
ORACLE®

何から始めるべき？どこまでやるべき？予防保守と運用設計のススメ

池田　大地　プリンシパルコンサルタント

クラウド・テクノロジーコンサルティング統括本部
DBアーキテクト部

以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメント（確約）するものではないため、購買決定を行う際の判断材料になさらないで下さい。オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

本セッションでお伝えしたいこと

「データベース運用における一般的な推奨項目」を85項目に整理しました。

まずはぜひ「データベース運用の点検」を実施してみてください。

- データベースで発生しうる障害は、そのほとんどが計画的な予防保守、各種運用で未然防止可能です。
- 障害を未然防止する、万が一起きてしまった場合も2次被害を防止する(長引かせない)ことが大切です。
- 一般的には、「予防保守」と言うと、パッチ適用、計画メンテナンスをイメージするかと思いますが、さらに「運用設計」を強化することで、より多くの障害を未然防止することが可能となります。
- 本セッションでは、できる限り多くの障害を予防するための必要事項として、「予防保守」を広義の意味で捉え、データベース全体の「運用設計」(運用の強化)に関してご紹介します。

アジェンダ

1. なぜ予防保守／運用管理が必要なのか？
2. データベース安定稼働実現のためのポイント
3. データベース運用の点検を実施しよう
4. データベース運用における一般的な推奨項目一覧
5. まとめ
- Appendix: データベース運用の標準化

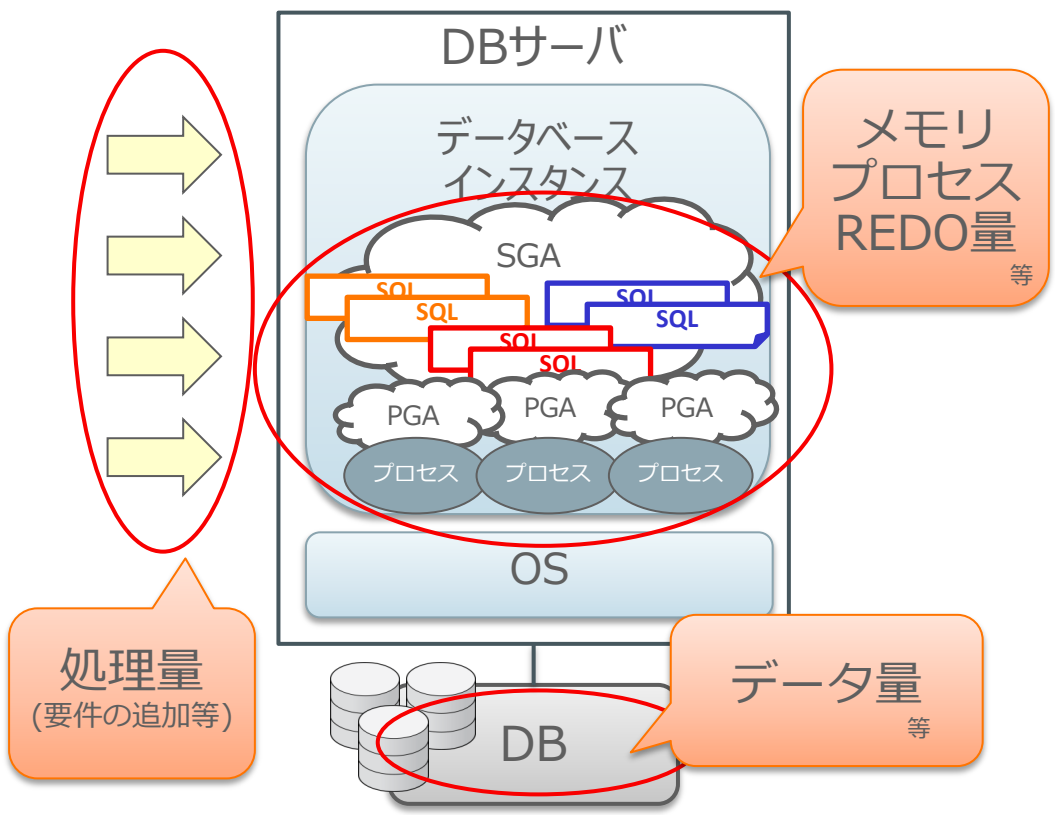
1. なぜ予防保守／運用管理が必要なのか？

なぜ予防保守／運用管理が必要なのか？

- Oracle Databaseはソフトウェア製品ですが、メンテナンスや監視が必要という観点では、エレベータと同じです。
- エレベータを導入したばかりの時は問題無く動きますが、毎日使っていて問題無いからと言って点検やメンテナンスをしなくてよいという訳ではありません。

- 点検やメンテナンスをしないでいると、ある日突然動かなくなったり、エレベータが落ちたりという事故が発生します。事故が起きてからあわてて部品を交換するのではなく、平常時からの定期的な予防保守、運用管理が重要なのです。

データベース運用における変化



防ぐためには・・・

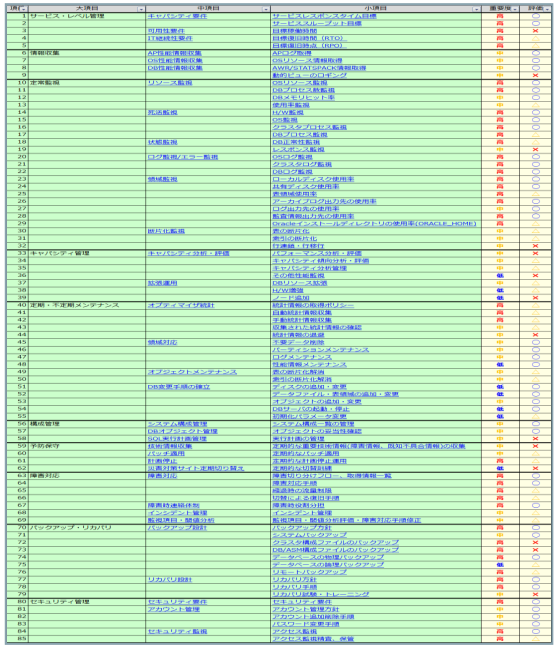
- 予防保守
  - ✓重要技術情報の精査、事前の改善策適用
  - ✓計画的なパッチ適用による既知不具合予防
- 運用設計／運用管理
  - ✓ヘルスチェックによる問題予兆の検知
  - ✓断片化メンテナンス等による障害予防
- 運用設計自体の定期点検
  - ✓システム変更／リプレース等のタイミングで、運用設計から乖離している項目がないか点検する

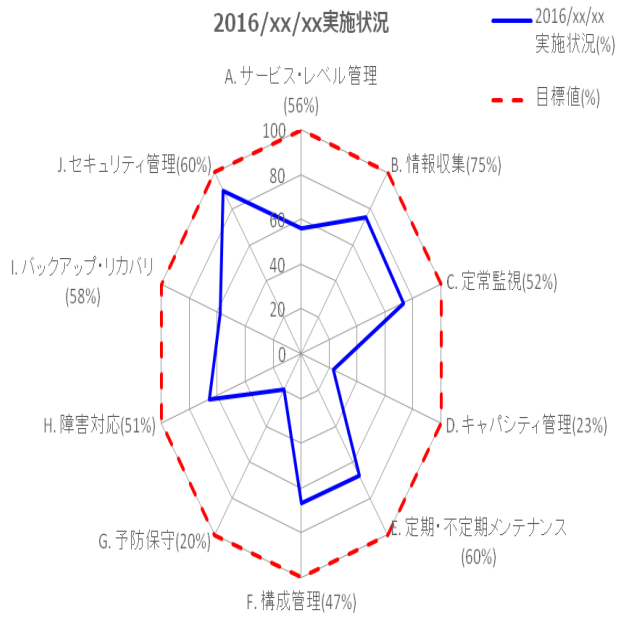
データベースの状態は刻々と変化する
⇒　これまで起きていなかった障害が顕在化する可能性がある

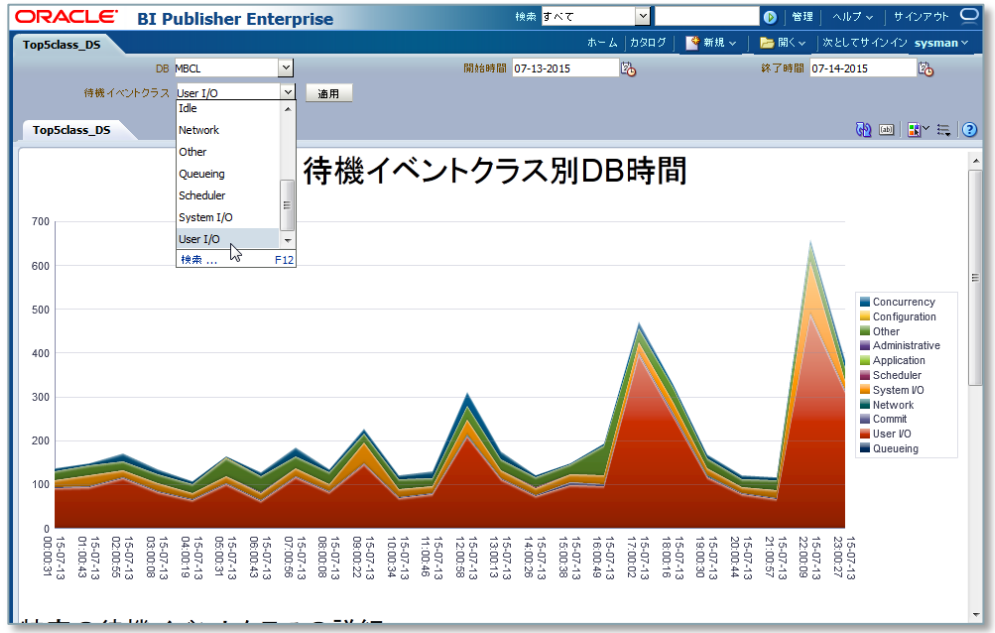
平常時からの定期的な予防保守、運用管理が重要

ではまず何をすればよいのか？

- 「課題の可視化」(見える化)をしましょう。







データベース運用状況を整理・点検し、今実施できている項目、できていない項目を可視化
⇒　運用の強化により障害を予防する

Enterprise Manager (EM)等を用いてデータベースの稼働状況を可視化 (例: ヘルスチェック)
⇒　状況の変化や、問題の予兆を早期に検知し、障害を予防する

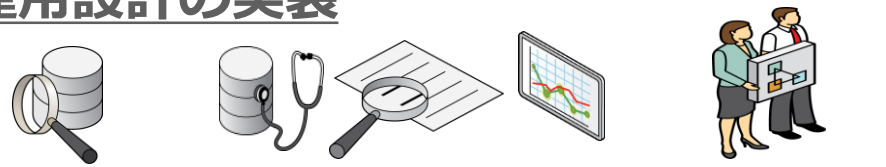
どこまでやるべきか？

一般的にポイントとなる個所は決まっている

ポリシーに基づいた運用設計

運用設計書を作成することで、担当者に依存しない運用が可能！

運用設計の実装



監視　　分析　　メンテナンス

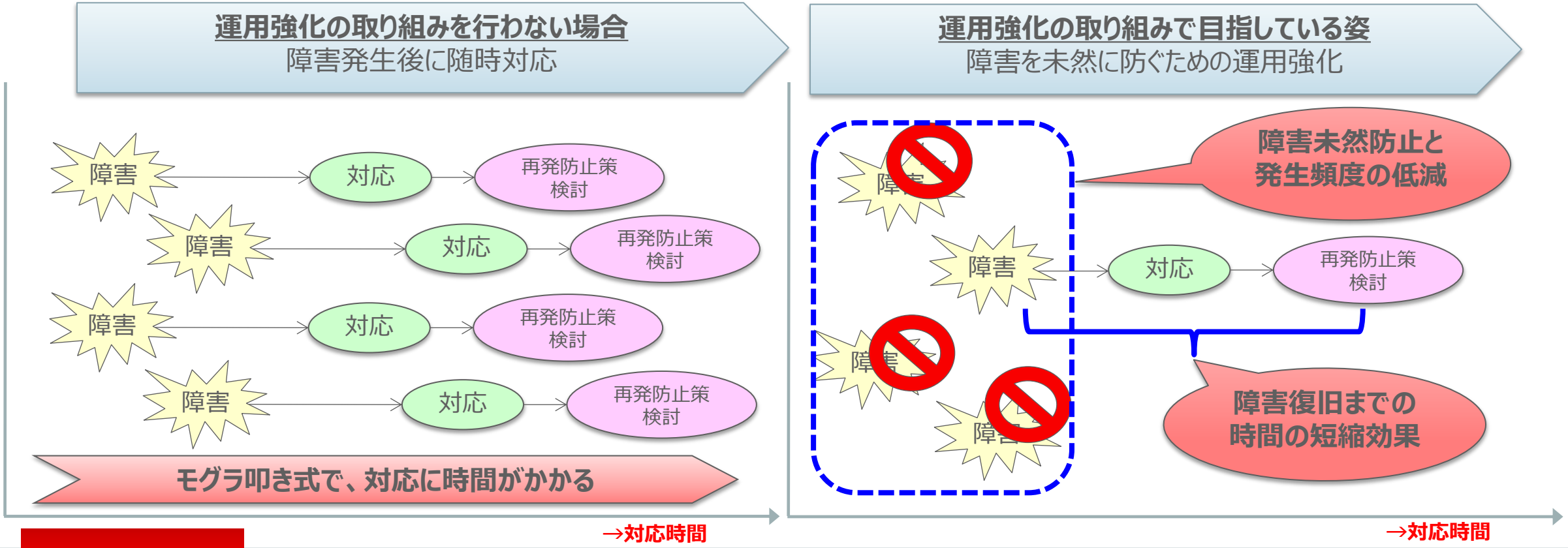
大項目	説明
サービス・レベル管理	キャパシティ要件の定義 等
情報収集	定期的な性能情報収集 等
定常監視	リソース監視、死活監視 等
キャパシティ管理	定期的なヘルスチェック 等
定期・不定期メンテナンス	断片化解消、統計情報取得 等
構成管理	オブジェクト構成管理 等
予防保守	パッチ適用、技術情報収集 等
障害対応	インシデント管理、横展開 等
バックアップ・リカバリ	バックアップ・リカバリ運用
セキュリティ管理	セキュリティ・監査運用

ポイントを考慮／軸にして漏れなく運用を設計することが大事

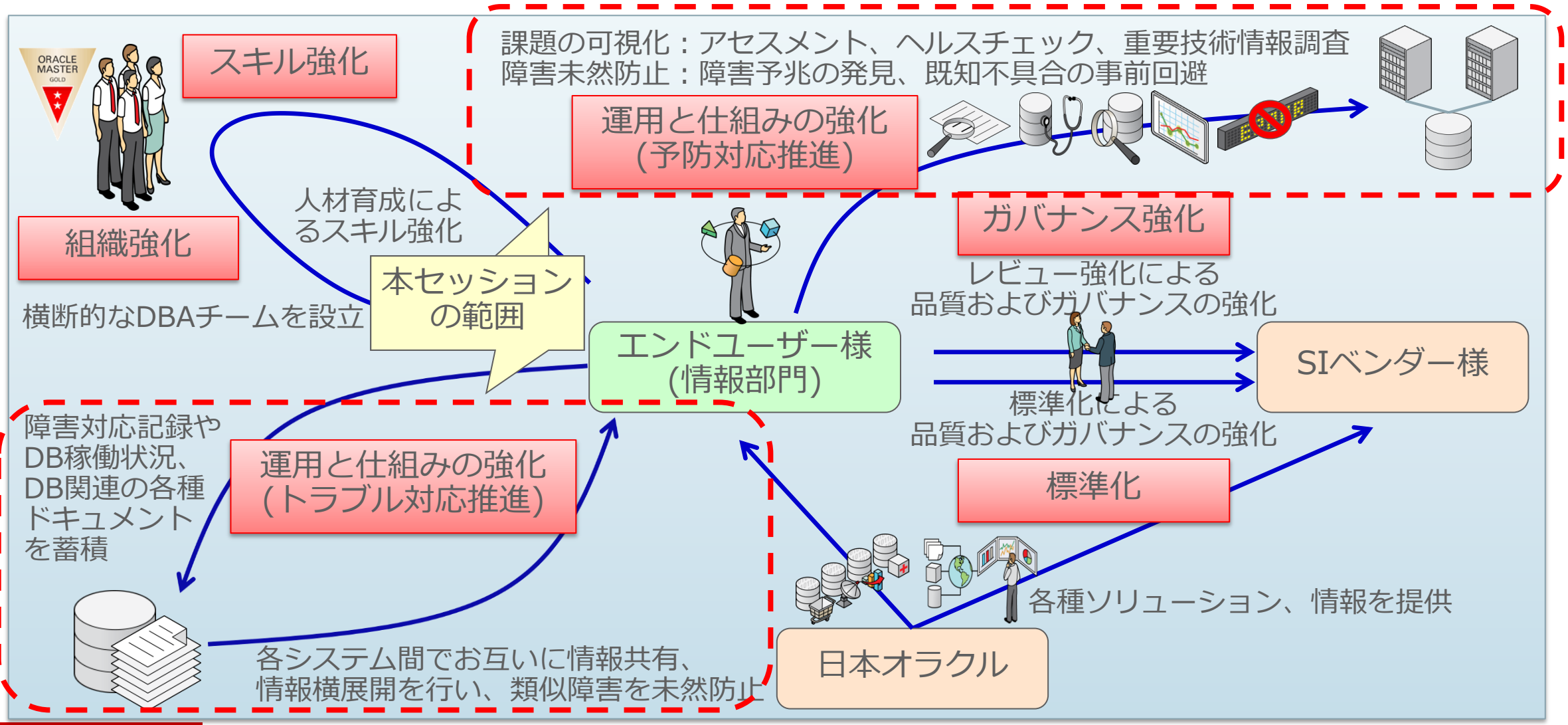
2. データベース安定稼働実現のためのポイント

オラクルコンサルタントが目指すシステム運用

- 安定稼働のためには、障害発生後に問題の早期解決を目指して随時対応を行うのみではなく、障害を未然に防ぐための運用の強化を目指すことが大事です。
- 安定稼働：障害発生を可能な限り予防し、発生した問題が迅速に解決される状態



システム安定稼働の実現に向けたフレームワーク

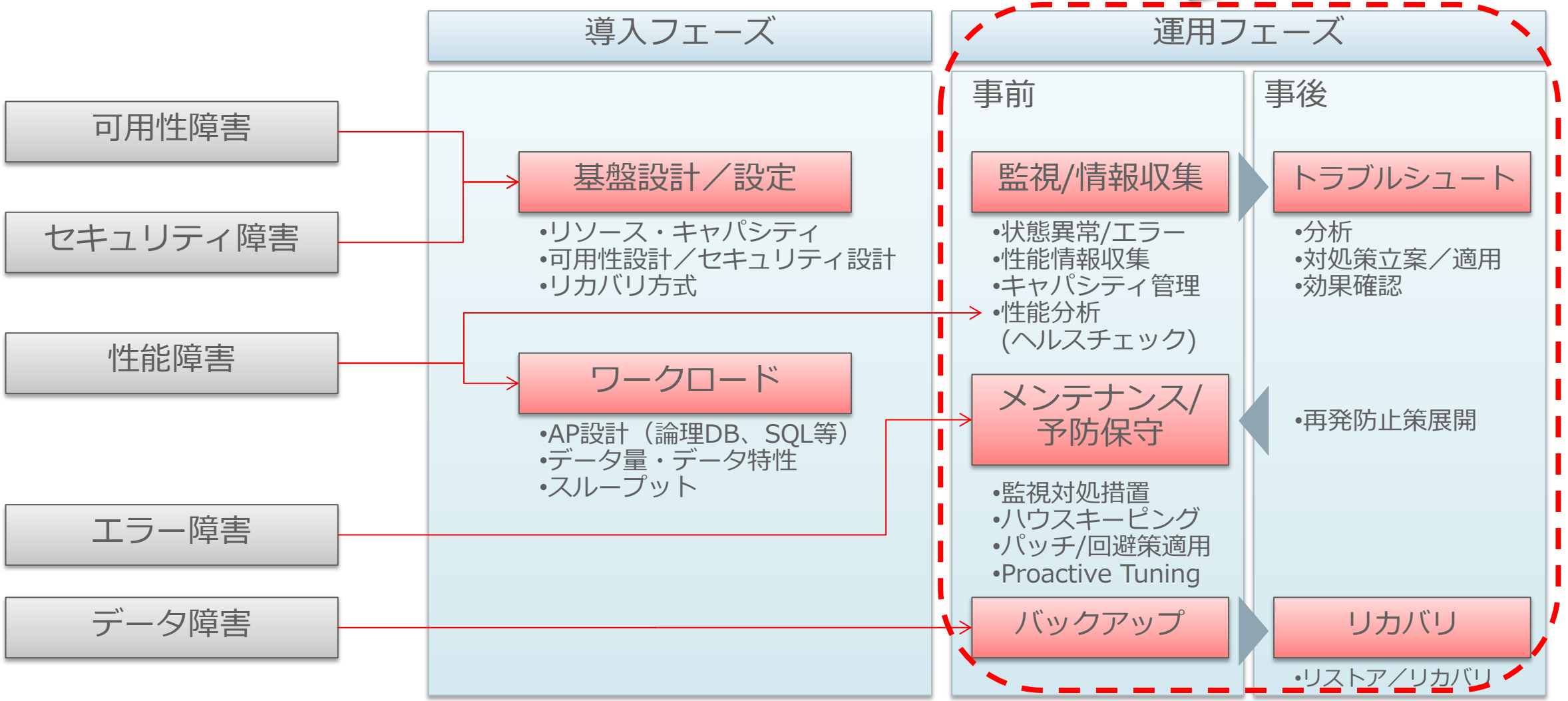


データベースで発生する障害の種類

障害タイプ	説明	例
性能障害	処理のスローダウン ・全体、特定の処理 ・定常的、断続的、一過性	・オンライン処理やシステム間連携処理がタイムアウトする ・バッチ・ウィンドウを突き抜けてオンライン開始できない ・アドホック・クエリーが現実的な時間で返らない
エラー障害	処理がシステム・エラーで失敗する ・全体、特定の処理 ・定常的、断続的、一過性	・リソース確保エラー、デッドロック検出エラー、SW不具合による内部エラーなど
可用性障害	システムダウン(完全利用不可状態) ・サーバ停止、DB停止 ・極端な性能劣化による実質的なダウン	・HW故障、NW障害、OS障害、DB障害などにより、一定期間、システムを利用できない
データ障害	データ不正 ・論理的/物理的なデータ不正 ・データ(トランザクション)・ロスト	・HDD故障、OS/SW不具合などにより、正しいデータが得られない、ユーザーがコミットしたデータが失われる
セキュリティ障害	データ漏えい、改ざん	・悪意を持ったユーザーによる管理者権限での機密情報へのアクセス ・SW脆弱性を利用した攻撃

障害タイプと対策領域(主な原因領域)

運用の強化によりできる限り
多くの障害を予防



※上記は主な原因領域をイメージ化したものです。全てを網羅しているものではありません。例えば、厳密には、性能障害の要因には前提として基盤設計も関連します。

データベース運用の全体概要

全体を網羅することが大事

定常監視

- ✓サーバが正常に起動しているか？（死活監視）
- ✓リソースは不足していないか？（OS/DBリソース監視）
- ✓領域は不足していないか？（領域監視）
- ✓エラーが発生していないか？（OS/DBエラー監視）
- ✓断片化によって効率の悪い状態になっていないか？（断片化監視）

バックアップ・リカバリ

- ✓障害発生時に正常に復旧できるためのバックアップ取得

情報収集

- ✓障害発生時に迅速な原因調査ができるように性能情報を収集（OS/DB/AP）

メンテナンス

- ✓パフォーマンスの要となるオプティマイザ統計の運用
- ✓不要になったデータを定期削除
- ✓非効率的な状態になったオブジェクト（表、索引）を定期的に再編成し、効率的な状態に戻す（断片化解消）
- ✓計画的なパッチ適用（予防保守）

分析・調査

- ✓性能障害の潜在的な要因が無いか、問題予兆がないか調査（キャパシティ管理、ヘルスチェック）
- ✓MyOracleSupport上で公開している重要技術情報を調査（予防保守）
- ✓障害発生時の分析、対処策立案／適用、効果確認、再発防止策の検討

3. データベース運用の点検を実施しよう

データベース運用の点検の進め方

全体の進め方イメージ

✓システム構築時の設計段階で「運用設計」をしっかりと行っておくことが基本。

✓システム変更／リプレース等のタイミングでデータベース運用の点検を実施する。

①データベース運用の点検

運用の課題を可視化
何ができていて、
何ができていないのか整理

②改善点の検討

対応方針の検討
①で未実施とわかった項目について、システムの重要性や運用体制を考慮しつつ、対応の要否や実装方式について検討する

③改善点の実装

運用強化
②で対応が必要と判断した項目について、実際のデータベース運用に組み込む
(例: 監視項目の追加 等)

データベース運用の点検イメージ(1/2)

点検イメージ例

うちのシステムの運用では、何がどこまでできていて、何ができていないのだろう…

大項目	説明
サービス・レベル管理	キャパシティ要件の定義 等
情報収集	定期的な性能情報収集 等
定常監視	リソース監視、死活監視 等
キャパシティ管理	スチェック 等
定期・不定期メン	統計情報取得 等
構成管理	構成管理 等
予防保守	
障害対応	
バックアップ・リ	
セキュリティ管理	セキュリティ・監査運用

各項目の実施状況を点検する

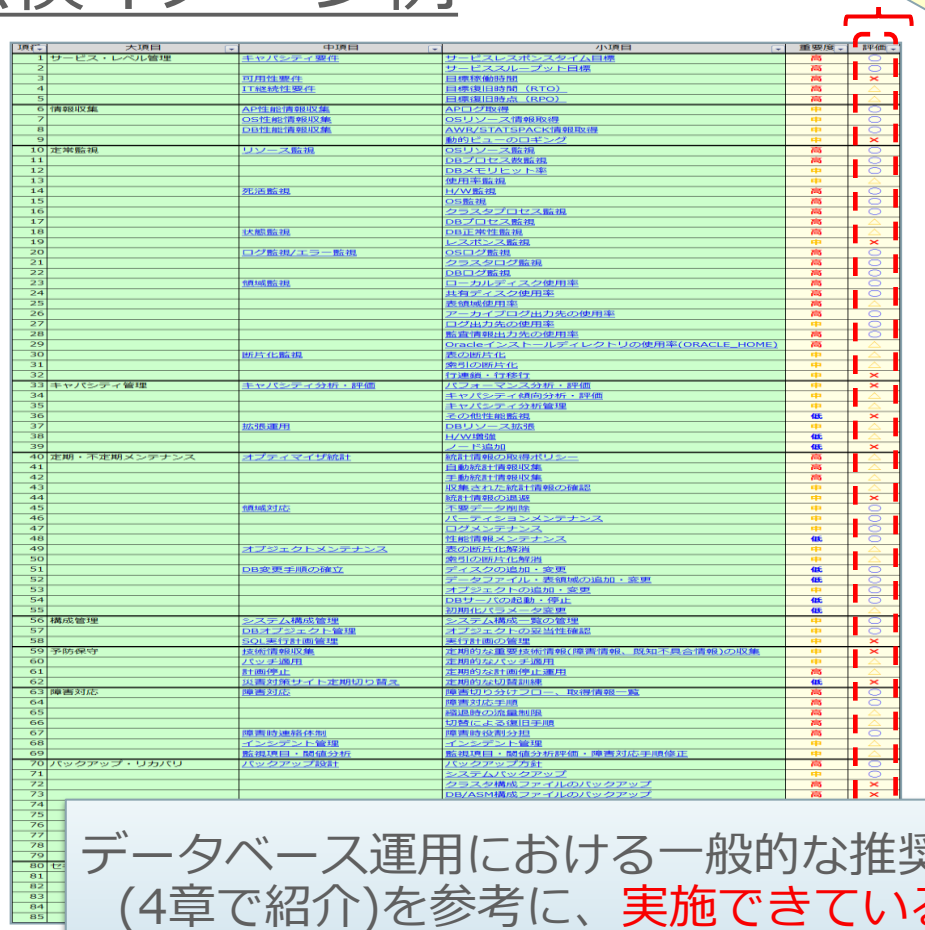
目的は、運用の課題を可視化するため

※補足)：
全ての運用項目を100%とすることがゴールではなく、本質は、システム特性にあう形での運用強化にある。

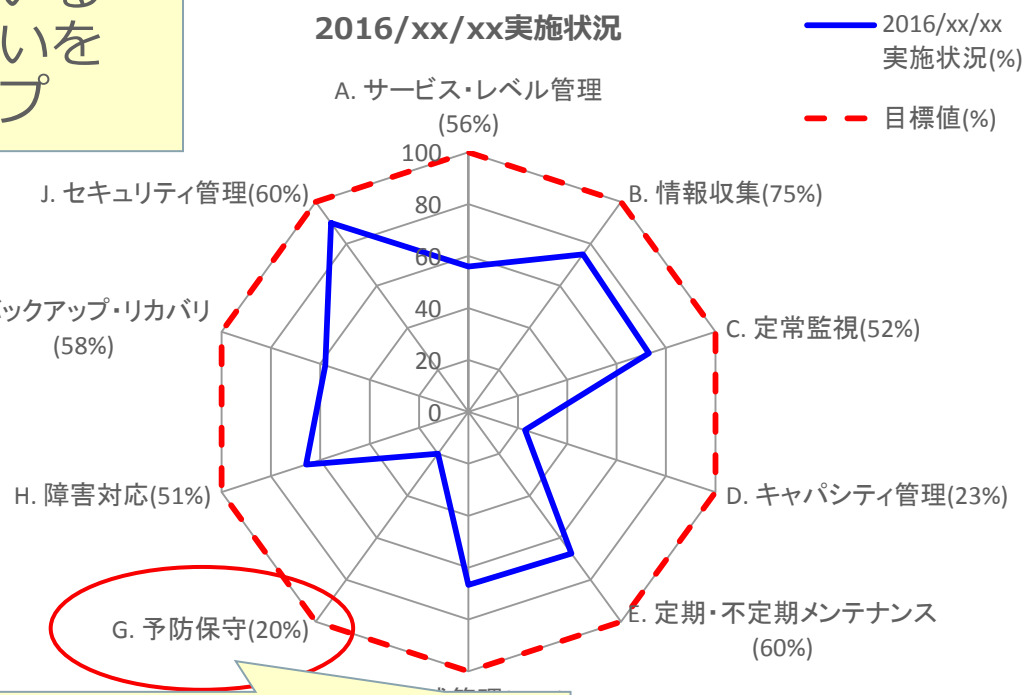
データベース運用の点検イメージ(2/2)

点検イメージ例

実施できている・できていないをリストアップ



グラフ化



弱い部分は改善点を検討する

データベース運用における一般的な推奨項目一覧(4章で紹介)を参考に、実施できている項目、できていない項目を確認する

✓既にできている部分はより一層強化
✓できていない部分は改善点を検討する
(現状は未実施の運用項目を追加導入する等)

4. データベース運用における一般的な推奨項目一覧

データベース運用における一般的な推奨項目一覧

- いくつかのプロジェクト支援におけるノウハウを基に、「データベース運用における一般的な推奨項目」として85項目に簡単に整理しました。
- データベース運用の点検に際しての参考情報としてご活用ください。
- 全般的に共通するポイントは下記の通りです。

✓各種監視について：　障害ケースを想定し網羅的に監視する／必要最低限の項目のみ監視対象にする
※網羅的に監視する一方で、重要でないアラートの発生を減らし、クリティカルなエラーの見落としを防止する目的

✓作業手順について：　実施が想定される作業については、あらかじめ作業手順書を作成しておく
※属人的な運用によるオペレーションミス、作業のばらつきを抑止する目的

✓自動化について：　自動化できるものは極力全て自動化する
※手作業によるオペレーションミス、作業のばらつきを抑止する目的

✓定期的な見直し：　監視項目、キャパシティプラン、運用フロー等、定期的に各項目を見直しする
※最新の運用状況と、あるべき運用設計とが乖離していないか確認する目的

✓可視化の検討：　収集した情報は可視化を検討する(例: Enterprise Managerを利用する)
※分析を効率的・効果的に行う、分かりやすい報告を行う目的

データベース運用における一般的な推奨項目一覧

- 補足:
  - 本推奨項目は、全てのシステムにおいてこのとおり実施することが最善であることを保証するものではありません。参考情報としてご活用ください。
  - 重要度の定義は下記の通りです。

重要度	定義
高	実施しない場合、システム停止もしくは重大な障害が発生し、サービスレベルを満たせない可能性がある項目
中	サービスレベルを保つために、実施することを推奨する項目
低	実施することが望ましいが、サービスレベルが保たれる場合には実施しなくても良い項目

サービス・レベル管理
各種サービスレベルを定義・管理

項番	中項目	小項目	重要度	推奨事項
1	キャパシティ要件	サービスレスポンスタイム目標	高	・サービスレスポンスタイムの目標値を定める (パフォーマンスチューニングの指標として位置づけられる)
2		サービススループット目標	高	・サービススループットの目標値を定める (キャパシティプランニングのベースラインとして位置づけられる)
3	可用性要件	目標稼働時間	高	・目標稼働時間を定める (一定期間においてシステムがどれくらい正常稼働しているかを示す数値)
4	IT継続性要件	目標復旧時間(RTO)	高	・目標復旧時間(RTO:Recovery Time Objective、ここでは最大許容停止時間と同義)を定義する ・DB障害、サイト障害それぞれにおいて定義する ・業務継続の必要性の有無、手動や代案での業務継続も含め、バックアップやリカバリ、災対サイトの構築など、要件に伴い設計が大きく変わる可能性があるため、明確かつ具体的な障害シミュレーションを行う
5		目標復旧時点(RPO)	高	・目標復旧時点(RPO:Recovery Point Objective)を定義する ・DB障害、サイト障害それぞれにおいて定義する ・物理障害、論理障害などによるデータの損失の範囲を定義する ・業務継続の必要性の有無、手動や代案での業務継続も含め、バックアップやリカバリ、災対サイトの構築など、要件に伴い設計が大きく変わる可能性があるため、明確かつ具体的な障害シミュレーションを行う

情報収集
キャパシティ管理、障害対応等のために定常的に「情報収集」を実施(1/2)

項番	中項目	小項目	重要度	推奨事項
6	AP性能情報収集	APログ取得	中	・AP側で処理した業務量や応答時間のログを記録して取得しておく (AP性能情報と、OS/DB性能情報、各種リソース利用状況を関連づけて現状把握を行う際に利用) ・取得情報の例: - 単位時間あたりのオンライントランザクション処理件数 - オンライントランザクションの平均応答時間 - バッチ処理で処理したデータ件数と処理時間 ・過去と比較できるよう、少なくともある程度の期間分(例:1ヶ月～3ヶ月分)を保存する
7	OS性能情報収集	OSリソース情報取得	中	・OSリソース情報(定常監視のOSリソース監視情報)を定期的にロギングする ・取得方法を策定する(例:OSコマンド、監視ツール等) ・情報に応じて、定期的にある程度短い間隔で取得する(例:5秒～10秒間隔で取得) ・過去と比較できるよう、少なくともある程度の期間分(例:1ヶ月～3ヶ月分)を保存する ・取得情報の例: - CPU使用率(USR/SYS) - CPUランキュー待ちの数 - メモリ使用率/使用状況/PageTable使用状況 - DISK I/O状況 - ネットワーク関連統計(パブリックネットワーク／インターコネクト) ・必要に応じて、CPU情報は各CPU毎の統計情報を取得する ・定期的に取得する性能情報には日付情報を付与してロギングする (例:vmstatコマンドの各出力行に日付を付与する) ・過去分のログ削除は自動化する

情報収集
キャパシティ管理、障害対応等のために定常的に「情報収集」を実施(2/2)

項番	中項目	小項目	重要度	推奨事項
8	DB性能情報収集	AWR/STATSPACK情報取得	中	・DB性能情報(AWR/STATSPACK)を定期的に収集する ・30分～1時間間隔でスナップショットを取得する ・必要に応じて取得タイミングを検討する (例:定時起動処理と重複しないよう、8:30、9:30、10:30…のように毎時30分にスナップショットを取得) ・過去と比較できるよう、少なくともある程度の期間分(例:1ヶ月～3ヶ月分)を保存する ・STATSPACKの場合、Level7で取得する(実行計画、セグメント統計を取得する) ・過去分の性能情報削除は自動化する
9		動的ビューのロギング	中	・セッション情報、DB性能情報を定期的にロギングする ・取得方法を策定する(例:ASH情報、SQLスクリプトを作成する等) ・情報に応じて、定期的にある程度短い間隔で取得する(例:5秒～数分間隔で取得) ・過去と比較できるよう、少なくともある程度の期間分(例:1ヶ月～3ヶ月分)を保存する ・取得情報の例: - セッション情報(V$SESSION、V$PROCESS) - システム統計(V$SYSSTAT) - 待機イベント情報(V$SYSTEM_EVENT) - ロック情報(V$LOCK) - リソース情報(V$RESOURCE_LIMIT) - 一時表領域の使用状況(V$SORT_SEGMENT) ・定期的に取得する性能情報には日付情報を付与してロギングする(例:SYSDATEを付与する) ・過去分のログ削除は自動化する

定常監視
障害、および、障害の予兆を迅速に検知するための「リソース監視」(1/2)

項番	中項目	小項目	重要度	推奨事項
10	リソース監視	OSリソース監視	高	・OSリソースを監視する(※) ・監視対象の例: - CPU: CPU使用率(全体、CPU毎、USR/SYS毎の使用率)、ランキュー待ちの数(全体/CPU毎) - メモリ: 使用率、空きメモリの量、スワップイン/スワップアウトの状況 - ディスクI/O: ディスクビジー率、IOPS、I/O待ち時間、平均I/Oキュー数 - ネットワーク: トラフィック量、エラー系統計 ・監視方法、閾値超過時の通知方法を策定する(例:EMによる監視、監視ツール等) ・定期的にある程度短い間隔で監視する (例:CPU、メモリ、ディスクI/Oは数秒～10秒間隔、ネットワークは1分間隔) ・監視閾値を設定する(例:警告80%、エラー90%) ・警告やエラーの通知は、例えば、定期的に監視し、数回連続して閾値を超えた場合に行う (例:CPU使用率90%以上を2分間以上継続している、かつ、1CPU(1CPUコア)あたりのランキュー待ちプロセス数が2を超えている) ・閾値を超えた場合のアクションをあらかじめ設定しておく

(※) 使用率の推移については、中長期的な期間分をロギングし保存する(キャパシティ分析に利用)。

定常監視
障害、および、障害の予兆を迅速に検知するための「リソース監視」(2/2)

項番	中項目	小項目	重要度	推奨事項
11	リソース監視	DBプロセス数監視	高	・DBプロセス数、セッション数を定期的に監視する(※) (例: v$resource_limitから、パラメータsessions、processesに近づいていないか監視する) ・監視方法、閾値超過時の通知方法を策定する(例:EMによる監視、監視ツール等) ・監視間隔は、システム特性や接続形態に応じて設定する(例:10秒間隔、15分間隔) ・監視閾値を設定する(例:上限パラメータ値の80%を警告、90%をエラーとする等) ・必要に応じて、アクティブセッション数についても定期的に監視する
12		DBメモリヒット率監視	中	・メモリヒット率の監視を行う(※) (例: バッファキャッシュ、ライブラリキャッシュ、PGA等) ・監視方法、閾値超過時の通知方法を策定する(例:EMによる監視、監視ツール等) ・監視間隔は、システム特性に応じて設定する(例:1時間間隔) ・警告やエラーの通知は、例えば、1回でも閾値を超過した場合に行う、あるいは、定期的に監視し、数回連続して閾値を超えた場合に行う(ベースラインやシステム特性に応じて策定する)
13		DBリソース監視	中	・環境の特性にあったリソース監視を行う(※) (例: 自動SGA環境における共有プールの拡張余力の監視、共有プールのIMMEDIATE拡張発生の監視、オープン・カーソル数制限使用率の監視) ・監視方法、閾値超過時の通知方法を策定する(例:EMによる監視、監視ツール等) ・監視間隔は、システム特性に応じて設定する(例:1時間間隔) ・警告やエラーの通知は、例えば、1回でも閾値を超過した場合に行う、あるいは、定期的に監視し、数回連続して閾値を超えた場合に行う(ベースラインやシステム特性に応じて策定する)

(※) 使用率の推移については、中長期的な期間分をロギングし保存する(キャパシティ分析に利用)。
DBプロセス数、DBメモリヒット率はAWR/STATSPACKで代替可能。

定常監視
障害を迅速に検知し、二次障害を防ぐための「死活監視」

項番	中項目	小項目	重要度	推奨事項
14	死活監視	H/W監視	高	・基本的な全ての主要部品の死活監視を実施する (例:電源、システムボード、CPU、メモリ、ディスクコントローラ、HDD、NIC、スイッチ) ・監視/通知方法を策定する(例:SNMPトラップを利用する等) ・監視間隔を設定する(※)
15		OS監視	高	・OSの死活監視を実施する ・監視/通知方法を策定する (例:EMによる監視、PINGコマンドに対して規定回数の応答がない場合にエラーとする等) ・監視間隔を設定する(※)
16		クラスタプロセス監視	高	・クラスタの主要プロセスの死活監視を実施する(例:OHAS等の主要プロセス、CRSリソース等) ・監視/通知方法を策定する(例:EMによる監視、OSプロセスの存在/数を定期確認する等) ・監視間隔を設定する(※)
17		DBプロセス監視	高	・DBの主要プロセスの死活監視を実施する (例:DBのバックグラウンドプロセス、リスナー、ASMのバックグラウンドプロセス等) ・監視/通知方法を策定する(例:EMによる監視、OSプロセスの存在/数を定期確認する等) ・監視間隔を設定する(※)

(※)監視間隔は、RTO要件を満たすように設計する。
「目標復旧時間」から「復旧に掛かる時間」を引いた時間が「検知に掛けられる時間」となる。これを踏まえ、例えば、「目標復旧時間」が1時間で、復旧作業(システム切替等)に30分掛かる場合、障害発生から30分以内に検知できるように、監視頻度を設計する。

定常監視
障害を迅速に検知し、二次障害を防ぐための「状態監視」

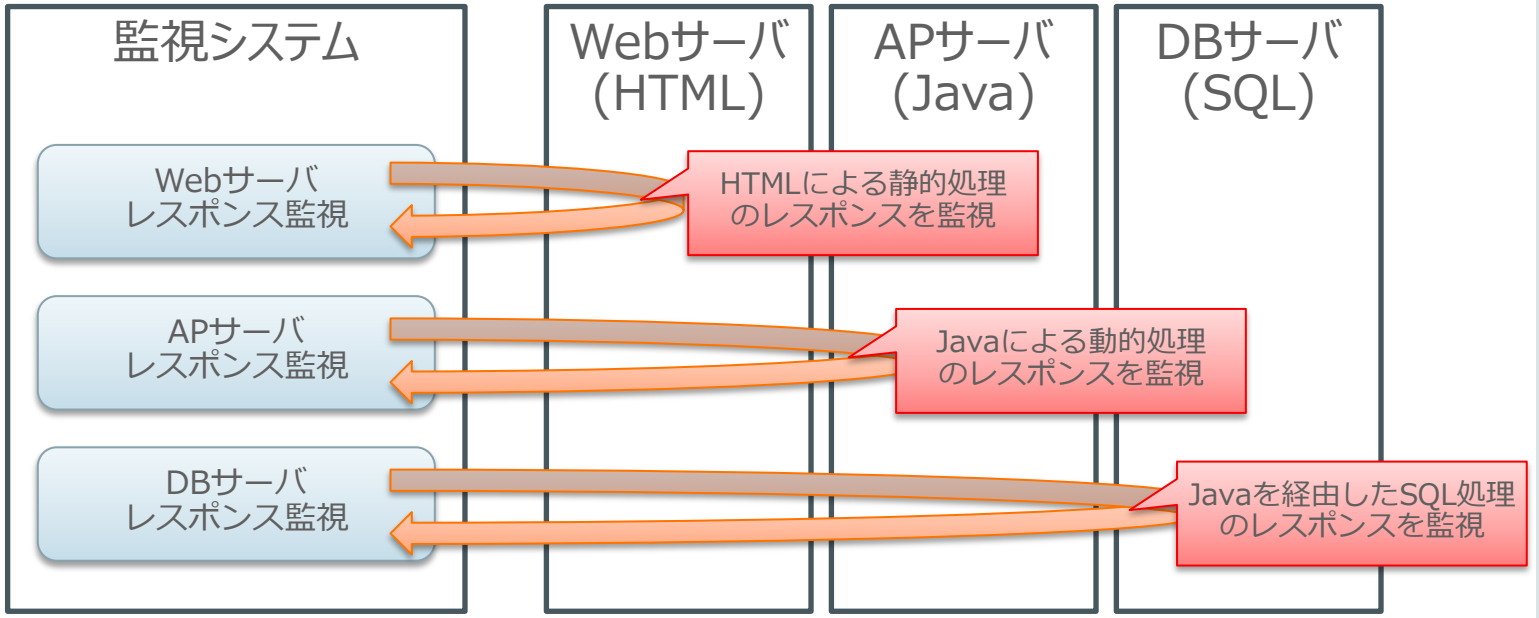
項番	中項目	小項目	重要度	推奨事項
18	状態監視	DB正常性監視	高	・DBに定期的にリスナー経由で接続、SQLを発行し、正常性を確認する ・監視/通知方法を策定する(例:EMによる監視、ツールによる監視等) ・監視間隔を設定する(例:5秒～10秒間隔)
19		レスポンス監視	中	・レスポンス監視(※)の仕組みを導入し、定期的に性能監視を実施する ・監視/通知方法を策定する(例:ツールによる監視等) ・監視間隔を設定する(例:5秒～10秒間隔)

(※)レスポンス監視

性能障害が発生した際に、どのサーバで性能問題が発生しているかを迅速に切り分けるための監視。

例えば、Webサーバ、APサーバ、DBサーバで構成しているシステムの場合は、右の図のように各サーバのレスポンス監視を用意し、定期的にレスポンスを取得することで、性能問題が発生しているサーバを容易に特定することができる。

レスポンス監視を導入することで、OSリソース以外がネックとなっているケース(例:APサーバのコネクション数不足やDBの内部ロック競合等)でも被疑サーバを絞り込むことができる。



定常監視
障害、および、障害の予兆を迅速に検知するための「ログ監視」

項番	中項目	小項目	重要度	推奨事項
20	ログ監視/エラー監視	OSログ監視	高	・OSシステムログ、エラーログを監視する ・監視対象のメッセージを選定する(例:警告やエラー等) ・監視間隔を設定する(例:常時監視)
21		クラスタログ監視	高	・クラスタログを監視する(例:OHAS、CSS、CRS等) ・監視対象のメッセージを選定する(例:エラー等) ・監視間隔を設定する(例:常時監視)
22		DBログ監視	高	・DBのアラートログ、ASMのアラートログ、リスナーログを監視する ・監視対象のメッセージを選定する (例:ORAエラー、OSエラー、REDOログスイッチ遅延等の各種メッセージ等) ・ORA-エラーは原則として全てのエラーを監視対象とする (システムとして無視可能なものは運用試験、実運用等を経て判断する) ・アラートログに出力されないユーザーエラー(例:ORA-1等)は、アプリケーション側のログに出力する必要があり、必要に応じてアプリケーション側のログも定期的に監視する ・監視間隔を設定する(例:常時監視)

定常監視
領域不足になる前に対応を行うための「領域監視」(1/2)

項番	中項目	小項目	重要度	推奨事項
23	領域監視	ローカルディスク使用率	高	・全てのローカルディスク領域の使用率を監視する(※) ・監視方法、閾値超過時の通知方法を策定する(例:EMによる監視、監視ツール等) ・監視間隔を設定する(例:5分間隔、15分間隔等) ・監視閾値を設定する(例:警告80%、エラー90%) ・閾値を超えた場合のアクションをあらかじめ設定しておく
24		共有ディスク(ASMディスクグループ)使用率	高	・全ての共有ディスク領域(例:全てのASMディスクグループ等)の使用率を監視する(※) ・監視方法、閾値超過時の通知方法を策定する(例:EMによる監視、監視ツール等) ・監視間隔を設定する(例:5分間隔、15分間隔等) ・監視閾値を設定する(例:警告80%、エラー90%) ・閾値については、必要に応じてデータファイルの自動拡張設定も考慮し設定する ・閾値を超えた場合のアクションをあらかじめ設定しておく
25		表領域使用率	高	・全ての表領域の使用率を監視する(※) ・監視方法、閾値超過時の通知方法を策定する(例:EMによる監視、監視ツール等) ・各種表領域の特性に見合った監視を行う(例:ユーザー表領域、一時表領域、UNDO表領域等) ・監視間隔を設定する(例:15分間隔、1時間間隔等) ・監視閾値を設定する(例:警告80%、エラー90%) ・閾値については、必要に応じてデータファイルの自動拡張設定も考慮し設定する ・閾値を超えた場合のアクションをあらかじめ設定しておく

(※) 使用率の推移については、中長期的な期間分をロギングし保存する(キャパシティ分析に利用)。

定常監視
領域不足になる前に対応を行うための「領域監視」(2/2)

項番	中項目	小項目	重要度	推奨事項
26	領域監視	アーカイブログ出力先の使用率	高	・全てのアーカイブログ出力先領域の使用率を監視する(※) ・監視方法、閾値超過時の通知方法を策定する(例:EMによる監視、監視ツール等) ・監視間隔を設定する(例:5分間隔、15分間隔等) ・監視閾値を設定する(例:警告80%、エラー90%) ・閾値を超えた場合のアクションをあらかじめ設定しておく
27		ログ出力先の使用率	中	・全てのログ出力先(例:DB、クラスタのログ)の使用率を監視する(※) ・監視方法、閾値超過時の通知方法を策定する(例:EMによる監視、監視ツール等) ・監視間隔を設定する(例:5分間隔、15分間隔等) ・監視閾値を設定する(例:警告80%、エラー90%) ・閾値を超えた場合のアクションをあらかじめ設定しておく
28		監査情報出力先の使用率	高	・監査情報(監査ログ)出力先の使用率を監視する(※) ・監視方法、閾値超過時の通知方法を策定する(例:EMによる監視、監視ツール等) ・監視間隔を設定する(例:5分間隔、15分間隔等) ・監視閾値を設定する(例:警告80%、エラー90%) ・閾値を超えた場合のアクションをあらかじめ設定しておく
29		Oracleインストールディレクトリの使用率(ORACLE_HOME)	高	・Oracleインストールディレクトリ(ORACLE_HOME)の使用率を監視する(※) ・監視方法、閾値超過時の通知方法を策定する(例:EMによる監視、監視ツール等) ・監視間隔を設定する(例:5分間隔、15分間隔等) ・監視閾値を設定する(例:警告80%、エラー90%) ・閾値を超えた場合のアクションをあらかじめ設定しておく

(※) 使用率の推移については、中長期的な期間分をロギングし保存する(キャパシティ分析に利用)。

定常監視
性能劣化要因を早期検出するための定期的なオブジェクトの「断片化監視」

項番	中項目	小項目	重要度	推奨事項
30	断片化監視	表の断片化	低	・表オブジェクトの肥大化／断片化を監視する(※) ・対象を選定する(例:データの更新や削除が頻繁にある表(特にOLTP表)) ・監視／確認方法を策定する(例:DBMS_SPACE.SPACE_USAGEを使用) ・監視／確認間隔を設定する(例:xx日毎、1カ月毎等) ・監視閾値を設定する(例:サイズがxxを超えており、かつ、断片化率がxx%超える)
31		索引の断片化	低	・索引オブジェクトの肥大化／断片化を監視する(※) ・対象を選定する(例:データの更新や削除が頻繁にある表(特にOLTP表)に付与された索引) ・監視／確認方法を策定する(例:ANALYZE INDEX VALIDATE STRUCTURE文を使用) ・監視／確認間隔を設定する(例:xx日毎、1カ月毎等) ・監視閾値を設定する(例:サイズがxxを超えており、かつ、削除リーフ行の割合が20%超える) ・ANALYZE文による確認についてはロック競合に注意する
32		行連鎖・行移行	低	・表オブジェクトの行連鎖／行移行を監視する(※) ・対象を選定する(例:列数が多く最初はNULL値がINSERTされ後からUPDATEされるような表 等) ・監視／確認方法を策定する(例:ANALYZE文を使用しDBA_TABLES等のCHAIN_CNTを確認) ・監視／確認間隔を設定する(例:xx日毎、1カ月毎等) ・監視閾値を設定する(例:検証により性能劣化が顕在化する閾値を確認する等) ・アプリケーションの特性や業務観点から、行連鎖や行移行が発生しないことが分かっている場合は、一度だけ確認し、その後定期的には確認しないという運用もあり

(※)この監視運用は、確認処理／構造検査においてパフォーマンス劣化を伴うことがあるため、パフォーマンス要件とサービス稼働要件において重視する内容によっては実施しない。

キャパシティ管理
稼働状況の「見える化」、問題の予兆検知のための「ヘルスチェック」

項番	中項目	小項目	重要度	推奨事項
33	キャパシティ分析・評価	パフォーマンス分析・評価(ヘルスチェック)	中	・OS性能情報、DB性能情報(例:AWR)等から、システムのヘルスチェック(パフォーマンス分析・評価・報告)を行い、稼働時のワークロード／傾向の把握、問題予兆の検知を行う ・定期的にヘルスチェックを実施する(例:1カ月毎、4半期毎等) ・過去のワークロードからベースラインを設定し、そこから乖離のあるデータがあった場合には原因を確認し、必要に応じて対処を検討／実施する ・分析の指標となる閾値を定める ・閾値自体の分析／見直しを適宜行う ・短期的な分析・評価と、一定期間の分析・評価を行うことで徐々に値が増加していないか等の変化、問題の予兆を検知する (例:月次での閾値評価と、4半期毎での傾向分析 等) ・分析対象の例: - CPU使用率(OS/DB) - メモリ使用率 - DBのワークロード(セッション数・SQL実行回数・REDO生成量・読み込みブロック数 等) - DBのメモリ統計(メモリヒット率・サイズの推移・IMMEDIATE拡張の有無 等) - DBの待機イベント情報 - DBのI/O統計(I/Oレスポンス 等) - 高負荷SQLの処理時間

キャパシティ管理
傾向分析、将来予測による「キャパシティ管理」(1/2)

項番	中項目	小項目	重要度	推奨事項
34	キャパシティ分析・評価	キャパシティ傾向分析・評価	中	・取得済みの各種情報から、各リソースの使用量の増減の傾向分析・評価を行う ・定期的に実施する(例:1カ月毎、4半期毎等) ・分析の指標となる閾値を定める(例:警告80%、エラー90%) ・閾値自体の分析／見直しを適宜行う ・RAC構成の場合、縮退時の想定使用量に耐えられるよう閾値を定める ・短期的な分析・評価と、一定期間の分析・評価を行うことで徐々に値が増加していないか等の変化、問題の予兆を検知する (例:月次での閾値評価と、4半期毎での傾向分析 等) ・分析対象の例: - CPU使用率 - メモリ使用率 - ローカルディスク使用率 - 共有ディスク(ASMディスクグループ)使用率 - 表領域使用率 - DBプロセス数(セッション数) ・アプリを新規にリリースする場合等には、その前後でもキャパシティを調査し分析・評価を行う ・閾値を超えた場合のアクションをあらかじめ設定しておく
35		キャパシティ分析管理	中	・将来的なリソース不足を事前に予測するための情報収集、回帰分析、計画を行う ・定期的なキャパシティ分析の報告会を実施する ・分析結果が関係者に共有される仕組みを確立する

キャパシティ管理
傾向分析、将来予測による「キャパシティ管理」(2/2)

項番	中項目	小項目	重要度	推奨事項
36	キャパシティ分析・評価	その他性能監視	低	・その他、環境に即した性能監視を行う(※) ・監視対象に応じて、監視方法、監視間隔、監視閾値を設定する ・監視の例: - 自動SGA環境における共有プールの拡張余力の推移 - 共有プール断片化監視 - DBインスタンスのレスポンス(ログオン時間) - キャッシュフュージョンのレスポンス - 待機時間割合の変化 - 表領域の断片化監視

(※)環境に応じて、重要度は変化するので留意する。(利用システムの特性によっては重要度:高として重点的に監視する必要がある場合も考えられる)

キャパシティ管理
計画的な「拡張運用」(1/3)

項番	中項目	小項目	重要度	推奨事項
37	拡張運用	DBリソース拡張 (1/2)	中	・定期的にキャパシティ傾向分析を行い、表領域拡張、REDOログ拡張、接続数拡張 等の計画的な拡張運用、手順について策定する(※) ・表領域の拡張については、表領域のタイプ(SMALLFILE、BIGFILE)に応じて、データファイルの拡張、または、データファイルの追加で拡張する ・表領域拡張時の考慮点の例: - 1つのデータファイルのサイズが非常に大きい場合、バックアップからのリストア時や将来的なストレージ移行時への性能面・管理面で影響が生じる可能性が考えられる - データファイルの数が非常に多い場合、OS側で管理できるファイルディスクリプタの上限値に達する可能性や、場合によっては性能面で影響が生じる可能性が考えられる - データファイルを追加する場合、忘れずにバックアップ対象に追加する ・業務量が増え、REDO生成量が増加すると想定される場合は、REDOログを拡張する(REDOログファイルのサイズ拡張、REDOログファイルのグループ数増加) ・REDOログは、目安として、30分から1時間に1回程度のログスイッチが発生するよう設計する ・オンラインREDOログファイルに関する考慮点の例: - ファイルサイズが小さすぎると頻繁なログスイッチが発生する - ファイルサイズが大きすぎるとアーカイブに時間がかかる、オンラインREDOログファイルが破損した場合のデータ損失範囲が大きくなる

(※)リソースの設定(例:データファイルの自動拡張設定)や同時に考慮すべきパラメータ等について一貫性のある設計となるよう、事前に拡張運用設計、拡張手順を検討・確立しておく。

キャパシティ管理
計画的な「拡張運用」(2/3)

項番	中項目	小項目	重要度	推奨事項
	拡張運用	DBリソース拡張(2/2)	中	・業務量の増加に応じて、セッション数、プロセス数の上限設定(sessions、processesパラメータ)の拡張を検討／実施する ・セッション数、プロセス数増加時の考慮点: - 事前にOSリソース(CPU、メモリ)、DBリソース(SGA等)の増強の必要性を確認し対処する - sessions、processes パラメータからデフォルト値が導出されるパラメータについて影響範囲を確認する
38		H/W増強(1/2)	低	・定期的にキャパシティ傾向分析を行い、CPU、メモリ、ディスク 等の計画的な増設・増強運用、手順について策定する(※) ・CPU追加時の考慮点の例: - 追加後のCPUをデータベースが認識しているか確認する - インスタンスケージング機能を使用している場合、cpu_countパラメータの値を再調整する - CPU数の増加に伴い影響を受けるリソース(各種ラッチの数、共有プールの分割数、パラレル処理の多重度、CPU数から導出される各種パラメータ)を考慮し、基本的には可能な限りDBを停止した状態でCPUを増設する - 同時実行性の向上によるリソース枯渇(ITL、UNDO表領域の枯渇 等)に注意し、必要に応じて事前に拡張を行う - CPU数の増加に伴い共有プールの分割数が増えることで、個々の分割された共有プールのサイズが小さくなることがあるため、必要に応じて共有プールの増加を検討する - CPU数から導出されるパラメータについて影響範囲を事前に確認する

(※)リソースの設定(例:データファイルの自動拡張設定)や同時に考慮すべきパラメータ等について一貫性のある設計となるよう、事前に拡張運用設計、拡張手順を検討・確立しておく。

キャパシティ管理
計画的な「拡張運用」(3/3)

項番	中項目	小項目	重要度	推奨事項
	拡張運用	H/W増強(2/2)	低	・メモリ追加時の考慮点の例: - 必要に応じてメモリ関連パラメータ(sga_target、pga_aggregate_target等)を調整する - pga_aggregate_target を変更すると、実行計画に変動が生じうるため注意する - 必要に応じてOSカーネルパラメータも調整する 例:SGAを増加する場合は、必ず「kernel.shmall」も考慮する (参考: DocID: 1745635.1 (KROWN:133788)) - Linux環境におけるHugepagesを構成している環境でSGAを増加する場合は、サイズの再見積もりとシェル制限を再設定する必要があるため注意する ・ASMディスク追加時の考慮点の例: - 事前にASMディスク追加からリバランス処理完了までの時間を見積もる (例:検証環境での検証や、検証から机上で見積もる) - リバランスによる負荷、および、完了までの処理時間を考慮し、高負荷時間帯のリバランス実行は避ける
39		ノード追加	低	・例えば、サーバ構成上、H/W増強による対応が難しい場合(例:搭載可能な最大のCPUを既に搭載している)は、ノード追加によるスケールアップを検討する(※) ・ノード追加時の考慮点の例: - ノード数増加に伴う共有プールのGCS領域を考慮し、SGAの拡張が必要か事前に考慮する - アプリケーションパーティショニングの設計(例:接続ノード等)見直しが必要か確認する

(※)リソースの設定(例:データファイルの自動拡張設定)や同時に考慮すべきパラメータ等について一貫性のある設計となるよう、事前に拡張運用設計、拡張手順を検討・確立しておく。

定期・不定期メンテナンス
実行計画のインプットとなる「オプティマイザ統計情報」を適切に収集/管理(1/2)

項番	中項目	小項目	重要度	推奨事項
40	オプティマイザ統計	統計情報の収集ポリシー	高	・オプティマイザ統計情報運用に関するポリシー(収集するのか、それとも収集しないで固定化するのか、収集対象、収集タイミング、収集方法、保存方針等)を策定する ・収集ポリシー: - 全て収集する、一部はヒント句で実行計画を固定化するため統計は収集しない、等 ・収集対象: - すべてのユーザーオブジェクト(表・索引)について統計情報収集の要否を策定する - ユーザーオブジェクト(表・索引)、ディクショナリ、固定表、システム統計の全てを考慮する ・収集タイミング: - 対象にあわせて収集するタイミングを策定する (例: 日次で収集、バッチ更新によって列値の分布が大幅に変化した後に収集、 パーティションメンテナンス後に収集する 等) - 統計収集処理自体もある程度の負荷がかかり、統計収集による実行計画の変化の可能性等も考慮し、必要以上に頻繁に取得することは避け、可能な限りDBの負荷が低い時間帯に取得する ・収集方法: - 自動統計情報収集、または、手動統計情報収集(手動での取得) - DBMS_STATSのコマンドオプション(プリファレンス)を選定する ・保存方針: - 必要に応じて定期的に開発環境に統計情報を移行し保存する ・オプティマイザパラメータ: - 統計情報の収集ポリシーは、オプティマイザパラメータ(bind peek)等と合わせて策定する ・パーティションオブジェクトについては可能な限りグローバル統計/ローカル統計を両方収集する

定期・不定期メンテナンス
実行計画のインプットとなる「オプティマイザ統計情報」を適切に収集/管理(2/2)

項番	中項目	小項目	重要度	推奨事項
41	オプティマイザ統計	自動統計情報収集	高	・自動統計情報収集の機能を使用する場合、メンテナンス･ウィンドウ(実行時間帯)の調整を行う
42		手動統計情報収集	高	・手動で統計情報を収集する場合、対象毎に収集のタイミング、収集方法(パラメータの指定等)を策定する
43		収集された統計情報の確認	中	・統計情報の収集後、適切に統計情報が取得されていることを確認する
44		統計情報の退避	中	・必要に応じて定期的に開発環境に統計情報を移行し保存する(退避運用)

定期・不定期メンテナンス
性能障害/領域枯渇予防のための「領域対応」

項番	中項目	小項目	重要度	推奨事項
45	領域対応	不要データ削除	中	・定期的に、破棄可能なデータを削除する ・データの保持期間についてのポリシー、ライフサイクルを定義する (必要データの保持期間や、不要データの削除方法、CRUD表 等) ・パーティションを利用している場合は、パーティションの世代管理を自動化する
46		パーティションメンテナンス	中	・パーティションの世代管理(追加、削除等)を自動化する ・パーティションメンテナンス後にグローバル統計/ローカル統計を再収集する ・索引が無効化されるメンテナンスの場合、UPDATE INDEXESオプションの利用や、メンテナンス後の索引の REBUILD を考慮する
47		ログメンテナンス	中	・OS、Oracleで出力するすべてのログファイル、トレースファイルをローテンション管理する (例: 更新日付が1か月以上前のログを毎日退避) ・保持期間、削除サイクル、削除タイミング、削除方法を策定する ・ローテート後、一定の期間で削除する ・ローテーション、削除は自動化する ・対象ログの例: アラートログ、トレースファイル、リスナーログ、audit(監査)ファイル等
48		性能情報メンテナンス	低	・定期収集した性能情報の削除 (例: STATSPACK情報のパージ、ロギングした動的ビューの情報のパージ 等) ・保持期間、削除サイクル、削除タイミング、削除方法を策定する ・削除は自動化する ・キャパシティ管理のために、例えば、性能情報(AWRレポート)を別サーバに退避し、DBサーバ上からはパージする、等の運用を行う(例: 過去1年分は退避、DBサーバ上では3カ月分を保持)

定期・不定期メンテナンス
性能障害/領域枯渇予防のための「オブジェクトメンテナンス」

項番	中項目	小項目	重要度	推奨事項
49	オブジェクトメンテナンス	表の断片化解消	中	・定期的に表の断片化を解消する ・対象表の利用形態、性質に合わせて、作業手順を策定する (例: shrink、move、truncate->imp、オンライン再定義等) ・対象表、実施頻度、実施時期を策定する
50		索引の断片化解消	中	・定期的に索引の断片化を解消する ・対象索引の利用形態、性質に合わせて、作業手順を策定する (例: rebuild、rebuild online、drop/create、coalesce) ・対象索引、実施頻度、実施時期を策定する

定期・不定期メンテナンス
「DB変更手順の確立」によるオペレーションミス防止(1/2)

項番	中項目	小項目	重要度	推奨事項
51	DB変更手順の確立	ディスクの追加・変更	低	・ディスク(物理ディスク、ASMディスク等)の追加手順を策定しておく ・キャパシティープランニングを考慮した追加フロー、追加の判断基準を策定しておく ・別チーム連携(インフラ、H/W等)がある場合、連携と作業範囲、申請フロー等を明確にしておく ・容量不足に備えた余剰領域を確保しておく (例: 各ディスクグループ毎に余剰LUNを確保、ディスクに空きスロットを確保 等)
52		データファイル・表領域の追加・変更	低	・データファイル、表領域の追加手順を策定しておく ・キャパシティープランニングを考慮した追加フロー、追加の判断基準を策定しておく ・追加後にバックアップ対象からもれない手順、フローを策定しておく ・追加後に災対サイトからもれない手順、フローを策定しておく
53		オブジェクトの追加・変更	中	・表・索引の追加指針や作成／変更手順を策定する (例: PCTFREE等の指定、オブジェクト名の制約 等) ・オブジェクト追加や再作成直後のオプティマイザ統計取得の収集を考慮する ・オブジェクト追加や再作成後の断片化監視、解消対象への追加を考慮する ・追加・変更時の一連の作業フロー(検証、設計書への反映、実施 等)を策定しておく ・追加・変更により影響がでる依存オブジェクトのステータスを事前に検証環境で確認する ・追加・変更による影響範囲を事前に検証環境で確認する(例:索引追加によるDMLへの性能影響)

定期・不定期メンテナンス
「DB変更手順の確立」によるオペレーションミス防止(2/2)

項番	中項目	小項目	重要度	推奨事項
54	DB変更手順の確立	DBサーバの起動・停止	低	・DBサーバの再起動手順を策定しておく ・再起動に要する時間を事前に確認しておく
55		初期化パラメータ変更	低	・初期化パラメータの変更手順を策定しておく ・パラメータを変更する際は、必ず変更前のパラメータ(例:spfile)のバックアップを取得する ・パラメータを変更する際は、変更履歴管理を別途行う (spfileの場合、バイナリでありパラメータファイルに履歴コメントを記載できないため) ・動的にパラメータを変更する際は、必要に応じて、次回再起動時に変更前の値にリセットされないよう、メモリ上とSPFILE上両方で変更する(alter system 文にて scope=both で変更する)

構成管理
各種構成を適切に管理

項番	中項目	小項目	重要度	推奨事項
56	システム構成管理	システム構成一覧の管理	中	・システム構成を管理する 例: - H/W関連(サーバ、CPU、メモリ、ストレージ、ネットワーク、ハード機器 等) - OS関連(OS種類、バージョン、OSパッチ、インストールパッケージ、カーネルパラメータ、環境変数 等) - DB関連(バージョン情報、パッチ、初期化パラメータ、利用オプション 等) - AP関連(ミドルウェア情報、APバージョン情報 等) ・システム構成変更時には、構成管理情報を更新する ・各種作業手順書等の運用に関するドキュメントとともに管理する
57	DBオブジェクト管理	オブジェクトの妥当性確認	中	・構成変更やリリースのタイミングでオブジェクトの妥当性確認を行う 例: - INVALIDオブジェクトがないか確認する - 作成するオブジェクトの命名規則は適切か確認する - プライマリキーや一意キーが自動生成名になっていないか確認する - 意図せず、表や索引にNOLOGGINGやDEGREE句などの設定を行っているオブジェクトがないか確認する ・必要に応じて、定期的に使用していない索引がないか確認する
58	SQL実行計画管理	実行計画の管理	中	・SQL実行計画を管理する ・重要なSQLの実行計画のリリース管理を行う(例:SPM機能を利用)

予防保守
重要技術情報の収集／計画的な対処による既知障害の予防

項番	中項目	小項目	重要度	推奨事項
59	技術情報収集	重要技術情報(障害情報、既知不具合情報)の収集・確認	中	・定期的にMy Oracle Support(MOS)の重要技術情報を精査する 例: - 毎月、重要技術情報を確認する (オラクル・サポート・ニュースレター、Database 技術情報トップページ (Doc ID 1632115.1) のホットトピック) - 四半期に一回リリースされる推奨パッチ(PSU)の修正リストを確認する - 月次～少なくとも四半期毎に、対象期間中に更新があった技術ドキュメントを確認する ・MOSの検索条件をあらかじめ定義する(例:対象製品、バージョン指定等) ・実施頻度、実施方法を策定する - 少なくとも四半期に一度以上実施する(例:月次～少なくとも四半期毎) - パッチ適用が可能な計画停止時期を考慮し、最適な実施頻度を検討する ・精査した情報に対して、対処を行う基準をあらかじめ明確にしておく。 (例:DB全体の性能劣化、DBの異常終了やハング、結果不正など、業務影響が大きいもの、 DBとして正しい動作ができない状態となる問題については必ず対処を行う等、 対処方法の検討をスムーズに行えるよう、影響度のレベルに応じた対応を定義しておく)

予防保守
計画的なパッチ適用による既知不具合事象の予防

項番	中項目	小項目	重要度	推奨事項
60	パッチ適用	定期的なパッチ適用	中	・定期的に、H/W、OSのパッチを適用する - DBや他のアプリケーション、ミドルウェアにおいて動作保障されるか、サポートされるかを確認の上、適用する ・定期的に、Oracleのパッチを適用する - DB、クラスタウェアのパッチに関して、基本的に下記を考慮する ・最新のPSRを適用する ・定期的にPSU、CPUを適用する ・個別パッチについては、他の回避策がない場合に適用する ・必ず検証環境でのテストを実施の上適用する ・パッチ適用を考慮し、1年に1回程度、計画停止期間を設けることを検討する (PSRは、一般的には1年程度の周期でリリースされるため) ・基本的に、既知不具合を予防するために、計画的な適用を検討する ・例: PSUがリリースされた際に、利用システムで適用必須の不具合修正が含まれていないか確認し、PSUの適用を検討／実施

予防保守
定期的な計画停止／災対サイトの定期切替訓練による障害予防

項番	中項目	小項目	重要度	推奨事項
61	計画停止	定期的な計画停止運用	高	・定期的な計画停止期間を策定する ・可能であれば、少なくとも1年に1回程度は計画停止することを検討する。 (PSRは、一般的には1年程度の周期でリリースされるため) ・計画停止期間中に、システムの各種保守作業、PSR/PSU/セキュリティパッチの適用、停止を伴う静的パラメータ設定によるチューニング、連続稼働で顕在化する問題への対処 等を実施する
62	災対サイト定期切替	定期的な切替訓練	低	・災対サイトがあるシステムにおいて、定期的にサイト切替訓練を実施する (訓練により障害発生時の迅速な切替につながり、2次障害の予防につながる) ・訓練を行い、以下等を確認する - 正常に切替、および、災対サイトの利用が可能か確認する - 切り替え時のシステム間連絡フロー・運用フローを確認する ・必要に応じて、災対サイトへの切替訓練時に、併せて本番環境のメンテナンスを実施する

障害対応
対応フロー確立による迅速な対応、長引かせない対応、2次障害予防

項番	中項目	小項目	重要度	推奨事項
63	障害対応	障害切り分けフロー、取得情報一覧	高	・障害の一次切り分けフローを障害ケース毎に策定しておく ・取得するログ情報の一覧をあらかじめ定義しておく ・初動調査用の情報取得手順、情報一括取得ツールを用意しておく(例:基本ログ一括取得)
64		障害対応手順	高	・想定される障害毎の対応手順、対応マニュアルを用意しておき、監視からエラーを検出した際に迅速に対応できるようにする ・セッション切断手順書を作成しておく(特定セッションを切断し再実行する場合に備えた用意) ・障害ケースに応じた情報取得手順、情報一括取得ツールを用意しておく(例:ERRORSTACK取得) ・切り分け後の復旧手順を確立しておく ・対処の所要時間も考慮したマニュアルを用意する (例:バックアップからのリストア、リカバリにかかる時間)
65		縮退時の流量制限(RAC環境)	高	・縮退運用を見越した設定値の確認、H/Wリソース確保(例:2ノード環境で、片系運用のケース) ・縮退運用を行う仕組み、手順を用意しておく (例:ロードバランサ、APコネクションプール数による制限等)
66		切替による復旧手順	高	・障害ケースに対して、待機系サイト、災対サイトへのフェイルオーバー・フェイルバックの復旧手順を策定しておく ・復旧までの所要時間を見積もっておく(RTOに対する充足性確認や影響範囲確認に利用) ・切替にある程度時間を要する場合は、DBサーバへの監視を一時的に抑止することを検討する

障害対応
インシデント管理／横展開／定期レビューによる障害予防

項番	中項目	小項目	重要度	推奨事項
67	障害時連絡体制	障害時役割分担	高	・障害発生時の連絡フロー、および、初動体制(例:専任DBA)を整えておく ・可能な限り速やかに一次切り分けや分類などが行える体制を組織全体で整えておく
68	インシデント管理	インシデント管理	中	・発生事象、原因、対処方法、防止策等の履歴を管理する ・それらがシステム関係者、および、関連チーム間で共有化／横展開され、横並びチェックが行える仕組みを策定する
69	監視項目・閾値分析	監視項目・閾値分析評価・障害対応手順修正	中	・定期的な運用見直し(監視項目、閾値項目、障害切り分け手順の追加や削除)を実施する ・運用レビュー会を定期的に開催する ・運用状況のチェックを運用オペレーター、運用設計者間にて定期的に実施する ・運用を定期的に変更する仕組み、承認フローを整えておく

バックアップ・リカバリ
障害ケースを想定し抜け漏れなくバックアップを取得・管理する(1/2)

項番	中項目	小項目	重要度	推奨事項
70	バックアップ設計	バックアップ方針	高	・RPO/RTOを基にバックアップ方針を策定する(バックアップ対象、バックアップ方法、取得タイミング、保存期間・世代、保存先) ・障害ケースごとのリカバリ設計を策定する(完全・不完全リカバリ、表領域単位のPoint-in-Timeリカバリ、論理バックアップからのインポート等) ・要件時間内にリカバリが完了するよう(SLAを満たすよう)方針を策定する ・バックアップは最低限2世代保持する(バックアップ中に障害が起きた場合に備えた対応) ・バックアップは本番環境とは物理的に異なる場所に保存する
71		システムバックアップ	高	・対物理障害を満たすためにバイナリバックアップを取得する(OS、ソフトウェア、レジストリ等) ・取得方法、取得手順(例:ストレージのスナップショット機能等)をあらかじめ用意しておく ・適切なタイミング(例:初期構築時、OS構成変更前後、パッチ適用前後)で取得する
72		クラスタ構成ファイルのバックアップ	高	・クラスタ構成ファイルのバックアップを取得する ・バックアップ対象を適切に策定する(例:RAC環境にて、OCR、OLR、投票ディスク) ・取得方法、取得手順(例:ocrconfigコマンド)をあらかじめ用意しておく ・適切なタイミング(例:初期構築時、CRSリソース変更前後、ノード追加前後)で取得する
73		DB/ASM構成ファイルのバックアップ	高	・DB/ASM構成ファイルのバックアップを取得する ・取得方法、取得手順(例:OSコマンド、ASMCMDコマンド)をあらかじめ用意しておく ・バックアップ対象を適切に策定する(例:DB/ASMのSPFILE、DB/GIのOracle Net構成ファイル) ・適切なタイミング(例:初期構築時、設定変更前後)でバックアップを取得する

バックアップ・リカバリ
障害ケースを想定し抜け漏れなくバックアップを取得・管理する(2/2)

項番	中項目	小項目	重要度	推奨事項
74	バックアップ設計	DBの物理バックアップ	高	・要件に則した物理バックアップを取得する (例:コールドバックアップ、ホットバックアップ、差分バックアップ) ・目標復旧時点までリカバリが可能であるようバックアップを取得する ・取得方法、取得手順(例:RMANコマンド、OSコマンド、ストレージコピー機能)を用意しておく ・バックアップ対象(例:データファイル、REDOログファイル、制御ファイル、アーカイブログ)を適切に策定する ・適切なタイミングで取得する ・必要な世代管理を行う ・適切な保存先を用意する(例:LTO、バックアップサイト)
75		DBの論理バックアップ	低	・要件に則した論理バックアップを取得する(例:DataPump Export) ・取得方法、取得手順(例:DataPump Export、従来型Export)を用意しておく ・バックアップ対象範囲(例:必要テーブル、複数テーブル間のリレーションを考慮した範囲)を適切に策定する ・適切なタイミング(例:対象テーブルに更新がないタイミング)で取得する ・必要な世代管理を行う ・適切な保存先を用意する(例:LTO、バックアップサイト) ・セキュリティを考慮した保存先を用意する(論理バックアップにデータを含むケースがあるため)
76		リモートバックアップ	高	・対サイト障害に備え、バックアップをリモートサイトに退避する (災対サイトがある場合には重要度低)

バックアップ・リカバリ
迅速な復旧のための障害ケース毎のリカバリ設計

項番	中項目	小項目	重要度	推奨事項
77	リカバリ設計	リカバリ方針	高	・RPO/RTOを基にリカバリ方針を策定する ・障害ケースごとのリカバリ設計を策定する(完全・不完全リカバリ、表領域単位のPoint-in-Timeリカバリ、論理バックアップからのインポート等)
78		リカバリ手順	高	・障害ケースごとのリカバリ手順を策定する ・リカバリ所要時間を把握しておく
79		リカバリ試験・トレーニング	中	・障害訓練を行う環境を用意する ・定期的なリカバリや切り戻しの訓練を行う (リストア・リカバリの手順は複雑なため、実際に実施する際に間違って2次障害を引き起こさないよう、定期的なリストア・リカバリ訓練の実施を検討する)

セキュリティ管理
情報保護のためのアカウント管理運用

項番	中項目	小項目	重要度	推奨事項
80	セキュリティ要件	セキュリティ要件	高	・セキュリティ要件を策定しておく
81	アカウント管理	アカウント管理方針	中	・不要なアカウントをロックする方針を策定しておく ・パスワードの定期的なメンテナンス方針を策定しておく ・DB管理者が交代する場合、DB管理者ユーザーのパスワード変更/アカウントロック/削除する等の運用を行う
82		アカウント追加削除手順	中	・必要最低限のユーザーのみ追加する ・必要最低限の権限のみを付与する ・不要なアカウントが存在する事を確認する仕組みを策定しておく ・不要なアカウントを削除する仕組みを策定しておく ・アカウント追加／削除手順を用意しておく
83		パスワード変更手順	中	・OSユーザやDBユーザのパスワード変更方針を策定しておく(例:定期的に行う) ・変更する場合の変更手順を確立しておく ・変更する場合の変更間隔／頻度／時期、パスワード管理ルール(パスワードの長さ、複雑度等)を決めておく ・変更時の作業ログの管理方法を確立しておく(例:CUIで変更した場合、ログにパスワードが出力されるためログを管理する必要がある) ・併せてアプリケーション側、スクリプト側のパスワードを変更する(アプリケーション側、スクリプト側にパスワードを保持している場合)

セキュリティ管理
不正防止のためのセキュリティ監視

項番	中項目	小項目	重要度	推奨事項
84	セキュリティ監視	アクセス監視	高	・監査ログを取得する対象操作を決める(例:ログイン/ログアウト、特権ユーザーによる操作) ・監査ログを定期的に削除、保管する手順を策定する(DBサーバ上での保存場所、保存期間、削除手順、DBサーバ外での保管場所、保存期間、削除手順) ・監査ログへのアクセス権の管理を行う(監査ログが不正に削除されないようにする必要がある) ・監査方法(例:統合監査、標準監査、DBA監査、ファイングレイン監査など)を策定する ・必要に応じて、長期に渡る監査証跡の保存や、統合監査システムへのデータ連携を行う
85		アクセス監視精査、保管	高	・監査ログを定期的に精査する仕組み・または手順を策定する ・監査ログを保管する手順を策定する ・監査管理者をDB管理者等とは別に用意し監査ログを管理する(改ざん防止)

5. まとめ

まとめ

「データベース運用における一般的な推奨項目」を85項目に整理しました。

まずはぜひ「データベース運用の点検」を実施してみてください。

データベースの稼働状況の可視化(ヘルスチェック)、
予防保守運用(パッチ適用、チーム作りによる障害予防)
のさらなる詳細はこの後のセッションにてご紹介します！

Appendix: データベース運用の標準化

データベース運用の標準化(1/2)

標準化に向けた整理の一例

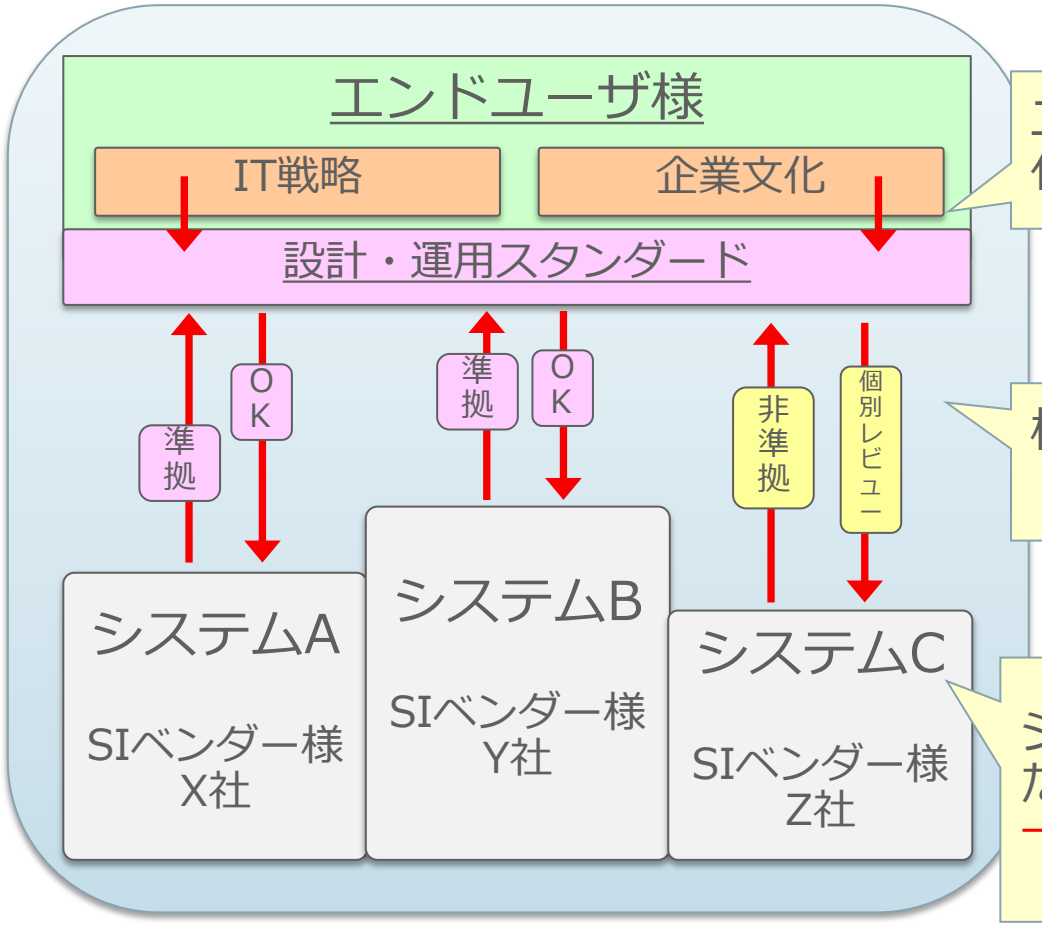
運用設計全体シート			SIベンダーX社担当 SIベンダーY社担当			システムA	システムB	システムC
						システムA	システムB	システムC
						Linux x86-6	Windows x64	AIX 5L
						11.2.0.4	12.1.0.2	11.2.0.3
						2ノードRAC	シングル	HAクラスタ
項番	大分類	中分類	小分類	推奨事項	重要度			
1	サービス・レベル管理	キャパシティ要件	サービスレスポンスタイム目標	・サービスレスポンスタイムの目標値を定める (パフォーマンスチューニングの指標として位置づけられる)	高	・・・・・	・・・・・	・・・・・
2			サービススループット目標	・サービススループットの目標値を定める (キャパシティプランニングのベースラインとして位置づけられる)	高			
3		可用性要件	目標稼働時間	・目標稼働時間を定める (一定期間においてシステムがどれくらい正常稼働しているかを示す数値)	高			
4		IT継続性要件	目標復旧時間（RTO）	・目標復旧時間(RTO:Recovery Time Objective、ここでは最大許容停止時間と同義)を定義する ・DB障害、サイト障害それぞれにおいて定義する ・業務継続の必要性の有無、手動や代案での業務継続も含め、バックアップやリカバリ、災対サイトの構築など、要件に伴い設計が大きく変わる可能性があるため、明確かつ具体的な障害シミュレーションを行う	高			
5			目標復旧時点（RPO）	・目標復旧時点(RPO:Recovery Point Objective)を定義する ・DB障害、サイト障害それぞれにおいて定義する ・物理障害、論理障害などによるデータの損失の範囲を定義する ・業務継続の必要性の有無、手動や代案での業務継続も含め、バックアップやリカバリ、災対サイトの構築など、要件に伴い設計が大きく変わる可能性があるため、明確かつ具体的な障害シミュレーションを行う	高			
6	情報収集	AP性能情報収集	APログ取得	・AP側で処理した業務量や応答時間のログを記録して取得しておく (AP性能情報と、OS/DB性能情報、各種リソース利用状況を関連づけて現状把握を行う際に利用) ・取得情報の例: - 単位時間あたりのオンライントランザクション処理件数 - オンライントランザクションの平均応答時間 - バッチ処理で処理したデータ件数と処理時間 ・過去と比較できるようある程度の期間分(例:1ヶ月～3ヶ月分)を保存する	中			

各システム毎の
運用状況を
一覧化、整理する

運用項目毎に横並びで一覧化する
⇒システムによってできていない
事項があれば改善点を検討する

データベース運用の標準化(2/2)

標準化による効果イメージ



エンドユーザ様のIT戦略・企業文化に沿った標準の策定

標準に準拠する事での品質向上
（非準拠の場合は個別レビュー）

システム毎のノウハウにばらつきがあったとしても、運用標準に準拠することで一定以上の同じ品質の運用が可能となる

Integrated Cloud

Applications & Platform Services


ORACLE®