Row Level Security in PostgreSQL

D2.1 v1.0

WP2 – Security, Privacy and Audit D2.1
Row Level Security in PostgreSQL

Dissemination Level: Confidential

Lead Editor: Simon Riggs

Date: 31 Oct 2014

Status: Final version

Description from Description of Work:

D2.1) Row Level Security in PostgreSQL: The open source database PostgreSQL will be extended with row-level security that is controlled by SELinux. Implementation will consider the needs of military high security, commercial high security requirements (e.g. medical data) as well as the needs of multi-tenancy environments for cloud database hosting.

Contributors:

Simon Riggs, 2ndQ
Craig Ringer, 2ndQ

Internal Reviewer(s):

Yeb Havinga, PV
Javier Navaridas, UNIMAN

Version History

Version	Date	Authors	Sections Affected
0.1	29/10/14	Simon Riggs	Initial version
0.2	31/10/14	Simon Riggs	Revisions after internal review
1.0	31/10/14	Jo Dix	Final version

Table of Contents

Abbreviations

RLS	Row Level Security

1. Summary

Row Level Security was committed to PostgreSQL code repository on 19 Sept 2014.

http://www.postgresql.org/message-id/E1XV0JP-0003h6-O6@gemulon.postgresql.org

2ndQuadrant's Craig Ringer received the primary credit for the work, which was committed to the repository by an independent reviewer, Stephen Frost, after peer review work and additions from other PostgreSQL community members.

The RLS feature will be available in the production release of PostgreSQL 9.5, due for release in Sept 2015.

RLS allows a Declarative Security Policy to be enforced against data in a table, while also allowing the query to be optimized fully and safely using the policy rules.

RLS is implemented wholly within the database and does not rely upon external modules or libraries to work correctly. Thus AXLE has delivered a mechanism that can work for all usage, for both analytical and data maintenance applications.

2. Analysis

The main use cases of Row Level Security are privacy and secrecy applications. These can range from the more obvious use by government and military databases, through to public databases. RLS also allows multi-tenancy applications more easily. AXLE has been required to contribute a generic implementation, even though our needs covered only a specific use case.

Medical data is a prime example of high privacy data that can be understood by all. Any database containing high volumes of medical data must be protected with good security to ensure legal conditions for managing and using such data are enforced. Fines are imposed on organizations and, in some countries, individuals who breach these conditions.

There are no exceptions to these rules just because databases are used for analytical and/or scientific research purposes; the "good of mankind" is not a defence in law. In fact, the larger the database, the more useful it is for potential abusers of information security and privacy, and also fines become larger.

The reason many analytical databases avoid these issues is simply that they are both hard to implement and costly in terms of analytical performance. We have faced up to those challenges by providing security with a good balance between usability and performance.

3. Implementation

The user visible mechanisms for RLS are described first, followed by key internal implementation details.

Row security can be enabled on individual database tables by issuing this command

```
ALTER TABLE patient
ENABLE ROW LEVEL SECURITY;
```

A table specific security policy can then be created for a table using this command

```
CREATE POLICY name ON table_name
  [ FOR { ALL | SELECT | INSERT | UPDATE | DELETE } ]
  [ TO { role_name | PUBLIC } [, ...] ]
  [ USING ( expression ) ]
  [ WITH CHECK ( check_expression ) ]
```

This allows additional filters/constraints to be made on the table.

USING (expression)

The expression will be used to filter rows visible to queries against the table.

WITH CHECK (check_expression)

The check_expression will be used to filter rows added to the table by commands.

These two clauses allow a POLICY to contain different filters for reads and writes.

The POLICY will be enforced either for all users, or for all users except the table owner and superuser, as specified by the row_security parameter set for the server.

The BYPASSRLS capability may be granted to individual users where necessary, such as specific users granted usage for backup, for example.

TRUNCATE commands are not covered by row security, though they already have their own existing privilege to protect usage.

Further details of these commands are available here

http://www.postgresql.org/docs/devel/static/sql-createpolicy.html

The USING and WITH CHECK clauses are enforced as indexable conditions, meaning they can be used against indexes. Technically they are implemented as sub-selects, allowing them to be optimized alongside other query terms. This allows security to be enforced without gross impact on performance, whether on OLTP or DSS.

The USING and WITH CHECK clauses are enforced using security barrier expressions so they are applied before all non-security related conditions. This ensures that no user defined functions can inspect unprivileged data, which is one of the potential “exploits” typically used against attempts to protect data using simple database views.

It must be noted that access to a database for analytical use provides much greater opportunity to circumvent security measures. Various explorative techniques exist to infer the existence of other data, typically described as “covert channels”. Whilst we cannot prevent such access, we can reduce its meaning; use of system assigned Primary Keys that have zero external information value is essential in secure databases.

Analytical users also have considerably more power to investigate data. Attempts to further secure systems have not yielded a useful balance between direct security and analytical usability. As a result, AXLE, in line with other industry groups and companies such as Oracle, need to point out that full security is only possible when coupled with other preventative and detective controls. That is why AXLE WP2 also includes T2.3 Auditing to provide the detective capability to monitor ad-hoc database analysis.

4. Conclusions and Future Work

D2.1 will be demonstrated at the project review, together with D2.4.

The Row Level Security feature is not yet certified by any major groups. Such testing is lengthy and expensive, requiring external spends that are outside of the current research project.

No further work is planned.