Codd's Twelve Rules

Derek Ignatius Asirvadem

In a 1985 *ComputerWorld* article, Dr E F Codd presented twelve rules that a database must obey, if it is to be considered truly relational. C J Date is credited with curating Codd's work after the latter's death, but as evidenced he and his minions market the *Relational Model/Tasmania* [1] [2] (which Codd intended for giving the then physical pointer-based systems a bit of relational capability, and which contradicts the *RM*, implements physical 1960's Record Filing Systems, which Codd intended it for) as "relational", and thus suppresses the *RM*. Thus we have Codd's *Relational Model* and Date's Anti-Relational Muddle.

During the early 1990s, it became popular practice to compile 'scorecards' for commercial DBMS products, showing how well they satisfy each of the rules. Unfortunately, the rules are subjective so the scorecards were usually full of footnotes and qualifications, and didn't reveal a great deal about the products. Today, the basis of competition for database vendors tends to revolve around performance, new features, the availability of development tools, the quality of vendor support, and other issues, rather than conformance to Codd's rules. Nonetheless, they are an important part of the history of the relational model, and their compliance indicates just how relational the evaluated database really is.

Pre-Relational Databases

Hierarchical and Network Databases were well-established, and vendors already had a secured market. In those days computer systems and disk space were expensive; IT staff were qualified, more professional, and standards and rules were highly regarded. The implementation of a database was undertaken with sobriety, as the difficulty and expense of changing it after the fact was considerable. Due to these factors, Normalisation was considered essential, normal, and it was correctly performed.

Normalisation

It needs to be understood that Normalisation pre-dated the Relational paradigm. The purpose (end result) of Normalisation is **elimination** of duplicate data, not merely reduction. This is in order to **eliminate** Update Anomalies. It was an absolute requirement for Pre-Relational databases, although the implementation was physical and product-specific, and it remains an absolute requirement for Relational databases. Codd's Twelve Rules assumes that pure Normalisation (at least **Codd's** Third Normal Form [3]) has been applied, and adds specific requirements for the Relational paradigm [2]. Thus Normalisation is not addressed in his Twelve Rules.

Relational

The *Relational Model* is not explained here, a short summary of major differences is provided. It is not merely "all data is presented as tables, rows and columns" (which is a naïve understanding of **Rule 1**, and taken as the *only* rule, beloved of the academics).

1960's Record Filing System (RM/T)	Relational Model	Note
All aspects are physical	All aspects are logical	In Modern times, The Logical is suppressed
"Primary Key" = Record Id	Primary Key = Relational Key (composite)	The notion of *Key* is lost
All files are Independent	Hierarchic (from Hierarchic DBMS) Access Path Independence (from Network DBMS)	
All connections are Non-Identifying	Identifying & Non-Identifying relations	As provided in IDEF1X
Referential Integrity (physical)	Relational Integrity (logical)	
Access Path Dependence	Relational Power	
Nil	Relational Speed	

1 It took ten years of argumentation with the academics of the day, who were pathetically decades behind the DBMS platform providers, to accept the RM, because they were stuck in physical pointer-based systems. In that effort, Codd wrote Relational Model/Tasmania, to demonstrate that even primitive pointer-based systems could benefit from a few of the logical features of the RM. It is not a substitute for the RM, indeed it contradicts the RM, it adds nothing to the RM, which is complete. It remains valid for its purpose: a fragment of the RM intended to elevate the functionality physical pointer-based systems. However, the academics, now being fifty years behind the RM and forty years behind implementations (genuine SQL platforms) only understand and implement RM/T, and actively suppress the RM.

2 Note gravely, there are two explicit Normal Forms in the Relational Model, which relational practitioners understand and use, but in fifty years, including over 200 papers purportedly about the RM, Date and his cohort of imbeciles have not mentioned it, let alone articulated it. Again, evidence of the suppression of the RM and misrepresentation of the RM/T as the "relational model".

3 4NF; BCNF; 5NF are relevant only for primitive Record Filing Systems marketed by the academics. Codd's 3NF, which is Full Functional Dependence (as distinct from the ever-changing "definition" marketed by academics, which gyrates about "transitive" and "partial" functional dependence) in a Relational context does not require it, it is superfluous.

Performance
Complete compliance to Codd's *Relational Model & Twelve Rules,* as well as his OLAP paper, produces a Relational database of the highest order, with excellent OLTP & OLAP performance.

Note that we have had ACID Transactions, that is to say, a full OLTP paradigm including methods and rules, since the 1960's. The SQL platforms have provided that from their first versions in the 1980's [4].

Compliance Grid
The first two columns define the requirement of a Relational DBMS platform [4], the third identifies the level of compliance of Relational Database designed by Software Gems, to each of Codd's Twelve Rules, along with explanations and qualifications. It is the implementation of our *Software Gems Quality* and *Performance Standards* at the database level, which is delivered in every database design assignment. This can also be stated as, the exact boundary between the Relational paradigm and established OLTP standards (of which our *Transaction Standard* is a particular). (The theory-only mindset cannot be applied to an implementation, particularly when the academics that concoct the theory are clueless about practice, or worse, applying it is an absurdity).

Definition	Explanation	Compliance
1 Information rule All information in a relational database is represented explicitly at the logical level and in exactly one way: by values in tables.	Basically the informal, naïve definition of a relational database.	Full compliance
2 Guaranteed access rule Each and every datum (atomic value) in a relational database is guaranteed to be logically accessible by resorting to a combination of table name, primary key value, and column name.	Stresses the importance of primary keys for locating data in the database. The table name locates the correct table, the column name finds the correct column, and the primary key value finds the row containing an individual data item of interest.	Full compliance
3 Systematic treatment of null values Null values (distinct from an empty character string or a string of blank characters and distinct from zero or any other number) are supported in a fully relational DBMS for representing missing information and inapplicable information in a systematic way, independent of the data type.	Requires support for missing data through NULL values.	Rejected • "Three-valued logic" breaks the Law of the Excluded Middle, it is dismissed • Nulls are not stored • Null appears in Views only
4 Dynamic online catalog based on the relational model The database description is represented at the logical level in the same way as ordinary data, so that authorised users can apply the same relational language to its interrogation as they apply to the regular data.	Requires that a relational database be self-describing. In other words, the database must contain certain system tables whose columns describe the structure of the database itself.	Full compliance • All control values are presented in tables • Security is implemented as an extension of the catalogue

4 Note that only IBM/DB2; SAP/Sybase; and MS SQL are SQL compliant platforms. Oracle and the freeware herds are neither.
While the list of non-compliant issues is endless, and noting that they change with every major version, the worst and most crippling issue is their offline anti-transaction mindset "MVCC", which is a fraudulent term because it does no concurrency control whatsoever, it merely attempts to resolve the multiple stale offline versions when someone COMMITS. ACID is not possible, and thus OLTP is not possible. "MVCC" is a single-user mindset, which is all that Stonebraker could understand. The adorable mantra that *readers do not block writers, writers do not block readers* is stupefying because the versions are offline and thus nothing blocks, the pain happens when reality bites, at a COMMIT. The fantasy of eliminating a Lock Manager is hysterically false because it does use a Lock Manager to resolve the inevitable conflicts.

Definition	Explanation	Compliance
5 Comprehensive data sublanguage rule A relational system may support several languages and various modes of terminal use (for example, the fill-in-the-blanks mode). However, there must be at least one language whose statements are expressible, per some well-defined syntax, as character strings, and that is comprehensive in supporting all of the following items: • Data (table) and View definition • Data manipulation (interactive and by program) • Integrity constraints • Authorisation • Transaction boundaries (begin, commit, and rollback)	Mandates using a relational database language, such as SQL, although SQL is not specifically required. The language must be able to support all the central functions of a DBMS - creating a database, retrieving and entering data, implementing database security, and so on. The wording of that rule was relevant in 1985 when SQL was not quite established, today it is the *de facto* standard.	Full compliance: SQL • Faithful extension of the catalogue and security facilities • All updates must be ACID Transactions (stored procs) • "DKNF" as Codd intended, as distinct from the hilarious academic definition, is provided
6 View updating rule All views that are theoretically update-able are also update-able by the system.	Views are virtual tables, de-normalised by definition. It is one of the most challenging rules to implement in practice, and no commercial product fully satisfies it today. Precisely because it is stupid.	Rejected • No update via Views (it is a theory-only requirement) • All updates must be ACID Transactions (stored procs)
7 High-level insert, update, and delete The capability of handling a base relation or a derived relation as a single operand applies not only to the retrieval of data but also to the insertion, update, and deletion of data.	Stresses the set-oriented nature of a relational database. It requires that rows be treated as sets in insert, delete, and update operations. The rule is designed to prohibit implementations that only support row-at-a-time, navigational modification of the database.	Full compliance • All updates must be ACID Transactions (stored procs)
8 Physical data independence Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representations or access methods.	Insulates the user or application program from the low-level implementation of the database. Mandates that specific access or storage techniques used by the DBMS, and changes to the structure of the tables in the database, should not affect the user's ability to work with the data.	Full compliance
9 Logical data independence Application programs and terminal activities remain logically unimpaired when information preserving changes of any kind that theoretically permit unimpairment are made to the base tables.	The essential rule mandates that the database must be completely independent of the application(s). Insulates the user or application programs from the logical organisation of the database.	Full compliance • True Open Architecture Database • Column list required
10 Integrity independence Integrity constraints specific to a particular relational database must be definable in the relational data sublanguage and storable in the catalog, not in the application programs.	The database language must support integrity constraints that restrict the data that can be entered into the database and the database modifications that can be made.	Full compliance
11 Distribution independence A relational DBMS has distribution independence.	The database language must be able to manipulate distributed data location on other computer systems.	• (Open to interpretation, and subject to the greater deployment options available in modern products) • Single server DTM model

Definition	Explanation	Compliance
12 Nonsubversion rule If a relational system has a low-level (single record at a time) language, that low level cannot be used to subvert or bypass the integrity rules and constraints expressed in the higher level relational language (multiple records at a time).	Prevents "other paths" into the database that might subvert the relational structure and integrity.	Full compliance