まとめ

データベース論Ⅰ

第15回

URL　http://homepage3.nifty.com/suetsuguf/

作成者　　末次文雄 ©

修得目標

①データベースシステム設計の基本を理解できる。

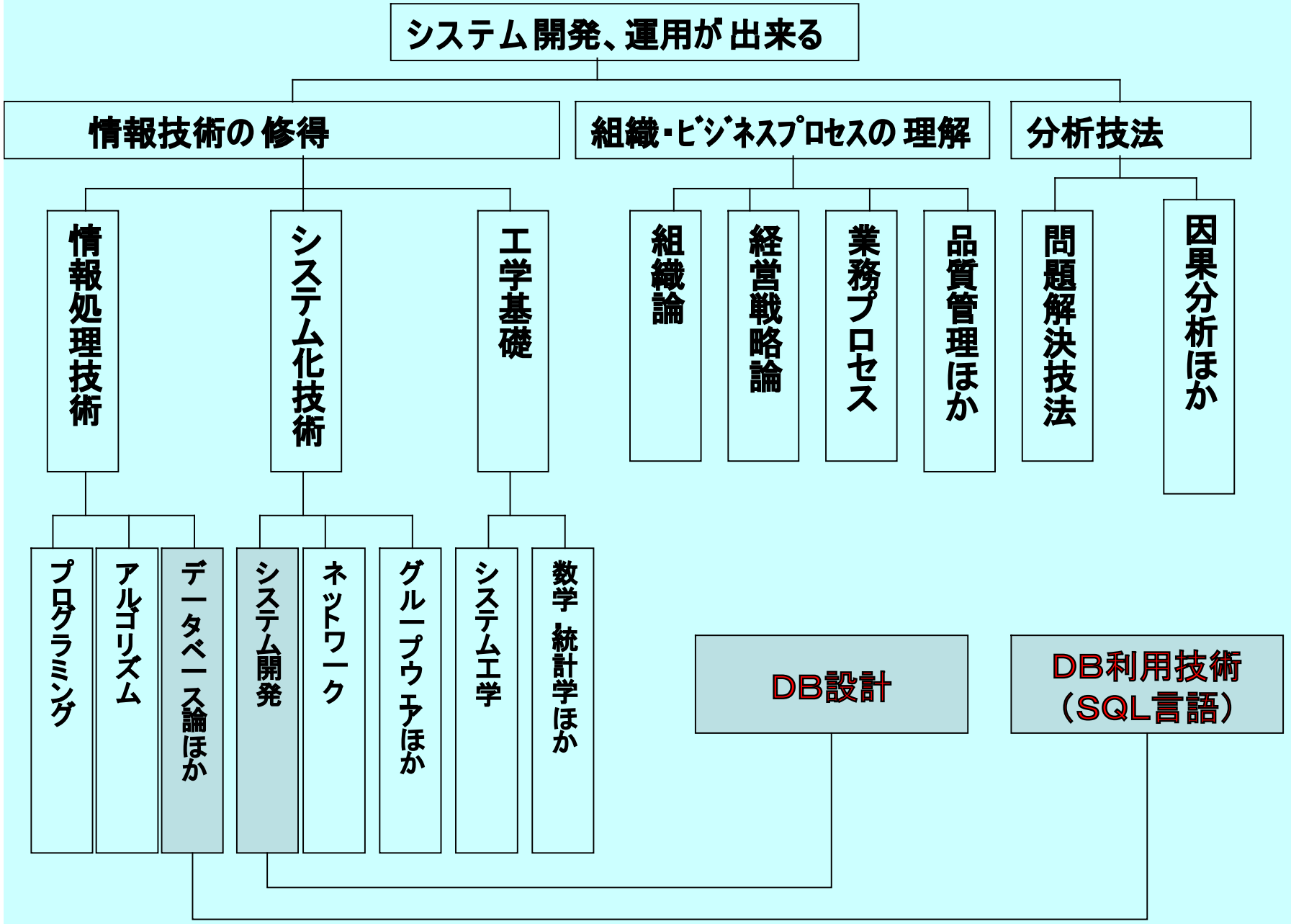
②リレーショナルデータベースのモデリング手法、
データ定義／データ操作方法、同時実行制御、
障害回復等について理解できる。

③上記を使い、簡単なデータベースシステムの設計
が出来る。

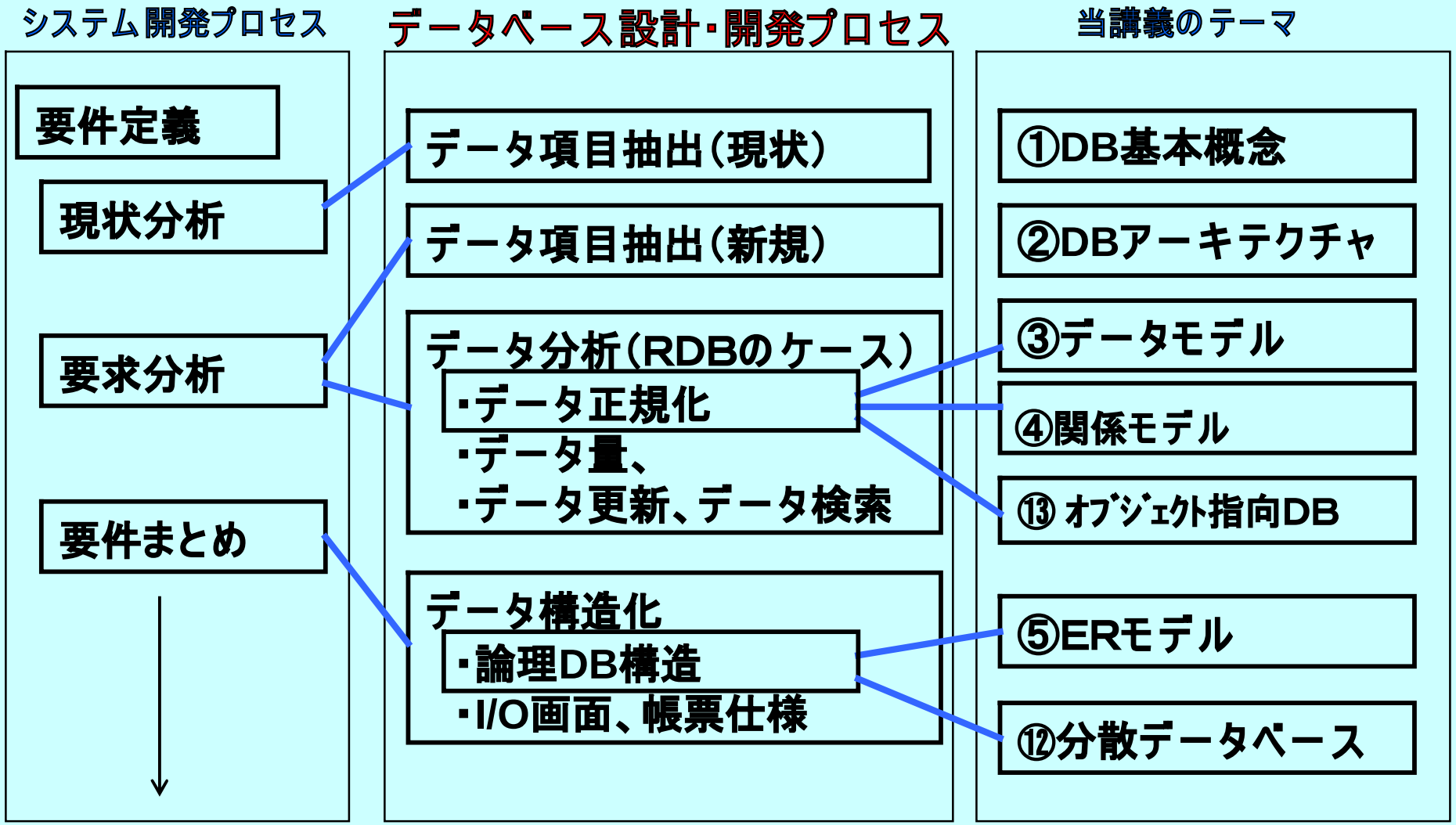
内容

第１回　データベースの基本概念
第2回　データベースのアーキテクチャ
第3回　データモデリングの基本
第4回　リレーショナルデータベース（関係モデル）
第5回　リレーショナルデータベース（ ER手法）
第6回　リレーショナルデータベース言語（SQL）1
第7回　リレーショナルデータベース言語（SQL）1
第8回　問合せ処理
第9回　トランザクション処理
第１０回　同時実行制御
第１１回　障害回復
第１２回　分散データベース
第１３回　オブジェクト指向データベース
第１４回　事例および技術動向
第１５回　まとめ

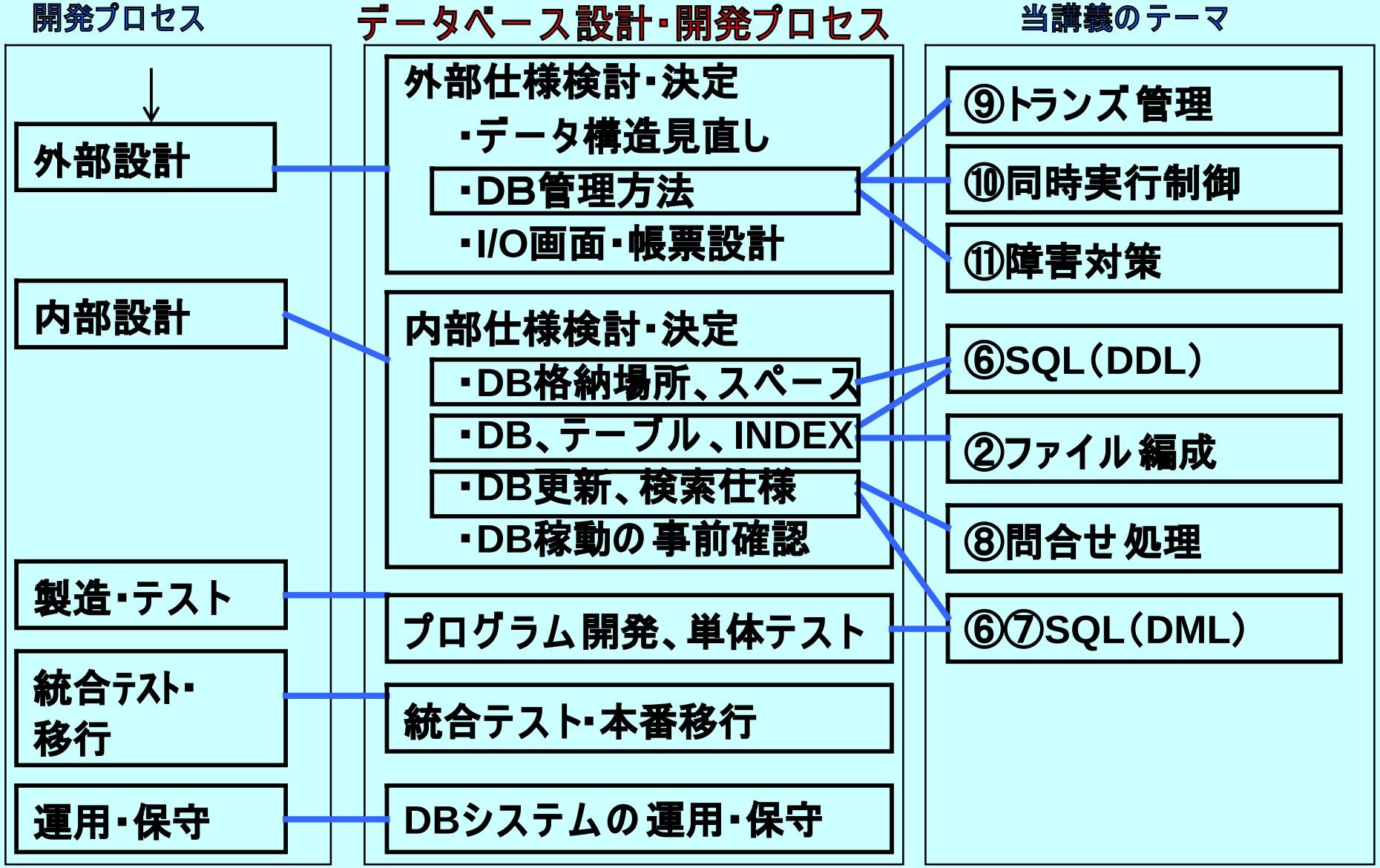
知的情報システム工学科における当演習科目の位置付け



データベース設計・開発のプロセス



（続き）


開発プロセス
データベース設計・開発プロセス
当講義のテーマ
外部設計
内部設計
製造・テスト
統合ﾃｽﾄ・
移行
運用・保守
外部仕様検討・決定
・データ構造見直し
・DB管理方法
・I/O画面・帳票設計
内部仕様検討・決定
・DB格納場所、スペース
・DB、テーブル、INDEX
・DB更新、検索仕様
・DB稼動の事前確認
プログラム開発、単体テスト
統合テスト・本番移行
DBシステムの運用・保守
⑨トランズ管理
⑩同時実行制御
⑪障害対策
⑥SQL（DDL）
②ファイル編成
⑧問合せ処理
⑥⑦SQL（DML）

データベースの定義

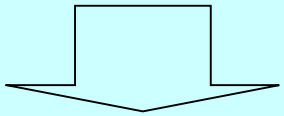
データベースの定義

- 組織体の、統合化された共有の情報資源。
- 種々の利用を可能とするために、

DBMSの管理下で、データを体系化し、
電子媒体に記憶させ、
必要時に、効率良く利用できる状態に
保たれたデータの集合である。

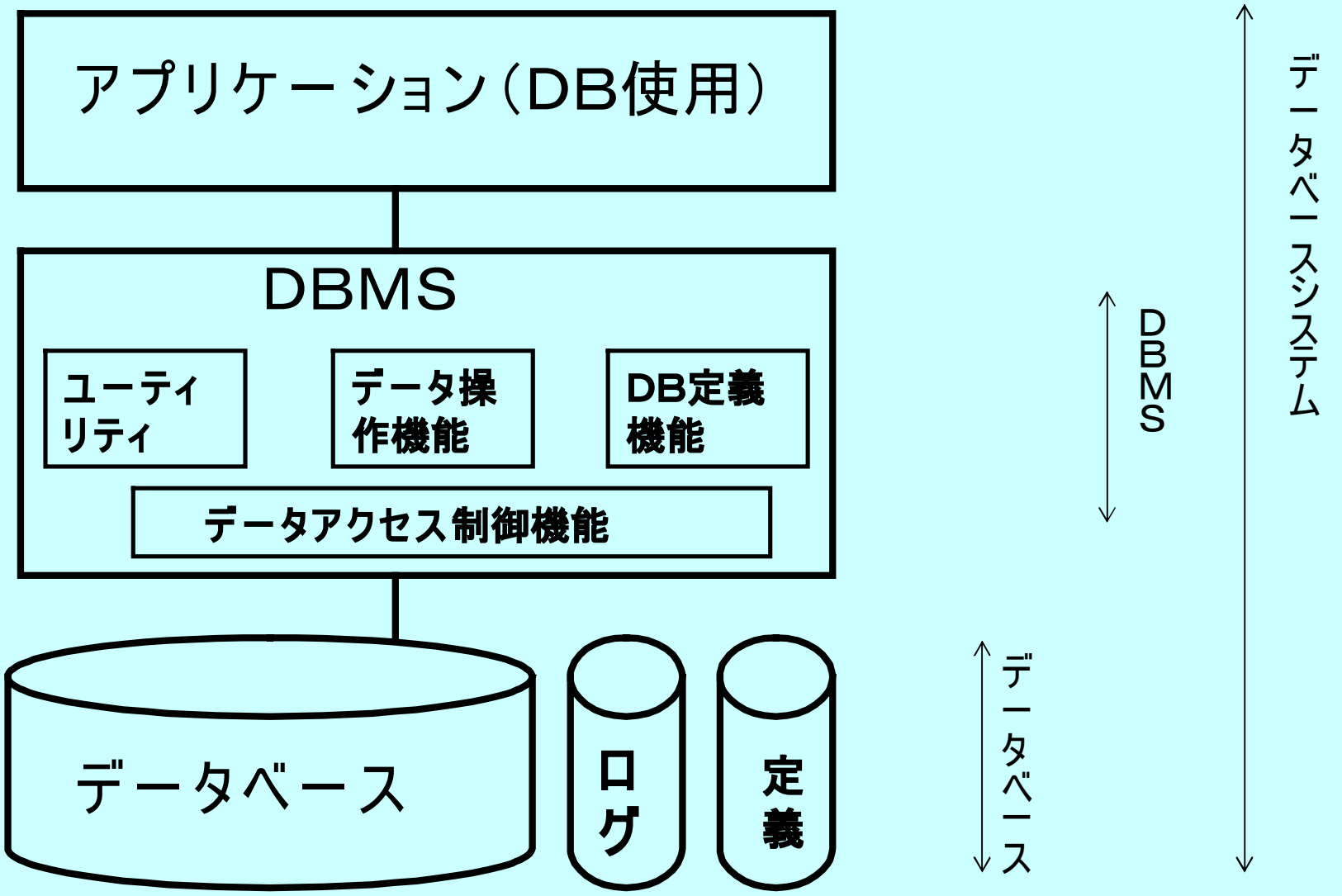
データ独立

・ファイルのデータ構造、データ定義、
格納場所などが変更されても、
それを使用するアプリケーション・プログラム
に対する影響を最小限に抑える。

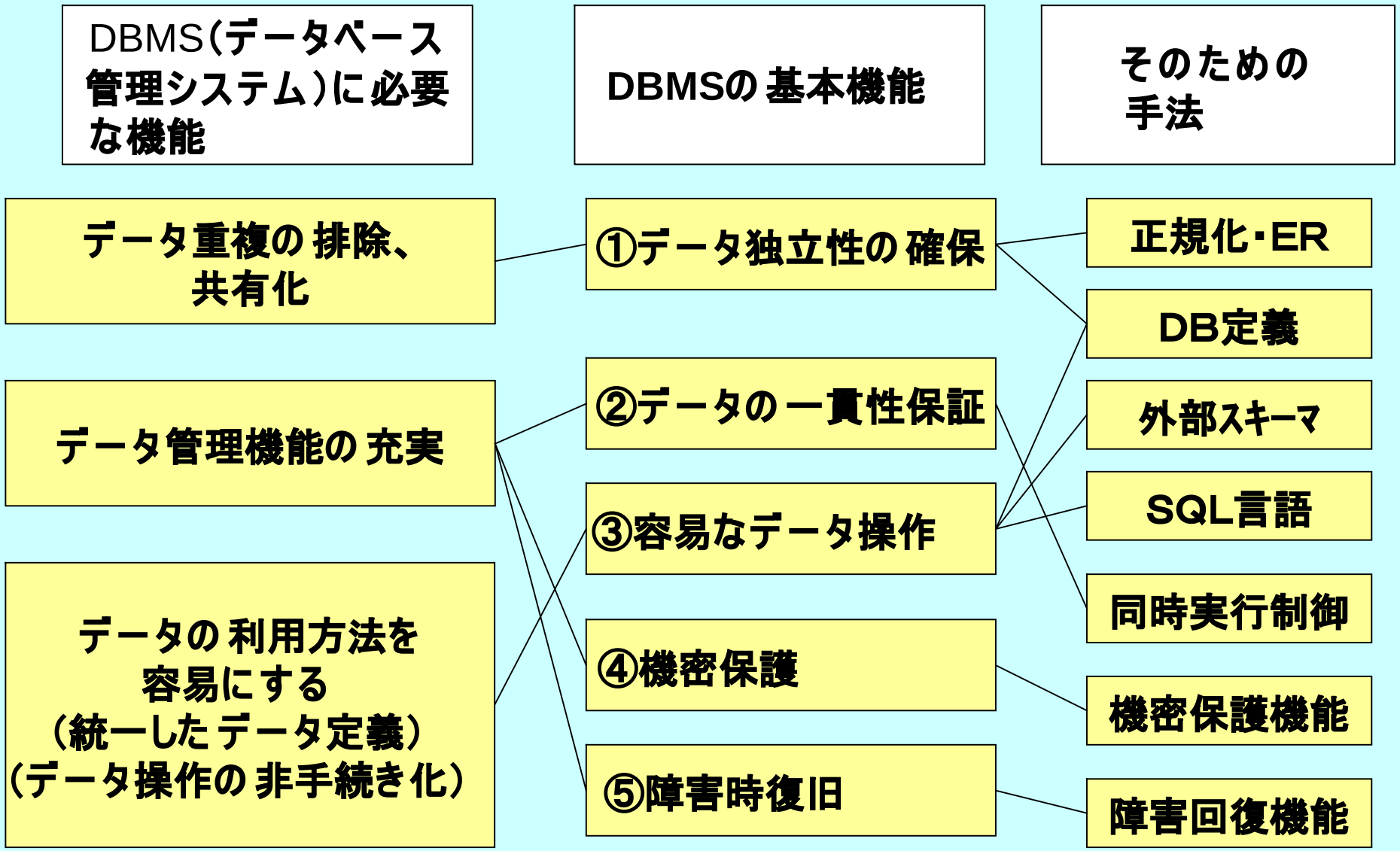


・そのためには、レコードやファイルの
－物理的な記憶媒体、記憶場所、
－ファイルの編成方法、
－レコード内のフィールドの配置
－レコードの順序
などの情報をﾌﾟﾛｸﾞﾗﾑから独立させて定義し保管する。

データベースシステムの構成



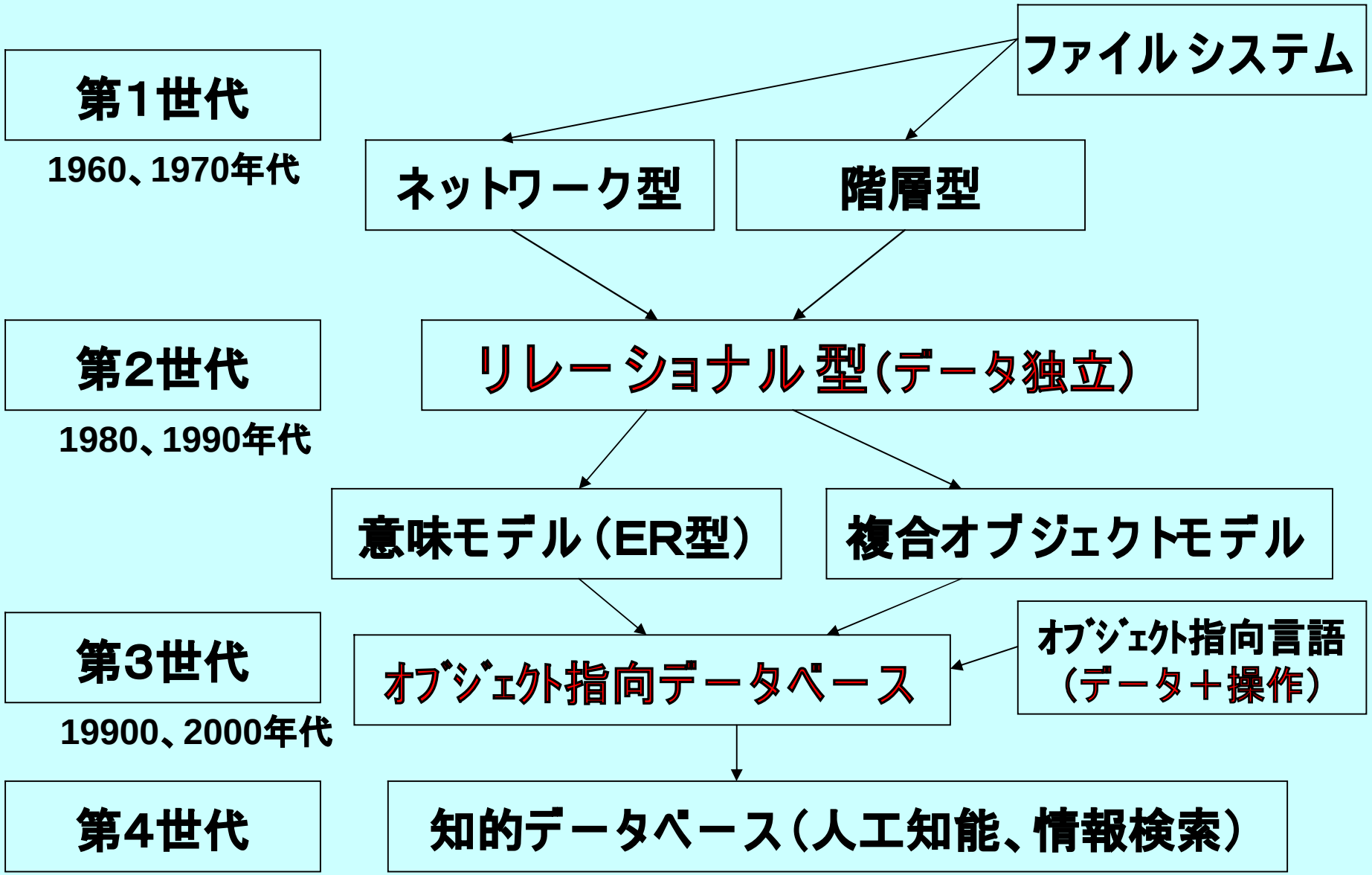
DBMSの 役割、機能と手法



データベースの３層構造

- 概念スキーマ（conceptual　schema）
  ーデータのモデリング結果を論理構造として
  を定義したもので特定のﾌﾟﾛｸﾞﾗﾑに依存しない。
- 外部スキーマ（external　schema）
  ー個々のﾌﾟﾛｸﾞﾗﾑから見たデータの構造を記述
  するものでViewという。
  ー概念スキーマの変更をプログラムに及ぼさない
  ために設ける。
- 内部スキーマ（internal　schema）
  ー概念スキーマを計算機上に実現する方法
  であり、データベースの物理的な格納方法や
  格納スペースや編成方法を記述するもの。

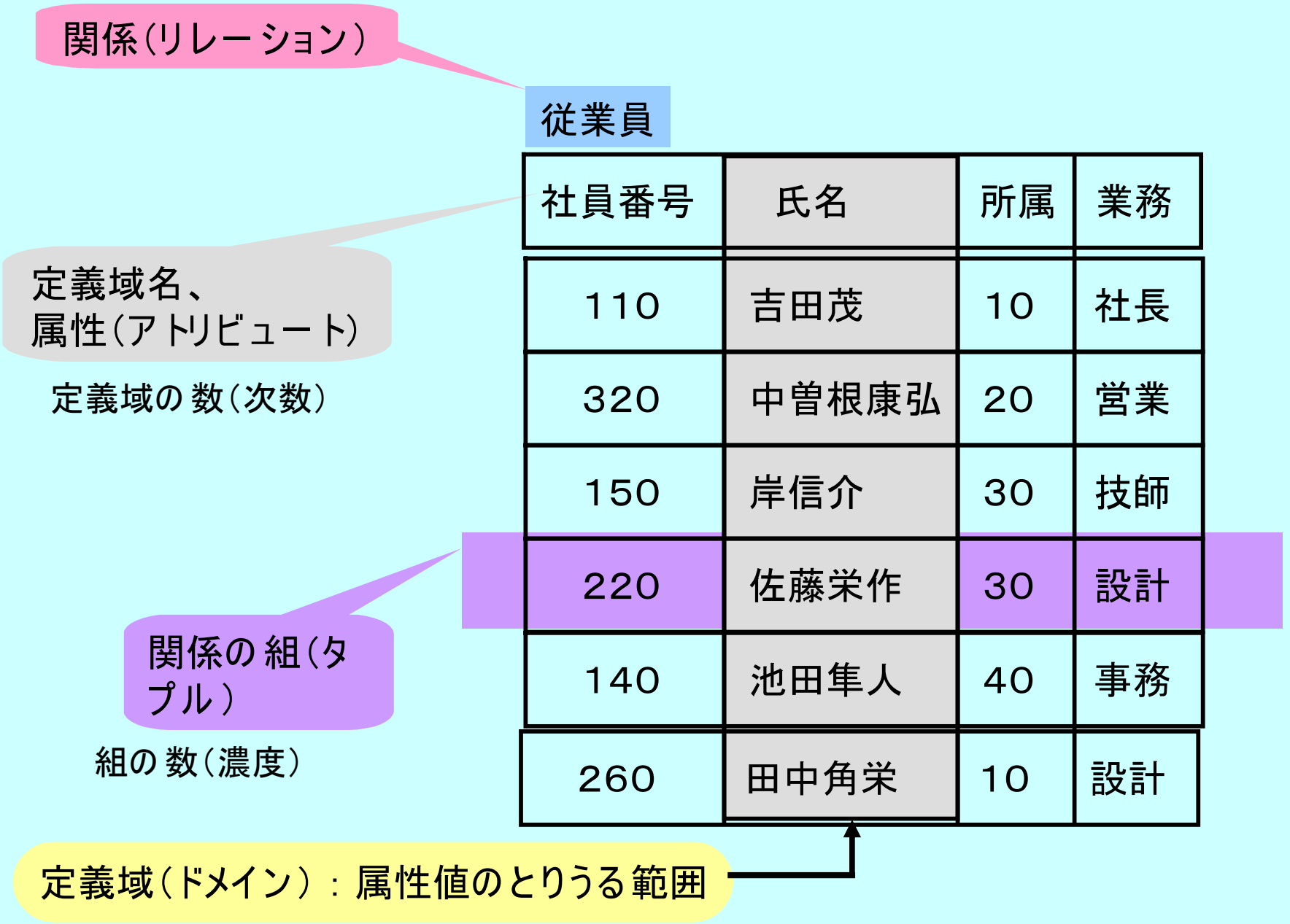
データモデルの発展経緯



関係データモデル

- Relational data model
- 1987年、データベース言語のSQLが国際標準
- 明解な数学的基盤の上に成り立つモデル
- フラットな表形式という非常に単純な表現形式
- モデリング時に、データ構造が固定されないので
  データ独立性が高い
  （表同士の関連(データ構造)を規定する定義が無い）
- データ利用時に、集合論に基づいた関係操作により
  動的に自由にデータの結合や分解が行える

関係(リレーショナル)データベースの構造



関係の集合演算の種類

1. 一般的な集合演算
   - ①和（集合演算）、Union
   - ②差（集合演算）、Difference
   - ③積（集合演算）、Intersection
   - ④直積（集合演算）、Cartesian　Product
2. 関係に特有な集合演算（関係演算）
   - ⑤選択（演算）、Selection
   - ⑥射影（演算）、Projection
   - ⑦結合（演算）、Join

データ正規化

第１正規形

・これ以上分解できないデータ項目のみで構成。

・繰り返しのデータ項目を持たせない。

・直積空間で、必ず単独の点として識別できる。

第２正規形

・キーの一部分に従属しているデータ項目を分離。

・キー以外の全てのデータ項目がキーに従属（完全従属）

第３正規形

・キー以外のデータ項目間の従属関係を分離。

・推移的従属データ項目の分離ともいう。

表(テーブル)におけるキーの種類

- 主キー(Primary Key)
  行を一意に識別するもの、複数の列指定可
- 候補キー(Candidate Key、代替キー)
  行のキー候補(社員番号と社会保険番号等)
- 外部キー(External Key、参照キー)
  他の表の主キーになっているもの
- 連結キー(Concatinated Key)
  複数の列からなる複合された組合せキー

いずれも、ERモデリングのためには、重要な情報となる。

ERモデルの基本概念

①エンティティ（Entity、実体、事象）

・企業、組織にとって関心があり、継続的に管理の対象になるものである。

・オブジェクト指向でのクラス概念の大きさに同じ

②属性（アトリビュート）

・エンティティの性質（プロパティ）を示すもので、各エンティティは、複数の属性を持つ。

③関連（Relationship）

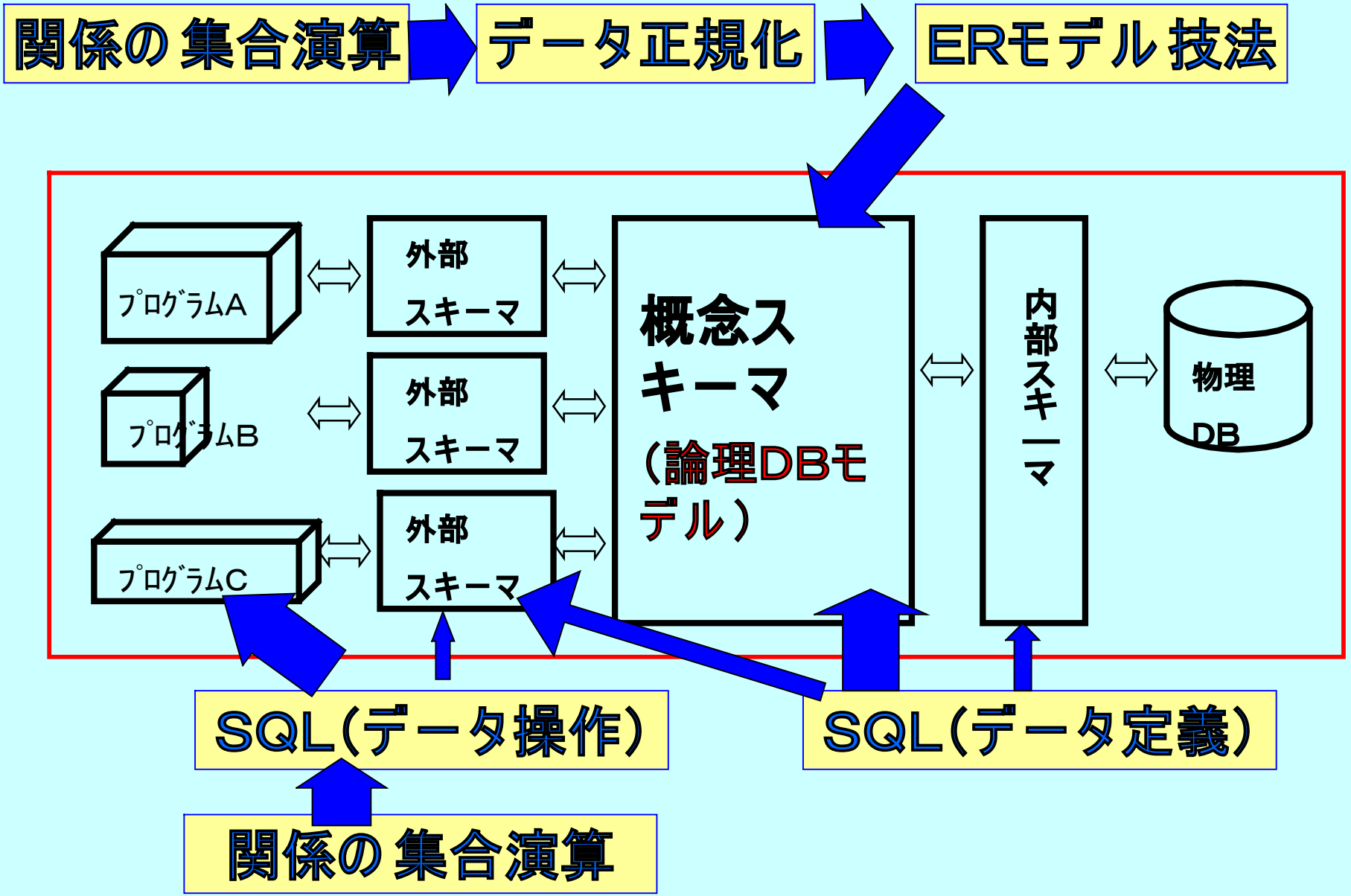
・エンティティ間の関係性のこと。

・オカレンスの発生に関して、種々の制約あり。

- 基数制約（カーディナリティ制約（1：1、1：N））
- 存在依存制約（Existence　Dependency）

・再帰的関連、選択的関連、3項関連の種類あり。

スキーマと技法・SQLの対応



SQLの特長

①関係データモデルに準拠

②データ操作は公的機関により標準化されている

③大半のRDBMS製品が採用している規格

④データアクセスだけでなく、データ管理機能も
　併せて装備している

⑤データをまとめて処理する方法（セット処理）、
　一件づつ処理する方法を共にサポートする

SQLの 機能

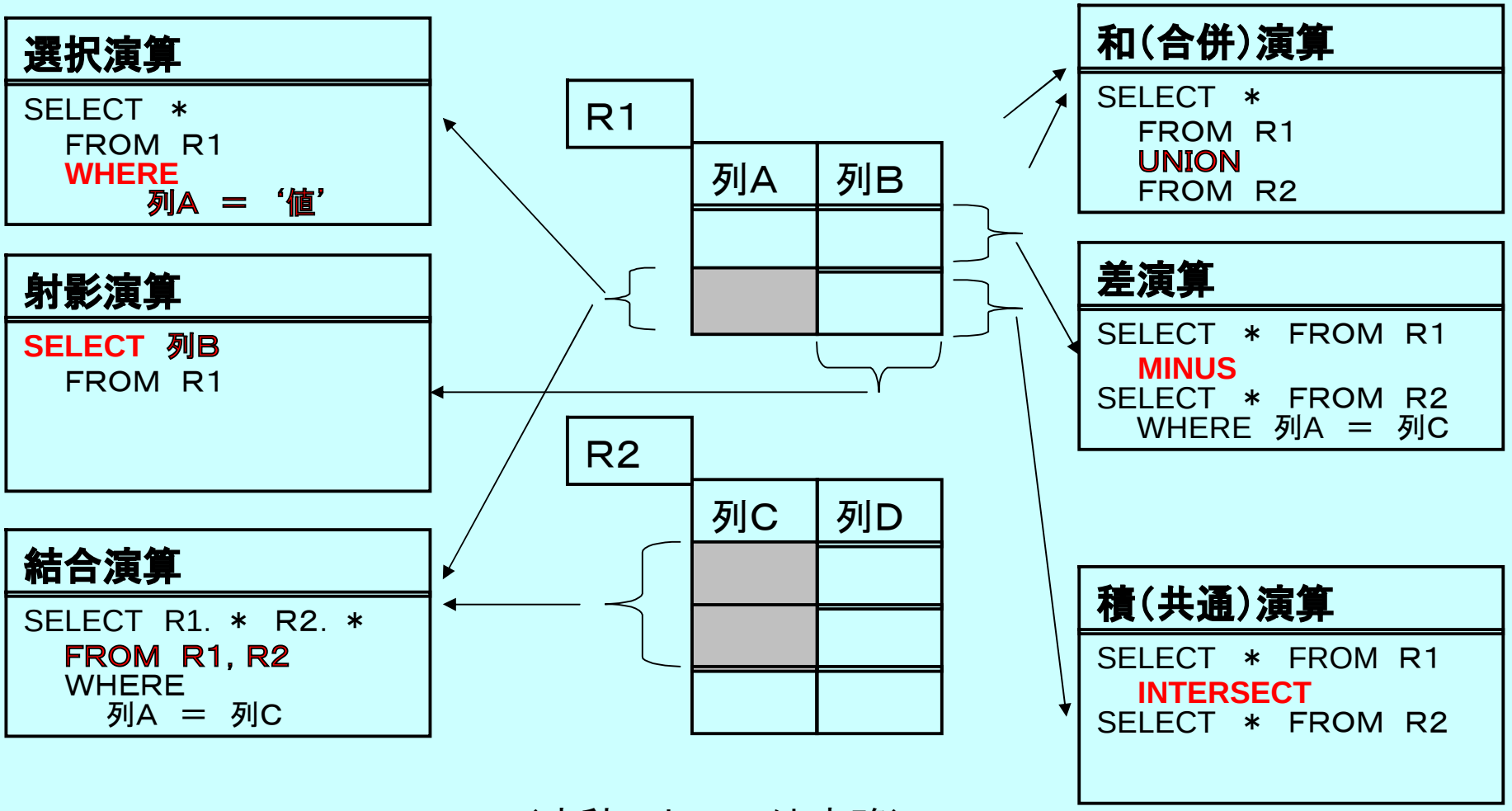
- **データ定義・管理**
  - **スキーマ定義**
    表（テーブル）、VIEW、INDEX作成
    （CREATE、ALTER、DROP）
  - **権限の 管理**
    データ利用の 権限付与、取消し
    （GRANT、REVOKE）
  - **処理の 制御**
    ﾄﾗﾝｽﾞｱｸｼｮﾝ管理、カーソル 操作
    （COMMIT、ROLLBACK、
    CURSOR）
- **データ操作**
  - **データ変更**
    データの 追加、削除、更新
    （INSERT、DELETE、UPDATE）
  - **データ検索**
    データの 取り出し（検索）
    （SELECT）

SQLの利用形態

- **独立言語方式** — SQLとは別の問合せ専用の言語を使う
  - **QBE** — 用意されたスケルトン（骨組み）に問合せ条件を入力する（Query By Example）
  - **コマンド方式** — SQLに付属するコマンドプロセッサーを使って、SQLコマンドを入力（エンドユーザー向き）
- **ホスト言語方式** — 通常のプログラミング言語からデータアクセスが出来る
  - **埋込み方式** — プログラム内にSQL命令を書く
    - **静的SQL** — 実行時、SQL命令が不変
    - **動的SQL** — 実行時、SQL命令を変える
  - **モジュール方式** — プログラムからサブプログラムを呼び出す（ストアドプロシジャー、関数方式ともいう）

表(テーブル)の検索操作

関係の集合演算がSQLで可能である。

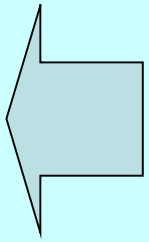


(直積のケースは省略)

データ操作（基本編）

単純なDB検索

```
SELECT  商品名,価格,仕入先
    FROM  商品価格TBL
    WHERE  価格＜10000
    ORDER  BY  商品コード；
```



商品価格TBL

商品コード	商品名	価格	仕入先	仕入単価

SELECT文の中の説明

SELECT　計算式
FROM　表名
WHERE　条件式
ORDER　BY　行の順序

グループ化のSQL使用例

グループ化したデータに条件設定。

```
SELECT  列1, 集計関数([＊ | ALL | DISTINCT]  列2)
        FROM  表A
        GROUP  BY  列1
            HAVING  集計関数の検索条件  ;
```

```
SELECT    科目コード, COUNT(＊), AVG(成績)
          FROM  表A
          GROUP  BY  科目コード
              HAVING   COUNT(＊)＞10     ;
```

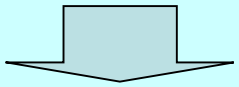
データ操作(副問合せ)

・問合せの中で、更に問合せを行う。
　結果が一つの事例

・複数あればエラー

『売上数の平均を下
回っている商品IDは？』

```
SELECT　AVG(売上数)　FROM　売上TBL;
答えは、5になる。
SELECT　商品ID　　FROM　売上TBL
　　WHERE　売上数<5　　;
```



```
SELECT　商品ID　　FROM　売上TBL
　　WHERE　売上数<
　　(SELECT　AVG(売上数)
　　FROM　売上TBL)　;
```

例：売上TBL

商品ID	売上数
A1	2
A3	8
C2	4
D1	5
D3	6

単純なDB更新

```
INSERT  INTO  商品価格TBL
    （商品ｺｰﾄﾞ、商品名、価格、仕入先、単価）
    VALUES （10030,’商品名BB’, 1500,
                ‘Y商事’,1200）；
UPDATE  商品価格TBL
    SET  商品名  ＝ ‘商品名称AAA’
     WHERE  商品コード ＝ 10010 ；
DELETE
    FROM  商品価格TBL
    WHERE  商品コード ＝ 10070 ；
```

追加

上書き

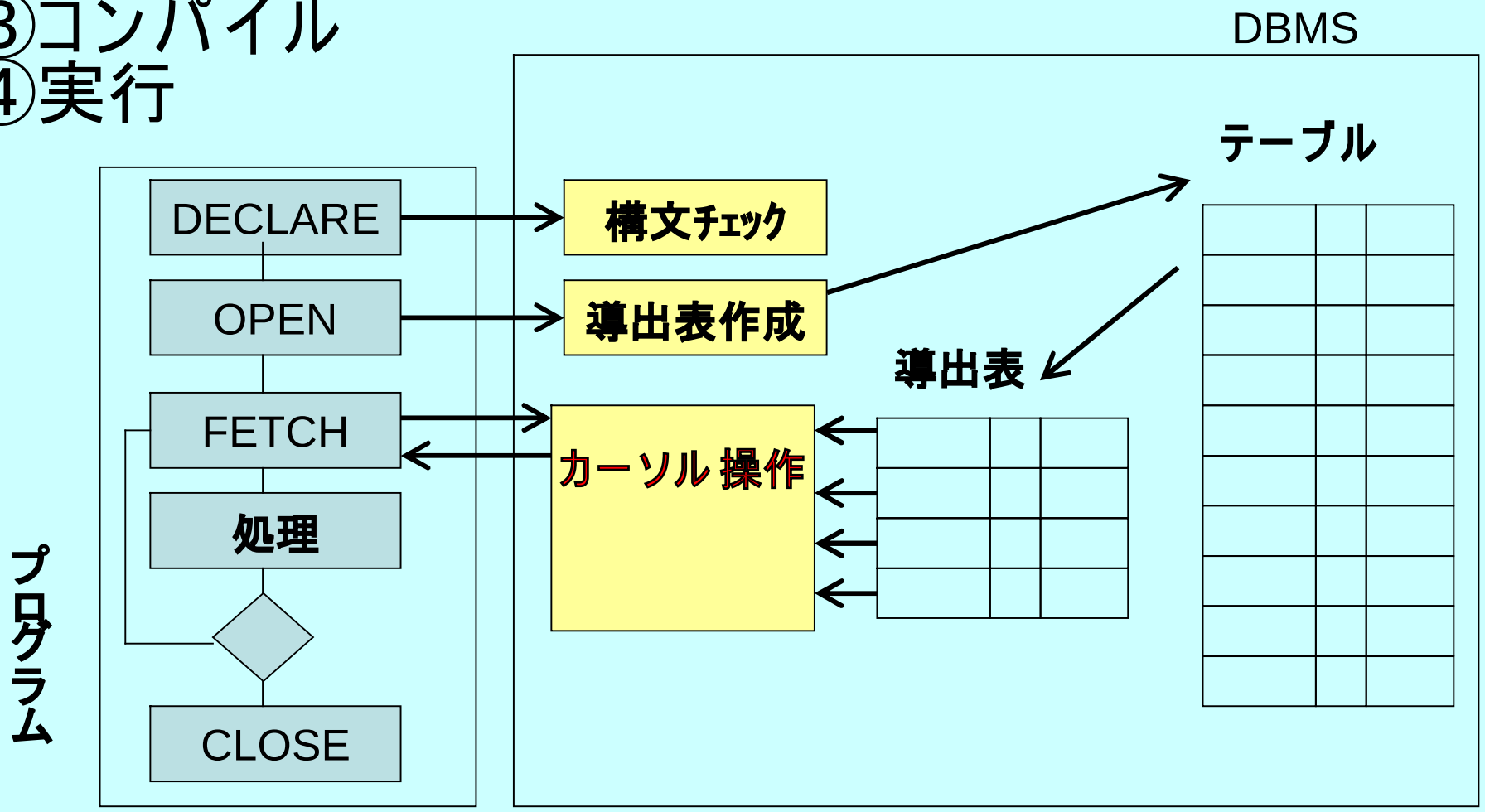
削除

商品価格TBL

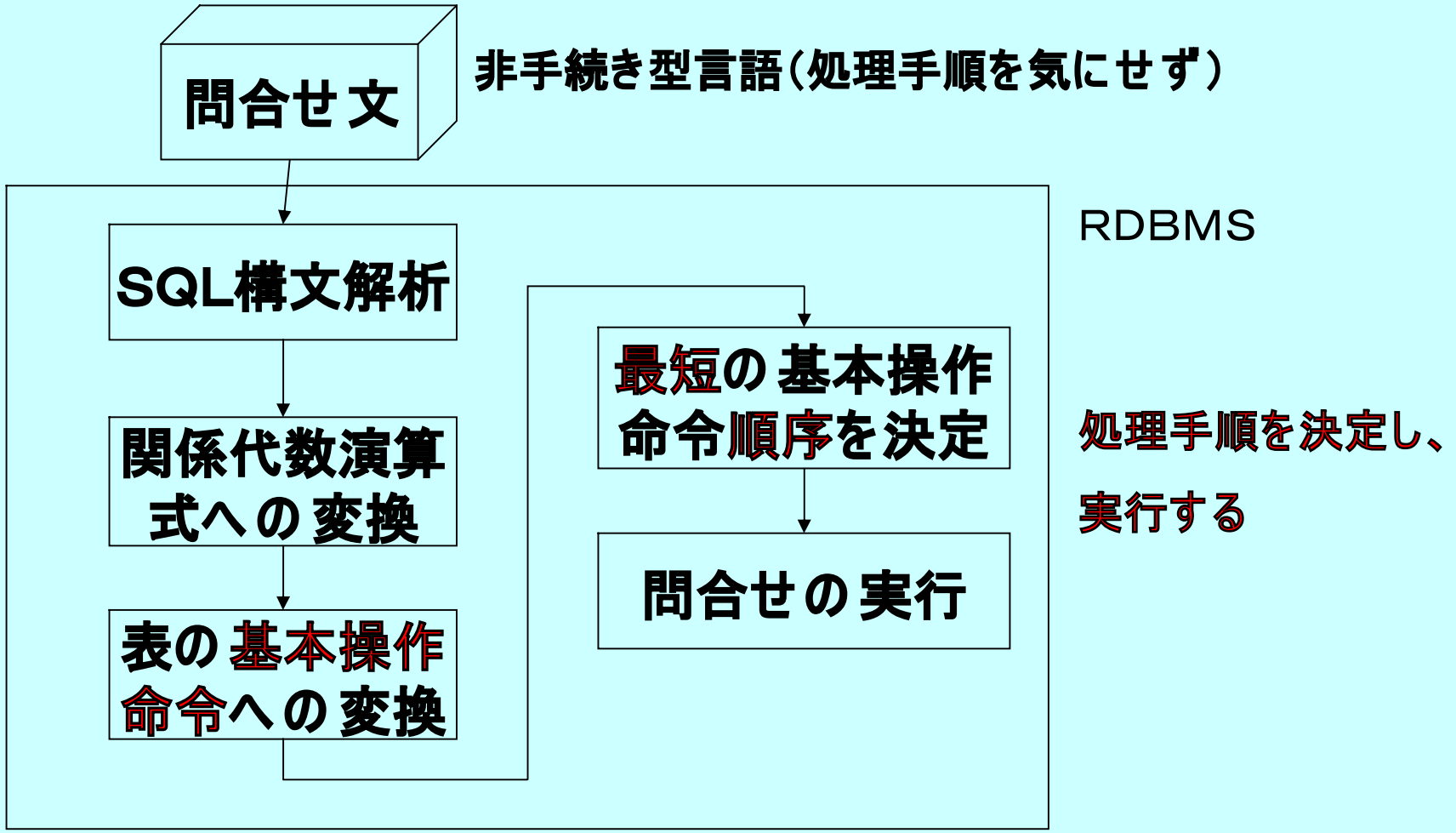
商品コード	商品名	価格	仕入先	仕入単価

カーソル操作の実行手順

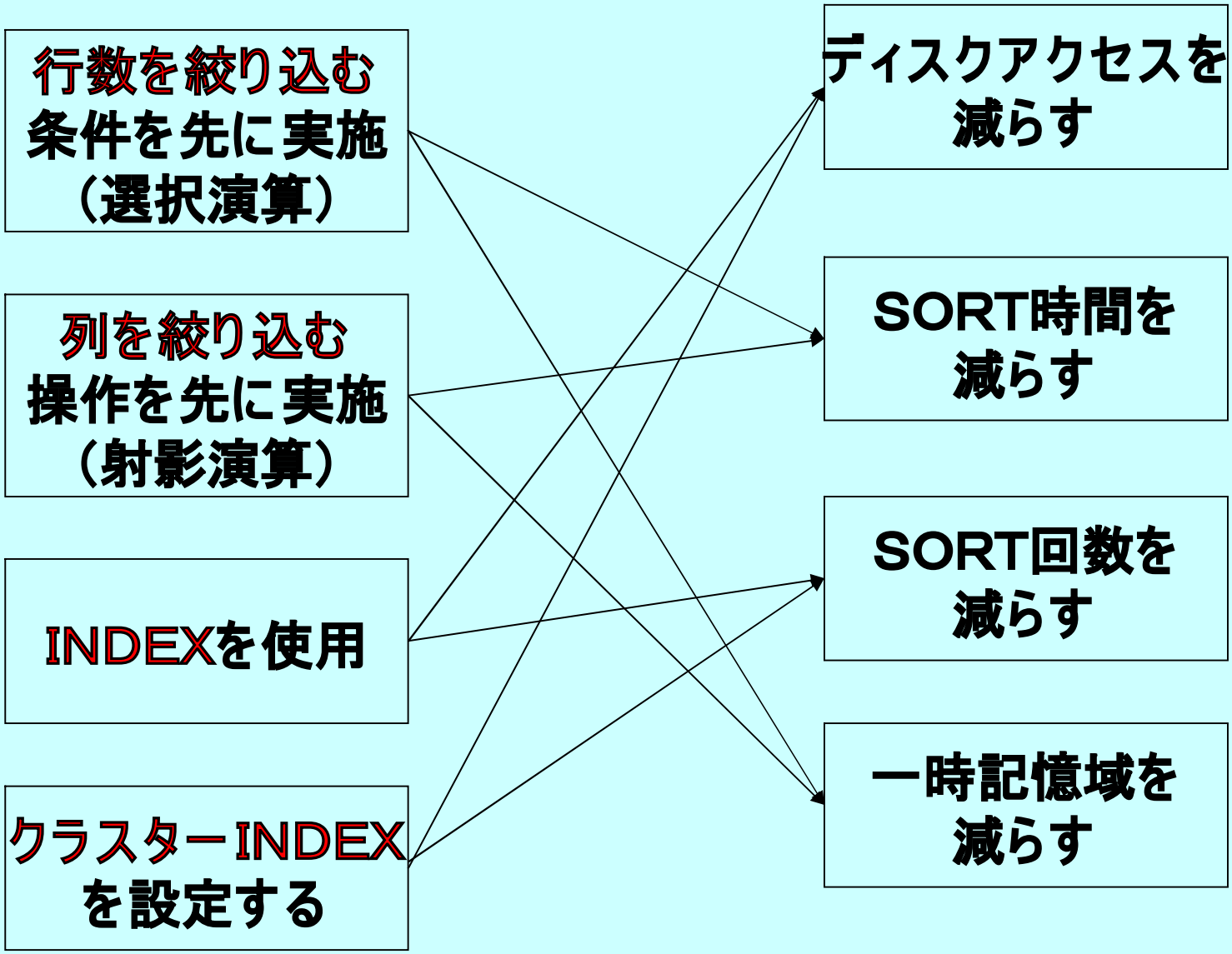
①ホスト言語にSQL文をコーディング
②プリコンパイル
③コンパイル
④実行



問合せ処理の手順


問合せ文
非手続き型言語（処理手順を気にせず）
RDBMS
SQL構文解析
関係代数演算式への変換
表の基本操作命令への変換
最短の基本操作命令順序を決定
問合せの実行
処理手順を決定し、
実行する

問合せ処理の最適化の方法



性能チューニング

1. データ構造の見直し
   - ①導出データの保管
   - ②逆正規化
2. INDEXの追加
3. その他
   - ①プログラムの改良
   - ②オプティマイザーの調整
   - ③RDBMSのパラメータ変更
   - ④HWの増強

トランザクション管理（ACID特性）

1） 一つのトランザクションについての管理項目

①Atomicity （原子性、非分割性） ← 障害回復制御

トランザクションは、その全てを反映するか、全て取り消すかの二者択一

②Consistency （一貫性、整合性） ← 同時実行制御

トランザクションの処理結果は、処理前後いずれも整合性が保持できる。
また、処理の順序、終了状態に関係なく保証できること。

2） 複数のトランザクションについての管理項目

③Isolation （独立性、分離性、隔離性） ← 同時実行制御

同時並行処理結果も、ある逐次処理結果も同じであること。
処理途中では、データは他から見えず、他の処理から影響を受けない。

3） 障害に対する管理項目

④Durability （耐久性、持続性） ← 障害回復制御

処理終了後は、その後の障害等で影響を受けず、不変に保たれること。

２相ロッキングプロトコル

①ロックは、ロックをかける操作と、ロックを解除する操作の二つに明確に分離する。

（ロック、ロック、ロック→解除、解除、解除）

成長相　　**縮退相**

②従って、一度ロックを解除した後では、再びロックをかけることを許さない規約である。

③ロック解除までの時間が長くなりやすく、デッドロックが生じやすくなる。

トランザクション制御のSQL文

トランザクション開始

```
SET TRANSACTION {READ ONLY | READ WRITE}
```

```
BEGIN WORK
```
（ORACLE）　（SQL-92）

```
BEGIN TRANSACTION
```
（SQL Server）

DBに反映する

```
COMMIT
```
（SQL-92）

```
COMMIT WORK
```
（ORACLE）

```
COMMIT TRANSACTION
```
（SQL Server）

元に戻す

```
ROLLBACK
```
（SQL-92）

```
ROLLBACK WORK
```
（ORACLE）

```
ROLLBACK TRANSACTION
```
（SQL Server）

（注記）制御用の機能、SQL文は、DBMSによって、違いが大きいので、各DBMSメーカーのマニュアルを参照のこと。

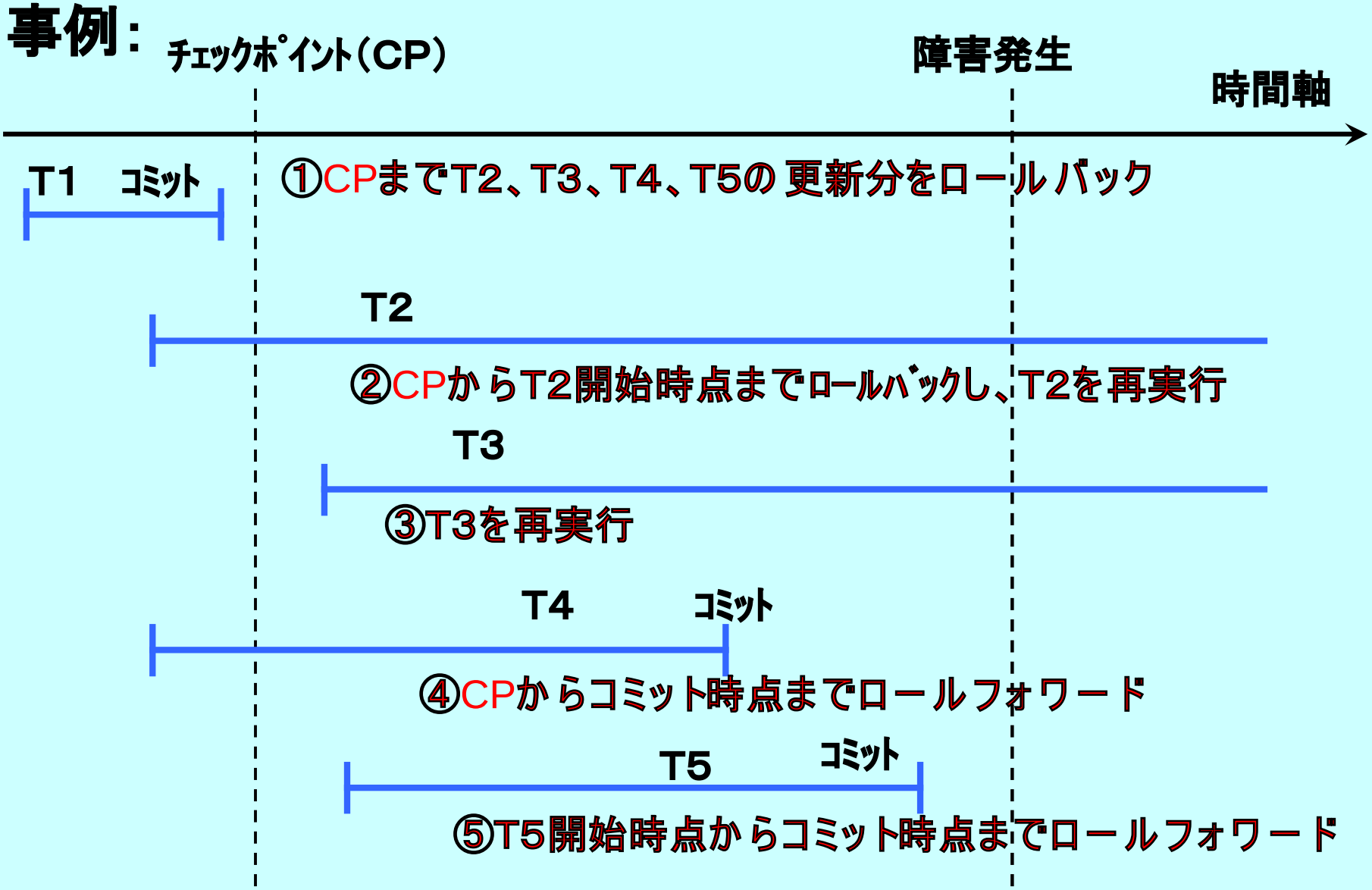
データベース障害の種類

種類	原因	対策
トランザクション障害 （tarnsaction failure）	・プログラム・エラー ・デッドロック・・・	・ロールバック処理
システム障害 （system failure）	・OSダウン ・装置障害・・・	・リカバリー処理（復旧） （チェックポイント使用）
媒体障害 （media failure）	・ディスクトラブル	・バックアップファイルからのリストア（復元） ・リカバリー処理（復旧）

障害回復の準備

- 障害発生時の状況
  ーキャッシュ領域の更新記録は消滅
  ーコミット後では、既にデータベース反映
  　済であり、簡単には元に戻せない
  ー媒体障害時は、データベースが消滅
- 準備項目としては、
  ーDBバックアップ取得
  ーログファイル取得
  ーチェックポイントの設定

システム障害時の回復方法



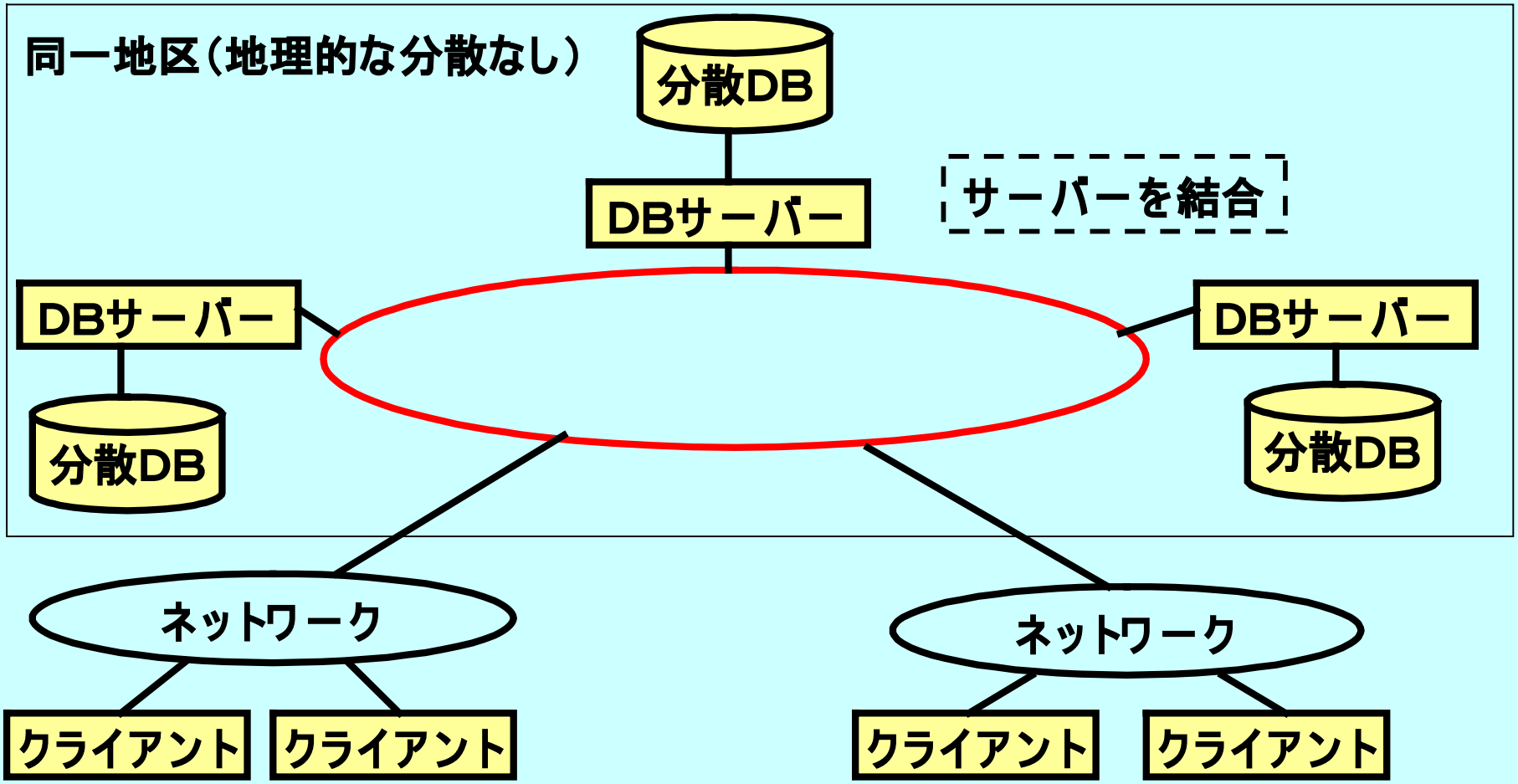
分散DBの 透過性

- 位置の 透過性
- 分割の 透過性
- 重複の 透過性
- 障害の 透過性
- DBMS種別の 透過性

いずれもDBの 利用者に、分散DBの 物理的特性を意識させないで利用できるようにする機能である。

管理の集中、処理の分散

・DB管理を容易にして、分散DBのメリットを実現するための現実解として、考えられたもの。



２相コミットメント制御

・複数の分散データベースの更新では、サイト間通信に時間を要し、各DBのコミット処理に時間差が生まれ、全てのコミット処理が終了する前に障害が発生する可能性がある。

・そこで、分散データベース間の整合性を保つためには、特別のコミットメント制御が必要となる。

２相コミットメント制御

・2相でコミットまたはロールバックを行う方式

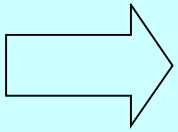
・1相目をコミット準備ﾌｪｰｽﾞとする（セキュア）

・2相目で、全て準備OKならば、コミットする。

オブジェクト指向の重要性

①ソフトウェア開発の危機

・高度情報社会の実現にあたり、
複雑化するシステムの開発が限界に達する
（開発工数、開発効率、開発期間）

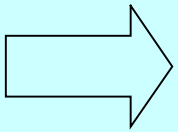


オブジェクト指向の設計による
ソフト部品化、再利用
（オブジェクト指向設計）

（続き）オブジェクト指向の 重要性

②処理対象データの 拡大

- **従来のビジネス領域では、リレーショナルDBで満足**
  - **コード化されたデータ**
  - **短いテキスト**
- **ハードウェアの 大容量化、高速化、品質向上**
  - **CPUの GHz、DASDのテラバイト化**
    NWの GHz化
  - **CADデータ、文書、絵、写真、動画、音声、音楽**
    などの 、即物的なマルチメディア型データが
    コンピュータで扱えるようになった。



ビジネス領域とは 別の 、各種のデータと操作を
統合して扱えるータベースが 必要とされる
（オブジェクト指向データベース）

オブジェクト指向の特徴

発想の原点

- 現実の世界は、いろいろなモノが役割を分担しながら機能を果たしている。
- 自分でできることは、自分でやり、自分の範囲外の仕事は人に任せて、結果だけをもらう

それをシステム作りの発想にしたものである。

「現実のモノ(オブジェクト)およびモノ同士の関係を、
そのままソフトウェアで表現することによって、
現実世界の仕組みを、
コンピュータ上で再現するもの」

OO－DBMSの定義

- オブジェクト指向の概念と
- データベース管理機能を持った

データベース管理システムである。

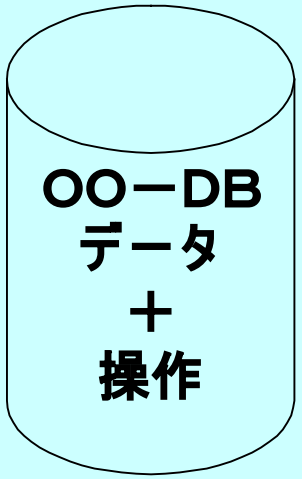
オブジェクト指向概念

- データと操作のカプセル化
  （データ抽象化）
- 継承の概念
- オブジェクト識別性

＋

データベース管理機能

- データ定義、データ操作言語
- 永続性
- トランザクション管理
- ACID特性管理

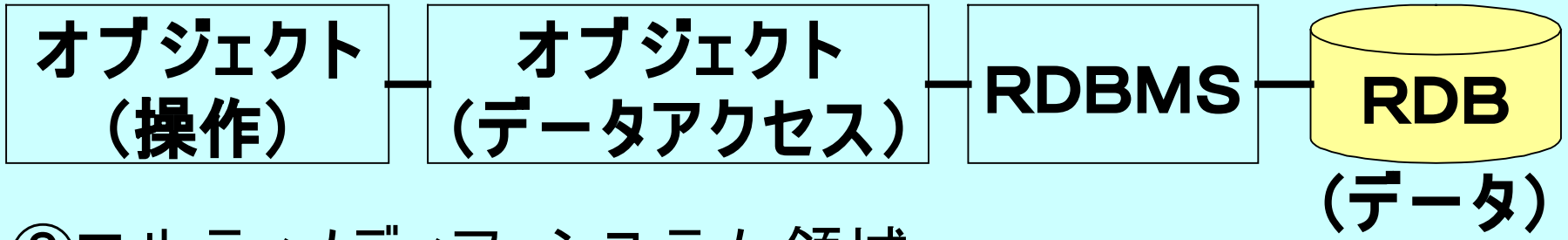


RDBとOODBの棲み分け

①ビジネスシステム領域

・ソフト部品化はオブジェクト指向設計が有利

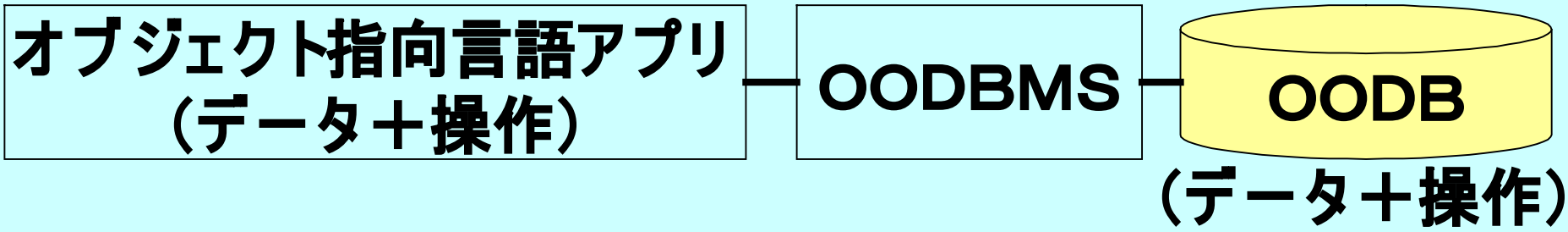
・データベースは、重複排除・活用上、RDBが有利



②マルティメディア・システム領域

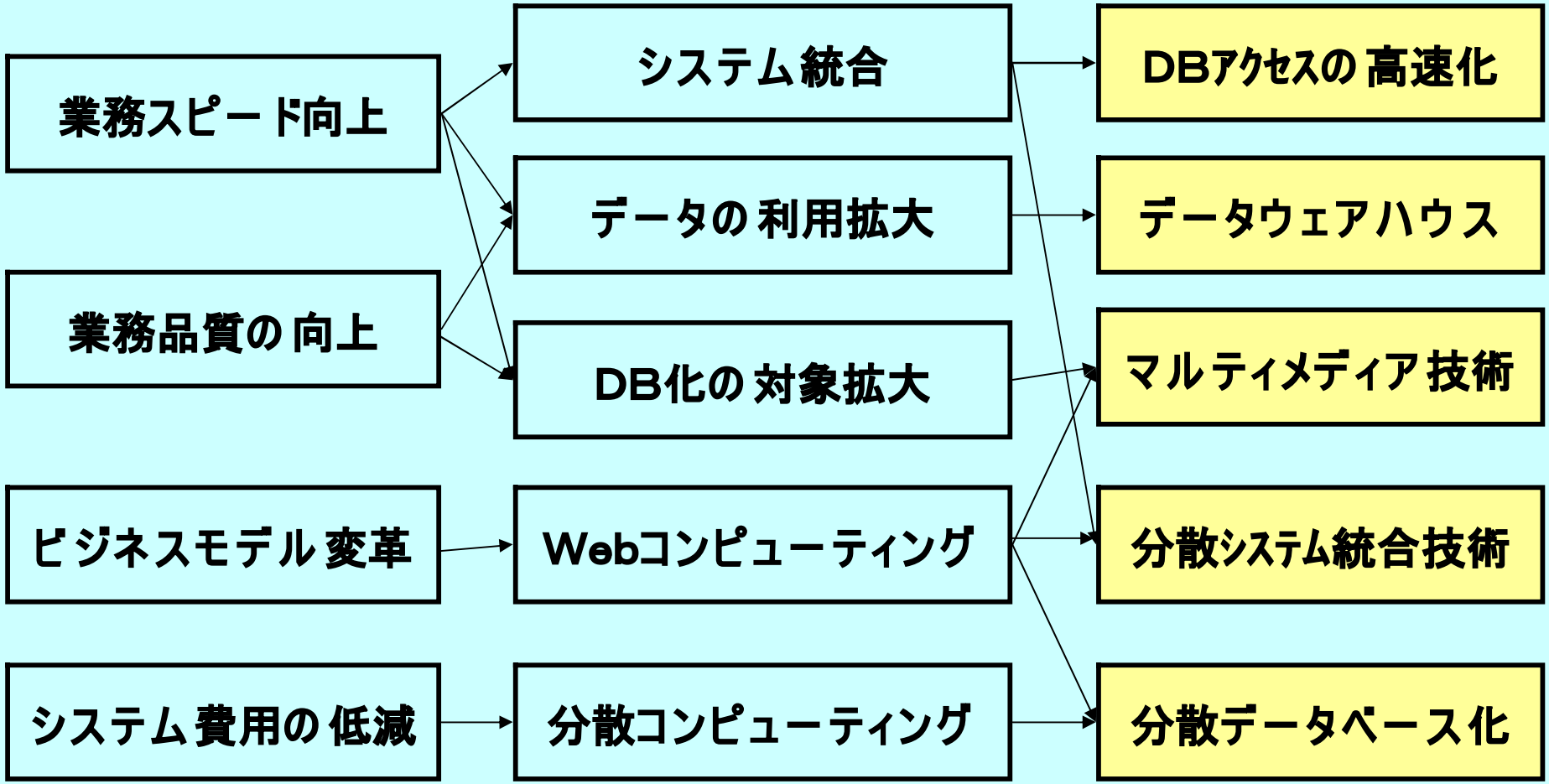
・即物的なシステムであり、オブジェクト設計が有利

・データベースも、OODBが有利



データベース技術動向

企業がデータベースに期待するもの



具体的なDB技術

1．DB処理の高速化

・データベースバッファー
・パーティショニング（分割して格納）
・ディスク・キャッシュ
・ディスク・アレイ（RAID）
・データベースサーバー
・DB並列処理、並列マシン

2．分散システム技術

・マルチスレッド・サーバープロセス
・分散OLTP
・WebサーバーとDBサーバーの連携
・分散システム統合技術（CORBA）

SEに必要な要件

①SEの心構え

- 好奇心→もっと知る→対象業務を好きになる
- 忍耐力、持久力
- 柔軟な考え方ができる
- スタートしなければゴール出来ない

②仕事の進め方

- 何事も計画を立ててから進める
- 書き出してみる（リスト、関連図）
- 自分の考えを話し、人からアイデアをもらう
- 5W3H（What, Why, Who、When、Where, How to, How much, How many）

③人材トレンド

- プロジェクト・リーダー経験者
- 業務系システムの構築・運用経験者
- データベースのスキルを持つエンジニア

“力のある”エンジニアへの需要の伸びは顕著

SEは、DB理論だけでなく、
SQLを実践してみることで、
はじめて身に付けることができる