データベース設計の基礎

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：設計・開発初心者（Lv1クマー）

・データベース設計はしたことない。
・管理はできても設計は・・・。
・正規化やテーブル設計について知りたい！

上記のような、データベース設計未経験の
設計・開発初心者を対象としております。

目次

1. はじめに
2. データベース設計の概要
3. 概要設計
4. 論理設計
5. 物理設計
6. おわりに

2. データベース設計の概要

2. データベース設計の概要

- 目的
  - **信頼性の高いデータベースの構築**
    - 整合性を維持する
    - 障害から迅速に、確実に回復する
  - **拡張性の高いデータベースの構築**
    - アプリケーションの追加・拡張が容易にできる
    - データが一元管理されている
  - **ユーザ要求の性能を満たすデータベースの構築**
    - 同時実行性が確保できる
    - 効率のよいデータアクセスが可能

2. データベース設計の概要

- 手順
  - **概要設計**
    - 業務で扱う情報を整理し、概念ER図を作成する
    - 必要な情報を予測して拡張性をもたせる
    - 正規化を行い、あるべき姿で情報を整理する
  - **論理設計**
    - SQLを分析して最適化処理を行う
    - 業務とユーザの対応づけ
    - セキュリティ設計を行う
    - 論理ER図の作成

2. データベース設計の概要

- 手順
  - **物理設計**
    - ハードウェアの資源を使いきり、ユーザ要件を満たすべき性能と障害回復を中心として可用性の高いデータベースを実現する
    - DBMSで利用する機能を選択、パラメータ値の設定、オブジェクトの作成、メモリ要件の定義、ディスク配置などの定義、方式設計の物理的な詳細を設定する

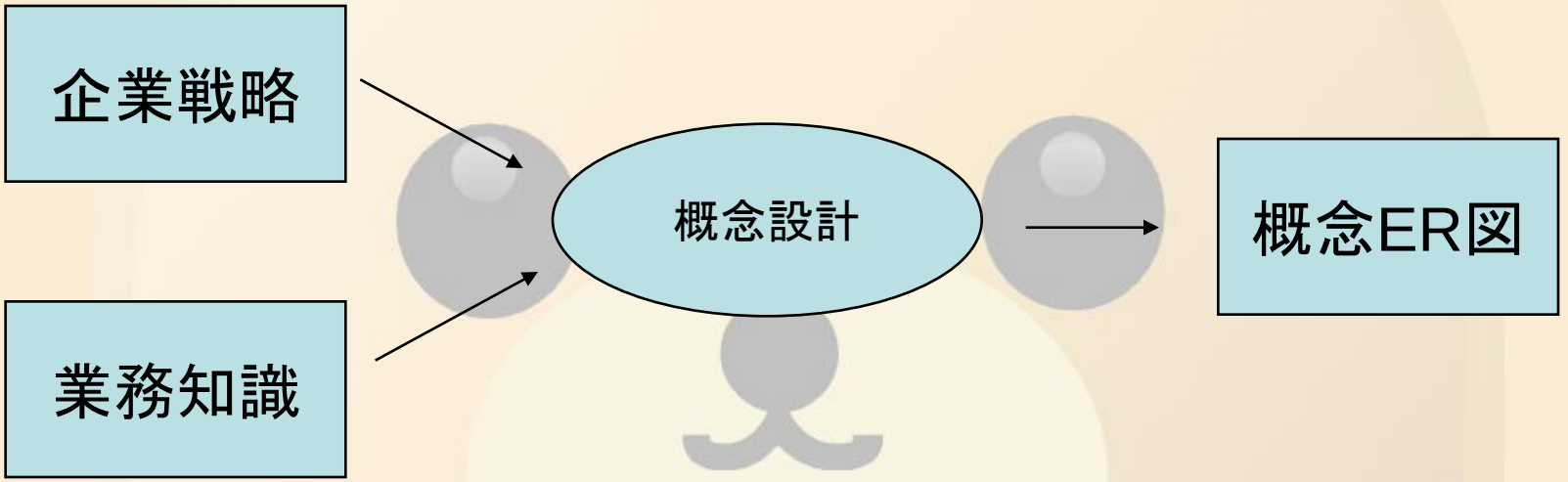
3. 概要設計

3. 概要設計

- 目的
  - 企業の方向性を認識し、拡張性の高いデータベース設計を行う。
  - 既存業務で使用している画面や帳票を使って漏れのない設計を行う。
  - 正規形を満たした理想的な形を作成する。
  - システムの方式に依存しないモデルを作成する。

3. 概要設計

- インプットとアウトプット



3. 概要設計

- 手順

  ①対象領域を明確にする
  　既存の画面・帳票から業務ルールを把握する。

  ②エンティティを洗い出す

  ③リレーションシップを検討する

  ④概念データモデルを検証する

1. はじめに

- サンプルシステム

Internet Explorer

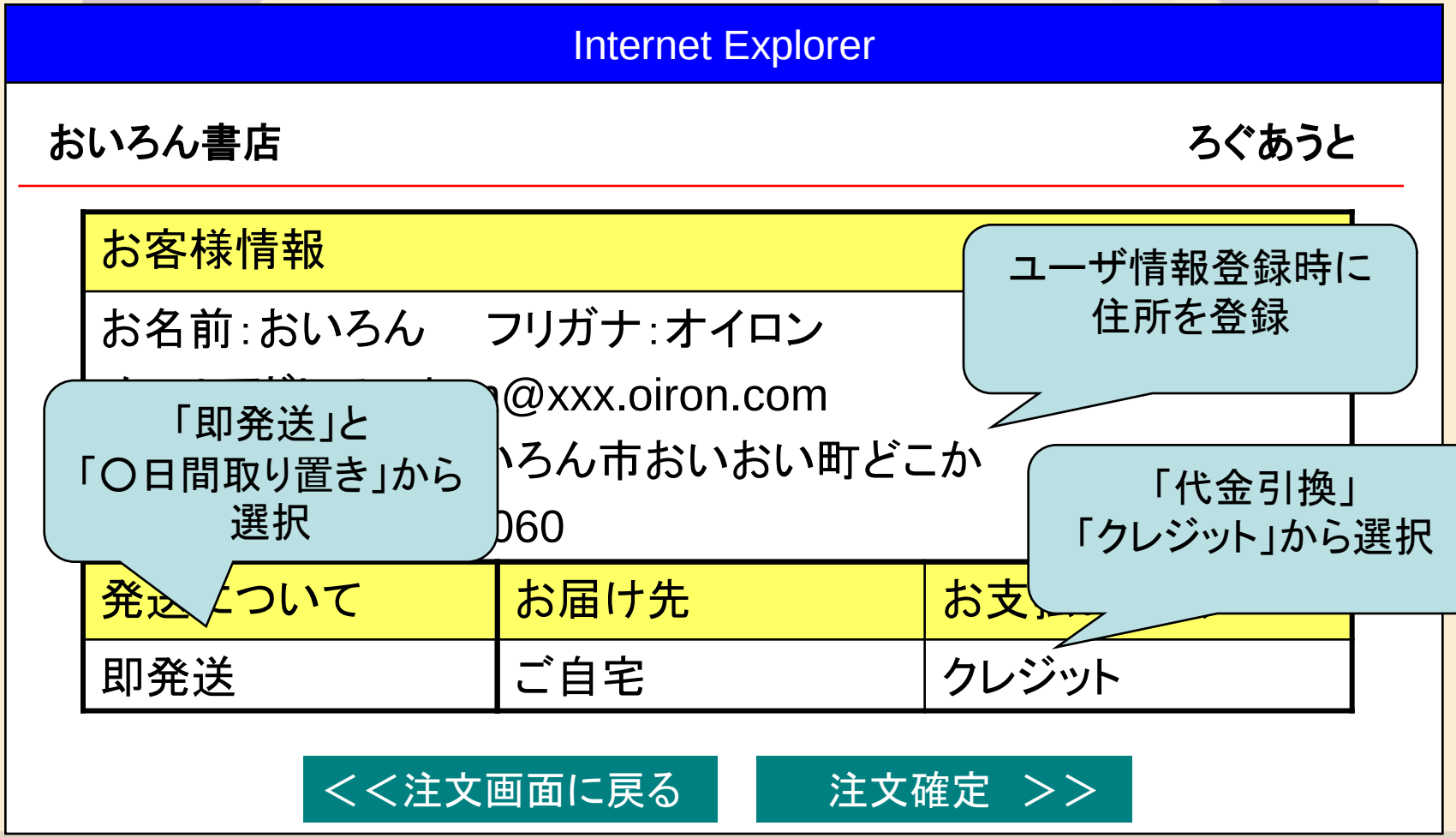
おいろん書店 **ろぐあうと**

注文番号:1234567890 **注文確認画面**

No	商品名	単価	数量	価格
1	現場で使えるDB設計	1,800	1	1,800
2	プロとしてのデータベースモデリング入門	1,400	1	1,400
3	おいろんの昔話	5,000	1	5,000
		小計金額		8,200
		合計金額		**8,200**

1. はじめに

- サンプルシステム



①対象領域を明確にする

- この画面は、注文を行う過程の「注文確認画面」である。
- 1回の注文で複数の商品を注文できる。
- 顧客の情報は前もって登録する
- 発送方法を「即発送」「取り置き」から選択できる
- 支払方法を「代金引換」「クレジット」から選択できる

②エンティティを洗い出す

- エンティティとは、ある目的を持って同じようなデータを集め、その目的を明確に表す名前をつけたもの。

画面上のカテゴリ（分類）名称		エンティティ名称
商品	→	商品
発送について	→	発送方法
お客様情報	→	顧客
お届け先	→	届け先
お支払い方法	→	支払方法

②エンティティを洗い出す

- 洗い出したエンティティで業務ルールを表現できるか、過不足をチェックする

  - エンティティが不足していないか？
  - 不要なエンティティがないか？

②エンティティを洗い出す

－不足しているエンティティの洗い出し

「誰が」「何を」「どうする」に該当するものを洗い出す
→洗い出せない場合は不足している

今回だと、「顧客」が「商品」を「注文する」

→「注文」エンティティがない

今回の場合、1度の注文で複数の注文商品が
存在するので、「注文明細」も必要となる

②エンティティを洗い出す

－不要なエンティティの洗い出し

エンティティの具体例（インスタンス）が1つしかない
（想定できない）場合は、エンティティになり得ない。

これは業務ルールに起因することがあるので、一概に
いえないものがある。

（例）届け先の入力を選択ではなく直接入力させる場合
などでは、エンティティとしては不要となる
（「注文」エンティティに含める）

②エンティティを洗い出す

顧客

顧客番号
名前 フリガナ Email 郵便番号 住所 電話番号

注文

注文番号
注文年月日 小計金額 合計金額 届け先名称 郵便番号 住所 電話番号

注文明細

注文番号 注文明細番号
数量 価格

発送方法

発送方法番号
発送方法名

商品

商品番号
商品名 単価

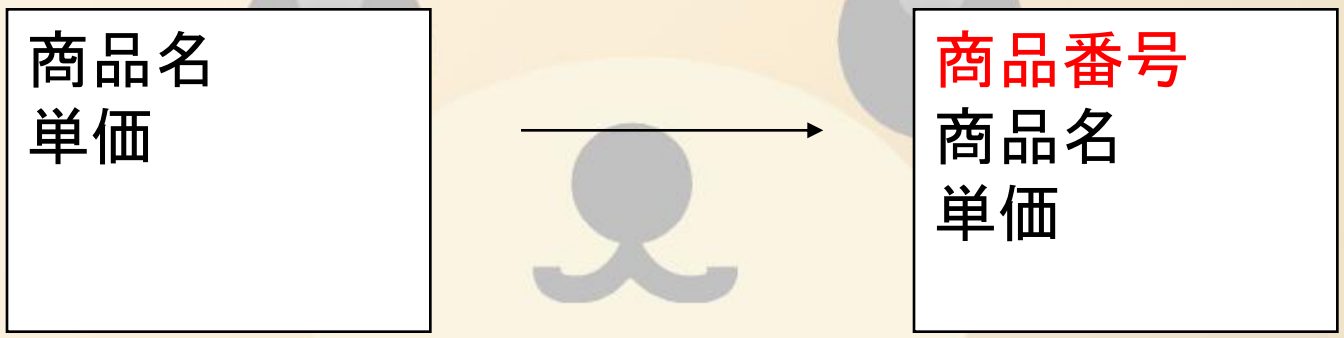
支払方法

支払方法番号
支払方法名

③リレーションシップを検討する

– 主キーを設定する

- 一意にインスタンスを識別するために設定する

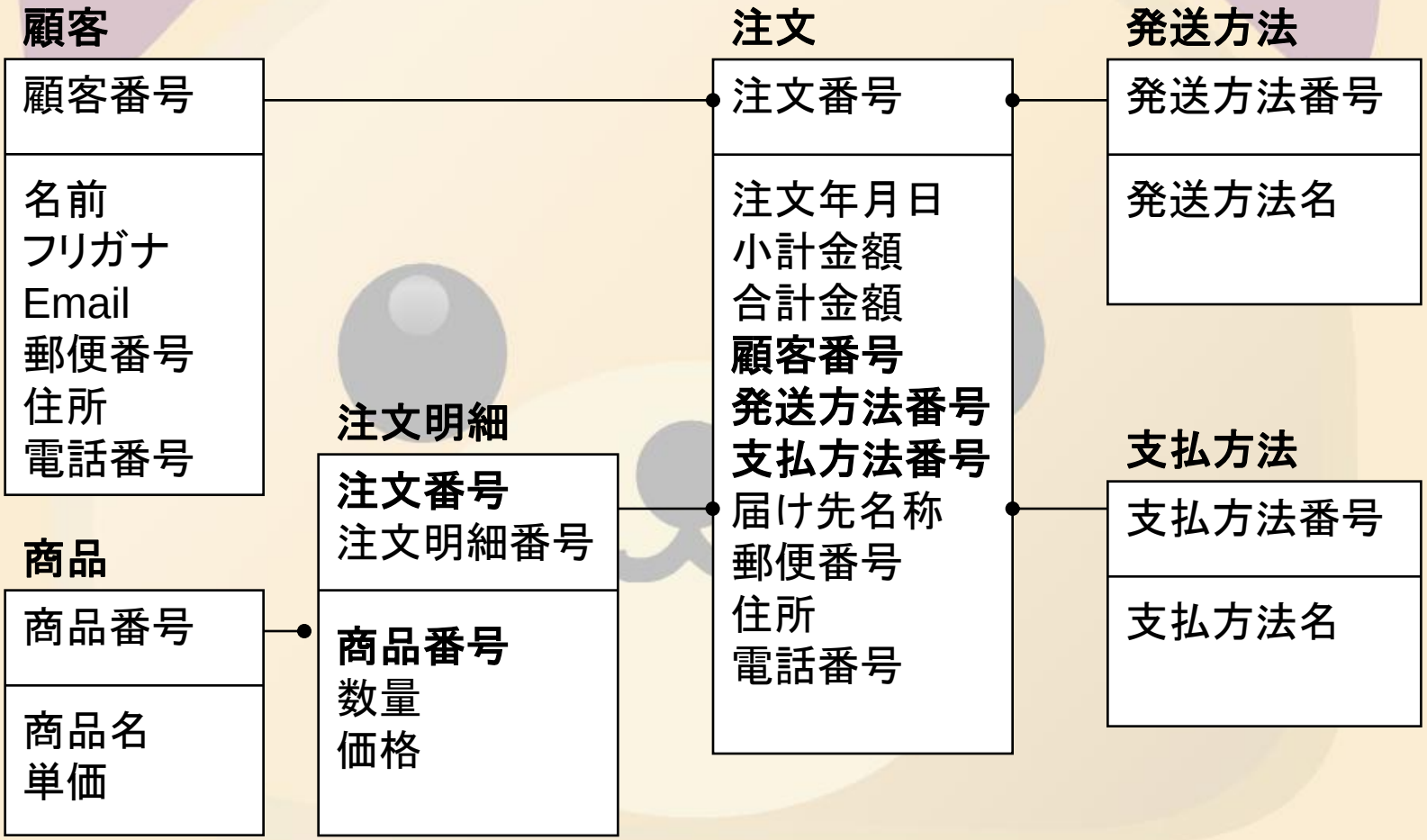


同一商品名、同一単価のデータがあると一意に識別できない
→必ず一意になるようコードを設定する

③リレーションシップを検討する

- 洗い出したエンティティで業務ルールを表現できるか、過不足をチェックする
- 「人」や「組織」や「顧客」などの「主体」を表現するようなエンティティと、「受注」「発注」「出荷」などの「行為」を表現するエンティティとの関係は、「主体」から「行為」に向かって1：多の関係となっている。

③リレーションシップを検討する



④概念データモデルを検証する

- 全データ項目がきちんと洗い出せているか確認する
- データの一元化がなされていなければ正規化を行う
- 正規化によりエンティティが変更された場合再度リレーションシップの定義を行う

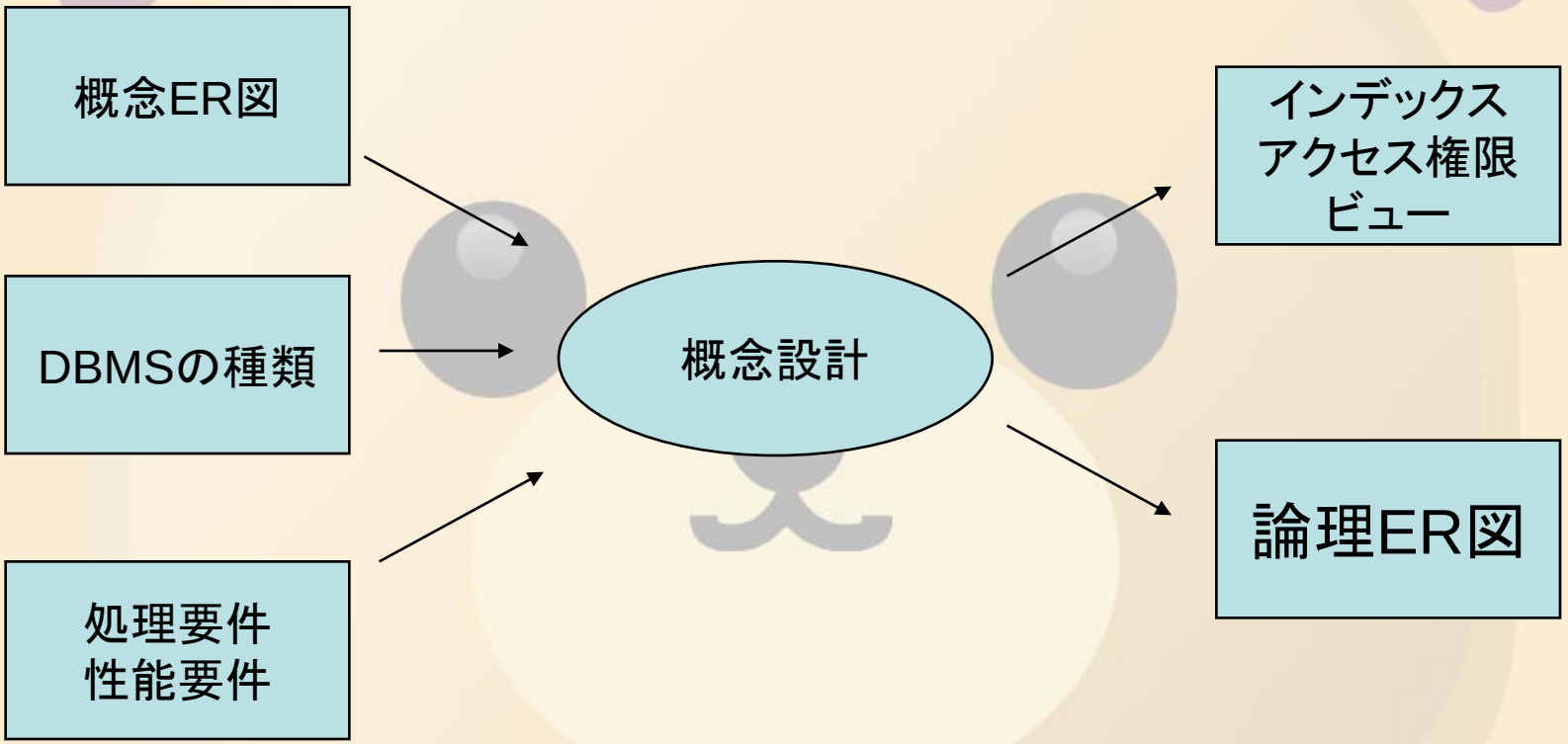
4. 論理設計

4. 論理設計

- 目的
  - 処理要件を考慮して適切なインデックスを設計する。
  - エンティティの統合、導出項目・重複項目の設定などの非正規化の検討を行う。
  - データの独立性、セキュリティの観点からビューを設計する。
  - プロセス開発工程と同期をとり、必要なエンティティや属性の追加を行う。

4. 論理設計

- インプットとアウトプット



4. 論理設計

- 手順

①業務に対するユーザ要件を検討する。

②データ量を見積もる。

③処理性能を出すための最適化の検討を行う。
　インデックス、非正規化の検討など

④バッチ処理などアプリケーションを考慮し、
　必要な属性を再検討する。

⑤一意識別子、一貫性制約の検討を行う。

⑥ビューとアクセス権限の設計を行う。

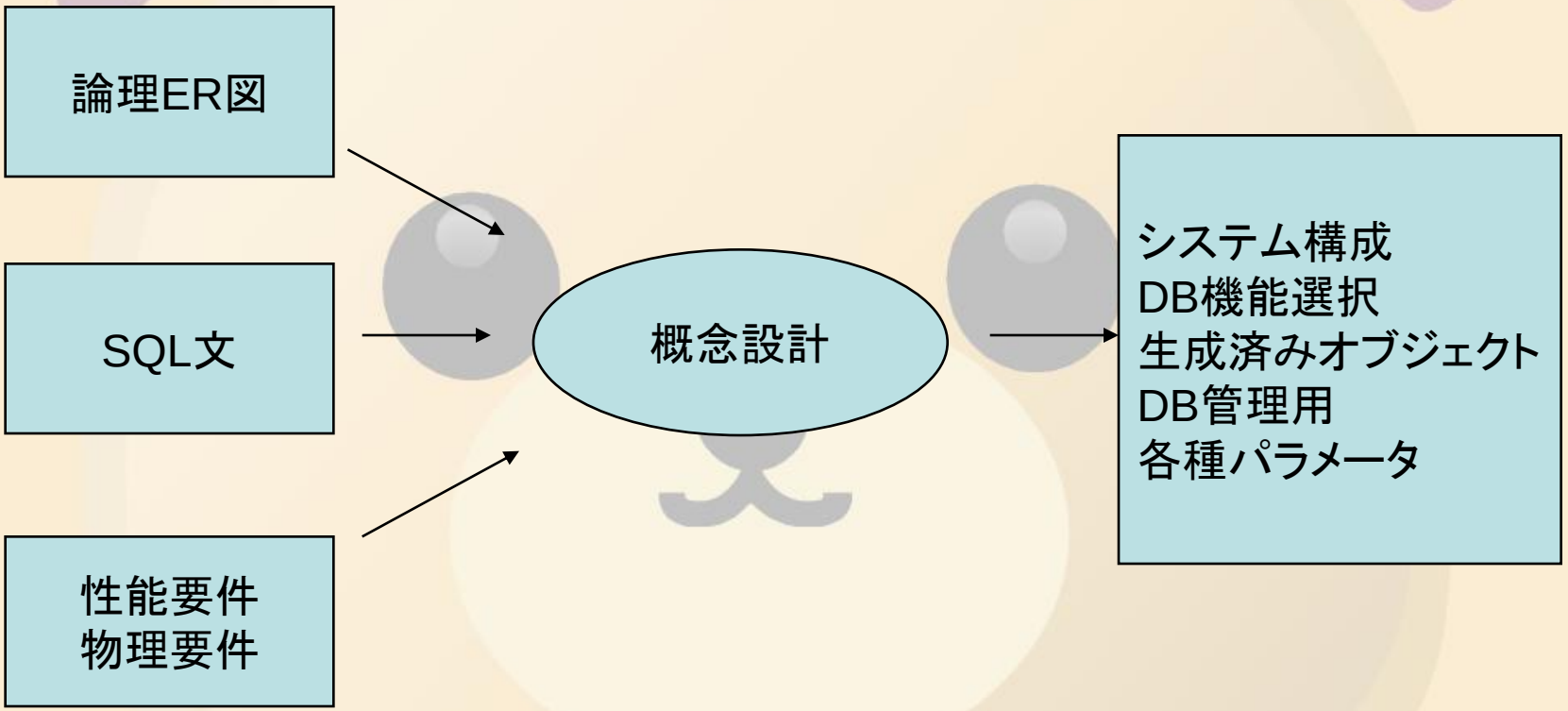
5. 物理設計

5. 物理設計

- 目的
  - ハードウェアの資源を使い切って、ユーザ要件を満たす性能と障害回復を中心として可用性の高いデータベースを実現する
    - 選択したDBの機能を十分に活用する
    - 限られたハードウェアの資源を使い切ることを考える。
    - モニタリングとチューニングの繰り返しで最適化する。
    - 性能を発揮するために、SQLの分析スキル、各種RDBMSの機能を熟知している必要がある。

5. 物理設計

- インプットとアウトプット



6. おわりに

6. おわりに

- 参考文献
  - 「プロとしてのデータモデリング入門」
    林　優子著　ソフトバンククリエイティブ
  - 「現場で使えるデータベース設計」
    NRIラーニングネットワーク株式会社
    中村　才千代　著　ソフトバンククリエイティブ
- 参考サイト
  - @IT　Database Expert
    http://www.atmarkit.co.jp/fdb/

6. 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？
  - 要件を熟知して、要件にあった設計をするが大切となります。
  - 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！！