Security-Enhanced PostgreSQL

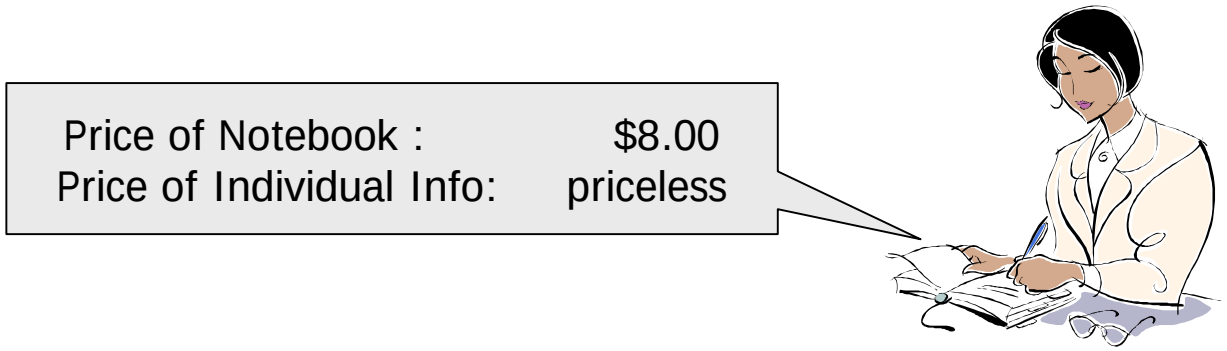
- System-wide consistency in Access Control -

NEC OSS Promotion Center
KaiGai Kohei <kaigai@ak.jp.nec.com>

Who is KaiGai ?

- Primary developer of SE-PostgreSQL
- 5 year's experience in Linux kernel development
  - Especially, SELinux and Security related.
- Experience in PostgreSQL
  - About 8 years as a user :-)
  - About 2 years for development of SE-PostgreSQL

Philosophical Background



- What do you really want to protect from harms?
  - Individual info, Corporate secrets, Authentication data,...
  - called as "Information Asset"
- Information Asset has to be stored in something.
  - Filesystem, Database, Paper, Brain, ...

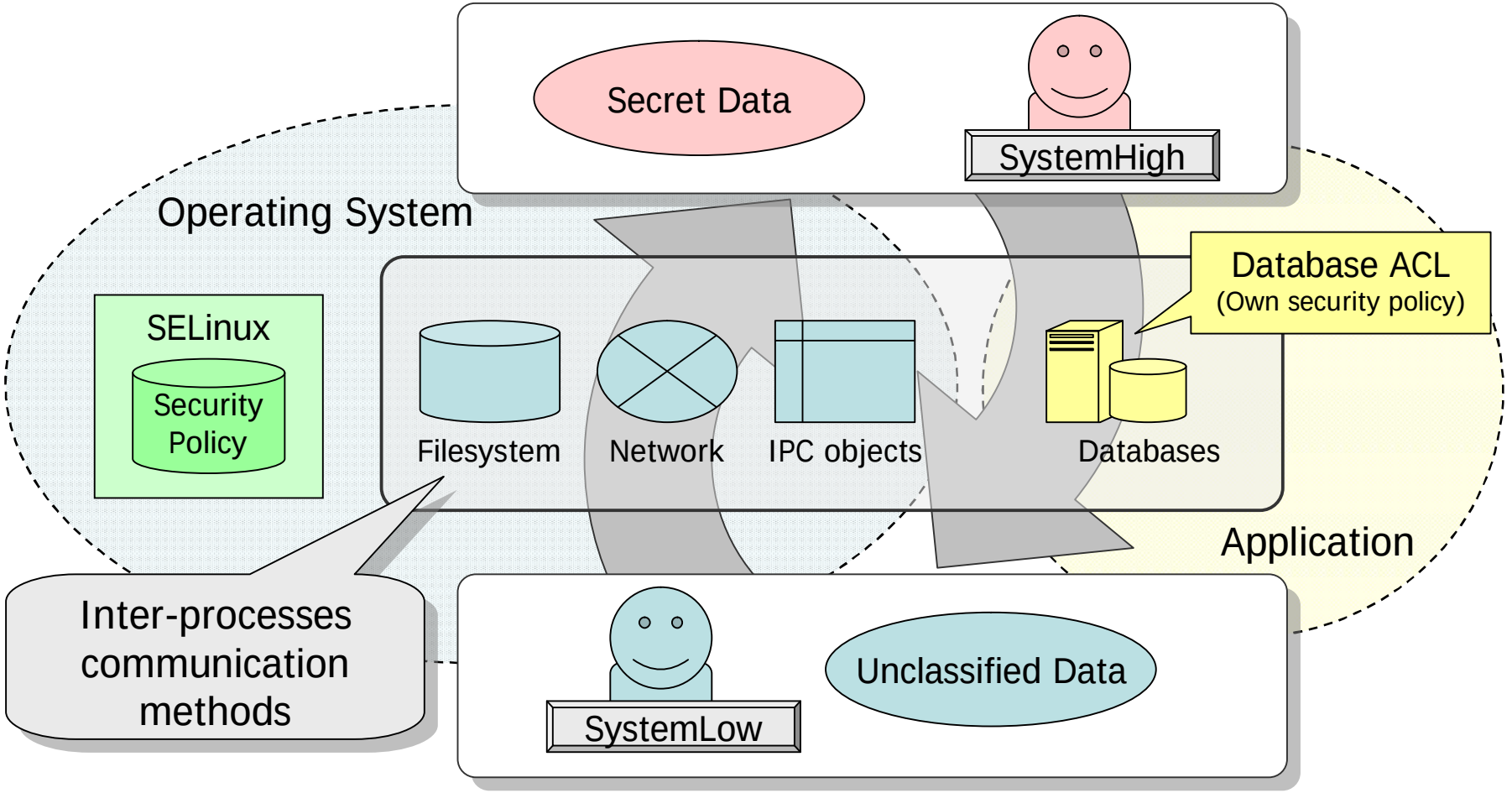
Philosophical Background

- What decides the worth of Information Asset?
  - Contents, not the way to store
- How access control mechanism works?
  - Filesystem: UNIX permission (rwxrwxrwx)
  - Database: Database ACL (GRANT/REVOKE)
  - ➡ Strongly depends on the way to store them!

We should apply consistent access control rules for same information assets, independent from the way to store them!

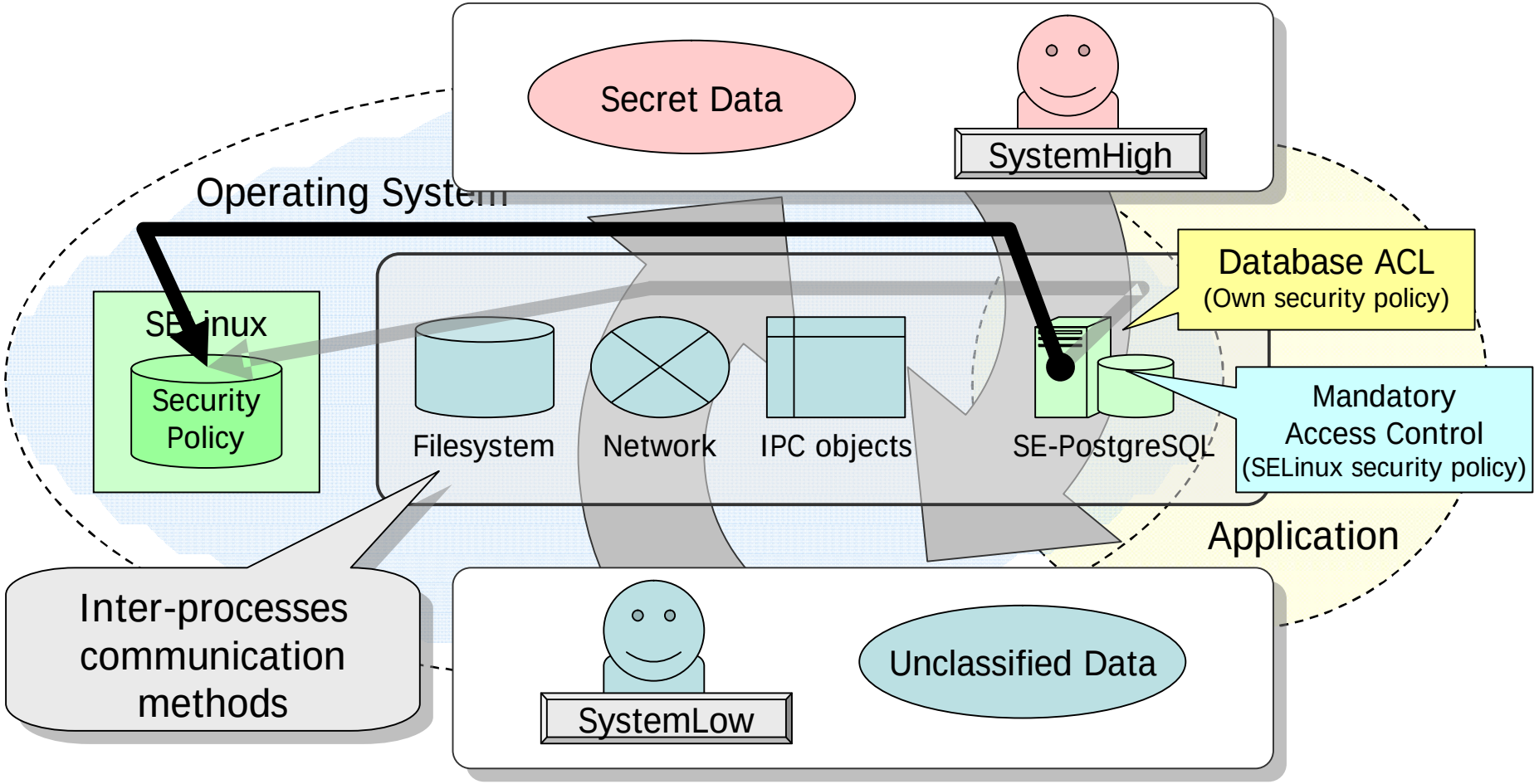
Consistency in access control policy

- Access control policy depending on the way to store Information Asset



Consistency in access control policy

- A single consistent security policy on whole of the system
- Any query, Any object without Any exception



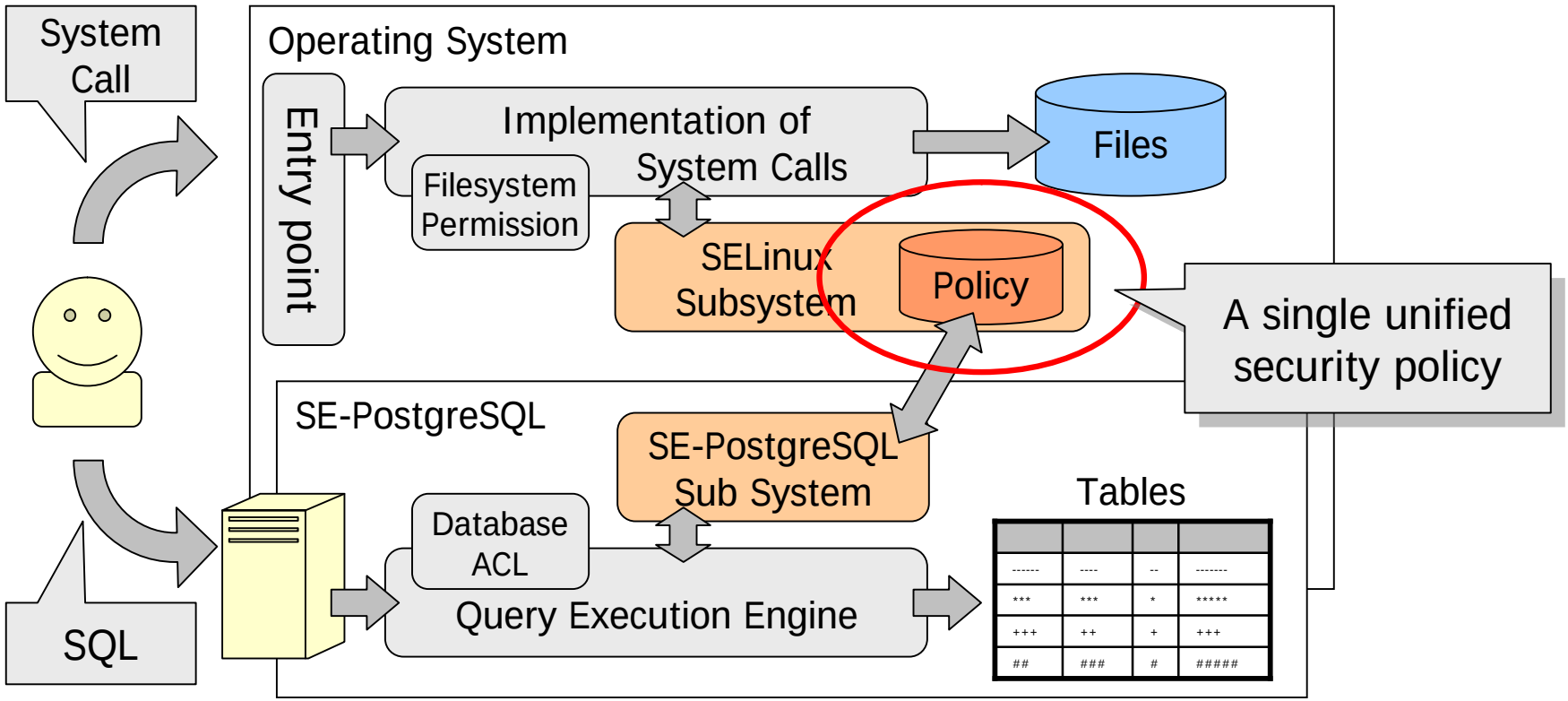
The Feature of SE-PostgreSQL

- "System-wide" consistency in access controls
  - A single unified security policy both OS/DBMS
  - Common security attribute representation
- Fine-grained Mandatory Access Controls
  - Tuple/Column-level access controls
  - Non-bypassable, even if privileged users
- The GOAL of SE-PostgreSQL?
  - Provision of System-wide Data Flow Controls
  - Prevention to leak/manipulate by malicious insider
  - Minimization of damages from SQL injection

"System-wide" consistency in access controls

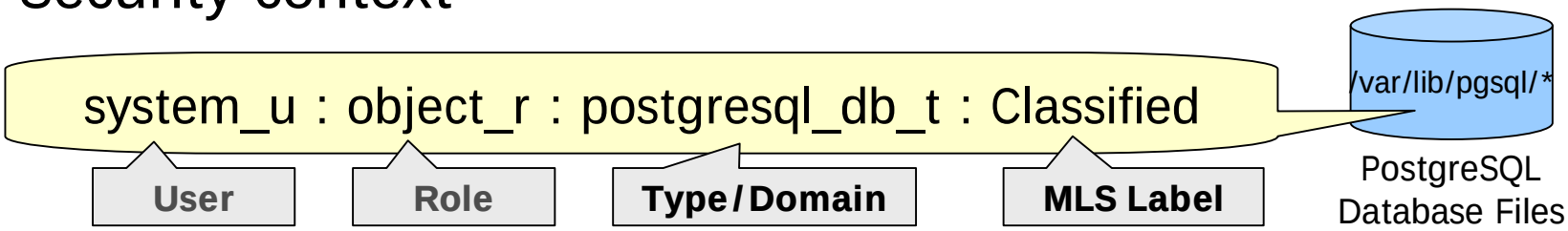
SE-PostgreSQL System Image

- A single unified security policy is applied,
  - when user tries to read a file via system-calls
  - when user tries to select a table via SQL-queries



How security policy works? (1/2)

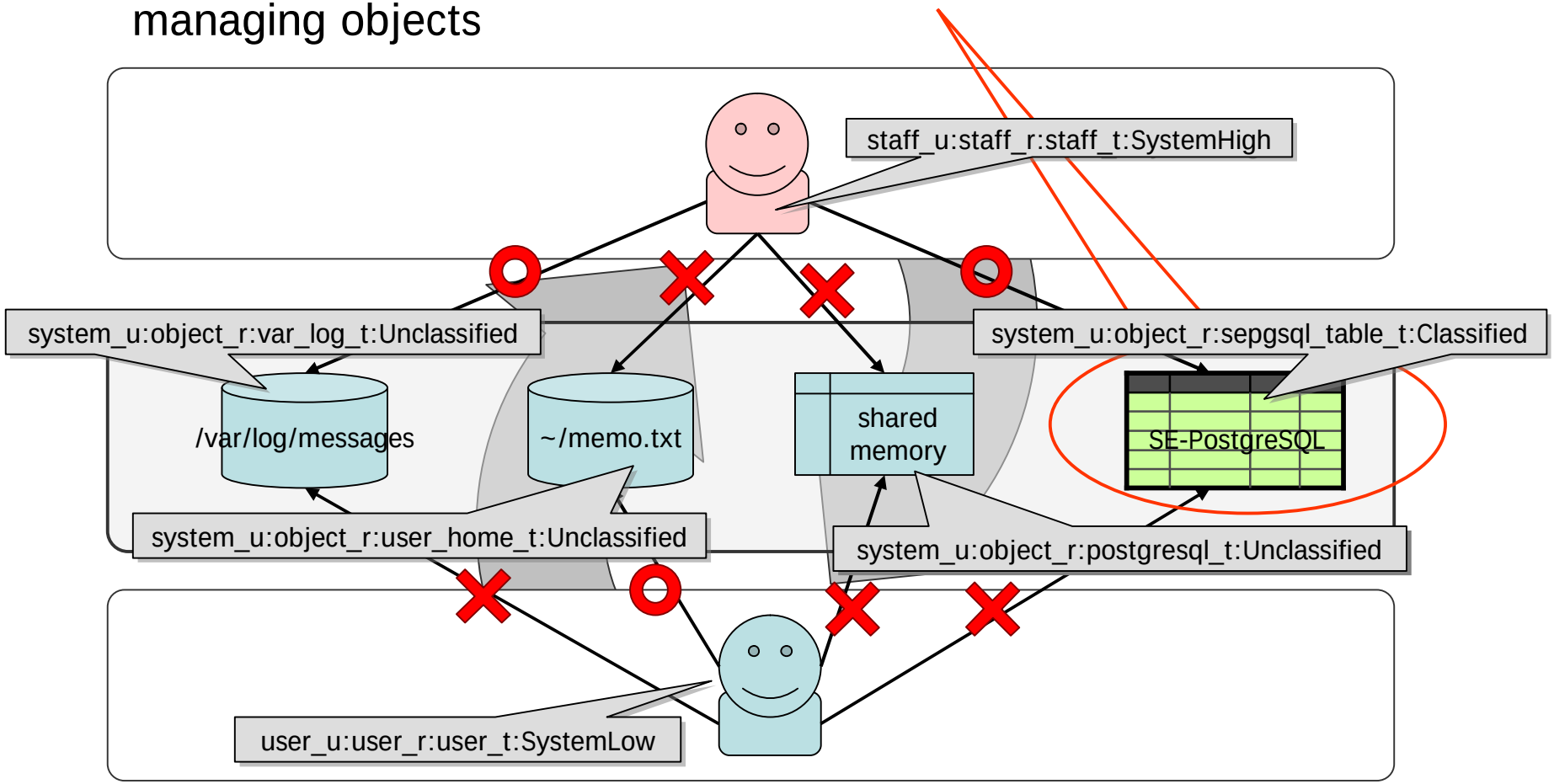
- SELinux makes a decision with security policy and context.
- Security context



  - Any process/resource have its security context.
  - It enables to show its attribute independent from its class.
- Security policy
  - A set of massive rules to be allowed
  - Rules are described as relationships between two security contexts and action.
    - **postgresql_t** is allowed to write files with **postgresql_log_t**.
    - **SystemHigh** is allowed to read file with **Classified**.

How security policy works? (2/2)

- Common attributes well formalized for various kind of resources.
- Object manager has to maintain proper security context of its managing objects



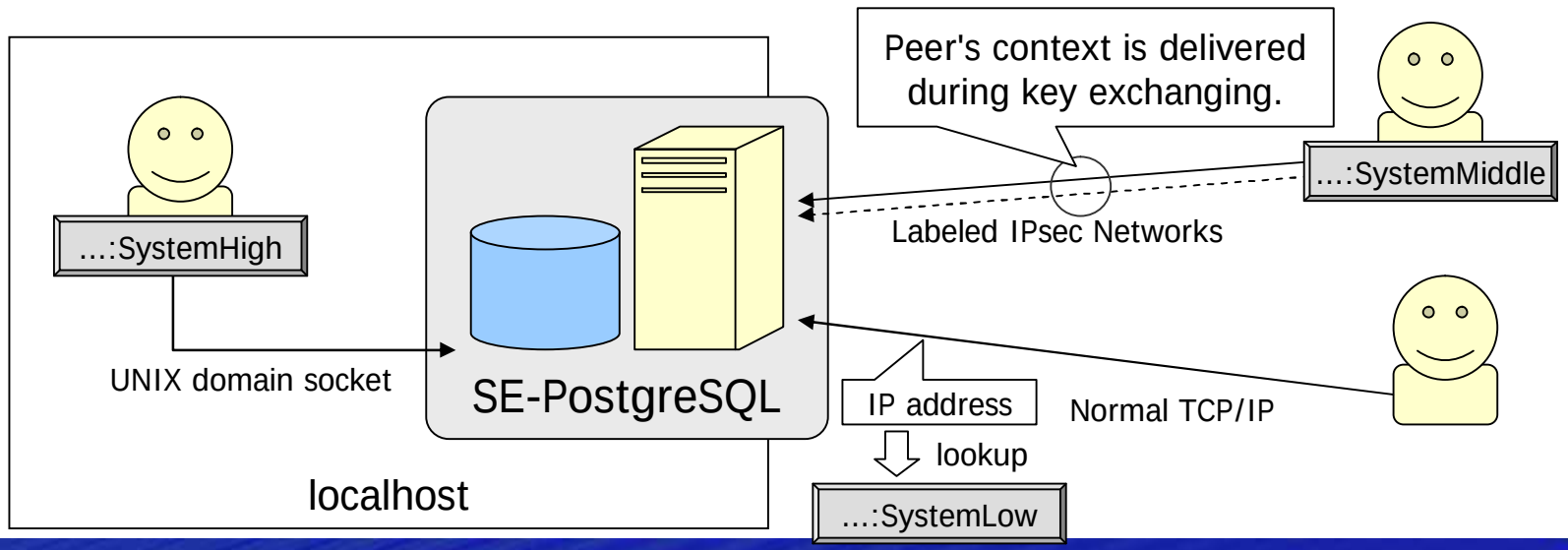
'security_context' system column

```
postgres=# SELECT security_context, * FROM drink;
             security_context              | id | name  | price | alcohol
-------------------------------------------+----+-------+-------+---------
 unconfined_u:object_r:sepgsql_table_t:s0 |  1 | water |   100 | f
 unconfined_u:object_r:sepgsql_table_t:s0 |  2 | coke  |   120 | f
 unconfined_u:object_r:sepgsql_table_t:s0 |  3 | juice |   130 | f
 system_u:object_r:sepgsql_table_t:s0:c0  |  4 | cofee |   180 | f
 system_u:object_r:sepgsql_table_t:s0:c0  |  5 | beer  |   240 | t
 system_u:object_r:sepgsql_table_t:s0:c0  |  6 | sake  |   320 | t
(6 rows)
```

- A new system column of **security_context**.
- It shows security context of each tuples.
  - ☑ In **pg_attribute**, it shows security context of the column.
  - ☑ ditto, for pg_class, pg_database, pg_class
- Default security context of newly inserted tuples
- Updating security context via writable system column

How clients' authority decided?

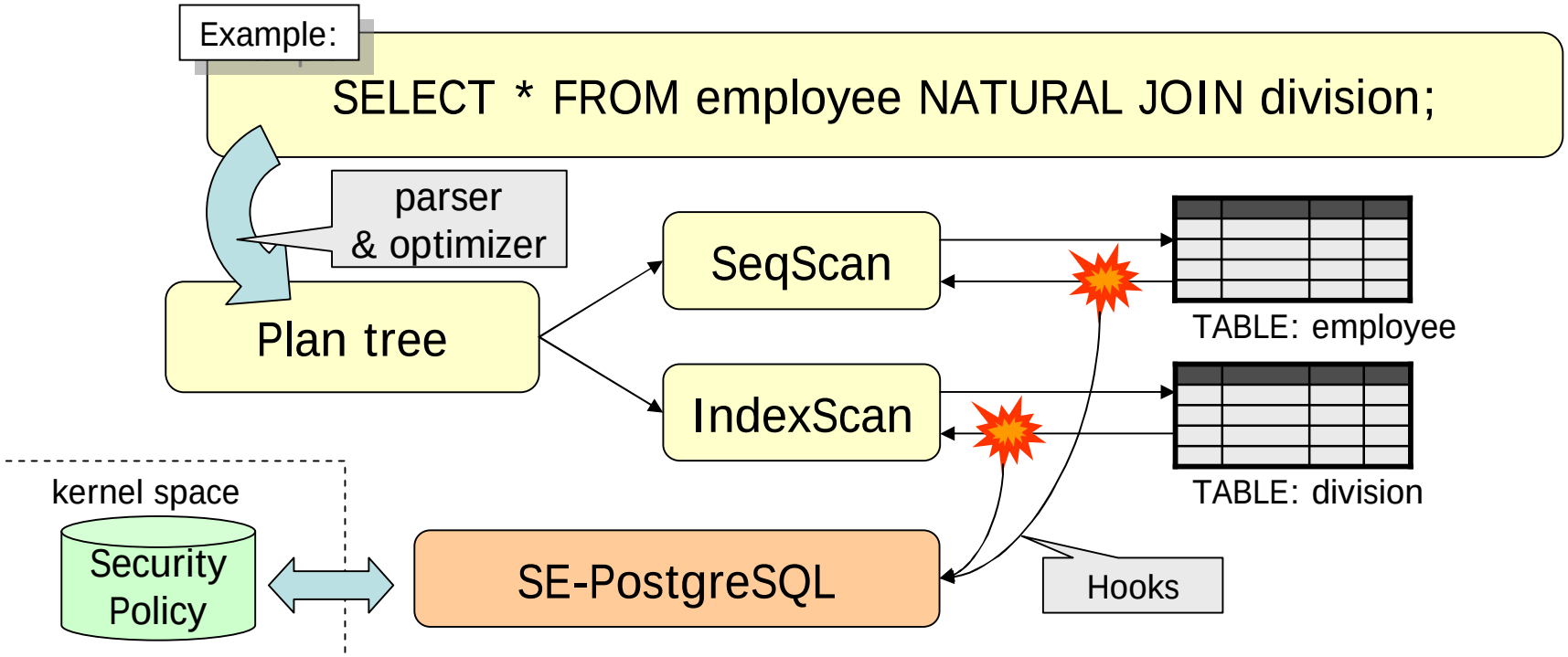
- Access controls, as if users access files via system calls.
  - But, queries come through networks.
- Labeled Networking Technology
  - SELinux provides **getpeercon()** API, that enables to obtain the security context of peer process.
  - SE-PostgreSQL applies it as a security context of client



Fine-grained Mandatory access controls

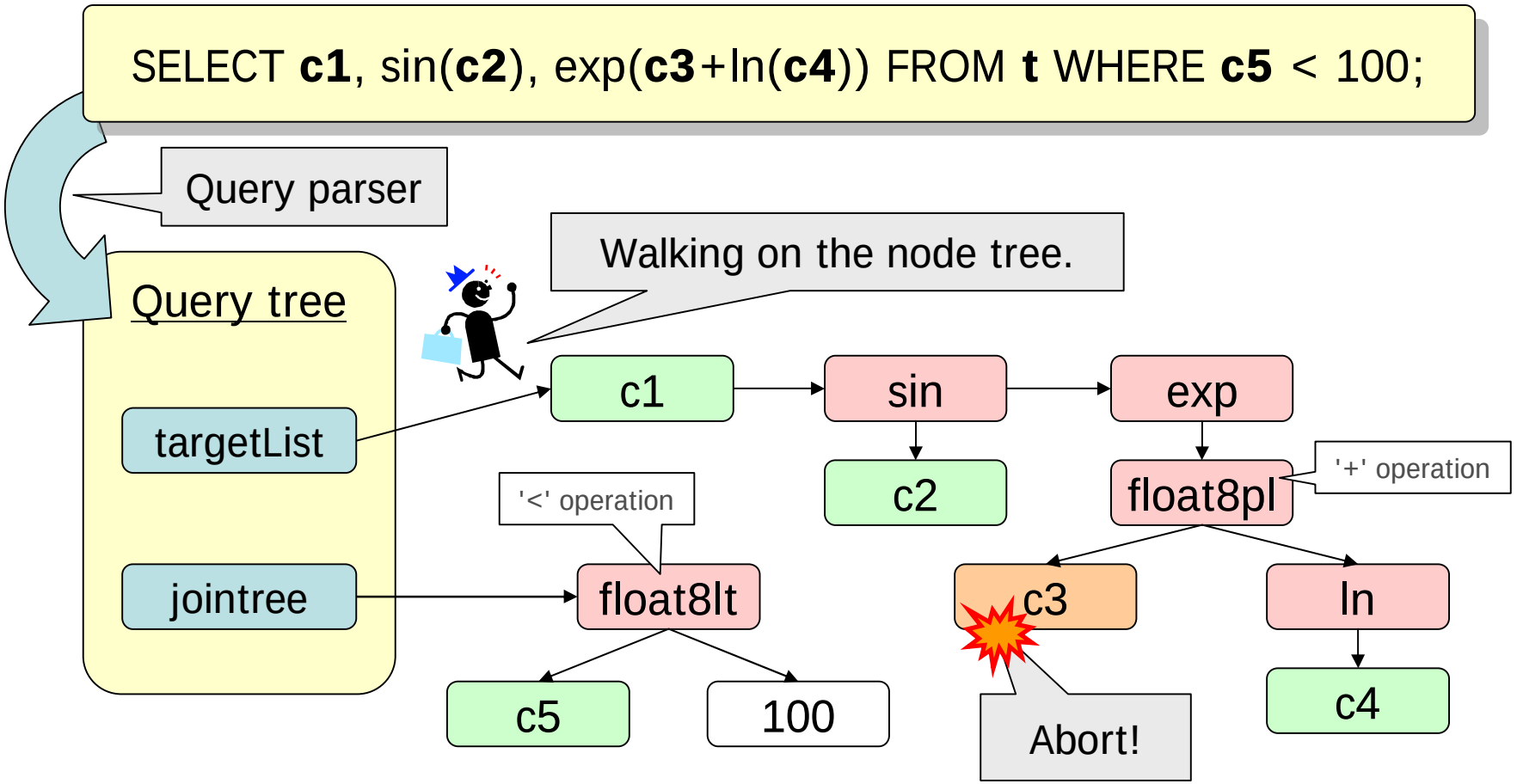
Tuple-level Access Controls

- SE-PostgreSQL filters any violated tuples from result set, as if they are not on the target table.
  - ditto, on UPDATE and DELETE statement
  - Checks at tuple insertion for INSERT statement



Column-Level Access Control

- SE-PostgreSQL checks any column appeared in queries.
  - Abort query execution, if violated usage found.



Case Study (1/2)

```
SELECT name, price * 2 FROM drink WHERE id < 40;
```

- db_column:{select} for **name** and **price** column
- db_column:{use} for **id** column
  - {use} permission means "referred but consumed internally"
- db_procedure:{execute} for **int4mul** and **int4lt** function
- db_table:{select use} for **drink** table
  - The current transaction will be aborted, if the client does not have enough permissions.

Implementation of operators.

And

- db_tuple:{select use} for each tuples
  - Any violated tuples are filtered from result set.

Case Study (2/2)

```
UPDATE drink SET size = 500, price = price * 2
                 WHERE alcohol = true;
```

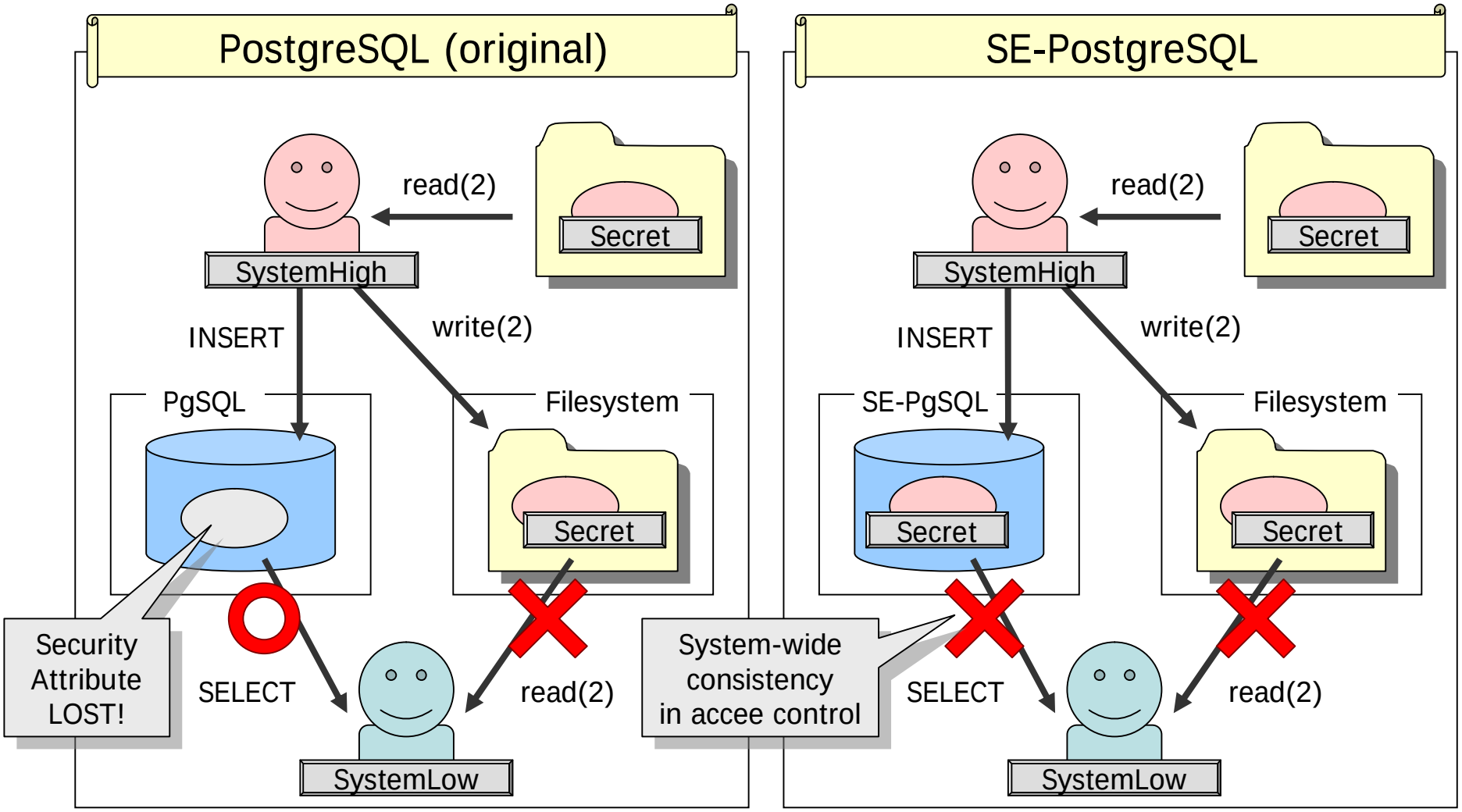
- db_column:{update} for **size** column
- db_column:{select update} for **price** column
  - price column is also read, not only updated.
- db_column:{use} for **alcohol** column
- db_procedure:{execute} for **booleq** and **int4mul** function
- db_table:{select use update} for **drink** table
  - The current transaction will be aborted,
    if the client does not have enough permissions.

And

- db_tuple:{select use update} for each tuples
  - Any violated tuples are excepted from the target of updating.

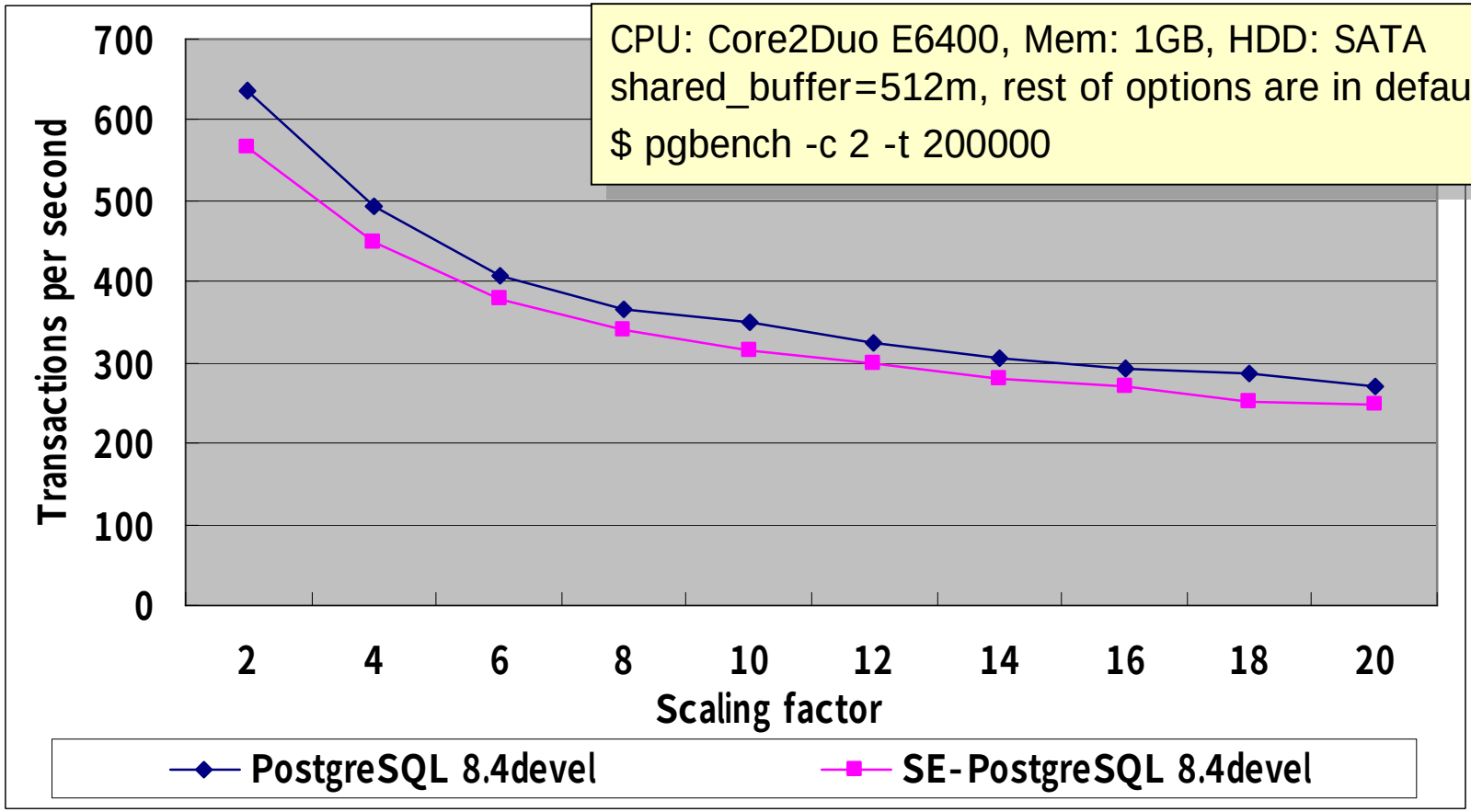
Demonstration

Data Flow Control Demonstration



Miscellaneous Topics

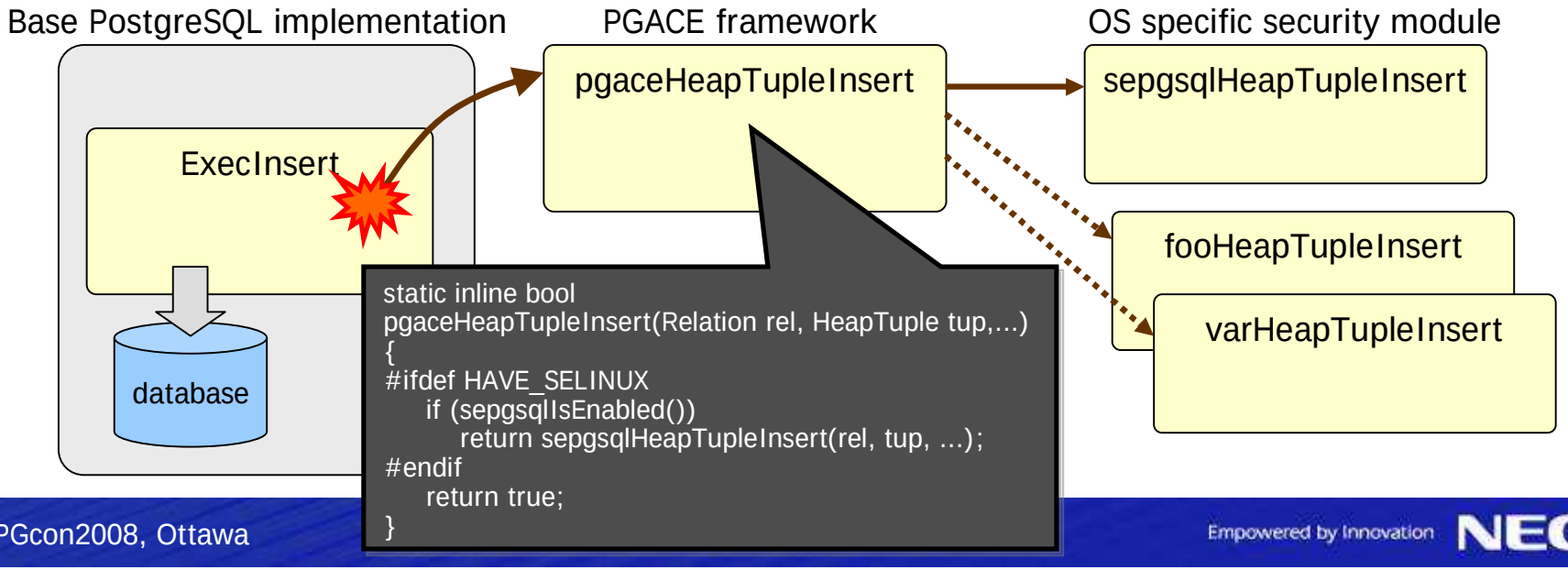
Performance



- about 10% security-tradeoff
- access vector cache (AVC) minimizes system-call invocation

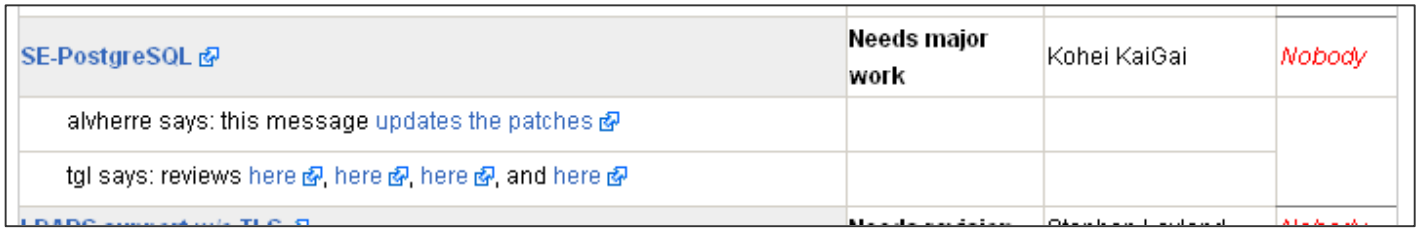
Platform dependency

- SE-PostgreSQL always needs SELinux to run.
  - Is SE-PostgreSQL available on disabled SELinux?
  - Is SE-PostgreSQL available on any other operating system?
- PostgreSQL Access Control Extension (PGACE)
  - A set of platform independent hooks
  - To apply various kind of security module with minimum impact



The current status of SE-PostgreSQL

- The current status
  - Now, it is available on Fedora 8 or later
  - Patches are reviewed at CommitFest:May
    - Thanks for many worthful comments/suggestions!
      http://wiki.postgresql.org/wiki/CommitFest:May



SE-PostgreSQL	**Needs major work**	Kohei KaiGai	*Nobody*
alvherre says: this message updates the patches			
tgl says: reviews here, here, here, and here			

- In the next
  - Now revising my patches for CommitFest:Jul
    - design improvement, documentation, regression test, ...
  - Security Policy Upstreaming

Summary

- "System-wide" Consistency in Access Controls

  ITS PHILOSOPHY:

  - Same access control policy should be applied to same information asset, independent from the way to store.
  - Key concept is sharing a single unified security policy.

- Fine-grained Mandatory Access Controls

  - Non-bypassable for everyone
  - Column-/Tuple-level flexibility
    - Any violated tuple is filtered, as if they don't exist.
    - Using violated column and others invokes execution aborts.

Any Question?

Thank you!

Acknoledgement:
Information-Technology Promotion Agency (IPA), Japan supported the development of SE-PostgreSQL as one of the Exploratory Software Projects in later half of 2006FY.

Resources

- Project Home
  - http://code.google.com/p/sepgsql/
  - SVN repository
    - svn co http://sepgsql.googlecode.com/svn/ sepgsql
  - Today's slide
    - http://sepgsql.googlecode.com/files/PGCON20080523.pdf
- RPM Packages
  - http://code.google.com/p/sepgsql/downloads/list
  - And, see the repository of Fedora project
- Logo

Currently, he has no name.