

データベーススペシャリスト試験は、旧試験制度の平成6年春から実施されているデータベーススペシャリスト試験、平成13年春からのテクニカルエンジニア（データベース）試験を経て、平成21年春から実施されている情報処理技術者試験の一つです。試験制度のスキルレベル4に相当し、高度情報処理技術者試験に含まれます。試験対象者は、システムエンジニアの中でも特にデータベース設計者とSQLプログラマ、データベース管理者です。

午前Ⅰの共通知識30問は応用情報技術者試験と同じ広い範囲から出題されます。午前Ⅱの専門知識25問はそれぞれ範囲を絞った分野からの出題です。午後Ⅰ試験はデータベース設計、同時実行制御、索引設計、運用管理などの分野から3問出題され、得意分野から2問選択します。午後Ⅱ試験はデータベース（物理）設計と概念データモデリングの2問が出題され、同じく得意分野から1問選択します。データモデルの作成、データベースの設計・運用など、実務経験に基づく専門知識、応用能力、実務能力を問う問題です。

旧試験制度のデータベーススペシャリスト試験開始から四半世紀が経過し、数年前から試験内容が少し変化しているようです。従来は概念設計や論理設計といったシステム開発の上流工程に重きを置いていましたが、データベースの実装、SQLなどの中流から下流工程に比重を少し移しているように見受けられます。

専門的、実務的な問題が出題されるとはいえ、問題を解く上で、その前提となっているデータベース技術に関する基礎的な知識は欠かせません。そこで、本書では、こうした基礎知識を修得しようとする方々のために、共通キャリア・スキルフレームワークに準拠し、データベースの広範で基礎的な技術項目について詳しく解説しました。また、基礎知識の理解度を試す演習問題や応用能力アップを図るための例題も収録しました。

基本情報技術者試験、応用情報技術者試験などの基礎技術を習得された方々は、是非、本書を利用して、データベーススペシャリスト試験の前提となる基礎知識・応用能力を身に付けていただければと思います。また、本書をデータベースの企業内教育用のテキストとして用いることもできます。

本書を用いてデータベーススペシャリスト試験の午前や午後問題に対応できる基礎知識・応用能力を身に付けたならば、次に、弊社発行の「徹底解説 データベーススペシャリスト本試験問題」、「データベーススペシャリスト専門知識+午後問題の重点対策」、「極選分析 データベーススペシャリスト予想問題集」などにも挑戦してみてください。本試験問題や予想問題に数多く挑戦すること（アウトプット学習）によって知識の整理ができ、表現力も身に付き、解決能力・実務能力のレベルアップが可能になります。本書と併せてご活用願いたいと思います。なお、本書は主としてデータベーススペシャリスト試験の受験者にデータベース関連の知識を体系的に修得してもらうことに主眼を置いています。一般的なデータベースの教科書としても十分に使えるものだと思います。

データベーススペシャリスト試験は情報処理技術者試験の中でも難関です。皆さまが、努力の結果、合格の栄冠を勝ち取られますよう心からお祈り申し上げます。

2018年9月

アイテック IT人材教育研究部

第 1 章 データベースとデータモデル	7
1.1 データベースとは	8
1.2 データモデルの考え方	12
1.3 概念データモデル	16
1.3.1 データモデル機能の機能要件	16
1.3.2 E-R モデル	19
1.3.3 拡張 E-R モデル	28
1.4 論理データモデル	33
1.4.1 階層モデル	33
1.4.2 ネットワークモデル	34
1.5 章末問題	36
第 2 章 関係モデルの理論	41
2.1 関係モデル	42
2.1.1 関係モデルの背景・目標	42
2.1.2 関係モデルの機能	44
2.1.3 関係モデルの要素	46
2.2 関係代数	50
2.3 正規化理論	57
2.3.1 正規化の意義	57
2.3.2 関数従属性	58
2.3.3 第 1 正規形	62
2.3.4 第 2 正規形	65
2.3.5 第 3 正規形と BCNF	67
2.3.6 第 4 正規形	74
2.3.7 第 5 正規形	77
2.4 章末問題	79
第 3 章 DBMS の機能	85
3.1 SQL 言語	86
3.1.1 データベース言語方式	86
3.1.2 SQL の特徴・言語レベル	87
3.1.3 ホスト言語方式	88
3.1.4 データ定義	91
3.1.5 データ操作	105
3.1.6 クライアントサーバ機能と情報スキーマ	140
3.1.7 静的 SQL と動的 SQL	142
3.1.8 ストアドプロシージャとトリガ	144
3.2 整合性管理	148
3.3 セキュリティ管理	154

3.4	トランザクション管理	158
3.5	同時実行制御（排他制御）	168
3.6	障害回復管理	181
3.7	章末問題	186
第 4 章	DBMS の実装技術	199
4.1	ファイル編成	200
4.2	インデックス法	204
4.3	バッファリング技法	214
4.4	問合せ処理の最適化	216
4.5	章末問題	222
第 5 章	分散データベース	225
5.1	分散データベースの概要	226
5.1.1	分散データベースの概念	226
5.1.2	データの分散戦略	231
5.1.3	分散データベースの透過性	235
5.2	分散問合せ処理	239
5.3	分散トランザクション処理	244
5.4	分散データベースリカバリ	249
5.5	レプリケーション	258
5.6	章末問題	263
第 6 章	情報資源管理とデータベース設計	265
6.1	システムカタログ	266
6.2	情報資源辞書システム	267
6.3	情報資源管理	271
6.4	リポジトリ	274
6.5	CASE ツールとリポジトリ	278
6.5.1	CASE 環境についての PCTE (ISO/IEC 13719)	278
6.5.2	CASE ツールとリポジトリ	279
6.5.3	メタデータベース	280
6.7	章末問題	282
第 7 章	データ分析・データモデル作成	283
7.1	データ定義の標準化	284
7.2	データベースの要件定義	288
7.3	データ分析・データモデル作成の概略手順	290
7.4	トップダウンアプローチ	297
7.4.1	エンティティの識別	298

7.4.2	エンティティのキーの識別	299
7.4.3	エンティティ間の関連の識別	301
7.4.4	ビジネスルールの識別	311
7.4.5	エンティティの属性の洗出し	316
7.4.6	データ正規化によるチェック	317
7.4.7	DFD との整合性検証	317
7.4.8	コード分析	318
7.4.9	データモデルとビジネスプロセスの整合性確認	319
7.4.10	データ制約の分析	324
7.5	ボトムアップアプローチ	326
7.5.1	ユーザビューの収集, データ項目の抽出とデータ定義	327
7.5.2	ユーザビューごとのデータ正規化と 部分図の作成	327
7.5.3	複数部分図の類似エンティティの統合	330
7.5.4	ユーザビューの設計	332
7.6	論理データモデルの作成	335
7.6.1	論理データモデルへの変換	335
7.6.2	テーブル構造	336
7.7	概念データモデル作成例題	350
7.8	論理データモデル作成 (DB 設計) 例題	364

第 8 章 データベースシステム設計と実装 377

8.1	DBMS の選定と導入の概要	378
8.2	スペース見積り	386
8.3	性能見積り	389
8.4	物理データベース設計	395
8.4.1	物理環境の確認とトランザクション分析	395
8.4.2	物理設計のポイントと非正規化	396
8.4.3	インデックスの利用と選定他	404
8.5	性能調整	407
8.5.1	効率向上トレードオフ	407
8.5.2	データの物理的な配置	410
8.5.3	アプリケーション性能調整	412
8.6	章末問題	417

第 9 章 データベースシステムの運用管理 421

9.1	運用管理の指針	422
9.2	性能管理	423
9.3	障害管理	426

9.4	セキュリティ管理	428
9.5	データベース保守管理	431
9.6	章末問題	432
第 10 章	データベースの技術動向	435
10.1	データベース言語	436
10.1.1	SQL/CLI	438
10.1.2	SQL/PSM	438
10.1.3	SQL-99	440
10.2	データウェアハウス	444
10.2.1	データウェアハウスの概要	444
10.2.2	データウェアハウスのアーキテクチャ	447
10.2.3	データウェアハウスの設計と構築	451
10.2.4	オンライン分析処理	454
10.2.5	データマイニング	458
10.3	オブジェクト指向とデータベース	462
10.3.1	オブジェクト指向の基本概念	462
10.3.2	オブジェクト指向システム開発	467
10.3.3	オブジェクト指向システム開発とデータベース	472
10.4	オブジェクトリレーショナルデータベース	476
10.5	Web とデータベース	479
10.6	データ記述言語 XML/JSON とデータベース	484
10.7	ビッグデータと NoSQL	493
10.7.1	ビッグデータの分析処理	493
10.7.2	分散処理フレームワーク	494
10.7.3	ストリーム処理	498
10.7.4	NoSQL	501
10.8	章末問題	507
	章末問題 解答・解説	511
	APPENDIX	541
1	主要な RDBMS の概要	542
2	代表的なデータベース設計ツール (E-R 図エディタ) の概要	543
	用語 INDEX	544
	参考文献	555

1.1 データベースとは

▶ DBMS

▶ データ定義言語

▶ DDL

▶ データ操作言語

▶ DML

▶ 同時実行制御

データベースは、現在のコンピュータシステムでは、特に事務処理向けではなくてはならないものである。データベースとは何かを簡単にいえば、「長期間に渡って必要とする情報のかたまり」であるが、もう少し具体的に定義するなら、「コンピュータシステムにおいて、**DBMS** (DataBase Management System：データベース管理システム) によって管理されるデータの集合体」となる。DBMS は、次のような機能を持っている。

- ・データベースの論理構造を定義する機能：この定義する言語を**データ定義言語 (DDL；Data Definition Language)** という。
- ・データベースの内容を問い合わせたり、更新したりする機能：この言語を**データ操作言語 (DML；Data Manipulation Language)** という。
- ・大量データの格納管理機能
- ・多くのデータベースユーザの同時アクセスにおいて、データに矛盾が生じないようにする機能：この機能を**同時実行制御**機能という。

企業の資産としてのデータ（情報）を効率的に管理する手段がデータベースである。データベースは、企業などの組織体の日々の業務を効率化するための基幹系（業務系）システムのデータを、体系的に蓄積・管理するためにできた。

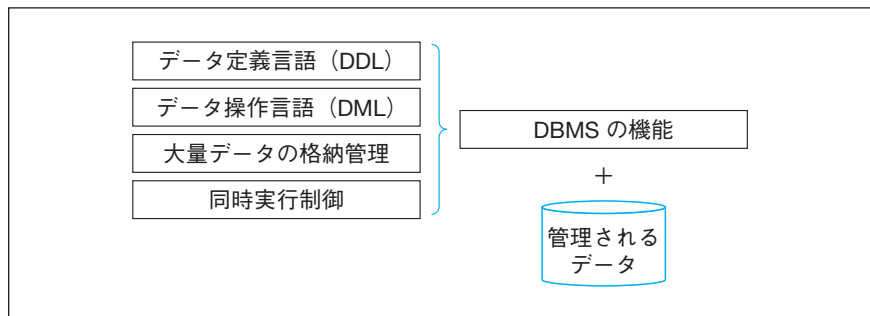


図 1-1 データベースの定義

データベース管理システムは、それまでのファイルシステムを発展させたものである。最初の商用データベース管理システムが出現したのは、1960 年代の後半である。ファイルシステムは、上記四つの機能の一部を備えているが、完全な形ではそれらの機能を備えていなかった。歴史的には、データベースシステムの代表例として列車、航空機の座席予約システム、銀行の勘定系システム（一般には経理会計業務システム、特に銀行の基幹系システム）などがある。

▶データの統合化

(4) データの統合化

ファイルシステムとデータベースの違いの第2の点は、データの統合化が図れることである。ただし、**データの統合化**は、関係データベースを単にファイルシステムの代わりに使用すれば自然と達成されるものではなく、関係データベースをどう使用するかによる。データベースを利用する狙いは、共通データベースを中心とした統合化にある。しかし、現実には関係データベースを単にファイルシステムの代わりに使用しただけの縦割りの情報システム（経理、販売、人事などが独立）があり、本来のデータベースの利点を生かしていないケースもよくあった。これは、プログラムとデータの独立性はある程度高められているが、データはプログラムに依存した形で設計されていて、データ間の統一性、統合化が達成されない場合に起こる。

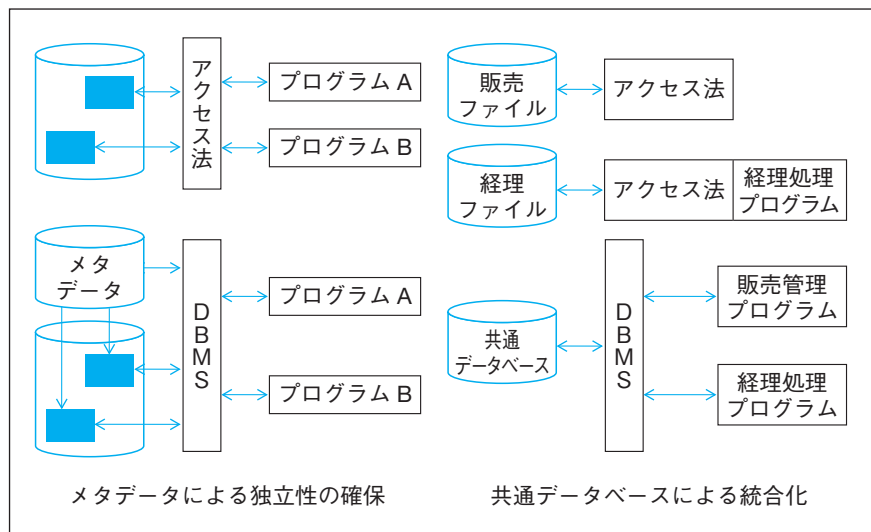


図 1-2 データベースによる独立性の確保と統合化



データベースとファイルシステムの最も異なる点は何でしょうか。

データベースはファイルシステムと比較して、データの整合性を保ち、プログラムからデータを独立させるように管理できます。

1.5 章末問題

問 1-1

■ H21 春-DB 午前Ⅱ 問 1

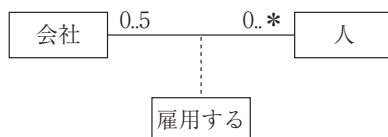
ANSI/SPARC3 層スキーマに関する記述として、**適切でないもの**はどれか。

- ア ANSI/SPARC3 層スキーマの意義は、物理的データ独立性及び論理的データ独立性を確保することである。
- イ 外部スキーマは、概念スキーマが変化しても応用プログラムができるだけ影響を受けないようにするための考え方である。
- ウ 関係データベースのビューやネットワークデータベースのサブスキーマは、概念スキーマに相当する。
- エ 内部スキーマは、概念スキーマをコンピュータ上に実装するための記述である。

問 1-2

■ H24 春-DB 午前Ⅱ 問 4

次の概念データモデルを関係データベース上に実装することとし、実装用のデータモデルを作成した。適切な多重度が指定されているものはどれか。ここで、モデルの表記には UML を用いる。



- ア


```

classDiagram
    class 会社
    class 雇用
    class 人
    会社 "0..5" -- "1" 雇用 : 雇用
    雇用 "1" -- "0..*" 人 : 雇用
  
```
- イ


```

classDiagram
    class 会社
    class 雇用
    class 人
    会社 "0..*" -- "1" 雇用 : 雇用
    雇用 "1" -- "0..5" 人 : 雇用
  
```
- ウ


```

classDiagram
    class 会社
    class 雇用
    class 人
    会社 "1" -- "0..5" 雇用 : 雇用
    雇用 "0..*" -- "1" 人 : 雇用
  
```
- エ


```

classDiagram
    class 会社
    class 雇用
    class 人
    会社 "1" -- "0..*" 雇用 : 雇用
    雇用 "0..5" -- "1" 人 : 雇用
  
```

3.4 ■ トランザクション管理

トランザクションが ACID 特性を保持しつつ同時実行できるようにする機能が、トランザクション管理機能である。トランザクション管理機能には、同時実行制御（排他制御）機能、障害回復機能、コミット制御機能（5章の分散データベース参照）がある。

(1) トランザクションの ACID（アシッド）特性

▶ トランザクション

トランザクション（transaction）とは、データベースの更新などを含む一つの作業単位で、コミット又はロールバックをもって完結する。集中型データベースや分散データベースシステムにおいて、トランザクションは原子性、一貫性、隔離性、耐久性といった **ACID 特性**を持たなければならない。ACID 特性を持つことでデータベースの一貫性が保たれる（表 3-4 参照）。

▶ ACID 特性

表 3-5 ACID 特性

特性	内容
▶ 原子性	原子性 (Atomicity) トランザクションが終了したとき、全ての処理が完了しているか、全く行われていないかどちらかの状態であること。 対応機能 障害回復機能、2 相 / 3 相コミット制御
▶ 一貫性	一貫性 (Consistency) トランザクション処理の終了状態に関わらず、データベースは一貫性を保っていること。 対応機能 同時実行制御、排他制御
▶ 隔離性	隔離性 (Isolation) 複数のトランザクションを同時に実行したときに、直列可能であること。すなわち直列に実行したのと同じ結果であること。更新途中のトランザクションの処理結果が他のトランザクションに見えないこと。 対応機能 同時実行制御、排他制御
▶ 耐久性	耐久性 (Durability) トランザクションが完了すると、その後の障害などによってデータベースの内容が変化しないこと。 対応機能 障害回復機能、2 相 / 3 相コミット制御

(2) 並列実行時の問題点

並列実行される複数のトランザクション T_1 、 T_2 が同時実行制御を行わない環境下で同時にデータベースの更新を行うと、次のような問題が生じる。

① ロストアップデート（lost update ; 更新の喪失）

同時実行制御を行わずに T_1 、 T_2 を実行すると、 T_1 か T_2 のどちらかの更

- ▶ ロストアップデート
- ▶ 更新の喪失

新結果が上書きされ、正しい結果が得られなくなる。更新結果が失われるのをこれを**ロストアップデート**（**更新の喪失**）という（図 3-61 参照）。

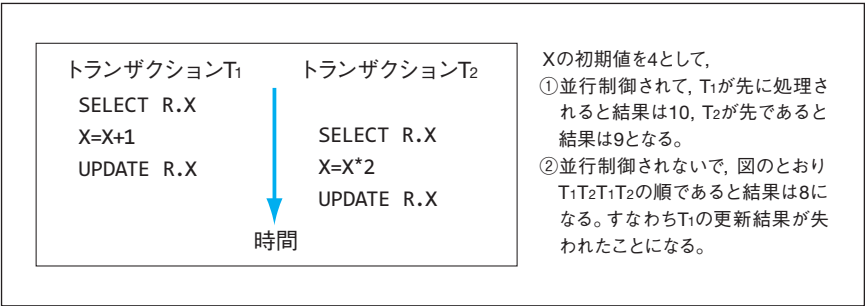
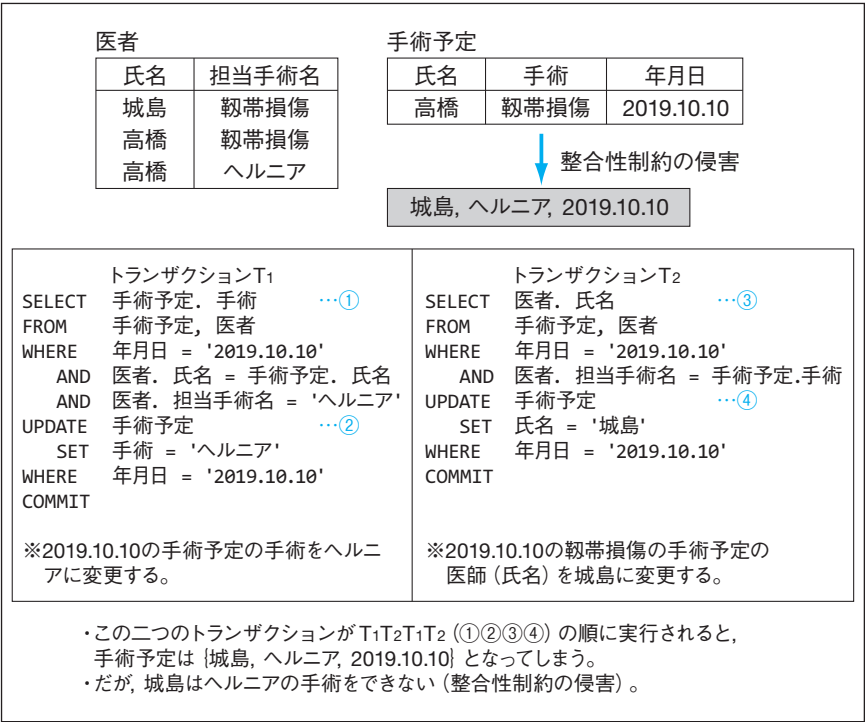


図 3-61 ロストアップデートの例

- ▶ 整合性制約の侵害

- ② 整合性制約の侵害
- 同時実行制御を行わずに T₁、T₂ を並列実行すると、整合性（インテグリティ）制約が侵害される場合がある。これを**整合性制約の侵害**という（図 3-62 参照）。





隔離性水準（アイソレーションレベル）を規定する三つの現象は何でしょう。さらにどの隔離性水準においても発生してはいけない現象は何でしょう。

- ・ 三つの現象…①ダーティリード（Dirty read）、②ノンリピータブルリード（Non-repeatable read）、③ファントム（Phantom）
- ・ 発生してはいけない現象…ロストアップデート（更新の喪失）



2相ロック方式となっている隔離性水準はどれでしょう。

READ UNCOMMITTED…共有ロックを掛けない、長期の専有ロック。

READ COMMITTED…短期の共有ロック、長期の専有ロック。

REPEATABLE READ…長期の共有ロック、長期の専有ロック。

SERIALIZABLE…長期の共有ロック（検索範囲にロック）、長期の専有ロック。

であるから、READ UNCOMMITTED と READ COMMITTED はコミットまで共有ロックを保持していないので、共有ロックに関して2相ロック方式となりません。したがって、2相ロック方式となっているのは REPEATABLE READ と SERIALIZABLE です。多くの DBMS が性能上の観点から隔離性水準の既定値が、READ COMMITTED となっているので注意が必要です。



隔離性水準（アイソレーションレベル）は、他のトランザクションの隔離性水準の影響を受けますか。

隔離性水準の既定値は、標準 SQL では SERIALIZABLE であり、現実の RDBMS では並列実行性能を考慮して READ COMMITTED が多いのですが、個々のトランザクションの隔離性水準は、SET TRANSACTION 文で設定できます。隔離性水準は、トランザクション間の相互の影響度合いを表すので、確かに他のトランザクションの更新操作の影響を受けますが、トランザクションごとに決まり、他のトランザクションの隔離性水準の影響を受けません。例えば、READ UNCOMMITTED にすると、ダーティリードが発生する可能性がありますが、これは他のトランザクションが同一データを更新中にアボートしたために発生するのであって、他のトランザクションの隔離性水準と無関係です。

10.2 ■ データウェアハウス

▶データウェアハウス : **データウェアハウス** (data warehouse) は、サブジェクト指向、統合化、恒常的、時系列という特性を持った意思決定支援のためのデータの集合といわれている。

10.2.1 データウェアハウスの概要

データウェアハウスは、従来型のシステムの問題点や情報資源の有効活用という経営課題を解決する新しい情報系システムである。

(1) データウェアハウスの意義

データウェアハウスは、次に挙げるような問題点や課題の解決策である。

① 現行システムの限界とデータウェアハウス

従来の基幹システムは業務ごとに個々別々に開発されたため、全社的なデータの整合性が図られておらず、企業を横断する体系化されたデータがない。ユーザニーズにあった情報を取り出そうとすると、次のような問題が生じる。

(a) データの信頼性の欠如

取り出した情報の結果がデータソースによって異なり、データの信頼性が低い。次のような原因が考えられる。

- ・ どの時点のデータなのかアプリケーションによってデータの同期がとられていない
- ・ アプリケーションによってデータ抽出のアルゴリズムが異なる
- ・ 出所を明示しないで外部データを取り込んでしまっている
- ・ 基礎になる共通のデータソースが存在しない、など

(b) プログラムの開発生産性が低い

異音同義語や同音異義語などを含んだデータソースから、必要なデータを見つけ、プログラムを作成するには時間がかかる。

(c) データから情報への変換の不具合

既存システムのデータの名前や桁などに統一性がないため、既存データから要求に見合った情報へ変換する場合に不具合が生じる。また、OLTP（オンライントランザクション処理）では正規化されたデータに対する更新が中心で短いサイクルのデータしか持たず、必要な履歴データを保持していないことも原因の一つである。

② 競争優位な情報資源

10.3.3 オブジェクト指向システム開発とデータベース

▶ 構造化設計

▶ データ中心設計

▶ オブジェクト指向設計

(1) オブジェクト指向分析設計とデータベース設計

① データ中心設計とオブジェクト指向設計

構造化設計は機能を中心に設計するため拡張性に乏しく、データベースの設計が弱いという欠点がある。**データ中心設計**はこの点を補うためにデータモデリングを優先するが、動的モデリングの機能が弱い。**オブジェクト指向設計**はこれら両方の設計手法を包含することで、両手法の問題点を解決する。

表 10-4 構造化設計／データ中心設計／オブジェクト指向設計の比較表

構造化設計	データ中心設計	オブジェクト指向設計
手続(機能)中心にシステムを構成。	データモデル作りを優先してシステムを構成。	概念オブジェクト中心にシステムを構成(オブジェクトと関連を現実世界と同じようにとらえる)
モデル化 ①オブジェクトモデル(E-R図) ②動的モデル(状態遷移図) ③機能モデル(データフロー図) このうち重点は③であり、結果をプロセス仕様書へまとめる。	モデル化 ①オブジェクトモデル(E-R図) ②動的モデル ③機能モデル このうち重点を①におく。 ②③は構造設計に依存する。	モデル化 ①オブジェクトモデル(クラス図) ②動的モデル ③機能モデル このうち重点を①におく。
特徴 ①変更要求は、機能に対する変更としてとらえる。その都度、機能に対する修正がでる。	特徴 ①変更要求をデータに基づく要求定義に置き換えてとらえる。データ構造は安定している。	特徴 ①変更要求があってもオブジェクト構造は変わらず、操作の追加や変更によって対応できる。
②明確に定義されたシステム境界があり、プログラムを通して実世界と相互作用するため、拡張性に乏しい。	②動的モデリングの機能がなく、この部分は構造設計に依存。	②外部世界に存在するオブジェクトや関係を単に追加すればよい(弾力的で拡張性が高い)
③プロセスのサブプロセスへの分割は任意性が含まれる。	③データモデルは理解しやすいが、操作までは入らない。	③問題領域内のオブジェクトとオブジェクト指向設計内のオブジェクトの直接的な類似関係によって、システムは非常に理解しやすくなる(要求とプログラムコードとの対応を追跡しやすくする)
④DBの扱いが本質的に不得手	④DBと親言語とのインピーダンスミスマッチ	④DBとプログラムの統合

第1章 データベースとデータモデル

問 1-1 ウ

ANSI/SPARC3 層スキーマ ■ H21 春 -DB 午前Ⅱ問 1

ANSI/SPARC3 層スキーマの外部スキーマは、利用者やアプリケーションから、直接、見える世界であり、関係データベースのビューやネットワーク型データベースのサブスキーマに相当する。(ウ)の「概念スキーマ」は「外部スキーマ」の誤りである。したがって、(ウ)が適切でない記述である。

ア：内部スキーマによって物理的データ独立性は保たれ、外部スキーマによって論理的データ独立性が保たれる。適切な記述である。

イ：外部スキーマは概念スキーマを外部に見せるためのインタフェースであり、概念スキーマが変化しても外部スキーマを恒常的に保つことが可能である。適切な記述である。

エ：内部スキーマはデータの格納構造を記述するので、適切な記述である。

問 1-2 エ

実装用データモデルの適切な多重度 ■ H24 春 -DB 午前Ⅱ問 4

この概念データモデルは UML のクラス図の多重度表記で表現されている。クラス図の多重度表記は、「最小数..最大数」の形で記述するが、最小数が 0 の場合は、その関連に参加していないインスタンスがあることを示す。例えば、会社と人の関連では互いに 0.5, 0..* で 0 が最小数であるが、会社から見ると全員とは雇用関係がない、人から見ると全部の会社とは雇用関係がないということを意味する。E-R 図の用語では、最小数に相当するものをパーティシペーション (participation)、最大数に相当するものをカーディナリティ (cardinality) などという。

問題で挙げられている概念データモデルは、①人は会社に雇用されていないか、最大五つの会社に雇用されている、もしくは雇用されたことがある (0.5 に対応) ことと、②会社は人を 1 人も雇用していないか、何人も雇用しており、上限がない (0..* に対応) ことを示している。選択肢の雇用は、E-R 図で、多対多の関連を 1 対多と多対 1 の二つの関連に分解する連関エンティティに相当する。会社から見て雇用は 0..* の多重度があるから、雇用の側に 0..* がある (エ) が正解である。なお、雇用から見ると会社と人はそれぞれ必ず一つ存在するので多重度は 1 である。UML のクラス図の表記では、意味的には 1..1 であるが、1 と記述する。

問 1-3 エ

E-R 図 ■ H21 秋 -AP 問 30

関連は、二つのエンティティ間の対応関係を表したもので、1 対 1, 1 対多, 多対多の 3 種類の関係のいずれかとなる。選択肢の内容から、E-R 図に表現されたエンティティは、在庫、仕入先、部品の三つであることが分かるので、部品在庫管理台帳から、各エンティティ間の関連を判断すればよい。1 番目の行と 2 番目の行は部品コードが同じであるので、部品と在庫の関係は 1 対多であることが分かる。次に、2 番目と 3 番目の行は仕入先が同じであるので、仕入先と在庫の関係は 1 対多であることが分かる。これらの条件を満たしているのは (エ) である。

ア：在庫と仕入先の関係が 1 対多になっているので、逆である。

イ、ウ：在庫と部品の関係が 1 対多になっているので、逆である。

1 主要な RDBMS の概要

製品名 (バージョン名は省略)	ベンダ	特徴・機能
	動作環境	
Oracle	オラクル	世界初の商用 RDBMS である。マルチバージョンデータベース方式を採用し、UNDO セグメント（ロールバックセグメント）を参照することで、専有ロックを待たずにデータを読める「読取り一貫性」機能を持つ。関連情報は、Web や専門書などから容易に取得できる。
	UNIX, Linux, Windows など	
Db2 (2017 年 DB2 から 名称変更)	IBM	Oracle と並び、歴史と実績を持つ。Oracle などと比較すると Web や専門書などによる関連情報が少ない。Db2 (DB2) と総称しているが、メインフレーム用、AS/400 用、UNIX/Windows 用は別製品である。
	UNIX, Linux, Windows, z/OS など	
SQL Server	マイクロソフト	サイベースの SQL Server が元になっていたが、バージョン 7 から独自路線となった。新しい DBMS である。.NET 環境を構築する場合に適している。動作環境は、Windows に限定されていたが、SQL Server 2017 から Linux 上でも動作可能となった。
	Windows, Linux	
PostgreSQL	オープンソース	世界初の ORDB である Postgres の問合せ言語を SQL 化したものである。当然であるが、ORDB の機能が充実している。オラクルと同じようなマルチバージョンデータベース方式を採用している。日本語化などに早くから対応しており、オープンソース DBMS として広く使われている。
	UNIX, Linux, Windows など	
MySQL	オープンソース	スウェーデンの MySQL AB 社が開発していたオープンソース DBMS である。現在は、サンマイクロシステムズ経由でオラクルが所有。機能が少ないが軽快で、小規模 Web サイトのデータベースとして広く使われている。当初は、コミット／ロールバックや副問合せなどの機能を持っていなかったが、4.0/4.1 からそれらの機能もサポートされている。
	UNIX, Linux, Windows など	
SQLite	オープンソース	アプリケーションに組み込んで利用される軽量データベースである。データベースは単一のファイルとして保存される。SQL-92 の機能を実装しているが、簡易化されている機能もある。例えば、LEFT OUTER JOIN はあるが、RIGHT OUTER JOIN はない。著作権を放棄したパブリックドメインソフトである。Android 端末の標準ライブラリでもある。
	UNIX, Linux, Windows, Android OS など	