chuden CTI

中部電力グループ

PostgreSQLサポートサービスに向けた取組
～安心して利用できる礎作り～

2014年9月18日

株式会社 中電シーティーアイ

アジェンダ

- はじめに
  - PostgreSQLサポートサービスとは
  - 質の高いサービスを提供するために必要なもの
- 取組① –製品機能の習得–
- 取組② –保守・運用体制の確立–
- 今後の活動

会社概要

項目	内容
社名	**株式会社中電シーティーアイ**
本社所在地	愛知県名古屋市
設立	2003年（平成15年）
従業員数	1021名（平成26年8月現在）
売上高	297億円（平成25年度）
主な事業内容	システム開発・システム保守・運用 ネットワークシステムインテグレーション データセンター システム運用管理 データ授受・プリンタ等の周辺処理サービス 科学技術（環境情報・技術開発・解析）

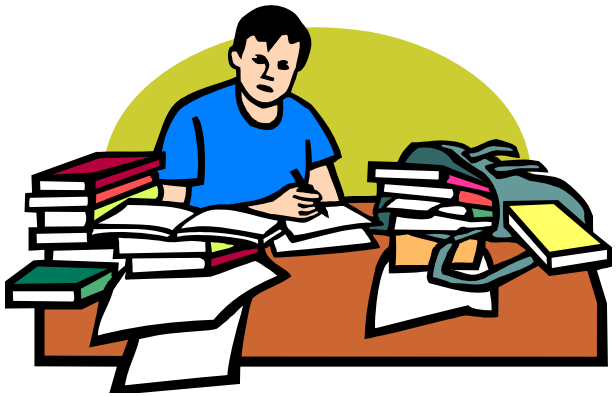
はじめに

【PostgreSQLサポートサービスとは】

PostgreSQL（OSS）の利用者に対する保守、およびOSSを利用するシステムの開発・保守・運用の支援を行うサービスである。

<u>質の高いサービスを実現するためには．．．</u>

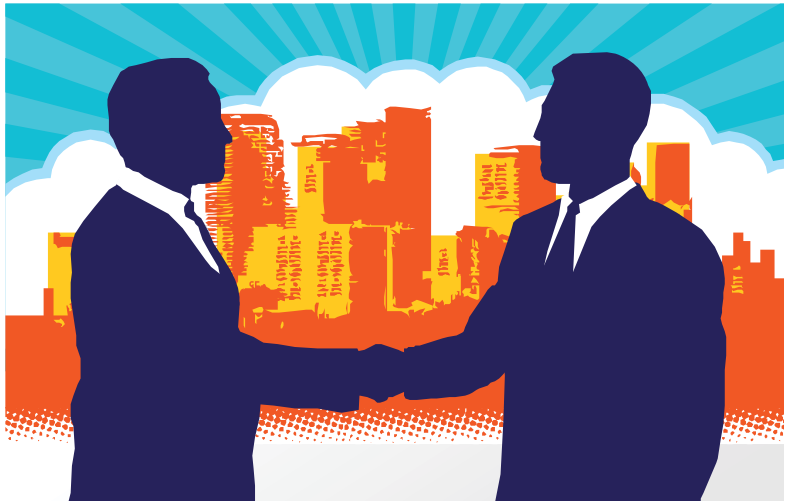
製品機能の習得	保守・運用体制の確立



はじめに

【取組の背景】

受 注 拡 大



- 商用ソフトウェアの保守限界に伴うバージョンアップ・リプレース費用や保守費用の増大への対応（コストダウン）
- 自社の強みを形成し、新たなビジネスを開拓（受注拡大）

製品機能の習得

①対象ソフトウェア領域の決定

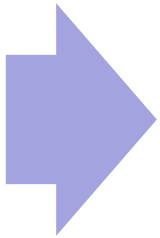
②机上での比較評価に基づくOSS選定

③商用ソフトウェアとの機能差を埋める方式検討

④実機検証

製品機能の習得

①対象ソフトウェア領域の決定
約400種類のOSSを、機能別の領域に分類し、
置き換え効果の大きい領域を対象として設定



WEB3層環境の範囲

＜機能別の領域＞

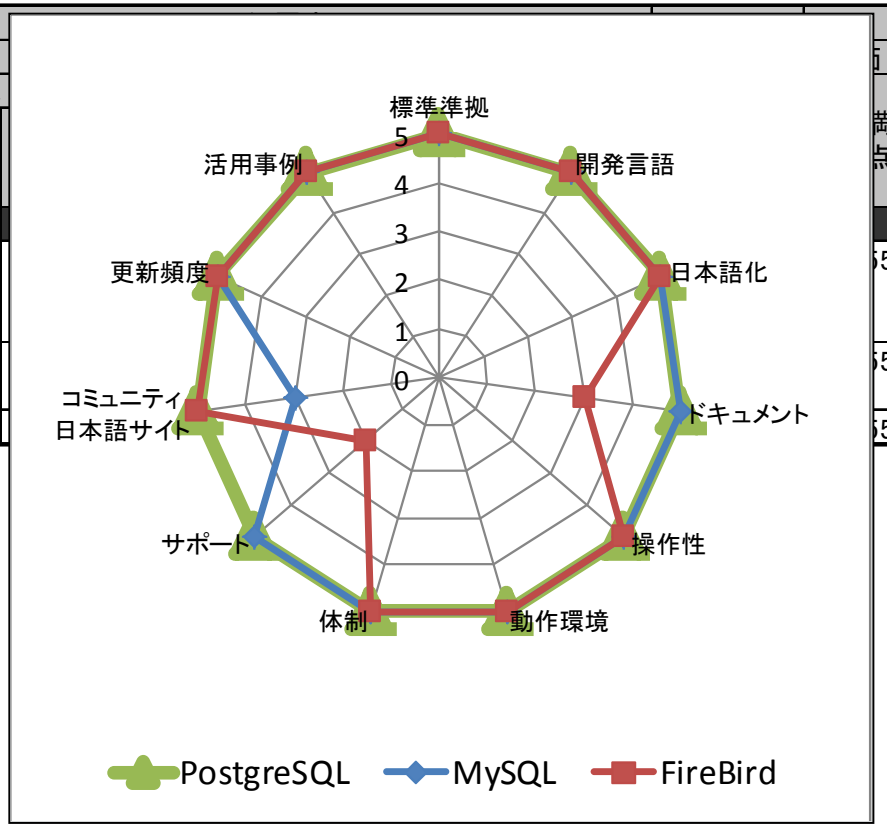
◇オペレーティングシステム（OS）
◇ Webサーバ
◇アプリケーションサーバ（APサーバ）
◇データベース
◇仮想化
◇クラスタ化
◇稼働管理
◇性能管理
◇電子メール
◇グループウェア
◇ビジネスインテリジェンス（BI）
◇統合アカウント管理
◇ディレクトリ連携
◇ DNSサーバ
◇ DHCPサーバ
◇ファイル共有
・・・etc

製品機能の習得

②机上での比較評価に基づくOSS選定
技術参照モデルに対する機能の充足性、および
開発サポート体制を評価

採用OSS製品選定

No	分類	レイヤ	OSS ミドルウェア	総評	レイヤ機能	基本 採点	基本 満点
–	基本機能	DBサーバ					
1			PostgreSQL	94%	88%	12	14
2			MySQL	92%	88%	12	14
3			FireBird	80%	69%	11	14



製品機能の習得

【コミュニティ版OSSを採択】

技術参照モデル(TRM)に対する機能の充足性を重視

当社の採択	通例の採択
コミュニティ版 ・最新機能が利用可能 ・検証やサポートの担保はない ・無償	ディストリビューション版 ・利用可能な機能は1世代以上前 ・検証やサポートはディストリビュータが担保 ・有償
○ ・商用ソフトウェアに近いまたは同等の機能提供が可能 ・OSSのメリットを最大限に活用可能	・商用ソフトウェアと同等の機能を実現できない場合がある

製品機能の習得

③商用ソフトウェアとの機能差を埋める方式検討
不足機能に対する補完OSSツールを選定し、
組合せ検証により必要な機能の安全性を評価

OSS補完ツール選定

ソフトウェア品質特性 (ISO/IEC9126-1)	機能名	商用ミドルウェア	PostgreSQL	GAP有無	選定ツール	PostgreSQL検証項目抽出観点	
機能性/セキュリティ	セキュリティ	格納データの暗号化 ・透過的に暗号化可能 ・バックアップデータの暗号化監査 ・必須監査 ・DBA監査 ・標準監査 ・ファイングレン監査	格納データの暗号化 ・暗号化パッケージを実行 ・バックアップデータの暗号化監査 ・トリガーベースで個別に設定 ・ログのレベルを上げることである程度可能	有	pgcrypto	・暗号化について ・log_statement(enum)の設定による、 ログ取得について。 ・監査用トリガプロシージャについて。	・暗号化データ ・未許可ユーザ ・トリガプロシー ・トリガプロシー
信頼性/障害許容性	クラスタリング	インスタンスのみ追加で負荷分散	追加の際、サーバ・ストーレージ一式が必要	有	pgpool-II	フェールオーバーについて。	pgpool-IIを利用
信頼性/障害許容性	データ・ ミラーリング (ストリーミング レプリケーション)	・スタンバイ側参照可能 ・スタンバイ側テスト可能	・スタンバイ側参照可能	有	pgpool-II	・ストリーミングレプリケーションについて。 PostgreSQLついては、参照系クエリのみ 負荷分散可能な構成が前提となっている。	・更新性能劣化 ・昇格の時間測
効率性/時間効率性	データアクセス	性能劣化しにくい	性能劣化しやすい	有	pg_reorg	・VACUUMによる性能改善について。 ・pg_reorgによる性能改善について。 ・HOTによる性能改善について。	・PostgreSQL ・VACUUMの効 ・pg_reorgがDB 検証。 効果と
信頼				有			妥当

PostgreSQL補完ツール

PostgreSQL JDBC Driver , pg_bulkload , pg_statsinfo
pg_reorg , auto_explain , ora2pg , pgcrypto , pgpool-Ⅱ

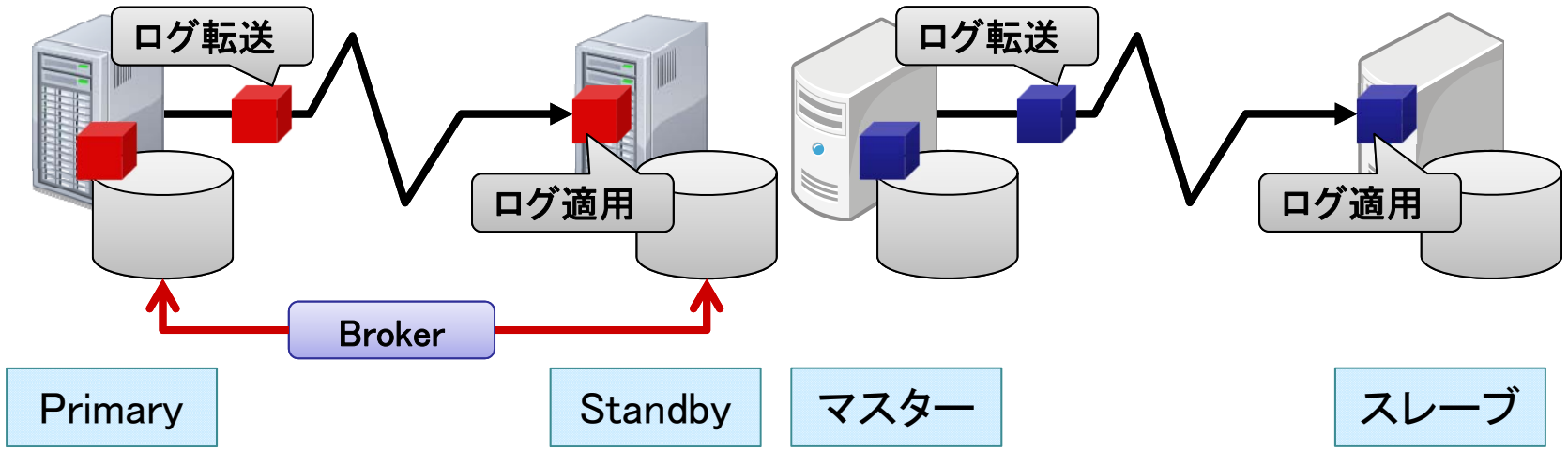
１０項目ＧＡＰ有

製品機能の習得

【Fit & Gap –データ・ミラーリング–】

商用ミドルウェア	PostgreSQL
特徴 ・REDO転送によるデータミラー ・自動フェイルオーバー ・テスト用途でスタンバイを活用可能	特徴 ・WAL転送によるデータミラー ・手動フェイルオーバー ・スレーブ側を参照利用可能

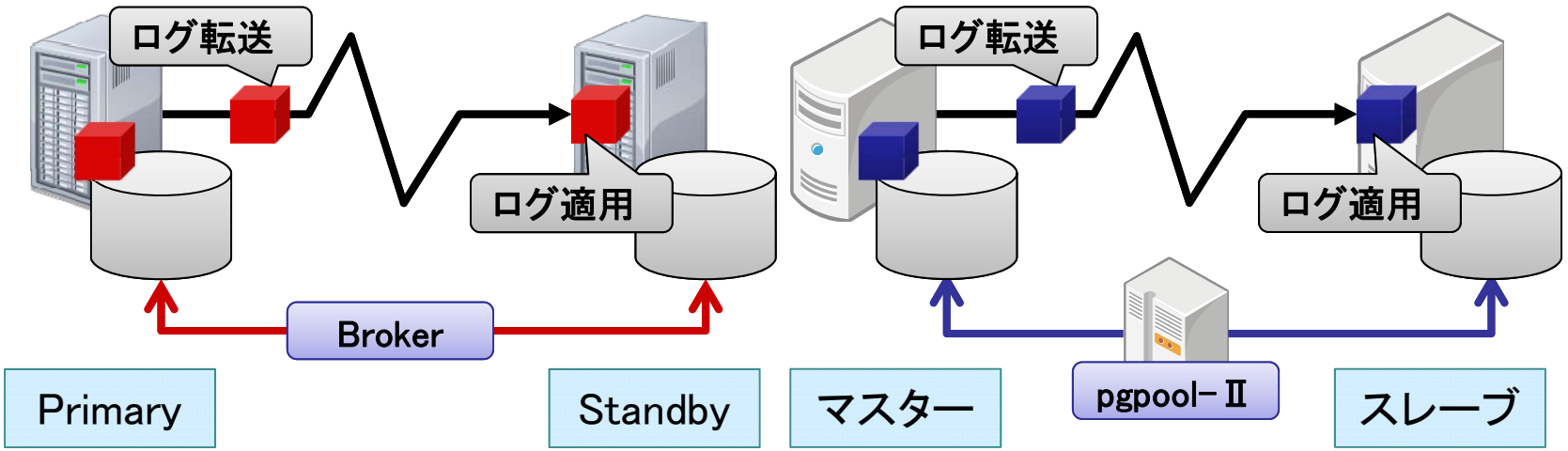
GAP有り



製品機能の習得

【Fit & Gap –データ・ミラーリング–】

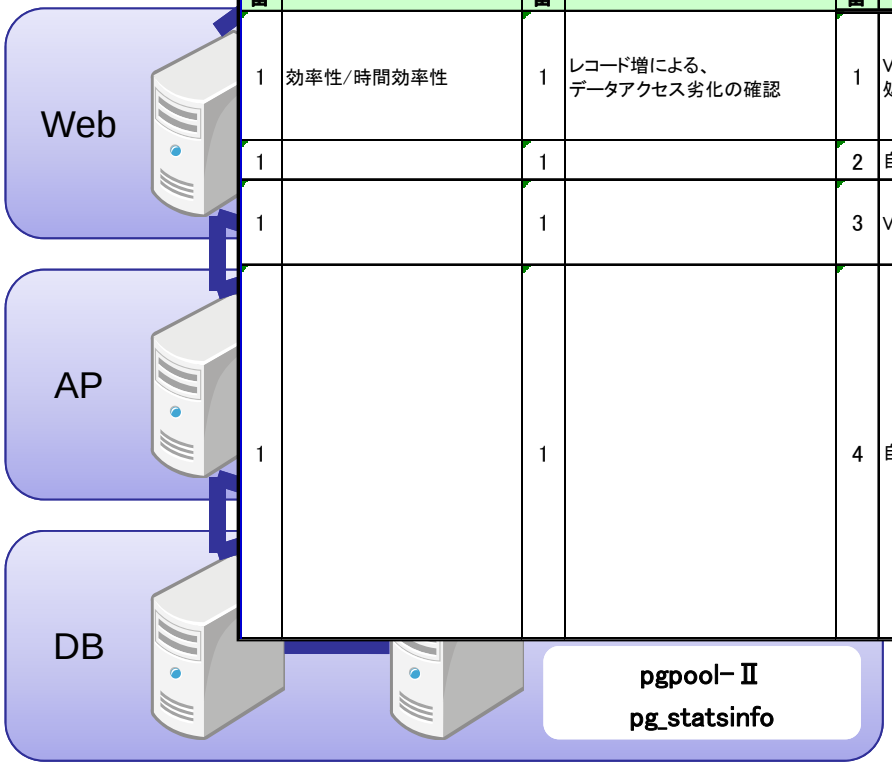
商用ミドルウェア	PostgreSQL + pgpool-Ⅱ
特徴 ・REDO転送によるデータミラー ・自動フェイルオーバー ・テスト用途でスタンバイを活用可能	特徴 ・WAL転送によるデータミラー ・自動フェイルオーバー ・テスト用に別途データベースクラスタを構築



製品機能の習得

④実機検証
OSS単体/組合せ検証により品質特性を評価
合わせて当社の中核となるOSS技術者の育成

品質・安全性の担保



大項番	ソフトウェア品質特性（ISO/IEC9126-1）	中項番	中項目名（括弧はFIT&GAP名）	小項番	小項目名	実施内容/補足事項
1	効率性/時間効率性	1	レコード増による、データアクセス劣化の確認	1	Vacuum, 自動Vacuumが行われない環境における、処理速度の遅延の定期記録	・autovacuum = off ・pgbenchやJdbcRunnerを走行させる。 ※補足（Postgresqlドキュメントによると、Vacuumされないテーブルに対して20億トランザクションが走査されると動作を停止します。）
1		1		2	自動Vacuum処理必要性ログ情報の確認	・自動Vaccumのログ出力内容を確認する。
1		1		3	Vacuum時のCPU/IO情報の取得方法の確立	・試験モデルを定め、vacuum時のCPU/IO性能を取得する。 (試験モデル検討が困難な場合は、pgbenchによる任意値を策定して作業を行う)
1		1		4	自動Vacuum設定の改善方法の確立	・log_autovacuum_min_durationに目標となる処理時間(ミリ秒)を設定し、同数値に近づけるべく、チューニング作業を実施する。 データベース数は一つとする。 ※補足条件: チューニング値は以下の値のみに注力すること。 1. autovacuum_naptime 2. autovacuum_vacuum_cost_delay/autovacuum_vacuum_cost_limit 3. autovacuum_vacuum_scale_factor, autovacuum_vacuum_threshold (行数閾) 参考(Postgres Plus AutoVacuumの負荷が高いとき): http://postgres.sios.com/modules/newbb/viewtopic.php?topic_id=108&forum=1

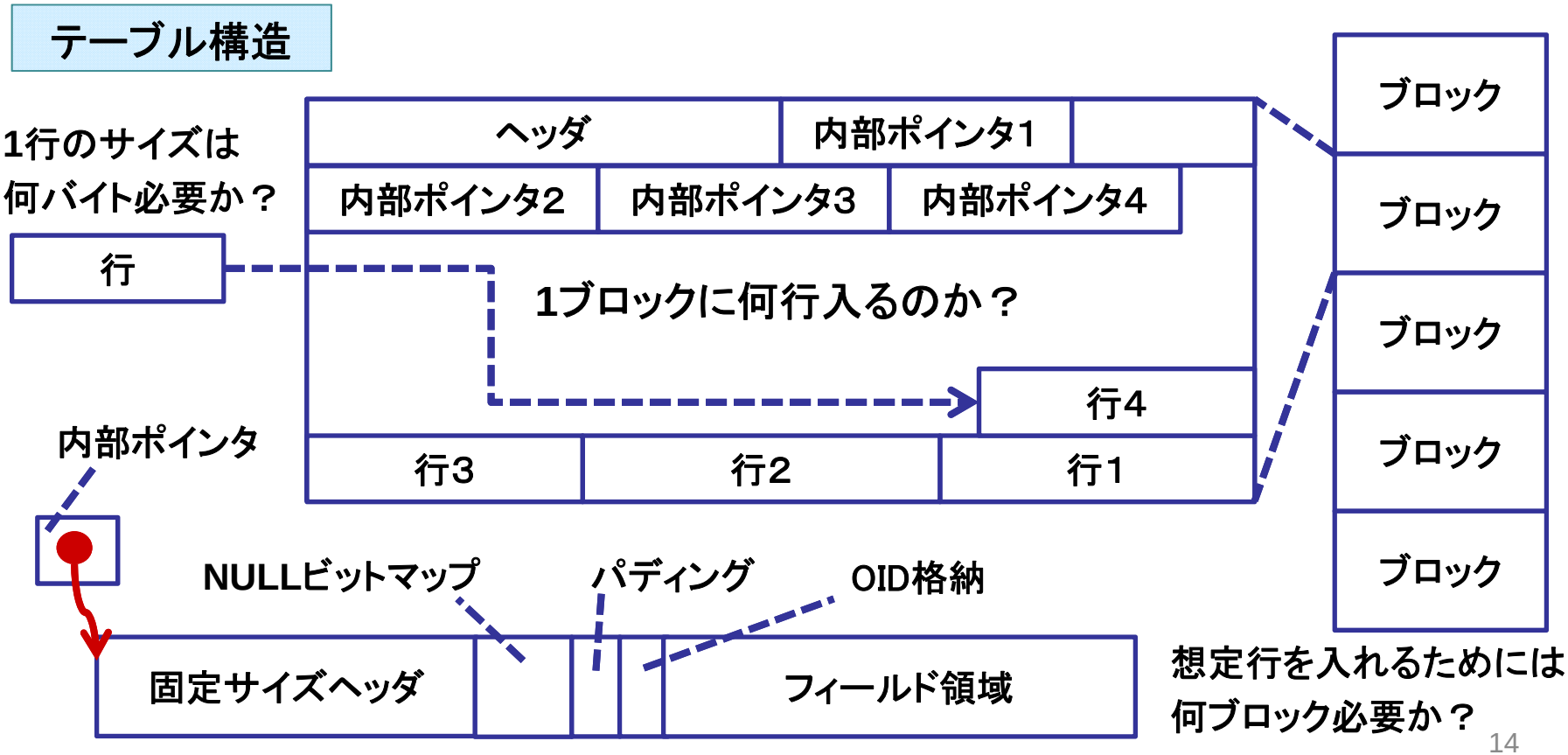
８６項目を検証 ※単体検証

製品機能の習得

【実機検証　-ディスクサイズの見積もり手法の確立-】

＜検証概要＞

ディスクサイズの見積り式を確立し、実機を用いて妥当性を確認する。



製品機能の習得

【実機検証　-ディスクサイズの見積もり手法の確立-】

＜行サイズを求める計算式＞

内部ポインタ	4

＜単位：Byte＞

固定ヘッダ	NULLビットマップ	パディング	OID格納	フィールド領域
23			0	

サンプルテーブル

CREATE TABLE t1 (i INTEGER NOT NULL, j INTEGER NOT NULL) ;

製品機能の習得

【実機検証　-ディスクサイズの見積もり手法の確立-】

＜行サイズを求める計算式＞

内部ポインタ	4

＜単位：Byte＞

固定ヘッダ	NULLビットマップ	パディング	OID格納	フィールド領域
23	0		0	

サンプルテーブル

CREATE TABLE t1 (i INTEGER NOT NULL, j INTEGER NOT NULL) ;

NULLビットマップサイズとは

NULLを許す列がテーブルに1つでも含まれる場合、列あたり1bit必要となる値

今回のサンプルテーブルではNULLを許可していないため、 0 である。

製品機能の習得

【実機検証　-ディスクサイズの見積もり手法の確立-】

＜行サイズを求める計算式＞

内部ポインタ	4

＜単位：Byte＞

固定ヘッダ	NULLビットマップ	パディング	OID格納	フィールド領域
23	0	1	0	

サンプルテーブル

CREATE TABLE t1 (i INTEGER NOT NULL, j INTEGER NOT NULL) ;

パディングサイズとは

固定ヘッダサイズとNULLビットマップサイズの合計が4の倍数にならない場合に、
切り上げ調整するために必要となる値
今回のサンプルテーブルでは、合計が23Byteなので、パディングサイズは 1 である。

製品機能の習得

【実機検証　-ディスクサイズの見積もり手法の確立-】

＜行サイズを求める計算式＞

内部ポインタ	4

＜単位：Byte＞

固定ヘッダ	NULLビットマップ	パディング	OID格納	フィールド領域
23	0	1	0	8

サンプルテーブル

CREATE TABLE t1 (i INTEGER NOT NULL, j INTEGER NOT NULL) ;

フィールド領域サイズとは

各データ型の格納サイズの合計から算出される値

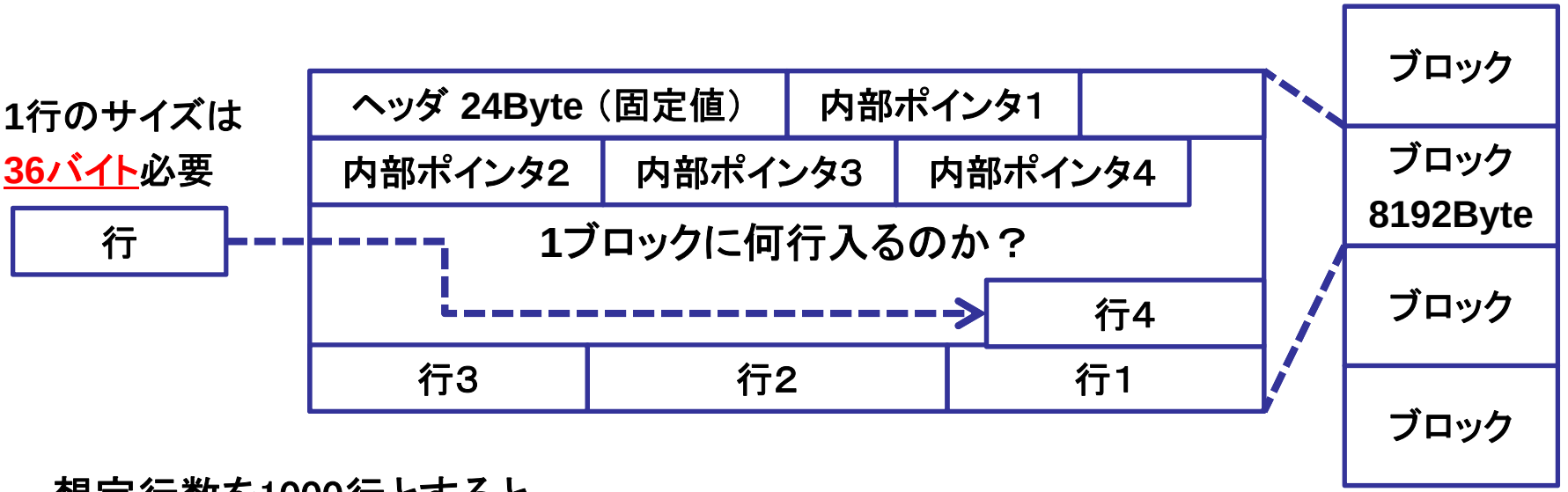
※　フィールド領域サイズは8Byteずつ増えるため、合計値が8の倍数にならない場合は、切上げ調整する。

今回のサンプルテーブルでは、INTEGER型（4Byte）が2列なので 8 である。

なので、1行のサイズは 4 + 23 + 0 + 1 + 0 + 8 = 36 Byte である。

製品機能の習得

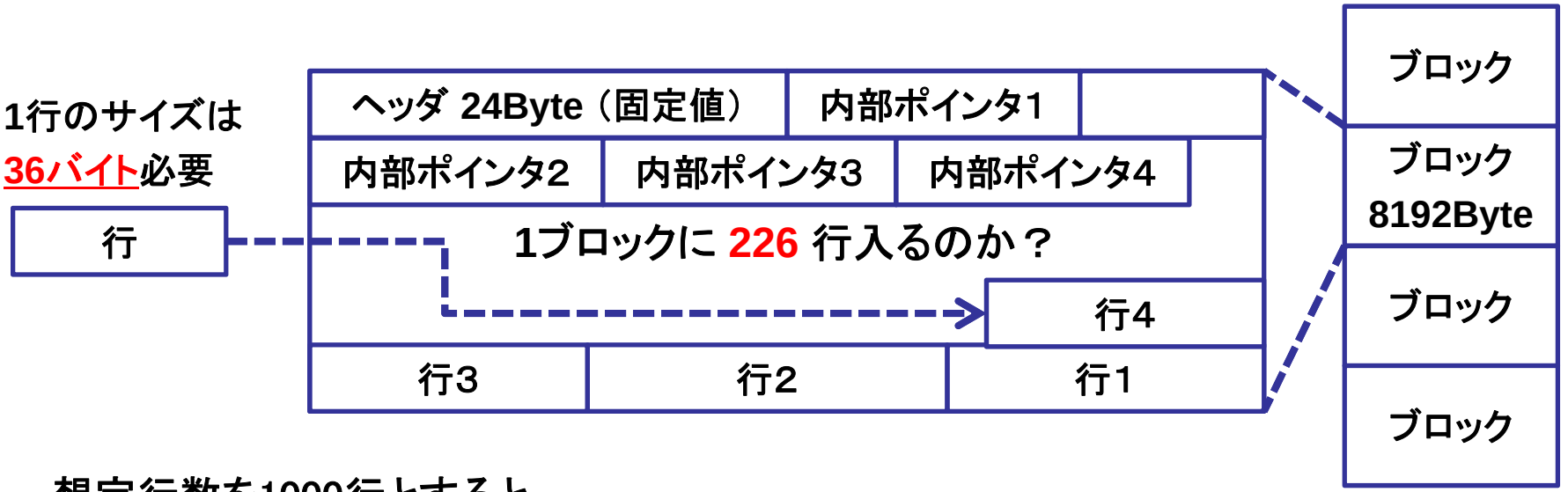
【実機検証　-ディスクサイズの見積もり手法の確立-】
＜テーブルサイズを求める計算式＞



想定行数を1000行とすると

製品機能の習得

【実機検証　-ディスクサイズの見積もり手法の確立-】
＜テーブルサイズを求める計算式＞



想定行数を1000行とすると
1ブロックに格納できる行数は、(8192 - 24) ÷ 36　＝ 226.888… 行である。
よって、1ブロックに226行（切捨て）格納することができる。
つまり、1000行格納に必要なブロック数は、1000 ÷ 226 ＝ 4.42477…である。
よって、1000行格納に必要なブロック数は 5 ブロック（切上げ）となる。
以上のことから、テーブルサイズは 8192 × 5 ＝ 40960 Byte であると計算できる。

製品機能の習得

①対象ソフトウェア領域の決定
約400種類のOSSを、機能別の領域に分類し、
置き換え効果の大きい領域を対象として設定

→ WEB3層環境の範囲

②机上での比較評価に基づくOSS選定
技術参照モデルに対する機能の充足性、および
開発サポート体制を評価

→ 採用OSS製品選定

③商用ソフトウェアとの機能差を埋める方式検討
不足機能に対する補完OSSツールを選定し、
組合せ検証により必要な機能の安全性を評価

→ 補完OSSツール選定

④実機検証
OSS単体/組合せ検証により品質特性を評価
合わせて当社の中核となるOSS技術者の育成

→ 品質・安全性の担保

運用・保守体制の確立

①障害時における解析手法の確立

②運用ルールの整備

③問合せ管理システムの構築

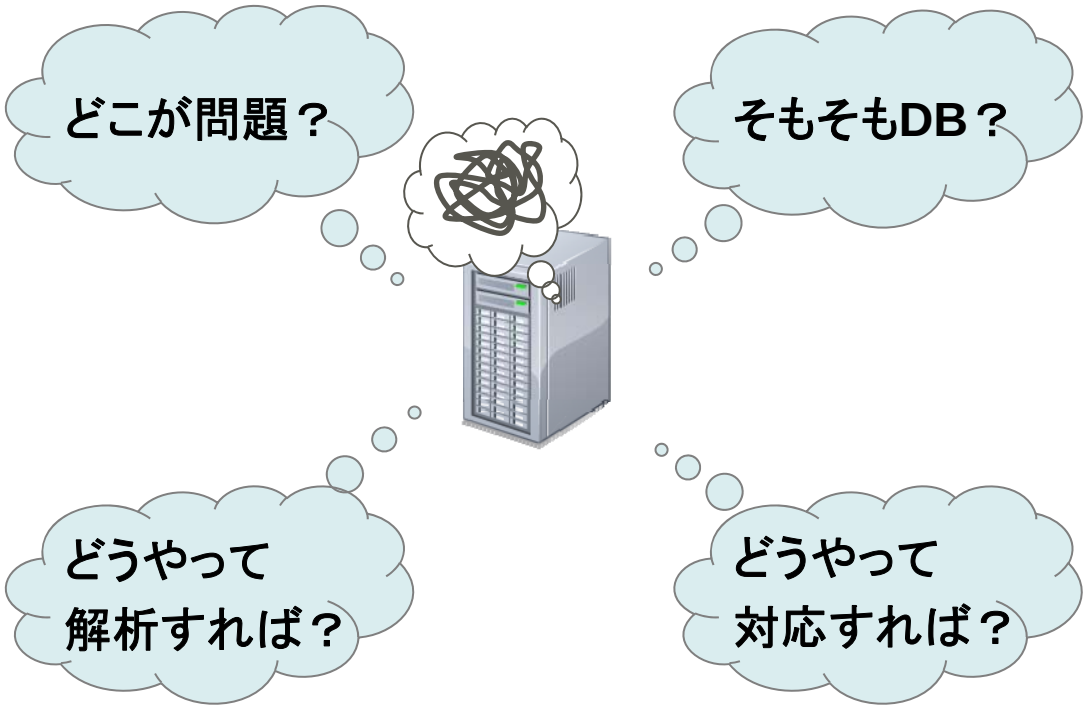
④教育コンテンツの作成

運用・保守体制の確立

①障害時における解析手法の確立
解析に必要な情報、ツールの利用方法や障害
解析結果を取り込んだ解析フローを整備

脱・属人化

障害が発生した時・・・



実施内容

- 障害事例の収集、および分類
- 解析手法（ツール）の洗い出し
- 各シチュエーションの解析方法の考察
- 障害時解析フローの作成

運用・保守体制の確立

【解析手法（ツール）の洗い出し】
障害解析に利用するツールの一覧化
若手・異動者・問合せ者への説明資料として活用

No.	ツール/コマンド名	
1	PostgreSQLログ	PostgreSQLで発生したエラー
2	稼働統計情報	稼働統計情報は、PostgreSQ DBの稼働状態やDB内で起
3	pg_ctl	PostgreSQLの起動、停止、
4	pg_isready	PostgreSQL が利用可能か
5	show	PostgreSQLのパラメータ設定
6	psql	PostgreSQLに接続するため 取得することも可能です。
7	EXPLAIN	PostgreSQLがSQL実行時に
8	システム管理関数	PostgreSQLを管理する上で
9	auto_explain	自動的に実行に時間かかる
10	pg_stat_statments	SQL の実行回数や実行時間
11	pg_buffercache	PostgreSQLの共有バッファ ます。
12	pg_xlogdump	pg_xlogdumpは、PostgreSQL
13	oid2name	データベースクラスタ内のオ
14	pg_freespacemap	VACUUMによって、空き領域
15	pgrowlocks	指定したテーブルにおける行
16	pgstattuple	行レベルの統計情報を出力
17	pgAdmin III	データベースやユーザなどを
18	phpPgAdmin	データベースやユーザなどを

障害発生時に解析や報告のため、使用するレポートツールを下表に記

項番	レポート	
1	pg_stat_reporter/pg_statsinfo	pg_statsinfoは、各種統計情 pg_stat_reporterは、レポート ポートを作成します。

pg_stat_statementsモジュール

- pg_stat_statements は、SQLの実行回数や実行時間などの統計情報を収集するモジュールです。収集された統計情報はビューを通して参照できます。
  - 実行回数の多いSQLや実行時間の長いSQLを調べる際に利用します。
  - 利用には、追加の共有メモリを必要とするため、pg_stat_statementsをpostgresql.conf中のshared_preload_librariesに追加し、ロードする必要があります。
  - モジュールの有効・無効化の際には、PostgreSQLの再起動が必要です。

設定例：

```
shared_preload_libraries = 'pg_stat_statements'

pg_stat_statements.max = 10000
pg_stat_statements.track = all
```

※このモジュールは、pg_stat_statements.max*track_activity_query_sizeバイトの追加の共有メモリを必要とします。パラメータpg_stat_statements.trackにnoneが設定されていても、モジュールがロードされている限り、常にこのメモリが消費されることに注意してください。

運用・保守体制の確立

【各シチュエーションの解析方法の考察】
障害シチュエーションに対する各レイヤ毎の
対応方法を一覧化

No	シチュエーション／障害	原因諸元	OS	RDBMS	備考
1	DBMSに接続できない	ネットワーク	1.DBMSサーバへのネットワーク経路確認 ・ping ・traceroute	-	OSコマンドでDBMSサーバへの接続可否を確認
2		プロセス	1.OS機能によるアクセス制御確認 ・iptables ・selinux	1.PostgreSQLの稼働状態を確認 ・PostgreSQLログ ・pg_ctl ・pg_isready 2.接続できない原因を確認 ・PostgreSQLログ 3.原因に応じて、以下を確認 3.1.PostgreSQLサーバへのアクセスが拒否された場合 3.1.1.listen_addressesの設定確認 ・SHOW 3.2.データベースへのアクセスが拒否された場合 3.2.1.pg_hba.confの設定値確認 ・・・	PostgreSQLログを用いて、接続ができない原因を確認
3	SQLがエラーになる	ロック	-	1.エラーになったSQLとその原因を確認 ・PostgreSQLログ 2.原因に応じて、以下を確認 2.1.デッドロックによるキャンセルされた場合 2.1.1.ロックの状況を確認 ・稼働統計情報 ・pgrowlocks 2.1.2.ロックを取得するSQLの確認 ・稼働統計情報 ・PostgreSQLログ 2.2.ロック待ちによるSQLタイムアウトされた場合 ・・・	PostgreSQLではデッドロックが発生した場合、デッドロックを発生させたSQLをエラーにする

２５事例 ※現在

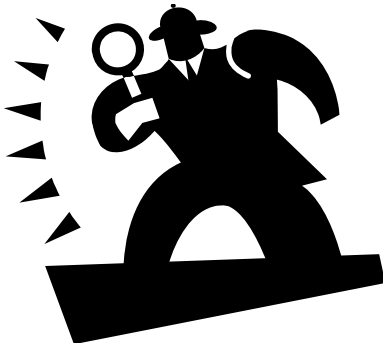
運用・保守体制の確立

【各種同系ツールの比較表】
確認方法が複数存在する各種同系ツールの比較を実施し、使用目的を明確化

確認観点	稼動統計情報（pg_stat_activity）	pgrowlocks	PostgreSQLログ（log_lock_waits）
ロック状態の確認方法	現在実行中のトランザクションを表示するビューを表示	テーブルにおける行のロック状態を返す関数を実行	任意の秒数以上※1、ロック取得に時間がかかったSQLをログに出力
取得に必要な設定	デフォルト設定	別途インストール	log_lock_waitsをonに設定後、リロード
オーバヘッド	ビュー参照負荷（通常メモリ上に保管されているため軽微）	ロック状態確認時に、テーブルにシーケンシャルスキャンを実行	ログへの書き込み（出力されるSQL数に依存）
取得するロック	ビューに対しAccessShareLock	確認するテーブルに対しAccessShareLock	なし
取得可能な情報の範囲	現時点のみ	現時点のみ	ログ出力設定以降
ロック待ち状態か	○	×	○
ロックを取得するプロセス	○	×	○
ロックを取得するSQL	○	×	○
ロックされた行のID	×	○	×
ロックが競合しているSQL	△	×	×
トランザクション開始からの経過時刻	○	×	×
ロック待ち時間	×	×	○
デッドロックによりキャンセルされたSQL	×	×	○
使用目的	実行中のSQLとロック競合状態の確認	PostgreSQLの学習やデバック	・ロック待ちとなったSQLとロック待ち時間 ・デッドロックによりキャンセルされたSQL

運用・保守体制の確立

【障害時解析フローの作成】
シチュエーション考察を基にフローを作成
フローの分岐に従い、障害解析および対応を実施



SQLがエラーになる

↓

エラーとなったSQLと
その原因の確認
（PostgreSQLログ）

エラーが出力されている①

SQLのタイムアウトが原因でエラーになった場合、以下ログ
メッセージが出力されている
ERROR: canceling statement due to statement timeout
STATEMENT:【エラーになったSQL文】

SQLのタイムアウト（statement_timeout）
設定値を確認（SHOW）

```
# SHOW statement_timeout
 statement_timeout
-------------------
 30
```

タイムアウトが
発生する秒数

SQLタイムアウトが発生

DBMSに遅延が発生している可能性が
ある場合は、障害解析フロー
「DBMSが遅い」を参照

エラーが出力されている②

デッドロックが原因でエラーになった場合
以下ログメッセージが出力されている
LOG: deadlock detected
DETAIL: Process 17759 waits for AccessExclusiveLock on relation
16503 of database 16463; blocked by process 17764.
Process 17764 waits for AccessExclusiveLock on relation 16506
of database 16463; blocked by process 17759.
Process 17759:【デッドロックを発生させたSQL】;
Process 17764:【デッドロックを発生させたSQL】;
HINT: See server log for query details.

Statement:【エラーになったSQL文】

デッドロックになったトランザクションが発行した
SQLを全て確認したい場合は、
log_min_duration_statemnet = 0にする。
ただし、実行されるSQLが全てログに
出力されるため、負荷がかかる。

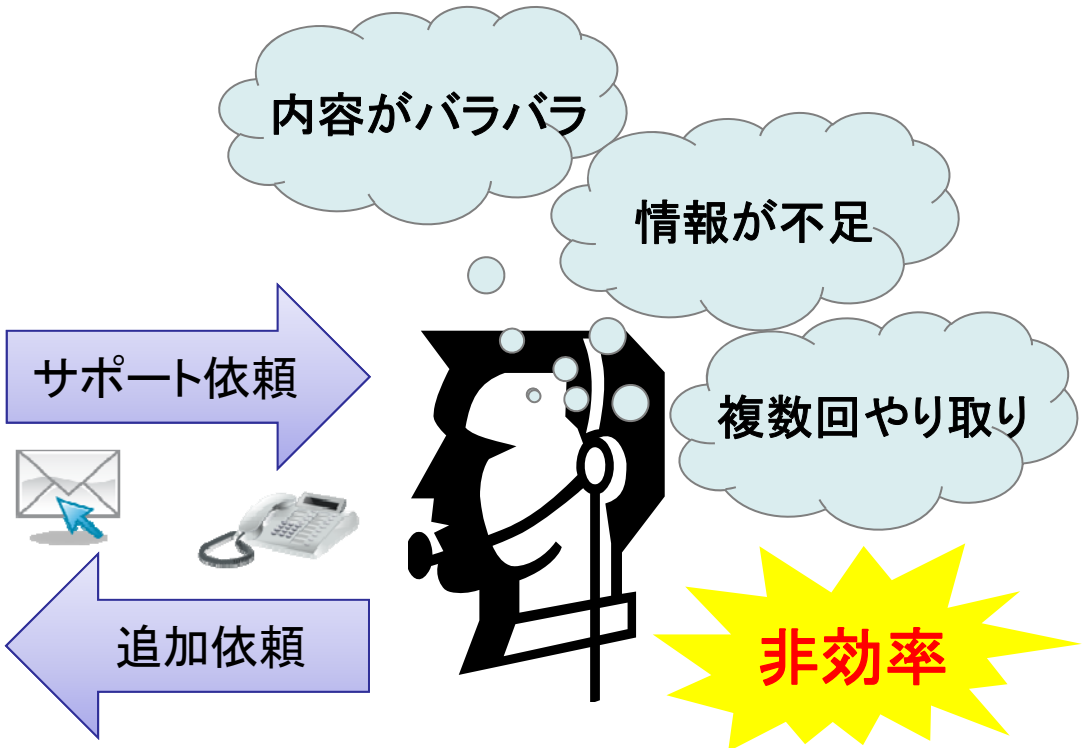
デッドロックが発生

運用・保守体制の確立

②運用ルールの整備
現状課題の洗い出しと対策の検討により、運用規定・フローを改善

業務効率化

運用ルールが確立していないと・・・



実施内容

- 現状の課題抽出
- 問合せフォーマットの決定
- 障害時取得情報の一覧化
- 運用フローの作成

運用・保守体制の確立

【問合せフォーマットの作成】
問合せ者、およびサポート間の認識のズレを排除
記述内容の深さによって、アプローチ方法を検討

問い合せ管理票

（送付先 ： ML のアドレスとか）

		発 行 日	
		管 理 No.(OSST記載)	
タイトル			
プロジェクト名			
システム名		略 号	
運用形態		CPU 情報	
OS 情報		バージョン	
製品情報		バージョン	
発 行 者		連 絡 先	
発行者所属		E-mail	
業務主担当者		連 絡 先	
業務主担当所属		E-mail	
インフラ主担当者		連 絡 先	
インフラ主担当者所属		E-mail	
その他関係者			

問い合せ内容

別紙「問合せ時取得情報」を参照し製品毎に対応する情報を添付すること。
※添付できない情報がある場合はその旨を記載すること。

（下記項目は当方が調査上必要とするための情報で、すべての項目記載を求めるものではない）
【現象（当現象が問合せ対象の製品によるものと判断した情報を含む）】

【問題が発生した製品名，バージョン，リリース，機能名など】

【問題発生時の稼働ソフトウェア】

【問題の発生時期，発覚時期，発生頻度】

【問題にいたった手順】

【問題の再現が可能であれば再現手順】

【ハードウェア，ネットワーク，ソフトウェアへの変更を行った場合は変更内容】

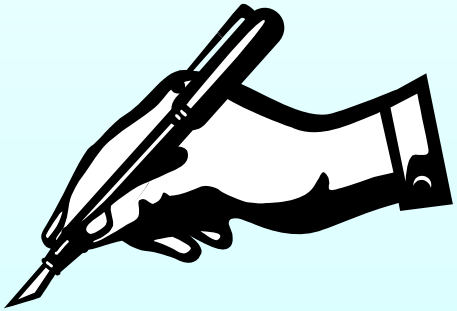
【問題発生時のエラー・メッセージや警告，エラー番号，ログ情報】

		希 望 期 限	
回 答 者		回 答 日	

回答

問合せ内容の統一

- 現象（当現象が問合せ対象の製品によるものと判断した情報を含む）
- 問題が発生した製品名、バージョン、リリース、機器名など
- 問題発生時の稼働ソフトウェア
- 問題の発生時期、発覚時期、発生頻度
- 問題にいたった手順
- 問題の再現が可能であれば再現手順
- ハードウェア、ネットワーク、ソフトウェアへの変更を行った場合の変更内容
- 問題発生時のエラー・メッセージや警告、エラー番号、ログ情報



運用・保守体制の確立

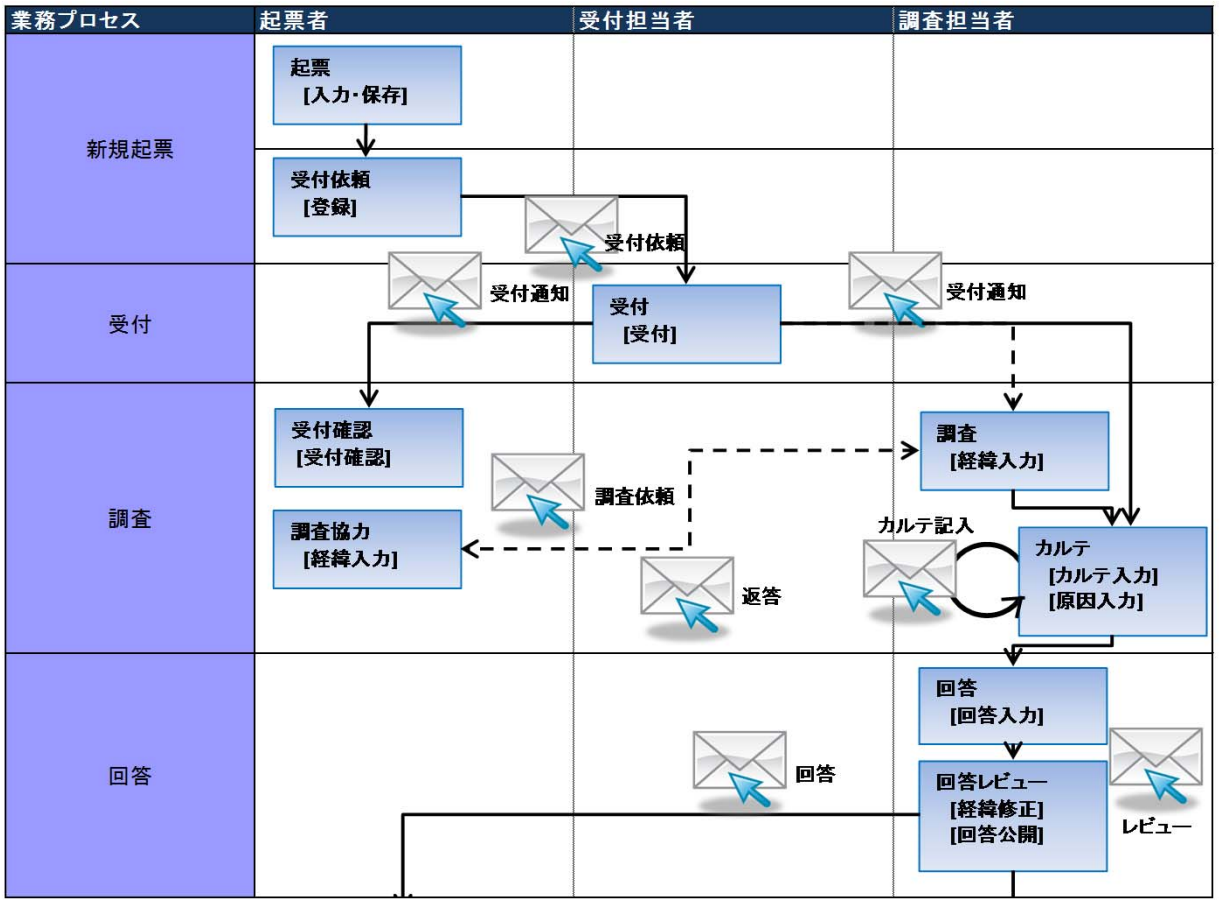
【障害時取得情報の一覧化】
問合せ時に最低限必要な情報を一覧化
問合せ者・サポート間のやり取りを効率化

ミドルウェア名	項目	取得方法	内容
共通	OS	コマンド取得	バージョン、32bit/64bit
PostgreSQL	設定ファイル	ファイル取得	postgresql.conf PostgreSQLの基本設定ファイル
			pg_hba.conf PostgreSQLのアクセス制御ファイル
			pg_ident.conf（外部認証方式時のみ存在） GSSAPIといった外部の認証方式を利用する場合に、OSユーザと データベースユーザを紐付けるために使用、必ずpg_hba.confとセットで使用
	ログファイル	ファイル取得	PostgreSQLのログファイル
	稼動状況の確認	コマンド取得	プロセスの状況を確認
	バージョン	コマンド取得	PostgreSQLのバージョン

コマンド例	出力例
# ps -ef | grep postgres	postgres 1477 1 0 15:21 ? 00:00:00 /usr/pgsql-9.2/bin/postmaster -p 5432 -D /var/lib/pgsql/9.2/data postgres 1479 1477 0 15:21 ? 00:00:00 postgres: logger process postgres 1481 1477 0 15:21 ? 00:00:00 postgres: checkpointer process postgres 1482 1477 0 15:21 ? 00:00:00 postgres: writer process postgres 1483 1477 0 15:21 ? 00:00:00 postgres: wal writer process
# postgres -V	Postgres (PostgreSQL) 9.2.3

運用・保守体制の確立

【運用フローの作成】
運用ルールを基に問合せのやり取りをフロー化
カルテ（対応記録）を活用し、問合せ対応を効率化

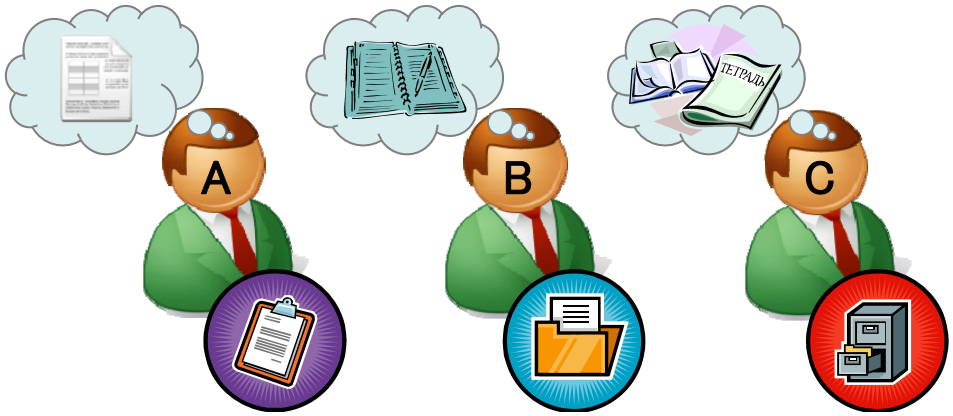


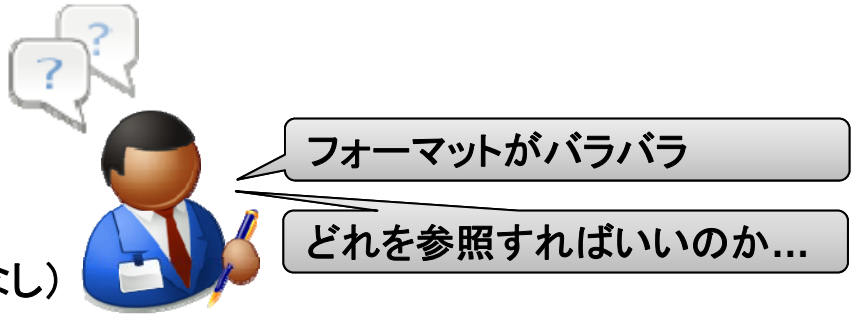
運用・保守体制の確立

③問合せ管理システムの構築
実務経験から得た技術情報を組織的に共有して活用

ナレッジの蓄積

属人化しているノウハウが共有できていない





新人
（実務経験なし）

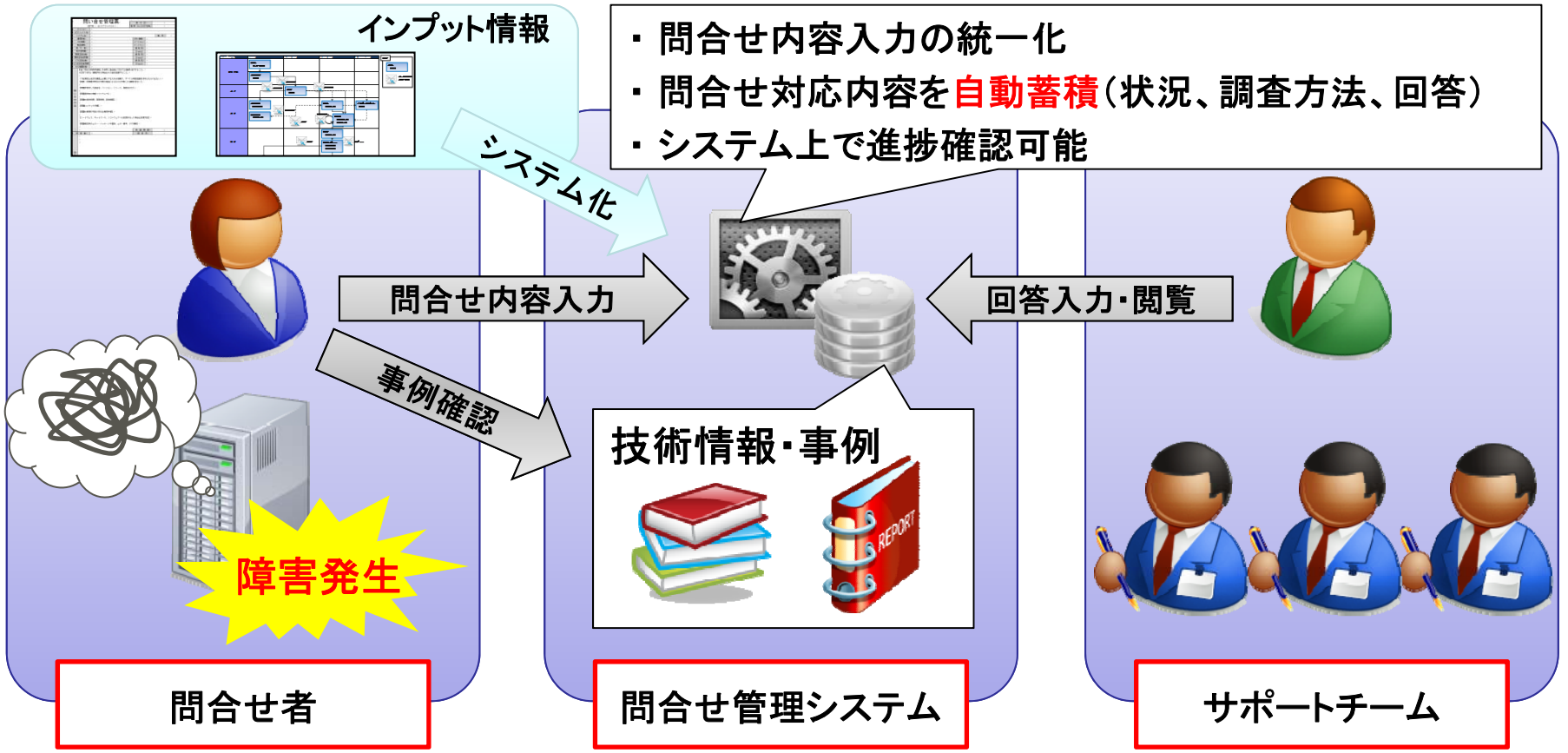
実施内容

- 運用ルールのシステム化検討
- 既存ノウハウの集約
- システム設計／構築

運用・保守体制の確立

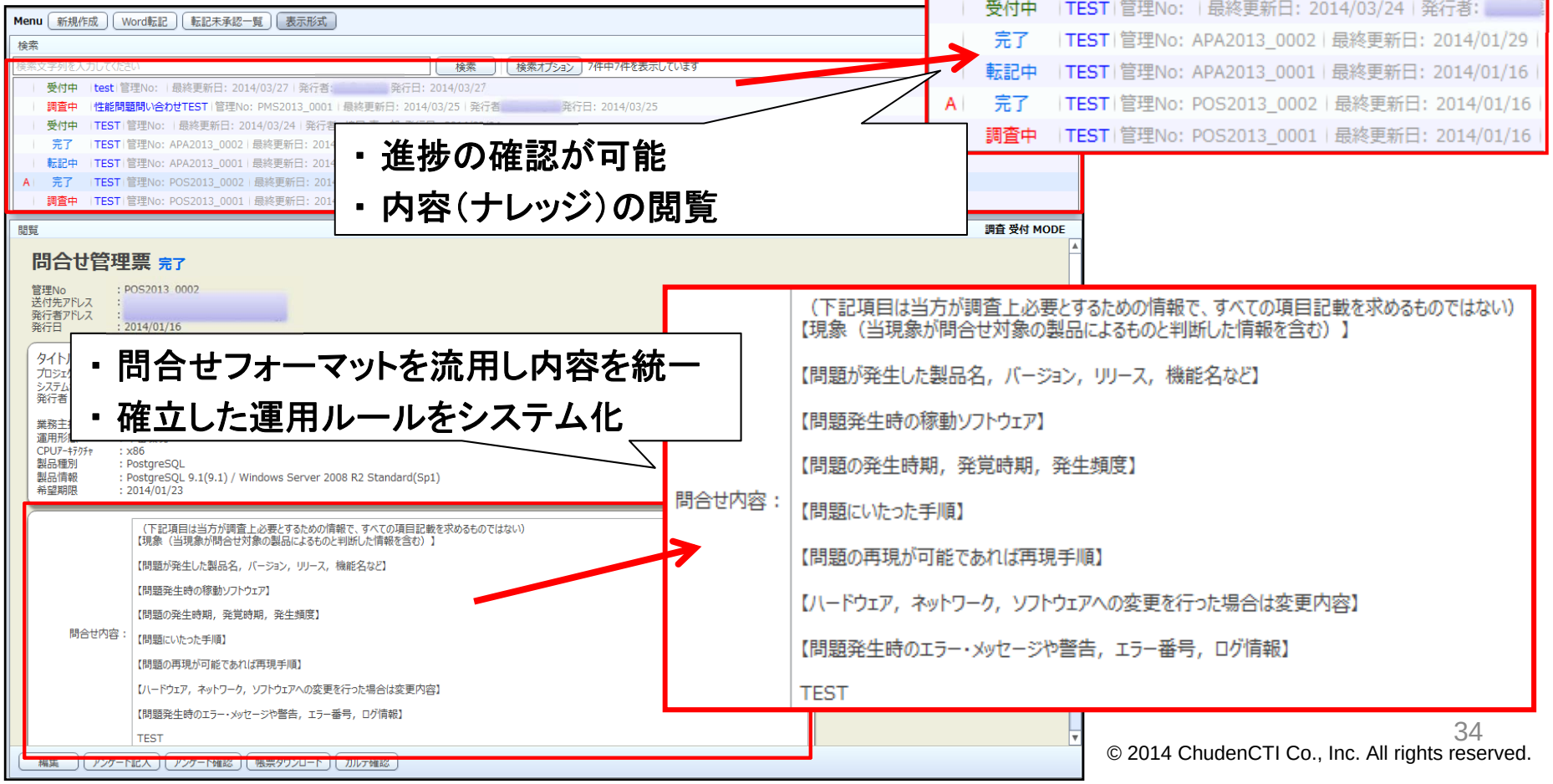
【問合せ管理システムの構築】

業務の中で“必然的”に利用されるシステム構築



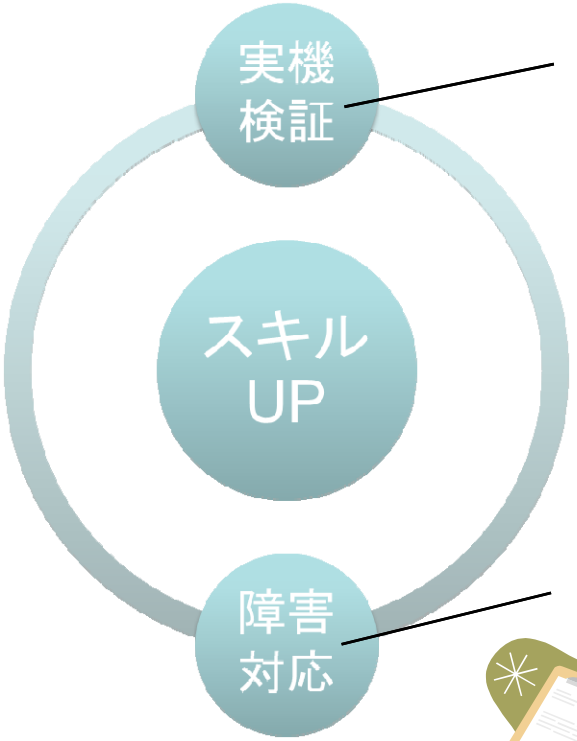
運用・保守体制の確立

【問合せ管理システム】
問合せ管理システムにより問合せ対応を自動蓄積
進捗・問合せ内容を閲覧し、情報共有を図る



運用・保守体制の確立

④教育コンテンツの作成
障害対応事例、また事例以外の障害イベントの調査を基にしたシナリオ形式のコンテンツ作成



技術検証によるスキルUP
再現性　易

障害対応によるスキルUP
再現性　難

疑似体験できる
コンテンツが必要

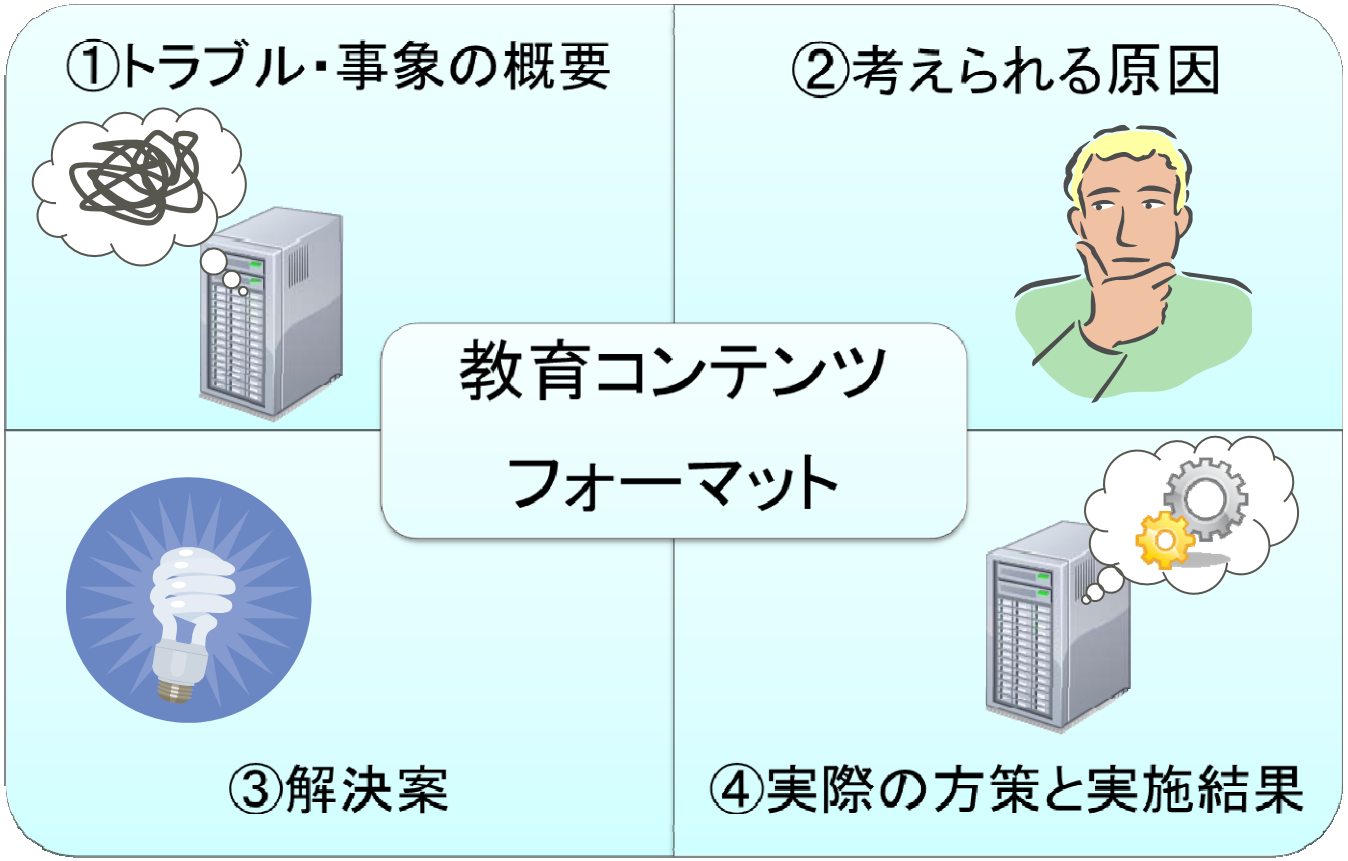
技術力継承

実施内容

- 障害事例の収集
- コンテンツ形式の検討
- 教育コンテンツの作成
- 教育の実施

運用・保守体制の確立

【教育コンテンツのフォーマット検討】
コンテンツのフォーマット（構成）をシナリオ形式にすることで拡張性高いコンテンツを作成

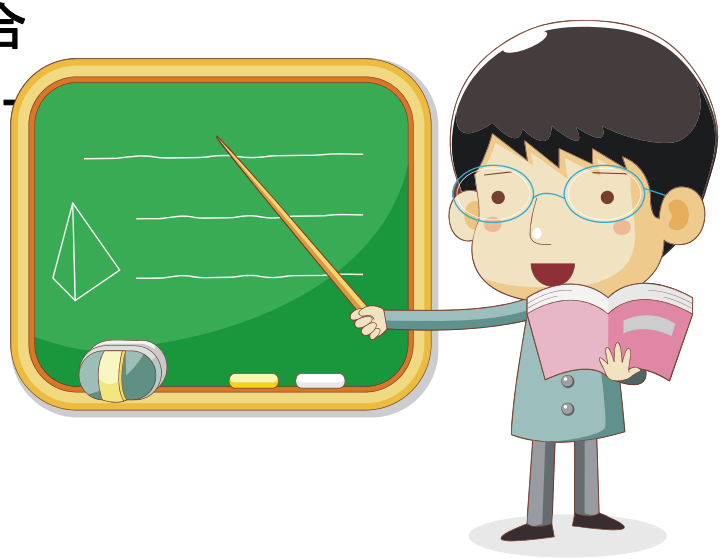


運用・保守体制の確立

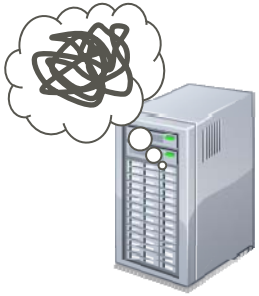
【教育コンテンツの作成】
コンテンツのフォーマット（構成）を基に障害事例を
ケース毎にコンテンツ化

１０事例 ※現在

- Case1 _ アーカイブログによるディスク逼迫
- Case2 _ フェイルオーバ後のデータ更新の重複
- Case3 _ ディスクソートによるSQL遅延
- Case4 _ スレーブにおける参照とリカバリの競合
- Case5 _ エンコーディングの指定ミスによるエラー
- Case6 _ 性能低下を引き起こす処理
- Case7 _ 外字移行トラブル
- Case8 _ メモリ設定過剰による性能低下
- Case9 _ 初期設定からの性能改善
- Case10 _ WAL破損からの復旧



Case3_ディスクソートによるSQLの遅延



・事象：
性能要件で定められたレスポンスタイムでPostgreSQLから
APサーバに応答が返ってこないという事象が発生

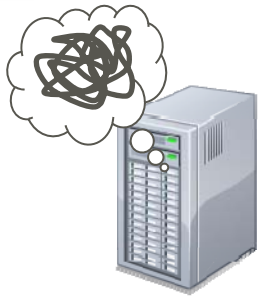
・わかっていること：

１．性能要件
「PostgreSQLからAPサーバへのレスポンスタイムが3秒以内であること」
従って3秒以上時間がかかっているSQLに対策が必要

２．ログの状況
postgresql.logにエラー出力なし

３．APサーバの性能
APサーバのリソース等の使用状況に問題はない

Case3_ディスクソートによるSQLの遅延



・わかっていること(続き):

4. 開発工程

開発工程とその工程で検討すべき性能要件諸元を定義

No	開発工程	検討すべき性能要件諸元
1	企画・要件定義	性能要件の策定、要件に応じてサーバの選定
2	論理設計	テーブル正規化の検討
3	物理設計	ディスク配置、テーブル定義、パーティショニング検討、パラメータ検討
4	運用設計	データベースメンテナンス計画作成
5	アプリケーション開発	SQL文の作成と見直し
6	試験	パラメータの見直し
7	保守・運用	データベースメンテナンスの実行と見直し

今回は、開発工程が完了した後に本事象が発生

上流工程ほど効果的な対策が可能だが、以下の対策を検討(コスト重視)

- SQL文の見直し
- パラメータの見直し

Case3_ディスクソートによるSQLの遅延

SQL遅延が発生する原因

No	要因・可能性	概要
1	実行計画が正しくない	長期間、ANAYZEが実行されない運用が続くと、古い統計情報を見続けることになる。その結果、SQLに対して適切な実行計画が作成されず、SQLの遅延が発生する可能性がある。
2	インデックスが定義されていない	テーブルに必要なインデックスが定義されていないと、必要がないシーケンスキャン(全行スキャン)が実行され、SQLの遅延が発生する可能性がある。
3	ディスクソートの発生	work_memが不足すると、ディスクソートが発生しSQLの遅延が発生する。
4	ロック待ちが発生している	ロック待ちが発生すると、SQLの遅延が発生する可能性がある。
5	SQLの書き方がおかしい	不要な列や行まで取得している処理がある場合、SQLの遅延が発生する。
6	ハードウェアリソースが不足している	メモリやCPU等のハードウェアリソースが不足した場合、SQLの遅延が発生する。

Case3_ディスクソートによるSQLの遅延

1. 実行に3秒以上かかっているSQLをログに出力
   postgresql.confの設定パラメータ log_min_duration_statement を設定
   今回は3秒以上かかるSQLを調べたいので、以下のように設定

```
$ vi $PGDATA/postgresql.conf
log_min_duration_statement = 3s        # -1 is disabled, 0 logs all statements
                                       # and their durations, > 0 logs only
                                       # statements running at least this number
                                       # of milliseconds
(省略)
$ pg_ctl reload
```

設定パラメータ log_min_duration_statement は、PostgreSQLの再起動なしに設定変更することが可能

Case3_ディスクソートによるSQLの遅延

2. 実行に3秒以上かかっているSQLを確認
   設定パラメータ log_min_duration_statement を設定した後、
   APサーバからSQLを発行し、ログを確認

> (ログ抜粋)
> LOG: duration: 8757.642 ms statement: SELECT aid,bid,abalance FROM accounts ORDER BY abalance DESC;

上記のログから、3秒以上かかったSQL文を確認

> SELECT aid,bid,abalance FROM accounts ORDER BY abalance DESC;

問題になっているSQLは、
「テーブルを全件参照し、降順に並び替える(ソート)SQL」
次項以降にSQLのどの部分に時間がかかっているかを確認

Case3_ディスクソートによるSQLの遅延

3. SQLのどの処理に時間がかかっているのかを確認
   EXPLAIN ANALYZEコマンドを実行

```
=# 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)
                        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)
                       ②
Total runtime: 7055.577 ms
(5 rows)
```

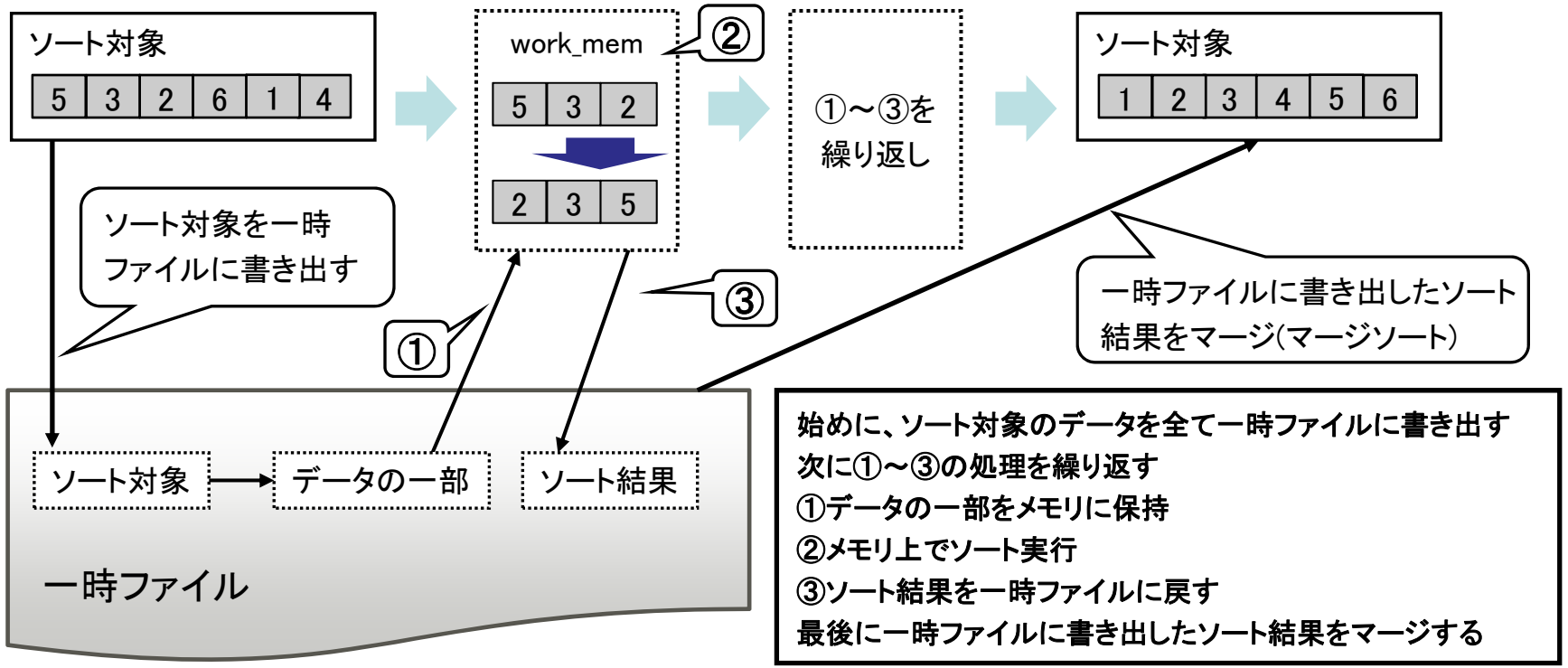
ソート処理で使用するwork_memが不足し、
ディスクへの書き込みが発生しています。

①、②actual time差の4820.318ミリ秒（5755.605 - 935.287）がソート処理

Case3_ディスクソートによるSQLの遅延

4．work_memが不足した場合

通常はメモリ内で実行されるソート処理が、work_mem不足の場合、
一時ファイルを使用するためディスクI/Oが発生し、処理性能が低下



Case3_ディスクソートによるSQLの遅延

・一時ファイルを使用して処理を行ったSQLをログに出力

postgresql.confの設定パラメータ log_temp_files を設定

一時ファイルを使用する全てのSQLをログ出力するため、以下のように設定

```
$ vi $PGDATA/postgresql.conf
log_temp_files = 0                      # log temporary files equal or larger
                                        # than the specified size in kilobytes;
                                        # -1 disables, 0 logs all temp files
(省略)
$ pg_ctl reload
```

設定パラメータ log_temp_files は、PostgreSQLの再起動なしに設定変更することが可能

ログにより、一時ファイルのサイズと、本件で問題となっているSQLを確認

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

Case3_ディスクソートによるSQLの遅延

- 一時ファイルを使用したソート（ディスクソート）が発生した場合、work_memの設定値を大きくする
  work_memの設定値を変更する方法として、以下が考えられる
  1．セッション内（トランザクション内）でwork_memの設定値を大きくする方法
  2．設定ファイル内のwork_memの設定値を大きくする方法

- 1の方法は、行数の多いテーブルのソート処理を行う際に、一時的にwork_memの値を大きくしたい場合に使用

- 2の方法は、すべての処理でwork_memの値を大きくしたい場合に使用

work_memに大きな値を設定すると、スワップが発生する可能性がある
work_memを大きくする解決方法は、消費メモリに影響
既にメモリに空き領域がない場合は、メモリ増設の検討が必要

Case3_ディスクソートによるSQLの遅延

- work_memの最適値の導出

  「Sort Method: external merge Disk」の表示が下記のように、

  「Sort Method: quicksort Memory」になるまで、work_memを増加

```
=# 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)
                    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)
                    ②
 Total runtime: 2588.906 ms
(5 rows)
```

work_memが確保できたため、
メモリ上でソートが実施されました。

3秒以内に短縮できました。

①、②actual time差の1027.056ミリ秒（2038.960 - 1011.904）がソート処理

Case3_ディスクソートによるSQLの遅延

1．セッション内でwork_memの設定値を大きくする方法

work_memを変更するには、SETコマンドを実行

```
=# SET work_mem='300MB';
```

トランザクション内でwork_memの設定値を大きくする方法

work_memを変更するには、SET LOCALコマンドを実行

コミットか、ロールバックによってwork_memの値はセッション内の値に戻る

```
=# BEGIN;
=# SET LOCAL work_mem='300MB';
=# 任意の処理
```

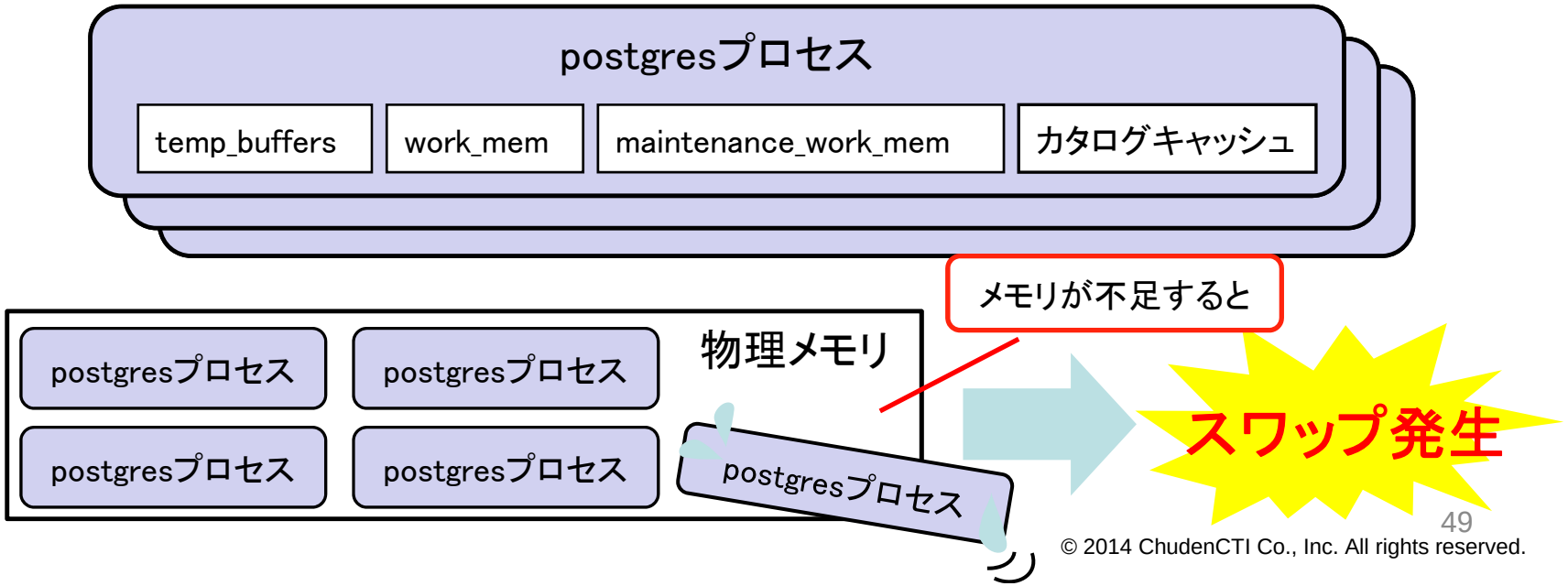
Case3_ディスクソートによるSQLの遅延

2. 設定ファイル内のwork_memの設定値を大きくする方法

```
$ vi $PGDATA/postgresql.conf
work_mem = 300MB                    # min 64kB
(省略)

$ pg_ctl reload
```

work_memはpostgresプロセス毎に確保されるため、postgresプロセスの増加により、メモリを圧迫し、スワップが発生する可能性がある



Case3_ディスクソートによるSQLの遅延

- work_memを使用する実行タイプ

No	分類	実行タイプ	work_memを大きくした場合
1	ソート	Sort	ORDER BYによる明示的なソートや、GROUP BY、DISTINCT、マージ結合等による暗黙的なソートを、一時ファイルを使用せずにメモリ上で処理することができる。(本件の実行タイプ)
2	ハッシュ	Hash	JOIN等によるハッシュ結合で作成するハッシュを、一時ファイルを使用せずにメモリ上で処理できる。
		HashAggregate	GROUP BYによりグループ化した結果を、count、sum等で集約を行う場合に、HashAggregateが選ばれやすくなり、高速に動作する。work_memが不足するとその他の集約方法が選ばれる。
3	ビットマップ	BitmapScan	インデックスが定義された複数の列を参照し、AND、ORで結合する際に、ビットマップがメモリ上に収まると、1行1ビットのビットマップが作成され、高速に動作する。収まらない場合は、1ページ1ビットのビットマップが作成され、取得したページから条件を満たさない行を排除するコストがかかる。

運用・保守体制の確立

①障害時における解析手法の確立
解析に必要な情報、ツールの利用方法や障害解析結果を取り込んだ解析フローを整備

→ 脱・属人化

②運用ルールの整備
現状課題の洗い出しと対策の検討により、運用規定・フローを改善

→ 業務効率化

③問合せ管理システムの構築
実務経験から得た技術情報を組織的に共有して活用

→ ナレッジの蓄積

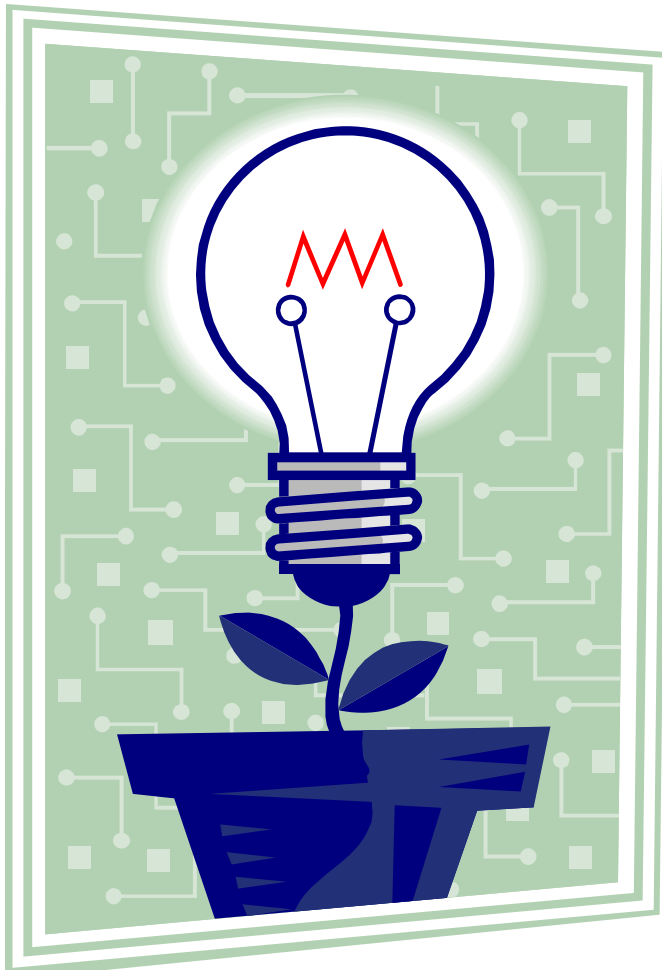
④教育コンテンツの作成
障害対応事例、また事例以外の障害イベントの調査を基にしたシナリオ形式のコンテンツ作成

→ 技術力継承

今後の活動

現在	今後
提供を受ける側で活動	提供する側で活動
・セミナー傍聴 ・事例入手 ・バグ情報、パッチ入手	・セミナーでの情報交換 ・事例紹介 ・バグの発見、フィードバック

今後の活動



新技術の動向にアンテナを張る！！
PostgreSQLのバージョンアップといった技術の
進歩に自分が取り残されないようにする

継続は力なり！！
サポート化への取組、サポート業務の中で
スキルアップ・自信がついた

脱・属人化　＜　自分の価値
組織としては脱・属人化、しかし、自分の価値
（スキル）を磨くことも重要

OSSサポートサービス

当社独自の付加価値のあるサービスをメニュー化し提供する。
当社独自の付加価値サービスが競争優位の源泉となることを狙う。

サービス項目		
OSS保守サービス	問い合わせ対応	技術・障害問い合わせ
		ソース解析・製品活用方法の提示
	基本情報の提供	
	製品の更新版の提供	
	問い合わせ管理の一元化	
OSS維持サービス	個別支援	
	マイナー/メジャーバージョンアップ調査報告	
	機能補完OSSツールに関する調査報告	

個別支援メニューによる「商用ソフトウェアからのOSSへの置き換え支援」を軸としたOSSサービスの積極的な活用をお客様に提案する。

ご静聴ありがとうございました。