Is Actian PSQL a Relational Database Server?

A Technical Whitepaper

Rick F. van der Lans
Independent Business Intelligence Analyst
R20/Consultancy

March 2014

Sponsored by

Table of Contents

1 Management Summary

In this era, when NoSQL products are receiving a lot of attention in the trade press, SQL is still, by far, the dominant database language. To paraphrase Michael Stonebraker, a database pioneer, SQL is *intergalactic dataspeak*. Most databases on which organizations rely today, are SQL-based.

The SQL language is an implementation of the *relational model*. The relational model is a method for organizing, querying, and manipulating data. It's set-oriented and based on first-predicate logic. Other database languages based on the relational model exist, such as Quel, Square, QBE, and D4.

With Codd's 12 rules it's possible to measure how relational a database server is.

In 1985, *E.F. Codd*, the founder of the relational model, defined a set of rules[1] for determining how well a database product supports the relational model. These rules make it possible to answer the question whether a particular product is a *relational* database server. They were urgently needed, because many vendors were labeling their products as relational, while they were not. So, the term relational became somewhat polluted and Codd wanted to fix and prevent this.

Actian PSQL is 83% relationally complete.

This whitepaper explains Codd's rules and shows how well *Actian's PSQL* database server supports them. The verdict is that PSQL scores a 10 (on a scale of 0 to 12). Nine rules are fully supported, two partially, and two not. Therefore, the overall conclusion is that PSQL is 83% relationally complete. This is an excellent score and puts PSQL in the list of most relational products that are out there.

Is it possible to be 100% relational? The answer is yes. Such products can be developed. In fact, there is one open source product that supports most of the rules: Alphora's DataPhor. However, the product is not (yet) a commercial success. In 1985, when Codd introduced these rules, he also wrote "No existing DBMS product that I know of can honestly be claimed to be fully relational, at this time." It looks as if this statement still holds for all the SQL products and probably for most database servers.

2 What is a Database Server?

What is a Database? – To be able to answer the question "Is PSQL a relational database server?", we have to answer the question "What is a database server?" first. And to answer that question, we have to know the answer to the following one "What is a database?".

If we type in the term database in Google, the number of hits is over 1,3 billion. This means that it's a widely used term. Therefore, you would expect that the term has been well-defined. Strangely enough, that's not the case. One of the reasons that different definitions exist, is that the concept can be viewed from two angles: from a technical and a more logical angle. From the technical angle, a database can be seen as all the data stored in one physical file, or the

[1] E.F. Codd, *Is Your Database Management System Really Relational? An Evaluation Scheme*, The Relational Institute, TRI Technical Report EFC-1 /10-28-86, October 28, 1985.

data in all the files managed by some software product, such as a database server. From a more logical angle, a database can be seen as all the data that users have access to, which doesn't have to correspond with the more technical view.

The definition used in this whitepaper is the one introduced by C.J. Date[2] in his renowned book "An Introduction to Database Systems":

> *A database consists of some collection of persistent data that is used by the application systems of some given enterprise and managed by a database management system.*

What is a Database Server? – Just like the term database, the term database server is popular as well. Everyone in the IT field has a feeling of what a database server is. Still, coming up with a generally accepted definition is not straightforward. The following one is used in this whitepaper:

> *A database server is a computer program that provides database services to applications.*

The next important question is what are database services? Most specialists expect a database server to support minimally the following services:

- Data definition services
- Data query services
- Data manipulation services
- Transaction management services
- Data security services
- Backup and recovery services
- Multi-user concurrency services
- Monitoring services

A database server supports at least data definition, query, data manipulation, transaction management, data security, and backup and recovery services.

Does this imply that if a product supports all the above services except for one or two, it's not a database server? For example, SQLite[3] supports all the above services except multi-user concurrency and monitoring services. Is it therefore not a database server? Many SQLite specialists would strongly disagree. In this whitepaper we assume that a database server has to support at least the first six services of the list.

What is a Database Management System? – Many of the articles and books referenced in this whitepaper are relatively old. In most of them the term database server is not used, but *database management system* instead. The two terms are considered synonyms of each other. Nowadays, the more popular term is database server, so we predominantly use that one.

[2] Chris J. Date, *An Introduction to Database Systems*, 8th edition, Addison-Wesley, 2003.
[3] R.F. van der Lans, *The SQL Guide to SQLite*, Lulu Publishers, 2009.

3 What is a SQL Database Server?

The question "What is a SQL Database Server?" is an easy one to answer:

> *A SQL database server is a database server that supports the SQL database language.*

SQL is an incredibly popular database language. It supports a small, but very powerful, set of statements for manipulating, managing, and protecting data stored in a database. This power has resulted in its tremendous popularity. In the early 1980s there were only ten to twenty SQL database servers, but today this number is at least multiplied by four. Almost every database server supports SQL or a dialect of the language. Currently, SQL products are available for every kind of computer, from small handheld computers to large servers, and for every operating system. International standards for SQL have existed since 1987.

For a few years, a new generation of database servers has appeared on the market, the so-called *NoSQL database servers*. As the name suggests, these products do not support SQL or do not support SQL as their primary database language. In any case, they are all not relational database servers. These products do have an incredible scalability level and can deliver unbelievable performance results, even when the size of the database is massive or the transaction rate very high. Lately, vendors of NoSQL products have started to develop (simple) SQL interfaces. They have discovered that for many tools, applications, and users, it's a big advantage when data can also be accessed through a SQL interface. So, *SQL-fication* of NoSQL has become popular.

The SQL-fication of NoSQL products has started.

4 Is Actian PSQL a SQL Database Server?

Is PSQL a Database Server? – Actian PSQL supports all the database services listed in Section 2. It can therefore be regarded as a full-blown database server.

PSQL can be accessed through interfaces such as ADO.NET, JDBC, and ODBC. PSQL is developed on top of the MKDE (Micro Kernel Database Engine). Besides PSQL, the MKDE also supports an older, record-oriented interface called *Btrieve*. The effect is that the data managed by the MKDE can be manipulated and queried via the set-oriented SQL interface (offered by PSQL) and the record-oriented Btrieve API.

The fact that PSQL has been implemented on the MKDE doesn't make it a non-database server. There are many more database servers that have a layered architecture in which each layer has a well-defined interface.

Is PSQL a SQL Database Server? – PSQL supports most of the SQL statements one expects from a SQL database server. It allows data to be manipulated, queried, managed, and protected through the SQL language.

Evidently, PSQL supports its own SQL dialect, like all the SQL products do. All these dialects are somewhat different. But the important aspect is that PSQL supports the core of SQL: those statements and features supported by all the other SQL products

PSQL supports all the SQL features that developers expect.

and that all developers expect. A few PSQL specifics have been added, such as decoupling tables from their files. But this is a customary approach, every SQL product supports proprietary extensions.

Is it possible to measure how well SQL is being supported? This can be done using the international SQL standard. Different versions of that standard have been released through the years. PQSL claims to support the SQL-92 standard, plus some features of more recent SQL standards.

5 SQL and the Relational Model

What is the Relational Model? – Singlehandedly, E.F. Codd has been responsible for the initial development of the relational model. In his ground-breaking articles "Derivability, Redundancy and Consistency of Relations Stored in Large Data Banks" and "A Relational Model of Data for Large Shared Data Banks", published respectively in 1969[4] and 1970[5], he introduced the rest of the world to his ideas of how databases should organize data. Later on, many others contributed to the further development of the relational model and related topics, including C.J. Date, R. Fagin, W. Kent, and M. Stonebraker.

In a nutshell, the relational model is a set of principles that describes how data should be organized, and how data should be presented to and manipulated by applications. The relational model has never dealt with how data is physically stored. In fact, whether data is stored on modern-day SSD's in a column-oriented fashion, or whether it's stored somehow in a fish bowl full of salt water, as long as the data is perceived by the applications in a relational style, then it's ok. This means that the relational model doesn't deal with indexes, table spaces, log files, storage formats, memory structures, backup procedures, query optimization, and so on. It's purely focused on the *what* and not on the *how*.

The relational model is focused on the what *and not on the* how.

Different Relational Models – There is not one relational model. Different researchers have defined different relational models. And they don't always agree on what the real relational model constitutes. Some have even defined different versions. Codd himself was very explicit in this. In late 1979, he introduced RM/T[6] (the T stands for Tasmania), which was an extended version of his original relational model. Later, in 1990, he introduced RM/V2[7]. In that same book he wrote "it seems prudent to develop a sequence of versions V1, V2, V3, … ." In other words, he was in favor of having multiple versions.

Due to these different versions, it can be confusing if someone indicates that a particular database server is not according to the relational model. What exactly does that mean? In this whitepaper, if we refer to the relational model, Codd's RM/V1 is meant, because when he specified the rules, that was the current version.

---

[4] E.F. Codd, *Derivability, Redundancy and Consistency of Relations Stored in Large Data Banks*, IBM Research Report RJ-599, August 1969.
[5] E.F. Codd, *A Relational Model of Data for Large Shared Data Banks*, Communications of the ACM, Volume 13, Number 6, June 1970.
[6] E.F. Codd, *Extending the Database Relational Model to Capture More Meaning*, ACM Transactions on Database Systems, Volume 4, Number 4, December 1979.
[7] E.F. Codd, *The Relational Model for Database Management,* Version 2, Addison-Wesley, 1990.

Is SQL Really Relational? – SQL has always been regarded as one of the first relational database languages. Other early ones were Quel, QBE, and Square. More recently, the language D4 was introduced in the product DataPhor.

SQL does a decent job in being an implementation of the relational model. However, it's not a perfect implementation. For example, how primary keys are defined and how duplicate rows are treated, is different from the relational model. Therefore, through the years, many discussions have been held and articles have been written to describe the digressions of SQL.

These digressions do not only apply to commercial SQL implementations, but also to the SQL standards from the first to the most recent version. This means that it's impossible for a vendor to implement a SQL dialect that supports the relational model as well as the SQL standard. Therefore, vendors have to choose, and most went for the practical alternative: follow the SQL standard.

Conclusion: PSQL is not more relational than SQL itself, because PSQL supports an extensive SQL dialect and when possible, it supports the SQL standard.

The Differences Between SQL and Relational – It would go too far to list all the aspects where SQL differs from the relational model. For an exhaustive list, we refer to C.J. Date's book[8] "SQL and Relational Theory".

An important difference between SQL and the relational model is that the former allows for duplicate rows, while the relational model doesn't.

SQL has never been 100% in line with the relational model. There have always been differences. An important difference is that SQL allows for duplicate rows in tables, which is considered a sin in the relational model. Because a relation is a set of rows and in set theory a set cannot contain duplicate elements, therefore, relations (the equivalent of tables in SQL) can't contain duplicate rows[7]. That difference is probably the reason why SQL uses the term table and not relation.

Donald D. Chamberlin, one of the main designers of SQL and one of the first implementers of a SQL product, was for a large part responsible for the language as we know it right now. In his book on DB2[9], he describes why the decision was made to allow duplicate rows in tables. The designers of SQL[10], at that time, adopted a very practical approach. They decided not to enforce no-duplication, because, if no query result is allowed to contain duplicate rows, a sort operation has to be performed every time. Because in the 1970s and 1980s sorts were still resource hungry operations, they decided to make it optional. If you don't want duplicate rows in your tables, define a primary key, and if you don't want duplicate rows in your query results, add `distinct` to every query. Chamberlin: "When the original SQL designers decided to allow users the options of handling nulls and [duplicate rows], they viewed these features as minor conveniences, not as major departures from orthodoxy, taken at the risk of excommunication."

Another difference is that in the relational model data can only be manipulated and queried using a set-oriented approach, not a record-oriented approach. Codd worded it as follows: "[Relations] have no positional concepts. One may shuffle the rows without affecting information

[8] C.J. Date, *SQL and Relational Theory; How to Write Accurate SQL Code*, http://it-ebooks.info/book/183/, IT eBooks, 2009.
[9] D.D. Chamberlin et al., *SEQUEL 2: A Unified Approach to Data Definition, Manipulation, and Control*, IBM R&D, November 1976.
[10] D.D. Chamberlin, *A Complete Guide to DB2 Universal Database*, Morgan Kaufmann Publishers, 1998.

content. Thus, there is no "nextness" of rows." This implies that if a database language supports a feature that allows an application to browse through the rows of a table by just indicating next, that feature is not relational. In addition to the set-oriented approach, most SQL products do support a record-oriented approach, one that is based on *cursors*. In other words, the cursor, that is used heavily when programming so-called embedded SQL, is not a relational concept.

6 Codd's Twelve Rules and Actian PSQL

Now that we have described what the relational model is, we can focus on the term *relational database server*, and determine whether PSQL is a relational database server.

> *A relational database server is a database server that allows the data to be manipulated according to the rules of the relational model.*

In the second half of the 1980s, the term relational was becoming commercially valuable. To benefit from this momentum, more and more products were classified as relational by their vendors, which not all of them were. This triggered Codd to define a set of rules that could be used to "measure" how relational a product really is. These rules became known under the name *Codd's Twelve Rules*. These rules were published in several articles[11,12,13] and can also be found in Wikipedia[14].

The twelve rules were in fact thirteen rules: rules 0 to 12. Together, they can be used as a yardstick. Through the years, many products have been tested on their relationality. A product can score 1 point, half a point, or zero points for a rule depending on how well the rule is supported. Note that products do not get a point when they support rule 0. Codd did some of these tests himself. For example, in 1985 he published an article in which he concluded that IBM's DB2 scored a 7, and Cullinet's IDMS/R and ADR's Datacom/DB a 0.

6.0 Rule 0: Single Foundation Rule

Rule: For any system that is advertised as, or claimed to be a relational DBMS, that system must be able to manage database entirely through its relational capabilities.

This rule indicates that a relational database server is allowed to support multiple database languages and interfaces, even non-relational ones, but there has to be at least one relational language or interface for invoking all the services supported by that database server.

This rule is essential. Codd indicated that this rule must hold whether or not the product supports any non-relational capabilities for managing data. If a product does not support this Rule 0, it's not worth rating it. That's why products don't get points for supporting this rule.

Conclusion: In Actian PSQL all supported database services can be invoked using SQL. There are no services for which a non-relational interface is needed. PSQL supports this rule and can be rated.

---

[11] E.F. Codd, *Is Your DBMS Really Relational*, Computerworld, October 14, 1985.
[12] E.F. Codd, *Does Your DBMS Run By The Rules?*, Computerworld, October 21, 1985.
[13] E.F. Codd, *The Relational Model for Database Management, Version 2*, Addison-Wesley, 1990.
[14] Wikipedia, see http://en.wikipedia.org/wiki/Codd%27s_12_rules, 2013.

6.1 Rule 1: Information Rule

Rule: All information in a relational database (including table and column names) is represented at the logical level in only one way: by values in tables.

In most SQL systems all values are stored in tables and can't be stored outside tables. In addition, information can only be presented to applications as values that are stored in the columns of tables. All this applies to PSQL as well.

Additionally, Codd indicated that this rule should be true not only for data values but for table names and column names as well. In other words, meta data should also be stored as values in tables. In SQL systems these tables are called catalog or system tables. In PSQL all data is stored in tables and is presented as values in tables. PSQL's meta data is also stored and presented in (catalog) tables.

Conclusion: PSQL supports Rule 1.

6.2 Rule 2: Guaranteed Access Rule

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.

In relational systems data can be found and identified using set-oriented queries. With such a query, an application retrieves a set of rows. What this rule indicates is that each value in the database should be accessible with a query that specifies the table, the column name to retrieve and a condition that identifies (at maximum) one row in the table. In fact, it must always be possible to identify each value through a table name, column name, and a primary key value.

This implies that for each table in a relational system a primary key has to be defined. If for a table a primary key is missing, it's not possible to identify each value; see also Section 5.

PSQL supports primary keys. However, as with almost all SQL systems, PSQL does not require that a primary key is defined for a table, it's optional. Tables can be created without primary keys. If there are no other concepts defined that demand the uniqueness of rows in a table, such a primary key-less table can have duplicate rows. The effect is that values in such a table can't be uniquely accessed by the combination of a table name, column name, and primary key value.

Conclusion: PSQL does *not* support Rule 2, because primary keys are optional for tables.

6.3 Rule 3: Systematic Treatment of Null Values

Rule: Null values (distinct from the empty character string of blank characters and distinct from zero or any other number) are supported in fully relational DBMS for representing missing information and inapplicable information in a systematic way (independent of data type).

What Codd meant with this rule is that it should *not* be necessary to define a special value (for

each data type or column a special one) to represent missing information. Defining that a column allows null values, should be enough, regardless of the column's data type. Codd also indicated that it's important that if a column is part of a primary key, it *must* be a null-not-allowed column.

PSQL supports null values as described in the SQL standard. The systematic way of treating null values in the SQL standard is according to how Codd had defined the concept. In addition, in PSQL, if a column becomes part of a primary key, it's automatically set to null-not-allowed.

Conclusion: PSQL supports Rule 3.

Note: About five years after publishing the twelve rules, Codd wrote an article[15] in which he specified that relational systems should support two types of null values: null applicable and null non-applicable. However, this was done after he had defined the rules, so they are not described in them. No SQL system supports multiple null values.

6.4 Rule 4: Dynamic On-Line Catalog Based on the Relational Model

Rule: The database description is represented at the logical level just like ordinary data, so that authorized users can apply the same relational language to its interrogation as they apply to the regular data.

This rule means that the system must support a set of catalog tables and that these catalog tables can be accessed using the same query language used for querying the "ordinary" tables. If this rule is true, users and developers benefit by having to learn only one language.

In PSQL all data is stored in catalog tables, and all the meta data is accessible through these catalog tables. Examples of such catalog tables are X$VIEW and X$FIELD. All these catalog tables can be queried in the same way as ordinary tables (using SQL). Even views can be defined on them[16].

Conclusion: PSQL supports Rule 4.

6.5 Rule 5: Comprehensive Data Sublanguage Rule

Rule: A relational system may support several languages and various modes of terminal. 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 (create table, drop table, and alter table)*
- *View definition (create view and drop view)*
- *Data manipulation (select, insert, update, and delete)*

[15] E.F. Codd, *The Relational Model for Database Management, Version 2*, Addison-Wesley, 1990.
[16] R.F. van der Lans, *Accessing the DDF Files Through Views*, cs.pervasive.com/blogs/sql4psql/archive/2010/03/22/accessing-the-ddf-files-through-views.aspx, March 2010.

- *Integrity constraints (create constraint and drop constraint)*
- *Authorization (grant and revoke)*
- *Transaction boundaries (start transaction, commit, rollback)*

To illustrate what is meant with the various items, examples of SQL statements are added between brackets.

This rule overlaps with Rule 0 somewhat. This rule indicates that there should be at least one language for invoking all the services listed above and, in addition, that language should be expressible with character strings. In 1985, when the rules were introduced, other relational languages were being developed of which some could not be expressed in character strings. QBE (Query By Example) is such an example[17]. This language only had a simple graphical interface. There was no version in which the statements could be specified in strings.

PSQL supports all the features in the list above in the form of SQL statements, and its SQL statements are expressible using a well-defined syntax as character strings. Evidently, this last requirement applies to every SQL implementation.

Conclusion: PSQL supports Rule 5.

6.6 Rule 6: View Updating Rule

Rule: All views that are theoretically updatable must be updatable by the system.

A view is a virtual table of which the content is derived from underlying base tables the moment they are queried. Besides being queryable, views can also be updatable. When views are updatable it means that they can accept inserts, updates, and deletes. When that happens, those changes are applied to the underlying base tables.

Whether a view is updatable depends on how it has been defined. For example, for a view that shows aggregated data, it's difficult to indicate how inserts on such a view should be reflected as inserts in the underlying base tables. Views that join data from multiple base tables, are not always updatable either, although some are. Much has been written on what the rules are of determining which views are theoretically updatable[18,19].

In PSQL the rule is that the (virtual) contents of a view can be changed only if a one-to-one correspondence exists between the rows of the view and the rows of the underlying base table(s). This means, for example, that if a view definition contains `distinct` in the `select` clause or contains a `group by` clause, it can't be updated. Now this restriction is acceptable, because those views are theoretically not updatable. However, PSQL also has the restriction that if a view is defined as a join, it's not updatable. Unfortunately, theoretically, some joins are updatable.

Conclusion: PSQL supports Rule 6 partially.

[17] M.M. Zloof, *Query By Example*, Proceedings NCC 44. Anaheim, California, May 1975.
[18] C.J. Date, *Relational Database Writings 1991-1994*, Addison-Wesley Publishing Company, 1995.
[19] C.J. Date, *View Updating and Relational Theory; Solving the View Update Problem*, see http://it-ebooks.info/book/1357/, IT eBooks, 2012.

6.7 Rule 7: High-Level Insert, Update, and Delete

Rule: 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.

In every relational system set-oriented queries can be specified. The result of such a query is a set of rows. This rule indicates that insert, update, and delete operations should also be set-oriented. For example, it should be possible to delete a set of rows in a table with one delete operation, and it must be possible to change values in a set of rows with one update operation. There should be no need to use a record-oriented interface for inserts, updates, and deletes.

According to Codd, the advantage of a set-oriented approach for these operations is that it allows the relational system to determine what the most efficient processing strategy is for executing them. In other words, like with queries, the optimizer can try to come up with the most efficient processing strategy. Programmers don't have to concern themselves with that aspect. They only have to focus on what has to be inserted, updated, or deleted.

From the first version, the SQL standard has included set-oriented inserts, updates, and deletes. The same applies for PSQL's implementation of SQL.

Conclusion: PSQL supports Rule 7.

6.8 Rule 8: Physical Data Independence

Rule: Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representations or access methods.

As indicated, the relational model doesn't concern itself with how data is stored and how data should be accessed. This rule emphasizes this aspect. Applications should only "see" tables, columns, and data values. They should not have to deal with how or where data has been stored and which indexes can and should be used to retrieve the data. If this is the case, the advantage is that those physical aspects can be changed for performance or concurrency reasons without impacting the applications. This improves productivity and maintenance.

In PSQL, indexes can be added and dropped, storage structures can be changed, and all these changes have no impact on application code. In fact, PSQL is one of the few SQL systems that even allows the data files to be decoupled from the table. This makes it possible to link another data file to a table. All these actions have no impact on the applications.

Conclusion: PSQL supports Rule 8.

6.9 Rule 9: Logical Data Independence

Rule: Applications programs and terminal activities remain logically unimpaired when information-preserving changes of any kind that theoretically permit unimpairment are made to the base tables.

Due to changes of information needs, the structures of base tables may have to change. When

those structures change, application codes may have to change as well. This rule indicates that it should be possible that those applications, for which the change is irrelevant, do not have to be changed. This can be done by simulating the old structures of the base tables with views. In other words, after the change the applications access the data through views that have the same structure and content as the old base tables.

Codd gives two examples: "(1) splitting a table into two tables either by rows using row content or by columns using column names, if primary keys are preserved in each result; (2) combining two tables into one by means of a non-loss join". This is called a *lossless join*.

This rule relates closely to Rule 6 for view updatability. Example 1 is difficult to support for PSQL, because it means that when a base table is split into two tables, a view has to be defined that reconstructs that table to hide that fact. Unfortunately, this view is non-updatable, because PSQL doesn't support updating of views when they are joins, whereas the original table was updatable.

In general, it's difficult for a system to support the logical data independence rule when it doesn't fully support Rule 6.

Conclusion: PSQL supports Rule 9 partially.

6.10 Rule 10: Integrity Independence

Rule: Integrity constraints must be specified separately from application programs and stored in the catalog. It must be possible to change such constraints as and when appropriate without unnecessarily affecting existing applications.

This rule consists of two parts. The first part of the rule indicates that a relational system should support entity integrity, referential integrity, and it should also make it possible to specify more specific integrity rules that reflect, for example, business policies or government regulations. It must be possible to define these additional integrity rules in the same language in which the database is queried and manipulated.

The second part of the rule specifies that integrity rules should not be part of the applications, but should be managed by the relational system. The advantage of this approach is that if integrity rules change, application code doesn't have to be changed.

PSQL supports entity integrity and referential integrity. Additional integrity rules can be specified as well. Especially the triggers allow for many different types of integrity rules to be specified and they are independent of the applications.

Conclusion: PSQL supports Rule 10.

6.11 Rule 11: Distribution Independence

Rule: A relational DBMS has distribution independence.

Many database servers allow for the data to be distributed over a network of servers. This rule dictates that when portions of the database are distributed over various locations, the distribution should be invisible to the applications and users of the database. To them, a distributed database should still look like one logical database. This also means that existing applications should continue to operate successfully when data is distributed and redistributed.

It's interesting to see that some of the new database servers developed for big data environments do allow processing to be distributed over large numbers of processors with an almost linear scalability. The distribution technique used here is usually called *sharding*. Unfortunately, for some of these products the applications have to know where the data is stored. So, some index or hashing structure is needed that guides the applications to the right server (shard). This can lead to a situation where, if data has to be redistributed, application logic has to be changed. Such products do not offer distribution independence.

In PSQL data can be distributed and all the distribution is hidden for the applications.

Conclusion: PSQL supports Rule 11.

6.12 Rule 12: Non-Subversion Rule

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).

If a system provides a low-level (record-at-a-time) interface, then that interface cannot be used to subvert the system for, for example, bypassing data security or data integrity rules.

In PSQL, applications have a choice, they can access the data using PSQL's set-oriented SQL interface or the record-oriented Btrieve interface; see Chapter 4. PSQL provides ways to configure databases in such a way that no other access methods such as Btrieve can, for example, subvert integrity rules set at the SQL level. For example, if triggers have been defined using SQL, they are still invoked when the Btrieve interface is used. Note that PSQL provides the flexibility for a database administrator to configure databases in such a way that lower-level access methods such as Btrieve can access data without enforcing the SQL integrity rules.

The same applies for data security. If specific security rules have been defined using SQL on a database that has been configured as "bound" (bound means all access methods are bound by one security model), then no other access methods such as Btrieve can access the data without supplying the security token set at the SQL level. Again, subversion is not possible using a low-level language.

Conclusion: PSQL supports Rule 12.

Note: Many of the new NoSQL products make it possible to subvert. If we take the Hadoop stack for example, the module called Hive runs on top of another module called MapReduce, which runs on top of the HDFS (Hadoop Distributed File System). Applications can choose which interface they want to use to access the data stored in HDFS. They can go straight to HDFS, they can use MapReduce, or they can opt for the easy-to-use SQL interface of Hive. However, the consequence is that if, for example, functionality is added in the Hive layer, all that functionality is bypassed when an application decides to use the MapReduce or HDFS interfaces. The good thing about the Hadoop stack is that the interfaces of all these layers have been very well-defined.

7 How Relational is Actian PSQL?

The final question is how relational PSQL is? Table 1 indicates how well PSQL scores on Codd's twelve rules. The result is a high score of 10 (out of 12), or in other words, PSQL is for 83% relationally complete.

	Score
1. Information rule	Yes
2. Guaranteed access rule	No
3. Systematic treatment of null values	Yes
4. Active catalog based on RM	Yes
5. Comprehensive data sublanguage	Yes
6. View updating rule	Partial
7. High-level insert, update, delete	Yes
8. Physical data independence	Yes
9. Logical data independence	Partial
10. Integrity independence	Yes
11. Distribution independence	Yes
12. Non-subversion rule	Yes
Total	10

Table 1 *How does PSQL score on Codd's twelve rules?*

How good is a score of 10? If we compare this score to other SQL products, then we have to conclude that there are not many SQL products with a higher score. In fact, many new SQL products have a lower score than PSQL. For example, some new ones do not even support high-level insert, update, and delete, do not support integrity rules, and do not have a catalog. Pure relational systems may score higher, but they barely exist.

It's important to note that at the time when Codd[1] wrote these rules, he wrote that "No existing DBMS product that I know of can honestly be claimed to be fully relational, at this time."

About the Author Rick F. van der Lans

Rick F. van der Lans is an independent analyst, consultant, author, and lecturer specializing in data warehousing, business intelligence, database technology, and data virtualization. He works for R20/Consultancy (www.r20.nl), a consultancy company he founded in 1987.

Rick is chairman of the annual European Enterprise Data and Business Intelligence Conference (organized in London). He writes for the eminent B-eye-Network.com[20] and other websites. He introduced the business intelligence architecture called the *Data Delivery Platform* in 2009 in a number of articles[21] all published at BeyeNetwork.com. For seven years he was a member of the standardization committee for the SQL standard.

He has written several books on SQL. His popular *Introduction to SQL*[22] was the first English book on the market in 1987 devoted entirely to SQL. After more than twenty years, this book is still being sold, and has been translated in several languages, including Chinese, German, and Italian. Quite recently, he published a book on Actian PSQL[23]. His latest book[24] "Data Virtualization for Business Intelligence Systems" was published in 2012.

He has worked together with consultancy organization Codd and Date, Inc., which was founded by the inventors of the relational model: E.F. Codd and C.J. Date.

For more information please visit www.r20.nl, or email to rick@r20.nl. You can also get in touch with him via LinkedIn (http://www.linkedin.com/pub/rick-van-der-lans/9/207/223) and via Twitter @Rick_vanderlans.

About Actian Corporation

Actian transforms big data into business value for any organization – not just the privileged few. Our next generation Actian Analytics Platform™ software delivers extreme performance, scalability, and agility on off-the-shelf hardware, overcoming key technical and economic barriers to broad adoption of big data, delivering Big Data for the Rest of Us™.

Stay connected with Actian Corporation at www.actian.com or on Facebook, Twitter and LinkedIn.

[20] See http://www.b-eye-network.com/channels/5087/articles/
[21] See http://www.b-eye-network.com/channels/5087/view/12495
[22] R.F. van der Lans, *Introduction to SQL; Mastering the Relational Database Language*, fourth edition, Addison-Wesley, 2007.
[23] R.F. van der Lans, *The SQL Guide to Pervasive PSQL*, Lulu Publishers, 2009.
[24] R.F. van der Lans, *Data Virtualization for Business Intelligence Systems*, Morgan Kaufmann Publishers, 2012.