DBパフォーマンスチューニングの基礎

HN　おいろん

1. はじめに

1. はじめに

- 自己紹介
  - 名前: 守田 典男 （HN:おいろん） 29歳
  - 職業: 某会社 技術社員
  - 業界歴: 開発(汎用):2年→DB:4年→
    開発・DB:2年
  - DB歴: Oracle 6年
    SQLServer2000、2005 1年半
    HiRDB 半年

1. はじめに

- 本セッションについて
  - 対象：開発初心者（Lv1クマー）

・「パフォーマンス」を考えるとはどういうこと？
・パフォーマンスチューニングはDBAの仕事
　であって、開発者には関係ないのでは？
・もっと処理が早いSQLを書きたい！

上記のような、DBパフォーマンスに興味が
ある方を対象としております。

目次

1. はじめに
2. DBパフォーマンスチューニングの考え方
3. SQLチューニング
   ～SQLのよりよい書き方～
4. おまけ：Oracle Vs SQLServer
5. おわりに

2. DBパフォーマンスチューニングの考え方

2. DBパフォーマンスチューニングの考え方

2-1. パフォーマンスチューニングの概要

2-2. パフォーマンスチューニングのサイクル

2-1.パフォーマンスチューニングの概要

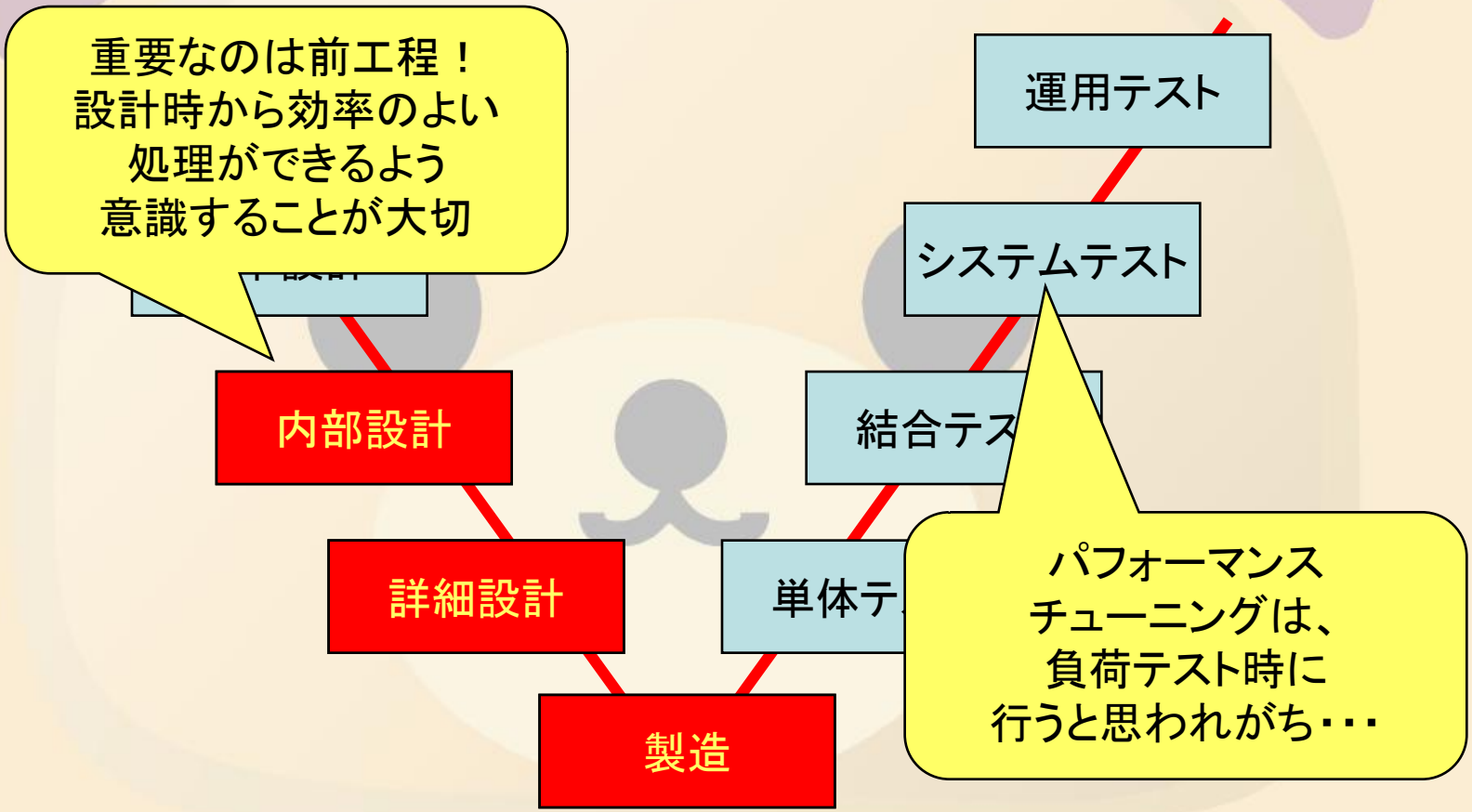
- 目的
  - **限られたシステム・リソースの中で、最大限のパフォーマンス効果を出すこと**

近年の目覚しいハードウェア技術の向上で、
パフォーマンスを意識することがなくなってきている。

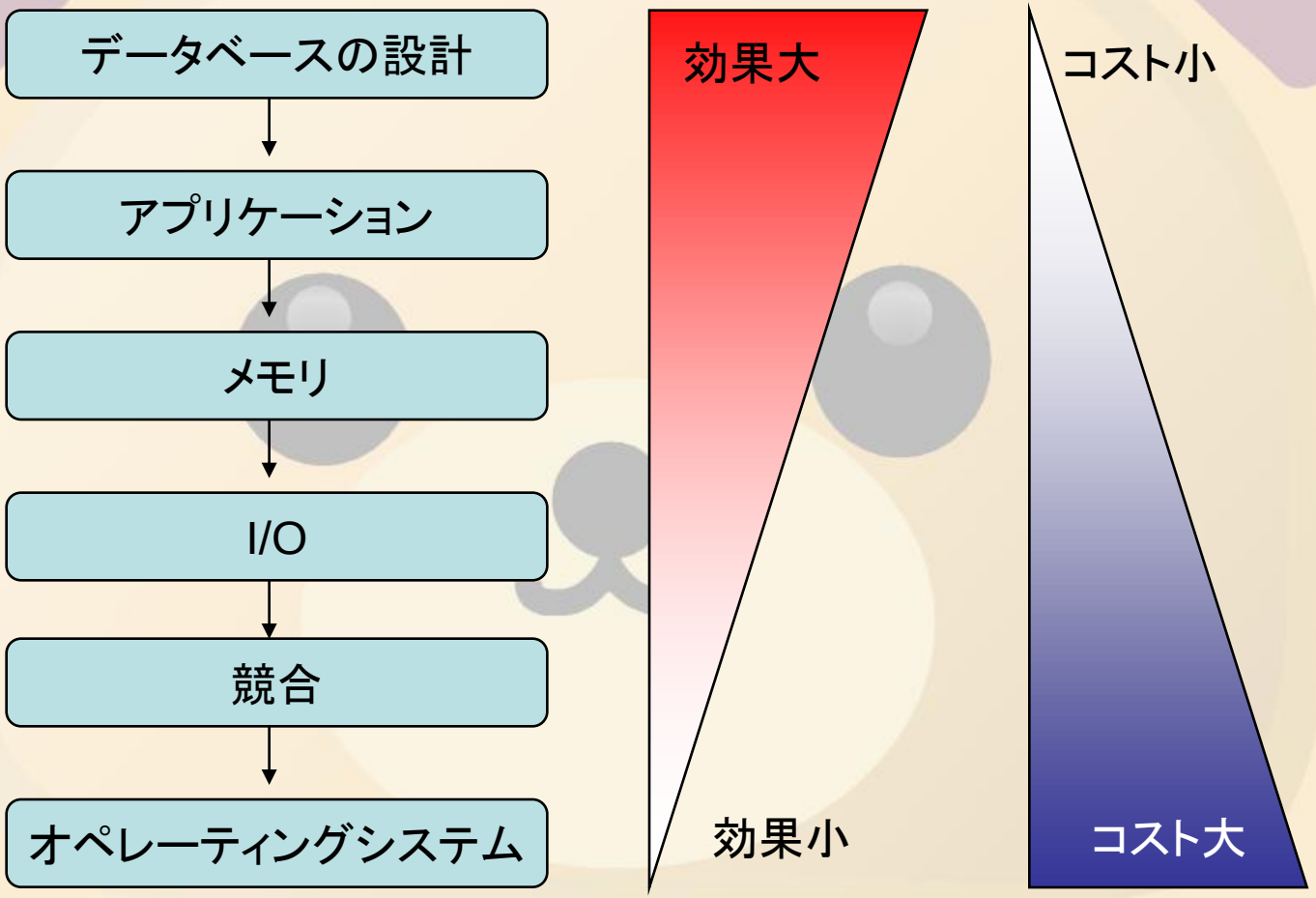
せっかくハードウェアを増強しても、効率よく処理されていなければ、まったくの無駄となる！

2-1.パフォーマンスチューニングの概要

- 工程



- チューニングの優先順位



2-1.パフォーマンスチューニングの概要

- 設計
  - テーブル設計
    - 正規化、非正規化など
  - データファイルの物理設計
    - データファイル、ログファイルの分散
- 開発
  - 開発標準、SQLコーディング規約の遵守
  - SQLチューニング

2-1. パフォーマンスチューニングの概要

2-2. パフォーマンスチューニングのサイクル

2-2.パフォーマンスチューニングのサイクル

- パフォーマンスチューニングの手順

2-2.パフォーマンスチューニングのサイクル

- 目標設定
  - パフォーマンスチューニング完了となる最終目標を決定する
    - 処理完了の目標時間の設定
    - アプリケーションの反応時間の目標時間の設定
- 現状分析
  - 問題となっている箇所の状況を測定・分析する
    - SQLの詳細情報の取得
    - OSリソースの情報取得

2-2.パフォーマンスチューニングのサイクル

- チューニング方針の決定・実行
  - 現状分析の結果より、ボトルネックをみつけ、解消するための措置を講じる
    - SQLの修正
    - インデックスの作成
- チューニング結果の確認
  - 実施したチューニングの効果を確認する。
  - 設定した目標に達していなければ、再度現状分析を行い、最適なチューニングを模索する

３．SQLチューニング
～SQLのよりよい書き方～

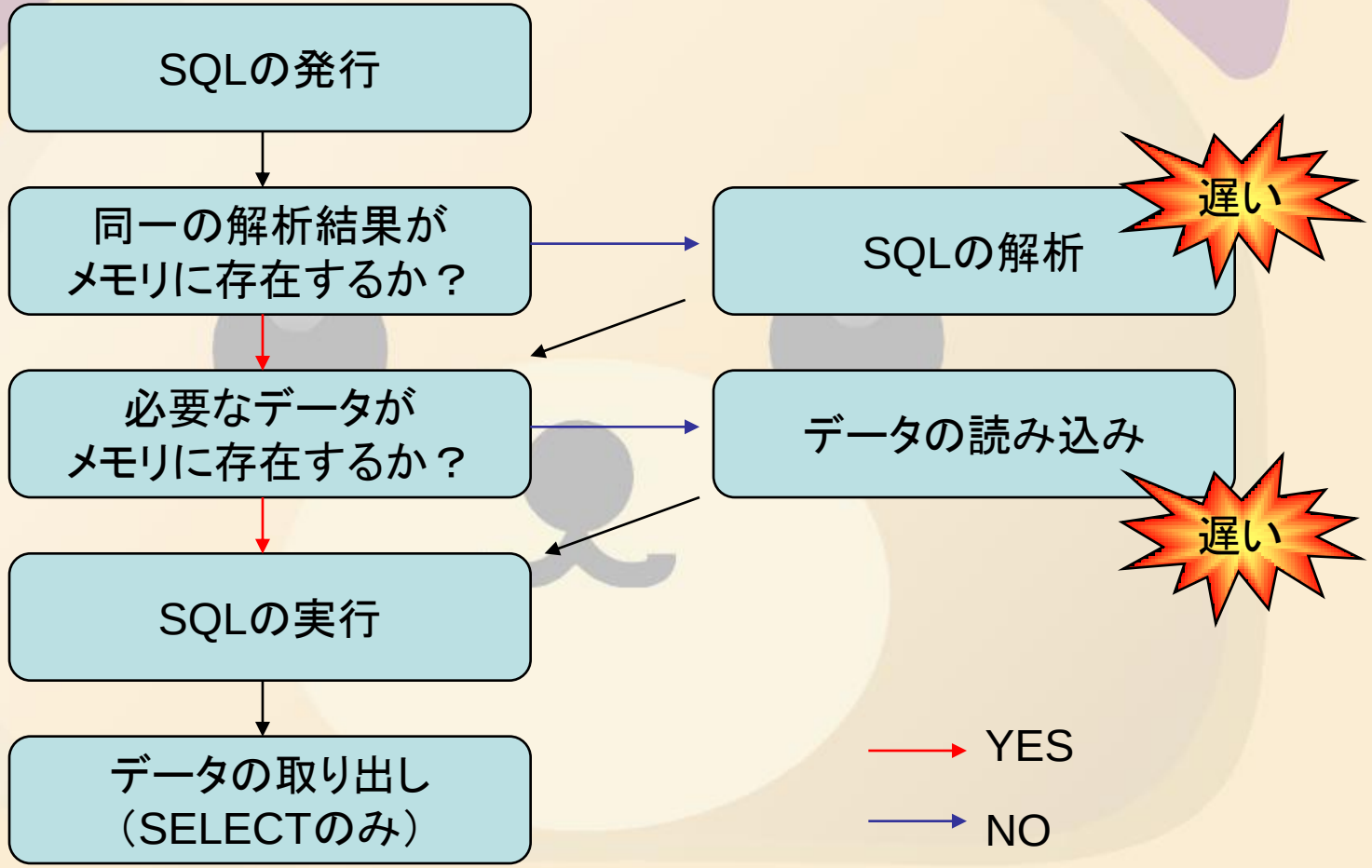
3-1. SQL実行のメカニズム

3-2. やってはいけないSQLの書き方
「こう書くと処理が遅くなるよ！」

3-3. ちょっとした一工夫
「こう書くとちょびっと速くなる！」

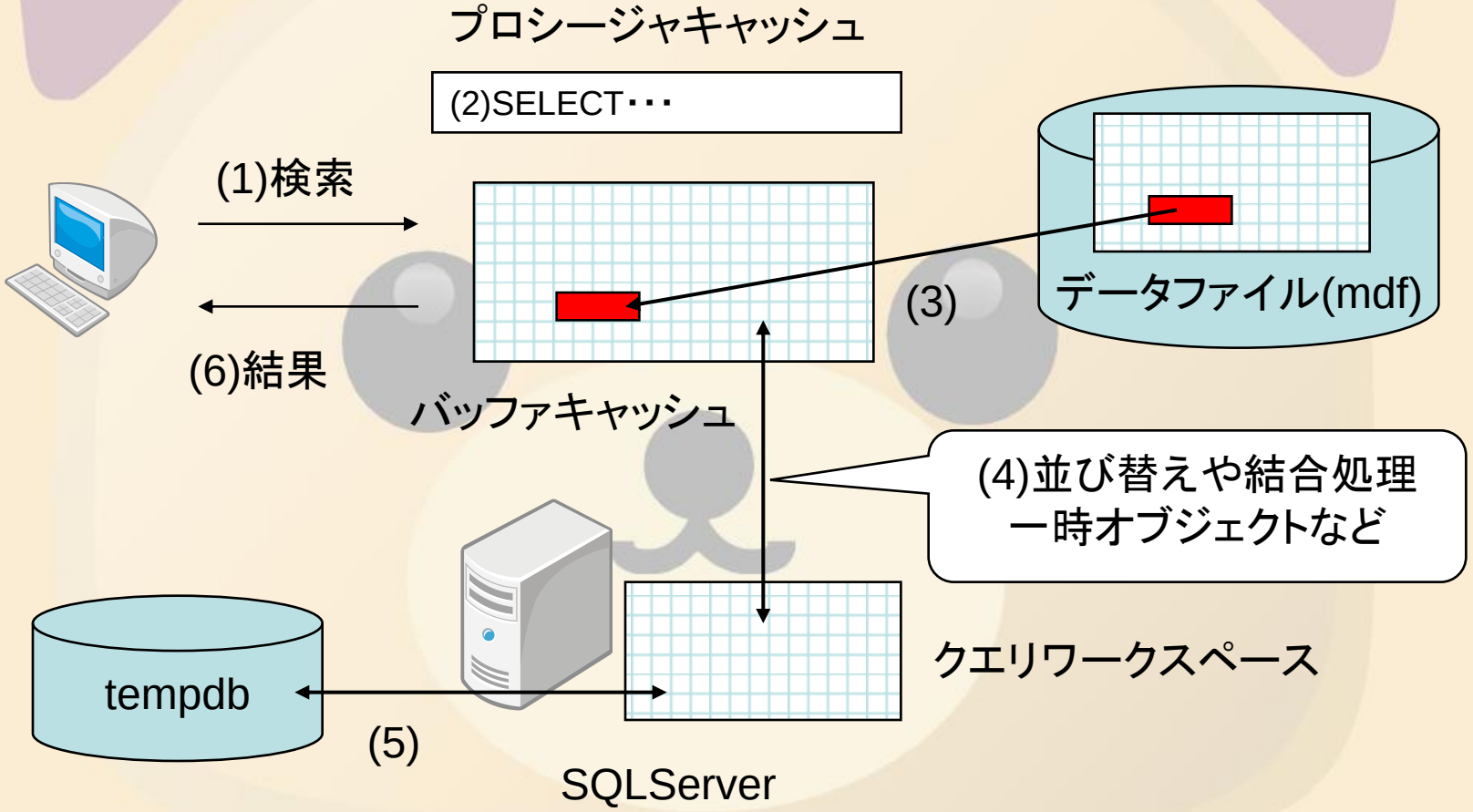
3-1.SQL実行のメカニズム

- SQLの処理フロー



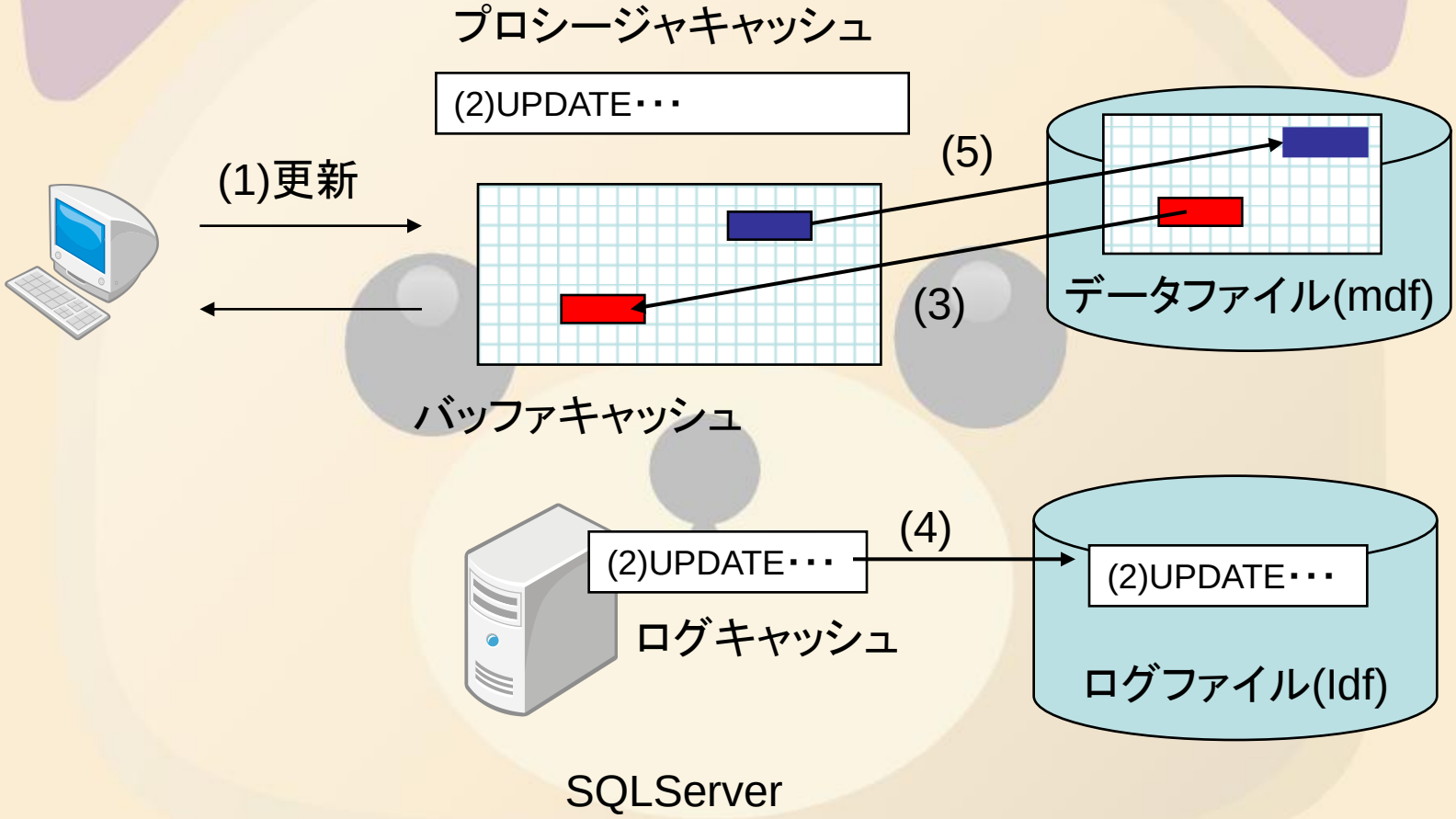
3-1.SQL実行のメカニズム

- SQLServerの場合（検索）



3-1.SQL実行のメカニズム

- SQLServerの場合（更新）



3-1.SQL実行のメカニズム

- SQLチューニングのポイント
  - SQL解析時間を少なくする！
    - SQLの共有化（コーディング規約の遵守）
      ※大文字、小文字が違っても共有されない！
    - バインド変数の使用、実行計画の共有化
  - ディスクからのデータ読み込みを少なくする！
    - 全表操作をできるだけなくす
    - 索引（インデックス）の利用

3. SQLチューニング

3-1. SQL実行のメカニズム

3-2. やってはいけないSQLの書き方
「こう書くと処理が遅くなるよ！」

3-3. ちょっとした一工夫
「こう書くとちょびっと速くなる！」

3-2.やってはいけないSQLの書き方

- ワイルドカード(“＊”)
  - 全列を指定するときワイルドカードを使うと読み替えが発生する。
    →オーバーヘッドの原因となる。

SELECT ＊ FROM 表1 WHERE 条件;

↓

SELECT 列1、列2、・・・ FROM 表1
WHERE 条件;

3-2.やってはいけないSQLの書き方

- 不要な列の指定
  - 不要な列を指定すると読み取るデータ量が増える
    →オーバーヘッドの原因となる。

```
SELECT　＊　FROM　表1　WHERE　条件;
↓
SELECT　列1、列2　FROM　表1
WHERE　条件;
```

3-2.やってはいけないSQLの書き方

- インデックスが使用されない書き方
  - 以下のような記載はインデックスが使用されない
    - Null値の検索
      - 特定の値に置き換えるなどの処置が必要
    - 暗黙の型変換
    - LIKE句の中間一致、後方一致
    - 否定形の使用（<>、!=、NOT　EQUALS）
    - 複合索引時に、列の順番を間違える
    - 索引列に関数を使用。（WHERE　列1　＊　1.1　＝　100）

たとえインデックスがあっても使われないので注意！！

3. SQLチューニング

3-1. SQL実行のメカニズム

3-2. やってはいけないSQLの書き方
「こう書くと処理が遅くなるよ！」

3-3. ちょっとした一工夫
「こう書くとちょびっと速くなる！」

3-3.ちょっとした一工夫

- 全表操作(フルスキャン)が有効なケース
  - 目安:抽出データが全データの10%から15%未満の場合(ソートが必要ない場合)
  - 理由:1回のスキャンで取得できるデータ量がフルスキャンの方が多いため。

3-3.ちょっとした一工夫

- ソートの回避
  - 次の句を使用するとソートが発生する
    - GROUP BY句
    - ORDER BY句
    - 集約関数（SUM、COUNT、AVG、MAX、MIN）
    - DISTINCT
    - 集合演算子（UNION、INTERSECT、EXCEPT）
      - 重複排除（ALL　オプション）でソートを回避できる
    - OLAP関数（RANK、ROW_NUMBERなど）

回避できる場合は極力回避すること。

4. おまけ:Oracle Vs SQLServer

4. おまけ：Oracle Vs SQLServer

4-1. それぞれの長所・短所

4-2. 結論

（注）：ここの内容は個人的な考えが
多く・・・いや、ほとんど含まれております。

4-1.それぞれの長所・短所

- Oracleの長所
  - マルチプラットフォーム
    - DBシェアが高い要因の1つ
      とりあえずOracleできればなんとかやっていける。
  - 高機能・高性能
    - こまかいところまで設定でき、データベースの
      動きなど、中身が把握しやすい。
  - 学習しやすい
    - 書籍やWebなど自習環境が整いやすい

4-1.それぞれの長所・短所

- Oracleの短所
  - 高い！！
    - Enterprise Edition　CPUライセンスで500万円。
    - 高機能を謳い文句にしながら、「別途オプション」ってどういうこと！？
      （Real Application Testing　125万円）
    - Oracle Master Platinum　は研修2つに試験料で50万円以上がかかる。　個人じゃ無理。
    - サポート契約しないと修正パッチもらえない。
    - バグ多いよ。

4-1.それぞれの長所・短所

- SQLServerの長所
  - 圧倒的な使いやすさ
    - Oracle大好き人間だった私の考え方を変えた。
  - 機能が豊富
    - Enterprise Editionですべての機能が使える。
    - パッチの適用もスムーズ。保守もしやすい。
  - 学習しやすい
    - Microsoftホームページに自習書やWebCastなど

4-1.それぞれの長所・短所

- SQLServerの短所
  - 変わったSQLがなぜか実行できる
    - UPDATEにFROM句が使えるなんて思ってもみませんでした・・・。常識？
    - テーブルに存在しない列をORDER BYに指定できる・・・。（SQLServer2000のみ）
  - エラーがよくわかんない
    - これはただ慣れていないだけかも・・・。
    - サポートオンラインに書いている内容が難しすぎて理解できない・・・。

4. おまけ：Oracle Vs SQLServer

4-1. それぞれの長所・短所

4-2. 結論

（注）：ここの内容は個人的な考えが
多く・・・いや、ほとんど含まれております。

4-2.結論

- 高機能、高性能はOracle、
  使いやすさはSQLServer
  （よく、Oracleはマニュアル車、
  　SQLServerはオートマ車と例えられる・・・）

- しかし
  昨今Windowsの高性能・安定化もあって
  個人的にはSQLServerに期待大。

5. おわりに

5. おわりに

- いかがでしたか？
  - 少しでも、DBを意識したり、興味を感じることができたでしょうか？
  - パフォーマンスの問題の8割はSQLなど開発者に起因するものといわれています。
  - 日頃から意識して、顧客に喜ばれるシステムを開発しましょう！！

ご清聴ありがとうございました！！