FUJITSU

shaping tomorrow with you

サポート事例のご紹介
PostgreSQL カンファレンス 2013

2013/11/08

株式会社富士通ソーシアルサイエンスラボラトリ
杉山 貴洋

当社のご紹介

FUJITSU

- 社名
  - 株式会社富士通ソーシアルサイエンスラボラトリ（略称：富士通SSL）
    - http://www.ssl.fujitsu.com/
    - http://www.facebook.com/FujitsuSSL
- 事業所
  - 武蔵小杉本社
  - 関西事業所（大阪）
  - 東海事業所（名古屋）
  - 東海事業所刈谷分室
- 設立
  - 1972/07/12
- 社員数
  - 1,214名（2013/03時点、連結ベース）
- 事業内容
  - SI事業
  - ソリューション事業
    - **PoweredSolution**

Powered Solution

- 関係会社
  - 株式会社SSLパワードサービス

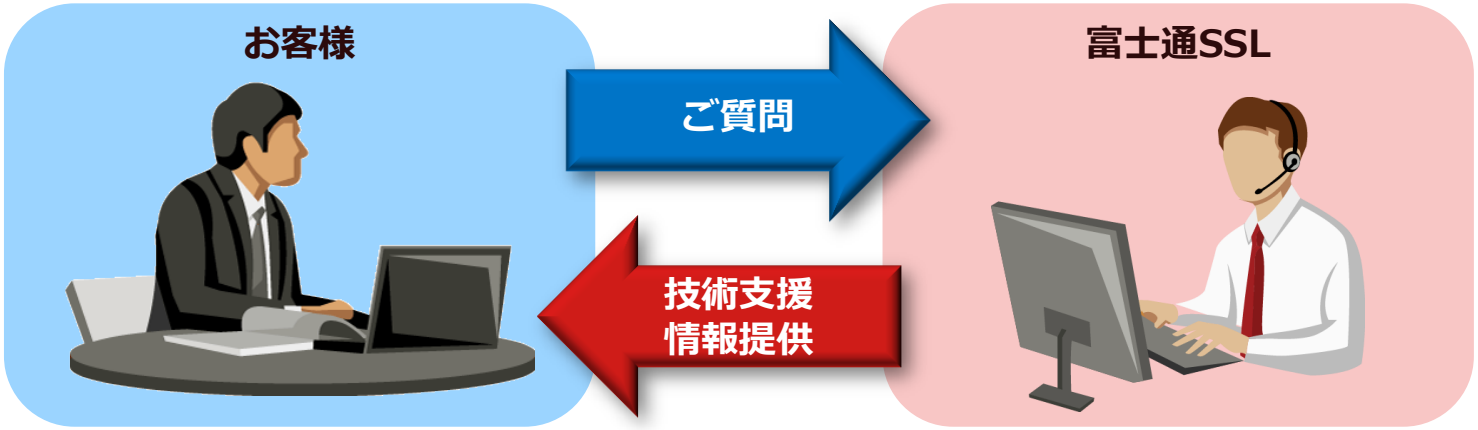
FUJITSU 富士通ソーシアルサイエンスラボラトリ

サービスご紹介（OSSミドルウェアサポート ソリューション）

FUJITSU

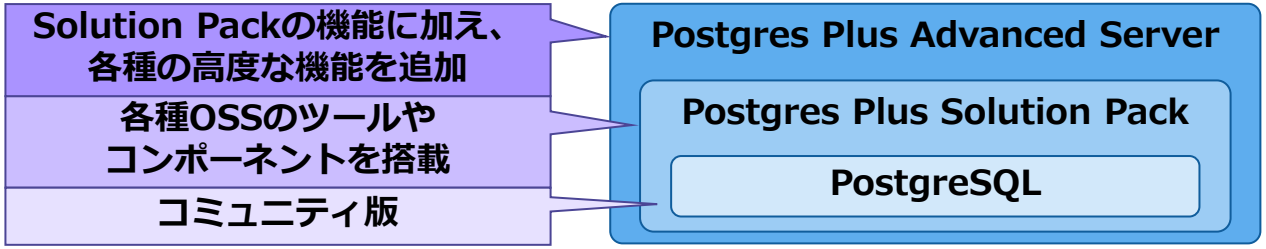
■ OSSミドルウェアサポート・サービス

- お客様にPostgreSQLなどのOSSミドルウェアをご活用いただくために、OSSミドルウェアの導入・運用サポートなどの技術支援や、セキュリティ情報・アップデート情報の提供を行います。



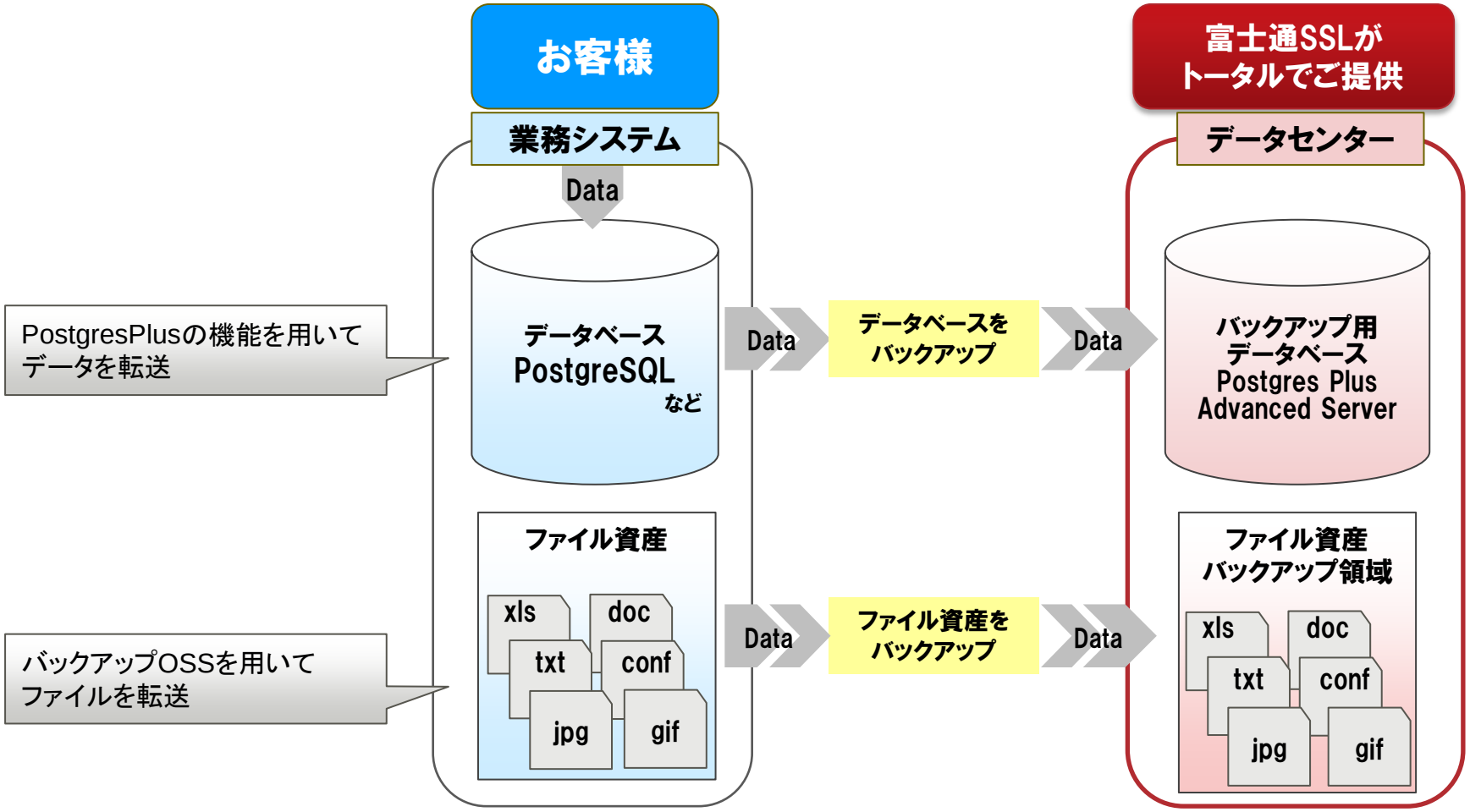
■ Postgres Plus サポート・サービス

- PostgreSQLをベースに開発された「Postgres Plus」のサポートを提供します。
- 一般的な利用において必要となるOSSのツール、コンポーネントなどをまとめた「Solution Pack」と、商用DBとの互換性や性能を重視して機能追加された「Advanced Server」をご用意しております。



Postgres PlusやOSSを利用した遠隔地バックアップサービス

■お客様の重要なデータを遠隔地へ自動バックアップします。



自己紹介

FUJITSU

- 名前
  - 杉山 貴洋
- 略歴
  - 2009/04(入社) - 2009/10
    - 研修、パッケージ開発
  - 2009/11 - 2011/09
    - アプリ開発
  - 2011/10 -
    - OSS調査・検証
    - OSSミドルウェアサポート（PostgreSQL担当）
- 得意分野
  - Web画面構築
    - 弊社開発OSS「Cocuuma*1」の画面開発担当
  - PostgreSQL関連の支援
    - PGECons（PostgreSQL エンタープライズ・コンソーシアム*2）において検証・執筆

*1:https://github.com/cocuuma/Cocuuma
*2:https://www.pgecons.org/

発表の主旨

FUJITSU

- サポート・サービスに寄せられたPostgreSQLに関する質問の中から、よくあるご質問（と、興味深い）を取り上げ、弊社のサポートチームが提示した回答内容をご紹介します。
- 同様の問題が発生した場合は、本発表内容を思い出していただければ幸いです。
  - 似た現象が発生した場合でも、今回の方法で解決しない可能性があることを予めご了承ください。

アジェンダ

FUJITSU

- 冗長構成について
- パラメータ設定について
- 性能劣化①
  - ロングトランザクションによるVACUUM阻害
- 性能劣化②
  - プリペアド文の利用による性能劣化

各事例のタイトル横に、動作環境のバージョンを記載しています。→
バージョンアップによって機能追加等が行われ、状況が変化する場合は、詳細説明において言及しています。

PostgreSQL［バージョン］

質問

- よくある冗長構成と、運用面での要検討事項などがあれば教えてください。

回答

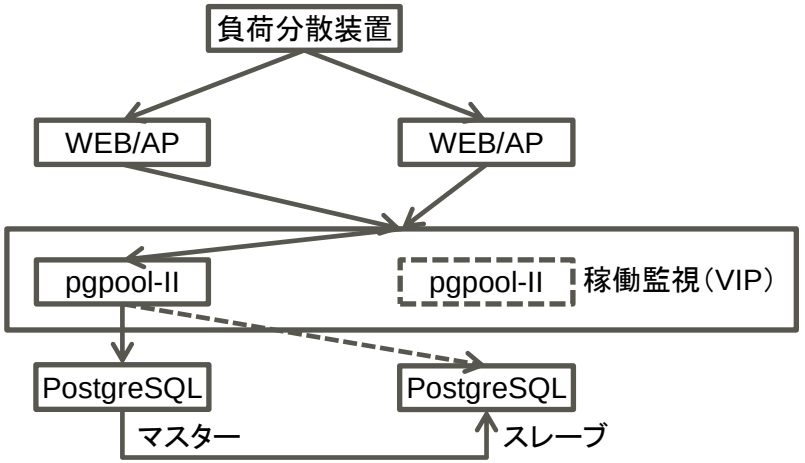
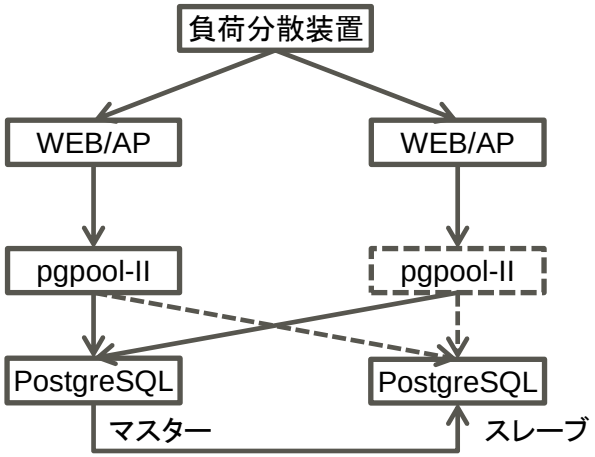
- 通常は、PostgreSQL の Streaming Replication と pgpool-II を組み合わせた構成を提示しています。
- 次頁以降に、冗長構成案と検討事項を記載します。

冗長構成について（回答詳細）

FUJITSU

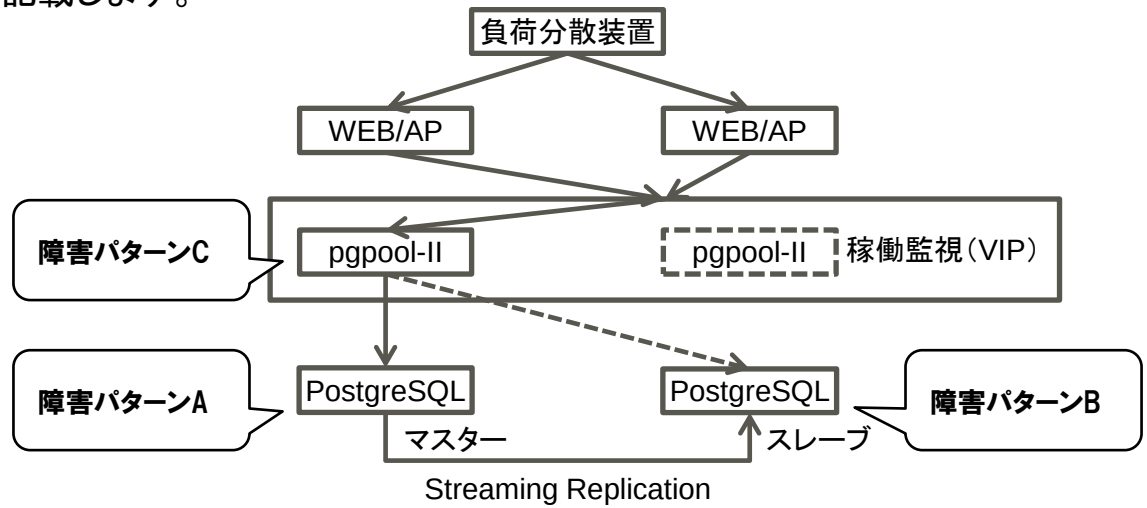
■ 弊社が通常提案する構成は以下の通りです。

- 共有ディスクによるコールドスタンバイ構成など、お客さまの要件に併せて変更させていただく場合があります。
- pgpool-II 3.2で実装された watchdog 機能については、検証中のため標準構成にはしておりませんが、運用サポートは可能です。

構成案	 Streaming Replication	 Streaming Replication
コスト	△ pgpool-IIを監視するソフトウェアにコストが発生	○
可用性	○	× pgpool-IIがSPOFとなる ※当該pgpool-IIを利用しているAP系が停止します。 サービスが停止するわけではありません。

冗長構成について（検討事項）

■ 障害パターンを以下に記載します。

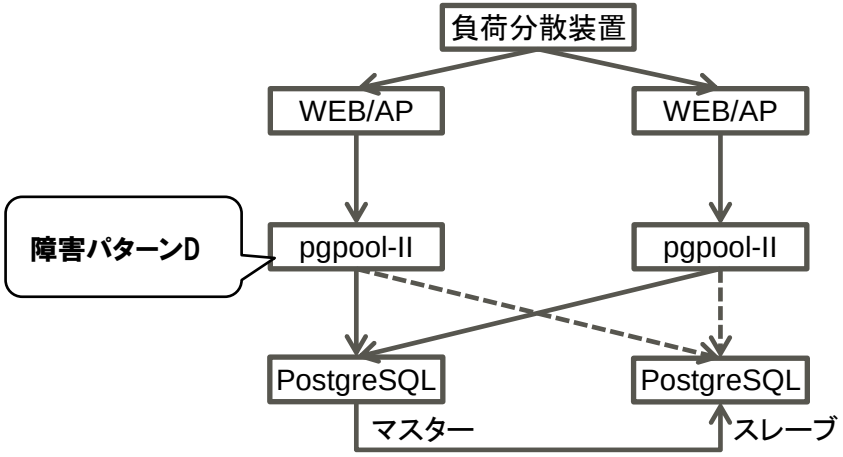


障害パターン	障害パターン概要	イベント順序	イベント概要	自動 / 手動	詳細
A	PostgreSQLのマスターが失陥した場合	1	pgpool-IIがマスタの障害を検知する	自動	ー
		2	pgpool-IIがフェイルオーバを実行する	自動	ー
		3	旧マスタをスレーブとして追加する	手動	ー
B	PostgreSQLのスレーブが失陥した場合	1	pgpool-IIがスレーブの障害を検知する	自動	ー
		2	pgpool-IIがスレーブを切り離す	自動	p.11
		3	再度スレーブとして追加する	手動	ー
C	pgpool-IIが失陥した場合	1	稼働監視ソフトがpgpool-IIの障害を検知する	自動	ー
		2	稼働監視ソフトが待機系のpgpool-IIを現用に切り替える	自動	ー

冗長構成について（検討事項）

FUJITSU

■ 障害パターンを以下に記載します。（PostgreSQLについては前頁と同様です。）



Streaming Replication

障害パターン	障害パターン概要	イベント順序	イベント概要	自動 / 手動	詳細
D	pgpool-IIが失陥した場合	1	クエリのエラーによってpgpool-IIの障害が発覚する	ー	ー
		2	障害発生したpgpool-IIを利用しているシステムが停止する	ー	ー

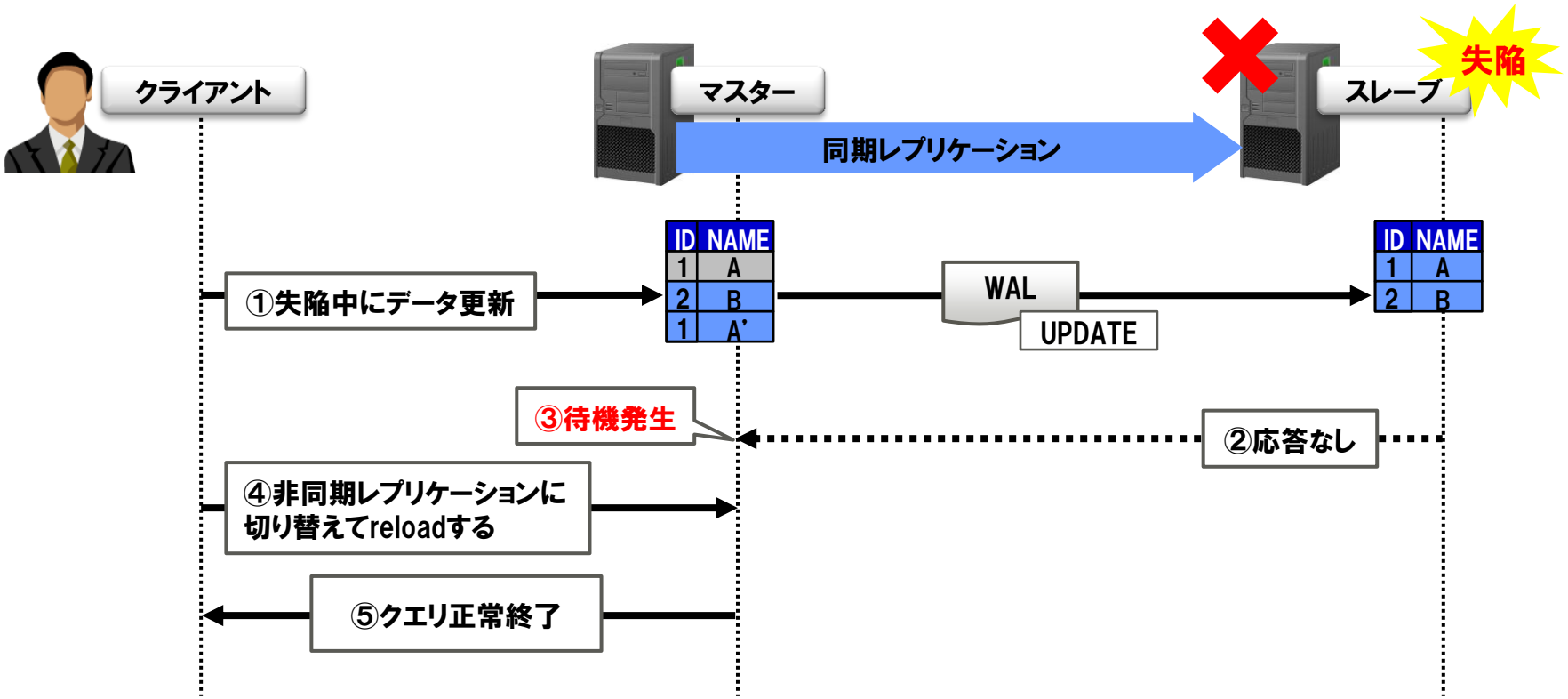
冗長構成について（検討事項詳細）

FUJITSU

■ 前述した冗長構成には、以下の様に検討が必要な事項があるため、併せてお客様へ提示しています。

■ 同期レプリケーションの場合は、スレーブが失陥するとシステムへの更新命令は応答が返ってこなくなるので注意が必要です。

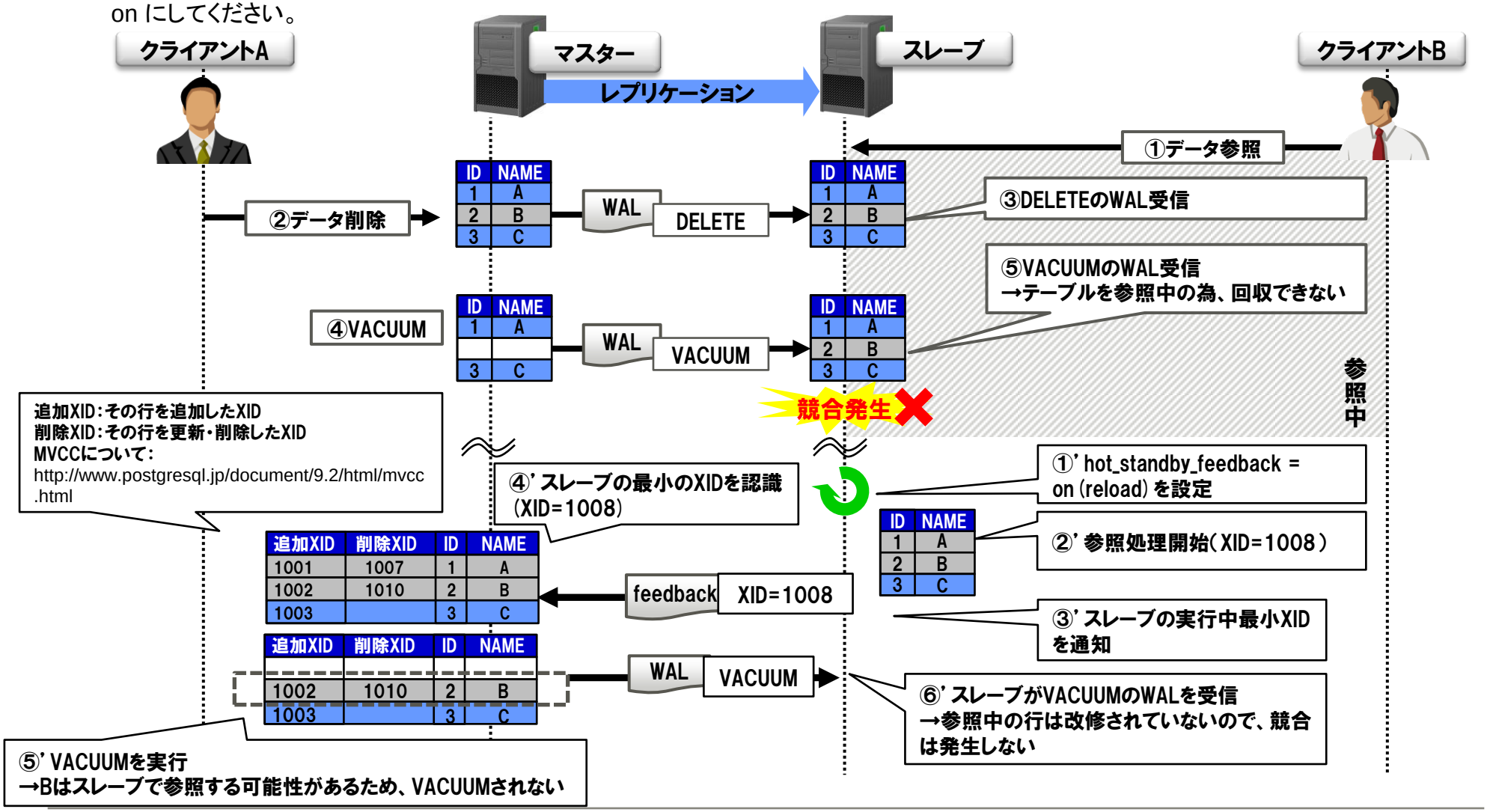
- スレーブの失陥を検知した場合は、synchronous_standby_names をコメントアウトして reload する必要があります。
- このとき、待機させられていたクエリは reload 後に正常終了します。



冗長構成について（検討事項詳細）

FUJITSU

- スレーブでの参照とマスタでのVACUUM処理で競合が発生する可能性があるので、参照分散する場合は、hot_standby_feedback を on にしてください。



冗長構成について（検討事項詳細）

FUJITSU

- ロングトランザクションの発生などで、PostgreSQL のプロセスを削除したい事象が発生する可能性がありますが、pgpool-II によるフェイルオーバを有効にしている場合、PostgreSQL のプロセスは直接削除せず、pgpool-II プロセスにマッピングしてから削除してください。
- 手順を以下に記載します。

項番	作業概要	実施コマンド / 作業
1	クエリのキャンセルを実行する。 クエリが実行中だった場合は、本処理でロングトランションは解消され、問題は解決されるはずです。	=# SELECT pg_cancel_backend([PostgreSQLのPID])
2	コネクションプールの状態を表示する	$ psql –h [pgpool のホスト] –p [pgpool のポート] –c “show pool_pools”
3	pgpool-II の PID と、PostgreSQL の PID を記録する	項番1の結果中の pool_backendpid 列に削除したいPostgreSQL のPIDが表示されていることを確認し、同行のpool_pidを記録する。
4	pgpool-II の プロセスを削除する	$ kill -9 [pool_pid]

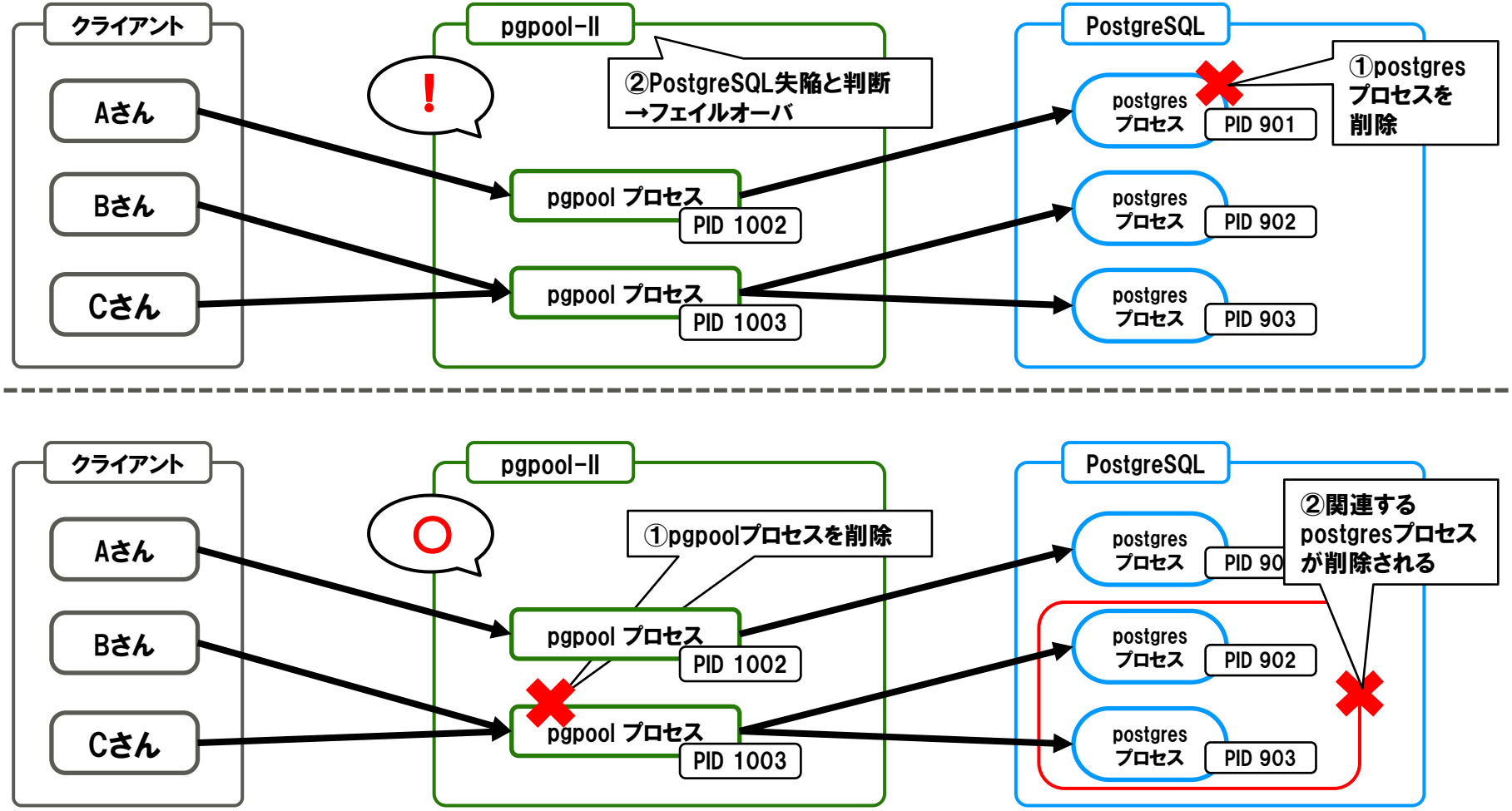
- （注意）一つのpgpool-II プロセスで複数の postgres プロセスを管理することが可能なため、上記項番3の実行によって削除されるPostgreSQL の PID が複数ある場合は、全てにおいて削除されても問題ないか確認を行ってください。

- 次頁にpgpool-II と PostgreSQL のコネクションプールに関する概念図と、PostgreSQL のプロセスを削除するための具体的な方法を記載します。

冗長構成について（検討事項詳細）

FUJITSU

- PostgreSQL のプロセスを直接削除してはいけない理由は、下記のように、pgpool-II が当該 PostgreSQL を失陥したとみなし、フェイルオーバを実行してしまうためです。



質問

- postgresql.conf に設定方針がわからないパラメータがあります。

回答

- postgresql.conf には様々な設定項目が用意されていますが、お客様が運用において質問されるのは、メモリとログに関するパラメータ設定が大部分を占めます。
- 弊社では、次頁にあるようなパラメータについて、ご紹介することが多いです。

パラメータ設定について（回答詳細）

FUJITSU

■ 設定パラメータの一部と、設定方針を以下に記載します。

項番	パラメータ	概要	設定方針	備考
1	shared_buffers	テーブルやインデックス読み込む共有メモリバッファを設定します。データアクセスはこの領域を通して行われます。	データベース内のすべてのテーブル、インデックスが shared_buffers 上に載るように設定します。 もしくは、実行される頻度が高いSQLを選択し、そのSQLがアクセスするテーブル、インデックスが shared_buffers 上に載るように設定します。	Linuxでは使用されていないメモリ領域をディスクI/Oのバッファキャッシュとして使用するため、shared_buffers には物理メモリの1/4くらいまでを上限に設定します。大きくしすぎると、OSのキャッシュ領域を圧迫して性能が低下する場合があります。
2	work_mem	ソート処理やハッシュテーブル操作時に利用するメモリサイズを設定します。	work_mem が不足するとディスク上に一時ファイルを作成してデータ操作を行うため、可能な限り一時ファイルが作成されないように設定します。	一時ファイルの作成状況確認について、詳細を p.17, 18 に記載します。
3	log_line_prefix	ログ出力時の接頭情報を設定します。	次の設定を推奨しています。 '[%m][%d][%h][%u][%e][%p] ‘ %m：タイムスタンプ %d：データベース名 %h：ホスト名またはIPアドレス %u：ユーザ名 %e：SQLSTATEエラーコード %p：PID	ー
4	log_min_duration_statement	SQLの実行完了時に設定値以上経過していた場合はログに出力します。	SQLの実行時間の想定最大値を設定します。（バッチ処理を除く）	ー

パラメータ設定について（work_mem）

FUJITSU

■work_mem設定値の算出方法について①

■ 一時ファイルが作成されてしまっているかどうかは、以下の二通りの方法で確認が可能です。

- log_temp_files = 0

postgresql.conf のlog_temp_filesに 0 を設定し、reload することで、一時ファイルが作成された際に以下の様なログが出力されます。

```
LOG:  temporary file: path "base/pgsql_tmp/pgsql_tmp21374.0", size 88064000
STATEMENT: SELECT aid,bid,abalance FROM accounts ORDER BY abalance DESC;
```

- EXPLAIN ANALYZE

怪しいクエリに目星がついている場合は、以下のようにEXPLAIN ANALYZE を実行することで、一時ファイルの作成を確認できます。

※実際に実行されるので、更新処理に対してEXPLAIN ANALYEを実行する場合は、トランザクションブロック内で実行した後、
ROLLBACK を実行してください。

②ソート処理を実行

①ディスク上で処理

③処理に時間を要する

```
=# EXPLAIN ANALYZE SELECT aid,bid,abalance FROM accounts ORDER BY abalance DESC;

                                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------
Sort  (cost=750622.37..760622.37 rows=4000000 width=12)
(actual time=5755.605..6798.813 rows=4000000 loops=1)
   Sort Key: abalance
   Sort Method: external merge  Disk: 86048kB
   ->  Seq Scan on accounts  (cost=0.00..106905.00 rows=4000000 width=12)
(actual  time=0.106..935.287 rows=4000000 loops=1)
Total runtime: 7055.577 ms
(5 rows)
```

パラメータ設定について（work_mem）

FUJITSU

■work_mem設定値の算出方法について②

■ 以下にwork_memを増加させて一時ファイルが作成されなくなった例を記載します。

- 本値はpostgresql.conf に直接記載することなくSETによる反映が可能です。
- 本値は最大で接続数毎に確保されるため、増加させすぎるとスワップの原因となる可能性があります。

①work_memを増加

③ソート処理を実行

②メモリ上で処理

④処理時間が短縮できる

```
=# SET work_mem='300MB';
=# EXPLAIN ANALYZE SELECT aid,bid,abalance FROM accounts ORDER BY abalance DESC;

                                                        QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
Sort  (cost=545536.37..555536.37 rows=4000000 width=12)
(actual time=2038.960..2346.492 rows=4000000 loops=1)
  Sort Key: abalance
   Sort Method: quicksort  Memory: 285805kB
   ->  Seq Scan on accounts  (cost=0.00..106905.00 rows=4000000 width=12) (actual time=0.034..1011.904
rows=4000000 loops=1)

 Total runtime: 2588.906 ms
(5 rows)
```

■質問

- ■運用後（または、再起動後）しばらくすると、応答速度が遅いクエリが発生します。原因と対策をご教示ください。

■回答

- ■動作環境を調査した結果、ロングトランザクションが原因となり、自動VACUUM がうまく機能していないことが判明しました。
- ■その結果、不要領域の増加を抑止できず、性能劣化が発生していました。
- ■原因と対策の詳細を以降に記載します。

性能劣化① ロングトランザクションによるVACUUM阻害（回答詳細）

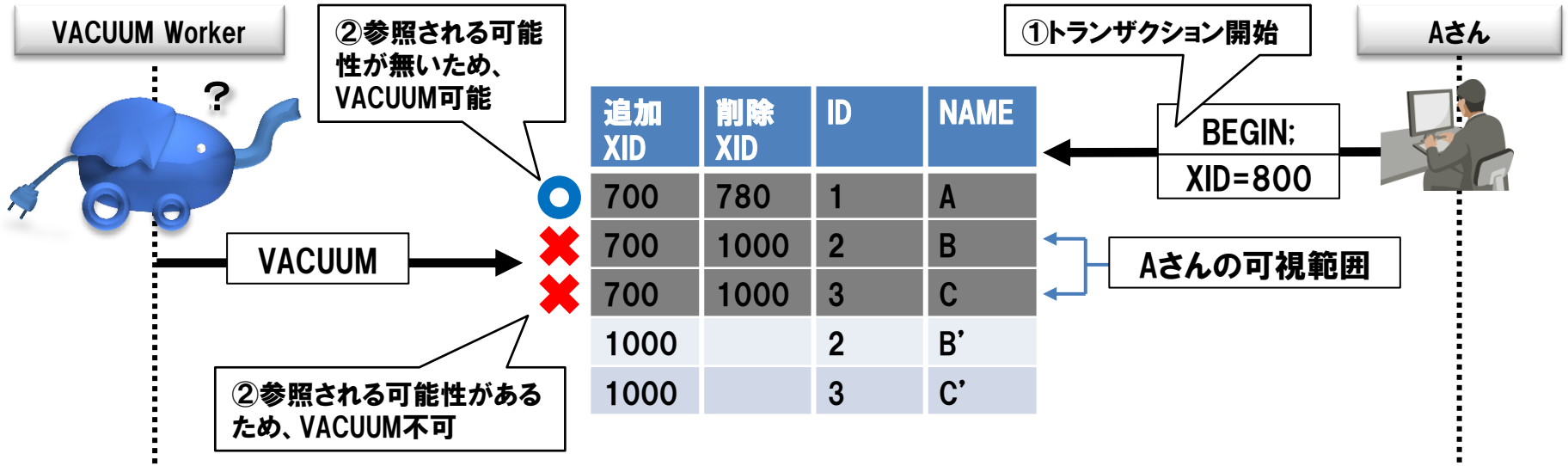
FUJITSU

■ ロングトランザクションが原因でVACUUM が阻害される事象を2パターンご紹介します。

■ パターン1 : 古いトランザクションが閉じられていない。

※ISOLATION LEVEL が REPEATABLE READ 以上で発生します。

参考 : http://www.postgresql.jp/document/9.1/html/transaction-iso.html



■ ログには以下のように出力され、エラーとならないため注意が必要です。

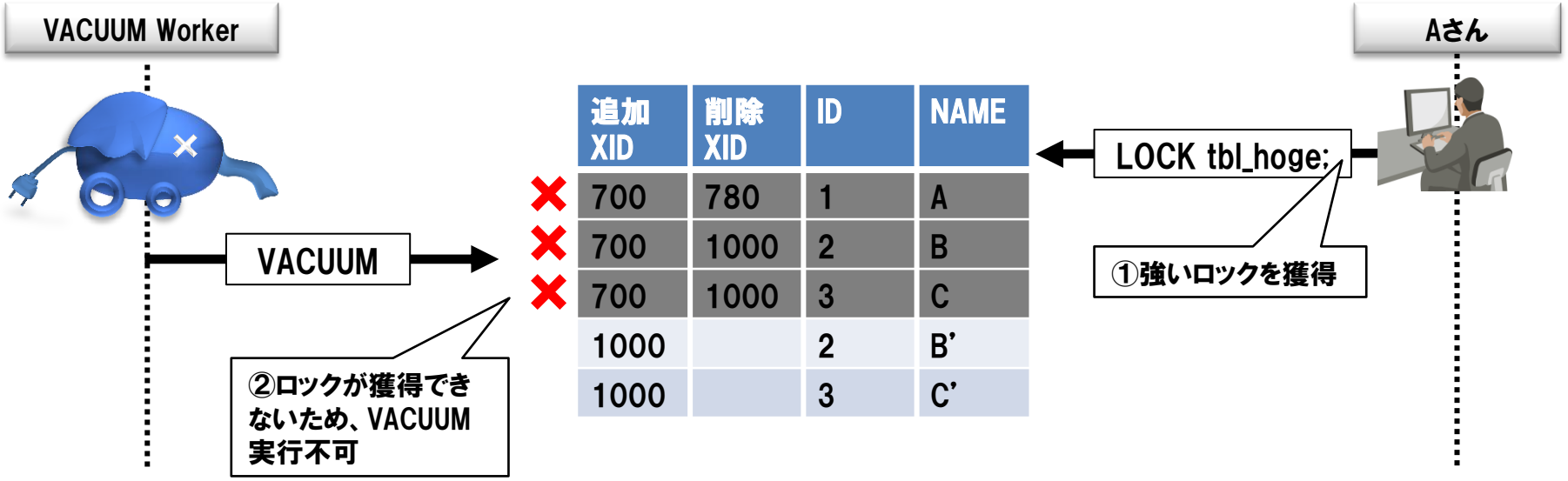
```
LOG:  automatic vacuum of table "db_hoge.public.tbl_fuga": index scans: 0
#011tuples: 0 removed, 20000 remain
```

性能劣化① ロングトランザクションによるVACUUM阻害（回答詳細）

FUJITSU

■ パターン2 ：ロックが競合する

- VACUUM は SHARE UPDATE EXCLUSIVE 以上のロックと競合します。
  http://www.postgresql.jp/document/9.1/html/explicit-locking.html
- このパターンでは、追加XID、削除XIDを考慮した可視範囲は関係ありません。



■ ログには以下のように出力され、エラーとならないため注意が必要です。

```
LOG:  skipping vacuum of "tbl_hoge" --- lock not available
```

性能劣化① ロングトランザクションによるVACUUM阻害（回答詳細）

FUJITSU

■ 解決方法

- ロングトランザクションを終了してください。
  - その後 VACUUM コマンドを実行するか、しばらくした後、自動VACUUM が実行されます。
- バッチ等のロングトランザクションになることがわかっている処理については、業務の更新処理が少ない時間帯に実行することで、不要領域の過剰な増加を防止できます。

■ 備考

- pgpool-II を利用している環境でロングトランザクションを kill する場合は、前述（p.13）した手順で実施してください。
- ロングトランザクションの確認は以下のように実施しますが、9.2 以降ではトランザクションの最後に実行したクエリが確認できるようになっています。
  - 小さな変更点ですが、うれしい変更だと思います。
- ~ 9.1

```
=# SELECT procpid,waiting,(current_timestamp-xact_start)::interval(3) AS duration,current_query FROM pg_stat_activity;
 procpid | waiting |    duration     | current_query
---------+---------+-----------------+---------------------------------
6110     | f       | 00:00:06.667    | <IDLE> in transaction
```

- 9.2 ~
  - 9.1 までの current_query が query と state に分割されました。

```
=# SELECT pid,waiting,(current_timestamp-xact_start)::interval(3) AS duration,query, state FROM pg_stat_activity;
  pid   | waiting |     duration     |  query                    |       state
--------+---------+------------------+---------------------------+---------------------
 28222  | f       | 00:00:11.605     |  select * from tbl_hoge;  |  idle in transaction
```

■質問

- ■応答速度が遅いクエリがあります。原因と対策を教えてください。

■回答

- ■動作環境を調査した結果、プリペアド文を利用したクエリに対して、最善ではない実行計画が採用されていることがわかりました。
  - プリペアド文とは、指定されたSQLの実行計画の作成結果をPostgreSQLに準備状態である文（SQL）のことを言います。
  - お客様は Java の PreparedStatement インターフェースを用いてプリペアド文を作成していました。
- ■次頁以降に、本事象の発生原因を記載します。

性能劣化② プリペアド文の利用による性能劣化（回答詳細）

FUJITSU

- プリペアド文利用時の性能劣化の発生原因を以下に記載します。
- 以下のようなテーブルが存在しているとき、id2 のインデックス idx_id2 が利用されていることを期待して、SELECT を実行した場合、$1（バインド変数）としている部分の内容がプランナに考慮されず、実行計画が劣化する可能性があります。
  - お客様の事例では、JDBC のPreparedStatement インターフェースを利用していましたが、下記検証では直接 PREPARE を実行しています。

id1（idx_id1）	id2（idx_id2）	name
1	1	Tetsuya
2	1	Toshihiko
（省略）		
999999	1	Mayu
1000000	2	Masanori

id1　：1 ～ 1,000,000、インデックスあり
id2　：1が999,999件、2が1件 、インデックスあり
name：任意、インデックスなし

```
=# PREPARE testSelect(int) AS SELECT * FROM tbl_hoge WHERE id2 = $1;
=# -- EXPLAIN ANALYZE の結果は以下の通り
=# EXPLAIN ANALYZE EXECUTE testSelect(2);

                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Seq Scan on tbl10  (cost=0.00..17906.00 rows=1000000 width=13) (actual time=123.309..123.310 rows=1 loops=1)
   Filter: (id2 = $1)
 Total runtime: 123.826 ms
(3 rows)
```

1,000,000行取得すると予測しているため、Seq Scanが選択されている。

実際は1行のみ取得されるので、Index Scanが選択されるべき。

性能劣化② プリペアド文の利用による性能劣化（回答詳細）

FUJITSU

- 前述の通り、プリペアド文を利用した場合にバインド変数部分が実行計画立案時に考慮されないため、実行計画が以下のように劣化する可能性があります。
- 問題点と、解決方法を以下にまとめます。
  - この他にも実行計画の劣化は発生する可能性があります。
    - 机上での検討ですが、ジョインの実行計画の劣化などもあるのではないかと考えています。

項番	問題	解決方法
1	Index Scan を選択してほしいときに、Seq Scanが選択される。	次頁の解決方法①、②、③を参照して下さい。
2	Seq Scanを選択してほしいときに、Index Scan が選択される。	次頁の解決方法①、②、③、④を参照して下さい。

性能劣化② プリペアド文の利用による性能劣化（回答詳細）

FUJITSU

■ 解決方法①：9.2 以降にバージョンアップする

- 9.2 以降では、バインド変数も実行計画立案時に考慮されるようになりましたので、アップデートすることで解決が可能です。

■ 解決方法②：プリペアド文を使用しない

- 下記のように、接続DBのパラメータに prepareThreshold =0 を追記して、実行計画が改善されるか確認してください。
- 接続パラメータ ”prepareThreshold” は、プリペアド文の利用を開始するまでのクエリ実行回数閾値を設定します。
- prepareThreshold =0 とすることで、プリペアド文は作成されなくなります。

```
String url = "jdbc:postgresql://localhost:5432/test?prepareThreshold=0";
```

■ 解決方法③：実行計画制御パラメータを利用する

- 実行計画の誤りが明確な場合は、以下のように 当該SQL 実行の前後に以下の様な制御を加えることが可能です。
- 副問い合わせなどの複雑なSQLであっても一文である限りは、一括で設定されるので注意してください。

```
Statement setToSeqScan = connection.createStatement();
setToSeqScan.execute(“SET enable_seqscan TO off”);
setToSeqScan.close();
（略：実行計画に問題のあるSQLをここで実行）
setToSeqScan.execute(“SET enable_seqscan TO default”);
```

■ 解決方法④：不要なインデックスを削除する

- 前頁の事例とは逆に、Seq Scan となるべき箇所で、Index Scan が選択される場合もあります。
- その場合は、不要なインデックスを削除することで解決が可能です。
- そもそも、不要なインデックスは削除してあるべきなので、優先順位は低いです。

FUJITSU

■ご清聴ありがとうございました。