

令和 5 年度春期応用情報技術者試験 午後問 6 徹底解説&復習レポート

1. システム開発の背景とビジネス要件の整理

システム開発の背景と戦略的重要性

G 社は、創立 20 年を迎えた従業員 500 人規模のソフトウェア開発会社です。同社ではこれまで、顧客企業や業種業界の変化に応じた組織変更を行うとともに、スキルや業務知識に応じた柔軟な人事異動を実施することで、人材の流動性を高めてきました。

今年度、G 社では「仕事にメリハリを付け、仕事の質を向上させることが業績向上につながる」という考えのもと、従来運用してきた評価指標に加え、働き方改革および従業員満足度向上に関する新たな KPI 項目を導入することを決定しました。これらの追加 KPI の達成状況を正確に把握し、各組織が計画的な目標達成を推進するための仕組みとして開発されるのが「KPI 達成状況集計システム（以下、K システム）」です。

業務上の施策（働き方改革や満足度向上）をシステムとして運用するためには、ビジネスルールを厳密に定義し、それをデータベース構造や集計処理へ正しく落とし込む必要があります。システム設計の初期段階でビジネス要件の精緻な整理を行うことは、その後のデータベース構築の正当性を担保する上で極めて重要な意味を持ちます。

組織構造と追加 KPI 項目の構造化整理

G 社の組織構造および職務区分、ならびに今年度追加された KPI 項目の仕様は以下の通り整理されます。

組織構造と職務区分の特徴

- **組織構造:** 最大 3 つの階層から構成され、1 階層から 3 階層までの各組織には必ず 1 名以上の従業員が所属します。複数階層や複数組織の兼務は行わない規定であり、各従業員は常に 1 つの組織だけに所属します。
- **職務区分:** 「管理職」と「一般職」の 2 つに区分されます。

追加された KPI 項目一覧

今年度から追加された 3 つの KPI 項目、定量成果目標、および評価方法は下表の通りです。

KPI 項目名	定量的成果目標	評価方法
年間総労働時間	1,980 時間以内／人	・ 一般職従業員の個人実績を組織単位で集計し、平均値の達成状況进行评估する。 ・ 年度途中入社、年度途中退職した従業員は、評価対象外とする。 ・ 個人実績の集計は、集計日時点で従業員が所属している直属の組織に対して行う。所属組織の上位階層、又は下位階層の組織の集計には含めない。
年次有給休暇取得日数	16 日以上／人	同上
年間研修受講日数	6 日以上／人	同上

ビジネスルールが集計論理に与える影響（So What? の分析）

表層的な要件の裏にあるビジネスルールは、システムにおける SQL 等の集計論理に対して以下のような決定的な影響を及ぼします。

- 「**管理職は KPI 評価対象外**」というルールの影響: 管理職も一般職と同様に月別目標値の入力を行います、評価対象となるのは一般職のみです。したがって、集計処理の段階で役職表の職務区分を参照し、管理職データを厳密に除外する条件指定が必要となります。
- 「**年度途中入社・退職者は評価対象外**」というルールの影響: 集計対象を「年度開始時点で在籍し、かつ集計日時点でも在籍している従業員」のみに絞り込む必要があります。年度途中に途中入社した従業員は 月別個人目標 の初期作成レコードが存在せず、途中で退職した従業員は集計日時点の 所属 に存在しないため、これらの特性を利用したテーブル結合論理（抽出フィルタリング）を設計しなければなりません。
- 「**直属の組織のみで集計**」というルールの影響: 上位組織や下位組織の実績を含めず、集計時点で従業員が直接所属している組織コードのみでグループ化・集計を行うロジックが求められます。

次章への接続

このように、企業の業務ルールや評価基準を明確化することは、関係データベース上でデータを保持するためのエンティティ設計や、エンティティ同士を結びつけるリレーションシップ（関連）を正しく決定するための不可欠な前提となります。次章では、これらの要件を具現化する E-R 図の設計構造について解説します。

2. E-R 図の設計とエンティティ間リレーションシップの解析（設問 1 対策）

E-R 図とデータモデルの整合性

関係データベースを中心としたシステム開発において、E-R 図（実体参照図）はシステムが扱うデータの構造と相互関係を可視化するための骨組みです。各エンティティの主キー属性を正しく設定し、エンティティ間のカーディナリティ（多重度）や参照制約を厳密に定義することは、データの重複を防ぎ、整合性を長期にわたって維持するために不可欠です。

新規追加エンティティと運用の分析

K システム構築のために追加された 3 つのエンティティ（KPI、月別個人目標、日別個人実績）の管理内容と運用方法は以下の通り定義されています。

表（エンティティ名）	管理内容	運用方法
KPI	KPI 項目と定量成果目標を管理する。	・ 参照だけ（更新は行わない）。
月別個人目	個人ごとの月別目	・ 年度開始時点で在籍している全従業員に対して、当該年度分のレコ

標	標値を管理する。	ードを、目標値を 0 として初期作成する。 ・初期作成したレコードに対して、各人で定量的な成果目標を意識した月別目標値を入力し、定期的に見直し、更新する。 ・年度途中入社に従業員については、初期作成レコードが存在しない。月別目標値の入力も行わない。 ・管理職従業員は KPI 評価対象外であるが、月別目標値の入力は一般職従業員と同様に行う。
日別個人実績	個人ごとの日別実績値を管理する。	・勤怠管理システム、研修管理システムで管理している追加した KPI 項目に関する全従業員の実績値を基に、日次バッチ処理によってレコードを作成する。 ・日別実績のない従業員のレコードは作成しない。

E-R 図構造の視覚的理解

本問における主要エンティティのリレーションシップと主キー・外部キーの接続構造をテキストダイアグラムで可視化すると以下のようになります。



設問 1 詳細思考プロセス解説

空欄 [a] の導出論理（組織エンティティと所属エンティティ間の関連）

- **思考のステップ 1（主キー・外部キーの対応確認）**：組織 エンティティには主キーとして 組織コード と 組織開始年月日 が存在します。一方、所属 エンティティには外部キーとして 所属組織コード と 所属組織開始年月日 が保持されており、組織 エンティティを参照する関係にあります。
- **思考のステップ 2（業務ルールの適用）**：問題文には「1 階層から 3 階層のそれぞれの組織には 1 名以上の従業員が所属している」と記述されています。これは、1 つの組織に対して、そこに所属する複数の従業員（所属レコード）が存在することを意味します。つまり、同じ 所属組織コード と 所属組織開始年月日 の組み合わせを持つレコードが 所属 エンティティ側に複数存在し得ます。
- **結論**：組織 から見て 所属 は「1 対多」の関係となります。したがって、凡例の表記に従い、空欄 [a] に入る矢印は ->（1 対多）となります。

空欄 [b] の導出論理（従業員エンティティの主キー）

- **思考のステップ 1（主キーの必要性の認識）**：従業員 エンティティの定義を確認すると、属性欄の最上部に空欄 [b] があり、主キー（実線の下線）が欠落しています。エンティティを一意に特定するための主キー属性を設定する必要があります。
- **思考のステップ 2（他エンティティからの参照関係の確認）**：従業員 エンティティと 1 対多の関連を持つ子エンティティ（月別個人目標 や 日別個人実績）を参照すると、外部キーとして 従業員コード が定義されています。また、所属 エンティティにおいても 従業員コード が参照用キーとして存在します。
- **思考のステップ 3（属性の同定）**：従業員を一意に特定し、他のエンティティからの参照を正しく成立させることができる属性は 従業員コード です。表記ルール（主キーには実線の下線を付す）に従い、解答は <u>従業員コード</u>（下線付き）となります。

組織変更・人事異動履歴管理の技術的意義（So What? の分析）

G 社における過去から未来にわたる組織変更や人事異動を E-R 図上で正しく表現することには、非常に大きな技術的意義があります。

組織 表および 所属 表において、開始年月日 と 終了年月日 を複合キー（主キーの一部）として保持する設計を採用しています。さらに、終了予定がない場合には 9999 年 12 月 31 日 という未来の固定日付を設定する運用を行っています。

この構造により、単に「現在の組織・所属」だけでなく、「過去の特定の年月日時点において、どの従業員がどの組織に所属していたか」という過去の事実（歴史的データ）を精度高く再現できるようになります。また、上位組織コード や 上位組織開始年月日 を保持することで、組織の階層構造の推移もデータベース上で完全に保持・追跡することが可能となります。

次章への接続

E-R 図によって厳密に定義されたデータ構造と履歴管理の仕組みをベースとして、K システムでは集計データをどのように加工し評価情報を作成していくのでしょうか。次章では、月次バッチ処理で実行される集計処理の流れと、一時表（ワークテーブル）を活用した処理設計について詳しく確認します。

3. 集計処理の流れと一時表（ワークテーブル）の役割

機能分割と再利用設計の意図

複雑な条件判断や大量のデータを伴うデータ集計において、1本の極めて巨大で複雑なSQL文のみで処理を行おうとすると、可読性の低下や保守性の悪化、さらにはデータベースエンジンへの過度な負荷を引き起こす要因となります。

Kシステムでは、集計リストの作成処理を複数の単純なステップに機能分割し、途中の計算結果を一時表（ワークテーブル）へ出力して後続処理に連携していくアプローチを採用しています。このように処理を分割・統合し、再利用可能な構造に設計することは、システム構築工数の削減および処理性能・メンテナンス性の向上を図る上で極めて重要です。

集計リスト作成処理（表4）のステップ別分解

月次バッチ処理において実行される集計リスト作成の処理内容は、以下の5つのステップとして整理されます。

【項番1】所属・役職の抽出

[入力表] 所属, 役職

|

▼（集計日時点で所属する一般職従業員を抽出）

[出力表] 従業員_所属_一時

【項番2】目標値の個人別集計

[入力表] 月別個人目標

|

▼（年度開始年月から集計月までの目標値を合計）

[出力表] 従業員ごと_目標集計_一時

【項番3】実績値の個人別集計

[入力表] 日別個人実績

|

▼（年度開始年月日から集計日までの実績値を合計）

[出力表] 従業員ごと_実績集計_一時

【項番4】組織・KPIごとの目標・実績統合集計

[入力表] 従業員_所属_一時, 従業員ごと_目標集計_一時, 従業員ごと_実績集計_一時

|

▼（3つの一時表を結合し、組織・KPI単位で集計）

[出力表] 組織ごと_目標実績集計_一時

【項番5】最終集計リストの作成

[入力表] 組織ごと_目標実績集計_一時

|

▼（組織ごとの目標平均値・実績平均値を算出して出力）

[出力表]（集計リストを作成・提示）

各ステップの具体的な処理内容は次の通りです。

1. **項番 1:** 所属 表と 役職 表を入力とし、集計日時点で在籍している一般職従業員と所属組織の対応関係を抽出して 従業員_所属_一時 に出力する。
2. **項番 2:** 月別個人目標 表を入力とし、年度開始年月から集計月までの期間における従業員・KPI 項目ごとの目標個人集計値を求め、 従業員ごと_目標集計_一時 に出力する。
3. **項番 3:** 日別個人実績 表を入力とし、年度開始年月日から集計年月日までの期間における従業員・KPI 項目ごとの実績個人集計値を求め、 従業員ごと_実績集計_一時 に出力する。
4. **項番 4:** 項番 1～3 で作成した出力表（3 つの一時表）を入力として結合し、組織・KPI 項目ごとの目標組織集計、実績組織集計、および対象従業員数を算出して 組織ごと_目標実績集計_一時 に出力する。
5. **項番 5:** 項番 4 の出力表を入力とし、各数値から目標平均値および実績平均値を算出して一覧化した集計リストを作成する。

一時表を利用した設計の利点評価（So What? の分析）

なぜ 1 つの大型 SQL 文で直接集計を行わず、このように一時表を多段に経由させる手法を取るのでしょうか。その主な理由は以下の通りです。

- **処理ロジックの単純化と可読性・保守性の向上:** 「対象従業員の絞り込み（項番 1）」「個人目標の合計（項番 2）」「個人実績の合計（項番 3）」という観点ごとに処理を独立させることで、SQL の構造が明確になり、バグの混入を防ぐとともに、後からの仕様変更にも容易に対応できるようになります。
- **構築工数の削減と再利用性:** 正規化された表を扱う処理において機能分割を行うことで、共通の抽出結果（例：対象従業員一覧）を再利用することができ、開発における全体の構築工数を削減できます。
- **ビジネスルールの正確な適用:** 例えば「年度途中入社を除外する」ロジックは項番 2 で自然に達成され（途中入社者は目標初期レコードが存在しないため）、「退職者を除外する」ロジックは項番 1 で達成されます。これらの中間結果を項番 4 で結合することにより、極めて複雑な評価除外ルールを間違いなく実現できます。

次章への接続

集計フロー全体の設計意図と一時表の役割を踏まえた上で、次章ではそれぞれのステップで実行される実際の SQL 文の構文および条件設定のテクニックについて詳しく解析していきます。

4. SQL 文の精読とテーブル結合・データ集計のテクニック（設問 2 対策）

条件絞り込みと結合テクニックの重要性

関係データベース言語である SQL において、狙い通りの集計結果を得るためには、「どの行を抽出するか（WHERE 句の条件述語）」、「複数の表をどのように結びつけるか（JOIN の選定）」、「データが存在しない場合（NULL 値）をどう扱うか（COALESCE 関数等）」を正確に制御する必要があります。これらは実務のシステム開発および試験対策の双方において最も差がつくコアな知識です。

設問 2(1) 各空欄 [c] ～ [i] の完全解説

試験問題（図 2）に提示された SQL コードブロックを元に、空欄を補完した実際の構文構造とその解説を示します。

項番 1 の SQL 文と空欄 [e], [f]

-- 項番 1：一般職従業員と所属組織の対応表を作成

```
INSERT INTO 従業員_所属_一時(従業員コード, 組織コード)
SELECT A.従業員コード, A.所属組織コード FROM 所属 A, 役職 B
WHERE TO_DATE(:集計年月日) [ e: BETWEEN ] A.所属開始年月日 AND A.所属終了年月日
AND A.役職コード = B.役職コード
AND [ f: B.職務区分 = '02' ]
```

- [e] (BETWEEN)：集計年月日が所属期間（所属開始年月日 から 所属終了年月日）の範囲内にあるかを判定する述語です。値 BETWEEN 開始 AND 終了 の形式で「指定した日付が範囲に含まれているか」を指定するため、BETWEEN が入ります。
- [f] (B.職務区分 = '02')：一般職従業員のみを厳密に抽出する条件式です。問題本文の通り、役職表の職務区分は「管理職：'01'、一般職：'02'」と定義されています。したがって B.職務区分 = '02' と指定します。

項番 2 および項番 3 の SQL 文と空欄 [e], [g]

-- 項番 2：年度開始年月から集計月までの目標個人集計値を算出

```
INSERT INTO 従業員ごと_目標集計_一時(従業員コード, KPI コード, 目標個人集計)
SELECT 従業員コード, KPI コード, SUM(月別目標値) FROM 月別個人目標
WHERE 年月 [ e: BETWEEN ] :年度開始年月 AND :集計年月
[ g: GROUP BY 従業員コード, KPI コード ]
```

-- 項番 3：年度開始年月日から集計日までの実績個人集計値を算出

```
INSERT INTO 従業員ごと_実績集計_一時(従業員コード, KPI コード, 実績個人集計)
SELECT 従業員コード, KPI コード, SUM(日別実績値) FROM 日別個人実績
WHERE 年月日 [ e: BETWEEN ] TO_DATE(:年度開始年月日) AND TO_DATE(:集計年月日)
[ g: GROUP BY 従業員コード, KPI コード ]
```

- [e] (BETWEEN) の共通適用理由：項番 1 と同様に、項番 2 における年月の範囲（:年度開始年月 ～ :集計年月）および項番 3 における年月日の範囲（TO_DATE(:年度開始年月日) ～ TO_DATE(:集計年月日)）の双方において、特定の範囲内を抽出するため BETWEEN が当てはまります。
- [g] (GROUP BY 従業員コード, KPI コード)：SELECT 句で SUM(月別目標値) や SUM(日別実績値) という集合関数（集計関数）が使用されています。「従業員コード」および「KPI コード」の単位ごとに集計を行う必要があるため、グループ化句として GROUP BY 従業員コード, KPI コード を明記します。

項番 4 の SQL 文と空欄 [h], [i], [c], [d]

-- 項番 4：組織、KPI 項目ごとの統合集計

```

INSERT INTO [ h: 組織ごと_目標実績集計_一時 ]
(組織コード, KPI コード, 目標組織集計, 実績組織集計, 対象従業員数)
SELECT A.組織コード, B.KPI コード, SUM(B.目標個人集計),
      SUM(COALESCE(C.実績個人集計, 0)), [ i: COUNT(*) ]
FROM 従業員_所属_一時 A
[ c: INNER JOIN ] 従業員ごと_目標集計_一時 B
  ON A.従業員コード = B.従業員コード
[ d: LEFT OUTER JOIN ] 従業員ごと_実績集計_一時 C
  ON B.従業員コード = C.従業員コード AND B.KPI コード = C.KPI コード
GROUP BY A.組織コード, B.KPI コード

```

- **[h] (組織ごと_目標実績集計_一時)** : 表 4 の項番 4 における出力表の名前そのものです。データ挿入先テーブルとして **組織ごと_目標実績集計_一時** が入ります。
- **[c] (INNER JOIN)** : 従業員_所属_一時 (集計日時点で在籍・退職していない一般職) と 従業員ごと_目標集計_一時 (年度開始時点で在籍している従業員) を結合します。ビジネスルールである「管理職除外」「年度途中入社除外」「年度途中退職除外」をすべて満たす対象者は、**両方の表に共通して存在する従業員のみ**です。したがって、両者に存在する行だけを残す内部結合 **INNER JOIN** を使用します。
- **[d] (LEFT OUTER JOIN)** : [c] の結合結果 (対象従業員・目標データ) に対して 従業員ごと_実績集計_一時 を結合します。『日別個人実績』の運用ルール上、実績が 1 日も存在しない従業員のレコードは作成されません。ここで内部結合を行うと、実績がない従業員が結合結果から除外され、その従業員が設定した「目標値」まで消えてしまいます。実績のない従業員も目標・対象人数に残すため、左側の表を全件保持する左外側結合 **LEFT OUTER JOIN** を指定します。
- **[i] (COUNT(*))** : 組織・KPI コードごとの「対象従業員数」を算出する集合関数です。GROUP BY 句によってグループ化されているため、グループ内の件数をカウントする **COUNT(*)** (または COUNT(A.従業員コード) など) が入ります。

項番 5 の SQL 文と空欄 [h]

-- 項番 5 : 最終集計リストの作成 (平均値の算出)

```

SELECT A.*, A.目標組織集計/A.対象従業員数, A.実績組織集計/A.対象従業員数
FROM [ h: 組織ごと_目標実績集計_一時 ] A
ORDER BY A.組織コード, A.KPI コード

```

- **項番 5 における [h]**: 項番 4 で集計結果を出力した一時表 **組織ごと_目標実績集計_一時** を入力として参照し、平均値を求めて最終リストを出力します。

設問 2(2) 実績がない場合の挙動と COALESCE 関数の役割 (So What? の深掘り)

下線①に示す事態が発生する条件

設問 2(2)「下線①『実績個人集計が NULL の際』という事態が発生する条件」の解答要求 (40 字以内) は以下の通りです。

- **【解答例】** : 該当従業員の KPI 項目に対する実績データが、1 件も存在しない場合 (35 字)

COALESCE 関数の演算挙動と数理的影響 (0/N = 0 vs NULL/N = NULL)

項番 4 の SQL 文では、実績集計値を求めて合算する際に SUM(COALESCE(C.実績個人集計, 0)) を使用しています。COALESCE(A, B) は、第一引数 A が NULL の場合に第二引数 B (ここでは 0) を返す関数です。

この関数の役割と、もし使用しなかった場合 (単に SUM(C.実績個人集計) とした場合) の SQL 実行上の挙動および数値計算上の悪影響を「So What?」の観点から深掘りします。

【COALESCE 関数を使用した場合 (正しい挙動)】

従業員 X (実績なし: NULL) ——COALESCE——> 0

従業員 Y (実績あり: 100) —————> 100

SUM 結果: 100

項番 5 の演算: $100 / 2$ (対象人数) = 50.0 (正確な実績平均値)

【COALESCE 関数を省略した場合 (不具合の発生)】

全員が実績なしの場合 (全員 NULL) :

SUM(NULL, NULL, ...) ——> NULL

項番 5 の演算: $NULL / N$ (対象人数) ——> NULL (0.0 ではなく未計算となる!)

1. **SQL エンジンにおける演算動作の相違:** LEFT OUTER JOIN により、実績が 0 件の従業員の C.実績個人集計 は NULL として結合されます。もし組織内の全員が期間中に実績ゼロであった場合、SUM(C.実績個人集計) の評価結果は数値の 0 ではなく **NULL** となります。
2. **算術演算 (除算) における比較:** 項番 5 において、組織の実績平均値を求めるために A.実績組織集計 / A.対象従業員数 (A / N) が実行されます。
 - **COALESCE 補正あり:** $0 / N$ の除算が行われ、数値 **0 (0.0)** という正確な平均値が得られます。
 - **COALESCE 補正なし:** $NULL / N$ の除算が行われます。SQL の仕様上、NULL に対するいかなる算術演算の結果も **NULL** と評価されます。
3. **ビジネス評価に与える悪影響:** 成果が上がっていない状態 (実績 0) であるにもかかわらず、システムの出力画面や集計リスト上では「実績 0.0」ではなく「未計算 (NULL / 空白)」と表示されてしまいます。これにより、達成状況の正確な分析や評価が不可能になるという重大なバグにつながります。

したがって、NULL を明確に数値の 0 に変換する COALESCE 関数の適用は、データ集計システムにおける整合性を担保するための不可欠なテクニックです。

次章への接続

SQL の構文とビジネスロジックの密接な関係を理解したところで、最後に本問題を通じて学んだデータベース設計・データ操作の重要概念を整理し、復習チェックリストとしてまとめます。

5. 初心者が押さえるべき復習チェックリストと総括

本問から得られる汎用的知見

令和 5 年度春期応用情報技術者試験 午後問 6 の解説を通じて、実務においても極めて汎用性の高いデ

データベース設計と SQL 操作のパターンを学ぶことができます。

特に、「履歴を保持するデータ構造（開始・終了年月日管理）の読み解き」「業務上の評価対象外ルールを SQL の結合（JOIN）の種類によって実現する手法」「一時表（ワークテーブル）を用いた処理の機能分割」、そして「NULL 値に対する適切なハンドリング（COALESCE 関数）」は、あらゆる業務システム開発に応用できる本質的なスキルです。

重要概念の対比・復習テーブル

概念	役割・機能	本問題における適用と注意点
INNER JOIN (内部結合)	両方の表に存在する（結合条件を満たす）レコードのみを抽出する。	【対象者の厳密な絞り込み】 退職者を除外した表と途中入社者を除外した表を内部結合することで、双方に存在する「評価対象の一般職従業員」のみを過不足なく抽出する。
LEFT OUTER JOIN (左外側結合)	結合相手の表にデータが存在しなくても、左側の表のレコードをすべて残す。	【実績ゼロの従業員の保護】 実績データ（日別個人実績）が存在しない従業員であっても、目標値や対象人数から消えてしまわないよう、左外側結合によって確実に保持する。
主キー (Primary Key) vs 外部キー (Foreign Key)	主キー: 表内の行を一意に識別する属性。 外部キー: 他の表の主キーを参照し、関係性を構築する属性。	【E-R 図の参照整合性】 従業員 表の主キー <u>従業員コード</u> が、月別個人目標 や 日別個人実績 などの子エンティティから外部キーとして参照され、データモデルの整合性を担保する。
NULL と COALESCE 関数	NULL: 値が存在しない状態。 COALESCE: NULL を検知し、指定したデフォルト値に変換する。	【集計値の不具合防止】 実績がない場合の NULL を COALESCE(実績個人集計, 0) によって 0 に変換し、SUM 集計や 0 / N の平均値計算の誤り（NULL 化）を防ぐ。

応用情報技術者試験突破のための 3 つの鉄則

💡 講師のワンポイントアドバイス：鉄則 1 業務ルール（文章）とテーブル結合（SQL）の対応関係を即座に見抜く！

- 「双方に存在するデータのみを対象とする（～かつ～）」という条件は **INNER JOIN** を選択します。
- 「実績がない（データが存在しない）場合も抽出対象に残す」という条件を見たら、即座に **LEFT OUTER JOIN** と **COALESCE**（または **NVL / IFNULL** 等による **NULL 補正**）の組み合わせを連想できるように練習を重ねましょう。

💡 講師のワンポイントアドバイス：鉄則 2 時系列・履歴管理データの構造パターンを押さえる！

- 組織変更や人事異動に対応するデータベースでは、開始年月日 と 終了年月日 を主キーの一部として持つ構造が定石です。
- 「現在時点」のデータを特定する際は、集計年月日 **BETWEEN** 開始年月日 **AND** 終了年月日 のように **BETWEEN** 述語を用いて条件指定するパターンを確実にマスターしてください。

💡 講師のワンポイントアドバイス：鉄則 3 一時表（ワークテーブル）による機能分割の処理フローを追跡する！

- 複雑な問題に直面した際は、いきなり全体の SQL を組もうとせず、各一時表が「何を入力として」「どのような条件で絞り込み・集計を行い」「何を出力しているか」を表形式やフローチャートで整理する習慣をつけましょう。処理を分割して考えることが、正確な解答への最短ルートとなります。