great deal to simplify operations that would be difficult or near impossible in SQL, across massive non-relational datasets.

Those are the features and benefits of Big Data. So, who is gaining value from it right now?

Current Big Data Users

For some industries, Big Data is not new. An outline of how Big Data is currently being used is given below.

Retail

Retailers are now able to take advantage of historical datasets that were previously difficult to analyze, overcoming the limitations of relational databases. For example, a retailer can profile which times of the day, days of the week and times of the year when particular customers are most likely to make a purchase. This can be correlated with how effective sending promotional emails were, in relation to the time of purchase.

A retailer might find that certain customers respond better to emails sent Monday to Thursday, at around noon, making the purchase at lunchtime. This can help drive sales and lock out competitors. Discounting can be better targeted, avoiding compromises on headline pricing, margins and brand value. Money can be saved on above-the-line advertising.

Retailers can build up analytical patterns. For example, figuring out whether particular purchases take place in a chain. Certain customers may be likely to make a follow-up purchase for product accessories within a defined number of days. This model can be refined to correlate customer actions with key life events, such as going to college or having a first baby.

Social media data can be trawled to find influencers who are likely to promote a particular product. These influencers can be given inducements, such as a free “preview” version of the product, creating a viral buzz. Again, these campaigns can be far cheaper than above-the-line advertising.

Healthcare

The invention of High-Throughput Screening (HTS) has led to an increase in clinical data available about individuals and populations. HTS uses robotics to automate the process of carrying out multiple tests on a clinical sample (or set of samples), generating massive quantities of data. Big Data is addressing the research challenge of relating data about populations to clinical treatment for individuals, using both the population’s and the individual’s data. This also has implications for preventative healthcare.

Big Data is being used in cancer research to address the problem of understanding proteins. Previously, computers tended to be used for researching DNA, given that genomes are coded in a way that stays constant, suiting the limitations of relational databases. In contrast, proteins undergo change. Advances in mass spectroscopy occurred at roughly the same time

as advances in Big Data, allowing massive volumes of data about proteins to be analyzed.

Sensor data gathered from patient monitoring systems could both be aggregated together to gather population data and be subject to real-time analysis. A current research challenge is how to use Big Data to make monitoring smarter, alerting clinical staff to small but statistically significant changes in a patient's state, informing clinical practice.

Telecoms

Telecommunications companies are making money from consumer location and site (or app) usage data (automatically gathered from phones). Consumers can opt-out of this data being collected about them but often do not, due to ignorance, apathy and a lack of understanding about how the data is used. Internet Service Providers can carry out similar data gathering on domestic broadband connections.

Big Data analysis is leading to non-intuitive discoveries about consumer behavior. For example, consumers are most likely to click an advert on a mobile device whilst sat in a cinema, at home on a Sunday morning or outside, fishing. In a similar way to how HBO positioned itself as a "gateway" through which movie companies needed to sell their product, telecoms companies can position themselves in a similar way with respect to consumer product advertising. Advertisers refusing to "partner" with the telecoms company could have their advertising blocked at the network, on a consumer-by-consumer basis, in a way that is very difficult to detect.

Insurance

Perhaps the most obvious application of Big Data to insurance is detection of fraudulent claims. Using historical data of known fraudulent claims specific to a particular insurer, they can leverage their own data to profile incoming claims. Workflows can be automatically generated, alerting a claims administrator.

Actuarial analysis can take advantage of Big Data's parallel processing to run complex calculations on large datasets. In particular, premium adequacy analysis can measure profit / loss outcomes on a risk-by-risk basis, helping to identify issues with the underwriting process.

Government

The criminal justice sector is making increasing use of Big Data, especially with data seized from corporations. It is now becoming normal for investigators and prosecutors to request huge swathes of data, often under a court order. Previously, the limitations of relational databases made this material difficult to analyze. Big Data environments can be used to discover, for example, that a trader made Google searches related to avoiding prosecution for insider trading in the 12 months prior to a series of suspect trades. The scope of analysis can be widened, detecting similar behavior by others and whether a particular manager was supervising them.

The fact that banks know the government has this capacity may help deter future criminal behavior. Given that banks still need to collect most of this data for their own internal security purposes, it will be difficult to avoid compliance through failing to collect data. Detecting suspicious “blank” gaps in data can be automated, aiding investigation.

In March 2012, Barak Obama allocated $200M in R&D funds for Big Data with respect to health, science, energy, geology and defense. The purpose of this funding is to “accelerate the pace of discovery in science and engineering, strengthen our national security, and transform teaching and learning” [6]. It seems likely that as the public sector invests in Big Data initiatives, lessons learned from these programs will be applied in the private sector.

Strategy

As noted above, some sectors are already gaining value from Big Data. Listed below are some of the ways that a Big Data implementation could be used strategically.

Circumventing Barriers To Entry

Business value may be found by using Big Data to circumvent data-oriented barriers to entry. The classic example of this type of circumvention is the development of the budget airline industry.

Up to the late 1990s / early 2000s, most passengers booked air tickets via a travel agent. Often, this was over the phone. Agents were effectively forced to use four "global distribution systems" (GDSs): Amadeus, Worldspan, Sabre and Galileo. Passengers could sense that something was wrong when they thought that ticketing was too expensive for flights that rarely left on time and required flying at awkward times. Incorrectly, they assumed this to be a customer service issue with the travel agent but found that the same hassle happened with multiple agents.

At the travel agency, GDSs had installed proprietary booking terminals. These terminals had skewed algorithms and data opaqueness, designed to maximize revenue. They were "must key in here" systems, in the sense that the GDS was guaranteed to capture data at the time of booking on their proprietary database. Budget airlines spotted that by entirely bypassing agents and GDSs they could provide passengers with a substitute service: letting them act as their own travel agent. At roughly the same time, hotel-booking websites emerged to service people creating their own travel itineraries. Budget carriers took a relaxed attitude to one-way bookings, allowing passengers to create complex itineraries.

All of this was data oriented. The huge capital investment GDSs had made in their IT systems effectively became sunk costs, as budget airlines could build their own system, put up their own website and manage their own data directly. They then were able to carry out analytics, profiling passenger data for the sale of value-added services like car hire or travel insurance. Gradually, they were able to create brand loyalty. Big Data now makes this type of disruption even easier. There is no obvious reason why many other industries cannot also be disrupted in this way.

Brand Positioning

Big Data can help circumvent players with high advertising budgets through analysis of social media. Third parties already exist who provide "Sentiment

Analysis" related to a client's brand on social media. Also, it can be used to identify influencers who may be looking for a way to monetize a Twitter feed or Facebook page. Early entrants can corner influencer engagement, making life difficult for other players.

If taken in-house, Big Data can be used to monitor brand awareness and sentiment in real-time by trawling social media data. The effectiveness of marketing campaigns can be more finely measured, both in quantitative and qualitative terms. Detailed analysis can be carried out on whether a brand is differentiated in the right way. Avoiding the need for expensive focus groups can save money.

Big Data can also "glue" together customer interaction platforms that previously were separate. For example, customer interactions with websites, mobile apps and call centers can be aggregated together, analyzed and then sent back to these platforms, creating a tailored, dynamic customer experience. As opposed to using a single above-the-line message repeated across these platforms, messaging can become contextual. A suite of centralized content can be sent in a specific order at a specific time. This allows promotion to move from an advertising based model to something closer to publishing.

Quality Positioning

Tied to brand positioning is improving your quality proposition, so as to move out of a price competitive environment, attracting higher margins. Many companies cannot do this due to data quality issues, despite having a marketing department who are confident of being able to reposition the brand.

Big Data can help mitigate one of the worst risks of using enterprise relational databases, incompetent software engineers who insist on using a denormalized data structure, despite four decades of software engineering practice to the contrary. These situations often result after a "cost saving" initiative. Attempts to use SQL to query this data will struggle, as implicit in the use of SQL is an assumption that the data structure has been normalized. This is called the problem of "dirty" data.

Big Data offers the possibility of being able to do something with this "dirty" data. An example is trawling to identify customers (including ex-customers) who have been the subject of persistent bad customer service. This information can be correlated with suppliers, business units and individual employees, identifying quality issues hidden from managers behind data opaqueness.

Data related to ex-customers can be trawled to find patterns like whether they ceased the relationship at certain times of the year or some months after asking for a change in the type of service they use. It is quite likely that the marketing department had already picked up on some of these issues through customer interaction but was struggling to "prove" it.

The electricity industry is at the forefront of using Big Data to improve its quality proposition. Electricity companies have problems integrating data from meters, geographical information systems and transformers. Analysis can find frequently overloading transformers. These peaks can be correlated with meter and billing data, identifying customers with problems like poor building insulation. Customer service initiatives can then focus on specific issues like insulation, improving brand perception, managing costs and improving quality.

Innovation

Big Data can facilitate the development of processes and production techniques that were previously infeasible. For example, the US Federal Reserve now has a Big Data environment for historical data, available to third party economists who have been hired to provide advisory reports. This could be scaled down to particular industries, where industry associations provide a similar service to members.

Innovation is happening in industries that previously were not considered data or IT centric. The dairy industry is experimenting with whether cows can be fitted with simple sensors, determining if they are in heat. Detailed temperature data can be gathered for each cow over time and analyzed to find indications of deteriorating health.

Robotic cow milking allows data to be gathered on a cow-by-cow basis, recording milk volume, qualitative information about the milk, animal temperature, animal weight, milking time and the amount of feed consumed. From this data productivity information can be determined for individual cows, alongside finding early indications of when veterinary intervention may be needed. This approach may be applicable in other industries where large amounts of sensory data are already gathered or could be gathered.

Reversing Outsourcing Relationships

The 2000s decade saw companies outsource anything and everything that was not seen as “core” to the business. Post-crash, this has started to look ill advised as these companies lost access to data sets related to their own transactions, which could have been used for risk analysis. Today, these companies are locked into supplier relationships courtesy of the fact that the supplier holds all the data. The management teams who did not understand this issue are long gone.

One of the most intimidating aspects of attempting to break these supplier relationships is that even if the supplier is willing (or could be forced) to supply historical transaction data, the client has no way of querying or using this data. Big Data can facilitate this, given that the format in which data is stored is no longer an impediment to it being used. This is not to say that Big Data

provides a definite solution to supplier lock-in, however, the supplier's bargaining position is now weaker given that their historical capital investment in IT systems is less relevant.

Mergers and acquisitions previously deemed too difficult

Many mergers and acquisitions are seen as infeasible due to both sides having investments in highly proprietary IT systems. Big Data mitigates some of the worst aspects of addressing this problem.

When a merger takes place a press release will say that the two companies intend to "merge" their systems. In reality, it is usual to pick one of the systems, use that for everyone and import data from the other system. The other system is kept running for a transition period of usually about two years, mostly for customer service and audit purposes.

The intimidating part is that the data models used in both systems are likely to be incompatible, with each company having had different ideas about how to classify and manage transactions. Big Data mitigates some of this as a Big Data environment can automate some transformation steps on very large amounts of data that previously required a new system to be written just for this task. It is not a panacea for all issues, however, in many instances it solves enough problems to make the data usable.

Summary

Big Data facilitates business strategy development in these ways:

- Barriers to entry around capital investments in large-scale databases can be circumvented, as Big Data systems provide similar performance at a lower cost.
- Customers can be offered substitutes, circumventing “must key in here” systems.
- Middlemen hoarding data in proprietary data hubs can be bypassed.
- “Sentiment Analysis” of social media can help bypass players with high advertising budgets, finding out if an offering has been differentiated in the right way.
- Social media influencers can be identified, creating marketing partnerships.
- Customer interaction platforms can be “glued” together, allowing a move from advertising to something closer to publishing for brand messaging.
- Historical datasets that were previously too difficult to analyze can now be examined for insights.
- Quality and customer perceptions of quality can be improved, helping to reposition a brand.
- Processes and production techniques that were previously infeasible can be innovated.
- Big Data helps to reverse one aspect of outsourcing, in that it is no longer necessary to have access to the supplier’s system for a client to query data exported from that system.
- Mergers previously considered infeasible can become more feasible through the use of Big Data.

If you need help developing a Big Data strategic plan for your organization or need architectural help with a Big Data implementation, I can be contacted at:

gary.mohan@plainprocess.com

References

[1] Codd, E.F. *A Relational Model of Data for Large Shared Data Banks.* Communications of the ACM 13, 6 (June 1970, 377-387). ACM Press, New York.

[2] Chamberlin, D.D. and Boyce, *R.F. SEQUEL: A Structured English Query Language.* In *Proceedings of the 1974 ACM SIGFIDET* (Now Sigmod) *Workshop on Data Description, Access and Control* (Ann Arbor, MI, May 1–3 1974, 249–264). ACM Press, New York.

[3] Nova Scotian Department (19th century body) *An Act Relating To The Gold Fields Of Nova Scotia 1862*. [See: http://archive.org/details/cihm_39184 Starting on page 44 of the PDF.]

[4] Dean, J. and Ghemawat, S. *MapReduce: Simplified Data Processing on Large Clusters.* In *OSDI'04: Sixth Symposium on Operating System Design and Implementation.* [See: http://research.google.com/archive/mapreduce.html]

[5] Dean, J. and Ghemawat, S. *MapReduce: A Flexible Data Processing Tool.* Communications of the ACM 53, 1 (January 2010, 72-77). ACM Press, New York.

[6] Office of Science and Technology Policy *Obama Administration Unveils "Big Data" Initiative: Announces $200 Million In New R&D Investments.* [See: http://www.whitehouse.gov/sites/default/files/microsites/ostp/big_data_press_release_final_2.pdf]