Codd's Twelve Rules

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 realising Codd’s work after the latter’s death.

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, Network and Hybrid 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. 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 Third Normal Form) has been applied, and adds specific requirements for the Relational paradigm. Thus Normalisation is not addressed in his Twelve Rules.

**Relational**
The main, and the only relevant difference, between Pre-Relational and Relational databases is that the relations are carried by (logical) keys, instead of (physical) record identifiers, the latter being significantly faster.

**Performance**
Complete compliance to Codd’s Twelve Rules, while producing a Relational database of the highest order, results in poor performance with respect to OLTP considerations. Software Gems implements true Relational databases with faithful compliance to the Relational paradigm, but with certain (performance oriented) qualifications to Codd’s Twelve Rules.

**Compliance Grid**
This grid 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).
<table>
<thead>
<tr>
<th>Definition</th>
<th>Explanation</th>
<th>Compliance</th>
</tr>
</thead>
</table>
| **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 definition of a relational database.                                          | 100%       |
| **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. | 100%       |
| **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.                                              |            |
| **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. | 100%       |
| **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 definition  
• View definition  
• Data manipulation (interactive and by program)  
• Integrity constraints  
• Authorization  
• 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. | 100% SQL  
• Faithful extension of the catalogue and security facilities  
• All updates must be transactional, via stored procs |
| **6 View updating rule.**  
All views that are theoretically updateable are also updateable by the system. | Deals with views, which are virtual tables used to give various users of a database different views of  
its structure. It is one of the most challenging rules to implement in practice, and no  
commercial product fully satisfies it today. | 0%  
No update via Views |
|   | High-level insert, update, and delete. | 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. | 100%  
All updates must be transactional, via stored procedures |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>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.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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. |  
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. |
|   | 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. Specifies 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. | 100% |
|   | Insulates the user or application program from the low-level implementation of the database. Specifies 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. | Insulates the user or application program from the low-level implementation of the database. Specifies 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. | 100% |
|   | 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 that 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. | 100%  
• True Open Database  
• Column list required |
|   | The essential rule that 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. | The essential rule that 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. | 100%  
• True Open Database  
• Column list required |
|   | 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. | 100% |
|   | 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. | 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. | 100% |
|   | 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 greater deployment options available in modern products)  
• Single server DTM model |
|   | The database language must be able to manipulate distributed data location on other computer systems. | The database language must be able to manipulate distributed data location on other computer systems. | (Open to interpretation, and subject to greater deployment options available in modern products)  
• Single server DTM model |
|   | 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. | 100% |
|   | Prevents "other paths" into the database that might subvert the relational structure and integrity. | Prevents "other paths" into the database that might subvert the relational structure and integrity. | 100% |