

BE A MODEL CITIZEN: HOW TO DESIGN A SUCCESSFUL DATA MODEL

George Dobbs, Dynamics Research Corporation

ABSTRACT

It remains a paradox that the very reason why DBAs exist, namely "The Relational Data Model", invented by E.F. Codd in the 1970s, receives very little attention at DBA gatherings. It's time to remember our roots: we must take a moment to remove one hat, the Physical Database Administrator, and put on another, the Logical Data Modeler, to rejuvenate our understanding of our very purpose. This paper will remind us of the importance of the Logical Data Model, how it fits into this thing called "Enterprise Architecture", and offer three secrets to designing a successful model.

INTRODUCTION

THE PARADOX

The people who build Databases are a special breed: they enthusiastically perform a job most people would shun. The very nature of their job is paradoxical: they must at once *make easily available* the very thing they are charged to *supremely protect*, nothing less than the most valuable asset of any Information-Age Company, its data.

What matters most to people in a functioning Organization is their Business Information: they want it all, they want it perfect and they want it now. They want it protected, pristine, backed-up, jacked-up, quantified and verified. In other words they want it *logically* accurate, and *physically* accessible.

Paradoxically, all of this responsibility seems to ultimately fall on the shoulders of one individual, the **DBA**, the "Physical Database Administrator."

A modern Database, much like a baby, requires a lot of physical attention, dare I say love, to keep it going. That means *someone* has to be its *mother*. That "someone", of course, is none other than the DBA.

Ironically, when all goes well, the DBA is invisible and ignored. When, however, the smallest problem arises, the DBA is vilified and hunted. There are myriad physical reasons why a Database would develop problems, most of which result in highly visible and undesirable consequences.

It is no wonder, then, that DBAs concentrate almost exclusively on preventing the highly visible failures. This means they tend to perform the overt mundane duties that prevent the most immediate dangers from disrupting their Database, namely the **Physical Tasks**. These are the well known, traditional DBA duties that we all know and love, for example, creating and maintaining tablespaces, tables, indexes, triggers, redo log files, partitions, initial parameters, block sizes, users, roles, privileges, packages, procedures, exports, backups, and so on.

But these duties represent only half the story. A Database needs more than just *physical* attention, yet paradoxically, no one ever seems to mention the *logical* attention it needs.

If the DBA is the Database's Mother, providing physical *protection* and *nurturing*, then who is the Father, providing *logic* and *purpose*? Enter the **Data Modeler**.

LOGICAL VS PHYSICAL, YIN AND YANG, NATURE VS. NURTURE

The roles of **DBA** and **Data Modeler** are quite different. It takes both roles, Yin and Yang, to rear a Database to healthy adulthood. Then why, do we always seem to hear about the DBA and almost never about the Data Modeler? Far too often the role of Data Modeler is either vastly diminished or unceremoniously handed over to the DBA as a secondary burden.

Sometimes, heaven forbid, it is abolished altogether. Nothing could be more deleterious to the long-term health of the Database. This phenomenon has analogies to the perennial “Nature vs. Nurture” debate in science class: how much of a Database’s success is due to its genetic design, its “logical model”, as opposed to its developmental manifestation, its “physical administration?” The answer is, of course, both parents of Database development are needed, the Logical and the Physical, and the unfortunate omission of the Logical Data Modeler from the nursery has had dire consequences indeed.

SUMMARY

A modern Database needs two types of minds, two distinct skill-sets, two parents, as it were, to conceive it and nurture it to maturity: the Logical parent and the Physical parent.

The first parent, the **Logical Data Modeler**, designs the very potential of the Database, and decides what it will be concerned with, how it defines its own purpose and how it will relate philosophically to the ideal world.

The second parent, the **DBA**, manifests the Database into physical reality, and carefully builds its environment, protects it from malcontents and teaches it how to adapt to the real world.

The Data Modeler’s tasks are inherently **Logical**.

The DBA’s tasks are inherently **Physical**.

BEGINNINGS

E.F. CODD

I often wonder how the above paradox came about. Why do DBAs and not Data Modelers dominate the Database World? And why are Database Professionals mostly concerned with physical performance as opposed to logical truthfulness? And just to make sure we’re all on the same page, just what do we mean by “Physical” vs. “Logical?” in the first place?

Fortunately we can shed light on these questions by understanding why we are all here to begin with. The very reason the modern Database Profession exists today is because of a man named E.F. Codd. His breakthrough paper, “The Relational Model of Data for Large Shares Data Banks” gave birth to the information age itself. Dr. Codd’s brilliant insight was born out of a *physical* need to share large amounts of data, yet his solution was inherently a *logical* masterpiece of the highest order.

In other words, he solved a *physical* problem by applying a *logical* solution.

Thus was spawned the roots of our current paradox: Logical vs. Physical, and here’s how. Prior to Dr. Codd’s conceptual breakthrough in the 1970’s, electronic data was stored differently depending on its *hardware*. It was stored in ways that best suited the particular “big iron” hardware upon which it resided. Each implementation was rather unique, and to access the data on any particular machine required extensive customization. Consequently, any small change in a “business rule” would require an incommensurately large change in the customized coding and data storage structures. Thus, although it was possible to “brute-force” the old systems to behave, one had to accept the relatively high cost of physically reconfiguring the hardware, data structures, and thousands of lines of procedural code.

BIRTH OF THE PARADOX

In other words, the Data Processing professionals of the 1970s spent most of their time manipulating the physical parts of their systems; a costly and slow way to do business, not to mention self-defeating. Unfortunately, this mindset remains today for many modern DBAs who support the mantra “thou shalt force a database it into submission with more indexes, more code, more hardware.”

Though it took several decades to take hold, all this emphasis on physical solutions changed with E.F. Codd’s revolutionary invention, which he named a “Relational Model for Data”. For the first time in the history of “Data Processing”, it was possible to transcend the limitations of hardware and to organize data in such a way to make it accessible without regard to its physical storage; again, truly the heart and soul of the Information Age.

As we will see later, even Dr. Codd did not fully realize the power of his “Relational Model”. His original goals were modest, indeed. He wanted only to propose a new way to store physical data, to make it more efficient for later retrieval. He did this by proposing a novel concept of how to logically organize data according to set-theory. What he did not realize is that he hit upon an extremely powerful idea with far wider implications. Not only was his Relation Model good for organizing electronic data but it was also a perfect method for organizing ALL information, even information about information (e.g. meta-data).

THE DATA MODEL

WHAT IS A DATA MODEL?

In the field of Systems Engineering the discipline of *Data Modeling* is arguably the *least* understood. To complicate matters, Analysts (including Zachman) speak of three *types* of Data Models, namely “Conceptual”, “Logical” and “Physical”. What is the difference between these *types* and how would the master of the Zachman Framework deploy these pieces to win the game? Before we answer this, we must first briefly explore the *source* of the Data Model’s power.

RELATIONAL THEORY

In the early 1970’s, Dr. E.F. Codd, working for IBM, introduced a brilliant solution for organizing information based on mathematical set-theory. His seminal paper, “A Relational Model of Data for Large Shared Data Banks” (Codd, 1970) remains unequalled today in its impact on information technology. It is heralded for not just spawning the multi-billion dollar Relational Database Management System (RDBMS) industry, which includes Oracle, Sybase, SQLserver, DB2, etc, but also for providing the theoretical foundation for numerous operating systems, the client-server revolution, object oriented databases, data warehousing, not to mention countless user applications, etc (Codd, 1990).

Relational Theory is based on three key ideas (Codd, 1970):

1. **Elemental Forms.** Organize your world by recognizing the things that *don’t change much*. Identify the smallest units of the Enterprise and describe them accurately and uniquely. Model them as they exist in the world; thus as the world changes, so *commensurately* does the model change.
Tech-speak: Identify *entities* and *relationships*, and assign finite attributes to them according to set-theory (nTuples).
2. **Extensibility.** Organize your world into *categories* of things, to which you can add any number of *specific* things.
Tech-speak: Structure your entities so that they can be iterated to infinity.
3. **Independence.** Organize your information so that it can be understood on its own merit and in such a way that it is valid and useful irrespective of *how* it might be used.
Tech-speak: Store data independently of how it will be accessed.

UNEXPECTED POWER

A Data Model, or more specifically, a *Relational Data Model*, gains its power from a simple idea: by constructing *surrogates* that *mimic the real world*. These “Relational” constructs, then, become a perfect, logical way to define and organize the very *real* parts of your Enterprise! The Relational Data Model, as originally conceived was intended to be merely a practical methodology to store business data independently of how it may be retrieved in the future. However, the amazing consequence was that its underlying principles offer an ideal way to describe *all* types of information; even information *about* information (meta-data). Most importantly, Relational principles work equally well at any level of abstraction: Conceptual, Logical or Physical. It is ironic that *despite so much success* and *intrinsic theoretical power*, the general knowledge about Relational Theory is woefully insufficient among Engineers, especially among Enterprise Architects.

RELATIONAL “BUSINESS” MODEL

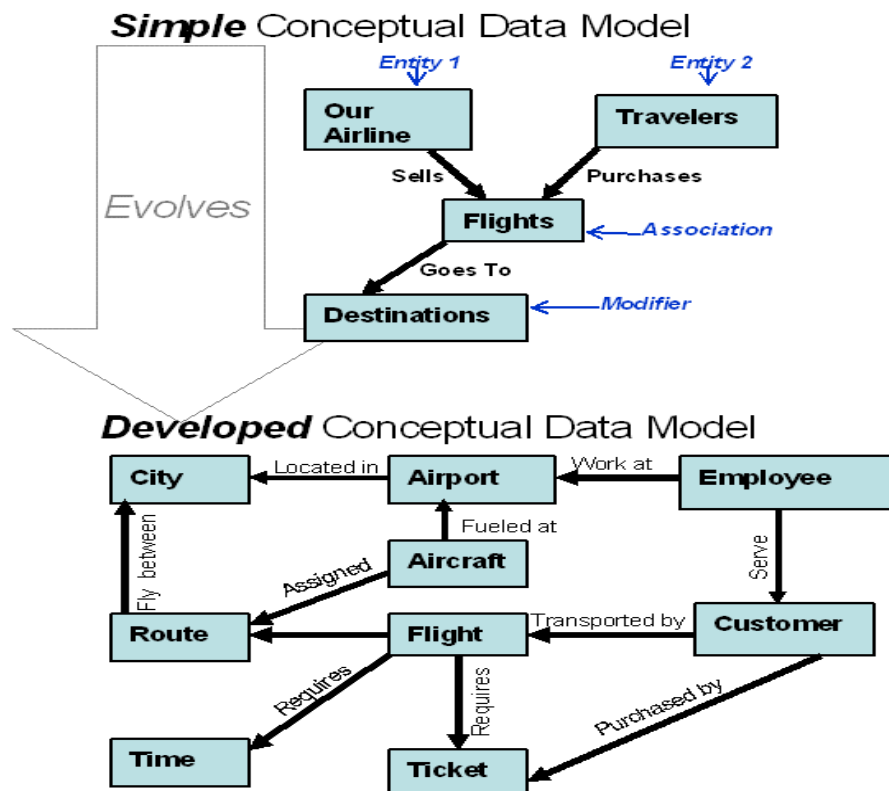
The key to understanding why the Relational Data Model is so powerful is first to realize it has the *wrong name*. Instead it should be called the Relational *Business* Model. Here’s why. The Data Model by its nature contains an *analogous structure* for every *entity*, *rule*, and *function* of the Enterprise. In fact, the very act of creating a Data Model elicits from stakeholders the desired *behavior* and *information* needed to define the Enterprise. And since Relational principles work equally well at any level of abstraction, be it Conceptual, Logical, or Physical, the model will easily accommodate the evolving Architecture as it grows from amorphous mission concepts, through incipient business objectives, and finally to mature physical manifestations.

THREE STAGES OF DEVELOPMENT

The Zachman Framework presents us with three types of Data Models, *Conceptual*, *Logical* and *Physical*. However, it is best to think of them as *one* Enterprise Data Model that evolves through 3 stages of development. The difference between the stages boils down to: 1) how much *detail* it has and 2) to *whom* you are speaking. In other words, when you begin defining your Enterprise, and speak to the Big-Picture People in Row 1, you will normally find their ideas painted in broad strokes. This is absolutely natural and necessary. The good news is, it is completely legitimate to use the principles of Relational Theory to capture these high level concepts. Furthermore, as you progress through the Framework and develop more precise artifacts, the business concepts become clearer and more fully attributed. As the concepts are fleshed-out, they are systematically incorporated into the Data Model until ultimately the model has outgrown the realm of *Conceptual* and matured into the realm of *Logical*. This same process continues until the Logical Data Model becomes complete enough to be manifested into reality, and thus become the blueprint for the *Physical* Data Model. Without going into detail about how the Physical Data Model is opportunistically manipulated to accommodate physical constraints, the important point is that it must always remain in sync with the Logical Data Model. So, at the end of the day, there is only *one* Enterprise Data Model that has been developed through three natural stages, Conceptual, Logical and Physical.

DATA MODEL IS A MICROCOSM

Just as the game of chess is a world unto itself, yet unabashedly mimics the “real-world” of Asia’s feudal era, the Data Model mimics the real-world of your Enterprise. To understand how a Data Model is really a precise, “shorthand” microcosm of the Enterprise itself, let’s take a look at the following example of the Airline Industry.



THE DATA MODEL MIMICS THE WORLD

CONCEPTUALIZE THE ENTERPRISE

If we take another look at the Zachman Framework, we see that even at the *most abstracted level* of defining the Enterprise, i.e. concepts found in **Row 1**, the *Conceptual Data Model* is ready and able to incorporate them all.

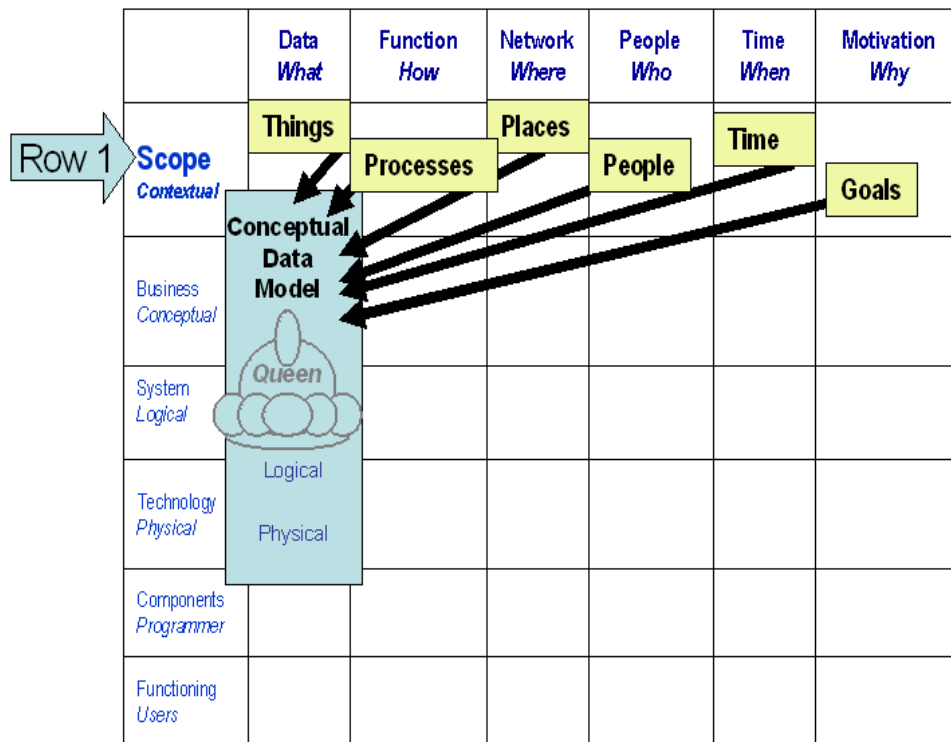


Figure 3: Conceptualize the Enterprise

Figure 3 shows that the high level (Scope) concepts from Row 1 are organized according to Relational principles and become the de-facto source of information to enable the creation of the Conceptual Data Model.

TAKING CHARGE

We have seen that the Top Row of the Framework contains concepts of the highest abstraction and that they are *incorporated into* the Conceptual Data Model. With this new power, the Data Model immediately takes charge of the board by directing the development of all other artifacts on Row 2, starting with **Business Objectives** (see Figures 4 & 5).

The Data Model facilitates creation of the other artifacts on its own row by providing:

1. **Common Terminology:** Provides the concepts that govern the *Business Objectives* artifact (conceptual/why in Row 2/Col 6)
2. **Foundation:** Provides the initial “straw man” objects that are primed to be transformed as further knowledge about the business is gained, and as the other artifacts are created.
3. **Bookkeeping:** Provides a means to keep track of all heretofore known business concepts.
4. **Communication.** Provides a concise and graphical way to communicate accumulated business knowledge to all stakeholders.

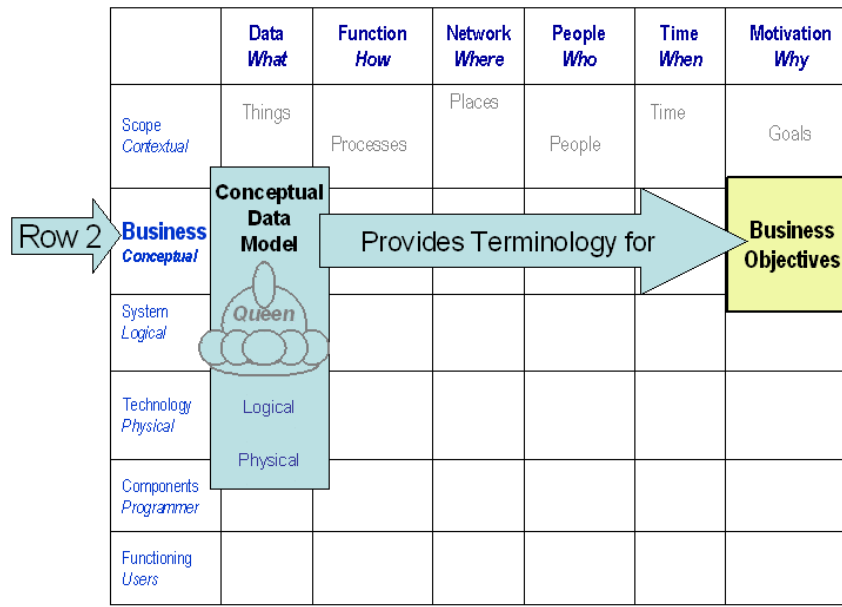


Figure 4: Data Model facilitates formulation of Business Objectives

Figure 4 illustrates that the Conceptual Data Model is a *repository* for the foundational concepts that facilitate the development of Business Objectives.

BUSINESS OBJECTIVES FORM STRUCTURES IN THE DATA MODEL

As we see in the above illustration, once the Data Model is “loaded” with concepts from the Top Row, it then acts as a repository of *terminology* and *concepts* from which to build the next-most-important artifact: **Business Objectives**. This artifact is crucial because it supplies the *motivation*, the very reason *why*, one needs to create the remaining Row 2 artifacts.

To understand how the Conceptual Data Model facilitates the formulation of Business Objectives, let’s imagine that an airline vice president, Mr. Bigwig, declares boldly that, in order to satisfy the primary mission of “*being the finest airline in the world,*” one of the objectives should be “*to have a worldwide presence.*” While this seems like a good idea, it is too vague to act upon. The Architect realizes he must rephrase Mr. Bigwig’s informal assertion using more precise language with terms taken directly from the Data Model. In other words he needs to transform Mr. Bigwig’s good-but-incomplete idea into an *actionable Business Objective*.

To accomplish this, the Architect would analyze Mr. Bigwig’s declaration in order to surmise its implicit concepts: *Countries, Cities, Airports, Terminals, Leases, Routes, Aircraft, Flights, Employees, Customers*, etc. These concepts are immediately incorporated into the Conceptual Data Model. Next, the Architect simply *uses the terms from the Data Model* to re-state Mr. Bigwig’s declaration as proper objectives: 1) “The company must LEASE AIRPORT TERMINALS in major CITIES throughout the world,” and 2) “The company must own sufficient AIRCRAFT, establish adequate ROUTES, schedule enough FLIGHTS and hire enough EMPLOYEES to become known to the desired CUSTOMER base within three years of operation.” In turn, the mature Business Objectives govern the production of all other Row 2 artifacts (see Figure 5). This process will be explored again later when we cover *Business Rules*, which are just *Business Objectives* that are more precisely articulated.

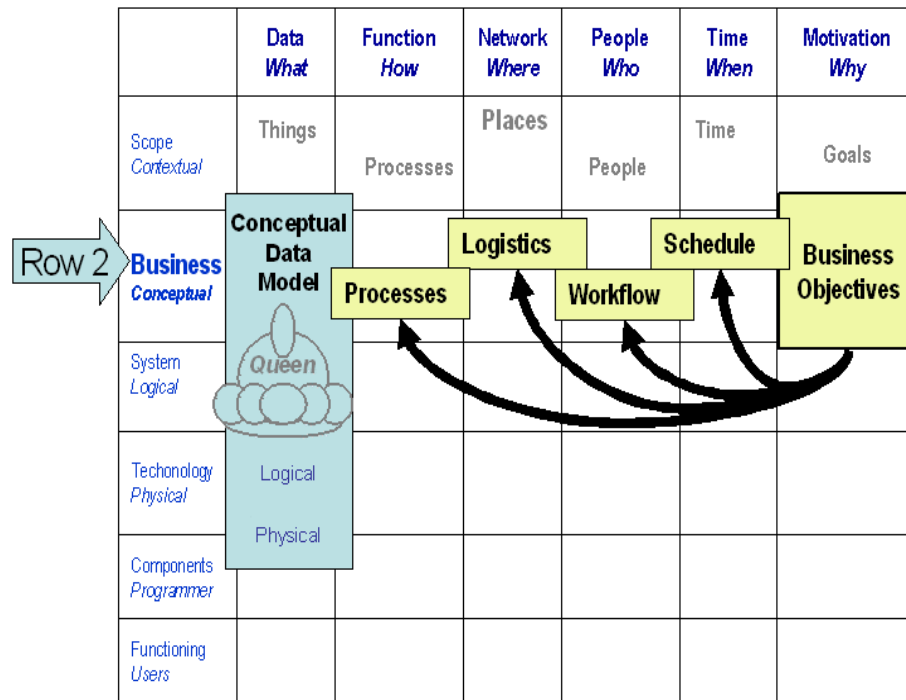


Figure 5: Business Objectives guide Row 2.

Figure 5 illustrates how **Business Objectives** govern the development of all other design artifacts in Row 2.

THE SKILL OF THE DATA MODELER

The completion of Row 2 (Business/Conceptual) artifacts represents a huge step forward in the development of the Enterprise Architecture. The knowledge, discipline and insight gained here are essential to the success of the Enterprise. All of this new knowledge is organized according to Relational principles, and fed into the Data Model, effectively honing *existing* business concepts and adding *newly discovered* ones (Harmon, 2003). *This is the most exciting and risky stage of the game*, and requires a great deal of skill to play. It is at this point in which the entire width and breadth of the Enterprise is systematically and precisely encoded into a Logical structure, one that abhors contradiction and incompleteness. For this reason, the Data Model provides yet another enormous benefit to the Enterprise Architect: it greatly facilitates the discovery of *inconsistencies*. When inconsistencies are found in the Data Model, feedback to the stakeholders ensues, resulting many times in the discovery of crucial yet overlooked information affecting the Enterprise, in turn forcing re-articulation of Objectives or Rules.

THE DATA MODEL CONSOLIDATES POWER

Through the iterative process above, we see that eventually the Data Model will start to stabilize, indicating that all the important business concepts have likely been accommodated. As we mentioned above, the *Conceptual* Data Model is distinguished from the *Logical* only by how *complete* it is (see Figure 7). So as the model evolves, and more business concepts are nailed-down (i.e. named, and assigned specific attributes), we begin to see the birth of the Logical Data Model.

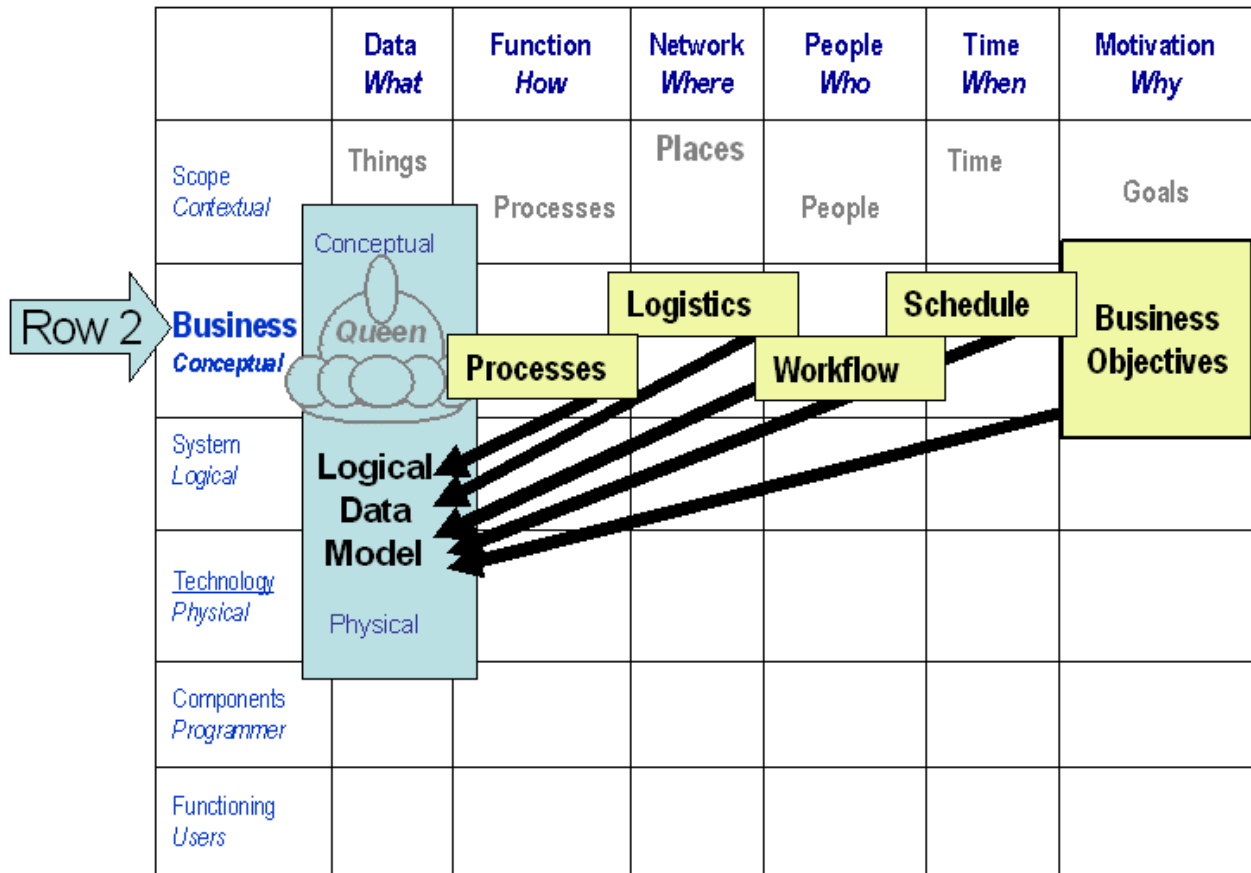


Figure 6: The Data Model Grows Up

Figure 6 illustrates how the completed Business artifacts from Row 2 supply detail and refined concepts to the Data Model. Eventually, the fully attributed Data Model evolves from Conceptual to Logical.

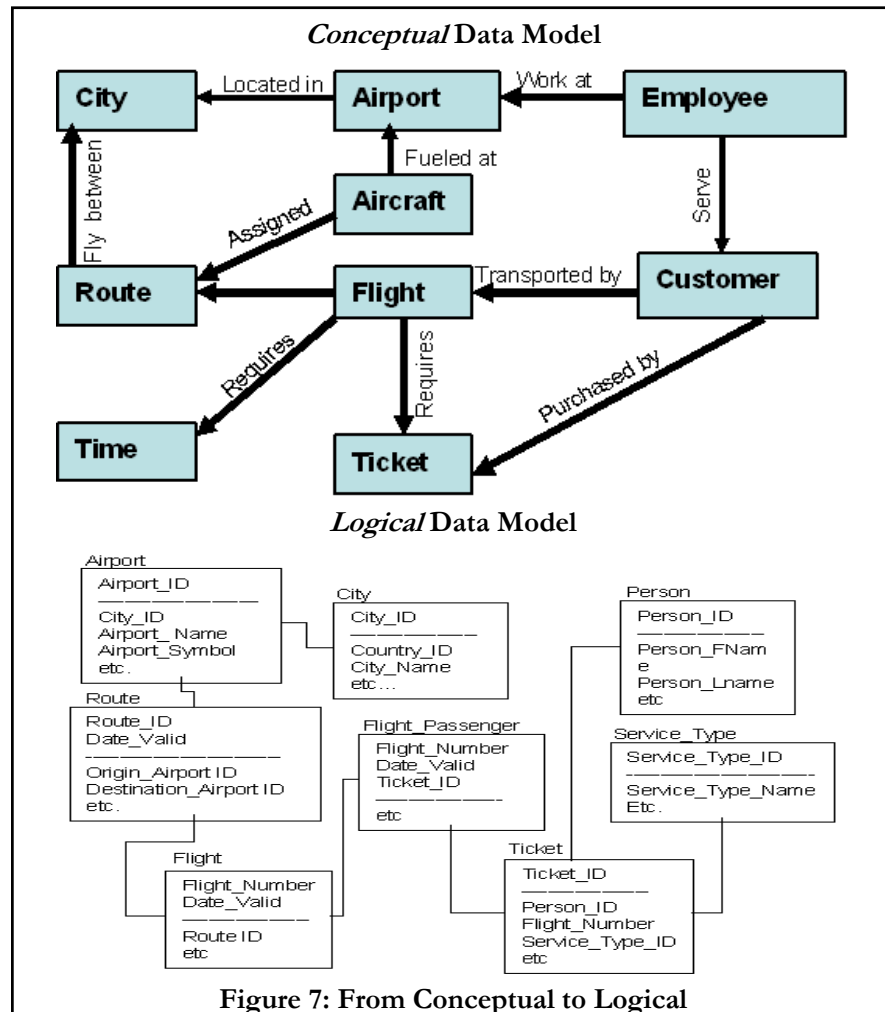


Figure 7 illustrates how the Logical Data Model is simply a more detailed version of the Conceptual Data Model. It specifies the *attributes* of the entities and their relationships. Also note that some entities can change names (*Customer* becomes *Person*) or even split (*Flight* becomes *Flight* & *Flight_Passenger*), and more precise relationships can emerge (*Service_Type* is now included in the definition of *Ticket*).

FORMALIZE BUSINESS RULES

The next important artifact that must be created is **Business Rules**. *Business Rules* are similar to *Business Objectives* (described previously) except they consist of formalized English that declare unequivocal edicts about the Enterprise (Ross, 2004). Just as the *Conceptual Data Model* guides the formulation of *Objectives*, so does the *Logical Data Model* guide the construction of *Business Rules*. The following example shows the *direct relationship between the Data Model and Business Rules*.

For example, let's say Mr. Bigwig makes the assertion that "*first class passengers should never be bumped due to overbooking.*" While this declaration is earnest, it is too vague to act upon. So, it must be reformatted into more precise terms. Again, the Data Model comes to the rescue. The Architect would search the Data Model for all entities that could have an affect on *class of service* versus *seat capacity*. He would then formulate the Business Rule using the precise terms of the Data Model (Ross, 2004). Consequently the above informal assertion would be transformed into an actionable **Business Rule** like this: "The Business must be able to determine SEAT COUNT availability per SERVICE TYPE prior to the assignment of TICKET to PASSENGER."

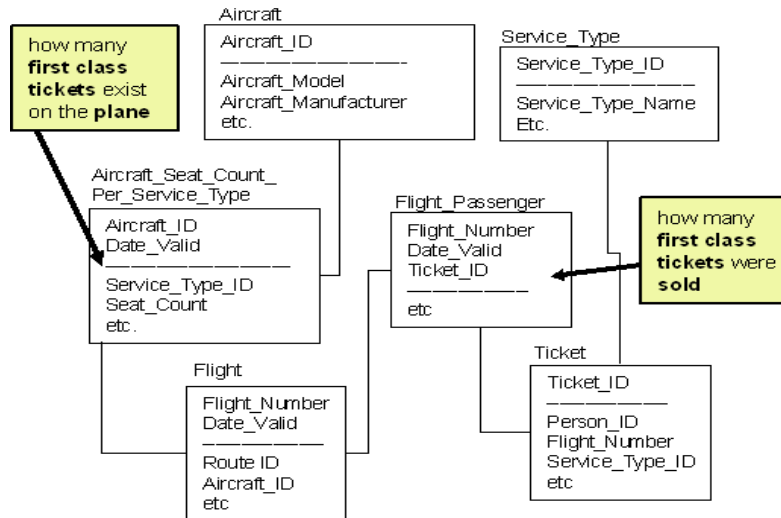


Figure 8: Formalized Business Rules from the Logical Data Model

Figure 8 illustrates how the stated Business Rule is perfectly correlated to the constructs of the Logical Data Model.

	Data <i>What</i>	Function <i>How</i>	Network <i>Where</i>	People <i>Who</i>	Time <i>When</i>	Motivation <i>Why</i>
Scope <i>Contextual</i>	Things	Processes	Places	People	Time	Goals
Business <i>Conceptual</i>	Conceptual	Processes	Logistics	Workflow	Schedule	Objectives
ROW 3 System <i>Logical</i>	Logical Data Model	Provides Terms and Constructs for				Business Rules
Technology <i>Physical</i>	Physical					
Components <i>Programmer</i>						
Functioning <i>Users</i>						

Figure 9: Foundation for Business Rules

Figure 9 illustrates that the Logical Data Model serves as a repository of all *entities* and *relationships* that are required to create formal Business Rules.

THE DATA MODEL ADAPTS AS THE GAME CHANGES

We have seen that the Data Model acts as a *repository* for concepts from which to formulate precise Business Rules. Just as important is the *reverse*, in which new Business Rules *cannot be satisfied* by the current Data Model. In this case, the Data Model itself needs to be altered to accommodate the newly discovered requirements. This happens all the time and is necessary and healthy for the Enterprise. Once the **Business Rules** are mature, the cycle is repeated as the Architect uses them to guide the development of the remaining Row 3 artifacts (see Figure 10).

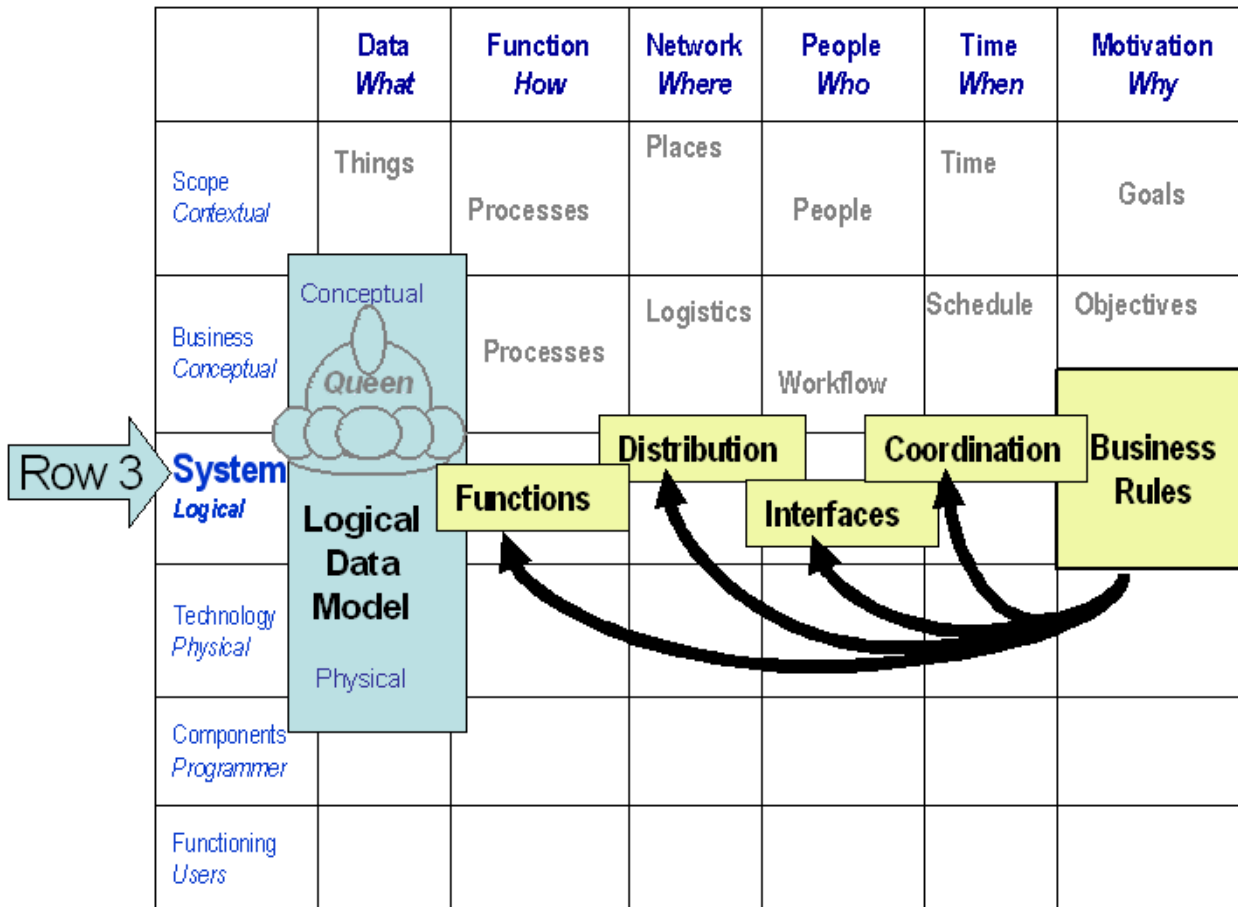


Figure 10: Business Rules guide Row 3 artifacts

THE PHYSICAL DATA MODEL RULES THE TECHNOLOGY

HOW TO BE A MODEL CITIZEN

By the time Business Rules have guided the creation of the remaining Row 3 artifacts, a great deal of business knowledge has been discovered, documented and agreed upon by stakeholders. Throughout this period, the Logical Data Model has acted as the *terminology repository*, the *relationship tool* and the *communication device* for accumulated Enterprise knowledge. As if this were not enough, the Data Modeler's work is still not complete: his next job is to assert his power into the real world. This is where the Physical Data Model comes into play. As before, the maturity cycle continues and all Row 3 artifacts are harvested intensely for very specific business requirements. *Here is where the game is won or lost.* Row 3 artifacts are so detailed, so clearly articulated, they leave *no doubt* about what the system is *expected to do*. All of these great expectations, e.g. queries, reports, response times, GUIs, etc, must now be placed in the hands of the Physical Data Model: if *it* fails, then the system fails.

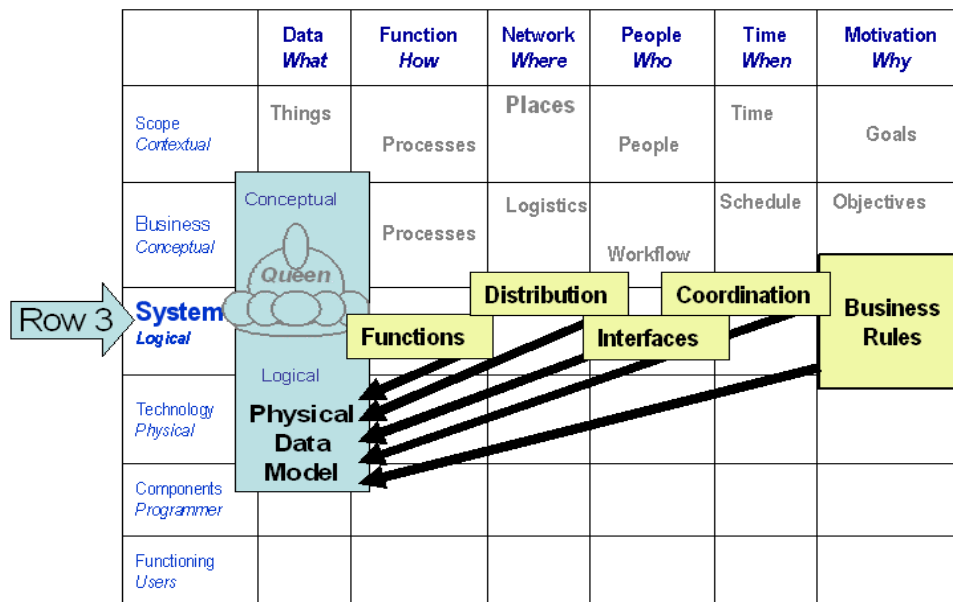


Figure 11: The Physical Data Model must accommodate great expectations.

GREAT RISK

So far we have seen the power and flexibility of the Data Model in its *Conceptual* and *Logical* stages. Remember that there is only *one* Enterprise Data Model and that it evolves through three stages, all of which must be kept in sync. The final stage of the Data Model is the Physical Data Model. This is the last step before any physical database objects are created (e.g. Oracle tablespaces, tables, triggers, indexes, etc). The Physical Data Model evolves directly from the Logical and in many cases incurs little change. However, a Physical Data Model may intentionally be made to deviate from its Logical ancestor in order to ameliorate perceived performance shortcomings (this is a huge subject and won't be fully covered here). *However, the crucial point is that all such constraints designed into the Physical Data Model will be manifested directly into the physical databases, and thus cascade throughout the entire system.* For this reason, the entire Information System's performance is affected by the Physical Data Model. Once again, the Data Model *rules!*

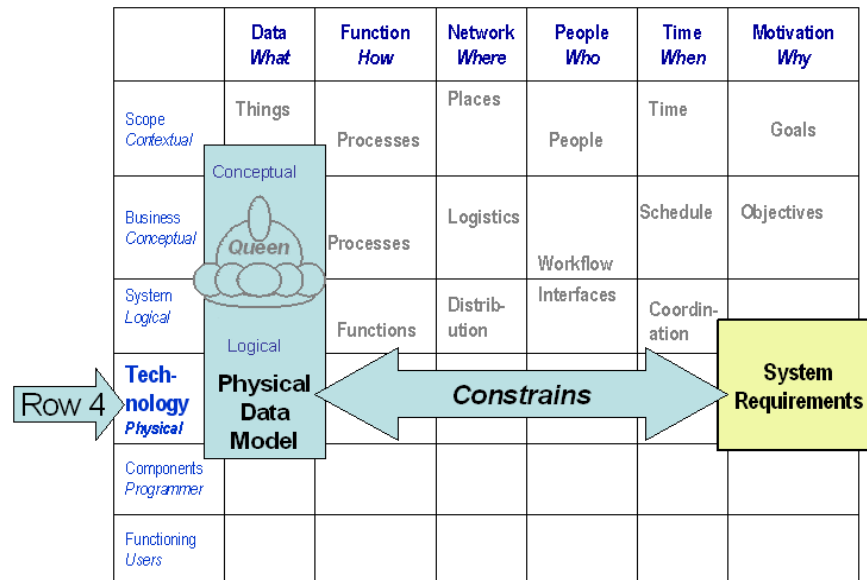


Figure 12: The Physical Data Model Constrains the System.

Figure 12 shows that the **Physical Data Model** both *constrains* and is *constrained by* **System Requirements**.

THREE SECRETS TO A SUCCESSFUL DATA MODEL

To help you get started in turning your Physical Databases into good Corporate Citizens, there are three secrets to making sure your Data Models are sent out into the world safe and healthy.

1. Know what the word "Model" really means
2. Know how to interpret Stakeholder VIEWPOINTS
3. Know how to synchronize the Data Model to the Business Model

1) KNOW WHAT THE WORD "MODEL" REALLY MEANS

A Model is a Small Thing that is analogous to a Big Thing. It exists so that you can understand the Big Thing better in order to control it. If you get your model right, it mimics the Big Thing to the point where you, the modeler, can not only gain deep understanding of it, but eventually, almost magically, make non-intuitive yet accurate predictions about how the Big Thing will behave far into the future.

To *model* the world, you must *know* the world. Study. Know how the world works. You must model the world as it really is, not as it has been described to you by the secretary or the CEO, who usually define things imprecisely because they are not trained to be abstract 'systems' thinkers, as are you. That is your job. You must include in your model all entities and concepts that impact your world, whether or not those things are immediately obvious to the non-analytical thinker.

As the world about you changes so must your model **commensurately** change. If you find you need to change your model a *lot*, when the world changes just a *little*, then your model is wrong, not the world. You must continually hone your model (yes, the word continually changes, so too must your logical model, and also therefore your physical database structures: get used to it) so that when little changes in the business environment occur, your model has already anticipated it and thus requires little or no change itself.

2. KNOW HOW TO INTERPRET STAKEHOLDER VIEWPOINTS

What is a **Stakeholder**? That's the person you have to please and for whom the Database must ultimately render real, day-to-day economical value (see the discussion on the Zachman Framework above to learn more about Stakeholders).

What is a Stakeholder **Viewpoint**? That's how he looks at the world, through his own peculiar spyglass, rightly or wrongly, accurately or not. That is how the Stakeholder defines and values his purpose and value to the Organization.

Why do we care what the Stakeholders' Viewpoint are? See the first paragraph of this section.

Why do we need to *analyze* or *second guess* the Stakeholders' Viewpoints? It is our job to reap the pearls of their wisdom and to learn enough about the business to politely ignore the rest. This is not as easy as it sounds. Remember, all Stakeholders have Viewpoints, but most do not possess the type of analytical thinking skills that they pay you for. They tend to lump together disparate concepts irrespective of their taxonomy or pertinence. However, we must still not allow an erroneous concept to become part of our Logical Data Model, even though the source of that concept may have come from a person who out-ranks you in the Organization. To be sure, the Data Model is the last thing on their mind, because most Stakeholders simply have no way of knowing how much of their success depends on your ability to model their world. For this reason, they are often reticent and unsympathetic to your task. But be brave, because if you can model *their* world accurately, you will be making *your* job infinitely easier in the long run. If you get the Logical Data Model right, you will be buying for yourself a ticket to freedom. Your future will be yours to control as you will be eliminating the dreaded future of constant table alterations and myriad lines of procedural code to accommodate some "newly discovered" business rule.

Once again, the Zachman Framework is the best tool to use to nail down Stakeholder Viewpoints. Study it.

3. KNOW HOW TO SYNCHRONIZE THE DATA MODEL TO THE BUSINESS MODEL

The way you synchronize the Data Model to the Business Model is via the Process of Formal Business Rules, as described in the body of this paper (section: "Formalize Business Rules", page 9).

The Formal Business Rule Document is just one cell of the Zachman Framework, yet it is one of the most difficult "design artifacts" to create. I promise you it is well worth the trouble. Fortunately its development works hand-in-hand with the development of the Data Model, so much so that the two endeavors should not be done separately.

If you make sure that your Logical Data Model can accommodate all Formal Business Rules, and conversely, that all Business Rules have corresponding constructs in the Data Model, you can't go wrong.

All formal nouns and verbs (entities and processes) gleaned from your Business Rules should have identical structures in the Logical Data Model. Furthermore, if you concentrate on your Logical Primary Keys, defined in common English, you cannot get your one-to-many relationships (cardinality) wrong.

Your Logical Primary Key is your best friend.

Do not forsake your Logical Primary Key; you will always regret it.

A Logical Primary Key is the smallest combination of Columns (attributes) that will uniquely identify a Row.

Just because you impose a physical sequence column on a Table does not mean you have correctly identified its natural or Logical key.

CITIZEN IN GOOD STANDING

We have seen that the Data Model has been an upstanding citizen in the world of Enterprise Architecture. From its birth through all developmental stages, gambits and diversions, the Data Model has been in the thick of the fight for a civilized Information Age. My fellow DBAs, you are now challenged to turn all your Physical Databases into good corporate citizens by getting your Logical Data Models right.

REFERENCES

Codd, E.F. "A Relational Model of Data for Large Shared Data Banks", Commun. ACM 13(6) 377-387: (1970), <http://www.informatik.uni-trier.de/~ley/db/journals/cacm/Codd70.html>

Codd, E.F., *The Relational Model for Database Management: Version 2*, Addison-Wesley, (1990)

Harmon, Paul, "Developing an Enterprise Architecture", Business Process trends, (2003)

Ross, R.G., "Applying the Business Rule Approach", Business Rule Solutions, LLC., Edition 3.1. (2004), <http://www.BRCommunity.com>

Zachman, J.A., "A Framework for Information Systems Architecture", IBM Systems Journal, vol. 26, no.3, pp.276-292. (1987), <http://www.zifa.com>

BIOGRAPHY

George Dobbs has been a Technology Consultant for over twenty years, for both commercial and federal government clients, specializing in Relational Data Modelling, Business Rules and Enterprise Architecture. He has been lecturing on the virtues of Relational Theory and the Zachman Framework for over a decade, including as guest professor for post-graduate seminars.