Alternative Technologies

Logic for Serious Database Folks Series

by David McGoveran, Alternative Technologies

LEGAL NOTICE AND LICENSE

IMPORTANT CONTENT NOTICE

It is anticipated that many readers of this document will have some familiarity with the writings of other authors, especially but not limited to E. F. Codd ("EFC"), C. J. Date ("CJD"), and H. Darwen ("HD"), or familiarity with Date and Darwen's The Third Manifesto ("TTM") and related writings. Material in this series will not comply with all of TTM and definitions therein, nor with any other interpretation of the work of EFC. Readers should not assume otherwise of an independent work. I will, from time to time, refer to specific passages in the works of EFC, CJD and HD for purposes of commentary or to illustrate some issue. When I disagree with a some construct by EFC, CJD, or HD and am aware of it, I will say so and give a justification.

CONTACT BY REVIEWERS

Reviewers are encouraged to convey their comments and criticisms to me directly via email addressed to **mcgoveran@AlternativeTech.com**., and including "REVIEW" in the subject line. Please provide document title, revision date, and page when referencing any passage. I will do my best to respond in a timely manner and will try to answer specific questions if there is a reasonably short answer.

www.AlternativeTech.com mcgoveran@AlternativeTech.com

Post Office Box 4699, Deerfield Beach, FL 33442 Telephone: 831/338-4621

Alternative Technologies

Series Introduction
Logic for Serious Database Folks Series

by David McGoveran, Alternative Technologies

"Against logic there is no armor like ignorance." – Laurence J. Peter

I. MOTIVATIONS

For well over twenty-five years I have been asked to explain various aspects of logic and its implications for issues pertaining to relational databases. The 1993-1994 four part series Nothing from Nothing (especially Part One, entitled "Classical Logic: Nothing Compares to You") offered an introduction (albeit rather brief) to the terminology and structure of propositional logic and predicate logic. Unfortunately, in the subsequent publications of database researchers and in the discussions that followed, it became clear to me that my efforts were simply too brief. This article serves as an informal introduction to the motivations and plan for what will ultimately be a book on logic with applications to relational database theory and practice, to be initially available for review in draft form as a series of articles.

Logic is often treated as a more or less elementary, albeit somewhat technical, subject. To some degree, I would agree. However, learning how to reason with logic in the abstract and learning to apply logic are very different endeavors. Certainly, it is possible for almost everyone to "be logical" at times – that is, to use elementary logic in their day to day thinking. On the other hand, doing this consistently is much more difficult and perhaps even undesirable in other day to day activities.

When reading about logic, we must always keep in mind that there are many, many different systems of logic – each with unique properties. If we want to apply logic, we must first decide which of these many systems to use for our intended purpose. Merely knowing how to reason within a particular logical system is inadequate. Our choice of formal logical systems will greatly influence our prospects for success or failure, inasmuch as that outcome will necessarily depend on the system's properties. This situation is true of formal systems other than logic and with which we may have passing familiarity. For example, there are many formal systems called

www.AlternativeTech.com mcgoveran@AlternativeTech.com

Post Office Box 4699, Deerfield Beach, FL 33442 Telephone: 831/338-4621

set theory, each with its own unique properties. What is *true* or *valid* in one formal system may not be *true* or *valid* in another formal system.

In technical fields such as electrical engineering and computing, hardware and software engineers learn to apply logic in developing algorithms and testing them: the former may be understood as logical deduction (programs as proofs[1]) and the latter as evaluation of formulae[2]. For the most part, engineers learn one or two distinct formal logical systems (though they subsequently may conflate them in practice), and how to write, prove, translate (e.g., to and from English), and evaluate the simple formulae of those logical systems. The theory taught engineers is almost always a gloss, focusing on those aspects of logic most commonly needed by engineers. In such instruction, little thought seems given to preparing engineers sufficiently that they will not be seduced into addressing issues pertaining to more complex formal logical systems using implicit, unqualified assumptions – and obtaining invalid results.

We'll see examples of such pitfalls later on, but they can be avoided by paying attention to the properties of formal logical systems and understanding what properties are required by a particular problem. (Again, we will see more specific examples of this later on when we have the tools to understand them.) Even if they do not make such errors, in practice they will – from time to time – fail to apply the discipline they have learned in a consistent manner across all aspects of a particular project or in the deployment and maintenance of the application.

In the database field, some understanding of logic is necessary to use a declarative query language such as the many dialects of the SQL query language, its extensions, and mimics. Simple expressions may be combined via logical connectives to yield more complex expressions which, when evaluated, yield a desired result. Those logical connectives must be used in a manner consistent with their formal definition and according to syntactic rules, which logicians call formation rules. With respect to a query language, syntactic correctness (which logicians call "well-formedness") is required and is checked by parsing. The query must be written such that the DBMS can evaluate the expression given the database and produce a reliable result. This is a process that requires the user to rely on technical properties of the underlying formal logical system even when not fully aware of those properties. Of course, the reliability of the underlying system may not be what the user or even the designer of the DBMS expects: every dialect of SQL of which I am aware provides the means to write syntactically correct statements that can produce surprising – sometimes even paradoxical – results when applied to certain data[3]. Understanding the foregoing requires a detailed analysis of the DBMS as implementing a formal logical system.

---

[1] This correspondence is called the Curry-Howard Equivalence.
[2] For example, suppose we have a simple program that takes a single input and produces a single output. Testing the program consists of specifying an input (i.e., an assignment to a variable) and then checking that the output matches that predicted. If the program is understood as a proof, this corresponds to evaluating the independent variable of the final line of the proof – the theorem – under the constraint that it be "true".
[3] This specific issue will be addressed in a later article.

www.AlternativeTech.com mcgoveran@AlternativeTech.com

Post Office Box 4699, Deerfield Beach, FL 33442 Telephone: 831/338-4621

It is easy to find examples in both electrical engineering and software engineering of systemic errors involving formal logical systems. New formal logical systems are proposed and implemented in hardware and software without first having elucidated the properties of those formal logical systems. While their utility applied to some specific application may be readily apparent to engineers, their limitations and the subsequent pitfalls in broader contexts are not. As a result, engineers and those who rely upon their design decisions (e.g., technology users, investors, company managers and stockholders, and indirectly, virtually every individual in our technology-dependent society) may find actual functional behavior not meeting expectations.

For example, fuzzy logic has found many legitimate uses in control systems and in modeling certain aspects of phenomena such as natural language. However, few engineers know either the meta-properties of fuzzy logic as a formal logical system. Neither are they aware that fuzzy logic is fundamentally at odds with certain logical constructs found in natural language expressions and so should be applied only in carefully constrained ways to avoid problems in that domain.[4] Another example exists in database theory and practice – the standard query language used in virtually all commercial DBMS products relies upon an unspecified formal logical system of a type known as many-valued logic. Its properties are not only unknown but unknowable, precisely because the formal logical system is ill-defined. One literally gambles when using such products! In a subsequent article we'll explore this specific example in greater detail and explain why commercial products manage to succeed financially despite manifest formal flaws.

On the other hand, there are many aspects of the theory of formal logical systems (known formally as *metamathematics* or *metalogic*) that are not so easily understood and upon which the elementary aspects of logic depend. To put it another way, what we are so frequently taught about logic – even in college level courses – is usually an enormous gloss. Sometimes the gloss is misleading. And sometimes it is just wrong. When taught the rudiments of any formal mathematical discipline (e.g., simple algebra), we are seldom introduced to issues that may concern experts in the foundations of mathematics nor do we have the prerequisites to understand them. The same situation applies to formal logical disciplines. The danger in being taught this way is that we become overconfident, mistakenly thinking our knowledge adequate to address all relevant problems pertaining to the (new to us) discipline. Like the old saying, we tend to use our familiar hammer to treat the new discipline like a nail. If it's a screw, we break it and don't obtain the benefits of its proper use. Sometimes we don't even realize the screw has been broken.

For example, in logic, we are taught a simplistic use of truth tables to evaluate the truth of a proposition. We are not told that truth assignment and evaluation are semantic issues nor are we

---

[4] McGoveran, D., "Fuzzy Logic and Non-distributive Truth Valuations" in Fuzzy Sets: Theory and Applications to Policy Analysis and Information Systems, P.P. Wang and S.K. Chang (editors), ©1980, Springer.

told that they are properly part of model theory (i.e., the theory of interpretations of the formal system) and not part of proof theory (i.e., the theory of deduction within the formal system). Neither are we informed of the limitations of truth tables[5] as an evaluation procedure. In consequence, we may erroneously think – as some database researchers apparently do – that *logical truth* reflects our commonly used notions of truth or falsity pertaining to some perceived, objective reality.

These subtle aspects of logic may not be particularly important consequences unless we have cause to engage in certain activities. However, when such activities are involved, a deep understanding of and respect for those subtleties becomes essential. Among those activities I would include, without implying any limitation, the following:

- Using a variation on a classical logic. By a classical logic I mean either two-valued propositional logic or two-valued first order predicate logic.

- Modifying a classical logic to create a different system of logic (adding truth values, changing inference rules, redefining terms, etc.).

- Designing a computing language – whether imperative or declarative, or something else, and whether for programming or query purposes. I would especially include language (e.g., SQL) standards.

- Creating, extending, or modifying a data model (relational or otherwise), let alone a data modeling methodology.

- Attempting to "mix" a system of logic and some other formal system (e.g., a version of set theory – of which there are many, arithmetic, abstract algebra, etc.).

- Discussing, understanding, or (possibly) avoiding paradox and its undesirable consequences.

- Discussing, understanding, and applying (let alone "designing") a type theory – that is, a systematic approach to abstract types or abstract data types.

- Discussing, understanding the consequences of, and safely applying recursion (which can appear in many forms).

- Discussing, understanding, or relying on the incompleteness theorems of Gödel, the halting problem, Church-Turing Thesis, decision problem, etc.

---

[5] For example, courses seldom introduce or motivate the method of semantic tableau.

www.AlternativeTech.com mcgoveran@AlternativeTech.com

Post Office Box 4699, Deerfield Beach, FL 33442 Telephone: 831/338-4621

I realize that readers may not understand or perhaps even recognize all the terminology in this list. If that is the case, do not worry: subsequent articles will provide definitions and examples.

II. IT'S ALL ABOUT FORMAL SYSTEMS

Throughout the articles to follow, we will be discussing the structure, properties, and applications of formal systems, of which those pertaining to various types of logic – formal logical systems – will ultimately be the most important. Formal systems can be qualified in various ways – they may be representational, deductive, uninterpreted or interpreted, and so on. We will define each of these types of formal systems in this series.

Engaging in activities like those in the foregoing bullet list requires a pretty good understanding of the more advanced aspects of formal systems and, more specifically, of formal logical systems. It is my intent and hope that the initial articles (approximately ten or so in number) in this series will provide at least a foundation for such understanding. I then intend to build on that foundation and apply it to database theory and possibly certain more general issues in computing.

Terms like *proof theory* and *model theory* (mentioned in passing above) refer to disciplines that address distinct aspects of all formal systems. If there is to be any hope of understanding and analyzing formal logical systems, these disciplines, and the components of formal systems to which they refer, must be kept separate. Otherwise, our reasoning will be flawed, often leading to subtle and ultimately grave consequences. Our understanding must be deep enough that we find distinguishing the proof theoretic aspects of logic from the model theoretic aspects of logic natural and almost automatic. Sometimes, given the properties of a particular formal logical system, one property (possibly *syntactic* and so *proof theoretic*) may logically imply another property (possibly *semantic* and so *model theoretic*) or vice-versa. Because of this, the literature often uses certain model theoretic terms interchangeably with proof theoretic terms without explicit qualification or restriction to the particular formal logical system – in consequence, the non-logician can be easily misled (if not completely confused).

Similarly, one needs to understand and carefully respect the crucial differences between (again, for example and without any intended limitation):

- deduction (or inference) versus computation
- object[6] language versus meta-language
- instantiation versus evaluation
- proposition versus predicate

www.AlternativeTech.com mcgoveran@AlternativeTech.com

Post Office Box 4699, Deerfield Beach, FL 33442 Telephone: 831/338-4621

- Boolean type (of a predicate variable) versus truth value
- contradiction versus paradox

III. THE USUAL APPROACH TO LOGIC

Traditional texts focus on teaching logic as an abstract system of deductive inference. Typically, some elementary terminology is introduced, the reader is taught how to construct formulae from primitive elements, how to apply rules of inference to given formulas to deduce new formulas, and how to evaluate the truth or falsity of formulas given the truth or falsity of their components. The reader may be given examples or exercises in applying logic to "real world" situations, such as giving (necessarily imprecise) translations of formulas into English[7], translating informal English assertions into formulas, and sometimes translating informal English arguments into formal deductions. Most often, the text will begin with propositional logic. From there, the text may introduce first order predicate logic, and possibly other systems of logic.

Usually, the overall approach such texts follow is one of teaching a formal language. In fact, the similarities to teaching a natural language are quite remarkable, the differences being due primarily to rigid rules of formal syntax replacing the grammar of a natural language. Note that, in this analogy, rules of inference (which govern the permissible structure of combinations of formula into deductive sequences) may be understood as corresponding to the grammatical rules for combining sentences into paragraphs and other structures.

If the text addresses metamathematical properties of logical systems such as consistency, completeness, and decidability, these subjects are most often given definitions and some brief explanatory discussion. Some texts then go on to discuss certain classic proofs of these properties. For example, the reader may be presented with an outline of the proofs of Gödel's Incompleteness Theorems. In consequence, the reader is usually left with the impression that the metamathematical properties of logical systems are almost an afterthought; while important, they seem secondary to manipulating formulae and evaluating their truth or falsity.

IV. A DIFFERENT APPROACH TO LOGIC AND ITS APPLICATIONS

This series follows a path to the understanding of logic and its applications that will be unfamiliar to most. It will first provide a structure for the understanding, evaluation, and use of formal systems in general and then follow this structure in presenting several formal logical systems. To be sure, a formal system is largely defined by its formal language. However, we cannot understand the properties of formal systems, let alone compare them and use them,

---

[6] For the record, throughout this series I will *never* use the term “object” in the sense it is used in “object oriented” in the software engineering, unless I give an example referring explicitly to “object oriented”.
[7] Any other natural language could have been used in these examples without invalidating the example.

www.AlternativeTech.com mcgoveran@AlternativeTech.com

Post Office Box 4699, Deerfield Beach, FL 33442 Telephone: 831/338-4621

without a more encompassing structure.

A preview of this approach can be seen by analogy with the study of natural languages. Imagine we want to translate a text about some subject from one natural language ("SL" for short) into another natural language ("OL" for short), the language of our intended result. The first language SL is called the *subject language*: It is the language used to write about the subject, using the vocabulary of the subject. The second language OL is called the *object language*: We translate the subject into the object.

In addition to the languages SL and OL, a translator needs a language in which to (a) describe or discuss the translation process, (b) think about, discuss, or evaluate the quality of the translation, and (c) validate the translation. For example, quite often natural language translators will use their native language for such purposes. This language may or may not coincide with either the subject language or the object language, but even if it does its use is quite different from the uses of those languages. The same situations apply to formal languages. A language used for such purposes is called a *meta-language* ("ML" for short).

There are numerous pitfalls that can beset the unwary translator. The translator must find *correspondences* C between terms of the SL and terms of the OL. These correspondences form a translation dictionary, expressed in the ML. If the language ML is not expressive enough to capture the nuances of both SL and OL, the translator is doomed from the start to a poor translation. If OL is not as expressive as SL, there will be words or phrases in the SL that are not in the OL. In such cases, "something is lost in translation" and an "under translation" will occur. If OL is more expressive than SL, there will be words or phrases in the OL that are not in the SL. In such cases, the translator must be careful not to import unintended constructs into the translation – sometimes called "over translating".

When translating among natural languages, it is not uncommon to find that some concepts translate well while others do not. The best translation that can be achieved in practice will be an under translation for some concepts while for others it will be an over translation. Translation problems such as those discussed here are even more likely if the OL is the formal language of some logical system, rather than a natural language.

Anyone who has ever explored the process known as *transliteration* knows how flawed the result can be in terms of failure to preserve the original intended meaning. For example, it was not until after Coca-Cola displayed its name in China as "Ke-kou-ke-la" that it discovered the literal translation meant "bite the wax tadpole" or "female horse stuffed with wax"! Kentucky Fried Chicken's slogan "finger-lickin' good" translated into Chinese became "eat your fingers off". And then there is the famous story of the Chevy Nova being marketed in Mexico, with no one noticing that "no va" in Spanish means "it won't go".

Now let's consider, again in preview, a special kind of formal translation called ***representation***. Representation is the process in which a formal system is used to represent and formalize a

www.AlternativeTech.com mcgoveran@AlternativeTech.com

Post Office Box 4699, Deerfield Beach, FL 33442 Telephone: 831/338-4621

second, possibly informal system as its subject.[8] It is a primary goal of this series to provide the reader with the tools necessary to establish a representation without introducing errors. At a high level of abstraction, database theorists must learn how to choose an appropriate formal system for representing and reasoning about some subject. At a lower level of abstraction, database practitioners must learn how to establish the representation so as to preserve the relevant properties of the subject – this is the essence of conceptual and logical modeling and ultimately database design.

Every subject has a corresponding subject language in which the *subject system*[9] is expressed. The object language we use to represent the subject belongs to a formal system, sometimes called the *object system*. Each of these languages will have certain assumptions and rules unique to them. More explicitly, every formal system which we use to represent or formalize some subject – the object system - has a formal language OL with its own assumptions AO (*a priori* "true" formulas or axioms) and rules RO. Similarly, the subject language SL may express assumptions AS about the subject and may have rules RS expressing its grammar or what constitutes a convincing argument. Again, elements of SL may be either formal or informal depending on the subject – for example, algebra or politics.

When we apply a formal system, we translate expressions in the language SL of a subject system into the language OL of the object system, establishing (and capturing in a dictionary) correspondences C between the terms of SL and the terms of OL. A particular correspondence need not be one to one; that is, a term in SL may correspond to a combination of terms in OL and a term in OL may correspond to a combination of terms in SL. The process of establishing the correspondences C is "observed", formalized, and described using a metalanguage ML. Every language has properties P (e.g., subject language properties SP and object language properties OP) that can only be determined, analyzed, expressed and compared using an ML.[10] We will use a formalization of these concepts to explain formal systems, their properties, and uses in this series.

In addition the problems encountered in trying to find the best correspondences between the terms of the SL and OL, we also have to contend with problems of preserving argument coherence. It might not be possible to obtain a faithful reproduction in OL of an argument presented in SL. When natural languages are involved, it is not uncommon to discover that an argument (or explanation) in the SL makes no sense in the OL, often because of contextual or cultural aspects of the natural language. In essence, the *rules* RS for what constitutes acceptable

---

[8] We will not use the term representation very often, introducing it here merely for pedagogical purposes. When a formal system *represents* some subject in a formal sense, we say that the subject is a *model* of the formal system. The formal specification of the relationship between the formal system and a model of it is called an *interpretation*.
[9] Notice that we have made no assumptions regarding whether the subject pertains to some alleged objective reality – we are only concerned here with the relationships between languages.
[10] Note that everything written herein is written in a metalanguage (technical English in this case) *by definition*: We are defining, describing and exploring the properties of our subject, formal systems and their application.

www.AlternativeTech.com mcgoveran@AlternativeTech.com

Post Office Box 4699, Deerfield Beach, FL 33442 Telephone: 831/338-4621

or convincing argument in language SL may not be the same as the rules RO followed in language OL.

As one final example of translation difficulties, the assumptions AS that one makes in SL might not be the same as those one makes in OL. How natural language text is to be understood – i.e., its meaning – often depends on the belief system of the user. Such assumptions are often implicit rather than explicit and users of the natural language are not even aware of them. In a formal argument, they are called *presuppositions*. Since the purpose of translation is to preserve meaning, taking assumptions underlying SL into account is crucial. Those assumptions must not only be translated from the SL into the OL, but must be compatible with those inherent in the OL (if any).

The foregoing are summarized in Figure 1.1 which shows – albeit rather schematically – the relationships between a metalanguage ML, an object language OL and a subject language SL. Aspects of the metalanguage ML are shown in the box at the top, of the object language OL in box at the left, and of the subject language SL in the box at the right.

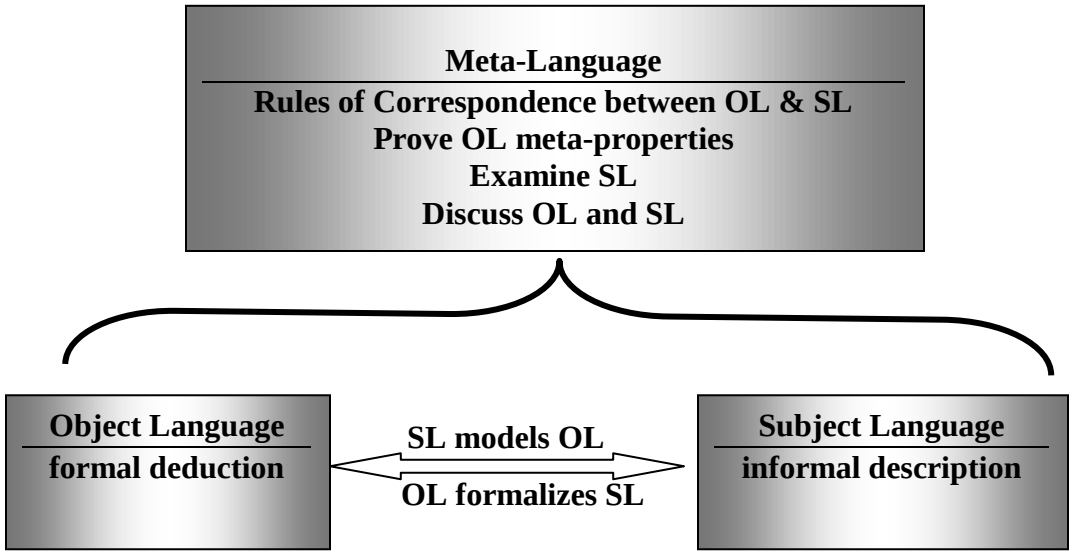

Figure 0.1: A Schematic Structure for Formal Systems

V. AN "ITERATIVE" PUBLICATION STRATEGY

Long ago, I stated my intent to write a book expounding on logic and its relevance to relational database theory and practice – or at least my thoughts on the subject. Unfortunately, life intervened and numerous personal issues prevented me from following up on that promise. So

www.AlternativeTech.com mcgoveran@AlternativeTech.com

Post Office Box 4699, Deerfield Beach, FL 33442 Telephone: 831/338-4621

much time has passed and life has taken me on so many unanticipated journeys, that, frankly, I am now a little reluctant to state that that book is in progress even though I am writing on the relevant topics. So, I've settled on a compromise: as a draft on some topic is completed, I will publish it as one of a series of articles. Then, when I am comfortable with the collection, I will edit them into a more cohesive text and articles will become chapters and the series will become a book.

This strategy runs certain risks and entails likely consequences, including:

- An article may refer to concepts that I have not yet written about (as, indeed, I have done in this introduction to the series). While I do not expect to make this a practice, I will sometimes do this intentionally by way of familiarizing the reader with otherwise unfamiliar terminology. I will provide an initial elementary or informal definition when the concept is introduced and, usually, a note that a more precise definition will follow if not immediately.

- The order in which articles initially appear may not provide the continuity desired and so will need to be improved in subsequent revisions.

- Explanation of some important concept might be inadvertently spread across two or more articles.

Despite these disadvantages, there are also potential advantages to the draft publication strategy:

- My readers can let me know if something is unclear.
- I don't have to find time to write the entire book up front.

The final product will hopefully be of higher quality than otherwise would be the case, at least in the sense of being more complete, more understandable, and with fewer oversights, glosses, or errors. At least, that is my hope and intent.

While I will attempt to be mindful of the opinions of authorities on my topics and any "standard" definitions of terms, I want to make it clear that I will be presenting my understanding, ideas, theories, and insights. I have no intentions of simply parroting material that can be read elsewhere. The material presented here is developed more or less afresh. From time to time, specific references may be cited as further explanation, source, or even some degree of external validation. However, this work is not an extension of any work by other authors and such works should not be presumed consistent with this series. Indeed, in a few cases, I will make it clear

www.AlternativeTech.com mcgoveran@AlternativeTech.com

Post Office Box 4699, Deerfield Beach, FL 33442 Telephone: 831/338-4621

that I explicitly disagree with the positions taken by others.[11] Some of the material presented in this series will be familiar, some reflective of the works of others, and some will even be original. I will strive for internal consistency among the concepts I present over familiarity or even compliance with any perceived authority. The reader is warned that the definitions presented here must be given priority over those previously learned if the text is to be understood.

Even if I were inclined to rely upon the explanations found in the literature, those explanations would not be suited for my purposes. Having researched the literature for over forty years, I have not found in any single text the combination of concepts addressed here treated with the care they are due. It is difficult to find authorities that have both a deep understanding of logic (or even formal systems in general) and of database theory, let alone database practice.[12] It is my strong opinion that gaping holes exist in the available formal treatments of the logical and mathematical underpinnings of database theory. Formal descriptions of the relational data model often introduce erroneous, inconsistent (with the assumed underlying formal logical system) or, at best, imprecise definitions. They sometimes assume and then rely on invalid or unprovable properties. I am aware of no literature that attempts to reconcile the process of database design (which properly involves what logicians call *semantics* and *model theory*) with DBMS design[13] and use (which properly involves what logicians call *meta-mathematics*, *syntactics*, and *proof theory*). Indeed, database design is usually given treatment as an art rather than a science! It is my intention to rectify this omission in the literature with the present work.

Given this attempt at originality, I of course run the risk of making suggestions or drawing conclusions that have been proven disadvantageous or even incorrect, respectively, by some authority, somewhere. If you find that to be the case, I eagerly anticipate being so informed with an appropriate reference. There is one caveat however, and that is that you make a good effort to be certain that the reference is truly applicable. Unfortunately, if you communicate a perceived issue, I need some assurance that any time spent on it is likely to be rewarding to both of us. Were my time less limited than it is, I would not put such a burden on my readers.

I will try very hard to define my terminology and use it consistently. Many of the terms that I will define are used differently in the literature, sometimes very informally, ambiguously, or

---

[11] This is particularly true regarding works on relational database theory and practice. For example, although I heartily support the general agenda of The Third Manifesto ("TTM"), I do not agree with some of its specifics or with the way in which formal systems are apparently understood by its authors. In this regard, any specific conflicts with TTM presented here are intended only as helpful criticisms of TTM.

[12] This observation is not meant to demean the work of such authors or their knowledge.

[13] Please note that I am *not* referring here to the design of the internal database used by the DBMS under the covers, but to the design of the DBMS itself, the software that must implement a formal object language of a logical system. Choosing to implement language "features" in a DBMS without understanding the specific formal logical system that will result (and its characteristics) leads to unexpected inconsistencies, limitations on applications, and functional limitations, not to mention characteristics incompatible with the relational algebra and the promise of the relational model. These topics are discussed at length in other chapters.

www.AlternativeTech.com mcgoveran@AlternativeTech.com

Post Office Box 4699, Deerfield Beach, FL 33442 Telephone: 831/338-4621

even erroneously. Understanding and clarifying the extant literature piece by piece is an impossible task, so please read this series and other sources with equally critical care. If you find that a collision occurs between your understanding of a term and my usage, try to be cognizant that such differences may be intentional and consider why I have defined the term as I have.

Throughout this series I will use various devices in an attempt to assist my readers – including footnotes, asides, emphasis, callouts, figures, and so on. I urge the reader not to ignore these, but give them your attention. When a term is first defined, it is given in ***bold and italics***. Other appearances of key terms will sometimes be *italicized* (but not bold), especially if I want the reader to note that it has a special technical definition. In some cases, a formal definition may not yet have been given, but I will make reasonable efforts to supply enough information for the reader to have a working definition in the interim. Do not fear: read on and all will become clear. From time to time, I will want to draw the reader's attention to a term, phrase, or sentence. In such cases, I will make use of underlining, possibly augmented with italics.

It is inevitable that, here and there and in the course of review, a topic, reference, figure or similar material may need to be added in a later version of an article. As I become aware of the need for such material, I will insert a short note in angle brackets as a placeholder until I am able to address the subject properly. I may also expand an article into multiple articles with more detail as the series develops, or in converting the series into a book (or possibly books).

Despite the foregoing, I do encourage my readers to read the articles in this series in order. While it is my intent to make each article dependent only on previously published draft articles, a few articles may become available out of order. As noted above, from time to time, I may feel it necessary to use a concept prior to discussing it, while providing a forward reference and a promise to go into more detail later in the article or even in a future article. And, of course, I may not succeed in these intentions despite my best efforts. Any such failings, errors, omissions, or other inadequacies are entirely my responsibility and I apologize for them in advance. I also promise to do my best to correct them if my readers will bring them to my attention.

VI. WHAT IS TO COME

The articles in this series will be divided into three parts. In Part I, all the foregoing potential problems with natural languages appear in similar form in the selection and use of formal systems. Their understanding and resolution is especially important if those formal systems are formal logical systems. We will formalize these potential problems of representation in the articles that follow, learning what is needed in a formal system to represent a subject, how to represent that subject, and how to know when something goes wrong and how to fix it.

After we study the structure of formal systems and their relationship to the subject matters they formalize, we'll learn how to describe the meta-properties of formal systems. We'll then learn about some important and specific formal systems, identifying their meta-properties.

www.AlternativeTech.com mcgoveran@AlternativeTech.com

Post Office Box 4699, Deerfield Beach, FL 33442 Telephone: 831/338-4621

In Part II of this series, we'll begin to apply all this knowledge to relational database theory, data modeling, and relational database design. While traditionally, the relational model has little to say about database design outside of normalization (which pertains only to the logical model), we will address the relationship between conceptual models, logical models, and physical models in light of the formal logical systems that underpin the relational model. We'll then review the relational model as it is traditionally understood, and go on to reconstruct it in the light of a new understanding of Codd's papers and our understanding of formal logical systems. Codd's papers contain numerous points that have been overlooked, most likely because his readers did not recognize the formal concepts to which he was referring and so missed their importance and potential.

In Part III, we will introduce some further applications of formal logical systems that are not normally considered part of the relational model and which will provide a very rich semantics. The relational model makes certain assumptions regarding what is or can be known about the subject represented by a database. These assumptions limit the utility of the relational model, especially with respect to knowledge discovery, complex analytics, and situations in which a rapidly evolving or changing logical model is implied. We will show how to extend relational semantics to deal with these situations. The result will be a new data model which is compatible with, but distinct from, the relational model.

You are about to begin a special journey in which I hope to convince you that there are previously unfamiliar treasures herein worth understanding. As you progress, return to Figure 0.1 frequently until you thoroughly understand it. It will often serve as a roadmap and protect you from going down perilous paths. Even if you find this journey laborious, I hope you find it cogent, viable, and rewarding.

[**Exercise**: In the foregoing discussion of object system and subject system, let the OL be elementary algebra and let the subject S be an inventory of coins (pennies, nickels, dimes, quarters, half dollars) as expressed in English. Give examples of expressions in OL, SL, rules of correspondence C, rules RO and RS, properties OP and SP, and at least one use of meta-language ML. How do all these examples change if S is changed to be an inventory of boxes of oranges?]

www.AlternativeTech.com mcgoveran@AlternativeTech.com

Post Office Box 4699, Deerfield Beach, FL 33442 Telephone: 831/338-4621

ACKNOWLEDGEMENTS

The author would like to thank reviewers Sevak Avakians (Intelligent Artifacts – www.intelligent-artifacts.com), Eugene Novagratsky, Chris Date, Hugh Darwen (www.thethirdmanifesto.com), Erwin Smout, and Fabian Pascal (www.dbdebunk.com) for their helpful comments and criticisms throughout the development of this series. Others also provided input to specific topics and will be acknowledged here as the series matures.

www.AlternativeTech.com mcgoveran@AlternativeTech.com

Post Office Box 4699, Deerfield Beach, FL 33442 Telephone: 831/338-4621