

平成 30 年度春期 応用情報技術者試験 午後問 6「備品購買システムの設計と実装」初心者向け完全復習ガイド

1. 本問題の全体像と出題趣旨の理解

1.1 出題趣旨とデータベース設計の核心

応用情報技術者試験の午後問題において、データベース分野（問 6）は単なる用語や構文の記憶力を試す場ではありません。IPA が公表した出題趣旨によると、本問題には次のような意図が明確に込められています。

出題趣旨 データモデルの作成は、現在の対象業務を単に表現するだけではなく、将来の変化に対応するための柔軟性まで考える必要がある。本問では、備品購買システムを題材に、E-R 図や SQL 文に関する基本的な理解、機能追加や仕様変更が発生した際のデータモデルへの対応に関する理解、能力を問う。

データベース設計における核心は、**「現在の対象業務を正確にモデル化すること」と「将来の仕様変更や機能追加に耐えうる柔軟性を確保すること」**の 2 つの視点を両立させる点にあります。本問題（R 社における表計算ソフトから Web ベースの備品購買システムへの移行）は、業務要件の整理から始まり、初期データモデルの作成、相見積り機能の追加に伴うモデル改修、SQL での集計処理、さらには返品というイレギュラー運用への対応まで、実務におけるデータベースエンジニアリングのライフサイクル全体を凝縮した良問です。

本ガイドでは、単に解答の正誤を確認するだけでなく、業務フローをいかにして論理的なデータベース構造へと落とし込んでいくのか、その思考プロセスをステップバイステップで紐解いていきます。

次節では、すべての設計の起点となる「業務フロー」と「初期データベース構造（E-R 図）」の解説に進みます。

1.2 業務フローと初期データベース構造（E-R 図）の解説

データベース設計の第一歩は、業務上の処理手順とそこで取り扱われるデータの関係性を正しく分析することです。業務フローを理解せずに正しいテーブル定義や**リレーションシップ（関連）**を構築することはできません。

問題文に提示された表 1「備品購買の処理の流れとシステム化対象」の No.1～8 の処理内容と、初期 E-R 図におけるエンティティ（実体）の対応関係を以下のように整理します。

表 1：備品購買の処理の流れとシステム化対象の対応整理

No.	処理名	概要	システム化対象	対応する初期 E-R 図のエンティティ
1	購買依頼	利用部門が総務部に、備品の調達を依頼する。	なし	—
2	<u>見積依頼</u>	総務部が依頼された調達の内容を確認し、在庫がない場合は取引先に見積りを依頼する。取引先のカatalogを使って商品番号、商品名、数量及び希望回答日を入力する。	あり	見積依頼、見積依頼明細
3	<u>見積登録</u>	総務部が取引先から回答された見積りの内容を登録する。見積有効期限や価格情報などを入力する。	あり	見積、見積明細
4	<u>発注</u>	総務部が取引先に、見積りに対して発注する。発注する商品の数量や希望納品日を入力する。	あり	発注、発注明細
5	<u>納品</u>	総務部が取引先から届いた商品を登録する。届いた商品の数量などを入力する。取引先の在庫状況によっては、複数回に分けて商品が届くことがある。	あり	納品、納品明細
6	<u>検収</u>	総務部が発注した商品が全て納品されたことを確認する。	あり	検収
7	請求	総務部が取引先から送られてきた請求書の情報を登録する。	なし	—
8	支払依頼	総務部が経理部に、請求書に対する支払を依頼する。	なし	—

※注記：処理名に下線が引かれているもの（No.2～6）がシステム化対象です。

業務フロー把握が E-R 図解説の第一歩となる理由

業務フローを正確に把握することが不可欠である理由は、**「どのようなデータが、どのタイミングで、どのような単位（1 件単位か、複数明細単位か）で発生するか」**というビジネスルールがそのまま E-R 図のエンティティや属性、**カーディナリティ（対応数）**の根拠になるからです。例えば、「取引先の在庫状況によっては、複数回に分けて商品が届くことがある」という業務ルールは、発注と納品の間の関係（1 対多）を決定づける直接の根拠となります。

この業務フローの理解をもとに、次の章では設問 1 の初期 E-R 図の補全作業（属性とカーディナリティの特定）を解説します。

2. 設問 1：初期 E-R 図の補全（属性決定とリレーションシップ）

2.1 属性の導出（空欄 a：希望回答日）

初期 E-R 図の「見積依頼」エンティティにある空欄 [a] に入る属性名を特定します。

導出のロジック

表 1 の No.2「見積依頼」の概要には、「取引先のカタログを使って依頼された備品の商品番号と商品名を調べ、数量及び希望回答日と一緒に記入する」と記載されています。

ここで、図 1 の初期 E-R 図における「見積依頼」と「見積依頼明細」の属性構造を比較してみましょう。

- **見積依頼明細**の属性：見積依頼番号（主キー）、見積依頼明細番号（主キー）、商品番号、商品名、商品数量
- **見積依頼（ヘッダー）**の属性：見積依頼番号（主キー）、取引先番号、取引先担当者、見積依頼日、見積依頼者、[a]

「商品番号」「商品名」「商品数量」といった個々の備品ごとの情報は「見積依頼明細」に含まれています。一方で、「希望回答日」は個別の商品単位ではなく、見積依頼手続き（ヘッダー）全体に対して 1 つ設定される日付情報です。したがって、見積依頼明細ではなく「見積依頼」エンティティの属性として定義されるのが論理的です。

- **解答：[a] 希望回答日**

💡 **講師の試験対策ワンポイント** データベース設計の基本ルールとして、**「1 回の伝票（ヘッダー）で共通する値（依頼日、回答希望日、依頼者など）」はヘッダーテーブルに配置し、「繰り返し発生する要素（行ごとの商品や数量）」**は明細テーブルに配置します。問題文の記述がどちらのレベルに属しているかを判定することが見極めのコツです。

2.2 エンティティ間のカーディナリティ特定（空欄 b, c）

E-R 図におけるエンティティ間の**リレーションシップ（関係性）**を特定するにあたり、業務ルールから「1 対 1」なのか「1 対多」なのかを判別します。

カーディナリティ特定のための比較分析

比較項目	発注 と 検収 (空欄 b)	発注 と 納品 (空欄 c)
対象エンティティ対	発注 — 検収	発注 — 納品
業務ルールの根拠	表 1 No.6「総務部が発注した商品が全て納品されたことを確認する」 → 1 つの発注に対して、全品納品完了時に 1 回の検収が行われる。	表 1 No.5「取引先の在庫状況によっては、複数回に分けて商品が届くことがある」 → 1 つの発注に対して、商品が複数回に分かれて納品される (分割納品)。
カーディナリティ	1 対 1	1 対多
記号	—	→

- 解答:[b]— (1 対 1 の線)
- 解答:[c]→ (1 対多の矢印)

実務アーキテクトの視点 カーディナリティの違いがもたらすシステムインパクト (So What?) 1

1 対 1 (発注と検収) : 発注に対して検収レコードは高々 1 件しか存在しません。そのため、主キー同士 (または発注番号) で単純結合 (INNER JOIN) してもデータ行数が増加せず、重複集計のリスクがありません。 **1 対 1 対多 (発注と納品) :** 1 つの発注に対して納品レコードが複数存在する (分割納品) ため、そのまま結合すると発注側のデータが納品回数分だけ重複して出力されます。したがって、後続の集計処理 (設問 3) において **GROUP BY** や **SUM** を用いて商品ごとに集計する設計が必須となります。

初期モデルの構造とカーディナリティを正しく把握したところで、次は相見積り機能の追加に伴うデータモデルの改修 (設問 2) へと進みます。

3. 設問 2 : 機能拡張への対応 (相見積り機能の実装とモデル改修)

3.1 1 対多構造への分解と主キーの設計 (空欄 d, e)

実務では、運用の変更や機能追加に伴うデータベース構造の変更が頻繁に発生します。本問では「相見積り機能 (複数の取引先へ同じ内容の見積依頼を出せる機能)」の追加が提示されています。

1 対多構造への分解ロジック

初期モデルでは、1つの「見積依頼」に対して1つの「取引先番号」が直接紐付いていました（1対1の想定）。しかし、相見積りでは「1つの見積依頼内容に対して、複数の取引先へ依頼を送信する」という1対多の関係が発生します。

同じ見積依頼内容（依頼日、依頼者、希望回答日など）を重複して持たせることなく、複数の取引先への依頼を管理するために、見積依頼エンティティを以下の2つに分離（正規化）します。

構造変更前後のデータモデル比較（スキーマ表現）

- **【変更前の構造】**
 - 見積依頼（<u>見積依頼番号</u>, <s>取引先番号</s>, 取引先担当者, 見積依頼日, 見積依頼者, 希望回答日）
- **【変更・分割後の構造】**
 - 1つ目のエンティティ（全体を束ねるヘッダー）：見積依頼（<u>見積依頼番号</u>, 見積依頼日, 見積依頼者, 希望回答日）
 - 2つ目のエンティティ（各取引先への依頼を管理する子エンティティ）：見積依頼取引先（<u>見積依頼番号</u>, <u>取引先番号</u>, 取引先担当者）

複合主キーが必要となる理由

2つ目のエンティティは、「どの見積依頼（見積依頼番号）」を「どの取引先（取引先番号）」に対して行ったかを特定する役割を持ちます。

単一の 見積依頼番号 だけでは複数の取引先を識別できず、逆に 取引先番号 だけではどの見積依頼かを特定できません。そのため、「どの見積依頼をどの取引先に出したか」を一意に識別するには、**見積依頼番号 と 取引先番号 の2つを組み合わせた複合主キー**が不可欠となります。

- 解答：[d] 見積依頼番号 （※d, e は順不同）
- 解答：[e] 取引先番号 （※d, e は順不同）

 **実務アーキテクトの視点** 実務において「1対1」から「1対多」へ要件が変化した場合、既存のテーブルに無理やり複数の取引先カラム（取引先 1, 取引先 2...）を追加するのはアンチパターンです。本問のように「ヘッダーエンティティ」と「関連（割り当て）エンティティ」に分解し、**複合主キー**を定義することこそが、将来の拡張に耐えうる柔軟なデータモデル構築の鉄則です。

3.2 関連エンティティへの影響と外部キーの設定（空欄 f）

親エンティティの構造変更は、それと参照関係にある子エンティティ（配下の明細テーブル）へ波及します。

見積依頼明細エンティティへの構造的影響

相見積り機能が導入されると、見積依頼明細（商品ごとの依頼内容）が「どの取引先に対する見積依頼の明細なのか」を特定する必要が生じます。

- **【変更前】見積依頼明細:** (見積依頼番号, 見積依頼明細番号, 商品番号, 商品名, 商品数量)
- **【変更後】見積依頼明細 [f]:** (見積依頼番号, 取引先番号, 見積依頼明細番号, 商品番号, 商品名, 商品数量)

ここで重要なのは用語の定義とキーの役割です。追加された **取引先番号** は、親テーブル（取引先ごとの見積依頼を管理するエンティティ）の参照先を示す**外部キー**として機能すると同時に、明細行を一意に特定するための**複合主キー**の一部（見積依頼番号, 取引先番号, 見積依頼明細番号）としても機能します。

したがって、変更を加えるべきエンティティ [f] は **見積依頼明細** となります。

- **解答: [f] 見積依頼明細**

リレーショナルデータベースにおけるキー伝播の整合性ルール

親テーブルのキー構造変更が子テーブルへ波及するメカニズムは、以下の**参照整合性**ルールに基づいています。

- **親テーブルの主キー変更:** 親テーブルの識別子（主キー）が単一キーから複合キーに拡張された場合、その親を参照する子テーブルの外部キーも同様に拡張されなければ**参照整合性**を保てない。
- **外部キーの追加伝播:** 「見積依頼明細」は、親テーブルを参照するための**外部キー**として **取引先番号** を新たに取り込む必要がある。
- **複合主キーの再定義:** 親の複合キー（見積依頼番号+取引先番号）に自身の **見積依頼明細番号** を加えることで、取引先ごとに異なる商品明細を保持できる**複合主キー**へと改修される。

💡 **講師の試験対策ワンポイント** 「親テーブルの主キーが変更されたら、その子テーブルのキー構造はどう変化するか？」を追跡する問題は、午後試験のデータベース分野における超頻出パターンです。「キーの変更は配下の明細テーブルへ伝播する」という原則を徹底的に叩き込んでおきましょう。

データベース構造の改修ロジックを理解したところで、続いて SQL による具体的なデータ抽出・集計クエリの解析（設問 3）に移ります。

4. 設問 3: SQL によるデータ抽出と集計（検収一覧表示クエリ）

4.1 サブクエリと条件指定（空欄 h: 抽出条件）

設問3では、検取作業を行うために「発注した各商品の数量」と「実際に納品された数量」を対比して商品番号順に一覧表示する SQL 文（図2）が提示されています。

解析に入る前に、問題文に示された SQL 文の全体像を以下に示します。

```
SELECT ORD.発注番号, ORD.商品番号, ORD.商品名, ORD.発注数量,
```

```
        COALESCE( [ g ], 0)
```

```
FROM
```

```
    (SELECT OD.発注番号, OT.商品番号, OT.商品名, OT.発注数量
```

```
        FROM 発注 OD
```

```
        INNER JOIN 発注明細 OT ON OD.発注番号 = OT.発注番号
```

```
        WHERE [ h ]
```

```
    ) ORD
```

```
LEFT OUTER JOIN
```

```
    (SELECT DE.発注番号, DD.商品番号, SUM(DD.納品数量) AS 納品数量計
```

```
        FROM 納品 DE
```

```
        INNER JOIN 納品明細 DD ON DE.納品番号 = DD.納品番号
```

```
        WHERE DE.発注番号 = :発注番号
```

```
    [ i ]
```

```
    ) DLI
```

```
ON ORD.発注番号 = DLI.発注番号
```

```
AND ORD.商品番号 = DLI.商品番号
```

```
[ j ]
```

サブクエリ（ORD）内の条件絞り込みとパフォーマンス最適化

まず、発注側データを抽出するサブクエリ ORD 内の WHERE 句（空欄 h）を考察します。

問題文には「”:発注番号”は、指定された発注番号を格納する埋込み変数である」と説明されています。

発注表（別名 OD）から指定された特定の 発注番号 のレコードを抽出する必要があるため、空欄

[h] には発注表の列名に別名を付与した条件式を記述します。

- 解答：[h] OD.発注番号 = :発注番号

 **実務アーキテクトの視点** なぜサブクエリの内部で :発注番号 の絞り込みを行うのか？ 最終的な外部結合の後ではなく、サブクエリ ORD および DLI の内部で事前に WHERE 絞り込みを行う手法は、データベースのパフォーマンス最適化における**述語プッシュダウン（Predicate Pushdown）**の基本的な考え方です。結合を行う前に処理対象のレコード数を最小限に絞り込むことで、メモリ消費量を抑え、クエリの実行速度を劇的に向上させることができます。

4.2 外部結合と集計関数・NULL 処理（空欄 i, g）

次に、分割納品された実績を商品ごとに集計するサブクエリ DLI と、メインクエリにおける COALESCE 関数の役割を分析します。

SQL 構成要素と役割の対照

空欄 / 該当箇所	SQL の記述	役割・技術的解説
空欄 i （サブクエリ DLI 内）	GROUP BY DE.発注番号, DD.商品番号	納品表（別名 DE）と納品明細表（別名 DD）を結合後、商品ごとに納品数量を合計（SUM(DD.納品数量) AS 納品数量計）するため、 GROUP BY 句により DE.発注番号 と DD.商品番号 でグループ化を行う。
空欄 g （メイン SELECT 句）	DLI.納品数量計	LEFT OUTER JOIN により納品実績がない場合、DLI 側の列は NULL になる。COALESCE(DLI.納品数量計, 0) と記述することで、NULL の場合は「0」に置換して出力する。

- 解答：[i] GROUP BY DE.発注番号, DD.商品番号
- 解答：[g] DLI.納品数量計

 **講師の試験対策ワンポイント** LEFT OUTER JOIN と COALESCE の最強タッグ（So What?） 発注した商品の中には、「まだ1回も納品されていない商品（未納品）」が存在します。もし INNER JOIN（内部結合）を使用すると、納品実績のない発注商品が検索結果から完全に脱落してしまいます。LEFT OUTER JOIN を使うことで、発注商品（ORD）を全件残したまま納品実績を結合し、さらに COALESCE 関数で NULL を「0」に変換することで、「発注数量：10 個、納品数量計：0 個」という未納品状態を正確に一覧表示できるようになります。

4.3 ソート順の指定（空欄 j：ORDER BY 句）

最後の一覧表示にあたり、問題文の「商品番号の昇順に一覧表示する」という要件を満たすための ORDER BY 句（空欄 j）を決定します。

ソートキー指定における重要な落とし穴

ORDER BY 句で指定する 商品番号 について、結合対象のどちらの表の列を指定するかが決定的なポイントとなります。

- ORD（発注側）：ORD.商品番号
- DLI（納品側）：DLI.商品番号

前述の通り、DLI 側は **LEFT OUTER JOIN** によって結合されているため、未納品の商品については DLI.商品番号 の値が **NULL** になってしまいます。ソートキーに **NULL** が含まれると、データベースの製品仕様によって先頭または末尾に偏り、意図通りの昇順になりません。

一方、発注側である ORD にはすべての対象商品の 商品番号 が必ず存在します (**NOT NULL**)。したがって、ソートキーには確実に値が存在する ORD.商品番号 を指定しなければなりません。

- 解答：[j] **ORDER BY ORD.商品番号**

 **講師の試験対策ワンポイント** 午後試験の設問指示には「表の列名には必ずその表の別名を付けて答えよ」という注記がよく含まれます。ORDER BY 商品番号 とだけ書くと減点対象になります。必ず外部結合の主軸側 (**NULL** にならない側) の別名を付けて ORD.商品番号 と記述してください。

データの抽出・集計クエリの構造を理解したところで、最後の設問である運用開始後のイレギュラー対応「返品処理」(設問 4) の解説へと進みます。

5. 設問 4：実務的イレギュラーへの対処（返品データの記録手法）

5.1 赤黒処理（マイナス値の活用）による履歴保持（30 字以内の記述解説）

システム運用開始後、「納品された商品の一部に問題があったため返品した。その情報をデータベースに記録したい」という実務上のイレギュラー要件が発生しました。

上書き更新（直接減算）が不適切な理由

最初に「登録済みの納品レコードの納品数量から返品数を直接引く（UPDATE する）」方法が検討されましたが、これには重大な欠点があります。

既存レコードの数量を直接減算（上書き）してしまうと、「最初になんの数値で納品されたのか」という過去の事実や、「いつ、いくつ返品されたのか」という処理の履歴が永久に消滅してしまいます。監査証跡やトラブル調査の観点から、会計・購買システムにおいて過去の確定データを直接上書きすることは厳禁とされています。

マイナス値レコード（赤黒処理）のデータ例と計算ロジック

本問では、データベースの定義や既存レコードに変更を加えず、「納品表」と「納品明細表」に新しいレ

コードを 1 件追加 (INSERT) する手法をとります。

以下の具体例で、集計 SQL がどのように動作するかを見てみましょう。

レコード種別	納品番号	商品番号	納品数量	備考
初回納品レコード	N001	P100	+10	取引先から 10 個納品された
返品レコード (追加)	N002	P100	-2	不良品のため 2 個返品した
集計結果 (SUM)	—	P100	8	SUM(納品数量) により自動計算

納品明細表に**「返品した商品の数量をマイナスの値に設定したレコード」**を追加することで、以下のような優れた効果が得られます。

1. **履歴保持:** 当初の納品データ (+10) と返品データ (-2) の両方が履歴として正しく残る。
2. **SQL の再利用:** 設問 3 で作成した集計クエリ (SUM(DD.納品数量)) のコードを変更することなく、そのまま実行するだけで「 $10 + (-2) = 8$ 」という正確な差引納品数量が自動計算される。

解答作成のポイントと文字数制約への収め方

問題文の要件「納品明細表に追加したのはどのようなレコードか。返品に関する情報を記録することを考慮して、30 字以内で述べよ」に従って記述を作成します。

- **盛り込むべき必須要素:** ①返品した商品の数量、②マイナスの値（負数）に設定したレコード
- **解答例 (25 文字):** 返品した商品の数量をマイナスの値に設定したレコード

 **実務アーキテクトの視点** 会計や物流の基幹システムでは、訂正・返品時に元データを消去・更新せず、逆符号のデータを打ち込んで相殺させる伝票処理を**「赤黒処理（赤伝発行）」**と呼びます。追記型 (Immutable) のデータモデリングを行うことで、データの信頼性と監査可能性が保証されます。

本問題全体の解法プロセスをすべて確認できました。最後に、試験本番および実務で役立つ知識を総括としてまとめます。

6. まとめと応用情報技術者試験データベース分野の攻略チェックリスト

6.1 本問題から学ぶデータベース設計の鉄則

平成 30 年度春期 午後問 6「備品購買システムの設計と実装」の解説を通じて、業務モデルからデータベース設計、SQL 実装、データ運用に至る一連の思考プロセスを学びました。

応用情報技術者試験の午後試験や実際のデータベース設計業務において、一貫して求められる鉄則を以下の復習チェックリストとしてまとめました。試験前の最終確認や復習にご活用ください。

データベース設計・復習チェックリスト

1. E-R 図・業務モデル

- ☐ 業務フロー（テキストや表）の記述から、エンティティに必要な属性を正しく抽出できるか（ヘッダー情報と明細情報の分離）。
- ☐ 業務ルール（「複数回に分けて届く」等）から、1 対 1・1 対多・多対多のリレーションシップ（カーディナリティ）を正しく見極められるか。
- ☐ 1 対多への変更（相見積り等）が発生した際、エンティティを分解し、適切な**複合主キー**を設定できるか。
- ☐ 親エンティティの主キー変更に伴い、子エンティティへ**外部キー**および**複合主キー**が伝播するメカニズムを理解しているか。

2. SQL・データ抽出

- ☐ 埋込み変数（:変数名）を用いた動的な条件絞り込み（WHERE 句）が記述できるか。
- ☐ 未実績のデータ（未納品等）を脱落させずに表示するために **LEFT OUTER JOIN** と **COALESCE**（NULL 置換）を適切に組み合わせられるか。
- ☐ 集計関数（SUM）を使用する際、非集計列を正確に **GROUP BY** に指定できるか。
- ☐ ソート（**ORDER BY**）指定時、結合によって **NULL** が発生する可能性がある列を避け、確実に存在する元の列（ORD.商品番号）を選択できるか。

3. 仕様変更・データ運用

- ☐ 既存データを上書き改ざんせず、履歴を維持するための「追記型更新（マイナスレコード/赤黒処理の活用）」の概念を理解しているか。
- ☐ マイナス値を追加することで、既存の集計 SQL（SUM 関数）のロジックを崩さずに**参照整合性**とデータ整合性を保てる理由を説明できるか。

単なる過去問の暗記にとどまらず、「なぜそのテーブル構造になるのか」「なぜその SQL になるのか」という論理的な裏付けを一つひとつ確認していくことこそが、応用情報技術者試験合格への確実な道であり、実務で活躍するデータベースエンジニアとしての強固な土台となります。