

令和 4 年度春期 応用情報技術者試験 午後問 6「クーポン発行サービス」徹底復習レポート

1. 本問の全体像と学習テーマの整理

受講生のみなさん、こんにちは。データベース設計と排他制御の講義を担当する専任教育エキスパートです。

オンラインサービスにおいて、自治体主導の大型割引キャンペーンや話題商品の販売開始など、特定のタイミングでアクセスが殺到する場面は日常茶飯事です。このときデータベース（RDB）内で適切に排他制御や連番管理が行われていないと、リソースの競合によるレスポンス遅延やタイムアウトが発生し、最悪の場合はシステムダウンやデータの重複といった致命的な障害に繋がります。

本問（令和 4 年度春期 応用情報技術者試験 午後問 6）は、ホテルやレストランの予約サイトを運営する「K 社」が新たに展開するクーポン発行サービスを題材に、業務要件に応じたデータベース構造の設計と、高負荷時におけるアクセス集中に耐えうる排他制御・連番管理方式の選定を迫体験する非常に実戦的な問題です。

まずは、本問の全体像と前提知識を 3 つの視点（出題概要、対象システム、スポンサ制限仕様）から表形式で整理しておきましょう。

出題趣旨とシステム概要

区分	詳細内容
出題趣旨	オンラインサービスにおける特定タイミング（商品・特典の提供開始等）のレスポンス遅延やタイムアウト対策として、業務特性および使用する RDB の設定に応じた適切な排他制御方式・連番管理方式を検討する知識・技能を問う。
対象システム	K 社運営の施設予約サービス（既存システム）および新設するクーポン発行サービス。関係データベース（予約サイトデータベース）によって一括管理される。
提携スポンサ	施設、都道府県、市区町村の 3 種類。K 社は提携スポンサと合意した割引金額・枚数でクーポンを発行する。

提携スポンサによる制限仕様の構造（表 1 の解説）

提携スポンサ	クーポンの獲得制限（獲得時チェック）	クーポンを利用した予約制限（予約時チェック）
施設	同一会員による同一クーポンの獲得枚数を 1 枚に制限（同一会員 1 枚限りの獲得制限）。	・ 設定した施設だけを予約可能にする。 ・ 利用金額が設定金額以上の予約だけに可能にする。
都道府県・市区町村	設定地区に居住する会員だけが獲得可能にする。	設定地区にある施設だけを予約可能にする。

クーポン発行サービスの業務特性（表 2 の解説）

利用局面	業務要件の概要
クーポンの	発行予定および発行中クーポンの情報は、会員向けメール配信で周知され、施設予約サー

照会	ビス上で検索・照会できる。
クーポンの獲得	・会員が獲得操作を行うことで利用可能となる。 ・獲得可能期間が定められている。 ・発行枚数が上限に達すると、以降の獲得はできない。
クーポンの利用	・獲得したクーポンは施設予約時の料金割引に利用できる。 ・1枚のクーポンは一つの予約だけに利用できる。 ・予約キャンセル時は、そのクーポンを別の予約に再利用可能。 ・利用期間を過ぎたクーポンは無効となる。

実務と試験における本問の価値

実務において、アプリ側でどんなに美しく画面を組んでも、データベース層の排他制御設計を誤れば「同じクーポンが複数人に二重発行された」「大量アクセスでロックが解放されず画面が固まった」といった大惨事を引き起こします。本問は、単なる SQL の文法テストではなく、「ロック解放待ちを取るか、主キー制約違反によるユーザーのやり直し（再オペレーション）を許容するか」という**実務直結のトレードオフ判断**を学べる最高の教材です。

応用情報技術者試験においても、E-R 図の読み取り（設問 1）、DDL・DML の記述（設問 1・3）、排他制御と SQL 構文（設問 2・3）、CRUD 図への変更波及（設問 4）と、DB 分野の必須テクニックが網羅されています。

それでは、すべての基本となる「E-R 図とテーブル構造の完全解説」から一歩ずつ紐解いていきましょう。

2. データベース設計と E-R 図の完全解説（設問 1(1)攻略）

データベース設計では、現実に存在する業務データ（実体＝エンティティ）同士がどのような関係（リレーション）で結びついているかを正しく把握することが最優先課題となります。

設問の空欄穴埋めに入る前に、本問の予約サイトデータベースに登場する 6 つのエンティティ全体の結びつきを完璧に整理しておきましょう。

全体像：6 つのエンティティが担う役割と結合ルール

図 1 の E-R 図には、6 つの主要なエンティティが存在します。これらは次のような論理的な関係性（1 対多、または 1 対 1）で結ばれています。

1. 「会員」と「予約」（1 対多） 1 人の会員は、何度も施設を予約できます。したがって「会員(1) ——> 予約(多)」の 1 対多の関係となり、「予約」テーブルには参照用の外部キーとして 会員コード が配置されます。
2. 「会員」と「クーポン明細」（1 対多） 1 人の会員は、様々なクーポンを複数獲得できます。したがって「会員(1) ——> クーポン明細(多)」の関係となり、「クーポン明細」テーブルにも外部キー 獲得会員コード が入ります。
3. 「施設」と「利用プラン」（1 対多） 1 つの施設（ホテルや旅館）は、「1 泊朝食付きプラン」「素泊まりプラン」など複数の利用プランを提供します。したがって「施設(1) ——> 利用プラン(多)」の関係です。
4. 「利用プラン」と「予約」（1 対多） 1 つの利用プランに対して、多くの会員から予約が入りま

す。したがって「利用プラン(1) → 予約(多)」の関係が成立します。

5. 「クーポン管理」と「クーポン明細」(1 対多) 「クーポン管理」はクーポンの種別(例: ○○市限定 1,000 円割引クーポン)を 1 レコードで保持します。会員が獲得ボタンを押すと、その種別から「1 枚単位の発行明細」が作られます。したがって「クーポン管理(1) → クーポン明細(多)」の関係です。
6. 「クーポン明細」と「予約」(0 対 1 / 1 対 1) 獲得された 1 枚のクーポンは、ある 1 つの予約に対して適用されるか、未アクセプトのまま保持されます。予約側から見ると「クーポンを使わない予約」も存在するため、破線外部キーとして クーポンコード と クーポン発行連番 を受け取ります。

記法ルールと表記の確認

- 実線下線 (属性名) : 主キー (Primary Key)。テーブル内でレコードを一意に識別する列。
- 破線下線 (属性名) : 外部キー (Foreign Key)。他テーブルの主キーを参照する列。

設問 1(1)の解答解説

以上の構造を踏まえて、空欄 a, b, c を導き出します。

【予約】		【利用プラン】	
+-----+		+-----+	
<u>予約コード</u>	<-----+	<u>施設コード</u>	
<ins>会員コード</ins>	<-----+	<u>プランコード</u>	
<ins>a: 施設コード</ins>	(1 対多)	最小利用人数	
<ins>b: プランコード</ins>		最大利用人数	
+-----+		+-----+	
		↑ c: 1 対多 (上向き矢印)	
		【施設】	
		+-----+	
		<u>施設コード</u>	
		+-----+	

空欄 a, b (「予約」テーブルに必要な外部キー)

- 解説ロジック:
 1. E-R 図を見ると、「利用プラン」から「予約」へ向かって矢印 (1 対多) が伸びています。
 2. 親テーブルである「利用プラン」の主キーを確認すると、施設コード と プランコード の 2 つの列を組み合わせた複合主キー (実線下線) になっています。
 3. 子テーブルである「予約」が「利用プラン」を参照するためには、親の複合主キーを構成する両方の列を外部キー (破線下線) として抱え込む必要があります。
- 解答例:

- a: 施設コード (または プランコード)
- b: プランコード (または 施設コード) ※順不同

空欄 c (「施設」と「利用プラン」間のリレーション)

• 解説ロジック:

1. 「施設」の主キーは 施設コード 単一です。「利用プラン」の主キーは (施設コード, プランコード) です。
2. 前述の通り、1 つの施設には複数の利用プランが紐付きます (1 対多)。
3. 図 1 の凡例より、1 対多のリレーションは「親 (1) から子 (多) へ向かう矢印 →」で表現します。
4. したがって、「施設 (下)」から「利用プラン (上)」に向かって伸びる上向きの矢印 ↑ を入れるのが正解です。

• 解答: ↑

参照整合性が確保された綺麗なデータベースモデルが描けましたね。次は、このテーブル構造の上に「ビジネスルール (業務要件)」を制約として埋め込む技術を見ていきましょう。

3. 業務要件を実装する SQL と UNIQUE 制約 (設問 1(2)・(3)攻略)

マスター講師として、実務でよく見かける落とし穴をお伝えしておきましょう。それは「アプリ側 (Java や Python 等) でチェック処理を書いているから、DB 側は適当でいいや」という思い込みです。排他制御の抜け漏れやバグにより、アプリのチェックを潜り抜けた不正データが侵入した際、**最後の砦として不整合を物理的にブロックするのがデータベースの「制約 (Constraint)」**です。本セクションでは、DDL による制約の追加と、制約が防いでいる業務上の意味を解説します。

設問 1(2)の解答解説: 既存テーブルへの制約追加 (DDL)

図 2 は、「クーポン明細」テーブルに“同一会員 1 枚限りの獲得制限”を実装するための SQL 文です。

[d] クーポン明細 ADD CONSTRAINT クーポン明細_IX1

UNIQUE(クーポンコード, 獲得会員コード, 獲得制限_1 枚限り)

• 解説ロジック:

- テーブルを新規作成する場合は CREATE TABLE ですが、問題文には「クーポン明細 テーブルに対して、UNIQUE 制約を付けた」とあります。すでに存在するテーブルの構造・定義を後から変更する DDL (データ定義言語) コマンドは **ALTER TABLE** です。
- 構文ルール: ALTER TABLE テーブル名 ADD CONSTRAINT 制約名 UNIQUE (列名 1, 列名 2, ...)

- 採点講評からの重要な学び: IPA の採点講評によると、設問 1(2)の正答率は低かったと報告されています。ALTER TABLE は列の追加 (ADD) や変更 (MODIFY / ALTER)、制約の追加 (ADD CONSTRAINT) など、実務でもシステム運用変更時に必須となる重要構文です。確実に暗記しておきましょう。

• 解答: ALTER TABLE

設問 1(3)の解答解説: UNIQUE 制約が実現する業務要件

下線①の解説と業務要件の対応

本文中の下線①「予約テーブルの“クーポンコード”，“クーポン発行連番”に対しても，UNIQUE 制約を付けた。」が実現する業務要件を答えます。

- **解説ロジック:**

1. 「クーポン明細」テーブルにおいて、発行されたクーポンは（クーポンコード，クーポン発行連番）のペアで世界に1枚だけの存在として一意識別されます。
2. 「予約」テーブルの列に（クーポンコード，クーポン発行連番）に対する UNIQUE（一意性）制約を付与すると、「予約」テーブル全体の中で、同じクーポンのペアが2つ以上の予約レコードに登録されることを物理的に禁止できます。
3. これは、問題本文の表2「クーポンの利用」に記述されている業務要件**「1枚のクーポンは一つの予約だけに利用できる。」**をデータベースレベルで厳格に担保したものです。

- **採点講評のインサイト:** 採点講評によると、設問1(3)の正答率は平均的であり、制約と業務要件の関係は正しく読み取れていた受験者が多かったとされています。確実に得点すべきポイントです。

- **解答（指定文字数：30字以内）:** 1枚のクーポンは一つの予約だけに利用できる。(22文字)

ビジネスルールをDBの制約でガードできたら、いよいよ本問のハイライトである「アクセス集中時の排他制御と連番管理」の深掘りに進みましょう！

4. 大規模アクセスに耐える連番管理方式の徹底比較（設問2・設問3攻略）

自治体が発行する高額割引クーポンなどでは、受付開始の瞬間に数万リクエストが同一データベースへ押し寄せます。このとき、クーポンに「1, 2, 3…」と抜けや重複のない連番をいかに高速かつ正確に振るかが設計の腕の見せ所です。

ここでは、L主任が考案した「ロックなし方式」と、M課長が提案した「ロックあり方式」のメカニズムを比較検証します。

2つの連番管理方式の比較表

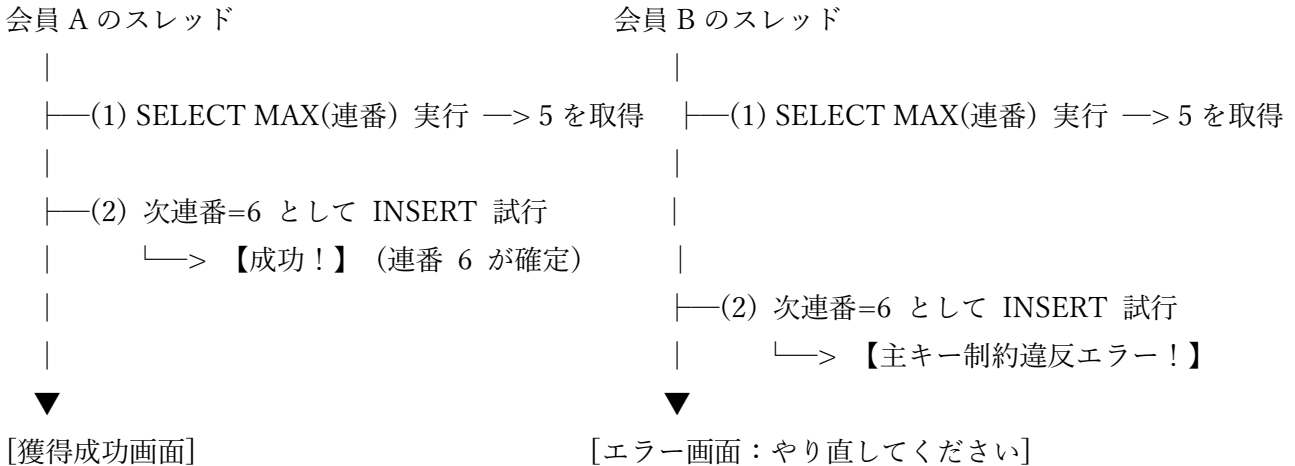
評価項目	ロックなし方式（図4）	ロックあり方式（図5）
制御メカニズム	排他ロックをかけずに MAX 値を参照し、INSERT 時に主キー制約違反（重複）で弾く。	UPDATE 文で「クーポン管理」の行を排他ロックし、連番を順次加算して INSERT する。
ロック解放待ち	発生しない（参照がバッテイングしない）	発生する（同一行の更新で他スレッドが待たされる）
主キー制約違反による再オペレーション	発生する（同時に同じ連番を生成しようとして落とされるため）	発生しない（ロックにより順番にカウントアップされるため）
発行上限到達後の動作	副問合せの SELECT 結果が NULL となり、クーポン発行連番に NULL を追	UPDATE の条件に合致せず更新がスキップされ、古い発行済枚数（＝発行上限枚数）のまま

	加しようとして主キー制約違反となる。	INSERT しようとして主キー制約違反となる。
--	--------------------	--------------------------

なぜ「再オペレーション」が起きるのか？ Concurrent Timeline 解析

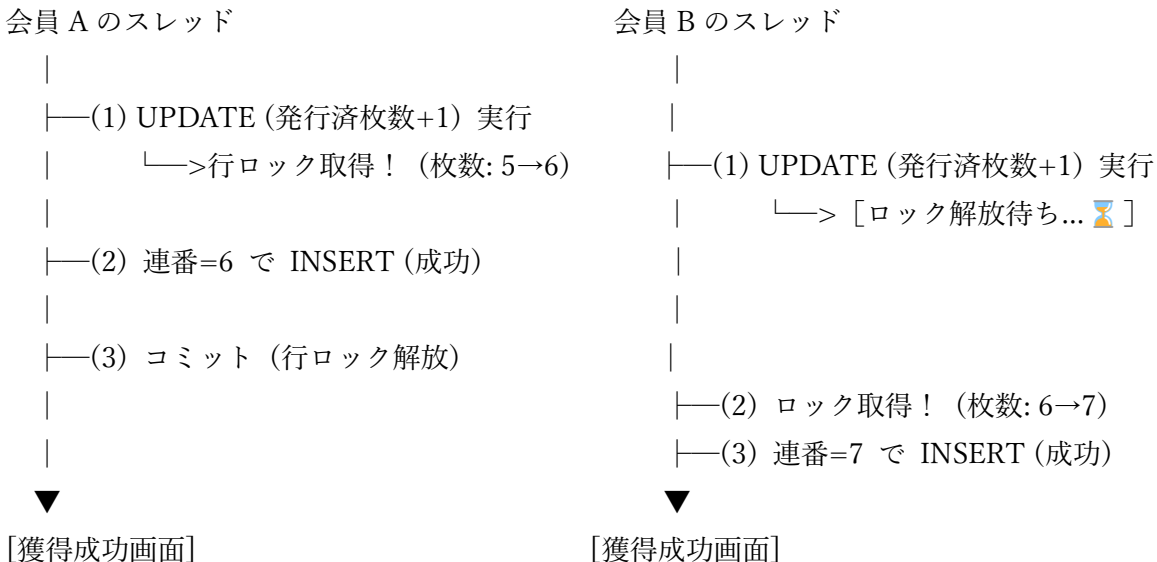
言葉の理解を深めるために、2 人のユーザー（会員 A・会員 B）が完全に同じタイミングでクーポン獲得ボタンを押した時のスレッド動作（タイムライン）を追ってみましょう。

【ロックなし方式のタイムライン】



ロックなし方式では、ロック解放待ちがないため処理自体は速いものの、タッチの差で負けた会員 B に画面上で「エラーが発生しました。もう一度やり直してください（再オペレーション）」を強いることになり、UX（ユーザー体験）が著しく低下します。

【ロックあり方式のタイムライン】



ロックあり方式では、会員 B が数ミリ秒待たされる（ロック解放待ち）ものの、エラーになることなく確実・順番にクーポンが発行されます。

設問 2 の解答解説（ロックなし方式の SQL 解析）

図 4 の SQL 文における空欄 e の穴埋めです。

INSERT INTO クーポン明細 (クーポンコード, クーポン発行連番, 獲得会員コード, 獲得制限_1 枚限り)

WITH 発行済枚数取得 AS (SELECT COALESCE(MAX([e]), 0) AS 発行済枚数

FROM クーポン明細 WHERE クーポンコード = :クーポンコード)

SELECT :クーポンコード,

(SELECT 発行済枚数 + 1 FROM 発行済枚数取得 WHERE (SELECT 発行済枚数 FROM 発行済枚数取得) < 発行上限枚数),

:会員コード, 獲得制限_1 枚限り

FROM クーポン管理 WHERE クーポンコード = :クーポンコード

- **解説ロジック:**

1. WITH 発行済枚数取得 AS (...) 内で、現在「クーポン明細」テーブルに存在する該当クーポンの「最大連番」を探しています。

2. 対象となる列名は **クーポン発行連番** です。

- **マスター講師の深掘り解説 (COALESCE 関数のエッジケース動作) :**

- 問題文にある通り、COALESCE(A, B) は「A が NULL でなければ A を、NULL ならば B を返す」関数です。

- クーポンがまだ **1 枚も発行されていない初期状態**では、MAX(クーポン発行連番) はデータが存在しないため **NULL** を返します。

- もし COALESCE がないと、NULL + 1 = NULL となり、1 枚目の発行から失敗してしまいます。

- そこで COALESCE(MAX(クーポン発行連番), 0) と記述することで、NULL を **0** に変換し、0 + 1 = 1 として記念すべき 1 枚目の連番「1」を安全に生成しているのです。

- **副問合せによる上限判定の仕組み:** WHERE (SELECT 発行済枚数 FROM 発行済枚数取得) < 発行上限枚数 の条件により、発行済枚数が上限に達している場合は SELECT 結果が NULL になります。その結果、クーポン発行連番 に NULL を挿入しようとして主キー制約違反 (NOT NULL 制約・主キー制約) が発生し、上限オーバーを未然に防ぎます。

- **解答: クーポン発行連番**

設問 3 の解答解説 (ロックあり方式の SQL 解析)

図 5 は、「クーポン管理」テーブルの 発行済枚数 列 (初期値 0) をカウントアップするロックあり方式です。

UPDATE クーポン管理 [f]

WHERE クーポンコード = :クーポンコード AND 発行済枚数 < [g];

INSERT INTO クーポン明細 (クーポンコード, クーポン発行連番, 獲得会員コード, 獲得制限_1 枚限り)

SELECT :クーポンコード, 発行済枚数, :会員コード, 獲得制限_1 枚限り

FROM クーポン管理 WHERE クーポンコード = :クーポンコード ;

空欄 f (カウントアップ処理)

- 解説ロジック:
 - UPDATE 文の基本構文は UPDATE テーブル名 SET 列名 = 式 WHERE 条件 です。
 - 発行済枚数 の値を 1 だけ加算 (カウントアップ) するため、SET 発行済枚数 = 発行済枚数 + 1 と記述します。
- 採点講評からの重要な学び: 採点講評によると、空欄 f (UPDATE 文の記述) は正答率が低かったと報告されています。SET 列名 = 列名 + 1 という加算構文は実務のカウンター処理でも高頻度で登場するため、記述漏れのないよう徹底的に頭に叩き込みましょう。
- 解答: SET 発行済枚数 = 発行済枚数 + 1

空欄 g (発行上限のチェック)

- 解説ロジック:
 - WHERE 句の条件 発行済枚数 < [g] は、発行済枚数が「上限枚数」に達していない場合のみ UPDATE を実行させるガード条件です。
 - 比較対象となる列名は「クーポン管理」テーブルの 発行上限枚数 です。
- 解答: 発行上限枚数

K 社が「ロックあり方式」を採用した決定性理由

高負荷試験を実施した結果、ピーク時のトラフィックにおいても「ロックなし方式」「ロックあり方式」の双方でレスポンスおよびスループットが規定値内に収まることが実証されました。

性能差が問題にならないのであれば、ユーザーにエラー画面を見せてやり直させる「ロックなし方式」を選ぶ理由はありません。ユーザー体験 (UX) を最優先し、再オペレーションが発生しない**「ロックあり方式」の採用が最適解**と判断されました。

システム方式が決まったところで、この変更が全体設計 (CRUD 図) にどう波及するかを最終確認しましょう。

5. システム処理フローと CRUD 図の変更分析 (設問 4 攻略)

システムの内部構造 (連番管理方式) を変更した際、データベースの基本操作である C (Create : 追加)、R (Read : 参照)、U (Update : 更新)、D (Delete : 削除) の役割分担がどのように変わるかを追跡することは、仕様変更時の影響調査において極めて重要です。

図 3 の CRUD 図における変更点 (下線②、下線③) を、対比マトリクスで分かりやすく整理しました。

連番管理方式の変更に伴う CRUD 図 (図 3) の対比マトリクス

処理名	対象	変更前後操作		操作が変わった理由・バックグラウンド
		変更前操作 (ロックなし方式)	変更後操作 (ロックあり方式)	
クーポン獲得	クーポン	R (参照の	RU (参照 +	ロックあり方式 (図 5) では、クーポン獲得時に「クーポン管理」の 発行済枚数 列をカウントアップ (UPDATE) する処理が追加

(下線 ②)	管理	み)	更新)	されたため。
クーポン獲得 (下線 ③)	クーポン 明細	CR (追加 + 参照)	C (追加の み)	変更前 (図 4) は「クーポン明細」の最大連番を SELECT MAX (参照: R) していたが、変更後は「クーポン管理」の値を使うた め参照が不要となり、INSERT (追加: C) のみとなったため。

設問 4 の解答解説

対比マトリクスの内容をそのまま解答形式にまとめます。

下線② (「クーポン獲得」における「クーポン管理」テーブルの操作)

- **解説:** 変更前の参照 (R) に加え、発行済枚数 のカウントアップ (UPDATE) が発生するた
め、U が追加されて RU となります。
- **解答:** RU

下線③ (「クーポン獲得」における「クーポン明細」テーブルの操作)

- **解説:** 変更前に必要だった MAX(クーポン発行連番) の参照 (R) が不要となり、INSERT に
よる新規作成 (C) のみとなるため C となります。
- **解答:** C

6. 初心者のための試験対策・復習チェックリスト

本問は、E-R 図の参照整合性から始まり、DDL・DML の基本文法、排他制御のタイムライン、そして
CRUD 図の更新まで、情報処理技術者試験 (AP/DB) の真髄が詰まった傑作問題でした。

最後に、マスター講師からみなさんへ、解きっぱなしにしないための「復習チェックリスト」を贈りま
す。

午後 DB 問題 攻略のための必須チェックリスト

- ☐ E-R 図と複合主キーの追跡
 - 親テーブルの主キーが「複合主キー (例: 施設コード+プランコード)」である場合、子
テーブル側にも両方の列が外部キーとして伝播することを一瞬で判断できるか?
 - 1 対多のリレーション矢印の向き (1 の親から多の子へ向けて描く) を正しく描ける
か?
- ☐ DDL (ALTER TABLE) と DML (UPDATE) の完勝
 - 既存テーブルへの制約追加構文 ALTER TABLE テーブル名 ADD CONSTRAINT ... を
ノーヒントで書けるか? (※採点講評で正答率低下と指摘!)
 - カウントアップの加算構文 UPDATE テーブル名 SET 列名 = 列名 + 1 WHERE ... を
ミスなく記述できるか? (※採点講評で正答率低下と指摘!)
- ☐ UNIQUE 制約と業務ルールのリンク
 - テーブルの列に UNIQUE 制約を貼ることで、「1 つのリソース (クーポン) は 1 つの処
理 (予約) だけに利用できる」という業務ルールを物理担保できるイメージが湧いてい
るか?
- ☐ 排他制御 (ロック) とユーザー体験のトレードオフ理解

- **ロックなし方式:** ロック解放待ちゼロで高速だが、競合時に主キー制約違反が起き、ユーザーに再オペレーションを強いる。
- **ロックあり方式:** 行ロックで後続処理を待たせる（ロック解放待ち発生）が、順次処理によりエラーなく確実に完了する。
- **[] 採点講評の傾向把握**
 - 問6全体としては「平均的な正答率」であったが、基本文法（ALTER TABLE や UPDATE SET）で差がついていることを意識できているか？

試験本番の午後問題は長文ですが、落ち着いて「業務要件（文章・表）」と「データベース構造（テーブル・SQL）」の対応関係を結びつけられれば、必ず解は導き出せます。本レポートを何度も見返し、自身の血肉としていってください。応援しています！