SELinux**ユーザ会 夏の勉強会**2007

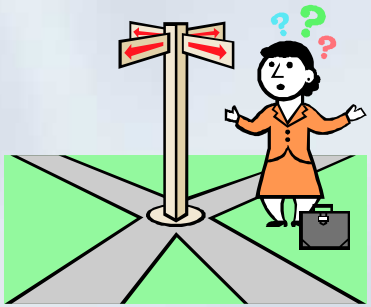
SE-PostgreSQL**開発チーム**
KaiGai Kohei <kaigai@kaigai.gr.jp>

Security-Enhanced PostgreSQL

~As a component of secure system environment~

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

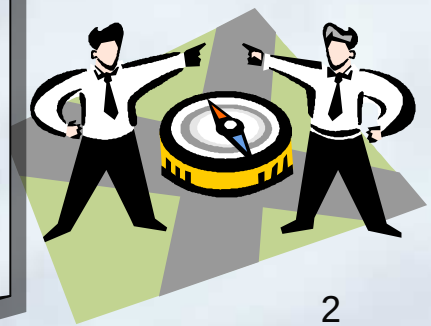


従来のPostgreSQL

- **OSはデータベースアクセス制御に関与できない**
- **OS/RDBMSのセキュリティポリシーは別々**
  - **OSと独立なDB認証**
  - **GRANT/REVOKEで設定するACL**

SE-PostgreSQL

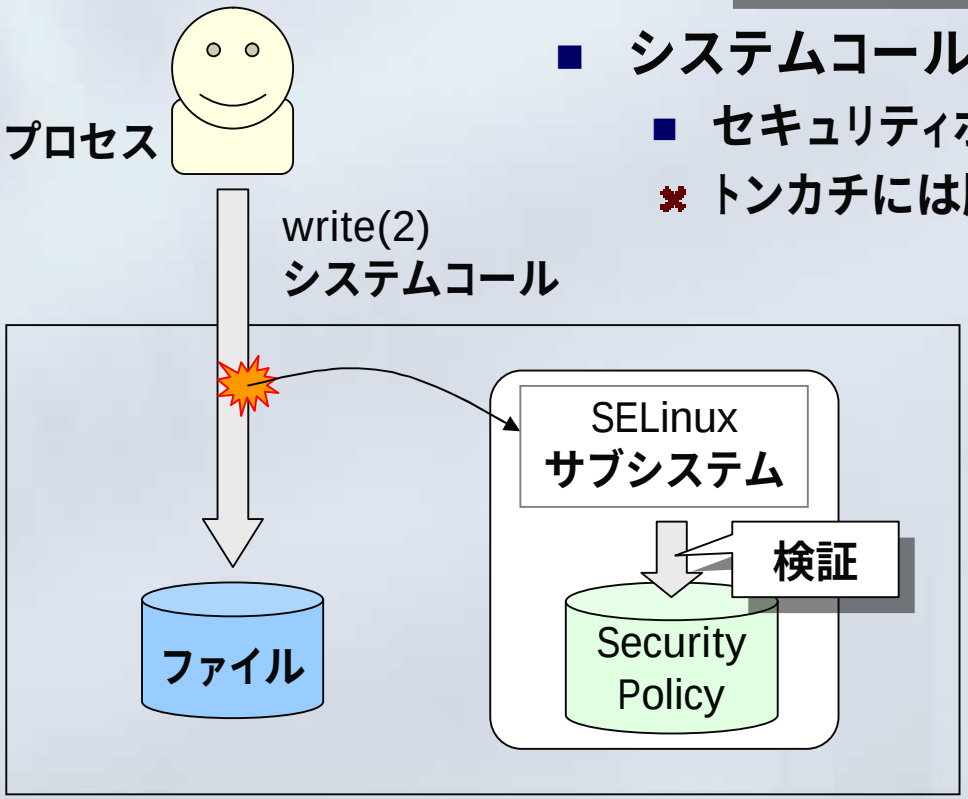
- **システムワイドに一貫したセキュリティポリシー**
  - **OS上と同じ権限でデータベースにアクセス**
  - **DBオブジェクトにもセキュリティコンテキスト**
- **細粒度・強制アクセス制御**



SELinuxのアクセス制御

SELinux ＝ OSのリファレンスモニタ

回避不可能性・耐タンパー性・検証可能性



- システムコール呼出しをフック
  - セキュリティポリシーに基づいて意思決定
  - トンカチには勝てない

システムコールを経由しない攻撃
→ SELinuxの対象外

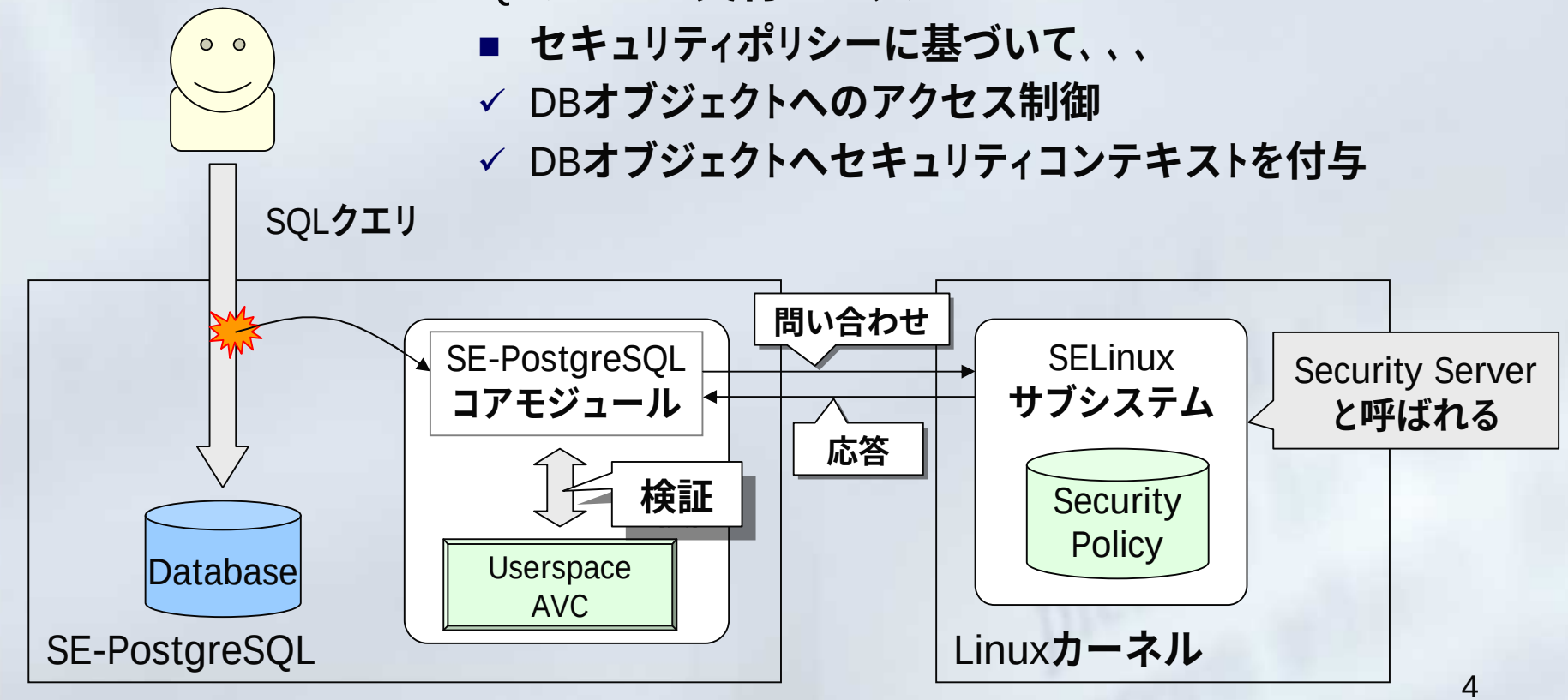
✓ トンカチでサーバを物理破壊

✓ 機密情報をデータベースに格納

SE-PostgreSQLのアクセス制御

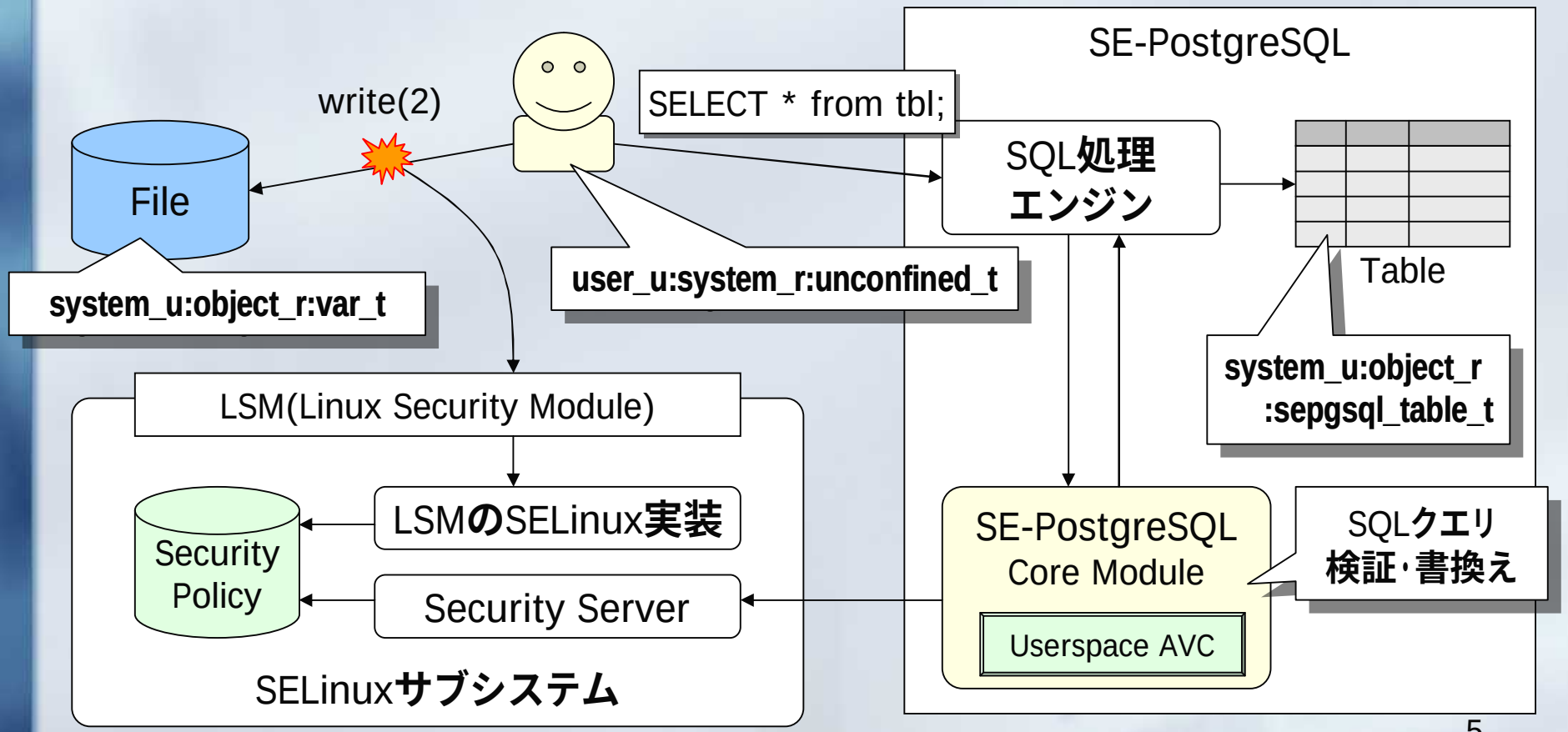
SE-PostgreSQL ＝ SQLのリファレンスモニタ

- SQLクエリの実行をフック
  - セキュリティポリシーに基づいて、、、
  - ✓ DBオブジェクトへのアクセス制御
  - ✓ DBオブジェクトへセキュリティコンテキストを付与

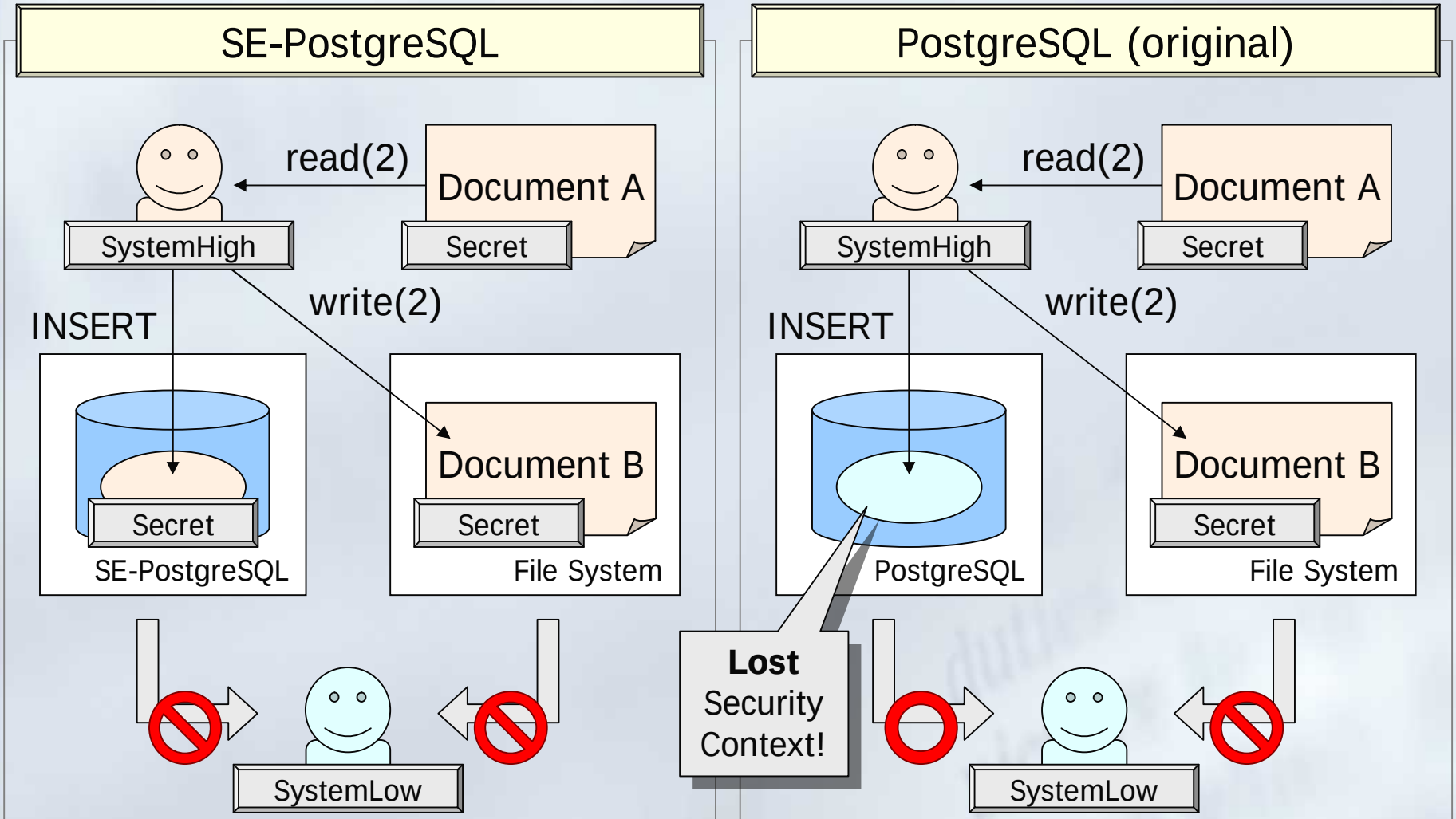


SELinux/SE-PostgreSQL連携

- **プロセスのセキュリティコンテキストをAs-Isで利用**
  例）対ファイル、対DBオブジェクトでも一貫した権限が適用



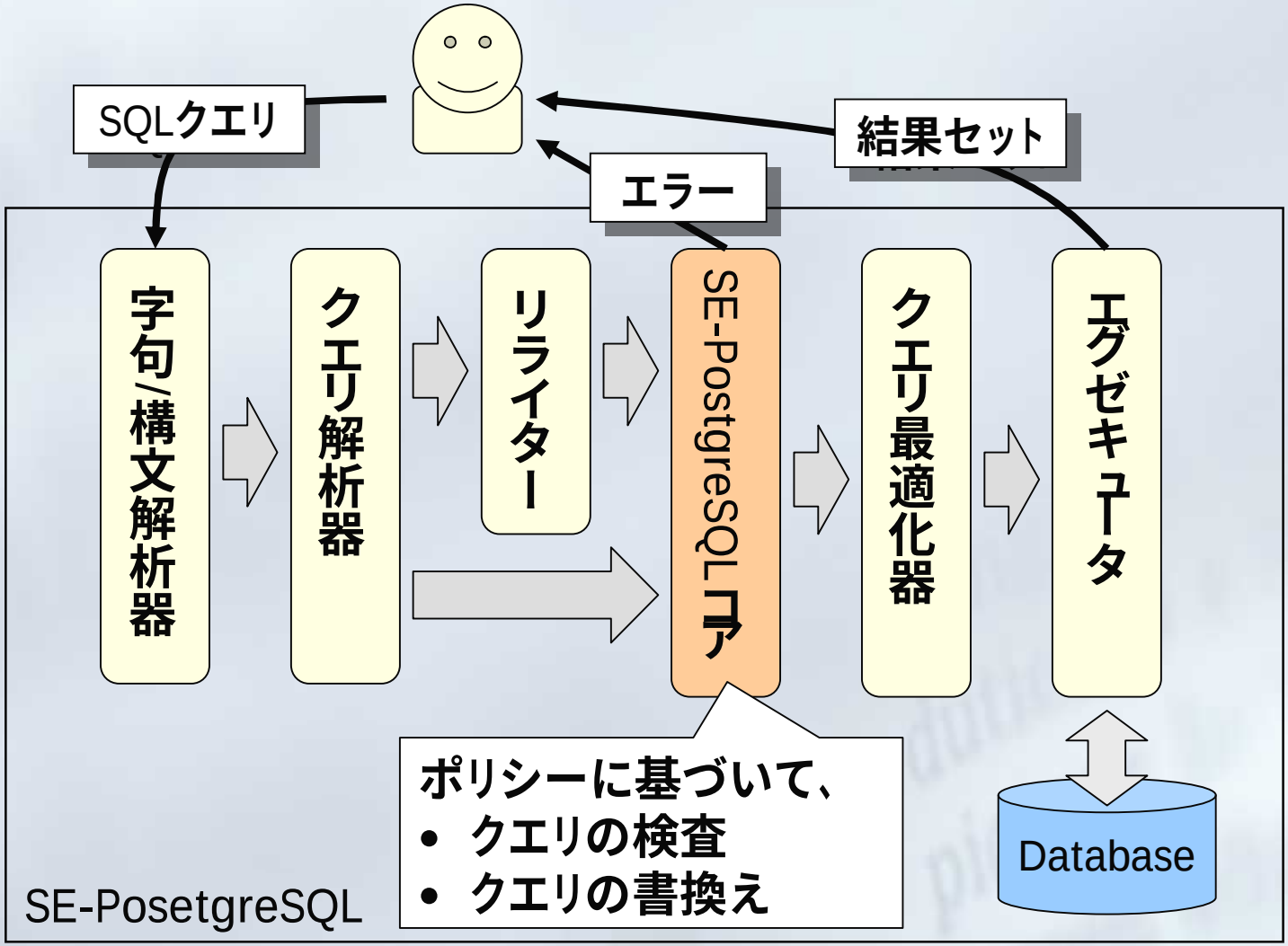
情報フロー制御


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!

細粒度・強制アクセス制御

- 細粒度アクセス制御
  - PostgreSQLのアクセス制御リスト
    - 行レベル/列レベルアクセス制御は存在せず
    - Binary Large Objectへのアクセス制御は存在せず
  - SE-PostgreSQLの利用するオブジェクトクラス
    - Database, Table, Column, Tuple, Procedure, Blob
    - 合計58種類のパーミッションを定義(重複含む)
- 強制アクセス制御
  - PostgreSQLの特権データベースユーザ
    - アクセス制御リストを無視する　(=OSのrootと同じ)
  - SE-PostgreSQL = 全てのSQLクエリを検証
    - ✓ リファレンスモニタとしての回避不可能性

SE-PostgreSQLの実装



Case Study (1)

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

- tbl1**に対する** table:{use select} **権限**
- c1, c2 **に対する** column:{select} **権限**
- c3 **に対する**column:{use}**権限**
- int4pl**関数、**int4gt**関数に対する** procedure:{execute}**権限**
  - **‘+’演算子、’>’演算子を実装している関数**
- **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:{use 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:{use 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:{use} **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:{use select update} **権限**
  - **権限のないタプルは更新の対象から除外**

セキュリティコンテキスト

- ‘security_context’ システム列
  - 各タプルのセキュリティコンテキストを参照できる

```
kaigai=# SELECT security_context, * FROM drink ORDER BY id;
              security_context               | id | name  | price | alcohol
---------------------------------------------+----+-------+-------+---------
 user_u:object_r:sepgsql_table_t:Classified |  1 | coke  |   110 | f
 user_u:object_r:sepgsql_table_t:Classified |  2 | tea   |   120 | f
 user_u:object_r:sepgsql_table_t:Classified |  3 | juice |   150 | f
 user_u:object_r:sepgsql_table_t:Secret     |  4 | water |   120 | f
 user_u:object_r:sepgsql_table_t:Secret     |  5 | wine  |   340 | t
 user_u:object_r:sepgsql_table_t:TopSecret  |  6 | beer  |   240 | t
(6 rows)
```

- システムカタログ(System Catalog)
  - テーブルの情報を格納する pg_class テーブル
  - カラムの情報を格納する pg_attribute テーブル
    - DBオブジェクトは、システムカタログ内のタプルとして管理される
    - これらのタプルのセキュリティコンテキストを利用する

クエリ書換え／行レベルアクセス制御

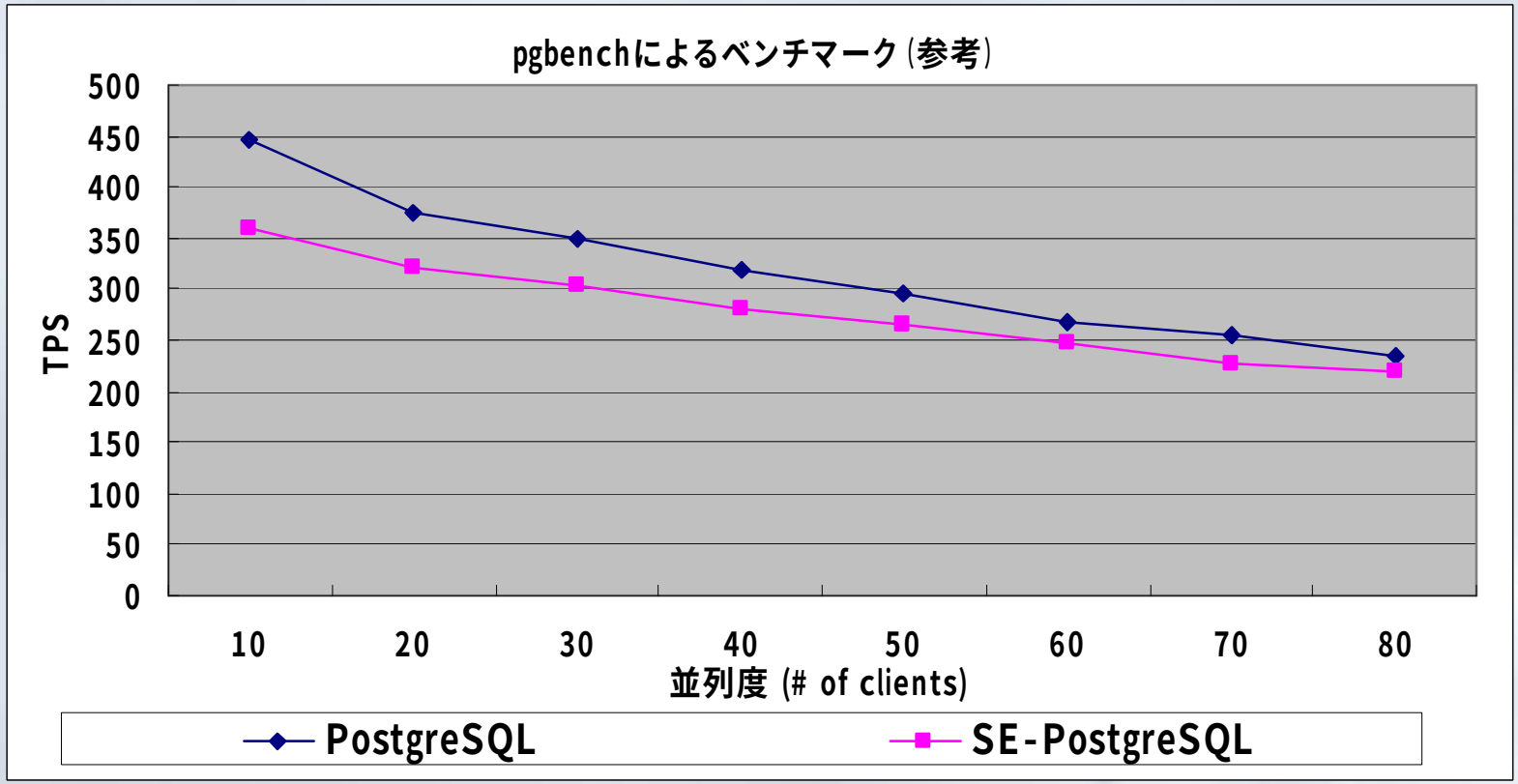
- sepgsql_tuple_perms()関数
  - タプルに対するパーミッションを持っていればtrueを返す
- 一般的なクエリ
  - SELECT * FROM t1 WHERE a > 0;
  - ⇨SELECT * FROM t1 WHERE a > 0
    and sepgsql_tuple_perms(t1.security_context, …);
- INNER JOINを含むクエリ
  - SELECT * FROM t1 JOIN t2 ON t1.a = t2.x;
  - ⇨SELECT * FROM t1 JOIN t2 ON t1.a = t2.x
    and sepgsql_tuple_perms(t1.security_context, …)
    and sepgsql_tuple_perms(t2.security_context, …);
  - ✓ 行レベルでフィルタリングした『後』でJOIN処理を行う

特殊なクエリ書換え処理

- OUTER JOIN を含む問い合わせ
  - SELECT * FROM t1 **LEFT OUTER JOIN** t2 ON t1.a = t2.x;
- OUTER JOIN の性質
  - 被外部結合側テーブル内のタプルを少なくとも一個、結果セットに含む
  - 即ち、単純な条件句書き換えではフィルタリング不可能
- 書き換え処理
  - サブクエリへと展開する
  - SELECT * FROM t1 LEFT OUTER JOIN t2 ON t1.a = t2.x;
  - SELECT * FROM
    (SELECT * FROM t1
        WHERE sepgsql_tuple_perms(t1.security_context,...) AS t1
    LEFT OUTER JOIN t2 ON t1.a = t2.x
        AND sepgsql_tuple_perms(t2.security_context, ...);

パフォーマンス (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

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に対して** 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**

暗黙のセキュリティコンテキスト

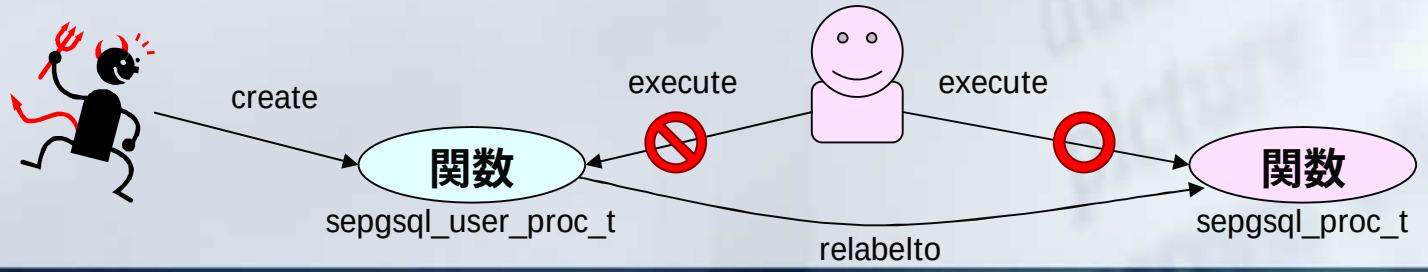
- 上位のDBオブジェクトのタイプを継承
  - databaseクラス ... SE-PostgreSQLドメインを継承
  - table,procedure,blobクラス ... Databaseのタイプを継承
  - column,tupleクラス ... Tableのタイプを継承
- TYPE_TRANSITION（タイプ遷移）ルール

```
TYPE_TRANSITION <subject domain> <target type>
                : <object class> <object type>;
```

  - 特定の組み合わせで、新しいタイプを定義することができる
- 標準のセキュリティポリシー
  - databaseクラス ... sepgsql_db_t タイプ
  - tableクラス ... sepgsql_table_t タイプ
  - procedureクラス ... sepgsql_proc_t, sepgsql_user_proc_t タイプ
  - blobクラス ... sepgsql_blob_t タイプ

ユーザ定義関数

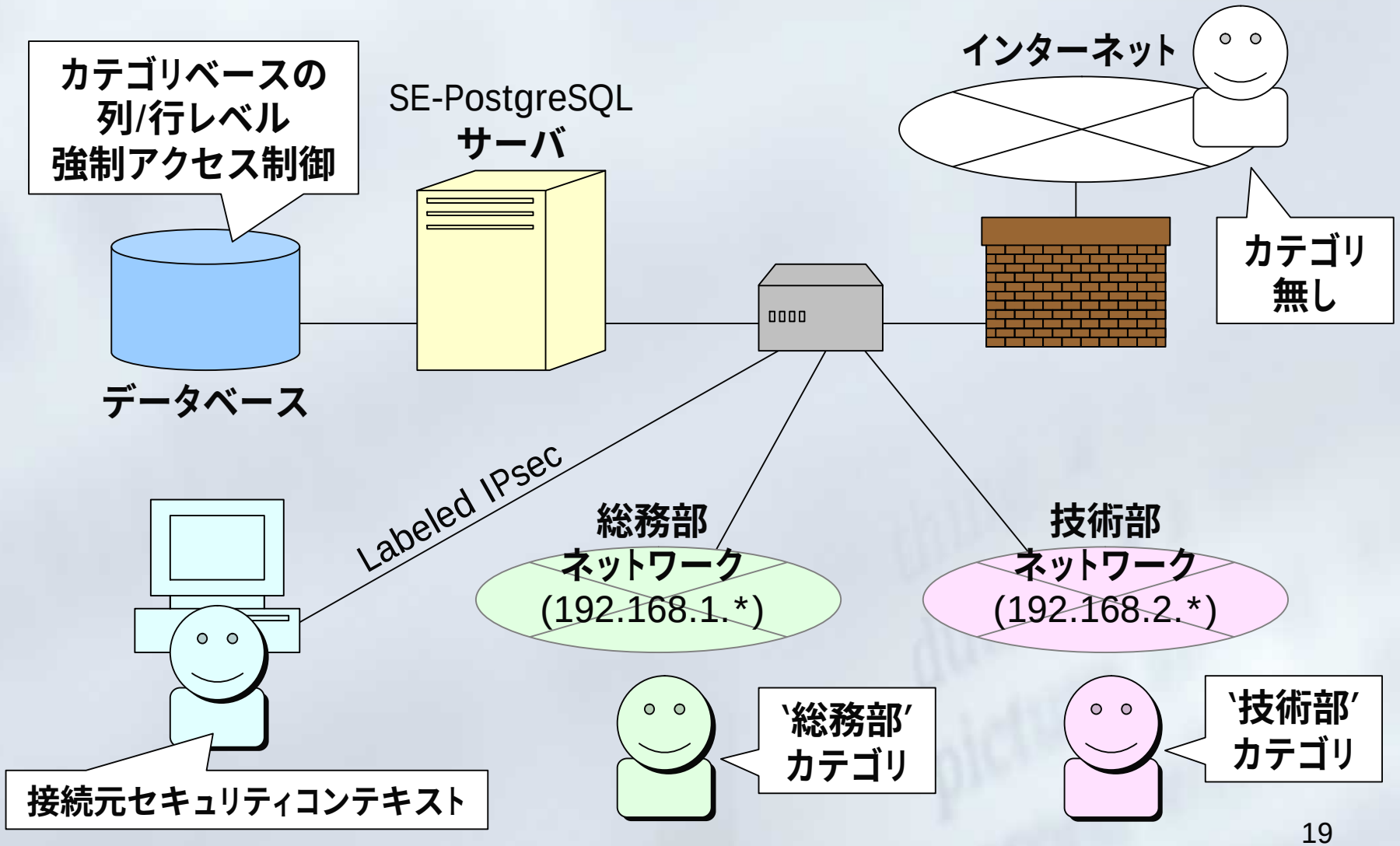
- Administrative Domain
  - unconfined_t, sysadm_t, postgresql_t(**サーバプロセス**)
  - **新しく作成した**SQL**関数には、**sepgsql_proc_t **タイプを付与**
- Generic Domain
  - user_t, staff_t, httpd_t **など**
  - **新しく作成した**SQL**関数には、**sepgsql_user_proc_t **タイプを付与**
- **ポリシー**
  - Administrative Domain**は、**sepgsql_user_proc_t **を実行できない**
  - sepgsql_user_proc_t → sepgsql_proc_t **への変更は可能**
    - **素性のわからない関数を誤って実行することを禁止**



ポリシーのカスタマイズ

- Booleanによるカスタマイズ
  - sepgsql_enable_unconfined (default: on)
    - Administrative Domainを有効化
  - sepgsql_enable_users_ddl (default: on)
    - Generic DomainによるDDL構文の実行を許可
  - sepgsql_enable_auditallow (default: off)
  - sepgsql_enable_auditdeny (default: on)
  - sepgsql_enable_audittuple (default: off)
    - Auditログの出力を制御
- タイプの変更によるカスタマイズ
  - sepgsql_table_t SELECT/INSERT/UPDATE/DELETE可能
  - sepgsql_fixed_table_t SELECT/INSERTのみ許可
  - sepgsql_ro_table_t SELECTのみ許可
  - sepgsql_secret_table_t Generic Domainはアクセス不可

SE-PostgreSQL利用環境



※ 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, pg_dumpall –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

今後の予定

- ’07/07/01 … SE-PostgreSQL1.0βリリース
  - 1.0に向けては機能フリーズとしました。
  - ドキュメントも併せて提供しています
    - The Security Guide of SE-PostgreSQL (日本語/英語)
- SE-PostgreSQL1.0正式版…8月上旬予定
- 対コミュニティ活動
  - PostgreSQLへは8.2.4へ向けて議論
  - Fedora Projectへのパッケージの提供
  - Linux Kernel側の動き
    - Initial Security Context取得のI/F ... 2.6.22でマージ
    - Object Class/Access Vector取得のI/F ... 2.6.23でマージ
    - ✓ SELinux Developer Summitでの議論が反映されました。☺

情報源

- 開発者向け情報源
  - http://code.google.com/p/sepgsql/
  - subversionリポジトリ
    - svn checkout http://sepgsql.googlecode.com/svn/ sepgsql
  - RPMパッケージ
    - Fedora 7/Fedora core 6向けパッケージを提供
  - ドキュメント
    - "The Security Guide of SE-PostgreSQL β"...日本語版/英語版
- 軽く宣伝
  - 9/7(金) 10:00～　於・東京工業大学
  - 未踏ソフトウェア創造事業(2006年下期/千葉PM)成果報告会
    - SELinuxのチュートリアルセッションも予定しています

IPA 未踏ソフトウェア創造事業(2006年度/下期)は、
SE-PostgreSQLの開発を支援しています。

Q&A

Any Question?

ありがとうございました