

【応用情報技術者試験】平成31年春期 問6「薬剤管理システム」徹底復習・解説レポート

1. 問題の全体像とデータベース設計のキーポイント

本問は、W病院における「薬剤管理システムの再構築」を題材として、概念データモデル（E-R図）の解読から、業務処理とデータベース操作のマトリクス分析、そして高度なSQL文（集合演算・相関サブクエリ・ビュー作成）の構築に至るまで、データベース設計の総合力を試す実践的な問題です。実際のシステム開発現場では、単一の設計書だけで完結することではなく、E-R図、CRUD表、列定義、業務仕様書、さらにはアプリケーションのSQL実装との間に一切の不整合（齟齬）がないよう横断的に設計を進める必要があります。本問を攻略するプロセスを通じて、こうした複数設計書間の連携と、ビューを活用した処理の簡素化手法を体系的に学ぶことができます。

業務背景とシステム要件の整理

W病院における再構築の狙いは、複数の外来診療科および院内薬局を利用する受診者への医療サービス向上です。この目的を達成するため、本システムには医師の処方箋作成を支援する2つの核となるチェック機能が実装されます。

- **薬剤併用チェック機能**
 - **目的：** 併用を禁止する“併用禁忌”や、併用に注意を要する“併用注意”の薬剤の組み合わせを未然に防ぎ、処方安全性を担保する。
 - **仕組み：** 処方箋発行の際、今回処方しようとしている薬剤と、過去6か月以内にW病院で発行された処方箋に記載された薬剤との組み合わせに対し、薬剤併用情報テーブルと照合して不適切な重複を抽出する。
- **在庫チェック機能**
 - **目的：** 院内薬局での処方を前提とし、発行した処方箋の薬剤が有効年月日（処方箋の期限）まで確実に確保されるよう在庫の保証を行う。
 - **仕組み：** 処方箋を発行し、有効年月日までに院内薬局で処方する可能性のある薬剤の処方量合計を「確保量」として集計・管理し、実在庫と照らし合わせる。

出題趣旨の構造化

IPAが公表している出題趣旨によると、本問は「複数のデータベース設計書間の関連を横断的に理解する能力」および「ビューを作成するためのSQLに関する理解」を問うことを目的としています。この趣旨を具現化するため、問題全体が以下のように密接に構造化されています。

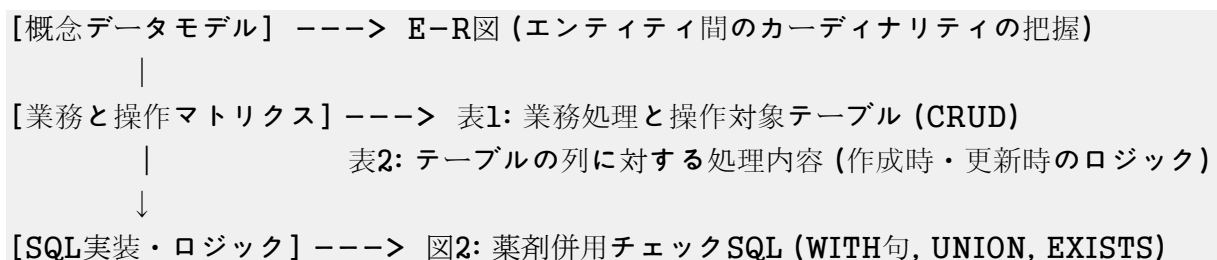


図3: 在庫確保量ビュー作成SQL (CREATE VIEW, SUM, 日付判定)

E-R図で定義されたエンティティ間の「1対1」や「1対多」のリレーションシップは、表1における業務処理の発生タイミング（同時作成されるか否か等）と直接連動しています。また、表2に規定された列ごとの更新ルールは、図2や図3で表現されるSQL文の抽出・更新条件と厳密に対応しています。データベース設計書同士の相互関係とロジックの一貫性を意識することは、試験対策のみならず実務においても不具合を防ぐための最重要事項です。それでは、各設問の解説を通じて、データベース構造の解明を進めていきましょう。まずは、システム全体の骨格となる「E-R図のリレーションシップ解説」から確認します。

2. 設問1：E-R図とエンティティ間リレーションシップの解説

概念データモデル（E-R図）におけるカーディナリティ（多重度）の正確な把握は、データベースの参照整合性を保ち、業務ルールを忠実に構造化する上で決定的な役割を果たします。本問では、業務処理の発生条件とエンティティの生成規則を読み解くことで、空欄aおよびbの適切なリレーションシップを導き出すことができます。

E-R図の記号凡例とカーディナリティの概念的背景

本問の図1「凡例」に用いられている表記法では、各端点の図記号がエンティティのオプションナリティ（任意性）とカーディナリティ（多重度）を明確に表現しています。

- 端点の「黒丸（●）」： そのエンティティが必須（1以上）で存在することを示します。
- 端点の「白丸（○）」： そのエンティティが任意（0以上）で存在し得ることを示します。
- 端点の「矢印（➤）」： 多（複数）のレコードが存在し得ることを示します。例えば「●---●➤」であれば、「1」側の端点が黒丸（必須存在）、「多」側の端点が黒丸と矢印（1以上存在）を指し、**「1対多（1以上）」**を意味します。一方、「●---○➤」であれば「1対多（0以上）」となります。この図記号の意味合いを正しく識別することが、凡例記号を正確に選択する鍵となります。

空欄a（受診者と外来受診）の導出分析

「受診者」エンティティと「外来受診」エンティティ間のリレーションシップaを特定します。主キーの観点から見ると、「受診者」の主キーは受診者IDであり、「外来受診」の主キーは受診者IDと受診予定日時の複合キーです。これは、1人の受診者が複数回外来受診を受ける可能性があることを意味しています。さらに、表1「業務処理と操作対象のテーブル」における受付の処理（1）「初回の外来受診」を確認すると、操作対象のテーブルとして「受診者」と「外来受診」の2つが「作成」欄に指定されています。すなわち、受診者が初めて病院を訪れた際、受診者レコードと同時に1件目の外来受診レコードが必ず作成されます。そして処理（2）「2回目以降の外来受診」では「外来受診」のみが作成されます。したがって、受診者1人に対して外来受診は必ず1件以上存在することになり、受診者から見た外来受診の関係は**「1対多（1以上）」**となります。「1」側の端点が黒丸（必須）」かつ「多」側の端点が黒丸と矢印（1以上）」の記号を選択するため、正解記号は「カ」*となります。

空欄b（薬剤と薬剤在庫）の導出分析

「薬剤」エンティティと「薬剤在庫」エンティティ間のリレーションシップbを特定します。両エンティティの主キーはともに薬剤コードです。表1における院内薬局の処理（9）

「初回発注の薬剤」の「作成」欄を見ると、「薬剤」と「薬剤在庫」の両方のテーブルが同時に作成されます。また、表2「テーブルの列に対する処理内容」の「薬剤在庫」テーブルの行を確認すると、作成時（9）の処理内容として「0を設定する。」と記載されています。さらに問題文の記述や解説情報にも「薬剤は必ず薬剤在庫をもち、在庫量が0の場合も薬剤在庫は存在します」と明記されています。つまり、薬剤が存在すれば、対応する薬剤在庫レコードが必ず1対1で存在し、その逆も成立します。したがって、関係性は両端が黒丸で結ばれる**「1対1」となります。凡例より、正解記号は「ウ」**となります。

設問1 正解まとめ表

対象エンティティ, 正解の記号, リレーションシップの名称, 判定根拠（業務ルール）

受診者 - 外来受診, カ, 1対多（1以上）, 初回外来受診時（処理1）に両レコードが同時作成され、2回目以降（処理2）も受診レコードが追加作成されるため、受診者に対して外来受診は1以上存在する。

薬剤 - 薬剤在庫, ウ, 1対1, 初回発注時（処理9）に両レコードが同時作成され、在庫量が0であっても薬剤在庫レコードは必ず存在するため。

エンティティ間の静的な関係性を正しく解読できたら、次はこれらのテーブルが実際の業務プロセスの中でどのように参照・作成・更新・削除されるかという「動的な操作（CRUDマトリクス）」の分析へと進みます。

3. 設問2：CRUDマトリクスと列処理の整合性分析

データベース設計における矛盾や欠陥を排除するためには、業務処理フロー（表1）とテーブル列の操作・更新ルール（表2）の間でロジックが完全に一致しているかを検証する作業が不可欠です。本設問では、業務処理のコンテキストと外部キーの整合性から空白項目を導出します。

（1）操作対象テーブル c, d の特定

空欄c（処方箋作成時のチェック結果による見直し）

表1の医師による「処方箋の作成」において、処理（5）「薬剤併用チェック」で「作成」および「削除」され、処理（6）「在庫チェック」で「更新」されるテーブルcを特定します。問題文の記述より、併用チェックや在庫チェックの結果、医師が処方する薬剤の見直しを行う際、薬剤の追加や変更（削除・作成・更新）が発生します。具体的に処方される個々の薬剤と分量を記録しているのは**「処方箋明細」テーブルです。また問題文には「見直しの結果、処方箋明細が0件になることもあるが、このような場合には、処方箋の発行は行わない運用とし、処方箋レコードは削除しない」と明記されているため、見直しによって追加・削除・更新される対象は「処方箋」ではなく「処方箋明細」であることが確定します。したがって、c = 処方箋明細** です。

空欄d (薬剤の入庫)

表1の処理 (11) 「薬剤の入庫」で「更新」されるテーブルdを特定します。表2の「薬剤在庫」テーブルの項目を確認すると、更新時の処理として「値を加える。(11)」と記載されています。薬剤が入庫した際には、対応する薬剤在庫テーブルの在庫量_大人1日に数量が加算されるため、対象テーブルは**「薬剤在庫」 となります。したがって、 d = 薬剤在庫**です。

(2) 受診者住所列の更新条件 e の解析

表2の「受診者」テーブルの住所列に対する更新時の処理内容eを導出します。表1で「受診者」テーブルが「更新」される業務処理を探すと、受付の処理 (3) 「受診者情報変更時の修正」のみが該当します。受診者の住所に変更があった場合はこの処理 (3) で値が書き換えられるため、表2の表記形式 (作成時の「住所を設定する。(1)」等) と対応させて記述します。したがって、 e = 値を更新する。(3) となります。

(3) 外来受診テーブルの列 f の特定

表2の「外来受診」テーブルにおける列fを特定します。表2の記述によると、列fは作成時 (処理1および2) には「NULLを設定する。(1)(2)」とされ、更新時 (処理4) に「値を更新する。(4)」と処理されます。図1のE-R図で「外来受診」の属性を確認すると、受診者ID (主キー)、受診予定日時 (主キー)、診療科コード、そして破線下線の外部キーである処方箋IDが存在します。受付段階 (処理1, 2) ではまだ処方箋は作成されていないため、処方箋IDにはNULLが設定されます。その後、医師が処理 (4) 「処方箋の登録 (発行前)」を行う段階で処方箋が作成され、そのIDが「外来受診」テーブルの外部キー列にセット (更新) されます。したがって、 f = 処方箋ID です。データベースの構造と各列の更新ロジックが矛盾なく整理されたことを確認できました。次は、これらのデータを活用して高度な判定を行う「SQLクエリの解説 (設問3・4)」へと進みます。

4. 設問3：薬剤併用チェックSQL (集合演算と存在確認) の構造分解

薬剤併用チェック機能を実現するSQL文 (図2) は、患者の過去の処方履歴および今回処方予定の薬剤から、重複なくチェック対象の薬剤集合を抽出し、それらの全組み合わせ (直積) が「薬剤併用情報」テーブルに登録されているかを判定する高度な構造を持っています。

SQL構文の全体構造解説

図2のSQLは、大きく分けて**「WITH句によるチェック対象薬剤の抽出 (一時テーブルの作成)」と「メインクエリによる併用情報テーブルとの照合 (関連サブクエリと存在確認)」**の2段階構成になっています。

[WITH句: チェック対象薬剤]

- └── ① 過去6か月以内に発行された処方箋の薬剤コード (SELECT文1)
- └── [g] (集合演算子: 複数のSELECT文の統合)
- └── ② 今回処方予定の処方箋の薬剤コード (SELECT文2)

[メインクエリ]

SELECT * FROM 薬剤併用情報 T1

WHERE [h] (存在確認句)

(CROSS JOINで①と②から生成した全薬剤ペア T2 と T1 を結合・照合)

空欄g (集合の統合) の導出

WITH句内部のチェック対象薬剤を作成するクエリでは、2つのSELECT文を統合しています。

- **1つ目のSELECT文:** 対象受診者の過去6か月以内の外来受診および処方箋に紐付く「処方箋明細」から薬剤コードを取得。
- **2つ目のSELECT文:** 今回の処方箋 (:処方箋ID) に紐付く「処方箋明細」から薬剤コードを取得。併用チェックを行うためには、過去の薬剤集合と今回予定の薬剤集合をひとつに足し合わせる (和集合を求める) 必要があります。複数クエリの結果セットを統合するSQL文法はUNION (または重複排除を行わない UNION ALL) です。なお、今回は続くメインクエリ側で全組み合わせを作成してチェックするため、重複が含まれていても機能上問題なく、どちらを用いても正解となります。したがって、 g = UNION 又は UNION ALL です。

空欄h (関連サブクエリと存在判定) の導出とCROSS JOINの役割

メインクエリでは、薬剤併用情報テーブル (T1) を参照しています。ここで注目すべきは、サブクエリ内部でチェック対象薬剤 (U1, U2) に対してCROSS JOIN (直積結合) を適用している点です。CROSS JOINを用いることで、「過去の処方薬剤」と「今回の処方薬剤」から構成されるすべての薬剤の全ペア (T2) を二次元的に展開・生成しています。これは関係代数における直積演算 (Cartesian Product) であり、漏れなくすべての組み合わせを生成するための極めてエレガントなアプローチです。そして、薬剤併用情報テーブル (T1) の薬剤コード1, 薬剤コード2と、T2で生成したペアの薬剤コード1, 薬剤コード2が合致するレコードが「サブクエリ側に存在するかどうか」を判定しています。指定した条件を満たす行が存在するかどうかを判定するSQL述語はEXISTS です。したがって、 h = EXISTS となります。

該当SQLコードブロック

WITH チェック対象薬剤 AS(

SELECT B1.薬剤コード FROM 処方箋明細 B1,

(SELECT A1.処方箋ID FROM 外来受診 A1, 処方箋 A2

WHERE A1.受診者ID = :受診者ID AND A1.処方箋ID = A2.処方箋ID AND

A2.発行年月日 >= TO_DATE(:半年前年月日)) B2

WHERE B1.処方箋ID = B2.処方箋ID

-- [g]: UNION 又は UNION ALL (過去の薬剤集合と今回の薬剤集合の統合)

UNION

SELECT C1.薬剤コード FROM 処方箋明細 C1

WHERE C1.処方箋ID = :処方箋ID)

```

SELECT * FROM 薬剤併用情報 T1
WHERE
  -- [h]: EXISTS (直積テーブルT2に存在する併用禁忌・注意ペアかを判定)
  EXISTS
  (SELECT T2.薬剤コード1, T2.薬剤コード2 FROM
    (SELECT U1.薬剤コード AS 薬剤コード1, U2.薬剤コード AS 薬剤コード2
     FROM チェック対象薬剤 U1 CROSS JOIN チェック対象薬剤 U2) T2
   WHERE T1.薬剤コード1 = T2.薬剤コード1 AND T1.薬剤コード2 = T2.薬剤コード2)

```

併用チェックのロジックを解読できました。続いて、もう一つの核心機能である「在庫確保量ビュー作成SQL」の解読に移ります。

5. 設問4：在庫確保量ビュー作成SQL（集計と条件判定）の完全攻略

院内薬局における在庫管理では、処方箋が発行された時点でその薬剤をリアルタイムに実在庫から引き落とすのではなく、有効期限内において処方される可能性のある「確保量」を仮想的に集計・管理する手法が採用されています。このように集計ロジックをビュー（VIEW）として共通化することで、個別アプリケーション側の処理を大幅に簡素化し、データ整合性を高めることができます。

空欄i（グループ集計関数）の解析

図3のSQL文は、CREATE VIEW 処方前確保在庫(薬剤コード, 確保量_大人1日) AS ... という形式で定義されています。最外層のクエリでは GROUP BY T3. 薬剤コード が指定されているため、SELECT句のiには薬剤コードごとの集計値を求める集計関数が必要です。問題文の記述「処方する可能性のある薬剤の処方量の合計を、確保量として管理するためのビュー」および、サブクエリ内で選択されている列名T2. 処方量_大人1日から、処方量の合計（SUM）を計算すればよいことがわかります。したがって、 **i = SUM(T3. 処方量_大人1日)** となります。

空欄j（有効期限の判定条件）の解析

サブクエリ内のWHERE句において、「処方箋」テーブル（T1）の有効年月日に対する条件式jを導出します。問題文に「参照時の日付をDATE型で返す日時値関数CURRENT_DATE」についての説明があります。発行された処方箋が現在有効（＝まだ期限切れになっておらず、確保すべき対象である）とみなされるのは、処方箋の有効年月日が参照日（当日）以降である場合です。したがって、T1. 有効年月日 >= CURRENT_DATE という不等式条件が必要となります。これにより、 **j = >= CURRENT_DATE** （または **>= CURRENT_DATE**）となります。

空欄k（未処方状態の判定条件）とNULLによる状態管理

同じくサブクエリ内のWHERE句で、「処方箋」テーブル（T1）の薬剤処方年月日に対する条件式kを解析します。表2の「処方箋」テーブルの記述を確認すると、処理（8）「薬剤の処方（薬剤の受渡し）」が行われた時点で、薬剤処方年月日列の「値を更新する。」と指定されています。逆に、処方箋が発行されたのみで、まだ院内薬局で薬剤の受渡しが行われていない状態では、薬剤処方年月日には初期値である **NULL** が設定されています。実務のデー

データベースアーキテクチャにおいても、決済完了日や受渡完了日といった「イベント発生日付列」がNULLか否かで、タスクの未処理・処理完了状態（ライフサイクル）を識別・管理する設計パターンが頻繁に用いられます。既に受渡された薬剤は実在庫から減算されるため「確保量（処方待ち）」として集計し続ける必要がありません。したがって、これから処方される可能性のある「未処方状態（処方待ち）の処方箋」のみを対象とするには、T1. 薬剤処方年月日 IS NULL という条件を指定する必要があります。したがって、**k = IS NULL** となります。

該当SQLコードブロック

```
CREATE VIEW 処方前確保在庫(薬剤コード, 確保量_大人1日) AS
SELECT T3.薬剤コード,
      -- [i]: SUM(T3.処方量_大人1日) (薬剤ごとの処方量合計を算出)
      SUM(T3.処方量_大人1日)
FROM
  (SELECT T2.薬剤コード, T2.処方量_大人1日
   FROM 処方箋 T1, 処方箋明細 T2
   WHERE T1.処方箋ID = T2.処方箋ID AND T1.発行年月日 <= CURRENT_DATE AND
        -- [j]: >= CURRENT_DATE (有効期限内の処方箋を抽出)
        T1.有効年月日 >= CURRENT_DATE AND
        -- [k]: IS NULL (未処方状態・処方待ちの処方箋のみを抽出)
        T1.薬剤処方年月日 IS NULL) T3
GROUP BY T3.薬剤コード
```

6. 復習のまとめ：応用情報データベース分野を攻略する3つの鉄則

平成31年春期 問6「薬剤管理システム」は、単なる知識の丸暗記では解き明かせない、「業務要件」「静的構造（E-R図）」「動的遷移（CRUD）」「実装（SQL/VIEW）」の四者が美しく統合された **応用情報技術者試験データベース分野における屈指の名作問題** です。出題者がこの一問に込めたデータベース設計の本質を体得し、今後の試験対策や実務における設計能力の向上に結びつけるための「3つの鉄則」として総括します。

鉄則1：業務ルールとE-R図の連動

カーディナリティ（1対1、1対多、多対多、1以上/0以上など）は、単なる見た目の関係性ではなく、必ず**「業務処理（レコード作成・更新）の発生タイミング」**から導出してください。

- 「処理Aの際に両テーブルが同時に作成されるか？」
- 「親レコードが存在しても子レコードが0件の状態で存在し得るか？」
- 「キー構造にどのような複合キーが設定されているか？」 これらを業務記述やCRUD表から読み解くことが正解への最短ルートです。

鉄則2：CRUD表とテーブル設計の双方向チェック

CRUD表（業務処理と操作対象テーブル）と、テーブルの列操作ルール（表2など）は常に双方向で検証してください。

- 業務処理で「作成・更新・削除」の対象となっているテーブルと、列レベルで「初期設定・更新」が規定されているテーブルに矛盾がないか。
- 外部キーの列に初期状態でNULLが入り、どの業務イベントで値が設定（更新）されるのか、ライフサイクルを追う意識を持ちましょう。

鉄則3：複雑なSQLの役割分解

長文のSQLが出題された場合は、一括で読解しようとせず、機能ごとのパーツに構造を分解してアプローチしてください。

1. **WITH句・サブクエリ**： どのような一時集合（中間表）を作ろうとしているか確認する。
2. **集合演算（UNION等） / 結合（CROSS JOIN等）**： 抽出されたデータをどう統合・掛け合わせているか把握する。
3. **相関サブクエリ（EXISTS等）**： どのキーを用いて存在判定を行っているか見極める。
4. **集計ビュー（CREATE VIEW / GROUP BY）**： 何の単位でグループ化し、どの集計関数（SUM, COUNT等）を適用しているか、条件（WHERE句の日付判定やNULL判定）の意図を分解して読み解く。

復習用 セルフチェックリスト

最後に、本問の解き直しを行う際に以下のポイントを自力で論理的に説明できるかチェックしてみましょう。

- **a受診者と外来受診**： 初回受診時に両方作成され、2回目以降は外来受診のみ増えるため「1対多（1以上：カ）」と即答できるか？
- **b薬剤と薬剤在庫**： 初回発注時に同時作成され、在庫0でも存在するため「1対1（ウ）」と説明できるか？
- **c処方箋明細**： 処方見直し時に追加・変更・削除され、0件でも処方箋自体は削除しないルールから導出できたか？
- **f処方箋ID**： 外来受診テーブルの作成時にはNULLであり、処方箋登録時にセットされる外部キーであると見抜けたか？
- **gUNION / hEXISTS**： 過去と今回の薬剤集合の統合、およびCROSS JOINによる全ペアの存在確認ロジックを理解できたか？
- **iSUM / j>= CURRENT_DATE / kIS NULL**： 確保量ビューにおいて、有効期限内かつ未処方（NULL＝ライフサイクル未完了）の処方量を集計する条件式を正確に組み立てられたか？