問1　コード決済業者の決済業務及びポイント管理業務におけるデータベースの実装・運用に関する次の記述を読んで，設問に答えよ。

A 社は，バーコードを用いたキャッシュレスの決済サービス，及び商品購入などに応じて一定のポイントを付与するポイントサービスを提供している。A 社では，ポイントサービス利用件数の急増を受けて，システムの変更を検討している。

〔業務・現行システムの概要〕

1. サービスの概要

(1) 決済サービス

① 決済サービスを利用する人（以下，決済会員という）は，決済会員登録を行い，金融機関などから一定の金額を決済サービスに入金した上で，物品購入などの支払に充てる。入金及び支払に伴う残高を決済残高という。

② 決済サービスに加盟する事業者（以下，決済加盟店という）は，決済サービスへの加盟申請，審査を経て決済加盟店登録を受ける。

③ 決済会員は，スマートフォンなどの携帯端末，決済加盟店の店舗に設置された PC，POS などの端末（これらをまとめて，以下，端末という）を使用して，バーコードを読み取ることで支払を行う。同じ決済会員が同時に複数の支払を行うことはない。

④ 決済は，決済会員の決済残高から支払金額を差し引くことで完了する。

⑤ A 社は，月に 1 回まとめて決済金額を決済加盟店に支払う。

(2) ポイントサービス

① ポイントサービスを利用する人（以下，ポイント会員という）は，ポイント会員登録を行ってポイントサービスを利用する。

② ポイントサービスに加盟する事業者（以下，ポイント加盟店という）は，ポイントサービスへの加盟申請，審査を経てポイント加盟店登録を受ける。

③ ポイント会員は，ポイント加盟店での支払，くじ引き，イベント参加などによってポイントを獲得する。

④ 獲得したポイントには，有効期限の年月が決まっている。獲得したポイントはポイント残高として貯めておき，支払に充てることができる。

⑤　獲得したポイントは，すぐにポイント残高に反映されることもあれば，翌日以降になる場合もある。

(3)　サービス間の連携

決済金額の一部をポイントで支払い，残りを決済残高から支払う場合など，両方のサービスを連携した決済を行うことがある。

2.　システム構成

現行システムのシステム構成を図1に示す。決済サービス，ポイントサービスそれぞれに専用のサーバ及びRDBMSを用いている。サーバ上ではサービスに用いるアプリケーションプログラム（以下，APという）が稼働している。両サービスを連携した処理は，決済APを介して決済DB，ポイントDBの両方にアクセスする。

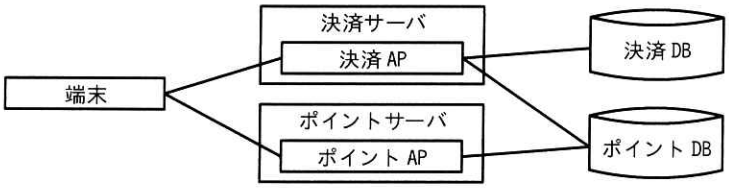

図1　現行システムのシステム構成

3.　テーブル構造

決済サービス，ポイントサービスのテーブル構造をそれぞれ図2，3に，主な属性の意味・制約を表1に示す。なお，属性名の#は番号，Cはコード，Fはフラグを略した記号である。

決済加盟店（決済加盟店#，ポイント加盟店#，名称，支払先口座，…）
店舗（決済加盟店#，店舗#，決済方式，所在地，…）
決済会員（決済会員#，ポイント会員#，表示名，最終入金#，認証，…）
決済残高（決済会員#，残高）
入金履歴（決済会員#，入金#，入金日，入金金額，入金方法）
決済履歴（履歴#，決済加盟店#，店舗#，決済会員#，決済日，決済金額，使用ポイント数）
加盟店支払（決済加盟店#，支払年月，締め日，支払予定日，支払予定金額）

図2　決済サービスのテーブル構造（一部省略）

ポイント会員ランク（ポイント会員ランクC，名称，達成ポイント数）
ポイント獲得方法（ポイント獲得方法C，名称）
ポイント付与率（ポイント獲得方法C，ポイント会員ランクC，付与倍率）
ポイント加盟店（ポイント加盟店#，名称，所在地，…）
ポイント会員（ポイント会員#，ポイント会員ランクC，氏名，生年月日，住所，認証，…）
ポイント残高（ポイント会員#，有効期限年月，残数，有効F）
会員ランク履歴（ポイント会員#，変更日，ポイント会員ランクC）
ポイント獲得履歴（履歴#，ポイント会員#，ポイント加盟店#，ポイント獲得方法C，獲得日，
有効期限年月，獲得ポイント数，残高反映F）
ポイント使用履歴（履歴#，ポイント会員#，ポイント加盟店#，使用日，使用ポイント数）

図3　ポイントサービスのテーブル構造（一部省略）

表1　主な属性の意味・制約

属性名	意味・制約
決済方式	バーコードの読取りを利用者の端末，店舗の端末のどちらで行うかを表す区分
入金#	決済会員ごとに決済残高への入金の都度払い出す連番
入金方法	入金時に使用した銀行口座，クレジットカードなどの情報
ポイント獲得方法C	ポイント加盟店での支払，くじ引き，イベント参加などポイント獲得の方法を表すコード
有効期限年月，残数	ポイントが有効な期間は，獲得方法によって異なる。“ポイント獲得履歴”テーブルには獲得したポイントの有効期限年月を設定し，“ポイント残高”テーブルには有効期限年月ごとにポイントの残数を記録する。
履歴#	“決済履歴”，“ポイント獲得履歴”，“ポイント使用履歴”テーブルの履歴#には，それぞれに一意な番号を自動的に設定する。
有効F	有効期限年月が現在の年月以降の場合はTRUE，それよりも前のものはFALSEを設定する。
残高反映F	獲得ポイント数を“ポイント残高”テーブルの残数に加算済みの場合はTRUE，未加算の場合はFALSEを設定する。

4.　主な処理

各サービスの主な処理の内容を表2に，表2中のS1，S2，P2，P3の処理概要とSQL文を表3～6に示す。各処理は一つ以上のステップから成る。表2～6中の二重引用符で囲んだ名前は，図2，3中のテーブル名である。

表 2　主な処理の内容

サービス	処理 ID	名称	内容
決済サービス	S1	決済残高入金処理	決済会員の指示によって金融機関からの入金実行を記録し，“決済残高”の残高に反映する。
	S2	決済残高消費処理	決済会員の決済指示を受けて，“決済残高”の残高を消費して，内容を“決済履歴”に記録する。
ポイントサービス	P1	ポイント獲得処理	ポイント会員がポイントを獲得する都度，獲得したポイント数を“ポイント獲得履歴”に記録する。
	P2	獲得ポイント反映処理	任意のタイミングで，ポイント会員のポイント獲得履歴を“ポイント残高”の有効期限年月該当行に反映する。
	P3	ポイント消費処理	ポイント使用指示を受けて，“ポイント残高”の残数を消費して，内容を“ポイント使用履歴”に記録する。

表 3　S1 の処理概要と SQL 文

ステップ	処理概要（上段）と SQL 文（下段）
S1-1	決済会員の“決済残高”の残高に入金金額を加算して更新する。 UPDATE 決済残高 SET 残高 = 残高 + :hv3 WHERE 決済会員# = :hv1
S1-2	“決済会員”の最終入金#を更新する。 UPDATE 決済会員 SET 最終入金# = 最終入金# + 1 WHERE 決済会員# = :hv1
S1-3	“入金履歴”の入金#に更新後の最終入金#を設定して行を追加する。 INSERT INTO 入金履歴（決済会員#, 入金#, 入金日, 入金金額, 入金方法） SELECT 決済会員#, 最終入金#, :hv2, :hv3, :hv4 FROM 決済会員 WHERE 決済会員# = :hv1

注記　ホスト変数 hv1～hv4 には，決済会員#，入金日，入金金額，入金方法をそれぞれ設定する。

表 4　S2 の処理概要と SQL 文

ステップ	処理概要（上段）と SQL 文（下段）
S2-1	決済会員の“決済残高”の残高が決済金額以上であることを確認し，残高が不足していれば，戻り値に残高不足エラーを設定して処理を終了する。 SELECT 残高 FROM 決済残高 WHERE 決済会員# = :hv1 AND 残高 >= :hv5 FOR UPDATE
S2-2	決済会員の“決済残高”の残高から決済金額を減算して更新する。 UPDATE 決済残高 SET 残高 = 残高 - :hv5 WHERE 決済会員# = :hv1
S2-3	“決済履歴”の各列に対応するホスト変数の値を設定して行を追加する。 INSERT INTO 決済履歴（決済加盟店#, 店舗#, 決済会員#, 決済日, 決済金額, 使用ポイント数）VALUES(:hv2, :hv3, :hv1, :hv4, :hv5, :hv6)

注記　ホスト変数 hv1～hv6 には，決済会員#，決済加盟店#，店舗#，決済日，決済金額，使用ポイント数をそれぞれ設定する。

表5　P2の処理概要とSQL文（未完成）

ステップ	処理概要（上段）とSQL文（下段）
P2-1	指定されたポイント会員#について“ポイント獲得履歴”からポイント残高未反映の行の獲得ポイント数を有効期限年月ごとにまとめて，“ポイント残高”の該当行があれば残数に獲得ポイント数を加算して更新し，行がなければ残数に獲得ポイント数を設定して行を追加する。 MERGE INTO ポイント残高 A USING (SELECT ポイント会員#, 有効期限年月, [a] AS 加算ポイント数 FROM ポイント獲得履歴 WHERE ポイント会員# = :hv1 AND [b] GROUP BY ポイント会員#, 有効期限年月) B ON (A.ポイント会員# = B.ポイント会員# AND A.有効期限年月 = B.有効期限年月) WHEN MATCHED THEN UPDATE SET [c] WHEN NOT MATCHED THEN INSERT (ポイント会員#, 有効期限年月, 残数, 有効F) VALUES([d] , TRUE)
P2-2	“ポイント獲得履歴”の該当行の残高反映Fの値を更新する。 UPDATE ポイント獲得履歴 SET 残高反映F = TRUE WHERE ポイント会員# = :hv1 AND 残高反映F IS FALSE

注記　ホスト変数hv1にはポイント会員#を設定する。

表6　P3の処理概要とSQL文

ステップ	処理概要（上段）とSQL文（下段）
P3-1	指定されたポイント会員#について“ポイント残高”から有効期限内の残数を集計して求めた使用可能ポイント数が使用ポイント数に満たない場合は，戻り値にポイント不足エラーを設定して処理を終了する。 SELECT SUM(残数) AS 使用可能ポイント数 FROM ポイント残高 WHERE ポイント会員# = :hv1 AND 有効F IS TRUE
P3-2	①のSQL文に基づくカーソルによって，“ポイント残高”からポイントが有効な行を有効期限年月の昇順に読み込み，行ごとに残数を使用ポイント数に合わせて消し込む。ホスト変数lv1に消し込むポイント数（以下，消込ポイント数という）を，lv2に有効期限年月を設定して②の更新を行い，使用ポイント数を全て消し込むまで繰り返す。 ① SELECT 有効期限年月,残数 FROM ポイント残高 WHERE ポイント会員# = :hv1 AND 有効F IS TRUE ORDER BY 有効期限年月 ASC ② UPDATE ポイント残高 SET 残数 = 残数 - :lv1 WHERE ポイント会員# = :hv1 AND 有効期限年月 = :lv2
P3-3	“ポイント使用履歴”の各列に対応するホスト変数の値を設定して行を追加する。 INSERT INTO ポイント使用履歴（ポイント会員#, ポイント加盟店#, 使用日, 使用ポイント数）VALUES(:hv1, :hv2, :hv3, :hv4)

注記　ホスト変数hv1～hv4には，ポイント会員#，ポイント加盟店#，使用日，使用ポイント数をそれぞれ設定する。

5.　現行システムで使用するRDBMSの排他制御機能

(1)　トランザクションのISOLATIONレベルごとの排他制御は，次のとおりである。

① READ COMMITTED では，行の参照時に対象行の共有ロックを取得し，参照終了時に解放する。行の更新時に対象行の専有ロックを取得し，トランザクション終了時に解放する。

② REPEATABLE READ では，行の参照時に対象行の共有ロックを，行の更新時に対象行の専有ロックを取得し，トランザクション終了時に解放する。

③ SERIALIZABLE は，行の参照時又は更新時に表単位の専有ロックを取得し，トランザクション終了時に解放する。

(2) READ COMMITTED 及び REPEATABLE READ では，SELECT 文に FOR UPDATE 句を指定すると，行の参照時に対象行の専有ロックを取得し，トランザクション終了時に解放する。

〔現行システムの決済処理〕

決済処理では，表 2 中の処理 S2 及び P3 を一つのトランザクションとして実行するために，2 相コミットプロトコルを用いている。

1. 決済処理の流れ

決済処理の正常実行時のシーケンス図を図 4 に示す。図中の①～⑬は実行の順番を，時点 A，B は実行の流れにおける特定の時点を表している。

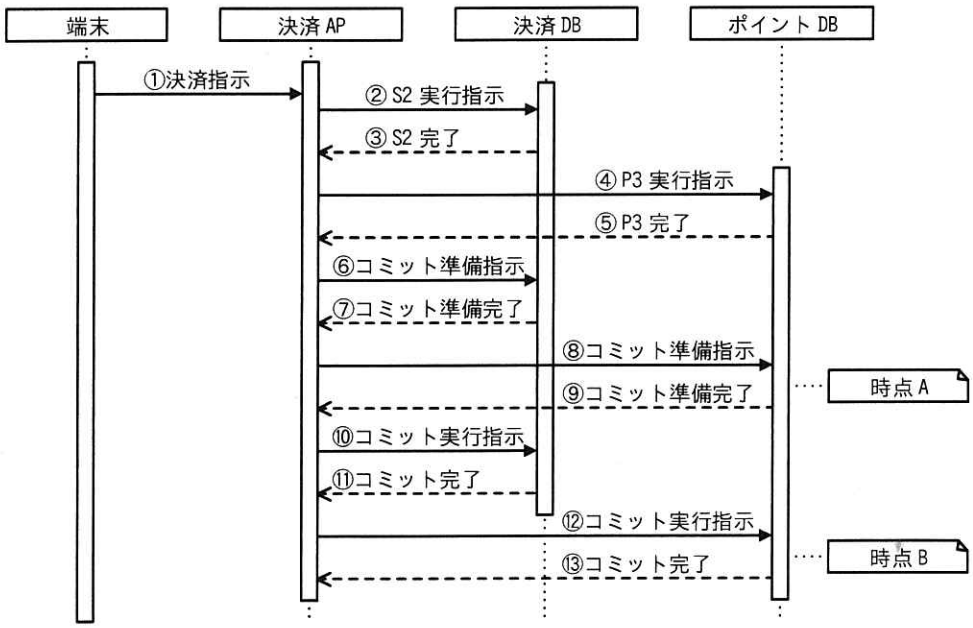

図 4　決済処理の正常実行時のシーケンス図（未完成）

2. 障害からの回復

決済処理の実行中に障害によってポイント DB が異常終了する状況を想定した障害・回復の事例を表 7 にまとめた。表中の時点 A，B は，図 4 中の時点 A，B に対応している。

表 7　決済処理中の障害・回復の事例（未完成）

事例	内容
事例 1	決済処理中の時点 A でポイント DB が障害によって異常終了した。障害除去後に再始動したところ，ポイント DB にはコミット準備のログがあり，トランザクションは未コミットの状態となった。決済 AP は，ポイント DB に状態を問い合わせ，決済 DB 及びポイント DB に［　e　］を行った。
事例 2	決済処理中の時点 A でポイント DB が障害によって異常終了した。障害除去後に再始動したところ，ポイント DB にはコミット準備のログがなく，トランザクションはロールバックされた。決済 AP は，ポイント DB に状態を問い合わせ，決済 DB に［　f　］を行った。
事例 3	決済処理中の時点 B でポイント DB が障害によって異常終了した。障害除去後に再始動したところ，ポイント DB はロールフォワードによってデータベースを回復した。決済 AP は，ポイント DB に状態を問い合わせ，［　g　］した。
事例 4	決済処理中の時点 A でポイント DB が障害によって異常終了した。㋐復旧までの時間が長くなることで，業務上の不都合が生じたので，決済 DB において仕掛り中のトランザクションをヒューリスティックな決定によって全てコミットした。障害除去後の再始動において，ポイント DB は未コミットのトランザクションをロールバックした。

〔システム変更案の検討〕

ポイントサービスでは，EC サイトを運営する加盟店の増加によって，サービス利用件数が急増し，ポイント DB の応答性能，可用性の低下が問題になっている。その対策として，次のシステム変更案を検討している。

1. システムの構成変更

検討中のシステム構成を図 5 に示す。業務機能別に会員サービス，決済サービス，ポイントサービスに分け，サービスごとに専用のサーバとデータベースを配置する。各サービスには独立した疎結合の AP 及びデータベースを用いて，サービス内部の実装は隠蔽する。サービスに連携するアプリケーションプログラミングインタフェース（以下，API という）だけを公開し，API を呼び出すごとに単一のトランザクションとして処理する。端末からのアクセス時には，会員サービスを入り口にして他のサービスに連携する。

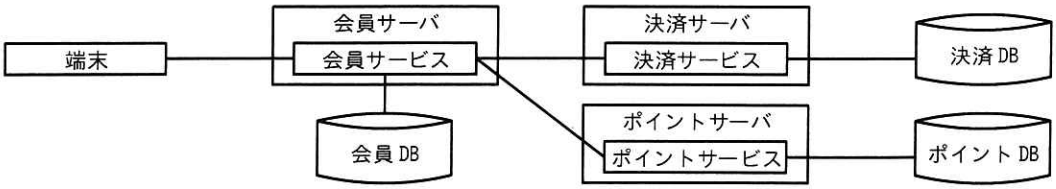

図 5　検討中のシステム構成

(1)　会員サービス

①　決済会員，ポイント会員の会員情報の登録，端末からの接続認証など会員に関わる機能をもつ AP，及び後述する複数サービスが関与する処理の実行制御を担う AP を配置する。

②　会員 DB には，現行システムと同じ RDBMS を使用する。“決済会員”，“ポイント会員ランク”及び“ポイント会員”テーブルを配置し，登録，認証に使用する。会員情報の登録，変更は他のサービスにも連携する。

(2)　決済サービス

①　決済 AP は，ポイントサービスと連携する処理を無効にすることを除いて変更せず，API を決済 AP に連携することで，特定の機能を呼び出して実行する方式に変更する。

②　決済 DB は，現行 RDBMS のデータベースをそのまま使用する。

(3)　ポイントサービス

①　ポイント AP は応答性能，可用性を高めるために再構築する。API をポイント AP に連携することで，特定の機能を呼び出して実行する方式に変更する。

②　ポイント DB を NoSQL に分類されるデータベースに変更し，図 3 中の全てのテーブルのデータを移行する。新しいポイント DB は，次の特徴をもつ。

・ワイドカラムデータストアをもち，複数ノードにデータを分散して配置することで，高速性，高可用性を実現している。

・RDBMS におけるテーブルと同様に，列及び主キーを定義できる。外部キーは定義できない。CRUD 操作命令を行う言語をサポートしていて，表 3～6 の SQL 文と同様の操作を記述できる。

・複数の CRUD 操作命令をまとめて一つのトランザクションとして実行する機能はなく，CRUD 操作命令の終了時点でデータへの操作が確定する。

・2相コミットを実現する機能はない。

2. 処理方式の設計

現行の2相コミットに替えて，次の処理方式を検討している。検討に当たって，ポイントDBに格納するデータの設計には，RDBMSのテーブル構造を用いる。

(1) 処理方式

① 複数サービスが関与する処理の実行制御を担うAP（以下，オーケストレーターという）を会員サーバに配置する。

② 端末からの要求ごとにオーケストレーターを割り当て，他サービスへの要求送信と結果受信とを同期的に行うことで，処理を順に実行していく。

③ オーケストレーターは，ワークフローの機能をもち，登録した処理を状態に応じて実行していく。タイムアウトなどの例外によって処理が異常終了した場合には，処理が成功するまで一定回数のリトライを行う。

④ リトライを経ても処理が失敗した場合，オーケストレーターは，それ以前に終了した処理について，ワークフローの逆方向に取消処理を順に実行する。取消処理では，データの変更を伴う場合に，そのデータ変更を取り消すトランザクション（以下，補償トランザクションという）を実行する。

(2) 実行制御テーブルの追加

会員DBに“決済指示”テーブルを追加し，オーケストレーターによるトランザクションの実行状態の記録，実行の制御に用いる。追加するテーブルのテーブル構造を図6に示す。“決済指示”テーブルの決済指示#は，決済指示ごとに一意な番号を自動採番し，トランザクションの識別子として使用する。オーケストレーターは，処理実行の指示及び補償トランザクション実行の指示に合わせて，“決済指示”テーブルに対応する行の全列の値を他サービスに連携する。

決済指示（決済指示#，加盟店#，店舗#，決済会員#，ポイント会員#，決済金額， 使用ポイント数，決済指示日時，処理ID，ステップ，モード，状態）

注記1 処理ID，ステップには仕掛り中の処理について表2～6の処理ID，ステップのいずれかを設定する。モードには通常モード，取消モードのいずれか，状態には進行中，完了，失敗のいずれかを表す値を設定する。

注記2 外部キーは会員DBに配置するテーブルとの間にだけ設定する。

図6 追加するテーブルのテーブル構造

(3) 決済処理の流れ

決済処理を次のように実行することにした。

① 決済サービスでは，S2 を一つの API として実行する。

② ポイントサービスでは，P3 のステップ P3-1～P3-3 のそれぞれに対応する API を用意し，会員サービスは API を順番に実行する。

③ 会員サービスでは，他のサービスへの実行指示に対する処理結果を“決済指示”テーブルの状態に記録していく。

検討中の決済処理の正常実行時のシーケンス図を図 7 に示す。図 7 中の①～⑮は実行の順番を，時点 C～E は実行の流れにおける特定の時点を表している。

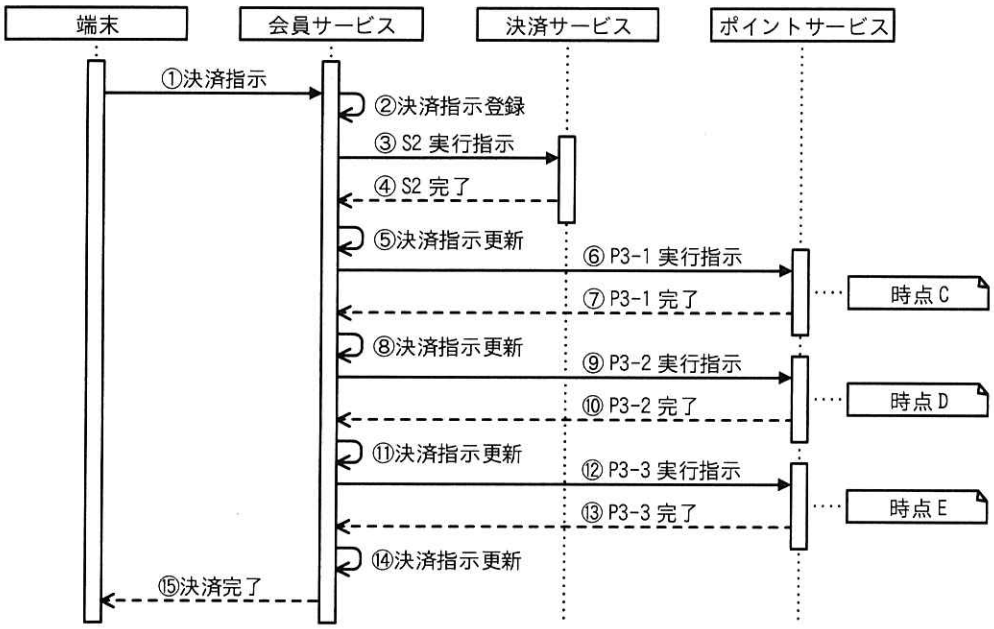

図 7 検討中の決済処理の正常実行時のシーケンス図

3. テーブル構造及び処理の変更

(1) リトライへの対応

図 7 中の実行指示に対応する処理の中には，リトライによる実行指示を受けたときに，重複更新の発生が懸念されるものがある。これを避けるためのテーブル構造及び処理の変更を行う。㋑決済サービスでは，対策として図 2 中の一つのテーブルに列を一つ追加し，処理 S2 の先頭にステップを一つ追加する。ポイントサービスでも同様の対策を行う。

(2) ポイントDBの整合性確保

表6の処理P3は，ステップごとにデータ操作が確定するので，ステップ実行の合間に他のトランザクションから同じ会員のデータ更新があると，ポイントDB内のデータに不整合が生じるおそれがある。これを防ぐために“ポイント会員”テーブルに更新中Fの列を追加してTRUE又はFALSEを設定した上で，㋒P2及びP3に，それぞれ同じ内容の処理を行うステップを追加する。

4. 補償トランザクションの設計

ここまでの検討結果を踏まえて，決済処理の補償トランザクションを表8にまとめた。表8中の処理IDは表2の処理ID，表8中のステップは表4，6のステップに対応し，補償トランザクションは，図7中の実行指示と同じ単位で実行するものとする。

表8 決済処理の補償トランザクション（未完成）

処理ID	ステップ	補償要否	補償可否	説明（補償不要の理由／補償不可の理由／補償処理の概要）
S2	S2-1	不要	-	参照だけでデータの変更がない。
	S2-2	要	可	h
	S2-3	要	可	i
P3	P3-1	不要	-	参照だけでデータの変更がない。
	P3-2	要	不可	㋓使用ポイント数を有効期限ごとの残数から消費した消込ポイント数が分からない。
	P3-3	要	可	

注記1 補償要否には変更の取消しが必要ならば“要”を，それ以外は“不要”を記入する。
注記2 補償可否には，現行システムのテーブル構造を前提に変更の取消しが可能ならば“可”を，可能でなければ“不可”を，補償が不要であれば“-”を記入する。
注記3 説明には，補償不要の場合は不要の理由を，補償不可の場合は取消しができない理由を，それ以外は，補償処理の概要を記入する。
注記4 網掛け部分は表示していない。

5. 障害からの回復

表8の補償要否が“要”のステップは，全て補償可否が“可”となるように対策を行った上で，決済処理実行中にポイントサービスが障害によって停止する状況を想定した障害・回復の事例を表9にまとめた。表中の時点C～Eは，図7中の時点C～Eに対応する。

表 9　決済処理中の障害・回復の事例

事例	内容
事例 5	ポイントサービスが時点 C で障害によって停止した。オーケストレーターは，P3-1 実行指示を一定回数リトライ後，S2 の取消実行指示を行った。
事例 6	ポイントサービスが時点 D で障害によって停止した。オーケストレーターは，P3-2 実行指示を一定回数のリトライ後，P3-1，S2 の取消実行指示を順に行った。
事例 7	ポイントサービスが時点 E で障害によって停止した。オーケストレーターは，P3-3 実行指示を一定回数のリトライ後，　P3-2 から S2 までの取消実行指示を順に行った。

設問 1　〔業務・現行システムの概要〕について答えよ。

(1)　表 5 中の［　a　］～［　d　］に入れる適切な字句を答えよ。

(2)　一貫性の観点から，処理実行時の ISOLATION レベルについて答えよ。

(a)　表 3 の処理 S1 の ISOLATION レベルは READ COMMITTED が適切である。REPEATABLE READ である必要がない理由を 30 字以内で具体的に答えよ。

(b)　表 6 の処理 P3 の ISOLATION レベルは SERIALIZABLE が適切である。REPEATABLE READ では不都合がある理由を 30 字以内で具体的に答えよ。

設問 2　〔現行システムの決済処理〕について答えよ。

(1)　図 4 について答えよ。

(a)　“①決済指示”に対応する“決済完了”の応答が欠けている。トランザクションが正常に終了するとみなした時点で決済 AP から端末に応答を返すとしたら，③～⑬のどのメッセージの受信後か。番号を一つ答えよ。

(b)　ポイント DB が④受信後の P3 実行において，ステップ P3-1 でポイント不足エラーを返した場合，⑥及び⑧で同じメッセージを送信してトランザクションを終了する。送信するメッセージの内容を答えよ。

(c)　時点 A と時点 B とでは，トランザクションの状態が異なる。状態の相違点を 45 字以内で具体的に答えよ。

(2)　表 7 について答えよ。

(a)　事例 1～3 の［　e　］～［　g　］に入れる適切な字句を次の中から選択して答えよ。

コミット準備指示，コミット実行指示，ロールバック実行指示， トランザクションを終了

(b) 事例 4 の下線㋐について，発生したと考えられる業務上の不都合を 30 字以内で具体的に答えよ。

(c) 事例 4 では，最終的にデータの状態はどのようになるか。40 字以内で具体的に答えよ。

設問3 〔システム変更案の検討〕について答えよ。

(1) “3. テーブル構造及び処理の変更”について答えよ。

(a) 本文中の下線㋑について，列を追加する図 2 中のテーブル名，追加する列名を答えよ。また，追加するステップの適切な処理概要を 40 字以内で答えよ。

(b) 本文中の下線㋒について，処理 P2 及び P3 の先頭に共通して追加するステップの適切な処理概要を 60 字以内で，末尾に共通して追加するステップの適切な処理概要を 20 字以内で答えよ。

(2) “4. 補償トランザクションの設計”について答えよ。

(a) 表 8 中の［ h ］，［ i ］に入れる適切な補償処理の概要をそれぞれ 40 字以内で答えよ。

(b) 表 8 中の下線㋓への対策として，ポイント DB に図 8 の“ポイント消込履歴”テーブルを追加して補償を可能にする。図 8 中の［ j ］に入れる一つ又は複数の適切な列名を答えよ。なお，主キーの構成列には巻頭の表記ルールに従って実線の下線を付けよ。

ポイント消込履歴（［ j ］，消込ポイント数）

図 8 “ポイント消込履歴”テーブルのテーブル構造（未完成）

(3) “5. 障害からの回復”について答えよ。

表 9 中の事例 5～7 のうち，ポイント DB 内のデータが一時的に不整合になる事例を一つ答えよ。また，不整合の内容を 50 字以内で具体的に答えよ。