# Corporate Data Obesity: 50 Percent Redundant

Hae Kyung Rhee

GJCST Computing Classification
H.2.m, K.6.4

***Abstract*-In this essay, we report what we have observed with regard to status quo of corporate information systems in real world from our experiences of twenty years of data management practices. It is considered to be serious in that data are too conveniently and frequently replicated to make information systems improperly behave in terms of their quality standards including response time. Average ratio of data replication in a site is astonishingly judged to be more than 50 percent of a whole corporate database. It is in reality about 65 percent in average to our knowledge. Presenting this paper to academia has been motivated by our strong belief and evidence that most of the redundancy can effectively and systemically be removed from the very start of information system development. We also noted that field workers including database administrators in corporate environment tend to think data part of IS and program part of IS mixed together from the start of IS design and popularity of this tendency eventually caused a lot of entanglement that could hardly be dealt with later by themselves. We therefore present a couple of mandates that must be respected in order not to get involved in such a perplexity**

*Keywords*-Corporate Data Obesity, Data Redundancy, Enterprise Data Map.

## I. CONCEPT OF OBESITY

It is not unusual to think that if a person is weighed more than about 20 percent of what needs to maintain for fitness then he or she is considered to be over-weighted. This is what we understand with regard to concept of obesity. It is no different for data in corporate environment. It will be astounding to recognize that the degree of data obesity in corporate is far more than 20 percent. It is in fact 65 percent in average for some dozens of large enterprises we have observed in depth for the past twenty years. To be exact in terms of terminology, the unit of obesity we mean is data attribute. For example, if there is a customer data and it is comprised of c-name and c-address, c-name and c-address are the data attributes. So, in case c-name appears more than once in a corporate database, it is called redundant or replicated. Although the reports on data abundance in corporate environment have been made in the literature, as far as we know, only the issue of data deluge [Cukier2010, KaBoZe2010] has been dealt with a couple of times in order to emphasize world-wide phenomenon of rapidity in increase of data in terms of volume. The issue of data obesity is new in the world-wide communities of database research and management information systems research. In this sense, it is almost impossible to find any past work in the literature made with regard to this issue. Note that the concept of data obesity is essentially irrelevant to data volume. Although introduction of some upper-level data stores like data warehouses (DW) or data marts (DM) other than the lower-level operational data stores (ODS) in corporate environment certainly contributes to abundance of data, DWs and DMs are out of scope in this essay. If we stick only to ODSs, we could observe that a lot of obesity is already there in corporate environment.

Note that, in a fairly large corporate such as General Electric or Samsung Electronics, there are approximately 15,000-to-20,000 data attributes in their database. Notice also that the level of redundancy in data attribute is not exactly the same as the level of redundancy in data volume. However, to make it comparatively simple to have some idea about redundancy in terms of data volume, since a lot of people in field work prefer this way of understanding, when we happen to hear that database size of some company is, for instance, 100 terabytes, it is legitimate or reasonable to think that the company in reality has a database of approximately 35-to-50 TBs. So, in case 50-to-65 TBs of data can be totally eliminated from the corporate database and this elimination does never affect harm the normal operation of the database at all. Redundancy demands a huge cost in terms of waste in storage and belatedness in response to database queries. Note that even 1 TB of data amounts to piling A4 size papers up about 100 kilometers high.

Redundancy or replication gives some illusion that it could contribute to enhancement of response time, but on the other hand things can get messy if we consider consistency of data. The quality of answers to data queries could be always in question, since making all the replica copies to have the same value usually takes a substantial amount of time due to non-automatic processes of such data value propagation. Manual propagation by considerate programming nevertheless unfortunately incurs unforced human errors and there is no guarantee for data consistency at all across a corporate database. Once an inconsistent value of data happens to be used to reply the queries, trust of information system would unbelievably collapse. Issue of mistrust would then raise the question of integrity with regard to a whole information system.

Therefore, limiting the occasions of data replication to be minimal is necessary whenever it is possible. Unless the rate of data redundancy is substantially reduced, say to about 15 percent by means of wary design from the outset of IS development, data normalization theories [YuJa2008] that have been esteemed almost over the past thirty years turn out to be “useless” at all in real world. To our knowledge

*About- Associate professor at Dept. of Computer Game & Information in Yong-In Songdam college.*
*(telephone:82-31-330-9234 e-mail;leehk@ysc.ac.kr)*

reduction comes quite before some tabular form of data begins to emerge in the process of IS development and that is just where we start to lay out job descriptions, in non-technical term. We will get back to this later in this essay after discussion with regard to how people in IT field are insensitive to the issue of redundancy.

## II. Unnecessary redundancy

an arena where data is represented in a form of table or relation, in expertise terminology, the concept of keys like primary key and foreign key is technically inevitable. Basically, if a particular key of table, say A, dubbed its primary key, is duplicated in another table, say B, as a part or component of key of B, that key is denoted as a foreign key in B, as it has been imported or borrowed from other table, which is A. This clarifies that origin of the key is from A, not B. This way of designating and incorporating such externality of key will bring IS about 15 percent of data redundancy contained intrinsically, which is technically unavoidable if we stick to the tabular representation of data. This portion of redundancy can be called redundancy of necessity. So, if data obesity ratio is said to be 65 percent, it is true that about 45 percent of the entire data is therefore classified to be unnecessary or superfluous in their nature.

Whether to remove this much of unnecessary redundancy or unwanted replication is up to decision of an individual data manager, but unless removal of them is done the information system would definitely be hampered or suffered by lack of consistency and further by eventual slowness in response time. Note that, normally in the database queries of any corporate, about half of them are update requests and the other half are retrieval requests. If this reality of read-write ratio, i.e. 0.5, is ignored, we are soon tempted to allow data duplication by assuming that reads are much more frequent than writes, and subsequently a fatal disaster would then be experienced sooner or later due mainly to data inconsistency dilemma.

The payoff for burden of upholding this unnecessary redundancy is really enormous. Usually, it would be about five times more costly than the case where the level of redundancy is minimally enforced. So, it is going to be 10 million dollars versus 50 million dollars when so called next generation, i.e. enhanced version, of information system is to be developed. As the degree of data redundancy increases, data consistency tasks among operational databases exponentially as well increase in proportion to the amount of increase in data redundancy. Note that there is inevitably redundancy between the lowest-level database and its upper-level data warehouses, since data in database are in principle shoveled upward to its data warehouses in the process of generating data warehouses. It is also a natural consequence that another layer of redundancy is unavoidable between data warehouses and their upper-level data marts.

In case data redundancy is existent, it is not difficult to find many of duplication are intrinsically semantic. Syntactic duplication is easy to find out, but it is almost impossible to determine whether any data is a semantic derivative of some other data. This semantic data duplicity is the major malice to make corporate database incurably obese. So, it is necessary to remove syntactic duplication, but it is exceedingly more crucial not to forge any possibility of semantic duplicity from the very outset of IS development. It really is almost impossible to check semantic equivalence, even periodically, once an information system is in operation day to day.

## III. De-normalization—panacea or deadly homepathy?

It is really unfortunate that we have never seen any data table or relation that even follows the rule of well-known first normal form (1NF) in real world corporate databases. So, sometimes it is ridiculed that real world databases only contain tables of non-normal form or zero normal form, since they have properties significantly inferior than 1NF in terms of data quality such as the degree of data redundancy and dependability of non-key data attributes to key attributes. The beauty of table normalization or table standardization by applying 1NF, 2NF, 3NF or Boyce-Codd NF is that whenever there is a data redundancy in a table then it is possible to remove it by decomposing or splitting the table into two.

In corporate IT field unfortunately a term "de-normalization" [JoJA2007] has gained so much popularity in a sense that field managers usually do not have a time to pay attention to and understand the theories behind normalization. They at first pretend to understand and use them, but in reality they sooner or later totally forget about them. By far, we are very unfortunate that we have never seen any database administrator who really does understand the basic difference between 1NF and 2NF. The reality is that they keep never trying or studying to grasp the meaning and benefit of making tables normalized and keep feigning to have started with 1NF initially for IS development and to proceed forward to make tables in up to 3NF and all of sudden for the sake of performance they inevitably and eventually come to resort to 1NF again. But this could be a sort of fictional story and hence never true at all, since they always had failed to tell us what the intrinsic difference between 1NF and 3NF is. A number of experiments [KSLM2008] already have shown that having tables in 3NF performs always better than 2NF or 1NF and that 3NF is considered to be quite optimal even in cases where seven-way table joins are conducted. Note that 7-way join means that combining seven different tables, each fairly large in our experiments, at the same time.

The real problem with IT field managers and even database administrators is that they hardly understand even what the 1NF is. Note that in any data-related literature for the past forty years of history, notion of "de-normalization" has never been introduced, but they pretty much fond of taking that jargon just in order to forget about normalization stuff and to wish to let themselves totally unaware of any impending issues related to data consistency. They seem to be soon relieved to hear by someone else that normalization

could always be compromised for the reason of performance. To our knowledge, they are misled by mainly outside IT consultants who have never been trained enough in basic knowledge in database. So, it is actually a very demanding burden to make them understand what the normalization theories are all about.

However, this is not too bad if we know that having tables even in 3NF could contribute to reduce the degree of data redundancy by at most about 5 percent, which is not too much. Consequently, the contribution of normalization would be only minor. But then, where is the majority of contribution come from? It comes much prior to the formulation of tables. In order to realize this, we have to know what and where the origin of data essentially is in corporate environment. Where is the place where redundancy really starts to build? It is at the very beginning of business processes, not where the normalization theories are just about to be applied. Wouldn't it be curious that where are all the data that are to be appeared eventually in tables come from?

## IV. Necessity of Business Processes Description

Let us turn our attention to how business processes are described so that field workers can communicate each other later on. They will certainly be in a form of business processes description or job description. So, the transformation of job descriptions into data tables might take a couple of interim stages, since descriptions themselves have a format different from table and there is no direct, straightforward method that can map the descriptions into tables. Then, how is job description comprised of? In it, there could appear data entity like employee or department which has fixed values for data attributes it is comprised of.

For example, a data entity 'employee' might consist of data attributes 'address' and 'social security number' and their values are normally fixed, i.e., not changed over time. In case in job description there is a description statement like "An employee sells a machine.", data entities 'employee' and 'machine' will have such fixed values, while on the other hand data entity 'sell' is different in that the values that data attributes of 'sell' like selling date or selling volume vary, i.e., changed each time the action or behavior 'sell' is performed. So, action entities are at the focal point in terms of creating different data values in the database. It can be considered that the source entity of action 'sell' is 'employee' and its destination entity is 'machine'. This way of writing job descriptions by taking action-oriented approach or behavior-oriented approach [KDLM2007] is straightforward. It could be fairly easy to understand for employees who have a mission of writing a description for jobs they actually perform.

Efforts to make job descriptions to be free from data redundancy are essential and valuable to check whether there is redundancy of any sort for each particular action. This means the action 'sell' above appears at most only once in job descriptions of whole business processes of a corporate. It is judged to be improper or abnormal if the action 'sell' appears more than once in entire job descriptions of the corporate. This kind of effort in reducing or removing actions redundancy has no relationship in what is known to be crucial like 1NF, 2NF or 3NF, as emphasized in the literature. But removal effort with regard to redundancy in data attributes directly associated with actions is far more important than the removal of redundancy in tables at a later stage of database creation. If the removal effort is not sufficiently done, redundancy thus retained intentionally or unintentionally would then automatically be transferred intact to tables at the instance of table creation.

From the perspective of who or what is in charge of dynamically creating data in corporate environment, it is fair to admit that behaviors, rather than fixed entities, play the major role of such creation. Fixed entities that are always expressed as nouns in description statements like 'employee' and 'department' normally generate only static data attributes and thus said to be only at the outskirt in data-creating activities. In this sense, it is meaningful if we preferably write job descriptions in a way of behavior-by-behavior. Each behavior then has a responsibility for creating only meaningful data attributes. In case a behavior does not contribute to generate certain attributes, it has no value of existence to be independent or stand alone. This means that in that case it is reasonable to place that behavior to be subsumed by some other behavior that is directly relevant and superior to it.

## V. Behavior-Oriented Job Descriptions

As we have observed over the past 20 years, the unit of resources that is assigned to an employee is normally a job. Definition of jobs has been in a sense pretty much well established in corporate. For example, we could count the number of jobs in a corporate without much difficulty. To our experience, a mid-size corporate has about 500 to 1,000 jobs and to perform those jobs it normally requires to maintain the number of employees of about twice as much as the number of jobs, since it is a usual practice to assign two persons to a single job in order to prepare for emergencies of just-in-case. So far, we have seen a number of corporate that have about 500 jobs and 1,000 employees in real world. This might be a kind of standard for mi-size corporate.

We were able to observe from our experience that each job in average could be comprised of some 20-to-30 actions or behaviors in case data-creating actions are only taken into account in job descriptions. So, if there are 500 different jobs in a corporate, then it means that there are about 10,000-to-15,000 behaviors altogether in that company. With no redundancy in actions, those some 10,000 behaviors must be unique in that they do not incur redundancy of any types so that each of them must appear once and at most once throughout the entire corporate database.

## VI. Enterprise Data Map

These behaviors are in a sense interconnected each other in a way that each data-creating action has one fixed entity on its left and one more fixed entity on its right. If we denote a interconnection would look like a type of 'E—B—E'. So,

behavior by B and a fixed entity by E, then the web of those the whole picture would look something like a rectangular type that would allow data accesses or data retrievals in either direction, clockwise or counter-clockwise, as depicted in arrows in Fig.1.

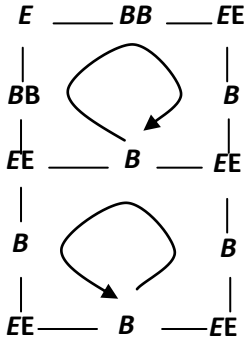


Fig. 1. Rectangular Path Formed in Enterprise Data Map, where B Denotes Behavior and E Denotes Entity

Rectangularity guarantees balance in response time in either direction of access, while if otherwise skewed case to one particular direction could induce degradation in response time. Although there are only seven actions in this picture, we could get a whole diagram that contains some 10,000 behaviors if we keep extending the picture by adding more behaviors to it. The entire picture of connection without allowing isolation of any picture fragment could be called an enterprise data map [Moon2004].

With this EDM, we are able to judge or realize where the origin of a particular data attribute is and how it flows throughout the entire data access paths already obtained and depicted in EDM. With EDM, it is very easy to find out visually where are data redundancies if there are any. As a diagram, one EDM can depict about 20 pages of A3-size in case font size of 5 is used. Drawing would be automatic if we use a software drawing tool such as ERwin [JoJB2007]. The EDM of such many pages would then easily fit into the wall of CEO's or CIO's office. Or it could also be displayed on CFO's office in case he is interested in figuring out how is the flow of all the data directly related to financial status quo of his company. Unfortunately, at the moment only a few corporate experienced the value of obtaining and maintaining the EDM, but we advocate that its use would significantly benefit many aspects of information system. We advocate that utilization of EDM would thereafter be plentiful according to your perspectives of looking at it.

## VII. Separation of Data from Program

It is needless to say that EDM is the must to be secured and kept as an asset prior to the programming of information system. We emphasize that any programming effort must be deferred until the finalization of EDM. EDM in this sense is the blueprint for any design like, for instance, building or road. To our knowledge, EDM is definitely the blueprint for information system prior to any programming effort. What we emphasize is that data itself is essentially data in that programming must begin to take place only after the data formulation has been made to sure to be completely wrapped up. Data-first programming-later approach is crucial for the success of information system. If data stuff and programming stuff are mixed together from the start of information system development, chaotic situations would duly be encountered in determining that whether an impending problem at issue is originally from data part or programming part. We emphasize that any data cannot be represented or expressed or substituted in a way of any programming means.

Note that if somebody happened to introduce a data 'whether-a-student-is-registered-or-not', then it is in fact a disguise as a data in that it essentially has a sort of algorithmic logic in that data. Presuming that a data like 'registration date' could reside somewhere else in the database already, 'whether-or-not' type of decision could then be definitely dealt with some conditional statements like 'if' in programming. Separation of data from programming must be strictly obeyed in a sense that, without separation, a bunch of semantic redundancy like this sort of disguise could later be insidiously come into the information system. If it seems that this way of algorithmic logic is certainly in a data, then it is not real data, since only the raw data is privileged to be called as data. Anything impure in a way of generating artifacts is not called the real data. For example, if data C is from the result of addition of raw data A and raw data B, then C is not in principle treated as data. Note that in the lowest infrastructural level database of corporate only such raw data are entitled to reside. Anything else must be deported to reside somewhere else like data warehouses.

## VIII. Conclusion

In sum, there are two major mandates that have to obey to make information systems free from data obesity. The first one is that efforts for removing data redundancy should be enforced from the start of information system development, which is from the starting point of securing job descriptions. The latter one is the strict separation of data arena and programming arena in developing information systems. Questions like whether this belongs to data or programs are better to be raised as frequently as possible in order not to bring any chance of confusion about which comes before and which comes after or later. To our knowledge, the degree of data obesity is guaranteed to be tolerated within at most 20 percent if these two mandates are strictly obeyed.

Removal of another 5 percent of data redundancy is later possible if we conduct a certain set of technical details. The well-known data table normalization or data table decomposition theories come into play for this further removal. So, the benefit accrued from the data redundancy removal efforts by application of normalization theories is considered to be far less than we get from the efforts made at the stage of job description, which is about 30-to-45 percent of removal in data redundancy in an entire corporate database. It is adding one more flower to a beauty itself already seized if the normalization theories are applied to

make tables best fit with minimal redundancy in them, but we certainly might have no regret at all when they happen to be not applied for some reason under the premise that data redundancy of all sort has already been sorted out and managed to be ruled out prior to table formulation.

The adage "Trying to start with guarantees almost half-way done already" still prevails in the world of information system development and making IS fit or well-being in any situation or environment comes true when we immersed to think in this manner. Consequently, the earlier we preoccupied with the trial of data redundancy removal, the better the outcome of information systems in terms of performance, clarity, transparency and promptness in response time.

## IX. References


1) [Cukier2010] Cukier, K. (2010, Feb. 25). Data, Data Everywhere. A Special Report on Managing Information, The Economist. Retrieved May 1, 2010,from http://www.economist.com/specialreports/displaystory.cfm?story_id=15557443.html

2) [KaBoZe2010] D. Katz, M. Bommarito, & J. Zelner (2010, March 1). The Data Deluge. The Economist print edition.
3) [JoJB2007] J. Jones & E. Johnson (2007). Building and Maintaining A Database from An ER Model. White Papers : Computer Associates.
4) [Moon2004] S. Moon (2004). Data Architecture, Hyung-Seol Publishing Company.
5) [KDLM2007] N. Kim, D. Lee and S. Moon (2007). Behavior-Inductive Data Modeling for Enterprise Information Systems. Journal of Computer Information Systems, Vol. 48, No. 1, 105-116.
6) [KSLM2008] N. Kim, S. Lee & S. Moon (2008). Formalized Entity Extraction Methodology for Changeable Business Requirements. Journal of Information Science and Engineering, Vol. 24, No. 3, 649-671.
7) [YuJa2008] C. Yu & H. V. Jagadish (2008). XML Schema Refinement through Redundancy Detection and Normalization. VLDB Journal, Vol. 17, 203-223.