Yokohama Linux User's Group
第７４回　カーネル読書会

The development of SE-PostgreSQL

KaiGai Kohei <kaigai@kaigai.gr.jp>

Agenda

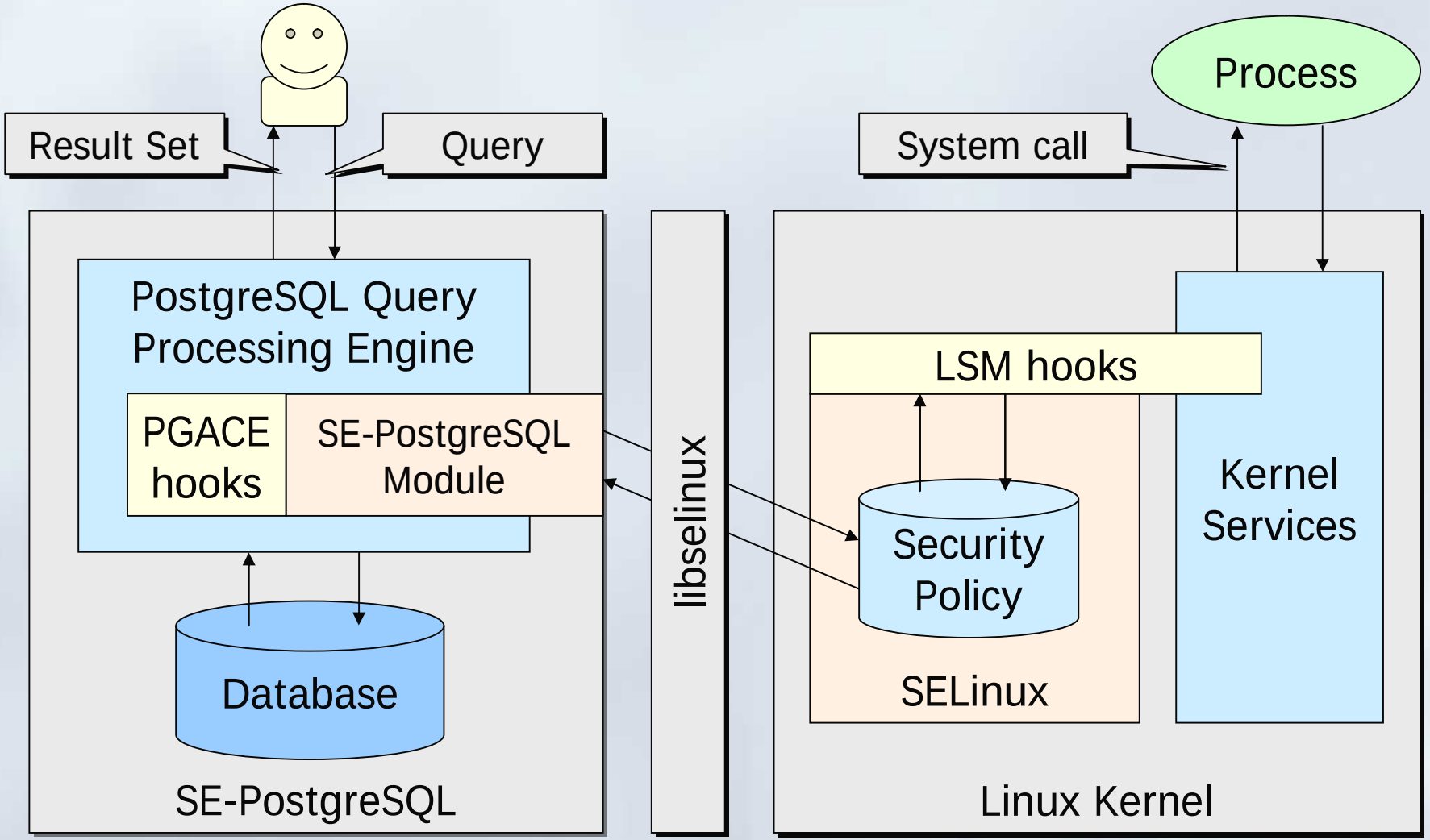
- SE-PostgreSQLとは？
  - SELinuxとの統合と情報フロー制御
  - 細粒度／強制アクセス制御
- SE-PostgreSQLアーキテクチャ
  - SQL実行フローとクエリ書換え処理
  - PGACE、セキュリティ属性の関連付け
- デモンストレーション
- コミュニティでの動向
- SE-PostgreSQLロードマップ

SE - PostgreSQLとは？

SE-PostgreSQLとは？

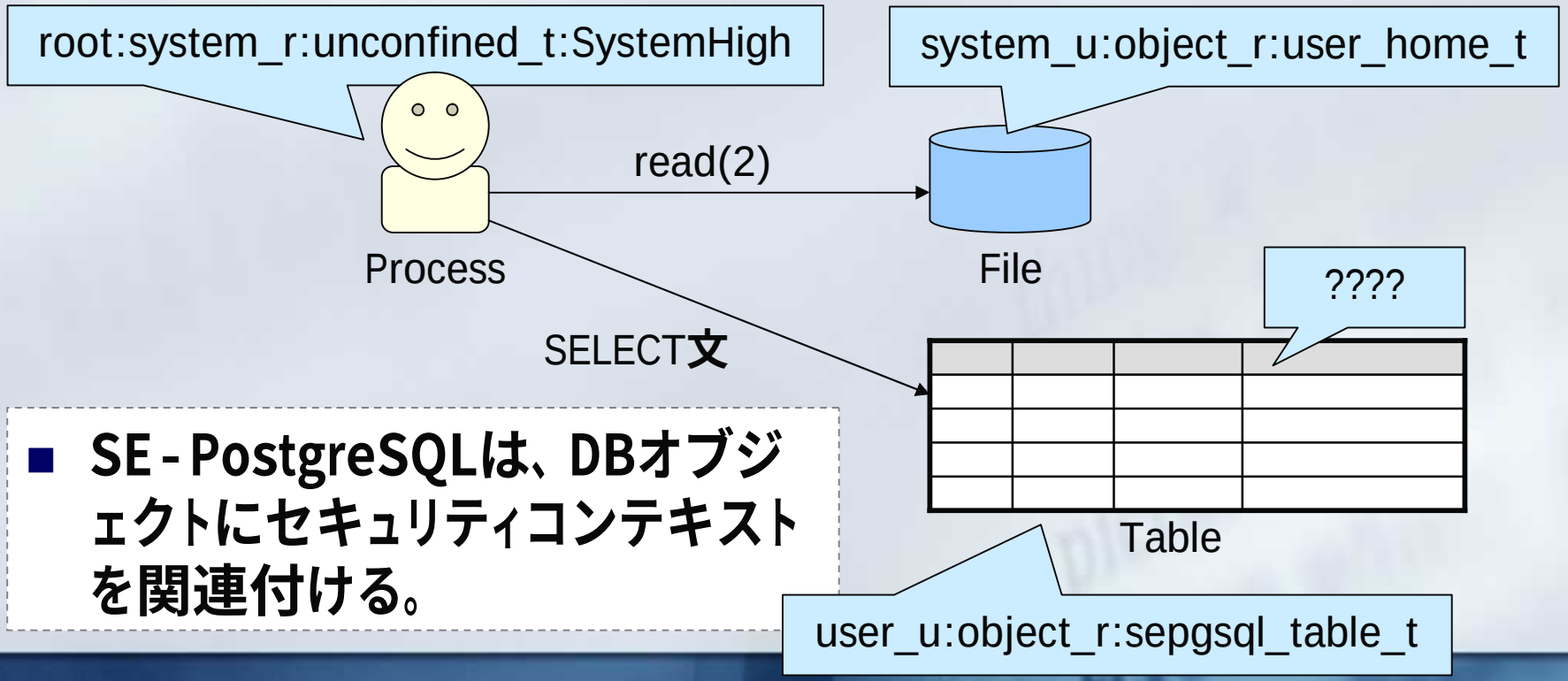
- OSと統合されたRDBMSのアクセス制御
  - SQLクエリに対するリファレンス・モニター
  - SELinuxセキュリティポリシーを利用したアクセス制御
  - クライアント・プロセスの権限をAs-Isで利用
- 強制アクセス制御
  - 特権ユーザであってもバイパス不可能
- 細粒度アクセス制御
  - 行レベル/列レベルアクセス制御
    - ✓標準PostgreSQLは表レベルまでの粒度

SELinuxとの統合(1)



SELinuxとの統合(2)

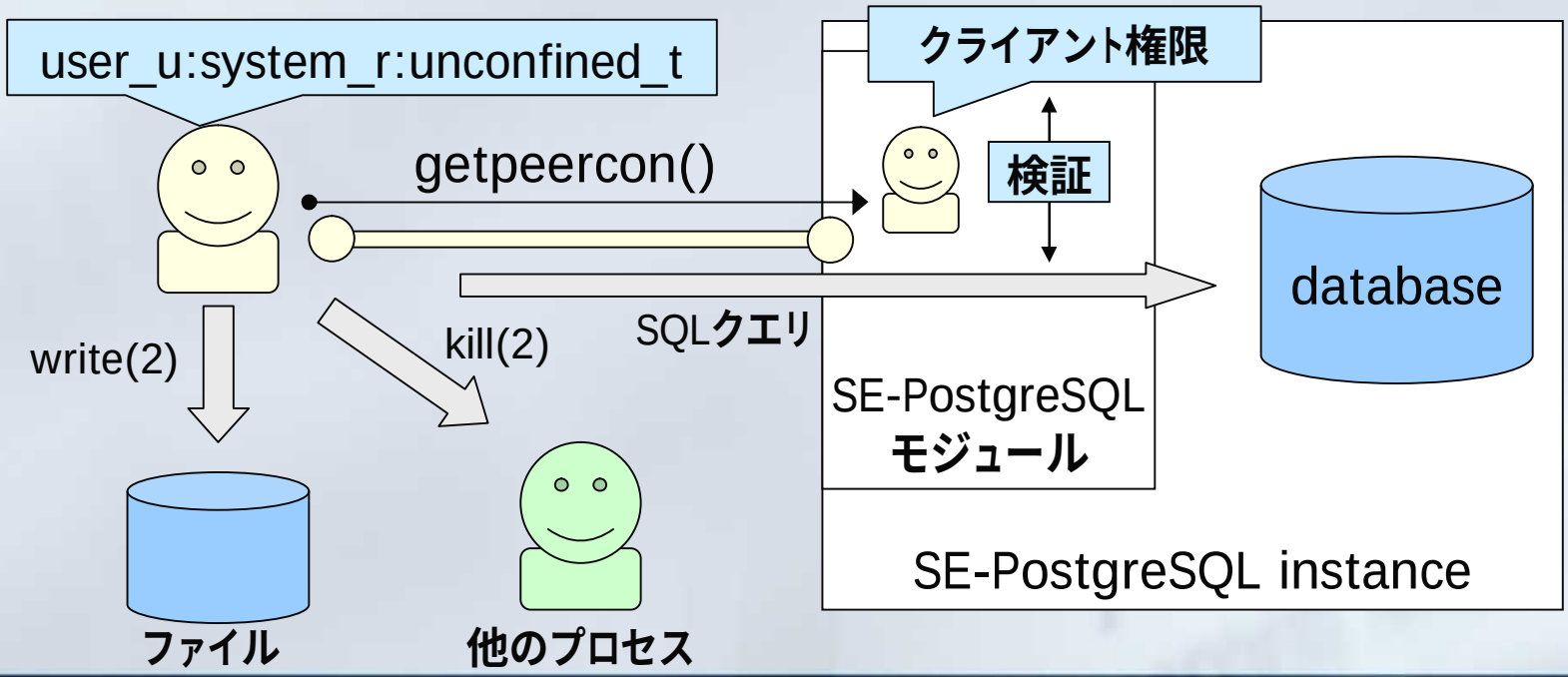
- セキュリティコンテキスト
  - セキュリティ属性を全て包含している
  - SELinuxがオブジェクトを識別する唯一のタグ



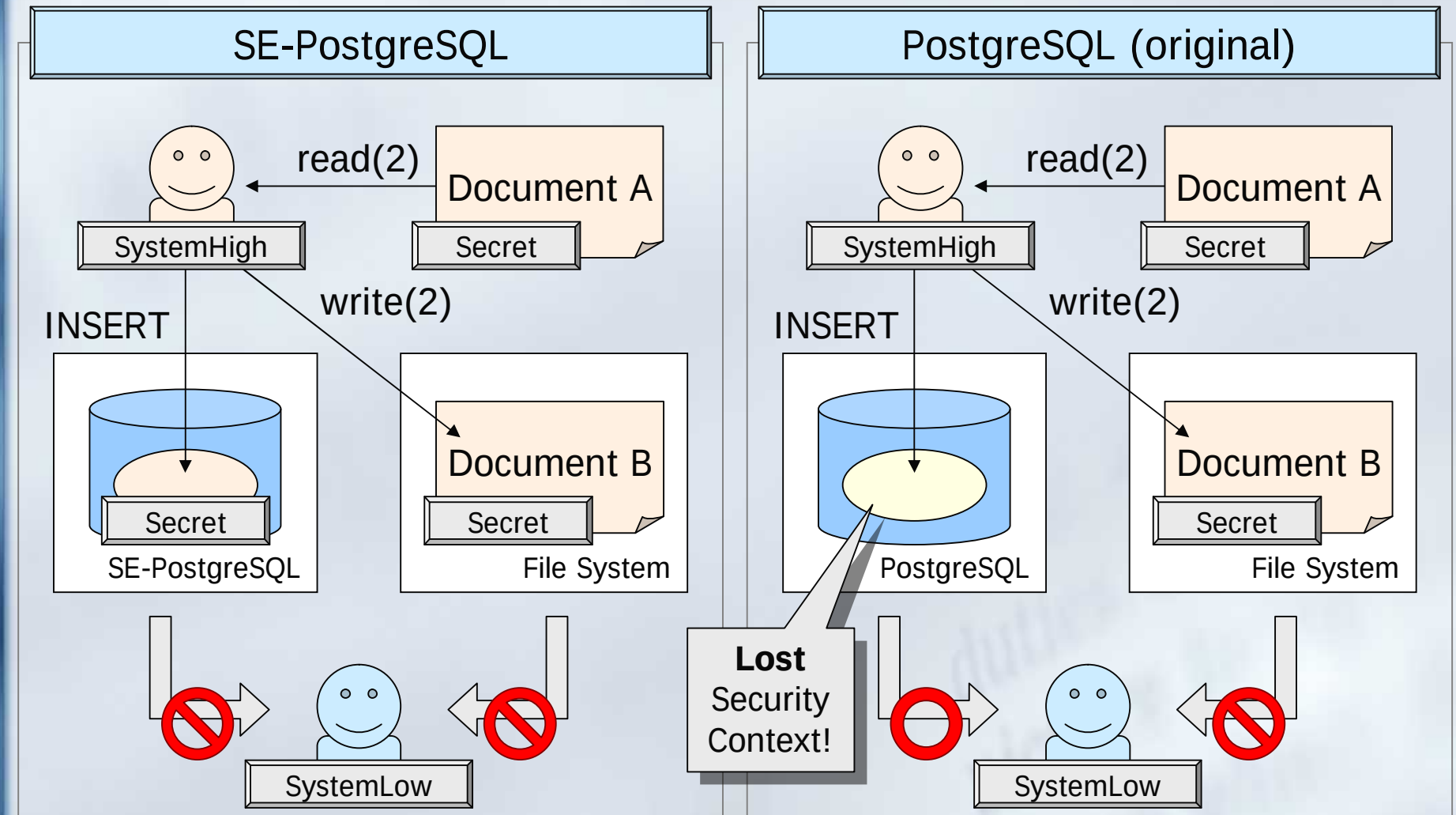
- SE-PostgreSQLは、DBオブジェクトにセキュリティコンテキストを関連付ける。

SELinuxとの統合(3)

- 接続元プロセスのセキュリティコンテキストを取得
  - SELinuxのAPI: getpeercon()
  - UNIXドメインソケット、TCP/IPソケット共に対応
- DB認証に依存せず、OS上と同じ権限での動作

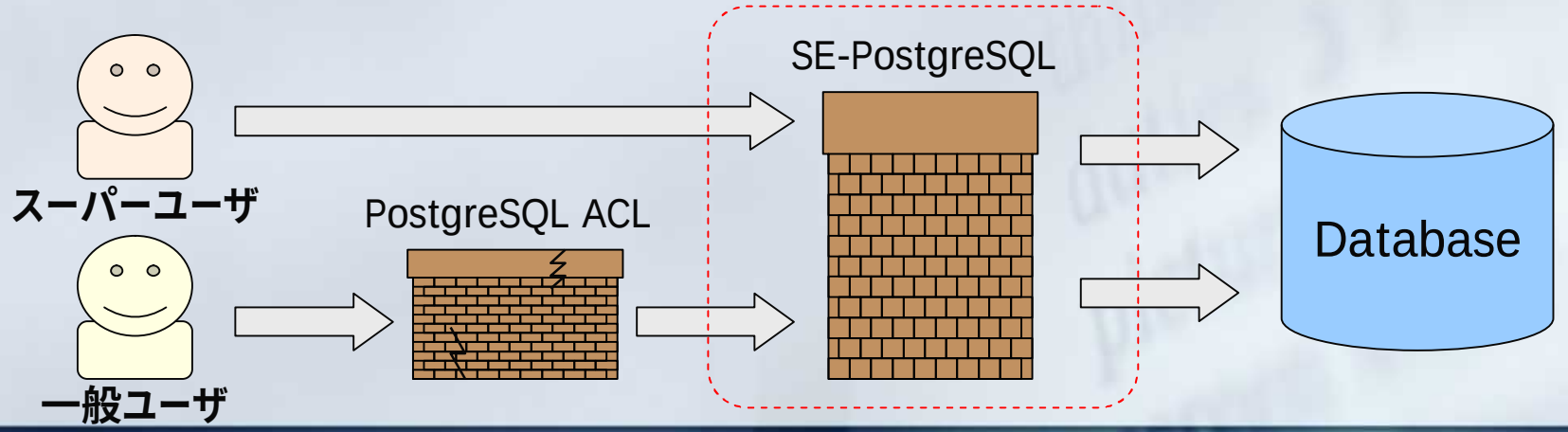


情報フロー制御


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

強制アクセス制御

- PostgreSQLのアクセス制御
  - "スーパーユーザ"はアクセス制御を回避
  - オブジェクト所有者によるACLの設定
- SE-PostgreSQLのアクセス制御
  - 全てのクライアントの、全てのSQLクエリが検査対象
  - セキュリティポリシーによる一元管理



細粒度のアクセス制御

- アクセス制御の対象となるDBオブジェクト

DBオブジェクト	PostgreSQL	SE-PostgreSQL
Database	○	○ (databaseクラス)
Table	○	○ (tableクラス)
Column	×	○ (columnクラス)
Tuple	×	○ (tupleクラス)
Sequence	○	△ (databaseクラス)
Function	○	○ (procedureクラス)
Schema	○	△ (databaseクラス)
Tablespace	○	△ (databaseクラス)
Role	○	△ (databaseクラス)
Large Object	×	○ (blobクラス)

DBオブジェクトへのアクセス制御

- テーブル/カラム/関数etcに対するアクセス制御
  - SQLクエリを検査、アクセスを要求している全てのテーブル/カラムに対してパーミッションを検査
  - セキュリティポリシーに違反すると、トランザクションをアボートし、実行を中止する。
  - SQLクエリの解析フェーズで評価される
- タプルに対するアクセス制御
  - SQLクエリを書き換え、セキュリティポリシーに違反したタプルを検索/更新/削除の範囲に含めない
  - 挿入時、明示的/暗黙的にセキュリティコンテキストを付与、タプルにtuple:{insert}権限がなければ無視される。
  - SQLクエリの実行フェーズで評価される。

Case Study (1)

```
SELECT c1, 1.05 * c2 FROM t1 WHERE c3 < 60;
```

- **t1に対する** table:select **権限**
- **c1, c2, c3 に対する** column:select **権限**
- int4mul()**関数、**int4lt()**関数に対する** procedure:execute**権限**
  - '*'**演算子、**'<'**演算子を実装するSQL関数**
- **対象のタプルに対して** 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**

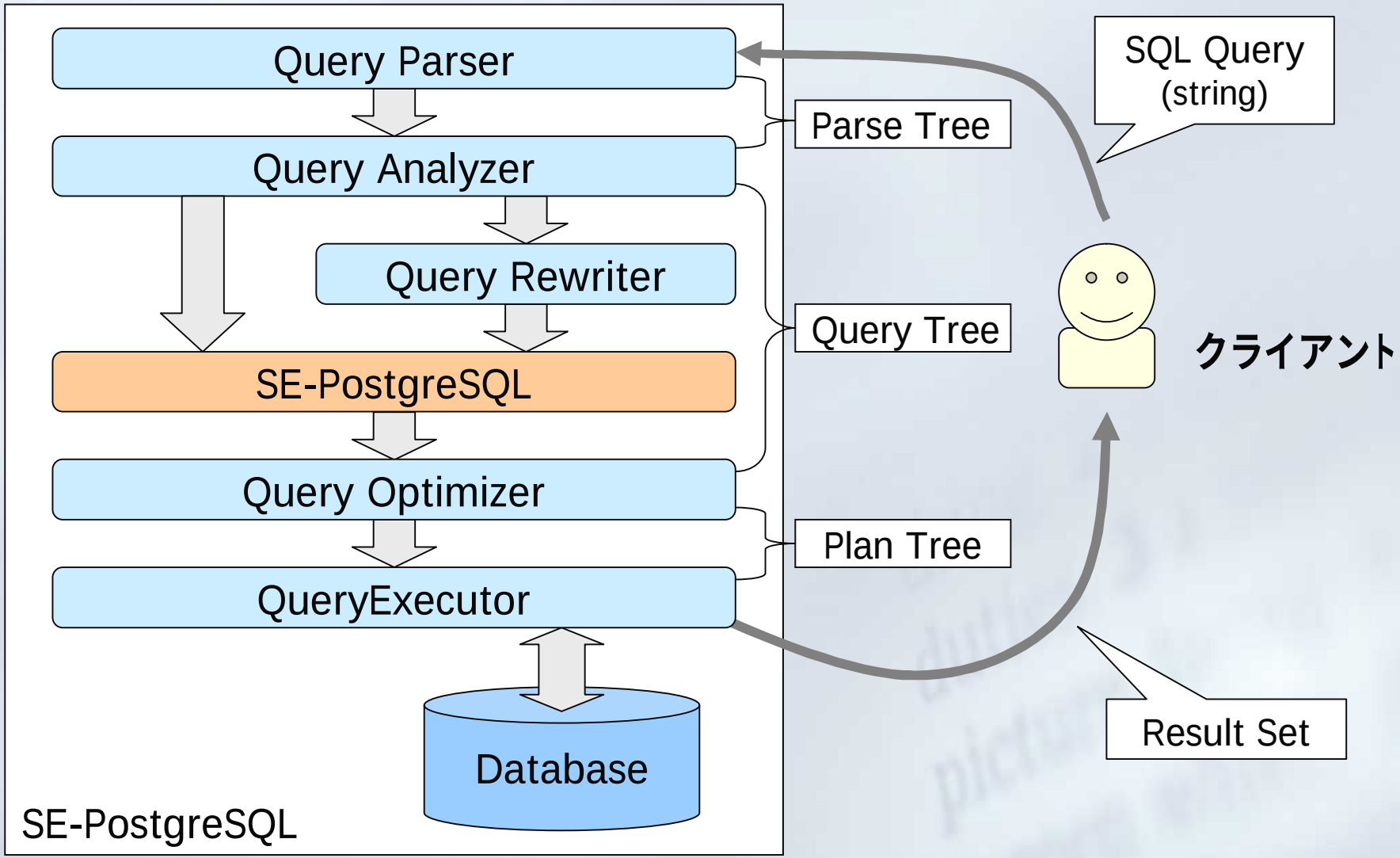
Case Study（2）

```
UPDATE t1 SET c1 = 16, c2 = 1.10 * c2
WHERE c3 = 8;
```

- **t1に対する** table:{select update} **権限**
- **c1に対する** column:update **権限**
- **c2に対する** column:{select update} **権限**
- **c3に対する** column:select**権限**
- int4eq()、int4mul()**関数に対する**procedure:execute**権限**
- **対象のタプルに対して** tuple:{select update}**権限**
  - **権限のないタプルは、更新の対象範囲から除外される。**

SE - PostgreSQLアーキテクチャ

SQL実行フロー


Query Parser
Query Analyzer
Query Rewriter
SE-PostgreSQL
Query Optimizer
QueryExecutor
Database
SE-PostgreSQL
Parse Tree
Query Tree
Plan Tree
SQL Query
(string)
クライアント
Result Set

クエリの書き換え処理

- 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, ...);
- 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, ...)
    and sepgsql_tuple_perms(t2, ...);

特殊なクエリの書き換え処理

- 外部結合(OUTER JOIN)を含むクエリ
  - SELECT * FROM t1 LEFT OUTER JOIN t2 ON t1.x = t2.x;
  - SELECT * FROM
    (SELECT * FROM t1 WHERE
    sepgsql_tuple_perms(t1, ...)) AS t1
    LEFT OUTER JOIN t2 ON t1.x = t2.x
    and sepgsql_tuple_perms(t2, ...);

サブクエリに書き換え

- 左外部結合(LEFT OUTER JOIN)
  - 結果セットに、少なくとも1個の左側テーブルのタプルを含む
  - PostgreSQLではON句によってフィルタリングできない

Case Study（3）

```
SELECT t1.name, t2.did, td.dname FROM
t1 LEFT OUTER JOIN t2 on t1.did = t2.did;
```

- SE-PostgreSQLの場合、JOINの前段階でフィルタリングを実行
- WHERE句につける場合と、OUTER JOINの結果が異なる

t1表

did	name
10	クレア
11	ハチマキ
11	タナベ
12	ハキム

LEFT OUTER JOIN

t2表

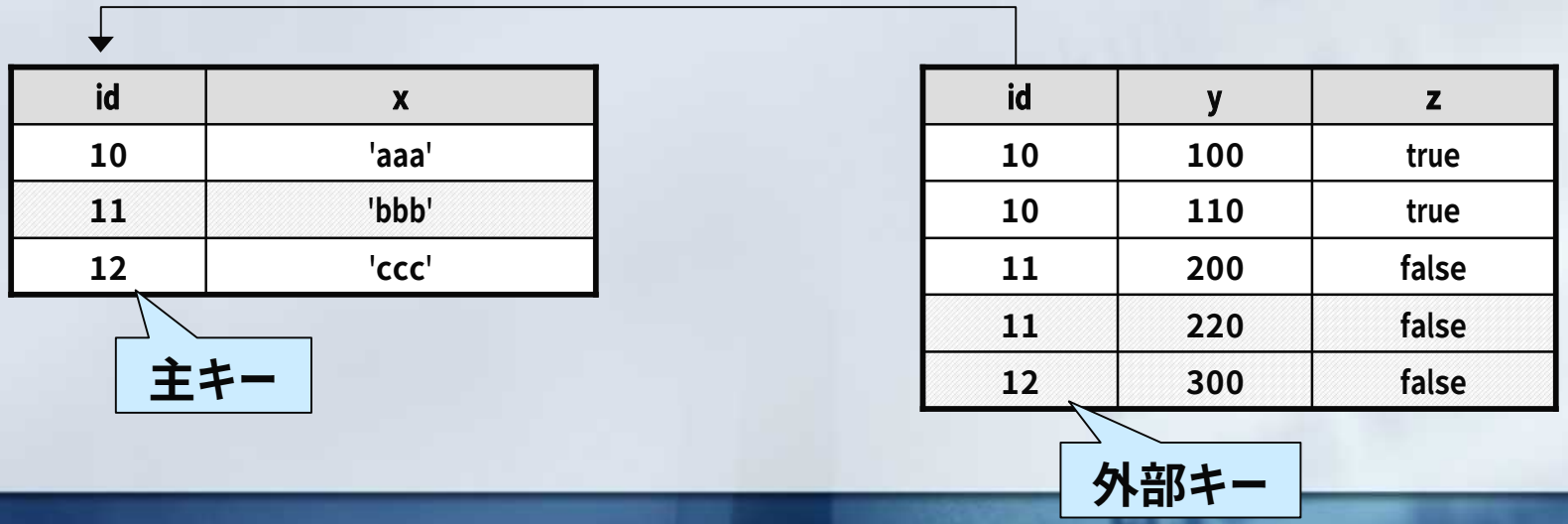
did	dname
10	管制課
11	デブリ課
12	軌道保安庁

name	did	dname
クレア	10	管制課
ハチマキ	11	デブリ課
タナベ	11	デブリ課
ハキム	12	NULL

WHERE句に条件を付けると、行全体がフィルタリングされてしまう。

Case Study(4)

- 外部キー制約
  - 主キー側の検索スコープ
    - 外部キーは、クライアントが参照できる範囲内に主キーが存在しなければならない。
  - ON UPDATE CASCADE ....
    - 参照側の更新を伴う場合、更新される全てのタプルに対して tuple:update 権限が必要。

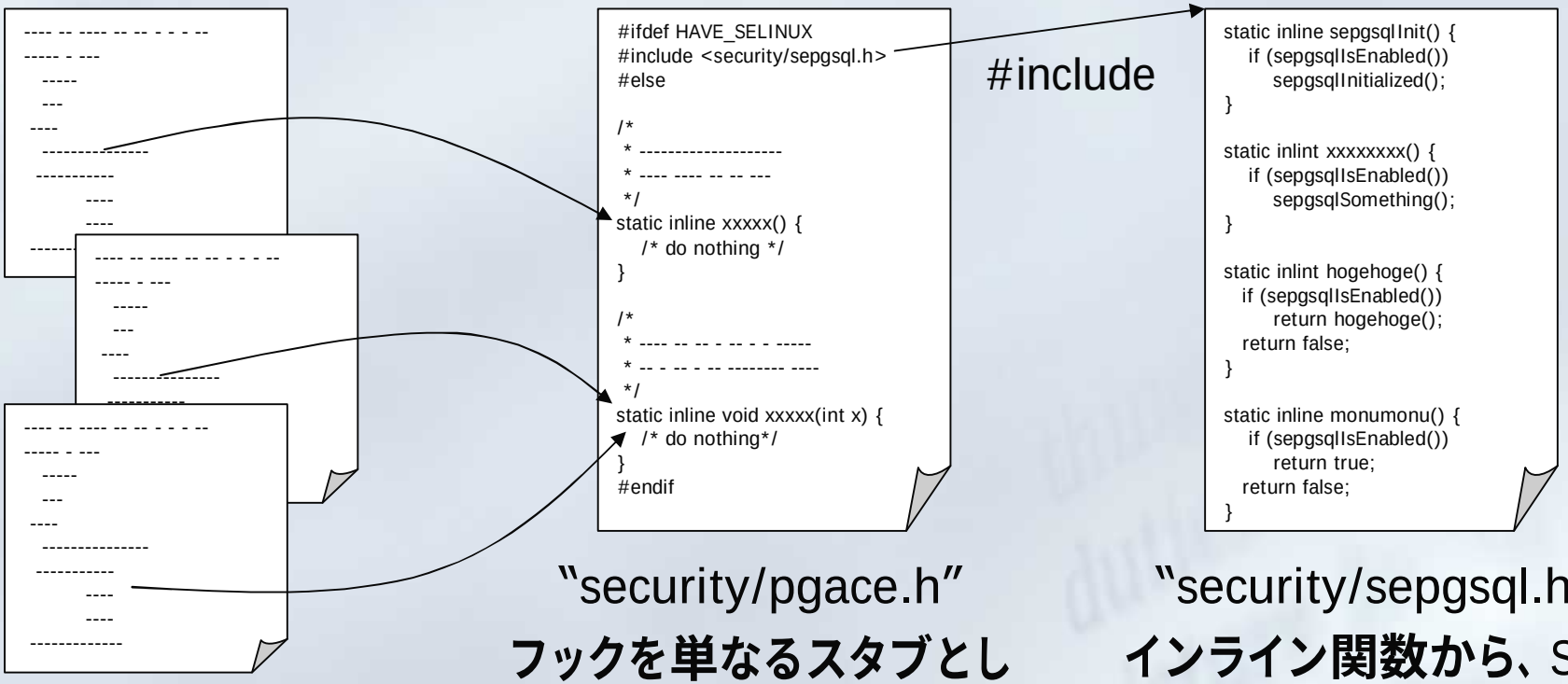


PGACE(1)

- RDBMS/OSセキュリティ統合のための基盤
  - Sun.comの人との議論の中で出たアイデア
  - 将来的には Trusted Solaris もPGACEを利用(？)
    - ✓ PostgreSQLコミュニティに提唱中
- PGACEの提供する枠組み
  - "Strategic Points"におけるフック関数
    - インスタンス初期化時、ExecUpdate()呼出時など
  - タプル ⇔ セキュリティ属性の関連付け
  - セキュリティ属性を参照/更新するシステム列
    - SE-PostgreSQLでは"security_context"列

PGACE(2)

- 実態はスタブの定義 ＋ 仕様のコメント
- #ifdefを使って切り替える



PostgreSQL
ソースコード

"security/pgace.h"

フックを単なるスタブとしてインライン関数定義

"security/sepgsql.h"

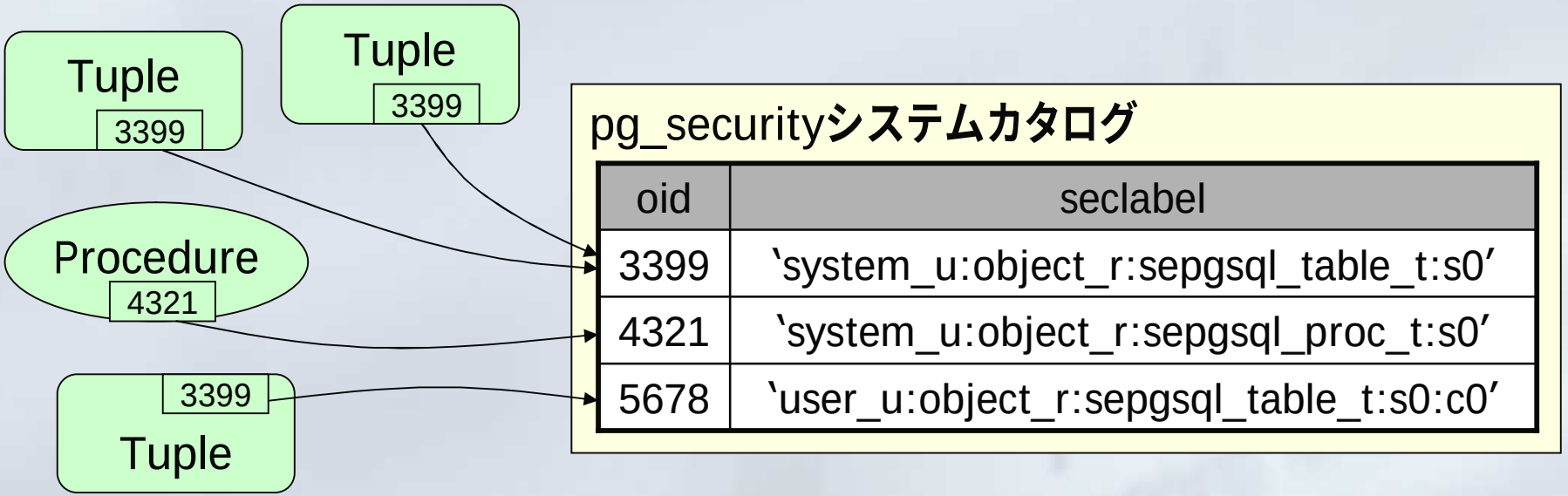
インライン関数から、SE-PostgreSQL実装を呼び出しアクセス制御

タプル ⇔ セキュリティコンテキスト

- タプルの内部表現
  - HeapTupleHeaderData 構造体
  - WALでは xl_heap_header 構造体
    （RDBMSのクラッシュ時にDBを復旧するログ情報）
  - ⇨ これらに、Oid型値を格納する領域を確保
- DBオブジェクト = システムカタログに格納されたタプル
  - テーブル(pg_class)、カラム(pg_attribute)、関数(pg_proc)
  - これらのタプルに関連付けられたセキュリティコンテキスト
    = DBオブジェクトのセキュリティコンテキスト
  - ⇨ 全て同じ方法でセキュリティコンテキストを管理している
- INSERT/UPDATE時のフックによって設定

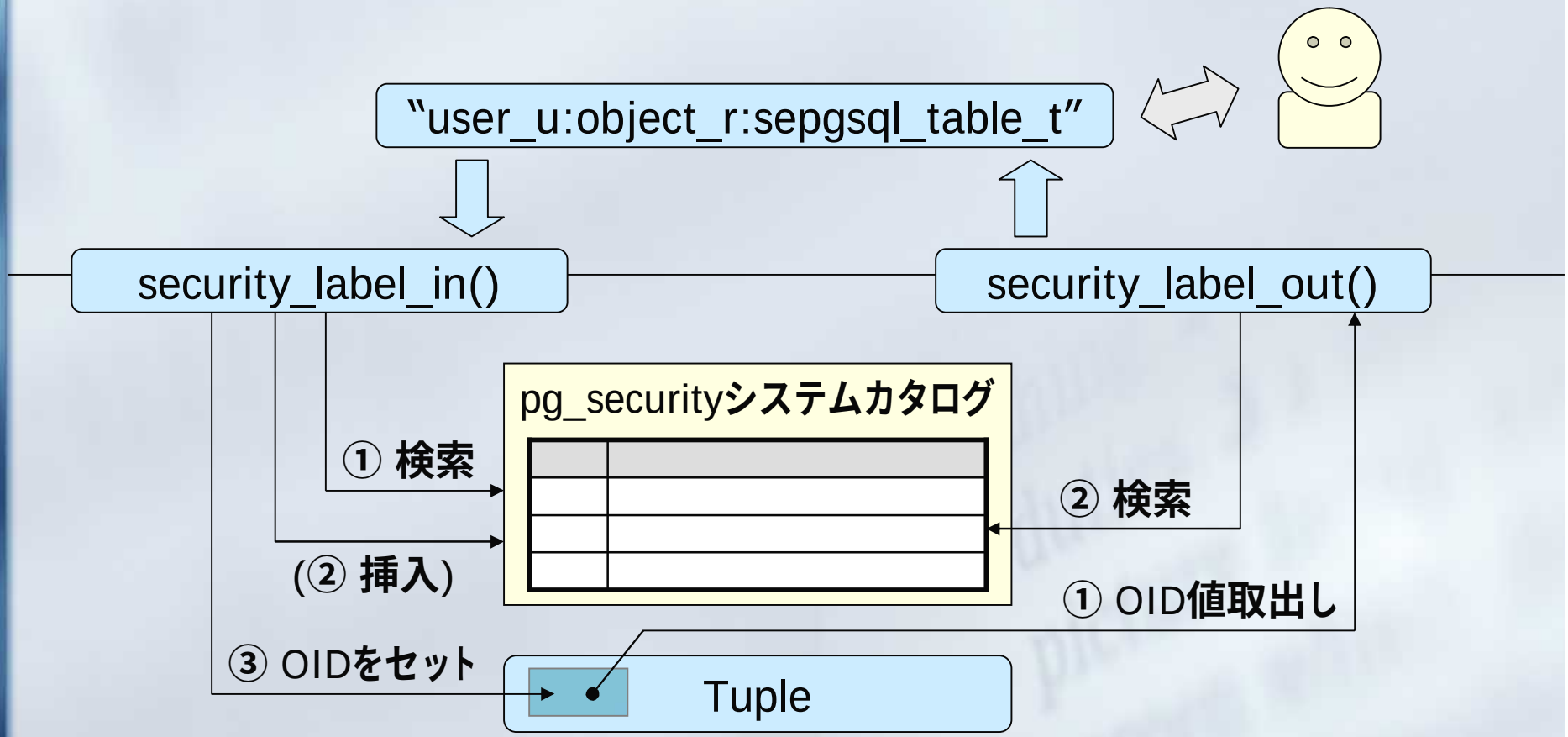
security_label型(1)

- セキュリティコンテキストは文字列表現
  - "system_u:object_r:sepgsql_table_t:SystemHigh"
  - ⇨ 全てのタプルが持つにはサイズが大きい。
- 複数のDBオブジェクトが文字列表現を共有
  - 各DBオブジェクトは、pg_securityのOid値を持つ



security_label型(2)

- ユーザ視点では、文字列で入出力可能に見える
- PostgreSQLのinput/outputハンドラを利用

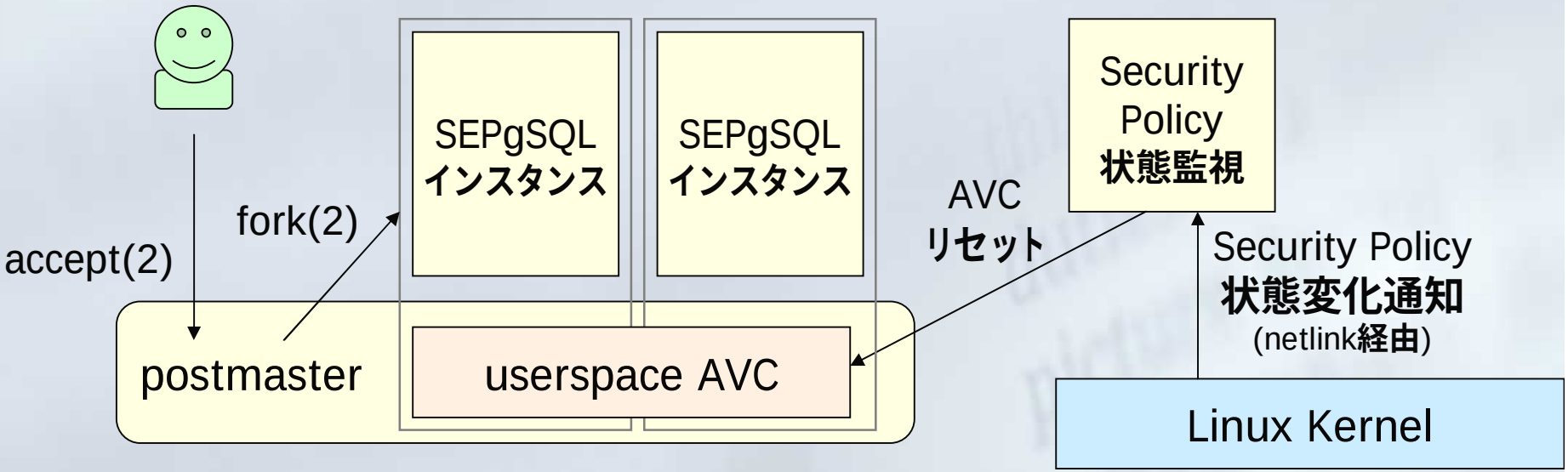


"security_context" システム列

- タプルのセキュリティコンテキストに対応するシステム列
  - security_label型として定義されている
  - セキュリティ属性は、タプルのデータ部ではなく、ヘッダ部分(HeapTupleHeaderData)に格納
- writable system column
  - 通常のシステム列は read onlyだが、UPDATE/INSERTでセキュリティコンテキストを指定するために必要
  - "security_context"システム列は、内部的に"ジャンク"として扱う　(= 値の計算のみ行う)
  - UPDATE/INSERTの直前に計算結果を取り出し、タプルに値をセットする方法で実装。

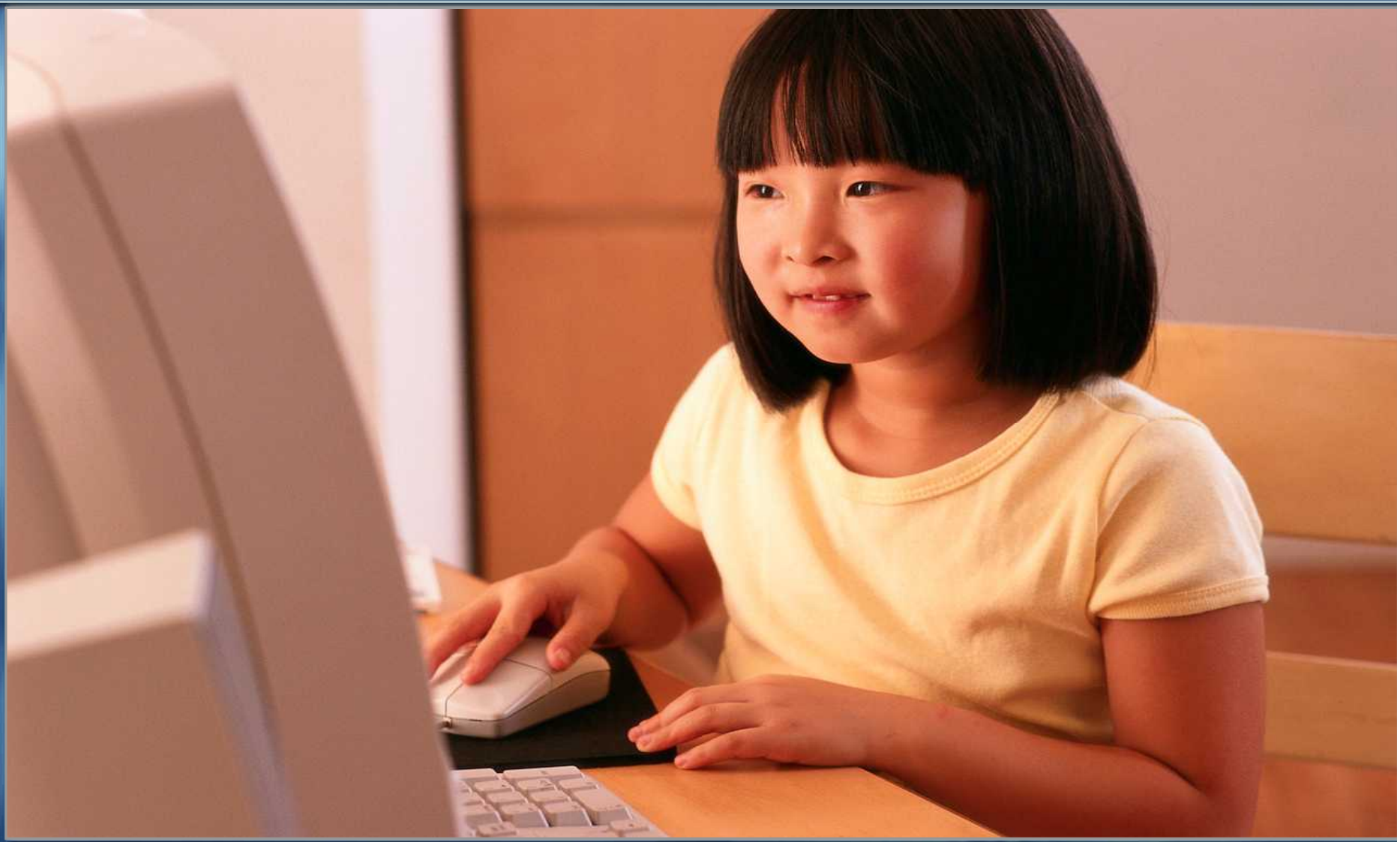
共有メモリ上のUserspace AVC

- AVC = Access Vector Cache
  - 利用頻度の高いセキュリティポリシーをキャッシュ/高速化
- 共有メモリに userspace AVC を配置することで
  - ポリシー再ロード時のInvalidationをアトミックに実行可能
  - カーネル呼び出しの回数を削減できる



デモンストレーション

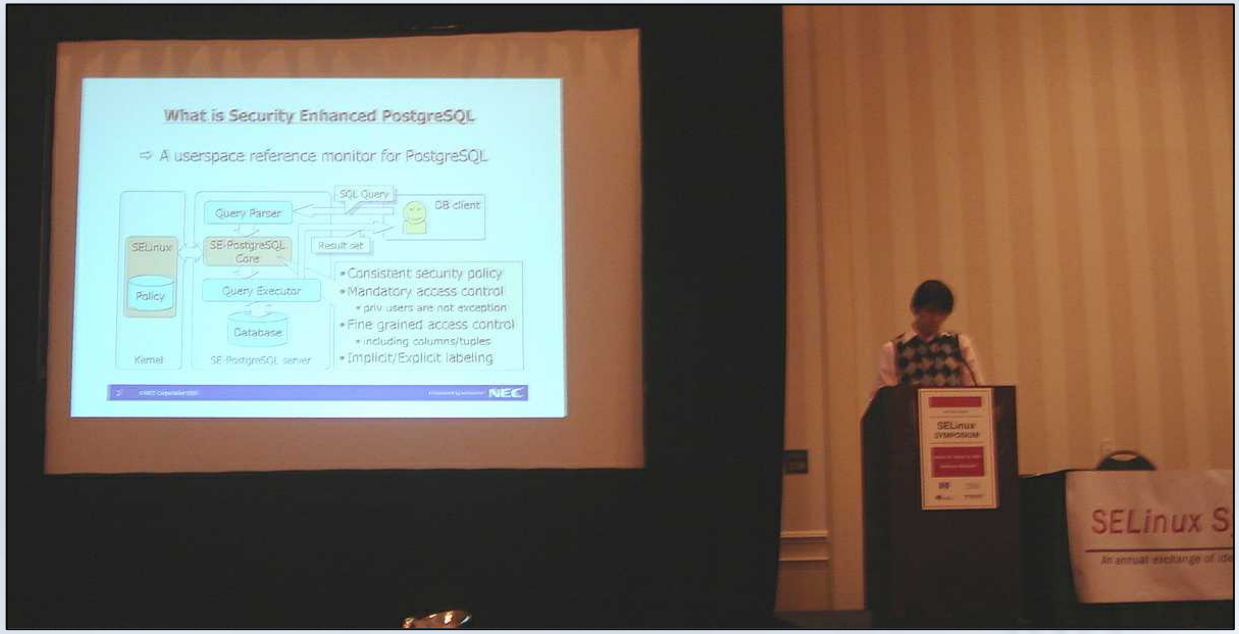
デモンストレーション中



コミュニティでの動向

SELinux Symposium 2007

- 3/14,15 於･ボルチモア
- Works in Progress セッションでの発表



- かなり詳細な質問を多数。高い注目度
  - BLP/Bibaモデルを正しく扱えるのか？SubQueryや集合演算の場合、ISO15408取得は？、、、等々

SELinux Developer Summit

- SE-PostgreSQL関連の問題提起
- initial SID context を取得するインターフェース
  - Invalidになったセキュリティコンテキストを代替するために、"unlabeled"相当のコンテキスト情報が必要
  - /selinux/initial_contexts/* にエクスポート
- Object Class/Permission番号の取得インターフェース
  - Object Class/Permission番号をハードコーディングすると、SELinux対応アプリがカーネルバージョンに強く依存する。
  - /selinux/class/ 以下にエクスポート
- 共有メモリ上の userspace AVC
  - libselinuxでも共有メモリ上のAVCを可能とする方向で合意

PostgreSQL Hackersでの議論

- SE-PostgreSQL α版リリース
  - Trusted Solaris開発者からのコンタクト
  - OS/RDBMSセキュリティの統合へ、共通のフレームワーク
  - ⇨ PGACE(PostgreSQL Access Control Extension)の開発へ
- PGACE/SE-PostgreSQLの提案
  - 細粒度アクセス制御(column)に興味を持つ人
  - DB一貫性とアクセス制御の妥当性について
    - ユニーク制約や外部キー制約
  - 現状、8.3は既に feature freeze になっているので、本格的な議論は 8.3beta 以降にスタートすることに。

SE - PostgreSQL ロードマップ

近い将来

- SE-PostgreSQL 1.0α版　　('07/03上)
  - コミュニティからのフィードバックを得るための叩き台
    - PGACEベースの実装に変更
    - initial SID context、object class番号の実行時取得
- SE-PostgreSQL 1.0β版　　('07/06末予定)
  - Stable1.0 に向けた機能フリーズ版
    - 日本語/英語のドキュメントを公開
    - Strict/Targeted両ポリシーへの対応
    - pg_dump/pg_restoreのSE-PostgreSQL対応
- SE-PostgreSQL 1.0公開　　('07/07末予定)

ちょっと先の話

- 対コミュニティ活動
  - Fedora extra リポジトリへのプッシュ
  - PostgreSQL 本流へのプッシュ
    - 次の次のバージョンあたりを目標に。
- SE-PostgreSQL 2.0　　(時期未定)
  - Polyinstantiationテーブルのサポート
  - 監査ログのOSとの統合
  - pl/pgSQLへの対応

質問タイム

Any Question?

情報源/メディアでの紹介

- **SE-PostgreSQL開発サイト**
  - http://code.google.com/p/sepgsql/
  - **ソースコードの** Subversion **リポジトリ、**RPM/SRPM**、ドキュメント**
- SELinux Symposium 2007/Developer Summi
  - ✓ http://selinux-symposium.org/2007/wipsbofs.php#sepostgresql
  - ✓ http://marc.info/?l=selinux&m=117493229609795&w=2
- **メディアでの紹介**
  - @IT**「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#sepostgresql
  - **IT-pro「SE-PostgreSQL アルファ版リリース」**
  - ✓ 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』の支援を受けて行われています。(2006年度/下期)**