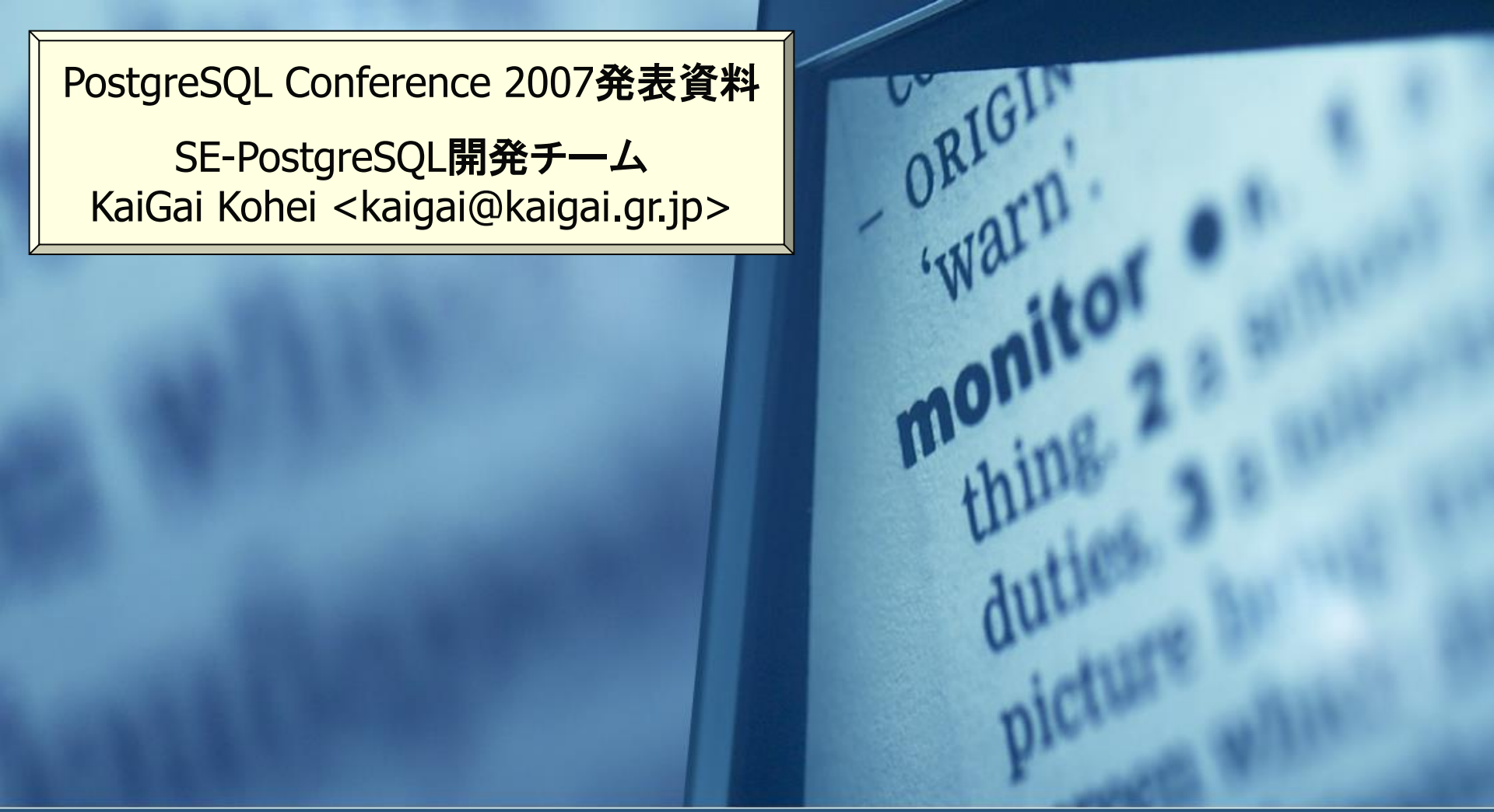


PostgreSQL Conference 2007発表資料

SE-PostgreSQL開発チーム
KaiGai Kohei <kaigai@kaigai.gr.jp>

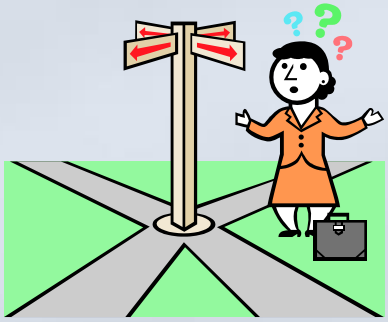
Security-Enhanced PostgreSQL

～統合されたOS・RDBMSセキュリティポリシー～

SE-PostgreSQLのコンセプト

OS(SELinux)のセキュリティポリシーに基づく
データベースのアクセス制御

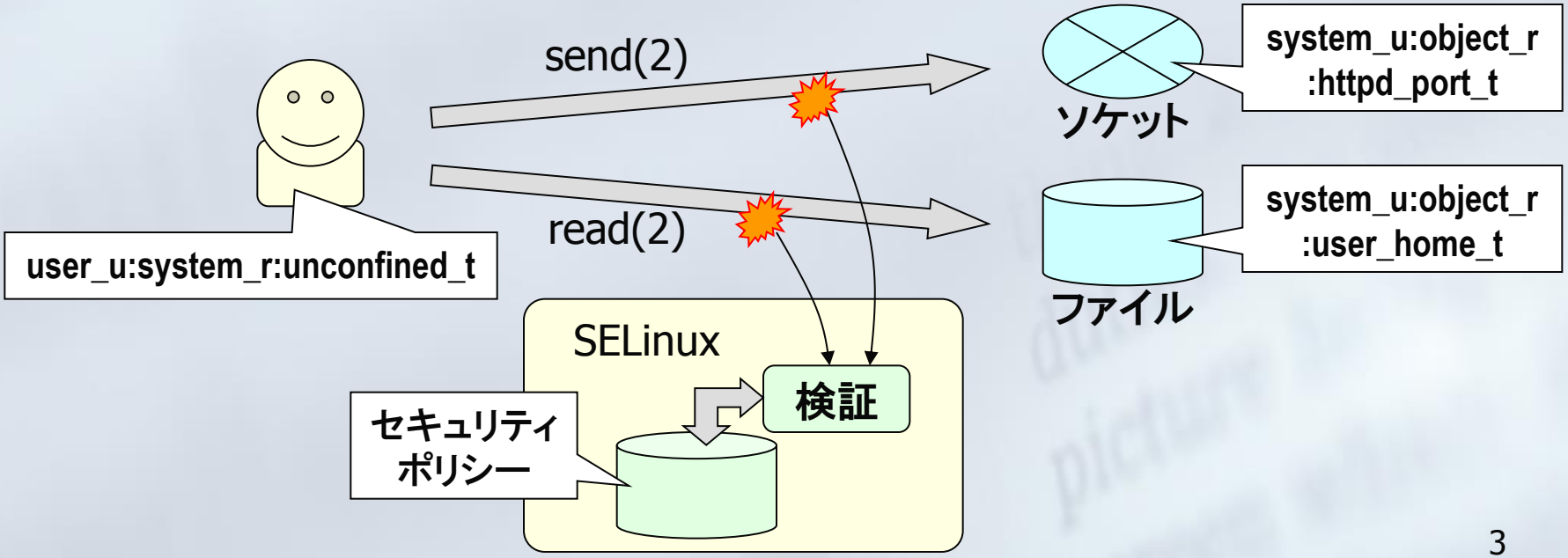
SE-PostgreSQL
- **システムワイドに一貫したセキュリティポリシー**
  - **情報の格納先がファイルでも、データベースでも、同じようにアクセス制御が適用される**



従来のPostgreSQL
- **OS/RDBMSで別々のアクセス制御ポリシー**
  - **OS上で権限の低いプロセスでも、DBに格納された機密情報にアクセスできてしまう。**

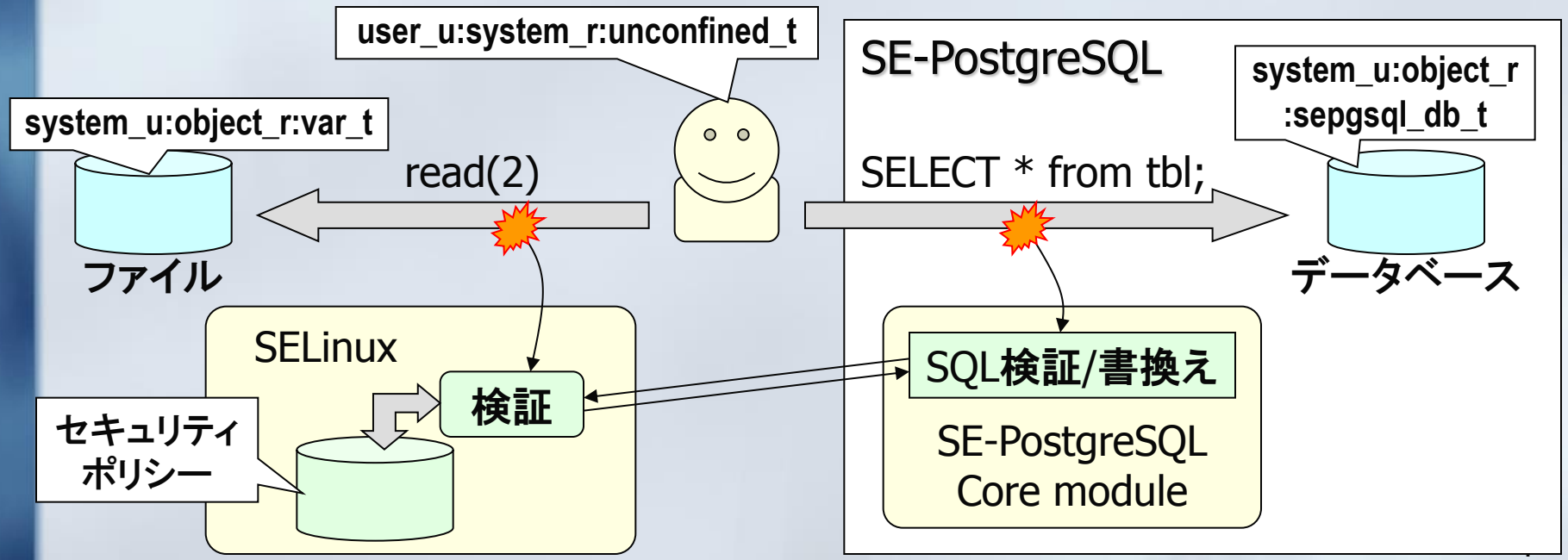
SELinuxのアクセス制御

- システムコールの実行をフックして権限チェック
  - プロセスの権限／システムコール種類／リソースの属性
    - 「誰が」「何に」「何をするか」・・・許可？禁止？
  - セキュリティポリシー ＝ ルールの集合
  - セキュリティコンテキスト ＝ 権限および属性

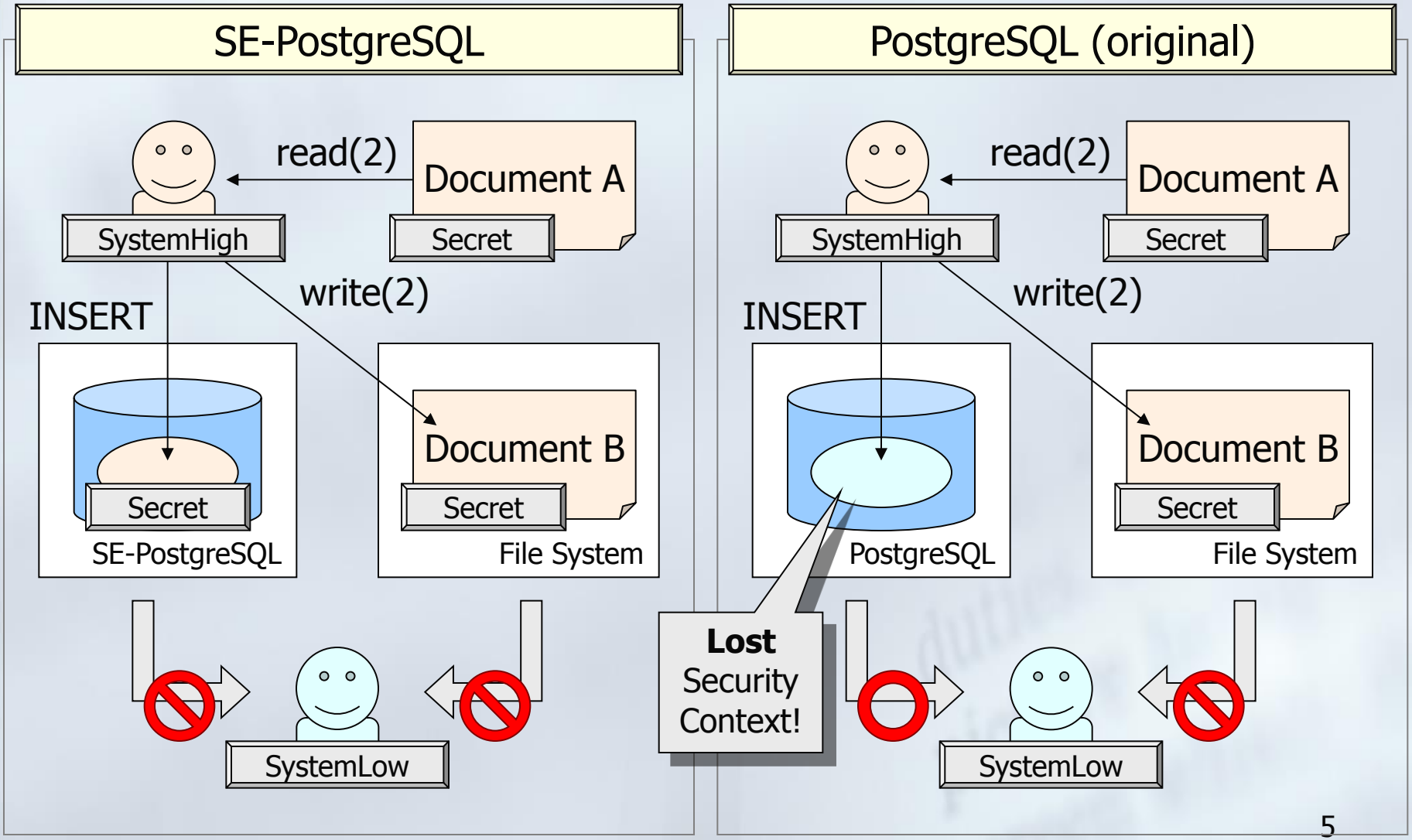


SE-PostgreSQL/SELinuxの連携

- SQLクエリの実行をフックして権限チェック
  - クエリの検査/書換えにより、権限のない操作を拒否する
    - OS(SELinux)のセキュリティポリシーに問い合せる
  - DBオブジェクトにセキュリティコンテキストを関連付け



情報フロー制御

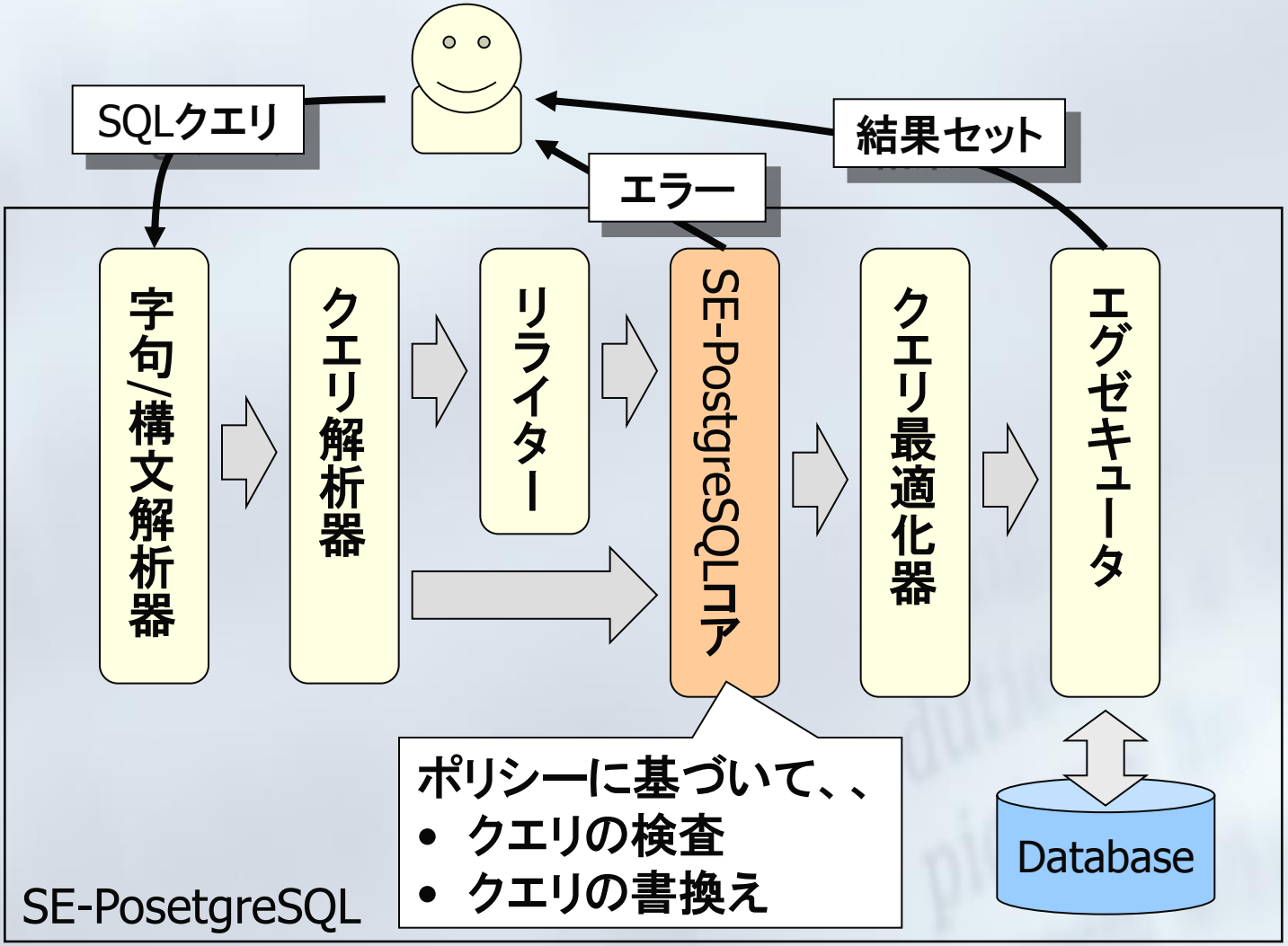

SE-PostgreSQL
read(2)
Document A
Secret
SystemHigh
write(2)
INSERT
Secret
SE-PostgreSQL
Document B
Secret
File System
SystemLow
PostgreSQL (original)
read(2)
Document A
Secret
SystemHigh
write(2)
INSERT
PostgreSQL
Document B
Secret
File System
SystemLow
Lost
Security
Context!

細粒度・強制アクセス制御

- PostgreSQLアクセス制御リスト
  - 特権DBユーザに対しては適用されない
  - テーブル以下の粒度でアクセス制御は不可
    - ✖ 列レベル/行レベルアクセス制御

- SE-PostgreSQL強制アクセス制御
  - 特権DBユーザを含む、全てのクライアントに対して適用
- 列レベル/行レベルアクセス制御
  - カラム/タプルの”セキュリティ属性”による制御
    - ✓ カラム：セキュリティポリシーに違反すると、エラーになる
    - ✓ タプル：セキュリティポリシーに違反すると、
      タプルは結果セットからフィルタリングされる

SE-PostgreSQLの実装



Case Study (1)

```
SELECT c1, c2 + 30 FROM tbl1 WHERE c3 > 10;
```

- tbl1に対する table:{select} 権限
- c1, c2, c3 に対する column:{select} 権限
- int4pl関数、int4gt関数に対する procedure:{execute}権限
  - ‘+’演算子、’ >’演算子を実装する関数
- 権限がないと、エラーを返してトランザクションをアボート

- 各タプルに対する tuple:{select} 権限
- 権限のないタプルは結果セットから除外される

Case Study (2)

```
UPDATE tbl2 SET x = 'abc', y = 1.05 * y WHERE z = true;
```

- tbl2に対する table:{select update} 権限
- x に対する column:{update} 権限
- y に対する column:{select update} 権限
- z に対する column:{select} 権限
- UPDATE構文であっても、カラムの参照を伴うケース
- int4mul関数、booleq関数に対する procedure:{execute}権限

- 各タプルに対する tuple:{select update} 権限
- 権限のないタプルは更新の対象から除外

クエリ書換え／行レベルアクセス制御

- sepgsql_tuple_perms()関数
  - タプルに対するパーミッションを持っていればtrueを返す
- 一般的なクエリ
  - SELECT * FROM t1 WHERE a > 0;
  - ⇨ SELECT * FROM t1 WHERE a > 0
    and sepgsql_tuple_perms(t1.security_context, ...);
- JOIN結合を含むクエリ
  - SELECT * FROM t1 JOIN t2 ON t1.a = t2.x;
  - ⇨ SELECT * FROM t1 JOIN t2 ON t1.a = t2.x
    and sepgsql_tuple_perms(t1.security_context, ...)
    and sepgsql_tuple_perms(t2.security_context, ...);

セキュリティコンテキスト

system_u:system_r:unconfined_t:SystemLow-SytemHigh

- SELinuxのセキュリティ属性を示す文字列
  - プロセスの権限やリソースの属性を意味する
  - アクセス制御対象のDBオブジェクトに関連付ける
- ‘security_context’ システム列
  - タプルのセキュリティコンテキストを参照・更新可能
  - テーブル/カラム/関数etc、のセキュリティコンテキストは？
    - 対応するシステムカタログを参照
    - pg_class, pg_attribute, pg_procなど

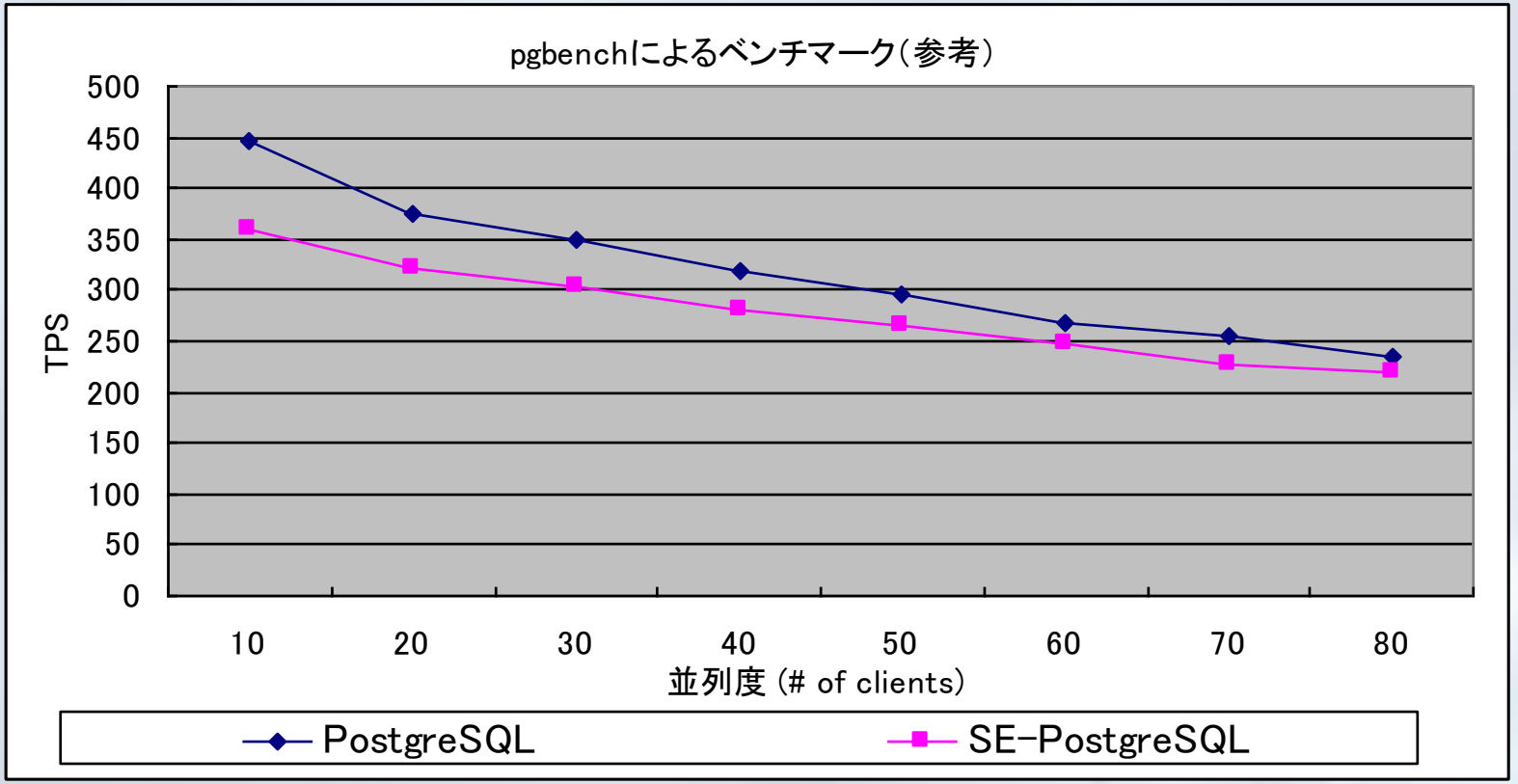
Case Study (3)

```
INSERT INTO tbl3 (id, name, reg_ymd)
VALUES(10, 'KaiGai', CURRENT_DATE);
```

- tbl3に対する table:{insert} 権限
- id, name, reg_ymd に対する column:{insert} 権限
- date()関数に対する procedure:{execute} 権限
- タプルに対する tuple:{insert} 権限
  - 新しいタプルのセキュリティコンテキストとは？
  - ✓ セキュリティポリシーに基づいて、暗黙的に付与される
  - ✓ 暗黙のセキュリティコンテキストへの tuple:{insert} 権限
  - ✓ 明示的にセキュリティコンテキストを指定することもできる
    - 'security_context'システム列に値を指定する

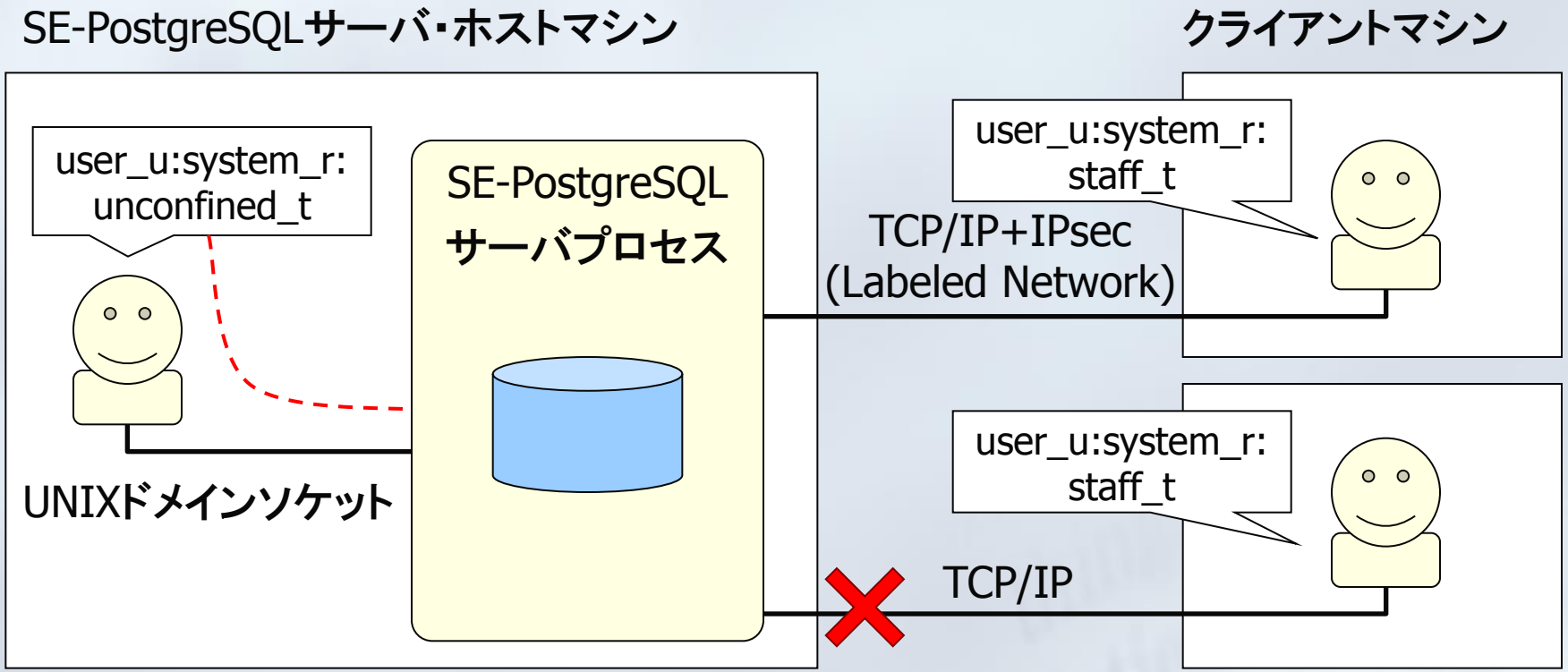
パフォーマンス (ご参考)

- pgbenchによる測定
- Core2DuoE6400, Memory: 1GB, HDD: Serial-ATA
- Scaling Factor = 10, Number of total transaction = 120,000



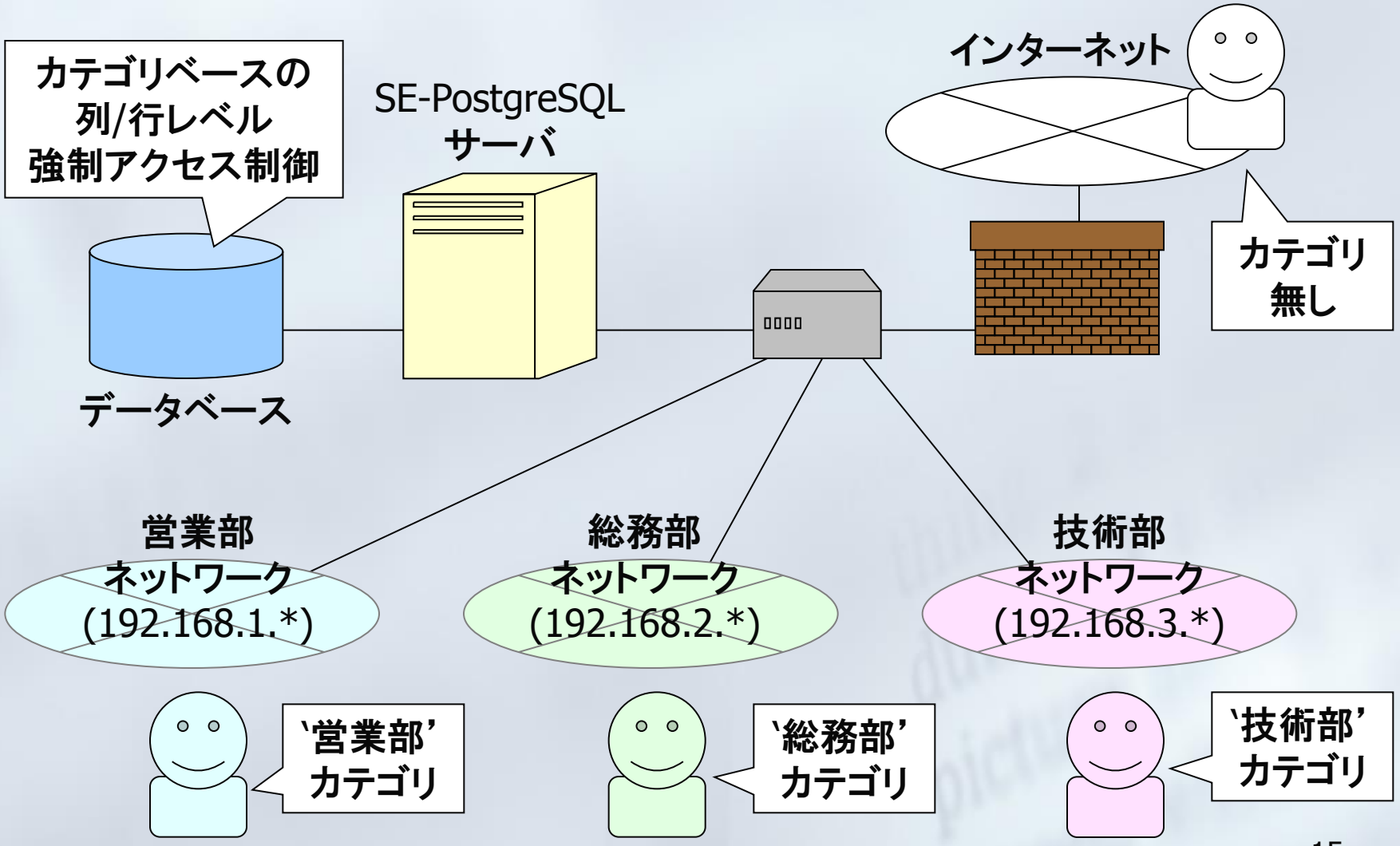
※ shared_buffers = 512MB で測定、残りのパラメータはデフォルト値

SE-PostgreSQL利用環境 (1)



- 同一ホスト内 ⇒ 特別な設定無しに利用できる
- 他のホストからの接続
  ⇒ IPsec + Labeled Networkを用いることで利用可能に

SE-PostgreSQL利用環境 (2)



※ 接続元IPアドレス/ポートに基づくクライアント権限は、現在開発中の機能

デモンストレーション

その他の諸機能

- SQL構文の拡張
  - テーブル・SQL関数etcのセキュリティコンテキストを設定

```
CREATE TABLE tbl (
    id integer primary key,
    data text CONTEXT='user_u:object_r:sepgsql_secret_table_t',
) CONTEXT='user_u:object_r:sepgsq_table_t:SystemHigh';
```

- バックアップ/リストア
  - pg_dump –enable-security オプション
- PGACE(PostgreSQL Access Control Extension)
  - SELinux以外のセキュアOS向けの共通フレームワーク
    - ✓ Trusted Solaris開発者との議論の中で生まれた

OSSコミュニティでの開発

- SELinuxコミュニティ
  - 設計段階で数多くのアイデア・フィードバック
    - Object Classの構成、Trusted Procedureなど
  - SELinux Symposium/Developer Summit 2007
  - カーネル側機能の対応
    - InitialSID, Object Class/Access Vector取得のI/F
- PostgreSQLコミュニティ
  - Trusted Solaris 開発者からのコンタクト
    - ✓共通フレームワークとして PGACE を開発、実装
  - 本格的な議論はPostgreSQL8.3betaの後に開始
  - Upstreamed PostgreSQL へのマージを目指す

今後の予定

- SE-PostgreSQL 1.0に向けて
  - 現在：1.0α版　　… コミュニティからのフィードバック
  - ’07/6末：1.0β版　… stable1.0に向けた機能フリーズ
  - ’07/7末：stable 1.0版（予定）

- 今後の開発方針
  - PostgreSQLコミュニティ、Fedora Projectへのプッシュ
  - Upstreamへのマージを目指す
  - 新機能の追加
    - ✓ Polyinstantiation Tableサポート
    - ✓ pl/pgSQLスクリプトのサポート
    - ✓ システム監査機能(auditd)との統合

情報源

- 開発者向け情報源
  - http://code.google.com/p/sepgsql/
    - Subversionリポジトリ/RPM,SRPM/ドキュメント
  - 「SE-PostgreSQL Security Guide 1.0」
- メディアでの紹介
  - 日本のセキュアOSを支える5つのプロジェクト

http://www.atmarkit.co.jp/fsecurity/special/100jpsecureos/jpsecureos02.html

  - セキュリティ強化OSS DBMS「SE-PostgreSQL」, α版公開

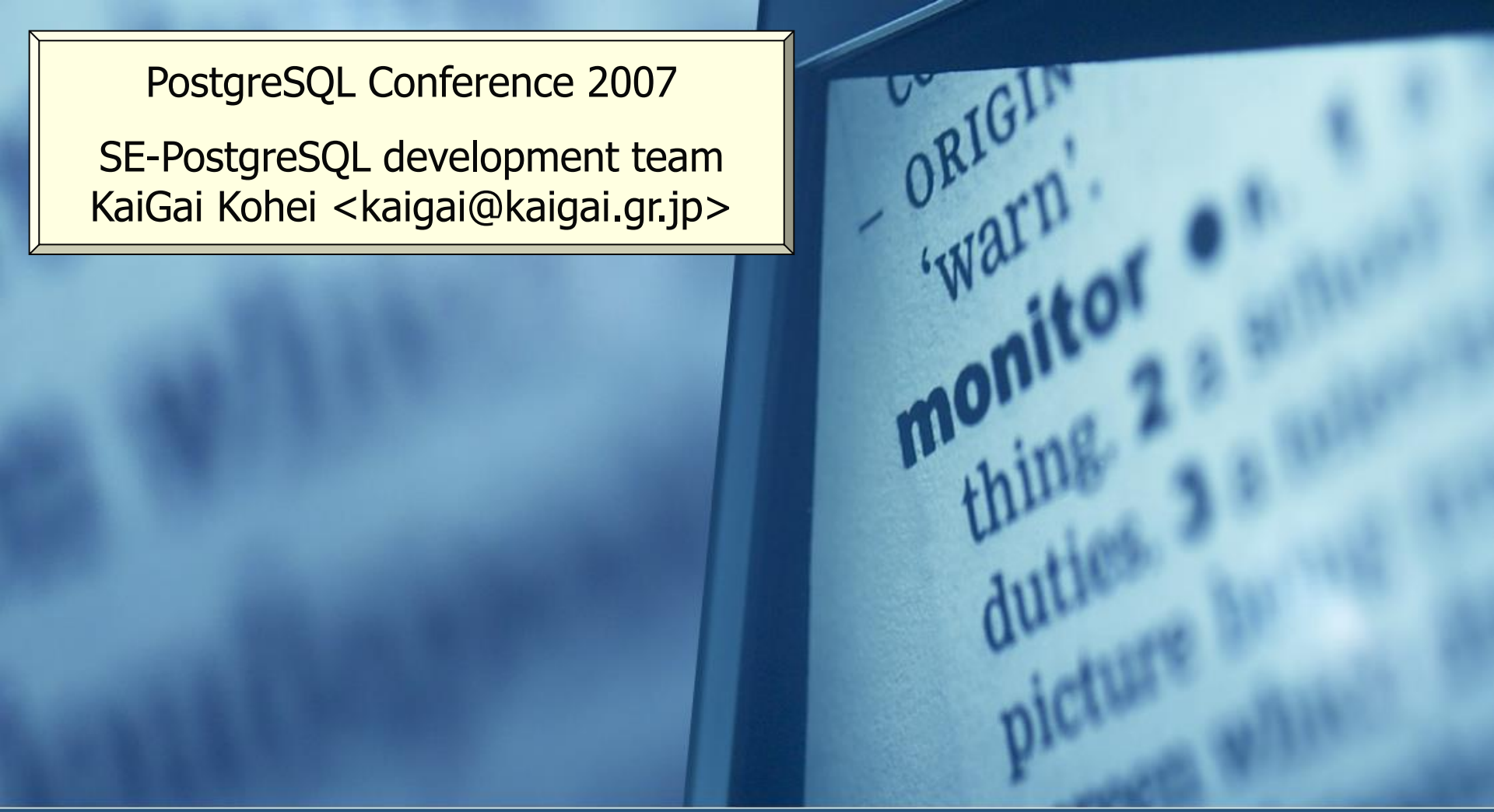
http://itpro.nikkeibp.co.jp/article/NEWS/20070305/263930/

SE-PostgreSQLの開発は、IPA 未踏ソフトウェア創造事業
(2006年度/下期)の支援を受けています。

Q&A

Any Question?

ありがとうございました



PostgreSQL Conference 2007

SE-PostgreSQL development team
KaiGai Kohei <kaigai@kaigai.gr.jp>

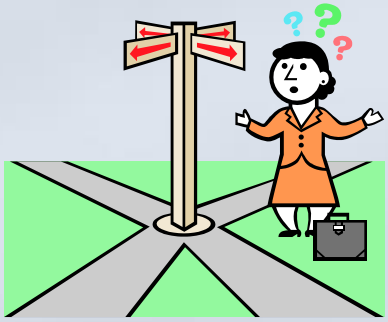
Security-Enhanced PostgreSQL

- Integrated security policy between OS and RDBMS -

The concept of SE-PostgreSQL

Access control for database objects,
based on the security policy of OS(SELinux)

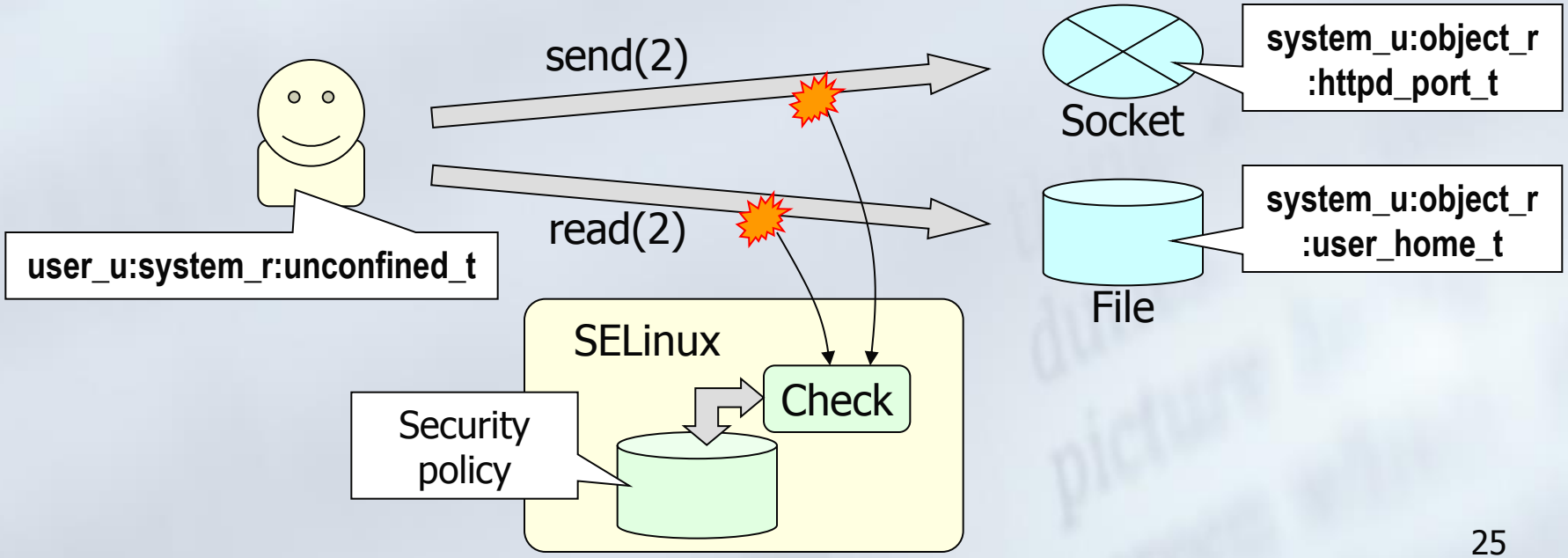
SE-PostgreSQL
- System wide consistent security policy
  - Same access control policy is applied for both files and databases



native PostgreSQL
- Sepatated security policy for OS and RDBMS
  - There are no relationship between authority of processes on OS and authority of database users.

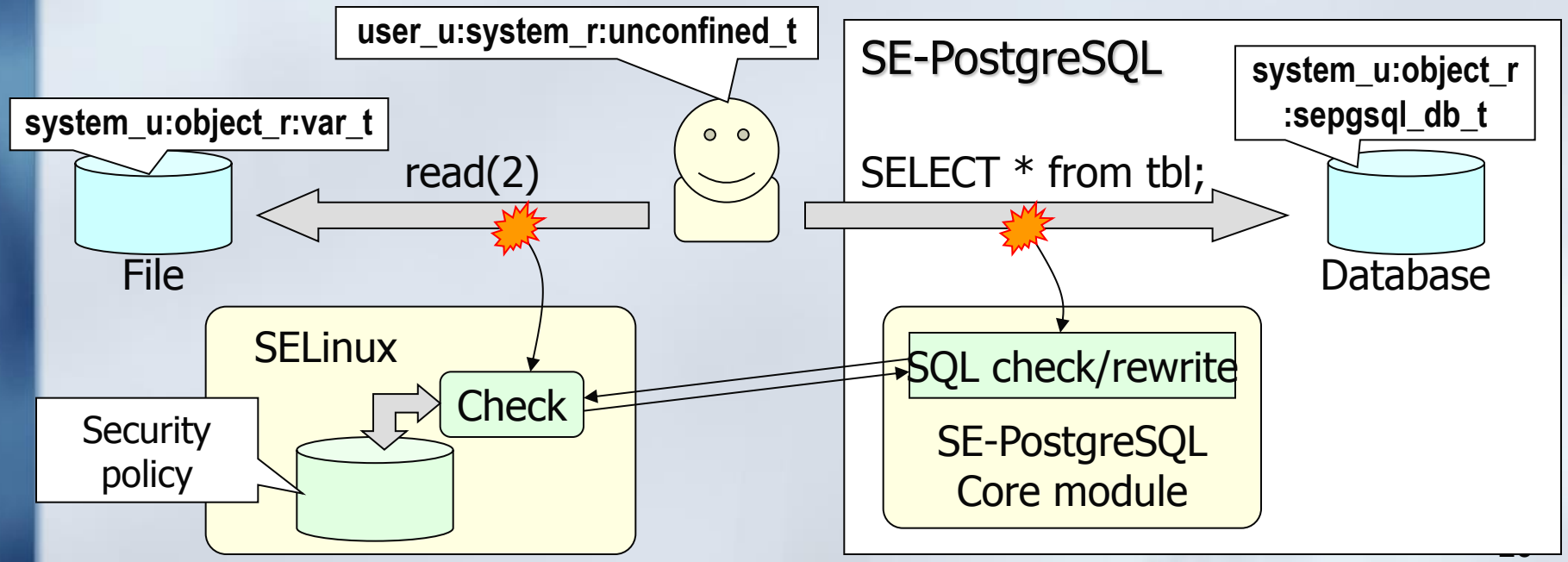
Access controls by SELinux

- SELinux hooks system calls to check it
  - Authotity of process/kind of system call/attribute of resource
    - combination of "subject","object", "action" ... allowed? denied?
  - Security policy = a set of rules
  - Security context = authority and attribute

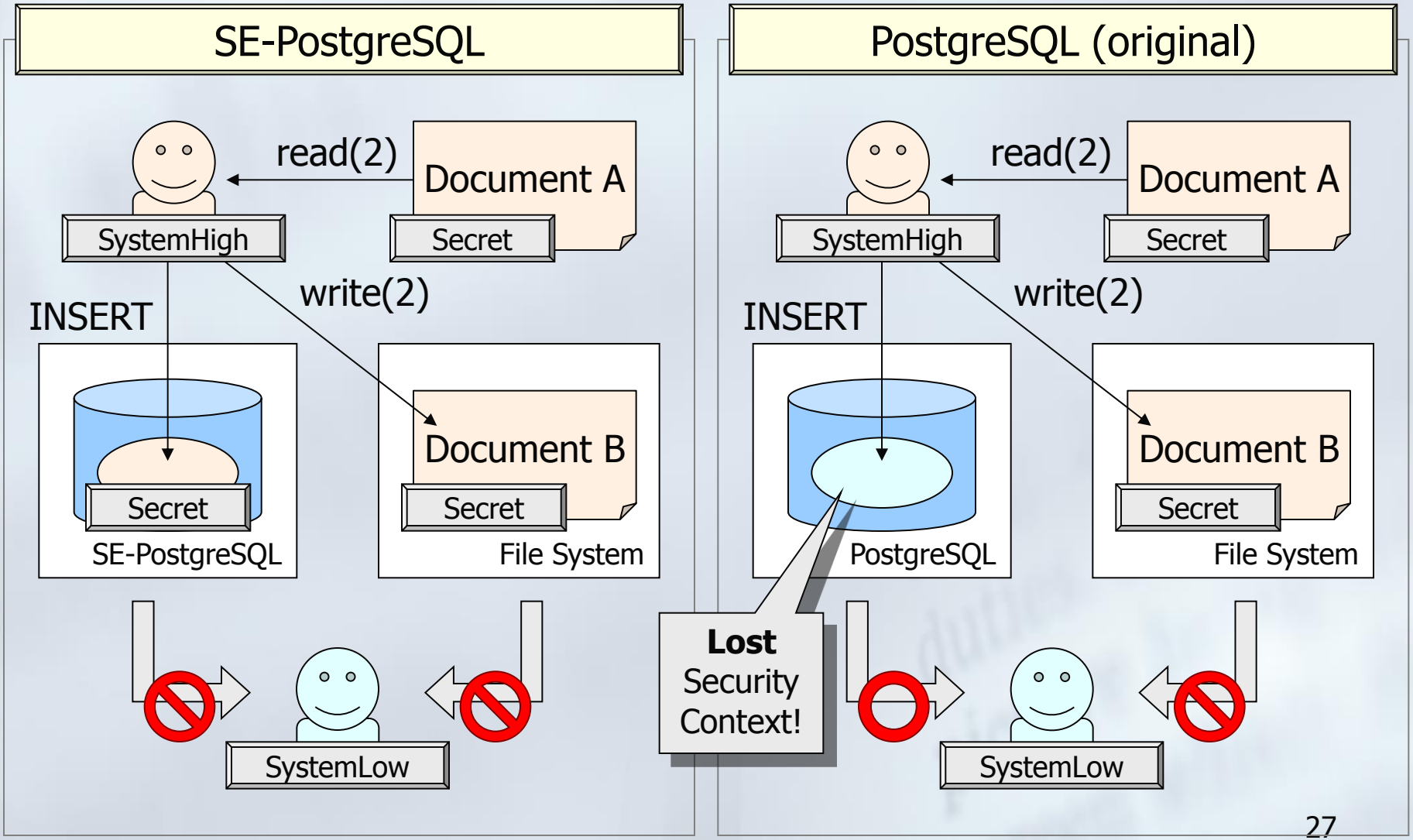


Collabotation between SE-PostgreSQL and SELinux

- SE-PostgreSQL hooks execution of SQL to check it
  - It checks and re-writes SQL query not to allow any operation without appropriate authorities.
    - It asks OS(SELinux)'s security policy to make a decision.
  - It associates a security context with database object.



Information flow control

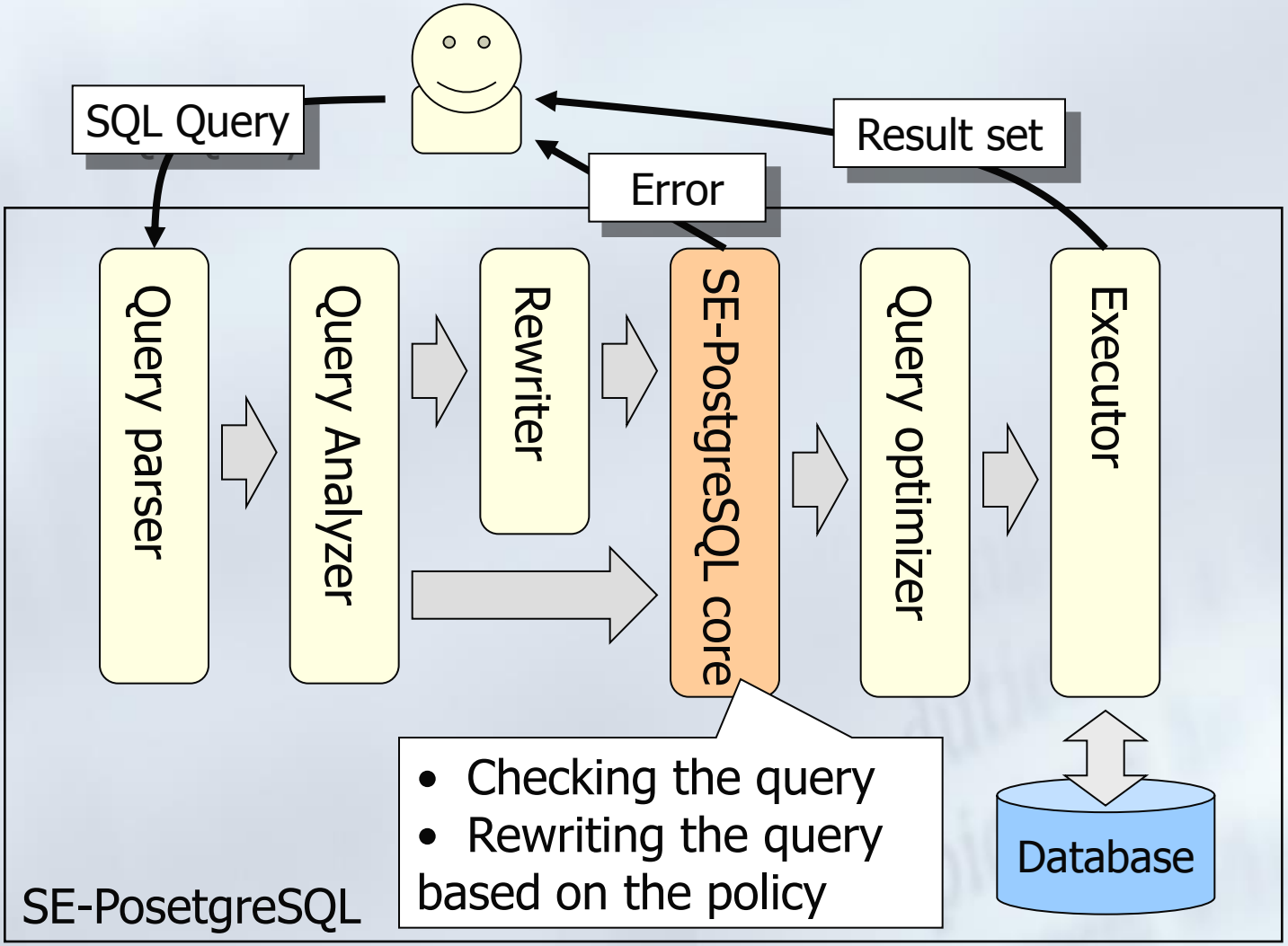


Fine grained mandaroty access control

- PostgreSQL access control list
  - Privileged DB user can bypass them
  - It cannot provide finer grained access controls than table level
    - ✖ Row-level/Column-level access control

- SE-PostgreSQL mandatory access control
  - It's applied for any client including privileged DB user.
- Row-level/Column-level access control
  - Controled by "security attribute" of column/tuple
    - ✓ Column case: current transaction will be aborted, if violated.
    - ✓ Tuple case: violated tuples are filtered from result set.

Implementation of SE-PostgreSQL



Case Study (1)

```
SELECT c1, c2 + 30 FROM tbl1 WHERE c3 > 10;
```

- table:{select} permission for tbl1 table
- column:{select} permission for c1, c2, c3 column
- procedure:{execute} permission for int4pl, int4gt function
  - Those functions implement '+' and '>' operator
- If client doesn't have enough permission, it returns an error and abort the current transaction.

- tuple:{select} permission for each tuple
- violated tuples are filtered from the result set

Case Study (2)

```
UPDATE tbl2 SET x = 'abc', y = 1.05 * y WHERE z = true;
```

- table:{select update} permissions for tbl2 table
- column:{update} permission for 'x' column
- column:{select update} permissions for 'y' column
- column:{select} permission for 'z' column
- Columns may be refered, even if it's UPDATE statement.
- procedure:{execute} permission for int4mul and booleq function

- tuple:{select update} permissions for each tuple
- Violated tuples are filtered from the result set

Row-level access control

- sepgsql_tuple_perms() function
  - returns true, if client have permissiond on the tuple
- Simple query
  - SELECT * FROM t1 WHERE a > 0;
  - ⇨ SELECT * FROM t1 WHERE a > 0
    and sepgsql_tuple_perms(t1.security_context, ...);
- A query including JOIN clause
  - SELECT * FROM t1 JOIN t2 ON t1.a = t2.x;
  - ⇨ SELECT * FROM t1 JOIN t2 ON t1.a = t2.x
    and sepgsql_tuple_perms(t1.security_context, ...)
    and sepgsql_tuple_perms(t2.security_context, ...);

Security context

system_u:system_r:unconfined_t:SystemLow-SytemHigh

- A string that means security attribute for SELinux
  - It identifies authority of process, attribute of objects.
  - It's associated with controled DB objects.
- 'security_context' system column
  - It enables to refer/update security context of tuple
  - How does it handle security contexts of tables, columns and functions?
    - Those are stored in tuples within related system catalogs.
    - pg_class, pg_attribute, pg_proc

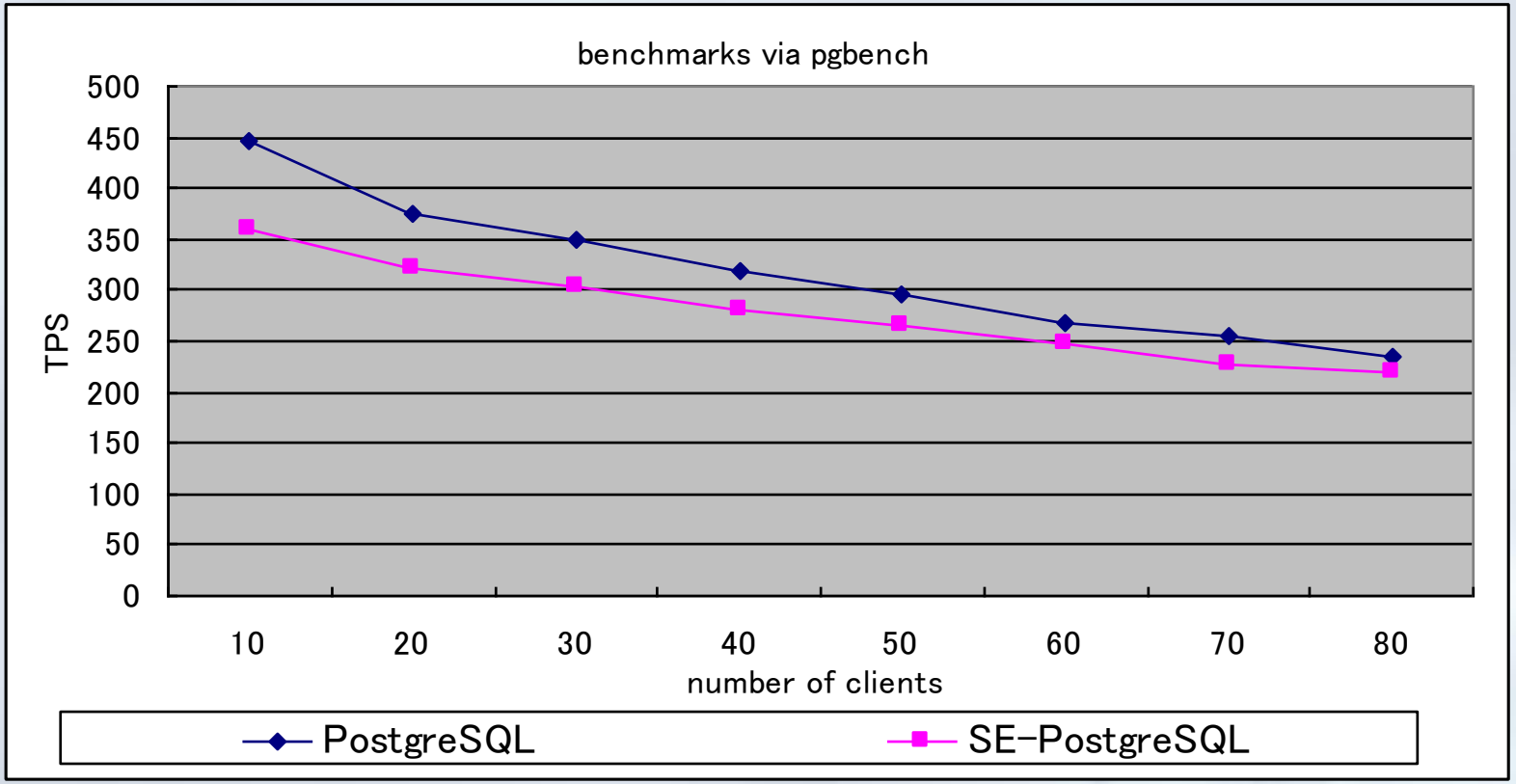
Case Study (3)

```
INSERT INTO tbl3 (id, name, reg_ymd)
VALUES(10, 'KaiGai', CURRENT_DATE);
```

- table:{insert} permission for tbl3
- column:{insert} permissions for id, name, reg_ymd
- procedure:{execute} permissions for 'date' function
- tuple:{insert} permission for newly generated tuple
  - What is the security context of newly generated tuple?
  - ✓ It's implicitly attached based on security policy.
  - ✓ tuple:{insert} permission for the implicit context.
  - ✓ We can specify the security context explicitly.
    - You should set a valid context for 'security_context' system column.

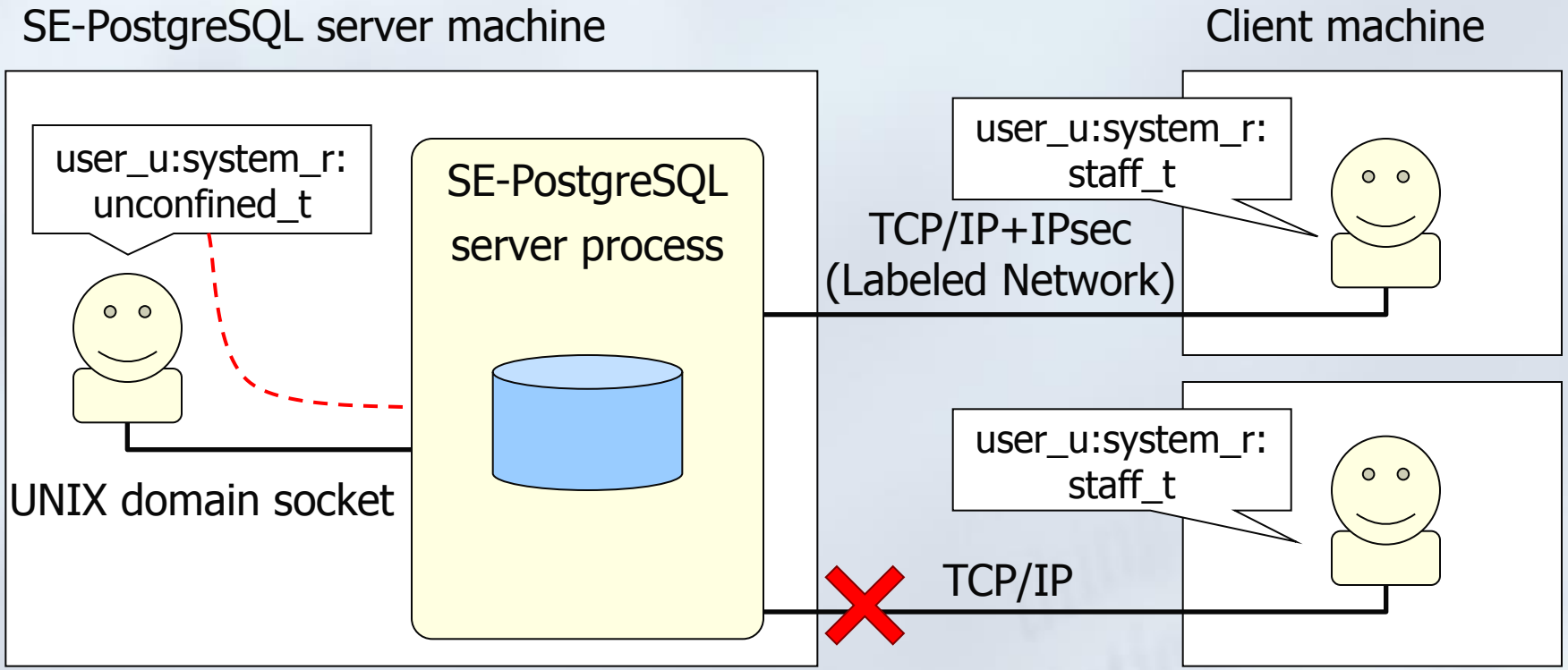
Performance (not a strict test)

- measurement via pgbench
- Core2DuoE6400, Memory: 1GB, HDD: Serial-ATA
- Scaling Factor = 10, Number of total transaction = 120,000



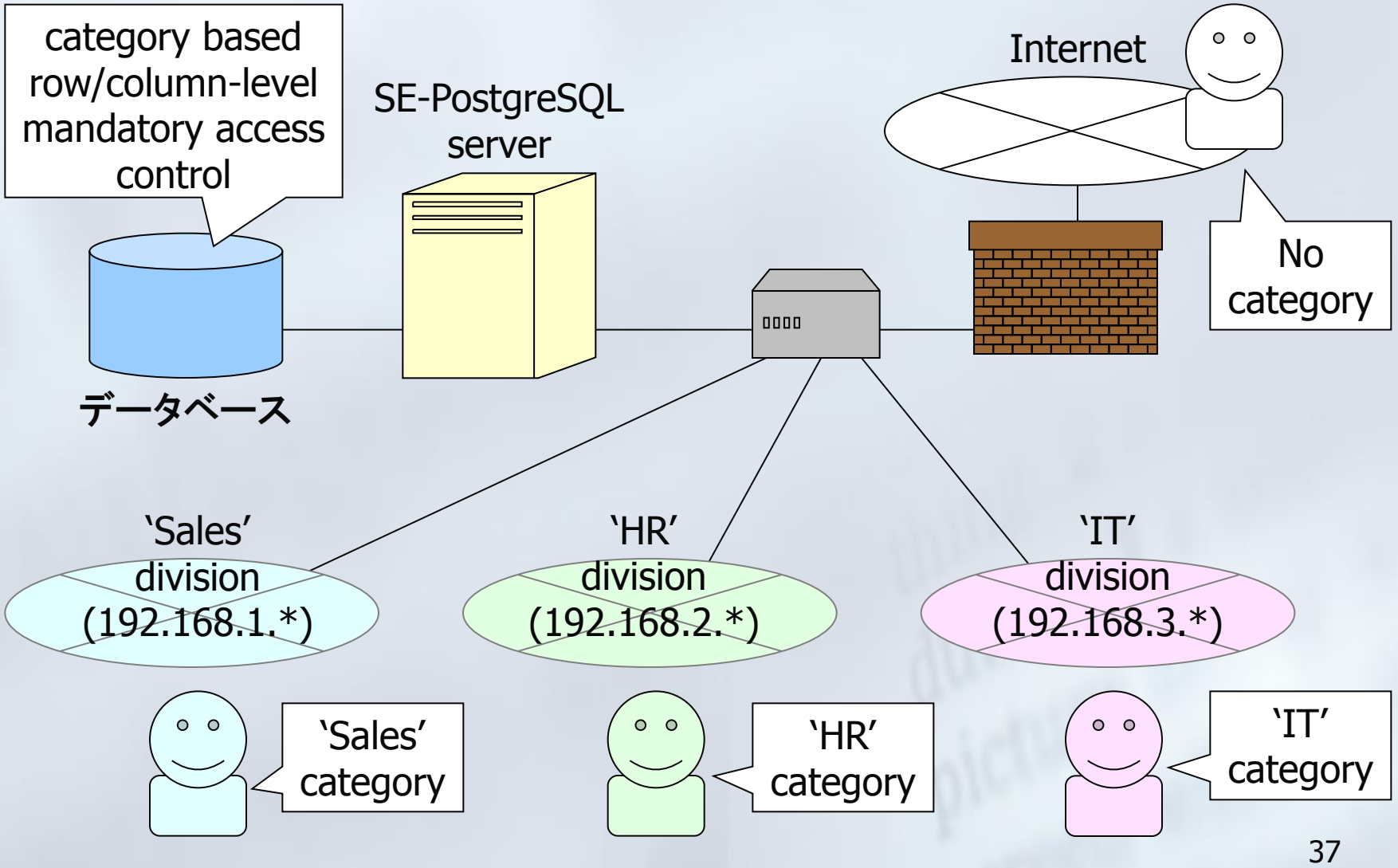
(*) We set shared_buffers = 512MB, rest of parameters are in default

System environment for SE-PostgreSQL(1)



- Connection from same host
  - No additional configurations are necessary
- Connection from other hosts
  - IPsec + Labeled Network are required to obtain a security context of client side.

System environment for SE-PostgreSQL(2)



(*) This features are now in development with SELinux community

Demonstration

Other facilities

- Extended SQL statement
  - We can configure security context of table, functions, etc

```
CREATE TABLE tbl (
    id integer primary key,
    data text CONTEXT='user_u:object_r:sepgsql_secret_table_t',
) CONTEXT='user_u:object_r:sepgsq_table_t:SystemHigh';
```

- Backup/Restore
  - pg_dump –enable-security option
- PGACE(PostgreSQL Access Control Extension)
  - Common framework for other secure operating system
    - ✓ It came from the discussion with Trusted Solaris developers

Development in OSS community

- SELinux community
  - Many ideas and feedbacks from early phase
    - composition of object classes, trusted procedure and so on
  - SELinux Symposium/Developer Summit 2007
  - Kernel extension
    - InitialSID, new interface to obtain Object Class/Access Vector
- PostgreSQL community
  - We has a discussion with Trusted Solaris developers
    - ✓ PGACE as a common framewors for both secure OSs.
  - Activities for upstreaming will start after PostgreSQL8.3 beta.
  - Of course, we intend to upstream SE-PostgreSQL

The current and future plan

- toward SE-PostgreSQL 1.0
  - Now: 1.0alpha to get feedbacks from community
  - ’07/6E: 1.0beta features freezed for stable 1.0
  - ’07/7E: 1.0stable stable 1.0 release
- next to SE-PostgreSQL 1.0
  - Push it into PostgreSQL community and Fedora project to upstream SE-PostgreSQL
  - new functionalities (not included in 1.0)
    - ✓ Polyinstantiation Table
    - ✓ pl/pgSQL script language
    - ✓ Integration with system audit (auditd)

Resources

- for Developers
  - http://code.google.com/p/sepgsql/
    - Subversion repository/RPM,SRPM/Documentation
  - 「SE-PostgreSQL Security Guide 1.0」
- introduced by media
  - Five secure OS related projects from Japan

http://www.atmarkit.co.jp/fsecurity/special/100jpsecureos/jpsecureos02.html

  - Security Enhanced OSS-DBMS "SE-PostgreSQL 1.0 alpha is released"

http://itpro.nikkeibp.co.jp/article/NEWS/20070305/263930/

Exploratory Software Project (2006/second semester) of IPA supports the development of SE-PostgreSQL.

Q&A

Any Question?

Thanks for your interest