Database Normalization as a By-product of Minimum Message Length Inference

David L. Dowe and Nayyar Abbas Zaidi

Clayton School of I.T., Monash University, Clayton, Vic. 3800, Australia
{david.dowe,nayyar.zaidi}@infotech.monash.edu.au

Abstract. Database normalization is a central part of database design in which we re-organise the data stored so as to progressively ensure that as few anomalies occur as possible upon insertions, deletions and/or modifications. Successive normalizations of a database to higher normal forms continue to reduce the potential for such anomalies. We show here that database normalization follows as a consequence (or special case, or by-product) of the Minimum Message Length (MML) principle of machine learning and inductive inference. In other words, someone (previously) oblivious to database normalization but well-versed in MML could examine a database and - using MML considerations alone - normalise it, and even discover the notion of attribute inheritance.

Keywords: Minimum Message Length, MML, Database Normalization, Machine Learning, Data Mining, Intelligent Databases.

1 Introduction

The table is a basic building block of a Relational Database Management System (RDBMS) [1, 2]. Consequently, the structure of one or more tables in the database is of great interest. Typically, the information is structured into tables during the Entity-Relationship (ER) diagram phase of conceptual database design. Database normalization [3] is a process of evaluating and correcting table structures to minimize data redundancies, thereby reducing the likelihood of data anomalies upon insertion (adding a new row), deletion (deleting a row) or modification (modifying a row). Normalization is typically the heart of any database design activity.

Database normalization [2] works through a series of stages called normal forms (NFs) (described in sec. 3). A primary reason is to minimize data redundancies and get rid of update, delete and insertion anomalies. Also, designers would like to apply the 'minimal data rule' to the structure, making sure that all information has been captured and every piece of information captured is meaningful. In short, after the initial design is complete in the form of an Entity-Relationship (ER) diagram, designers generally analyze the relationships that exist among attributes within entities through normalization and improve the structure if need arises.

J. Li (Ed.): AI 2010, LNAI 6464, pp. 82–91, 2010.

As described, there can be many motivations behind a database normalization. In this paper, we present a novel information-theoretic perspective of database normalization. We consider the structure of the table(s) as a modelling problem for Minimum Message Length (MML) (see sec. 2). MML seeks a model giving the shortest two-part coding of model and data. If we consider table structure as a model which encodes data, MML advocates that we should be particularly interested in the variation of the encoding length of model and data as the normalization process re-structures tables for efficient design. We will consider a simple example and apply normalization to illustrate our point. As we will see in sec. 4, normalization into higher forms minimizes code length (or message length) by re-structuring the relational tables. Hence, if we apply the MML principle to a relational database, then - provided we have sufficient data - we are likely to get our database normalized.

The rest of the paper is organized as follows: we introduce the minimum message length (MML) framework in sec. 2. In sec. 3, we explain a typical normalization procedure with an example. We explain the MML view of normalization in sec. 4. We conclude in sec. 5.

2 Minimum Message Length

Minimum message length (MML), introduced by Wallace and Boulton [4], is an important stream of studying the complexity of a data set [4–10]. It is based on Shannon's theory of information and equivalently on the theory of Turing machines and Kolmogorov complexity [6, 7, 10]. MML considers any given string S as being a representation in some (unknown) code about the real world. It seeks a ([concatenated] two-part) string $I = H : A$ where the first part H specifies (or encodes) a hypothesis about the data S and the second part A is an encoding of the data using the encoded hypothesis. If the code or hypothesis is true, the encoding is efficient (like Huffman or arithmetic codes). According to Shannon's theory, the length of the string coding an event E in an optimally efficient code is given by $-\log_2(\text{Prob}(E))$, so the length of A is given by:

$$\#A = -\log_2(f(S|H)) \tag{1}$$

where $f(S|H)$ is the conditional probability (or statistical likelihood) of data S given the hypothesis H. Using an optimal code for specification, the length $\#H$ of the first part of the MML message is given by $-\log_2(h(H))$, where $h(\cdot)$ is the prior probability distribution over the set of possible hypotheses. Using equation (1), the total two-part message length $\#I$ is:

$$\begin{aligned} \#I = \#H + \#A &= -\log_2(h(H)) - \log_2(f(S|H)) \\ &= -\log_2(h(H) \times f(S|H)) \end{aligned} \tag{2}$$

The minimization of $\#I$ is equivalent to the maximization of $h(H) \times f(S|H) = \text{Prob}(H, S)$, that is the joint probability of hypothesis and data. It is thus formally equivalent to choosing the hypothesis H of highest Bayesian posterior *probability* (*not* a density) given S [11, secs. 2 and 6.1][8, sec. 11.3.1][9, footnote 158].

3 A Typical Normalization Procedure - An Example

As hinted in sec. 1, normalization works through a series of stages called normal forms. The first three stages are called first normal form (1NF), second normal form (2NF) and third normal form (3NF). From a structural point of view, 2NF is better than 1NF, and 3NF is in turn better than 2NF. For at least many purposes, 3NF is sufficient. A properly designed 3NF structure can also meet the requirements of higher normal forms - such as, e.g., Boyce-Codd Normal Form (BCNF), fourth normal form (4NF) or fifth normal form (5NF).

As an example, suppose we wish to create a database of university student enrolments. Let us assume that a student can take more than one unit, perhaps as many units as she likes. A student can also take the same unit more than once but not in one semester. Each student has a supervisor (denoted as 'lecturer' in the following discussion). A supervisor can have more than one student, but a student can only have one supervisor. Based on this information, we can proceed with the normalization procedure as follows, initially using this small example of $L = 11$ rows in 1NF:

1NF: The term 1NF describes a tabular data format where the following properties hold. First, all of the key attributes are defined. Second, there are no repeating groups in the table -i.e., in other words, each row/column intersection (or cell) contains one and only one value, not a set of values. Third, all attributes are dependent on the primary key (PK). Based on the information given to us, we can structure data in the table Student-Rec and can use Stud-ID, Unit-No and Yr-Sem attributes as parts of the PK. The table in 1NF is shown in table 1.

2NF: A table is in 2NF if the following conditions hold. First, it is in 1NF. Second, it includes no partial dependencies, that is no attribute is dependent on

Table 1. Student-Rec in 1NF. PK = (**Stud-ID**, **Unit-No**, **Yr-Sem**)

Stud-ID	Stud-Name	Stud-Address	Stud-Course	Unit-No	Unit-Name	Lect-No	Lect-Name	Yr-Sem	Grade
212	Bob Smith	Notting Hill	MIT	FIT2014	Database Design	47	Geoff Yu	2007	D
212	Bob Smith	Notting Hill	MIT	FIT3014	Algorithm Theory	47	Geoff Yu	2007	HD
212	Bob Smith	Notting Hill	MIT	EE1007	Circuit Design	47	Geoff Yu	2006	P
213	John News	Caufield	BSc	FIT3014	Algorithm Theory	122	June Matt	2007	HD
213	John News	Caufield	BSc	EE1007	Circuit Design	122	June Matt	2007	HD
214	Alice Neal	Clayton S	BSc	FIT2014	Database Design	122	June Matt	2007	HD
214	Alice Neal	Clayton S	BSc	FIT3014	Algorithm Theory	122	June Matt	2007	D
215	Jill Wong	Caufield	MIT	FIT2014	Database Design	47	Geoff Yu	2007	D
215	Jill Wong	Caufield	MIT	FIT2014	Database Design	47	Geoff Yu	2008	D
216	Ben Ng	Notting Hill	BA	EE1007	Circuit Design	47	June Matt	2007	P
216	Ben Ng	Notting Hill	BA	MT2110	Mathematics-II	47	June Matt	2007	D

Table 2. Student in 2NF. PK = **Stud-ID**

Stud-ID	Stud-Name	Stud-Address	Stud-Course	Lect-No	Lect-Name
212	Bob Smith	Notting Hill	MIT	47	Geoff Yu
213	John News	Caufield	BSc	122	June Matt
214	Alice Neal	Clayton S	BSc	47	Geoff Yu
215	Jill Wong	Caufield	MIT	47	Geoff Yu
216	Ben Ng	Notting Hill	BA	122	June Matt

Table 3. Unit in 2NF and 3NF, PK = **Unit-No**

Unit-No	Unit-Name
FIT2014	Database Design
FIT3014	Algorithm Theory
EE1007	Circuit Design
MT2110	Mathematics-II

Table 4. Stu-Unit-Rec in 2NF and 3NF. PK = (**Stud-ID**, **Unit-No**, **Yr-Sem**)

Stud-ID	Unit-No	Yr-Sem	Grade
212	FIT2014	2007	D
212	FIT3014	2007	HD
212	EE1007	2006	P
213	FIT3014	2007	HD
213	EE1007	2007	HD
214	FIT2014	2007	HD
214	FIT3014	2007	D
215	FIT2014	2007	D
215	FIT2014	2008	D
216	EE1007	2007	P
216	MT2110	2007	D

Table 5. Student in 3NF. PK = **Stud-ID**

Stud-ID	Stud-Name	Stud-Address	Stud-Course	Lect-No
212	Bob Smith	Notting Hill	MIT	47
213	John News	Caufield	BSc	122
214	Alice Neal	Clayton S	BSc	47
215	Jill Wong	Caufield	MIT	47
216	Ben Ng	Notting Hill	BA	122

only a portion of the primary key. The table Student-Rec in table 1 has partial dependencies. It is clear that Unit-Name only depends on Unit-No and not on the whole PK - that is, (Stud-ID, Unit-No,Yr-Sem). Also Name, Address, Course, Lect-No and Lect-Name depend only on Stud-ID. To structure the tables in 2NF, we need to eliminate these partial dependencies. A proposed design modification is shown in tables 2, 3 and 4. Table 1 is split into three tables: Student, Unit and Stu-Unit-Rec. Note that there are no partial dependencies in all of these three tables. In each table, each non-key attribute depends on all attributes in the PK. For our example, the 2NF tables are tables 2, 3 and 4.

3NF: A table is in 3NF if the following holds. First, it is in 2NF. Second, it contains no transitive dependencies. A transitive dependency exists when there are functional dependencies [1] such that $X \rightarrow Y$, $Y \rightarrow Z$ and X is the primary key attribute. Looking at our design in 2NF, there exists a transitive dependency in the Student table (table 2), where Stud-ID $\rightarrow$ Lect-No $\rightarrow$ Lect-Name. This transitive dependency can be eliminated by breaking the Student table into the Student and Lecturer tables as shown in tables 5 and 6. Note the Unit and Stu-Unit-Rec tables in tables 3 and 4 are already in 3NF. For our example, the 3NF tables are tables 3, 4, 5 and 6.

[1] The attribute B is fully functional dependent on the attribute A if each value of A determines one and only value of B.

Table 6. Lecturer in 3NF, PK = **Lect-No**

Lect-ID	Lect-Name
47	Geoff Yu
122	June Matt

4 MML Interpretation of Normalization

Our simple example of the normalization process from the previous section has (ultimately) resulted in four distinct tables - namely, Student (table 5), Lecturer (table 6), Unit (table 3), and Stu-Unit-Rec (table 4). Normalization is nothing but judicious re-structuring of information via tables.

We now flesh out ideas from [9, footnote 187], [12, pp454-455] and [10, sec. 7.6]. In an MML or information-theoretic version of normalization, we can think of tables as a certain model (or hypothesis). Following equation (2), we can write the first-part message length (encoding the model) as:

$$\#H = | < T > | + | < A > | + \sum_{t=1}^{T} AP_t \tag{3}$$

where T is the number of tables, A is the number of attributes, $< T >$ is an encoding of T, $< A >$ is an encoding of A, $| < T > |$ is the length of encoding T and $| < A > |$ is the length of encoding A. AP_t denotes the encoding length of table t's attributes and its primary key. It is defined in equation (4) as:

$$AP_t = \log_2(\mathrm{A}) + \log_2 \binom{A}{a_t} + \log_2(\mathrm{a_t}) + \log_2 \binom{a_t}{p_t} \tag{4}$$

where a_t is the number of attributes in the t^{th} table, p_t denotes the number of attributes in the primary key. (We know that $1 \leq a_t \leq A$, so $\log_2(A)$ is the cost of encoding a_t, and $\log_2 \binom{A}{a_t}$ is the cost of saying which particular a_t attributes are in the t^{th} table. Similarly, since $1 \leq p_t \leq a_t$, $\log_2 a_t$ is the cost of encoding p_t, and $\log_2 \binom{a_t}{p_t}$ is the cost of saying which particular p_t attributes are in the primary key of the t^{th} table.) Note that this is only one way of specifying the model. We have taken only the number of tables, attributes in each table and attributes constituting the PK in each table into account in specifying a model. Other models could be used. Note that the foreign keys (FKs) are not specified in this model - as the model encompasses information about the attributes in each table along with primary keys (PKs), the FKs can be found out by tracking the PK attribute of one table appearing in another table. For the sake of simplicity, we will not consider the effect(s) of $| < T > |$ and $| < A > |$ in the following discussion, as $| < A > |$ appears in the encoding of each normalized form. We could (and implicitly do) assume a uniform prior on $| < T > |$, but we could equally well instead have used (e.g.) a unary code ($\Pr(T) = 2^{-T}$, $| < T > | = T$) or the very slowly growing Wallace tree code [7, fig. 2.13 and sec. 2.1.14]. Hence, neglecting (near-)constant terms, we can (re-)write equation (3) as:

$$\#H = \sum_{t=1}^{T} AP_t \tag{5}$$

In the following discussion we will assume that there are m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8, m_9 and m_{10} unique instances of Stud-ID, Stud-Name, Stud-Address, Stud-Course, Unit-No, Unit-Name, Lect-No, Lect-Name, Yr-Sem and Grade respectively (refer to table 7).

Table 7. Number of unique instances for each attribute in table 1, 1NF of our initial example

Stud-ID	Stud-Name	Stud-Address	Stud-Course	**Unit-No**	Unit-Name	Lect-No	Lect-Name	**Yr-Sem**	Grade
m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}
5	5	5	5	4	4	2	2	3	3

The number of rows in the 1NF form of the table is an important variable. We have denoted it by L in the preceding equations. $L = 11$ in table 1 and depends on how many students are taking how many courses in each semester. We will later show that there is not a huge need for normalization if each student is taking only one unit, as 2NF will encode the same (amount of) information as 1NF. As more students take more courses, the need for normalization arises.

Let us consider data in the 1NF Student table in table 1. We can write the 1NF encoding length (I_{1NF}) as the sum of the length of model (H_{1NF}) and length of data (A_{1NF}) encoded by this model as follows:

$$\begin{aligned} I_{1NF} &= \#H_{1NF} + \#A_{1NF} \\ &= \#H_{1NF} + L \times (\log_2 m_1 + \log_2 m_2 + \log_2 m_3 + ... + \log_2 m_{10}) \end{aligned} \tag{6}$$

$\#H_{1NF}$ in the preceding equation (equation (6)) can be computed from equations (4) and (5). As there is only one table, $T = 1$. There are 10 attributes ($A = 10$) and 3 attributes in the primary key ($p = 3$).

Consider the three tables used here in 2NF - i.e., Student, Unit and Stu-Unit-Rec (shown in tables 2, 3, 4). We can write the 2NF encoding length (I_{2NF}) as the sum of the length of model (H_{2NF}) and length of data (A_{2NF}) encoded by this model. Examining the 3 tables and their attributes, this gives:

$$\begin{aligned} I_{2NF} &= \#H_{2NF} + \#A_{2NF} \\ &= \#H_{2NF} + m_1 \times (\log_2 m_1 + \log_2 m_2 + \log_2 m_3 + \log_2 m_4 + \log_2 m_7 + \log_2 m_8) \\ &\quad + m_5 \times (\log_2 m_5 + \log_2 m_6) \\ &\quad + L \times (\log_2 m_1 + \log_2 m_5 + \log_2 m_9 + \log_2 m_{10}) \end{aligned} \tag{7}$$

Like $\#H_{1NF}$ (from equation (6)), $\#H_{2NF}$ in the preceding equation (equation (7)) can also be computed from equations (4) and (5). There are 10 attributes ($A = 10$) in total and $T = 3$ tables. The Student table has 6 attributes ($a_1 = 6$) and 1 PK attribute ($p_1 = 1$). Similarly, the Unit table has 2 attributes ($a_2 = 2$) and 1 PK attribute ($p_2 = 1$). The Stu-Unit-Rec table has 4 attributes ($a_3 = 4$)

and 3 PK attributes ($p_3 = 3$). The $\#A_{2NF}$ part in equation (7) is the sum of the encoding lengths of the data in these 3 tables. Note the multiplication factors m_1, m_5 and L in the encoding term, since there are m_1 rows in the Student table, m_5 rows in the Unit table and L rows in the Stu-Unit-Rec table.

Moving from $2NF$ to $3NF$, the Student table in $2NF$ is split into Student (table 5) and Lecturer (table 6). We can write the 3NF encoding length (I_{3NF}) as the sum of the length of model (H_{3NF}) and length of data (A_{3NF}) encoded by this model, noting that we replace the cost of the student table (table 2) in 2NF from equation (7) with the costs of the new (and more concise) student table (table 5) and the lecturer table (table 6).

$$\begin{aligned} I_{3NF} &= \#H_{3NF} + \#A_{3NF} \\ &= \#H_{3NF} + m_1 \times (\log_2 m_1 + \log_2 m_2 + \log_2 m_3 + \log_2 m_4 + \log_2 m_7) \\ &\quad + m_7 \times (\log_2 m_7 + \log_2 m_8) \\ &\quad + m_5 \times (\log_2 m_5 + \log_2 m_6) \\ &\quad + L \times (\log_2 m_1 + \log_2 m_5 + \log_2 m_9 + \log_2 m_{10}) \end{aligned} \tag{8}$$

$\#H_{3NF}$ can also be computed from equations (4) and (5). There are $A = 10$ attributes and $T = 4$ tables in $3NF$. The Student table has 5 attributes ($a_1 = 5$) and 1 PK attribute ($p_1 = 1$). Since the Unit and Stu-Unit-Rec tables are already in $3NF$, we have $a_2 = 2, p_2 = 1, a_3 = 4$ and $p_3 = 3$ from the previous discussion. The Lecturer table has 2 attributes ($a_4 = 2$) and 1 PK attribute ($p_4 = 1$).

The encoding length of data along with the model for each NF for our initial small example (of only $L = 11$ rows in 1NF) is shown in table 8. As we have moved to higher NFs, we have made our model more complicated as depicted by the encoding length ($\#H$), but the data in the second part of the message ($\#A$) is encoded more efficiently and its length has vastly decreased. As can be seen from equations (6), (7) and (8), all encodings depend on the parameter L. We see an improvement of 2NF over 1NF even for $L = 11$ rows in this small example.

Table 8. Code length (bits) of model and data for different NFs on small example

	$\#H$ (first part's length)	$\#A$ (second part's length)	total message length
1NF	10.22	203.03	213.25
2NF	36.45	154.89	191.34
3NF	46.26	153.84	200.10

Due to space constraints we have not included a lot of data in table 1. In this particular example with $L = 11$ rows in 1NF, the total message length appears (slightly) higher for 3NF than 2NF. This should not be surprising considering the amount of data we have. Let us note from equations (7) and (8) that

$$I_{3NF} - I_{2NF} \;=\; (\#H_{3NF} - \#H_{2NF}) \;+\; m_7 \log_2 m_7 \;+\; (m_7 - m_1) \log_2 m_8$$

On the not unreasonable assumption that $m_7 = m_8$, then $I_{3NF} - I_{2NF} = (\#H_{3NF} - \#H_{2NF}) + (2m_7 - m_1)\log_2 m_7$.

Provided that $m_1 > 2m_7$ then, as m_1 or m_7 increases, the term $(2m_7 - m_1)\log_2 m_7$ will become increasingly negative, eventually becoming larger in magnitude than $(\#H_{3NF} - \#H_{2NF})$, whereupon I_{3NF} will become less than I_{2NF}, at which point MML will then forever after prefer 3NF to 2NF. This comparison between m_1 and $2m_7$ is because in going from 2NF to 3NF we are removing a column of m_1 entries in the 2NF Student table (table 2) and replacing it with a new 3NF table (table 6, Lecturer) of 2 columns and m_7 rows.

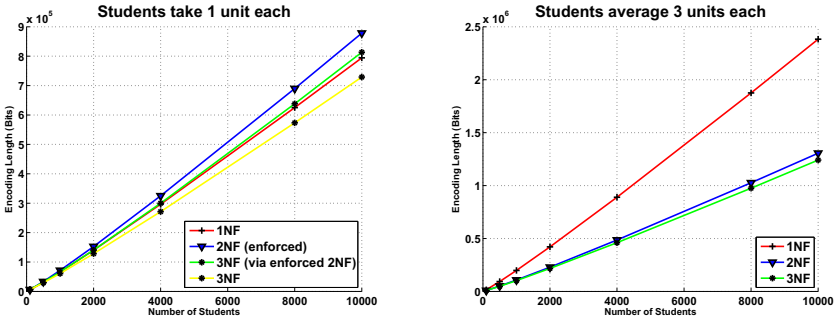
So, now let us suppose that we have a more realistic (but still quite small) example of $m_1 = 100$ students, $m_5 = 30$ units and each student is taking an average of 3 courses (note $L = 300$), setting the number of lecturers equal to $m_7 = 15$. The encoding lengths are given in table 9, which is also a cross-section of figure 1(b).

Table 9. Encoding length (in bits) of model and data for different NFs, Number of Students (m_1) = 100, Number of Units (m_5) = 30, Number of Lecturers (m_7) = 15, $L = 300$

	$\#H$ (first part's length)	$\#A$ (second part's length)	total message length
1NF	10.22	14210	14220
2NF	36.45	8150	8186
3NF	46.26	7876	7922

To illustrate this point graphically, in figure 1 we see the effect on encoding length by varying and increasing L and the number of students (m_1). If each student is only taking one unit ($m_1 = m_5$), 2NF will not be beneficial even if the number of students is increased from 10 to say 10000. This is depicted in figure 1(a). Because $L = m_1$ and there is insufficient data to infer the partial dependencies required for 2NF, the original 1NF table is adequate for 2NF. Indeed, enforcing the premature creation of (superfluous) tables (enforced 2NF) can understandably be seen to increase the message length. Despite this, the transitive dependencies of Stud-ID → Lect-No → Lect-Name (with $m_1 > 2m_7$) result in message length improvements when we go to 3NF. Taking the unnecessarily enforced 2NF is improved by then converting it to 3NF. But best of all is to take the original 1NF table as our 2NF (as there is insufficient data to suggest otherwise) and then convert this to 3NF. Figure 1(a) bears out this analysis.

As can be seen from figure 1(b) (of which the total message length in table 9 is a special case with $m_1 = 100$ and $L = 3m_1 = 300$), normalization from 1NF to 2NF is really beneficial as students enrol in more than one unit. The emphatic message length benefits visible in figure 1(b) in going from 1NF to 2NF are most probably to be expected, with the pronounced benefit of normalization being self-evident when the number of students (enrolling in more than one unit) is large. The transitive dependency that we observed in figure 1(a) applies again



(a) (b)

Fig. 1. Variation in total message length (I) by varying number of students (m_1) and L for different NFs. The number of Units (m_5) is set to 30 and the number of Lecturers (m_7) is set to 15. 1(a): $L = m_1$, 1(b): $L = 3m_1$.

here (with $m_1 \geq 50 > 2m_7 = 30$) as the number of students (m_1) and their enrolments increases, whereupon MML again prefers the 3NF model.

5 Conclusion and Future Work

We have presented database normalization as a consequence of MML inference. With an example, we demonstrated a typical normalization procedure and analyzed the process using the MML framework. We found that with higher NFs, the model is likely to become more complicated, but the data encoding length is decreased. If there is a relationship or dependency in the data (according to database normalisation principles), then - given sufficient data - MML will find this. This suggests that normalization is - in some sense - simply following MML.

Though we have limited ourselves here to 1st, 2nd and 3rd normal forms (NFs), applying MML can also be shown to lead to higher NFs such as Boyce-Codd Normal Form (BCNF), 4NF and 5NF. Indeed, recalling the notion of MML Bayesian network (see, e.g., [7, sec. 7.4][8][9, sec. 0.2.5 and footnotes 53, 75 & 218][10]), normalizing and breaking down tables into new tables can be thought of as a (MML) Bayesian net analysis - using the fact that (in some sense) databases could be said to have no noise. And, in similar manner, (the notion of) attribute inheritance (where different types of employee - such as pilot and engineer - have their own specific attributes as well as inheriting common employee attributes) can also be inferred using MML. General statistical consistency results (see, e.g., [8, sec. 11.3][9, secs. 0.2.3 - 0.3.1][12, pp436-437][10, sec. 5.2]) appear to guarantee that - given sufficiently large data-sets and sufficient search time - MML will converge upon the data generation process, whatever it is, whatever the appropriate (possibly previously unseen) normalization (or regularities). Our initial results here provide clear supporting evidence of this claim.

References

1. Codd, E.: A relational model of data for large shared data banks. Communications of the ACM (1979)
2. Date, C.: An Introduction to Database Systems. Addison-Wesley Longman, Amsterdam (1999)
3. William, K.: A simple guide to five normal forms in relational database theory. Communications of the ACM (1983)
4. Wallace, C.S., Boulton, D.M.: An information measure for classification. Computer Journal 11, 185–194 (1968)
5. Wallace, C.S., Freeman, P.R.: Estimation and inference by compact coding. Journal of the Royal Statistical Society series B 49(3), 240–252 (1987)
6. Wallace, C.S., Dowe, D.L.: Minimum message length and Kolmogorov complexity. Computer Journal 42(4), 270–283 (1999)
7. Wallace, C.S.: Statistical and Inductive Inference by Minimum Message Length. Information Science and Statistics. Springer, Heidelberg (May 2005)
8. Comley, J.W., Dowe, D.L.: Minimum message length and generalized Bayesian nets with asymmetric languages. In: Grünwald, P., Pitt, M.A., Myung, I.J. (eds.) Advances in Minimum Description Length: Theory and Applications, pp. 265–294. M.I.T. Press, Cambridge (April 2005)
9. Dowe, D.L.: Foreword re C. S. Wallace. Computer Journal 51(5), 523–560 (2008); Christopher Stewart WALLACE (1933-2004) memorial special issue
10. Dowe, D.L.: MML, hybrid Bayesian network graphical models, statistical consistency, invariance and uniqueness. In: Wood, J., Forster, M.R., Bandyopadhyay, P. (eds.) Handbook of the Philosophy of Science - (HPS Volume 7) Philosophy of Statistics, vol. 7, pp. 861–942. Elsevier, Amsterdam (2010)
11. Wallace, C.S., Dowe, D.L.: MML clustering of multi-state, Poisson, von Mises circular and Gaussian distributions. Statistics and Computing 10, 73–83 (2000)
12. Dowe, D.L.: Minimum Message Length and statistically consistent invariant (objective?) Bayesian probabilistic inference - from (medical) "evidence". Social Epistemology 22(4), 433–460 (2008)