Relational model defined by codds twelve rules computer science essay

Technology, Computer

ASSIGN BUSTER

This report tries to explain what Codds Twelve Rules means. And by comparing MySQL with relational model as defined by Codd’s Twelve Rules, this report also gives an abstract view on how MySQL comply with Codd’s Twelve Rules. This report is based on MySQL 5 InnoDB engine.

Edgar F. Codd is famous for his contribution to relational model of database in 1970s. However, in 1980s the term “ relational” was used by many database vendors to describe their database products which may not comply with the model that Edgar F. Codd has proposed. In order to clarify his model of relational database, and provide people a simple standard that can indicate to what extent a database software conforms to his model, the Codd’s Twelve Rules were propose.

There are 13 rules in Codd’s Twelve Rules. Our textbook omits the first one,’rule 0?, so this report will start from the second one in Codd’s rules, ‘ rule 1’.

Rule 1: The Information Rule

This rule requires all data in relational database management system(RDBMS) should be stored as values in tables at logical level. Some DBMS use Key-Value to store data, ‘ Redis’ for example, which contradict the Information Rule, so these DBMS will not be regarded as relational DBMS.

MySQL dose store all data in the form of tables with values in columns of rows. Users can only access to values that are stored in tables. Even the data descript the database itself is store in tables, i. e. table ‘ tables’ in ‘

Information schema’ stores the description of all the tables that have been created. So, MySQL meets the requirement of rule 1.

Rule 2: The Guaranteed Access Rule

Users must be able to access to values by providing table name, the value of primary key and the name of the columns. In another word, the DBMS should support primary key in tables and enforce each tables contains primary key in order to prevent data duplication.

MySQL does support to define primary key in tables. Yet, users can also create tables that don’t have it. For example, create one table has columns ‘ a’ and ‘ b’ without primary key. In that circumstance, there may be several rows that has the same value in column ‘ a’ , preventing users to access to the value of column ‘ b’ in the row he want. So, MySQL does not fulfill the requirement of Rule2 and it gives user more flexibility by accepting tables without primary key.

Rule 3: Systematic Treatment of NULL Values:

The database must support ‘ NULL’ as a value other than ‘ 0’ or ’empty string’, as a representation of “ data missing or inapplicable”. And the database can provide systematic way to manipulate NULL value.

MySQL fulfill this requirement by supporting NULL value and treat it in a systematic way. In MySQL, ‘ NULL’ is supported and is regarded as missing data following ANSI/ODBC SQL standard. MySQL implements ternary logic. Users can not compare values with NULL, even NULL with NULL by using ‘=’, because NULL is missing data. The results of those compares are ‘ unknown’. MySQL provides ‘ IS NULL’ and ‘ IS NOT NULL’ statement in order to treat the compares with value ‘ NULL’.

Rule 4: Active online catalog based on the relational model

Data dictionary of one DBMS should be stored as ordinary data in the form of tables. Authorized users must be able to using the query language (SQL for example) that they used to query ordinary data to access to database catalog or structure.

MySQL stores database catalog data using tables — the same way it store ordinary data. These tables are in system database such as ‘ Information_schema’. For example, table ‘ tables’ in ‘ Information_schema’ contains information about all tables in MySQL, like ‘ TABLE_NAME’, ‘ TABLE_TYPE’. Authorized users can use SQL to query this table in order to access to data catalog of current tables. So, MySQL well implements this Rule.

Rule 5: Comprehensive data language

The DBMS must support at least one language that can be used directly by users or within application queries. This language must also supports all aspects of database use including data (view) definition, data manipulations, integrity constraints, securities and transaction managements. SQL is a language that is comprehensive enough to support all these requirements. So, any DBMS that implements ANSI/ODBC SQL will comply with this rule.

MySQL follows the ANSI/ODBC SQL standard, yet there are several differences between them in several cases. The difference can be seen in documents of MySQL. All these differences are just about statement syntax, i. e MySQL doesn’t support ‘ select A? a,¬A¦ into table’, users should using ‘ Insert into A? a,¬A¦ select’ to do the same works. But after all, all database use in MySQL can be implemented by using SQL regardless of whether the syntax is different from standard SQL. So, MySQL fulfills Rule 5.

Rule 6: View Updating

This rule means that the alteration that user makes in a view will result in the alteration of tables from which the view is created, if this view is theoretically updatable.

In MySQL, many theoretically updatable views can be updated, yet, there are many limits. For example, due to the documentation of MySQL, ‘ delete’ and ‘ update’ cannot be used to update a view that has more than one underlying table. So, MySQL does not fulfill this rule.

Rule 7: The RDBMS may handle individual records but it must primarily handle sets of records

This rule means users can use one single command to query, insert, delete and update sets of values in multiple rows or multiple tables.

MySQL can handle operation of multiple rows in one table. Because it uses SQL, that has commands that can handle operation of sets of records, as its data language. For example, MySQL can insert multiple records with this statement, ‘ INSERT INTO table_name (a, b, c) VALUES (3, 4, 5), (6, 7, 8)A? a‚¬Â¦A? a‚¬Â¦’. But MySQL cannot handle operation of sets of records that are from different tables in one command. But users can also handle this issue by using transaction that containing a series of SQL commands. So, MySQL implements this rule by allowing user to operate command on multiple rows in one table, while does not support operation of multiple tables in single command.

Rule 8: Physical Data Independence

This rule means that alterations that have been made to database in physical level, for example, export one database, and open it in another computer will not result in the changes in logical level. And users can still access to the data without altering their commands.

MySQL can export one database by creating ‘ back up’ file. This file can be restore by MySQL in another computer. The physical underlying of this database has changed while the table structure will not be changed and users can access to this restored one without any adjustment on their queries. So, MySQL does provide some extent of physical data independence in InnoDB engine. However, if users want to change the store engine of a table from transactional one to non-transactional, the logical level will also change. In sum, MySQL provide physical data independence in InnoDB engine, but changing the store engine may result in change in database logic.

Rule 9: Logical Data Independence

This rule means that the changes of logical level in the database will not lead to changes of queries that based on former structure. For example, users can split one table into two, while use the same query as before.

In MySQL, adding columns to a table will not require changes in application or queries that are base on the structure of this table. However, other changes of logical level, such as combine two tables into one, may call for an alteration of the application based on the structure. So, MySQL does not comply with this Logical Data Independence rule.

Rule 10: Integrity Independence

This means that integrity definition of data in one DBMS should be regarded as one part of data dictionary, and be stored in the same form as ordinary data. This also requires that this integrity definition can be access by users using language, SQL for example, to query, define or alter the integrity independence.

MySQL fulfills this rule. It stores data dictionary in tables in ‘ information schema’. For example, the column ‘ COLUMN_KEY’ in the table ‘ COLUMNS’ defines whether this column is primary key or has other constraints. And ‘ KEY_COLUMN_USAGE’ table defines which key columns have constrains. Users can access to integrity definition data by query these tables using ordinary SQL statement.

Rule 11: Distribution Independence

Today, many DBMS introduce the function to using distribute data in different locations. However, due to this rule, where this data be distributed and how DBMS manage them should not be visible to users. Users can use the data in the same way as they use data that been stored in one place.

The InnoDB engine does not provide the ability to store data in different locations. MySQL has a distributed engine called MySQL Cluster. In InnoDB engine, MySQL introduce XA Transaction which is based on X/Open XA specification since 5. 0. 3. This specification provides users the ability to employ multiple resources in one transaction. However, users must know the underlying works, and if the structure of the distributed DBMS changes, the XA Transaction statement may also need to be adjusted. So, MySQL does not comply with the rule 11.

Rule 12: The Nonsubversion Rule

Sometime the DBMS provide API or other low-level interface for users to handle complicated transactions. However, those interfaces must not break all the rule above and bypassing integrity constraints and security.

MySQL provides APIs for different applications or programming languages as low-level interface. There are back doors in them, custom command ' SHOW' for example. However, these backdoors are only maintained for the compatibility with the former edition.

Summary

In Sum, due to the comparison between MySQl and Codd’s rules, MySQL implement most of these rules, though there are still some limitations. It can be regarded as a DBMS that is relational.