0001 3-1 SQL検索 > SELECT・条件指定 ｜ 難易度：基礎

列 discount がNULLである行だけを検索したい。SQLとして最も適切なものはどれか。

ア：IS NULL を使う：WHERE discount IS NULL
イ：等号比較を使う：WHERE discount = NULL
ウ：二重等号を使う：WHERE discount == NULL
エ：NULLを関数のように扱う：WHERE NULL(discount)

答え・解説 正答：ア

ア：NULLは通常の比較演算子ではなく IS NULL で判定する。
イ：= NULL の結果は真にならず、NULL判定には使えない。
ウ：SQL標準の等価演算子は = であり、さらにNULLは = では判定しない。
エ：NULLを判定するこのような標準SQL構文はない。
総合解説：NULLは未知値を表すため、NULLとの比較には三値論理が関係する。NULLそのものの判定には IS NULL / IS NOT NULL を使う。

対象：2026年度 / 基準日 2026-09-02

0002 3-1 SQL検索 > SELECT・条件指定 ｜ 難易度：基礎

受注明細から、金額が1万円以上の行だけを集計対象にしたい。集約前に行を絞り込むために用いる句はどれか。

ア：HAVING句
イ：WHERE句
ウ：ORDER BY句
エ：GROUP BY句

答え・解説 正答：イ

ア：HAVING句は主に集約後のグループを絞り込む。
イ：WHERE句はグループ化や集約の前に行を絞り込む。
ウ：ORDER BY句は結果の並び順を指定する。
エ：GROUP BY句は集約単位を定義する。
総合解説：WHERE句はグループ化や集約の前に行を絞り込む。

対象：2026年度 / 基準日 2026-09-02

0003 3-1 SQL検索 > SELECT・条件指定 ｜ 難易度：標準

SQLの条件「score BETWEEN 60 AND 79」の説明として最も適切なものはどれか。

ア：scoreが60より大きく79より小さい行だけを対象とする。
イ：scoreが60以下または79以上の行を対象とする。
ウ：scoreが60以上79以下の行を対象とする。
エ：scoreが60または79の行だけを対象とする。

答え・解説 正答：ウ

ア：境界値60と79も含まれる。
イ：これは範囲外を表す条件であり、BETWEENの意味と異なる。
ウ：BETWEENは通常、下限と上限の両方を含む。
エ：BETWEENは端点だけでなく、その間の値も含む。
総合解説：BETWEENは通常、下限と上限の両方を含む。

対象：2026年度 / 基準日 2026-09-02

0004 3-1 SQL検索 > SELECT・条件指定 ｜ 難易度：標準

顧客名が『A』で始まり、その後に0文字以上が続く行を検索する条件として、最も適切なものはどれか。

ア：末尾一致を指定する：WHERE customer_name LIKE '%A'
イ：先頭1文字の後にAを指定する：WHERE customer_name LIKE '_A%'
ウ：等価比較で%を含む文字列を指定する：WHERE customer_name = 'A%'
エ：先頭一致を指定する：WHERE customer_name LIKE 'A%'

答え・解説 正答：エ

ア：これは末尾がAの文字列を対象にする。
イ：_はちょうど1文字に対応するので、2文字目がAの文字列を対象にする。
ウ：=では%は通常ワイルドカードとして扱われず、文字列A%との等価比較になる。
エ：%は0文字以上の任意の文字列に対応する。
総合解説：%は0文字以上の任意の文字列に対応する。

対象：2026年度 / 基準日 2026-09-02

0005 3-1 SQL検索 > SELECT・条件指定 ｜ 難易度：標準

SELECT DISTINCT department_id FROM employee; の主な効果はどれか。
ア：department_idの重複する値を除いて結果を返す。
イ：employee表からdepartment_idがNULLの行だけを除外する。
ウ：department_idを昇順に並べ替える。
エ：department_idに一意制約を追加する。

答え・解説 正答：ア

ア：DISTINCTはSELECT結果の重複行を除去する。
イ：DISTINCTはNULL除外を目的とする句ではない。
ウ：並べ替えはORDER BYで指定する。
エ：DISTINCTは検索結果に作用するだけで、表定義を変更しない。
総合解説：DISTINCTはSELECT結果の重複行を除去する。

対象：2026年度 / 基準日 2026-09-02

0006 3-1 SQL検索 > SELECT・条件指定 ｜ 難易度：標準

サブクエリがNULLを含む可能性があるとき、「NOT IN (サブクエリ)」で期待した行が返らないことがある主な理由はどれか。
ア：NOT INは数値列には使用できないから。
イ：NULLとの比較によりUNKNOWNが生じ、WHERE条件をTRUEにできない場合があるから。
ウ：NOT INは必ず内部結合に変換され、NULL行を全て返すから。
エ：サブクエリは必ず1行しか返せないから。

答え・解説 正答：イ

ア：NOT INは数値を含む一般的な比較に使用できる。
イ：NULLを含むNOT INでは三値論理によりUNKNOWNが生じる点に注意が必要である。
ウ：そのような一般則はない。
エ：IN/NOT INのサブクエリは複数行を返せる。
総合解説：NULLを含むNOT INでは三値論理によりUNKNOWNが生じる点に注意が必要である。

対象：2026年度 / 基準日 2026-09-02

0007 3-1 SQL検索 > SELECT・条件指定 ｜ 難易度：応用

『status='A' または status='B' のうち、amount>=10000の行』を確実に検索したい。意図を最も明確に表す条件はどれか。

ア：ANDの優先順位に任せて括弧を付けない：WHERE status='A' OR status='B' AND amount>=10000
イ：statusにAとBを同時要求する：WHERE status='A' AND status='B' AND amount>=10000
ウ：AまたはBを括弧でまとめてから金額条件をANDする：WHERE (status='A' OR status='B') AND amount>=10000
エ：status条件と金額条件をORで結ぶ：WHERE status IN ('A','B') OR amount>=10000

答え・解説 正答：ウ

ア：ANDがORより優先されるため、status=Aの行には金額条件が適用されない。
イ：同一列が同時にAかつBであることを要求してしまう。
ウ：OR条件を括弧でまとめてから金額条件とANDで結ぶと、意図が明確で誤解がない。
エ：金額だけ満たす他statusの行も返ってしまう。
総合解説：OR条件を括弧でまとめてから金額条件とANDで結ぶと、意図が明確で誤解がない。

対象：2026年度 / 基準日 2026-09-02

0008 3-1 SQL検索 > SELECT・条件指定 ｜ 難易度：応用

同じSELECT文を同じデータに対して繰り返し実行する。ORDER BYを指定していない場合の結果順序について、最も適切な説明はどれか。

ア：主キー順に必ず返る。
イ：物理格納順に必ず返る。
ウ：直前に実行したORDER BYの順序が次回も継承される。
エ：行の順序は保証されないので、必要な順序がある場合はORDER BYを明示する。

答え・解説 正答：エ

ア：主キーがあってもORDER BYなしで主キー順が保証されるわけではない。
イ：実行計画や並列処理などにより返却順は変わり得る。
ウ：ORDER BYは個々の問い合わせの結果順序を指定する。
エ：関係は本質的に順序を持たず、ORDER BYなしの返却順は保証すべきでない。
総合解説：関係は本質的に順序を持たず、ORDER BYなしの返却順は保証すべきでない。

対象：2026年度 / 基準日 2026-09-02

0009 3-1 SQL検索 > 結合・副問合せ・集合演算 ｜ 難易度：基礎

INNER JOINの説明として最も適切なものはどれか。
ア：結合条件を満たす行の組合せだけを結果に含める。
イ：左表の全行を必ず残す。
ウ：右表の全行を必ず残す。
エ：両表の行を条件なしに全組合せする。

答え・解説 正答：ア

ア：内部結合は両側に対応する行があり、結合条件を満たす組合せを返す。
イ：これはLEFT OUTER JOINの特徴である。
ウ：これはRIGHT OUTER JOINの特徴である。
エ：これはCROSS JOINに相当する。
総合解説：内部結合は両側に対応する行があり、結合条件を満たす組合せを返す。

対象：2026年度 / 基準日 2026-09-02

0010 3-1 SQL検索 > 結合・副問合せ・集合演算 ｜ 難易度：基礎

顧客表を左側、注文表を右側にしてLEFT OUTER JOINした。注文が1件もない顧客を含めたい場合の説明として正しいものはどれか。
ア：注文がない顧客は必ず除外される。
イ：注文がない顧客も残り、注文表側の列はNULLとして現れる。
ウ：注文がない顧客は顧客表側の列がNULLになる。
エ：注文表の未対応行だけが残る。

答え・解説 正答：イ

ア：それは内部結合の挙動である。
イ：LEFT OUTER JOINは左側の未対応行を保持し、右側をNULLで補う。
ウ：保持されるのは左表の行であり、NULL補完されるのは右表側である。
エ：LEFT JOINは左表の未対応行を残す。
総合解説：LEFT OUTER JOINは左側の未対応行を保持し、右側をNULLで補う。

対象：2026年度 / 基準日 2026-09-02

0011 3-1 SQL検索 > 結合・副問合せ・集合演算 ｜ 難易度：基礎

3行の表Aと4行の表Bを条件なしでCROSS JOINした。重複除去などを行わないとき、結果行数はいくつか。
ア：7行（3+4と加算した値）
イ：4行（片方の表Bの行数だけを使った値）
ウ：12行（3×4の直積）
エ：1行（CROSS JOINを集約と誤解した値）

答え・解説 正答：ウ

ア：CROSS JOINは行数を加算する演算ではない。
イ：片方の行数だけにはならない。
ウ：各A行と各B行の全組合せが生成されるため3×4=12行である。
エ：集約を行っていないため1行にはならない。
総合解説：CROSS JOINは直積なので、3×4=12行となる。

対象：2026年度 / 基準日 2026-09-02

0012 3-1 SQL検索 > 結合・副問合せ・集合演算 ｜ 難易度：基礎

相関副問合せの特徴として最も適切なものはどれか。
ア：副問合せが外側の問合せとは完全に独立している。
イ：必ず1行だけを返さなければならない。
ウ：SELECT句には記述できない。
エ：内側の副問合せが外側の問合せの列を参照する。

答え・解説 正答：エ

ア：これは非相関副問合せの説明である。
イ：EXISTSなど複数行でも成立する用途がある。
ウ：DBMSやSQL構文によりスカラ副問合せをSELECT句に記述できる。
エ：相関副問合せは外側行の値を参照して条件を評価する。
総合解説：相関副問合せは外側行の値を参照して条件を評価する。

対象：2026年度 / 基準日 2026-09-02

0013 3-1 SQL検索 > 結合・副問合せ・集合演算 ｜ 難易度：標準

EXISTS (subquery) がTRUEになる条件として最も適切なものはどれか。
ア：副問合せが少なくとも1行を返す。
イ：副問合せがちょうど1行を返す。
ウ：副問合せのSELECTリストがTRUEという値を返す。
エ：副問合せにNULLが1つも含まれない。

答え・解説 正答：ア

ア：EXISTSは副問合せ結果に行が存在するかを判定する。
イ：1行に限定されない。
ウ：EXISTSはSELECTリストの値そのものではなく行の存在を判定する。
エ：NULLの有無はEXISTSの成立条件ではない。
総合解説：EXISTSは副問合せ結果に行が存在するかを判定する。

対象：2026年度 / 基準日 2026-09-02

0014 3-1 SQL検索 > 結合・副問合せ・集合演算 ｜ 難易度：標準

UNIONとUNION ALLの一般的な違いとして最も適切なものはどれか。
ア：UNION ALLだけが列数の異なる問合せを結合できる。
イ：UNIONは重複行を除去し、UNION ALLは重複行も保持する。
ウ：UNIONは必ず入力順に並べ替える。
エ：UNION ALLはNULLを含む行を除外する。

答え・解説 正答：イ

ア：どちらも対応する列数と型の整合が必要である。
イ：集合演算では重複処理の違いが重要である。
ウ：最終結果の順序はORDER BYなしでは保証されない。
エ：NULL行除外がUNION ALLの役割ではない。
総合解説：集合演算では重複処理の違いが重要である。

対象：2026年度 / 基準日 2026-09-02

0015 3-1 SQL検索 > 結合・副問合せ・集合演算 ｜ 難易度：標準

二つのSELECT結果をUNIONで結合する際に最も重要な前提はどれか。
ア：両SELECTの表名が同じであること。
イ：両SELECTのWHERE条件が同じであること。
ウ：対応する列の数が一致し、互換性のあるデータ型であること。
エ：両SELECTの行数が同じであること。

答え・解説 正答：ウ

ア：異なる表からの結果でもUNIONできる。
イ：条件は異なっていてよい。
ウ：集合演算は位置対応で列を組み合わせるため、列数と型の整合が必要である。
エ：入力行数が異なっていても集合演算できる。
総合解説：集合演算は位置対応で列を組み合わせるため、列数と型の整合が必要である。

対象：2026年度 / 基準日 2026-09-02

0016 3-1 SQL検索 > 結合・副問合せ・集合演算 ｜ 難易度：標準

employee(emp_id, manager_id, name) から社員名と直属上司名を同じ表を使って取得したい。基本的な方法として最も適切なものはどれか。
ア：employee表をUNION ALLで自分自身と連結する。
イ：employee表にGROUP BYだけを指定する。
ウ：employee表をCROSS JOINして結合条件を付けない。
エ：employee表に別名を二つ付け、employee同士を自己結合する。

答え・解説 正答：エ

ア：UNION ALLは行集合を縦に連結する操作で、上司と部下の対応付けにはならない。
イ：GROUP BYは対応する上司行の列を取得する手段ではない。
ウ：全組合せになり、直属上司だけに対応付けられない。
エ：同一表の行同士の関係を取得するには、別名を付けた自己結合が適する。
総合解説：同一表の行同士の関係を取得するには、別名を付けた自己結合が適する。

対象：2026年度 / 基準日 2026-09-02

0017 3-1 SQL検索 > 結合・副問合せ・集合演算 ｜ 難易度：標準

顧客を左表、注文を右表としてLEFT JOINし、注文のない顧客も残したい。WHERE句に『order.amount > 0』を置くと注文なし顧客が消える主な理由はどれか。

ア：注文なし顧客では右表列がNULLになり、そのWHERE条件がTRUEにならないから。
イ：LEFT JOINは常に内部結合へ自動変換されるから。
ウ：amount列のNULLは自動的に0へ変換されるから。
エ：WHERE句は結合より前に必ず評価されるから。

答え・解説 正答：ア

ア：外部結合後に右表列をWHEREで絞ると、NULL補完行が除外されて実質的に内部結合に近い結果になることがある。
イ：LEFT JOINそのものが常に内部結合になるわけではない。
ウ：一般にNULLが自動で0になるわけではない。
エ：論理的には外部結合による行形成後のWHERE条件でNULL補完行が除かれる点が重要である。
総合解説：外部結合後に右表列をWHEREで絞ると、NULL補完行が除外されて実質的に内部結合に近い結果になることがある。

対象：2026年度 / 基準日 2026-09-02

0018 3-1 SQL検索 > 結合・副問合せ・集合演算 ｜ 難易度：標準

「注文が一度もない顧客」を求める方法として、NULLの影響を避けやすい代表的な書き方はどれか。

ア：注文表とINNER JOINし、結合できた顧客だけを返す。
イ：顧客ごとに対応する注文が存在しないことをNOT EXISTSで判定する。
ウ：注文表をCROSS JOINし、全組合せを返す。
エ：注文表だけをSELECT DISTINCTする。

答え・解説 正答：イ

ア：これは注文がある顧客を返す方向になる。
イ：NOT EXISTSは対応行の不存在を直接表現でき、NULLを含むNOT INの三値論理問題を避けやすい。
ウ：不存在の判定にならない。
エ：顧客表との対応関係を判定していない。
総合解説：NOT EXISTSは対応行の不存在を直接表現でき、NULLを含むNOT INの三値論理問題を避けやすい。

対象：2026年度 / 基準日 2026-09-02

0019 3-1 SQL検索 > 結合・副問合せ・集合演算 ｜ 難易度：応用

式の一部として「salary > (SELECT salary FROM employee WHERE department_id=10)」を使う。副問合せが複数行を返す可能性がある場合の問題として最も適切なものはどれか。

ア：DBMSが自動的に最大値だけを採用するので問題ない。
イ：複数行なら自動的にIN条件へ書き換えられる。
ウ：単一値を要求する位置なので、複数行ならエラーになる可能性があり、集約やANY/ALL等の意図に合う構文を検討すべきである。
エ：副問合せはWHERE句内では常に複数行を許す。

答え・解説 正答：ウ

ア：一般にそのような自動選択は期待できない。
イ：比較演算子が自動的にINへ変わるわけではない。
ウ：スカラ値が必要な位置に複数行結果を置くと不適切であり、要件に応じて比較方法を明示する必要がある。
エ：置かれる文脈により単一値が要求されることがある。
総合解説：スカラ値が必要な位置に複数行結果を置くと不適切であり、要件に応じて比較方法を明示する必要がある。

対象：2026年度 / 基準日 2026-09-02

0020 3-1 SQL検索 > 結合・副問合せ・集合演算 ｜ 難易度：応用

同じ列構成の「現行顧客」と「退会顧客」の検索結果を縦にまとめたい。顧客ID同士の対応付けではなく、両集合の行を一つの結果集合にしたい。最も適切な操作はどれか。

ア：INNER JOINを使う。
イ：FULL OUTER JOINだけを使う。
ウ：GROUP BYを使う。
エ：UNIONまたはUNION ALLを目的に応じて使う。

答え・解説 正答：エ

ア：JOINは列方向に関連行を対応付ける操作であり、行集合を縦に連結する目的とは異なる。
イ：外部結合も行の対応付けが中心で、同一列構成の集合を縦に統合する用途ではない。
ウ：GROUP BYは集約単位を形成する。
エ：縦方向の行集合の結合には集合演算を用いる。重複除去の要否でUNION/UNION ALLを選ぶ。
総合解説：縦方向の行集合の結合には集合演算を用いる。重複除去の要否でUNION/UNION ALLを選ぶ。

対象：2026年度 / 基準日 2026-09-02

0021 3-1 SQL検索 > 集約・分析・ウィンドウ関数 ｜ 難易度：基礎

部署ごとの給与平均を求めるとき、部署単位のグループを作るために使用する句はどれか。
ア：GROUP BY
イ：ORDER BY
ウ：WHERE
エ：DISTINCT

答え・解説 正答：ア

ア：GROUP BYは指定したキーごとに行をグループ化し、集約関数を適用できる。
イ：結果の並び順を指定する句である。
ウ：集約前の行を絞る句である。
エ：結果の重複行を除去する句である。
総合解説：GROUP BYは指定したキーごとに行をグループ化し、集約関数を適用できる。

対象：2026年度 / 基準日 2026-09-02

0022 3-1 SQL検索 > 集約・分析・ウィンドウ関数 ｜ 難易度：基礎

COUNT(*) と COUNT(column_name) の違いとして最も適切なものはどれか。
ア：両者は常に同じ件数を返す。
イ：COUNT(*)は行数を数え、COUNT(column_name)はその列がNULLでない行を数える。
ウ：COUNT(*)はNULLを含む行を除外する。
エ：COUNT(column_name)は重複値を自動除去する。

答え・解説 正答：イ

ア：column_nameにNULLがあれば異なる。
イ：COUNT(expr)はNULLの入力を数えず、COUNT(*)は行そのものを数える。
ウ：COUNT(*)は行を数える。
エ：重複除去にはCOUNT(DISTINCT column_name)を用いる。
総合解説：COUNT(expr)はNULLの入力を数えず、COUNT(*)は行そのものを数える。

対象：2026年度 / 基準日 2026-09-02

0023 3-1 SQL検索 > 集約・分析・ウィンドウ関数 ｜ 難易度：基礎

部署ごとの売上合計を求め、その合計が100万円以上の部署だけを残したい。最も直接的に用いる句はどれか。
ア：WHERE
イ：ORDER BY
ウ：HAVING
エ：VALUES

答え・解説 正答：ウ

ア：WHEREは通常、集約前の個々の行を絞る。
イ：並べ替えのための句である。
ウ：集約結果に対する条件はHAVINGで表す。
エ：行値を構成するための構文で、集約後の絞込みではない。
総合解説：集約結果に対する条件はHAVINGで表す。

対象：2026年度 / 基準日 2026-09-02

0024 3-1 SQL検索 > 集約・分析・ウィンドウ関数 ｜ 難易度：標準

ウィンドウ関数の特徴として最も適切なものはどれか。
ア：必ず1グループを1行に縮約する。
イ：WHERE句でのみ使用できる。
ウ：JOINを置き換えるためだけの機能である。
エ：関連する行集合を参照して計算しつつ、通常は元の各行を結果に残せる。

答え・解説 正答：エ

ア：これは通常のGROUP BY集約の特徴である。
イ：ウィンドウ関数は論理的な評価順序上、WHERE句では直接使えない。
ウ：順位や累計など幅広い分析に使われる。
エ：ウィンドウ関数はOVER句で行集合を定義し、集約関数のように行を1グループ1行へ縮約せず計算できる。
総合解説：ウィンドウ関数はOVER句で行集合を定義し、集約関数のように行を1グループ1行へ縮約せず計算できる。

対象：2026年度 / 基準日 2026-09-02

0025 3-1 SQL検索 > 集約・分析・ウィンドウ関数 ｜ 難易度：標準

ROW_NUMBER() OVER (PARTITION BY department_id ORDER BY salary DESC) の PARTITION BY department_idの役割はどれか。
ア：部署ごとに独立したウィンドウ区画を作り、各部署内で行番号を振る。
イ：結果全体をdepartment_idで並べ替えるだけである。
ウ：同じdepartment_idの行を1行に集約する。
エ：department_idがNULLの行を除外する。

答え・解説 正答：ア

ア：PARTITION BYはウィンドウ関数の計算対象を区画に分ける。
イ：並び順はOVER内のORDER BYが担う。
ウ：ウィンドウ関数では行を縮約しない。
エ：PARTITION BYはNULL除外の指定ではない。
総合解説：PARTITION BYはウィンドウ関数の計算対象を区画に分ける。

対象：2026年度 / 基準日 2026-09-02

0026 3-1 SQL検索 > 集約・分析・ウィンドウ関数 ｜ 難易度：標準

同点の成績がある場合でも各行に必ず異なる連番を付けたい。最も適切な関数はどれか。
ア：RANK()
イ：ROW_NUMBER()
ウ：DENSE_RANK()
エ：COUNT(*)

答え・解説 正答：イ

ア：RANKは同値に同順位を与え、次順位に欠番が生じる。
イ：ROW_NUMBERは同順位を作らず各行へ一意の連番を付ける。
ウ：DENSE_RANKも同値には同順位を与える。
エ：件数を数える集約であり、行ごとの連番付与ではない。
総合解説：ROW_NUMBERは同順位を作らず各行へ一意の連番を付ける。

対象：2026年度 / 基準日 2026-09-02

0027 3-1 SQL検索 > 集約・分析・ウィンドウ関数 ｜ 難易度：標準

「WHERE SUM(amount) > 100000」のように、同じ問合せレベルで集約関数をWHERE句へ直接書くのが不適切な主な理由はどれか。
ア：SUMはSELECT句でしか使えないから。
イ：WHEREは数値比較を禁止しているから。
ウ：WHEREによる行の絞込みは集約結果が形成される前に論理的に評価されるから。
エ：SUMはNULLを1件でも含むと必ずエラーになるから。

答え・解説 正答：ウ

ア：SUMはHAVINGやORDER BYなどでも利用できる。
イ：通常の数値比較はWHEREで行える。
ウ：集約値に対する条件はHAVINGや副問合せを用いて表現する。
エ：SUMは一般にNULLを無視して集計する。
総合解説：集約値に対する条件はHAVINGや副問合せを用いて表現する。

対象：2026年度 / 基準日 2026-09-02

0028 3-1 SQL検索 > 集約・分析・ウィンドウ関数 ｜ 難易度：標準

日付順の売上累計を各行に表示したい。考え方として最も適切なものはどれか。
ア：GROUP BY dateだけで各行の累計が自動計算される。
イ：COUNT(*)をORDER BYなしで使えば金額累計になる。
ウ：DISTINCT amountで重複金額を消せば累計になる。
エ：SUM(amount)をウィンドウ関数として使い、日付でORDER BYして先頭から現在行までのフレームを指定する。

答え・解説 正答：エ

ア：GROUP BYは日付単位の集約であり、先頭からの累計にはならない。
イ：COUNTは件数であり金額合計ではない。
ウ：重複除去は累計計算ではない。
エ：累計は順序付きウィンドウと適切なフレームで表現できる。
総合解説：累計は順序付きウィンドウと適切なフレームで表現できる。

対象：2026年度 / 基準日 2026-09-02

0029 3-1 SQL検索 > 集約・分析・ウィンドウ関数 ｜ 難易度：応用

部署ごとの給与上位3名だけを取得したい。ウィンドウ関数で順位を求める場合、最も一般的で適切な構成はどれか。

ア：内側の問合せでROW_NUMBER等を計算し、外側の問合せで順位列を3以下に絞る。
イ：同じ問合せのWHERE句にROW_NUMBER() <= 3を直接置く。
ウ：GROUP BY department_idだけで上位3名を自動抽出する。
エ：DISTINCT department_idで3名を残す。

答え・解説 正答：ア

ア：ウィンドウ関数はWHEREより後に評価されるため、計算結果を外側で絞る構成が分かりやすい。
イ：標準的な論理評価順ではWHEREでウィンドウ関数結果を直接参照できない。
ウ：GROUP BYだけでは上位3行の選択にならない。
エ：DISTINCTは重複除去であり順位に基づく抽出ではない。
総合解説：ウィンドウ関数はWHEREより後に評価されるため、計算結果を外側で絞る構成が分かりやすい。

対象：2026年度 / 基準日 2026-09-02

0030 3-1 SQL検索 > 集約・分析・ウィンドウ関数 ｜ 難易度：応用

同じORDER BY値をもつ行が複数ある累計計算で、ROWSフレームとRANGEフレームの結果が異なることがある。最も適切な説明はどれか。

ア：ROWSは重複行を自動削除し、RANGEは削除しないから。
イ：ROWSは物理的な行位置を基準に範囲を定め、RANGEはORDER BY値の同値行を同じ境界として扱うためである。
ウ：RANGEは数値列にしか使えず、ROWSは文字列列にしか使えないから。
エ：ROWSはGROUP BYを強制し、RANGEはHAVINGを強制するから。

答え・解説 正答：イ

ア：フレーム指定は重複除去機能ではない。
イ：同値のORDER BY値がある場合、フレーム単位の違いが累計結果に影響し得る。
ウ：そのような単純な型制約の違いではない。
エ：フレーム指定はGROUP BY/HAVINGを強制しない。
総合解説：同値のORDER BY値がある場合、フレーム単位の違いが累計結果に影響し得る。

対象：2026年度 / 基準日 2026-09-02

0031 3-2 DDL・DML > テーブル・データ型・DDL ｜ 難易度：基礎

テーブルの構造そのものを定義する代表的なSQL文はどれか。
ア：SELECT
イ：UPDATE
ウ：CREATE TABLE
エ：DELETE

答え・解説 正答：ウ

ア：SELECTはデータを検索する。
イ：UPDATEは既存行の値を変更するDMLである。
ウ：CREATE TABLEはテーブル定義を作成するDDLである。
エ：DELETEは行を削除するDMLである。
総合解説：CREATE TABLEはテーブル定義を作成するDDLである。

対象：2026年度 / 基準日 2026-09-02

0032 3-2 DDL・DML > テーブル・データ型・DDL ｜ 難易度：基礎

値の長さが大きく変動する名称列に対して、一般に固定長CHARより可変長VARCHARが適する主な理由はどれか。
ア：VARCHARは主キーに絶対使えないから。
イ：CHARでは文字比較が一切できないから。
ウ：VARCHARなら長さ制限を考えなくてよいから。
エ：実データ長に応じた格納を想定しやすく、固定長の余分な埋めを避けやすいから。

答え・解説 正答：エ

ア：VARCHARを主キーに使えるDBMSは一般的に存在する。
イ：CHARでも比較は可能である。
ウ：最大長や業務上の制約は依然として設計すべきである。
エ：文字型はデータの意味、最大長、処理特性を考慮して選ぶ。
総合解説：文字型はデータの意味、最大長、処理特性を考慮して選ぶ。

対象：2026年度 / 基準日 2026-09-02

0033 3-2 DDL・DML > テーブル・データ型・DDL ｜ 難易度：基礎

日付を「YYYYMMDD」の文字列ではなくDATE型で保持する主な利点として最も適切なものはどれか。
ア：日付としての妥当性、比較、日付演算などをDBMSの型機能で扱いやすい。
イ：DATE型ならNULLを格納できなくなる。
ウ：DATE型なら索引が不要になる。
エ：文字列型では一意制約を定義できない。

答え・解説 正答：ア

ア：業務上の意味に対応したデータ型を選ぶと整合性と操作性を高めやすい。
イ：NULL可否は別の制約で決まる。
ウ：検索要件に応じて索引は別途検討する。
エ：文字列列にも一意制約は定義できる。
総合解説：業務上の意味に対応したデータ型を選ぶと整合性と操作性を高めやすい。

対象：2026年度 / 基準日 2026-09-02

0034 3-2 DDL・DML > テーブル・データ型・DDL ｜ 難易度：標準

既存テーブルに新しい列を追加したい。一般に使用するDDLはどれか。
ア：CREATE VIEW
イ：ALTER TABLE
ウ：INSERT INTO
エ：GRANT SELECT

答え・解説 正答：イ

ア：ビューを定義する文である。
イ：ALTER TABLEは既存の表定義を変更するために用いる。
ウ：行を追加するDMLである。
エ：権限を付与する文で、列追加ではない。
総合解説：ALTER TABLEは既存の表定義を変更するために用いる。

対象：2026年度 / 基準日 2026-09-02

0035 3-2 DDL・DML > テーブル・データ型・DDL ｜ 難易度：標準

DROP TABLEとDELETE FROM tableの違いとして最も適切なものはどれか。
ア：DELETEは列定義も削除する。
イ：DROP TABLEはWHERE句で一部行だけ削除できる。
ウ：DROP TABLEは表定義そのものを削除し、DELETEは表を残したまま対象行を削除する。
エ：両者は完全に同じ操作である。

答え・解説 正答：ウ

ア：DELETEは行を削除するDMLである。
イ：DROP TABLEは表自体を削除する。
ウ：構造の削除とデータ行の削除を区別する。
エ：DDLとDMLで目的が異なる。
総合解説：構造の削除とデータ行の削除を区別する。

対象：2026年度 / 基準日 2026-09-02

0036 3-2 DDL・DML > テーブル・データ型・DDL ｜ 難易度：標準

金融金額を保持する列で、丸め誤差を避けて小数点以下の桁数を厳密に管理したい。一般に優先して検討すべき型はどれか。
ア：浮動小数点型だけを無条件に使う。
イ：文字列型にして全てアプリケーションで計算する。
ウ：BOOLEAN型を使う。
エ：固定精度のDECIMAL/NUMERIC型

答え・解説 正答：エ

ア：浮動小数点は表現誤差が生じ得るため、厳密な十進金額には注意が必要である。
イ：型による数値妥当性やDB側演算を活用しにくい。
ウ：真偽値用で金額を表さない。
エ：金額のように十進精度が重要な値は、固定精度十進型が適する。
総合解説：金額のように十進精度が重要な値は、固定精度十進型が適する。

対象：2026年度 / 基準日 2026-09-02

0037 3-2 DDL・DML > テーブル・データ型・DDL ｜ 難易度：標準

注文番号は全注文で必須で、欠落をDB側でも禁止したい。表定義でまず検討する制約はどれか。
ア：NOT NULL
イ：DEFAULT NULL
ウ：ORDER BY
エ：HAVING

答え・解説 正答：ア

ア：必須属性の欠落を禁止するにはNOT NULLが基本となる。
イ：欠落を許しNULLを既定にするため要件と逆である。
ウ：表定義の制約ではない。
エ：集約結果に対する条件である。
総合解説：必須属性の欠落を禁止するにはNOT NULLが基本となる。

対象：2026年度 / 基準日 2026-09-02

0038 3-2 DDL・DML > テーブル・データ型・DDL ｜ 難易度：標準

多くのビューやアプリケーションが参照する列名を変更する。最も適切な進め方はどれか。
ア：列名変更は物理構造だけの話なので影響分析は不要である。
イ：依存するビュー、SQL、ETL等を洗い出し、互換性と移行手順を検証してから変更する。
ウ：本番で直接変更し、失敗したらその場で考える。
エ：表の全データを削除すれば影響はなくなる。

答え・解説 正答：イ

ア：SQLやビューが列名を参照しているため影響し得る。
イ：DDL変更は依存オブジェクトやアプリケーションへ波及するため影響分析が必要である。
ウ：変更管理と事前検証が必要である。
エ：データ削除では依存関係の問題を解決しない。
総合解説：DDL変更は依存オブジェクトやアプリケーションへ波及するため影響分析が必要である。

対象：2026年度 / 基準日 2026-09-02

0039 3-2 DDL・DML > テーブル・データ型・DDL ｜ 難易度：応用

主キーcustomer_idの型を変更する際、複数表の外部キーがこの列を参照している。最も重要な設計上の確認はどれか。
ア：主キー表だけ型を変えれば参照表は自動的に無関係になる。
イ：外部キーを残したまま型不整合でも必ず動作する。
ウ：参照側外部キーの型・制約・移行順序も含めて一貫して変更できるか確認する。
エ：参照表は検索にしか使わないので型は何でもよい。

答え・解説 正答：ウ

ア：参照表の外部キー型や制約との整合が必要である。
イ：型互換性や制約定義に問題が生じ得る。
ウ：キーの型変更は参照関係全体の整合性と移行計画に影響する。
エ：結合や参照制約で型整合が重要である。
総合解説：キーの型変更は参照関係全体の整合性と移行計画に影響する。

対象：2026年度 / 基準日 2026-09-02

0040 3-2 DDL・DML > テーブル・データ型・DDL ｜ 難易度：応用

会員区分を1文字コードで保持しており、許容値はA/B/Cだけである。データ型選択だけでなくDB側で業務ルールを表現したい。最も適切な考え方はどれか。
ア：VARCHARなら自動的にA/B/C以外を拒否する。
イ：列を削除して区分をコメント欄に記載する。
ウ：ORDER BYでA/B/Cだけに限定する。
エ：文字型を選んだ上でCHECK制約などにより許容値A/B/Cを明示する。

答え・解説 正答：エ

ア：文字型は通常、業務上の列挙値までは制限しない。
イ：構造化された業務属性を失い、整合性を管理しにくい。
ウ：ORDER BYは並び順を指定する句で値域制約ではない。
エ：型だけでは表しきれない業務上の値域は制約で補完する。
総合解説：型だけでは表しきれない業務上の値域は制約で補完する。

対象：2026年度 / 基準日 2026-09-02

0041 3-2 DDL・DML > 制約・ビュー ｜ 難易度：基礎

PRIMARY KEY制約の説明として最も適切なものはどれか。
ア：行を一意に識別し、主キー列には重複値やNULLを許さない。
イ：重複を許すがNULLだけ禁止する。
ウ：NULLを許すが重複だけ禁止する。
エ：参照先の表を自動的に削除する。

答え・解説 正答：ア

ア：主キーはエンティティ識別の中心となる制約である。
イ：一意性も要求される。
ウ：主キーにはNULLを許さない。
エ：主キーの役割ではない。
総合解説：主キーはエンティティ識別の中心となる制約である。

対象：2026年度 / 基準日 2026-09-02

0042 3-2 DDL・DML > 制約・ビュー ｜ 難易度：基礎

外部キー制約の主な目的はどれか。
ア：参照元列の値を必ず連番にする。
イ：参照元の値が参照先の候補キー等に存在することを要求し、参照整合性を保つ。
ウ：参照元表の全列を一意にする。
エ：SELECT結果の並び順を固定する。

答え・解説 正答：イ

ア：採番は外部キーの役割ではない。
イ：外部キーは表間の参照関係の整合性をDB側で維持する。
ウ：外部キー自体は一意性を要求しない。
エ：並び順とは無関係である。
総合解説：外部キーは表間の参照関係の整合性をDB側で維持する。

対象：2026年度 / 基準日 2026-09-02

0043 3-2 DDL・DML > 制約・ビュー ｜ 難易度：基礎

商品価格priceを0以上に制限したい。表定義で直接表現する方法として最も適切なものはどれか。
ア：ORDER BY price
イ：GROUP BY price
ウ：CHECK (price >= 0)
エ：UNION price

答え・解説 正答：ウ

ア：並べ替えであり値域制約ではない。
イ：集約単位を作る句であり制約ではない。
ウ：CHECK制約は列や行の値が条件を満たすことを要求できる。
エ：集合演算であり制約ではない。
総合解説：CHECK制約は列や行の値が条件を満たすことを要求できる。

対象：2026年度 / 基準日 2026-09-02

0044 3-2 DDL・DML > 制約・ビュー ｜ 難易度：標準

候補キーの一つであるemailに重複を許さず、主キーは別のcustomer_idとしたい。基本的に適切な制約はどれか。
ア：emailにもPRIMARY KEYを追加し、同じ表に独立した主キーを二つ作る。
イ：emailにORDER BY制約を付ける。
ウ：emailを外部キーにすれば自動的に一意になる。
エ：emailにUNIQUE制約を付ける。

答え・解説 正答：エ

ア：通常、一つの表にPRIMARY KEY制約は一つである。複合主キーは一つの制約として定義する。
イ：ORDER BYは制約ではない。
ウ：外部キーは参照整合性を表し、一意性は別に必要である。
エ：主キーとは別の候補キーの一意性を表すにはUNIQUEが適する。
総合解説：主キーとは別の候補キーの一意性を表すにはUNIQUEが適する。

対象：2026年度 / 基準日 2026-09-02

0045 3-2 DDL・DML > 制約・ビュー ｜ 難易度：標準

外部キーにON DELETE CASCADEを設定した場合の一般的な効果はどれか。
ア：参照先の親行が削除されると、それを参照する子行も連鎖的に削除される。
イ：親行の削除を常に禁止する。
ウ：子行の外部キーを必ず0にする。
エ：削除ではなく子行の並び順を変更する。

答え・解説 正答：ア

ア：CASCADEは親行削除の影響を参照元へ伝播させる参照動作である。
イ：これはRESTRICT/NO ACTION系の考え方である。
ウ：CASCADEは子行そのものを削除する。
エ：参照動作は並び順とは無関係である。
総合解説：CASCADEは親行削除の影響を参照元へ伝播させる参照動作である。

対象：2026年度 / 基準日 2026-09-02

0046 3-2 DDL・DML > 制約・ビュー ｜ 難易度：標準

ビューの主な利用目的として最も適切なものはどれか。
ア：必ずデータを別領域へ複製して保持する。
イ：基礎表への問い合わせを定義として保存し、利用者へ必要な列・行だけの論理的な見え方を提供する。
ウ：主キー制約を自動的に全表へ追加する。
エ：物理ディスクのRAID構成を定義する。

答え・解説 正答：イ

ア：通常のビューは定義を保持し、必ず実体データを複製するわけではない。
イ：ビューはデータの論理的抽象化、アクセス範囲の簡素化などに利用できる。
ウ：ビューの役割ではない。
エ：ビューは論理データ表現でありストレージ構成とは異なる。
総合解説：ビューはデータの論理的抽象化、アクセス範囲の簡素化などに利用できる。

対象：2026年度 / 基準日 2026-09-02

0047 3-2 DDL・DML > 制約・ビュー ｜ 難易度：標準

人事表から氏名と部署だけを一般利用者に見せ、給与列は直接見せたくない。ビューを使う設計として最も適切なものはどれか。

ア：給与列も含むSELECT *のビューを作り、利用者側に見ないよう依頼する。
イ：基礎表への全権限を与え、ビューは作らない。
ウ：公開用ビューには氏名と部署だけを含め、利用者にはビューへのSELECT権限を与え、基礎表への不要な権限を与えない。
エ：給与列をNULLへUPDATEしてから公開する。

答え・解説 正答：ウ

ア：技術的な制御になっていない。
イ：給与列への直接アクセスを防げない。
ウ：ビューと権限制御を組み合わせることで公開範囲を限定できる。
エ：元データを破壊する必要はない。
総合解説：ビューと権限制御を組み合わせることで公開範囲を限定できる。

対象：2026年度 / 基準日 2026-09-02

0048 3-2 DDL・DML > 制約・ビュー ｜ 難易度：標準

複数アプリケーションが同じDBを更新する。参照整合性を各アプリケーションのコードだけで確認し、DBには外部キーを定義しない場合の主なリスクはどれか。

ア：DBMSが自動的に全アプリのコードを検証するのでリスクはない。
イ：外部キーを定義しない方が参照整合性は必ず強くなる。
ウ：参照整合性はSELECT時だけの問題なので更新時は無関係である。
エ：アプリケーションごとの実装漏れや直接更新により、孤児行などの不整合が入り得る。

答え・解説 正答：エ

ア：DBMSは外部アプリの全ロジックを自動検証しない。
イ：逆であり、DB側の強制がなくなる。
ウ：不整合は更新時に発生する。
エ：共有DBでは重要な整合性ルールをDB制約でも表現することで一貫した強制がしやすい。
総合解説：共有DBでは重要な整合性ルールをDB制約でも表現することで一貫した強制がしやすい。

対象：2026年度 / 基準日 2026-09-02

0049 3-2 DDL・DML > 制約・ビュー ｜ 難易度：応用

集約結果を示すビューに対して、基礎表のどの行をどう変更するか一意に決められないため更新が制限されることがある。最も適切な理由はどれか。

ア：ビューの1行が複数の基礎行から計算された結果で、更新先を一意に対応付けられない場合があるから。
イ：ビューはSQL文ではないので更新できないから。
ウ：ビューには列名が存在しないから。
エ：ビューを作ると基礎表が読み取り専用になるから。

答え・解説 正答：ア

ア：集約・結合などを含むビューでは更新可能性が制限されることがある。
イ：ビューはSQLで定義される論理表である。
ウ：ビューにも列がある。
エ：ビュー作成自体で基礎表が必ず読み取り専用になるわけではない。
総合解説：集約・結合などを含むビューでは更新可能性が制限されることがある。

対象：2026年度 / 基準日 2026-09-02

0050 3-2 DDL・DML > 制約・ビュー ｜ 難易度：応用

一つのトランザクション内で複数行を入れ替えるため、文単位では一時的に制約違反になるが、COMMIT時には整合する設計がある。DBMSが対応する場合に検討できる機能はどれか。

ア：ORDER BYをDEFERREDにする。
イ：制約の検査をトランザクション終了時まで遅延するDEFERRABLE/DEFERREDの仕組み。
ウ：ビューをDROPしてからCOMMITする。
エ：全制約を永久に無効化する。

答え・解説 正答：イ

ア：ORDER BYは制約ではない。
イ：遅延可能制約は一時的な中間状態を許し、COMMIT時に整合性を検査する用途に使える。
ウ：制約検査時期を制御する方法ではない。
エ：最終整合性を保証できず、要件に合わない。
総合解説：遅延可能制約は一時的な中間状態を許し、COMMIT時に整合性を検査する用途に使える。

対象：2026年度 / 基準日 2026-09-02

0051 3-2 DDL・DML > INSERT・UPDATE・DELETE等 ｜ 難易度：基礎

既存テーブルに新しい1行を追加するための代表的なSQL文はどれか。
ア：UPDATE
イ：DELETE
ウ：INSERT
エ：DROP TABLE

答え・解説 正答：ウ

ア：既存行の値を変更する。
イ：既存行を削除する。
ウ：INSERTは表へ行を追加するDMLである。
エ：表定義そのものを削除する。
総合解説：INSERTは表へ行を追加するDMLである。

対象：2026年度 / 基準日 2026-09-02

0052 3-2 DDL・DML > INSERT・UPDATE・DELETE等 ｜ 難易度：基礎

UPDATE employee SET status = 'R'; のようにWHERE句を付けなかった場合、一般に何が起こるか。
ア：先頭1行だけ更新される。
イ：主キーがNULLの行だけ更新される。
ウ：構文エラーになり必ず実行できない。
エ：対象表の全行が更新対象になる。

答え・解説 正答：エ

ア：そのような一般則はない。
イ：WHERE条件がないためその限定はない。
ウ：WHEREなしのUPDATE自体は有効なSQLである。
エ：WHEREがなければUPDATEは表の全行へ適用される。
総合解説：WHEREがなければUPDATEは表の全行へ適用される。

対象：2026年度 / 基準日 2026-09-02

0053 3-2 DDL・DML > INSERT・UPDATE・DELETE等 ｜ 難易度：基礎

DELETE FROM log_table; を実行した場合の一般的な効果はどれか。
ア：表定義は残したまま、対象表の全行を削除する。
イ：表定義ごと削除する。
ウ：1行だけ削除する。
エ：列を1本削除する。

答え・解説 正答：ア

ア：WHEREなしのDELETEは全行削除となる。
イ：表定義の削除はDROP TABLEである。
ウ：WHEREがないため全行が対象となる。
エ：列削除はALTER TABLE等のDDLで行う。
総合解説：WHEREなしのDELETEは全行削除となる。

対象：2026年度 / 基準日 2026-09-02

0054 3-2 DDL・DML > INSERT・UPDATE・DELETE等 ｜ 難易度：標準

INSERT文で「INSERT INTO t (col1,col2) VALUES (...)」のように列リストを明示する主な利点はどれか。
ア：必ず実行速度が10倍になる。
イ：値と列の対応を明確にし、列追加などの表変更に対する意図を分かりやすくできる。
ウ：主キー制約を無効化できる。
エ：NULLを自動的に0へ変換する。

答え・解説 正答：イ

ア：性能向上を保証する機能ではない。
イ：列リストの明示は可読性と保守性を高める。
ウ：制約は引き続き適用される。
エ：列リスト明示にその効果はない。
総合解説：列リストの明示は可読性と保守性を高める。

対象：2026年度 / 基準日 2026-09-02

0055 3-2 DDL・DML > INSERT・UPDATE・DELETE等 ｜ 難易度：標準

1行のstatusとupdated_atを同じUPDATE文で変更したい。最も適切な説明はどれか。
ア：UPDATEは1文につき1列しか変更できない。
イ：一度DELETEしてからINSERTし直すしかない。
ウ：SET句で複数の列代入を指定できる。
エ：ALTER TABLEで行値を変更する。

答え・解説 正答：ウ

ア：複数列を指定できる。
イ：通常はUPDATEで直接変更できる。
ウ：UPDATEでは一つの文で複数列を更新できる。
エ：ALTER TABLEは構造変更のDDLである。
総合解説：UPDATEでは一つの文で複数列を更新できる。

対象：2026年度 / 基準日 2026-09-02

0056 3-2 DDL・DML > INSERT・UPDATE・DELETE等 ｜ 難易度：標準

各商品のpriceを、同じカテゴリの平均価格を基準に更新したい。設計上の注意として最も適切なものはどれか。
ア：副問合せはUPDATEでは一切使えない。
イ：平均値は必ず整数なので型確認は不要である。
ウ：WHERE句がなくてもカテゴリごとに自動的に更新対象が限定される。
エ：更新対象と参照対象の関係、サブクエリが返す行数、同時更新時の評価を確認してSQLを設計する。

答え・解説 正答：エ

ア：DBMSやSQL構文によりUPDATEで副問合せを利用できる。
イ：平均値の型や丸めを確認すべきである。
ウ：条件は明示する必要がある。
エ：更新文に副問合せを使う場合、単一値性や相関条件、DBMSの更新規則を確認する必要がある。
総合解説：更新文に副問合せを使う場合、単一値性や相関条件、DBMSの更新規則を確認する必要がある。

対象：2026年度 / 基準日 2026-09-02

0057 3-2 DDL・DML > INSERT・UPDATE・DELETE等 ｜ 難易度：標準

archive_orderへ、order表の2025年度分だけを同じ列構成でコピーしたい。適切な考え方はどれか。
ア：INSERT INTO archive_order (...) SELECT ... FROM order WHERE ... の形を用いる。
イ：UPDATE archive_order SELECT ... とする。
ウ：DROP TABLE orderを実行する。
エ：ORDER BYだけを実行する。

答え・解説 正答：ア

ア：INSERTはVALUESだけでなくSELECT結果を挿入元にできる。
イ：UPDATEは既存行変更で、検索結果の行追加にはINSERT SELECTが適する。
ウ：元表を削除する操作でありコピーではない。
エ：並べ替えだけではデータを挿入できない。
総合解説：INSERTはVALUESだけでなくSELECT結果を挿入元にできる。

対象：2026年度 / 基準日 2026-09-02

0058 3-2 DDL・DML > INSERT・UPDATE・DELETE等 ｜ 難易度：標準

外部キー制約のある子表へ、親表に存在しないキー値をINSERTしようとした。一般的に期待される動作はどれか。
ア：自動的に親行が新規作成される。
イ：外部キー制約に違反するため、挿入が拒否される。
ウ：外部キー制約はSELECT時だけ有効なので挿入できる。
エ：子行のキーが自動的にNULLへ変換される。

答え・解説 正答：イ

ア：通常の外部キーにその自動作成機能はない。
イ：DMLは定義済みの整合性制約に従う必要がある。
ウ：参照整合性は更新時に検査される。
エ：そのような自動変換は一般的ではない。
総合解説：DMLは定義済みの整合性制約に従う必要がある。

対象：2026年度 / 基準日 2026-09-02

0059 3-2 DDL・DML > INSERT・UPDATE・DELETE等 ｜ 難易度：応用

在庫quantityを1減らす処理で、在庫が1以上のときだけ減算したい。並行実行も考慮して、単一UPDATEで条件を表す考え方として最も適切なものはどれか。

ア：先にSELECTで在庫を読み、長時間待ってから無条件UPDATEする。

イ：在庫列を文字列へ変更する。

ウ：UPDATEで「SET quantity=quantity-1 WHERE item_id=? AND quantity>=1」のように条件付き更新し、更新件数を確認する。

エ：WHEREを省略して全商品を減算する。

答え・解説 正答：ウ

ア：SELECT後に他トランザクションが変更する競合窓がある。

イ：並行制御の解決にならない。

ウ：読取りと更新を分離するより、条件付き更新を一つのDMLにまとめると競合窓を小さくできる。必要に応じてロック・分離レベルも組み合わせる。

エ：対象行と条件を限定できない。

総合解説：読取りと更新を分離するより、条件付き更新を一つのDMLにまとめると競合窓を小さくできる。必要に応じてロック・分離レベルも組み合わせる。

対象：2026年度 / 基準日 2026-09-02

0060 3-2 DDL・DML > INSERT・UPDATE・DELETE等 ｜ 難易度：応用

親行を削除しようとしたが、子表から外部キーで参照されており、参照動作は削除を許可しない設定である。最も適切な説明はどれか。

ア：親行は必ず削除され、子行は孤児行として残る。

イ：DELETEは制約を無視するDMLなので必ず成功する。

ウ：子表の全行が自動的にNULLになる。

エ：参照整合性を守るため親行削除は拒否され、先に子行を処理するか参照動作の設計を見直す必要がある。

答え・解説 正答：エ

ア：制約が削除を禁止する設定なら拒否される。

イ：DMLにも制約は適用される。

ウ：SET NULL等を明示的に設計した場合に限られる。

エ：親削除の可否は外部キーの参照動作に従う。

総合解説：親削除の可否は外部キーの参照動作に従う。

対象：2026年度 / 基準日 2026-09-02

0061 3-3 トランザクション・排他制御 > ACID・COMMIT・ROLLBACK ｜ 難易度：基礎

ACID特性のうち、トランザクション内の一連の処理が「全て実行されるか、全て取り消されるか」であることを表すものはどれか。
ア：Atomicity（原子性）
イ：Consistency（一貫性）
ウ：Isolation（独立性）
エ：Durability（永続性）

答え・解説 正答：ア

ア：原子性はトランザクションを不可分な処理単位として扱う性質である。
イ：整合性制約を保つ性質を指す。
ウ：並行実行時に互いの中間状態から隔離される性質を指す。
エ：COMMIT済み結果が障害後も保持される性質を指す。
総合解説：原子性はトランザクションを不可分な処理単位として扱う性質である。

対象：2026年度 / 基準日 2026-09-02

0062 3-3 トランザクション・排他制御 > ACID・COMMIT・ROLLBACK ｜ 難易度：基礎

ACID特性のうち、COMMITが成功したトランザクションの結果が、その後の障害があっても失われないよう保証する性質はどれか。
ア：Atomicity（原子性）：処理を全か無かで扱う性質
イ：Durability（永続性）：COMMIT済み結果を障害後も保持する性質
ウ：Consistency（一貫性）：整合性制約を保つ性質
エ：Isolation（独立性）：並行処理の中間状態を隔離する性質

答え・解説 正答：イ

ア：全か無かの性質である。
イ：永続性はコミット済み結果を恒久的に保持する性質である。
ウ：整合性の保持に関する性質である。
エ：並行トランザクション間の見え方に関する性質である。
総合解説：永続性はコミット済み結果を恒久的に保持する性質である。

対象：2026年度 / 基準日 2026-09-02

0063 3-3 トランザクション・排他制御 > ACID・COMMIT・ROLLBACK ｜ 難易度：標準

COMMITの説明として最も適切なものはどれか。
ア：現在のトランザクションの変更を全て取り消す。
イ：表定義を削除する。
ウ：現在のトランザクションの変更を確定する。
エ：デッドロックを必ず発生させる。

答え・解説 正答：ウ

ア：これはROLLBACKである。
イ：DROP TABLE等のDDLの役割である。
ウ：COMMITはトランザクションを正常終了し、変更を確定する。
エ：COMMITはデッドロックを発生させるための文ではない。
総合解説：COMMITはトランザクションを正常終了し、変更を確定する。

対象：2026年度 / 基準日 2026-09-02

0064 3-3 トランザクション・排他制御 > ACID・COMMIT・ROLLBACK ｜ 難易度：標準

送金処理の途中で業務エラーを検出し、トランザクション開始後の変更を確定せず取り消したい。用いる操作はどれか。
ア：COMMIT
イ：CREATE TABLE
ウ：ORDER BY
エ：ROLLBACK

答え・解説 正答：エ

ア：変更を確定するため目的と逆である。
イ：表定義作成でありトランザクション取消しではない。
ウ：検索結果の並び順指定である。
エ：ROLLBACKは現在のトランザクションで行った未確定変更を取り消す。
総合解説：ROLLBACKは現在のトランザクションで行った未確定変更を取り消す。

対象：2026年度 / 基準日 2026-09-02

0065 3-3 トランザクション・排他制御 > ACID・COMMIT・ROLLBACK ｜ 難易度：標準

一つの長いトランザクションで、途中までの処理は残し、その後の一部だけを取り消して続行したい。DBMSが対応している場合に使う機能はどれか。
ア：SAVEPOINTを設定し、必要時にROLLBACK TO SAVEPOINTを使う。
イ：毎回COMMITしてから後続処理を同じトランザクションとして扱う。
ウ：DROP TABLEで途中状態を消す。
エ：SELECT DISTINCTで変更を取り消す。

答え・解説 正答：ア

ア：セーブポイントはトランザクション全体を中止せず、一部を巻き戻すために使える。
イ：COMMITするとその時点でトランザクションが終了し、原子性の境界が変わる。
ウ：構造削除であり部分ロールバックではない。
エ：検索の重複除去であり更新取消しではない。
総合解説：セーブポイントはトランザクション全体を中止せず、一部を巻き戻すために使える。

対象：2026年度 / 基準日 2026-09-02

0066 3-3 トランザクション・排他制御 > ACID・COMMIT・ROLLBACK ｜ 難易度：標準

受注登録で「受注ヘッダ作成」と「複数明細作成」が全て成功したときだけ受注を成立させたい。最も適切なトランザクション設計はどれか。
ア：ヘッダ登録直後に必ずCOMMITし、明細失敗時もヘッダだけ残す。
イ：ヘッダと明細の登録を同一トランザクションに含め、全て成功時にCOMMITする。
ウ：各列を別トランザクションにする。
エ：SELECTだけトランザクションに含め、更新は含めない。

答え・解説 正答：イ

ア：業務要件の全か無かを満たさない。
イ：業務上不可分な更新を同一トランザクション境界に含めることで原子性を確保する。
ウ：整合した受注単位を作りにくい。
エ：更新間の原子性を確保できない。
総合解説：業務上不可分な更新を同一トランザクション境界に含めることで原子性を確保する。

対象：2026年度 / 基準日 2026-09-02

0067 3-3 トランザクション・排他制御 > ACID・COMMIT・ROLLBACK ｜ 難易度：応用

大量処理で一つのトランザクションを何時間も開いたままにする設計の一般的な問題として最も適切なものはどれか。

ア：長いほど必ず同時実行性が向上する。
イ：長いトランザクションではCOMMITが不要になる。
ウ：ロック保持やバージョン保持、障害時の巻戻し量などが増え、他処理や資源へ影響し得る。
エ：長いトランザクションはACIDと無関係なので設計上考慮不要である。

答え・解説 正答：ウ

ア：一般にはロックや資源保持が長くなるため逆効果になり得る。
イ：最終的な確定処理は必要である。
ウ：トランザクション境界は業務整合性だけでなく同時実行性、ログ、回復コストも考慮する。
エ：整合性・回復・同時実行性に強く関係する。
総合解説：トランザクション境界は業務整合性だけでなく同時実行性、ログ、回復コストも考慮する。

対象：2026年度 / 基準日 2026-09-02

0068 3-3 トランザクション・排他制御 > ACID・COMMIT・ROLLBACK ｜ 難易度：応用

アプリケーションで2つのUPDATEを必ず一体として成功・失敗させたいが、接続が各文を自動コミットする設定になっている。最も適切な対応はどれか。

ア：1文目の後に待ち時間を入れる。
イ：2文目をSELECTへ変更する。
ウ：ORDER BYを付ける。
エ：明示的なトランザクションを開始し、2文を同一トランザクションに含めて最後にCOMMIT/ROLLBACKする。

答え・解説 正答：エ

ア：原子性は確保されない。
イ：必要な更新要件を満たさない。
ウ：トランザクション境界には影響しない。
エ：自動コミットのままでは各文が別々に確定され、2文全体の原子性を保証できない。
総合解説：自動コミットのままでは各文が別々に確定され、2文全体の原子性を保証できない。

対象：2026年度 / 基準日 2026-09-02

0069 3-3 トランザクション・排他制御 > 分離レベル・並行実行異常 ｜ 難易度：基礎

ダーティリードの説明として最も適切なものはどれか。

ア：他トランザクションがまだCOMMITしていない更新値を読み取ること。
イ：同じ行を二度読んだら確定済み値が変わること。
ウ：同じ検索条件で再検索したら行集合が増減すること。
エ：二つのトランザクションが互いのロックを待つこと。

答え・解説 正答：ア

ア：未確定データを読む現象をダーティリードという。
イ：これは非再現読取りに関係する。
ウ：これはファントムリードに関係する。
エ：これはデッドロックである。
総合解説：未確定データを読む現象をダーティリードという。

対象：2026年度 / 基準日 2026-09-02

0070 3-3 トランザクション・排他制御 > 分離レベル・並行実行異常 ｜ 難易度：基礎

非再現読取りの説明として最も適切なものはどれか。

ア：未確定値を読むこと。
イ：同一トランザクションで同じ行を二度読んだ間に他トランザクションが更新・確定し、値が変わって見えること。
ウ：検索条件に合う新しい行が出現することだけを指す。
エ：更新をCOMMITできないこと全般。

答え・解説 正答：イ

ア：これはダーティリードである。
イ：同じ行を再読した結果が変わる現象である。
ウ：これはファントムリードに関係する。
エ：非再現読取りの定義ではない。
総合解説：同じ行を再読した結果が変わる現象である。

対象：2026年度 / 基準日 2026-09-02

0071 3-3 トランザクション・排他制御 > 分離レベル・並行実行異常 ｜ 難易度：基礎

ファントムリードの典型例として最も適切なものはどれか。

ア：同じ主キー行の未確定値を読む。
イ：トランザクションがCOMMIT後もロックを永遠に保持する。
ウ：同じ条件で範囲検索を再実行したとき、他トランザクションのINSERT/DELETEにより該当行集合が増減して見える。
エ：SQL構文が間違ってエラーになる。

答え・解説 正答：ウ

ア：ダーティリードに関係する。
イ：ファントムリードの定義ではない。
ウ：条件に合う行集合そのものが変化する現象をファントムリードという。
エ：並行実行異常ではない。
総合解説：条件に合う行集合そのものが変化する現象をファントムリードという。

対象：2026年度 / 基準日 2026-09-02

0072 3-3 トランザクション・排他制御 > 分離レベル・並行実行異常 ｜ 難易度：基礎

更新消失（Lost Update）の典型的な状況はどれか。

ア：未確定値をSELECTするだけで更新は行わない。
イ：別々の行を独立に更新する。
ウ：SELECT結果をORDER BYする。
エ：二つのトランザクションが同じ元値を読み、それぞれ計算して更新した結果、後の更新が先の更新効果を上書きする。

答え・解説 正答：エ

ア：更新消失ではない。
イ：同じデータを競合更新しないなら典型的な更新消失ではない。
ウ：並行更新異常と無関係である。
エ：読取りと更新の競合により一方の変更が失われる現象である。
総合解説：読取りと更新の競合により一方の変更が失われる現象である。

対象：2026年度 / 基準日 2026-09-02

0073 3-3 トランザクション・排他制御 > 分離レベル・並行実行異常 ｜ 難易度：標準

READ COMMITTEDの一般的な考え方として最も適切なものはどれか。
ア：各文は開始時点までに確定したデータを読み、同一トランザクション内でも文をまたいで見える確定データが変わることがある。
イ：トランザクション開始時のスナップショットだけを全期間固定して見る。
ウ：他トランザクションの未COMMIT更新も常に見える。
エ：全トランザクションが完全に直列実行される。

答え・解説 正答：ア

ア：READ COMMITTEDは未コミット値を避けつつ、文ごとに新しい確定状態を見る実装が一般的である。
イ：これはREPEATABLE READ系の説明に近い。
ウ：READ COMMITTEDの目的と異なる。
エ：SERIALIZABLEの目標に近い。
総合解説：READ COMMITTEDは未コミット値を避けつつ、文ごとに新しい確定状態を見る実装が一般的である。

対象：2026年度 / 基準日 2026-09-02

0074 3-3 トランザクション・排他制御 > 分離レベル・並行実行異常 ｜ 難易度：標準

REPEATABLE READを選ぶ主な目的として最も適切なものはどれか。
ア：他トランザクションの未確定値を積極的に読む。
イ：同一トランザクション内で読み取ったデータの一貫した見え方をREAD COMMITTEDより強く保つ。
ウ：全てのロックを無効化する。
エ：COMMIT済みデータも見えなくする。

答え・解説 正答：イ

ア：分離を弱める方向である。
イ：REPEATABLE READは同じトランザクション中の再読結果の安定性を高める。具体的なファントムの扱いはDBMS実装も確認する。
ウ：分離レベルはロック無効化の指定ではない。
エ：そのような目的ではない。
総合解説：REPEATABLE READは同じトランザクション中の再読結果の安定性を高める。具体的なファントムの扱いはDBMS実装も確認する。

対象：2026年度 / 基準日 2026-09-02

0075 3-3 トランザクション・排他制御 > 分離レベル・並行実行異常 ｜ 難易度：標準

SERIALIZABLE分離レベルが目指す性質として最も適切なものはどれか。
ア：全トランザクションを必ず物理的に1件ずつ順番に実行する。
イ：未確定データを全員が共有する。
ウ：並行実行の結果が、何らかの直列実行順序で得られる結果と同等になるようにする。
エ：COMMITを禁止する。

答え・解説 正答：ウ

ア：実装はロックやMVCC等で並行実行しつつ直列化可能性を保証し得る。
イ：分離とは逆である。
ウ：直列化可能性は並行実行を許しながら論理的に直列実行と同等の整合性を目指す。
エ：COMMITは必要である。
総合解説：直列化可能性は並行実行を許しながら論理的に直列実行と同等の整合性を目指す。

対象：2026年度 / 基準日 2026-09-02

0076 3-3 トランザクション・排他制御 > 分離レベル・並行実行異常 ｜ 難易度：標準

一般に、より強いトランザクション分離を選択するときに考慮すべきトレードオフはどれか。
ア：分離を強くすると必ずI/Oが0になる。
イ：分離を強くすると制約が全て不要になる。
ウ：分離レベルはSELECT構文の書式だけに影響する。
エ：整合性を高めやすい一方で、待機・競合・再実行などにより同時実行性や性能へ影響する場合がある。

答え・解説 正答：エ

ア：そのような保証はない。
イ：制約と分離は役割が異なる。
ウ：並行実行時の可視性・競合制御に関わる。
エ：分離レベルは整合性要求と性能・競合のバランスで選ぶ。
総合解説：分離レベルは整合性要求と性能・競合のバランスで選ぶ。

対象：2026年度 / 基準日 2026-09-02

0077 3-3 トランザクション・排他制御 > 分離レベル・並行実行異常 ｜ 難易度：標準

在庫数1の商品に対して二つの注文処理が同時に「在庫1」を読み、それぞれ在庫0を書き込むと、二件とも成功したように見える危険がある。最も適切な対策の考え方はどれか。

ア：条件付きUPDATEや適切な行ロック、分離レベルを用い、競合をDB側で検出・直列化する。
イ：二つの処理を別々のDBに無条件で書く。
ウ：在庫列を文字列にする。
エ：SELECT結果をキャッシュして長時間使う。

答え・解説 正答：ア

ア：在庫のような競合更新では「読んでから無条件更新」ではなく、原子的条件更新や排他制御を設計する。
イ：整合性問題を増やす。
ウ：並行制御にはならない。
エ：古い値で更新する危険が高まる。
総合解説：在庫のような競合更新では「読んでから無条件更新」ではなく、原子的条件更新や排他制御を設計する。

対象：2026年度 / 基準日 2026-09-02

0078 3-3 トランザクション・排他制御 > 分離レベル・並行実行異常 ｜ 難易度：標準

「当直医は最低1人必要」という制約をアプリケーションが二つの行の読取りで確認している。二人の医師が同時に互いを当直中と見て、自分の行だけを当直外へ更新すると、最終的に0人になる可能性がある。この問題への対策として最も適切な考え方はどれか。

ア：各医師が別行を更新しているので並行制御は不要である。
イ：複数行にまたがる制約を並行実行下でも守れるよう、SERIALIZABLE等の適切な分離や明示ロック、制約設計を検討する。
ウ：NULLを禁止すれば必ず防げる。
エ：ORDER BYで医師名を並べれば防げる。

答え・解説 正答：イ

ア：複数行にまたがる業務制約が破られ得る。
イ：個別行が競合しなくても、読取り集合と書込み集合の関係で業務制約が破られることがある。
ウ：NULL制約は当直人数の整合性を保証しない。
エ：並び順は並行制御ではない。
総合解説：個別行が競合しなくても、読取り集合と書込み集合の関係で業務制約が破られることがある。

対象：2026年度 / 基準日 2026-09-02

0079 3-3 トランザクション・排他制御 > 分離レベル・並行実行異常 ｜ 難易度：応用

SERIALIZABLEを実装するDBMSで、整合性を守るため一方のトランザクションが直列化失敗として中止される場合がある。アプリケーション設計として適切なのはどれか。

ア：失敗した文だけを必ず無条件でCOMMITする。
イ：直列化失敗は無視し、成功扱いにする。
ウ：エラーを検出し、トランザクション全体を安全に再実行できるよう設計する。
エ：全ての制約を削除する。

答え・解説 正答：ウ

ア：中止されたトランザクションの整合性を損なう。
イ：更新が反映されていない可能性がある。
ウ：強い分離では競合時に待機だけでなく中止・再試行が必要になる実装がある。
エ：根本的な対策にならない。
総合解説：強い分離では競合時に待機だけでなく中止・再試行が必要になる実装がある。

対象：2026年度 / 基準日 2026-09-02

0080 3-3 トランザクション・排他制御 > 分離レベル・並行実行異常 ｜ 難易度：応用

長時間の分析処理で同一トランザクション中に一貫したスナップショットが必要だが、更新系OLTPの停止は避けたい。設計判断として最も適切なものはどれか。

ア：必ず全表を排他ロックして更新を停止する。
イ：READ UNCOMMITTEDを選べば一貫性が最も高い。
ウ：トランザクションを使わず各SELECTを完全に独立実行すれば同じスナップショットになる。
エ：必要な一貫性を満たすスナップショット系の分離レベルを選び、DBMSのMVCC特性や長時間スナップショットの資源影響も評価する。

答え・解説 正答：エ

ア：必要以上に同時実行性を損なう可能性がある。
イ：一般に最も弱い分離である。
ウ：各文の見える確定状態が変わる可能性がある。
エ：分離レベルは業務要件だけでなくDBMS実装、MVCCのバージョン保持、更新競合への影響を合わせて評価する。
総合解説：分離レベルは業務要件だけでなくDBMS実装、MVCCのバージョン保持、更新競合への影響を合わせて評価する。

対象：2026年度 / 基準日 2026-09-02

0081 3-3 トランザクション・排他制御 > ロック・デッドロック ｜ 難易度：基礎

共有ロック（Sロック）と排他ロック（Xロック）の一般的な関係として最も適切なものはどれか。
ア：複数のSロックは共存できることが多いが、Xロックは競合する他ロックを排除して更新を保護する。
イ：Sロックは必ず更新専用で、Xロックは読取り専用である。
ウ：SロックとXロックは常に完全互換で待機を生じない。
エ：Xロックは他トランザクションの全データベース操作を必ず停止する。

答え・解説 正答：ア

ア：読取り共有と更新排他の基本的な互換性を理解する。
イ：役割が逆である。
ウ：更新競合を防げないため誤りである。
エ：通常はロック粒度に応じた対象範囲で競合する。
総合解説：読取り共有と更新排他の基本的な互換性を理解する。

対象：2026年度 / 基準日 2026-09-02

0082 3-3 トランザクション・排他制御 > ロック・デッドロック ｜ 難易度：基礎

デッドロックの説明として最も適切なものはどれか。
ア：一つのトランザクションがCPUを100%使う状態。
イ：複数トランザクションが互いに相手の保持する資源解放を待ち、循環待ちとなって進めない状態。
ウ：同じSELECTを二回実行すること。
エ：全トランザクションが同時にCOMMITする状態。

答え・解説 正答：イ

ア：高負荷だがデッドロックの定義ではない。
イ：循環待ちがデッドロックの中心である。
ウ：デッドロックとは無関係である。
エ：待ち循環ではない。
総合解説：循環待ちがデッドロックの中心である。

対象：2026年度 / 基準日 2026-09-02

0083 3-3 トランザクション・排他制御 > ロック・デッドロック ｜ 難易度：基礎

二つのテーブルA,Bを両方更新するトランザクションが多数ある。デッドロック発生を減らす基本策として最も適切なものはどれか。

ア：処理ごとにA→BとB→Aをランダムに変える。
イ：ロックを長時間保持する。
ウ：全トランザクションでA→Bのようにロック取得順序を統一する。
エ：全ての更新を無条件に再帰SQLへ変える。

答え・解説 正答：ウ

ア：循環待ちが起きやすくなる。
イ：競合時間を増やす可能性がある。
ウ：資源取得順序を統一すると循環待ちを形成しにくくなる。
エ：ロック順序問題の解決にならない。
総合解説：資源取得順序を統一すると循環待ちを形成しにくくなる。

対象：2026年度 / 基準日 2026-09-02

0084 3-3 トランザクション・排他制御 > ロック・デッドロック ｜ 難易度：標準

行ロックと表ロックの一般的なトレードオフとして最も適切なものはどれか。

ア：表ロックは必ず行ロックより同時実行性が高い。
イ：行ロックは管理が不要なのでオーバーヘッド0である。
ウ：ロック粒度は性能や競合に影響しない。
エ：細粒度ロックは同時実行性を高めやすい一方、ロック管理数やオーバーヘッドが増えることがある。

答え・解説 正答：エ

ア：広い範囲をロックするため競合しやすい。
イ：多数ロックの管理コストがある。
ウ：重要な設計要素である。
エ：ロック粒度は同時実行性と管理コストのトレードオフで選ぶ。
総合解説：ロック粒度は同時実行性と管理コストのトレードオフで選ぶ。

対象：2026年度 / 基準日 2026-09-02

0085 3-3 トランザクション・排他制御 > ロック・デッドロック ｜ 難易度：標準

競合確率が高く、更新前に対象行を確保して他更新を待たせたい場合に適する考え方はどれか。
ア：更新予定行を明示的にロックする悲観的排他制御を検討する。
イ：ロックを一切使わず常に最後の書込みを採用する。
ウ：全ての列をNULLにして競合を避ける。
エ：ORDER BYで更新順を表示するだけにする。

答え・解説 正答：ア

ア：競合が多い場合、先にロックを取得して競合を直列化する方法が有効なことがある。
イ：更新消失の危険がある。
ウ：業務データを破壊する。
エ：実際の排他制御にはならない。
総合解説：競合が多い場合、先にロックを取得して競合を直列化する方法が有効なことがある。

対象：2026年度 / 基準日 2026-09-02

0086 3-3 トランザクション・排他制御 > ロック・デッドロック ｜ 難易度：標準

競合が比較的少ないWebアプリで、version列を使ってUPDATE時に「読み取ったversionと同じなら更新し、versionを+1する」方式を採用した。この方式の主な目的はどれか。
ア：全行を物理的に表ロックする。
イ：他処理による先行更新を検出し、更新消失を防ぐ楽観的排他制御を行う。
ウ：主キーを自動採番する。
エ：SELECT結果を暗号化する。

答え・解説 正答：イ

ア：version列方式はロック保持を最小化して競合検出する考え方である。
イ：version条件が不一致なら競合があったことを検出できる。
ウ：version列の目的は採番ではなく競合検出である。
エ：排他制御とは無関係である。
総合解説：version条件が不一致なら競合があったことを検出できる。

対象：2026年度 / 基準日 2026-09-02

0087 3-3 トランザクション・排他制御 > ロック・デッドロック ｜ 難易度：標準

更新トランザクションでユーザー入力待ちを含め、ロック取得後に数分間停止する設計を避けるべき主な理由はどれか。

ア：ロック中はCPUが必ず停止するから。
イ：COMMITが構文エラーになるから。
ウ：ロック保持時間が長くなり、他トランザクションの待機やデッドロック機会を増やすから。
エ：SELECTが一切使えなくなるから。

答え・解説 正答：ウ

ア：CPU全体が停止するわけではない。
イ：長時間でもCOMMIT構文自体は有効である。
ウ：対話待ちをトランザクション内に長く含めない設計が重要である。
エ：ロックモードと対象により読取り可否は異なる。
総合解説：対話待ちをトランザクション内に長く含めない設計が重要である。

対象：2026年度 / 基準日 2026-09-02

0088 3-3 トランザクション・排他制御 > ロック・デッドロック ｜ 難易度：標準

DBMSがデッドロックを検出し、一方のトランザクションを中止して循環待ちを解消した。アプリケーション側の対応として最も適切なものはどれか。

ア：エラーを無視して更新成功として扱う。
イ：中止されたトランザクションの一部結果だけをCOMMITする。
ウ：全ユーザーの表定義を削除する。
エ：中止を検出し、必要ならトランザクション全体を安全に再試行する。

答え・解説 正答：エ

ア：実際には更新が取り消されている可能性がある。
イ：原子性を損なう。
ウ：対策として不適切である。
エ：デッドロックは完全にゼロにできない場合があり、被害者として中止された処理の再実行設計が必要である。
総合解説：デッドロックは完全にゼロにできない場合があり、被害者として中止された処理の再実行設計が必要である。

対象：2026年度 / 基準日 2026-09-02

0089 3-3 トランザクション・排他制御 > ロック・デッドロック ｜ 難易度：応用

大量の行ロックを保持した結果、DBMSがより粗い表ロックへ昇格する実装を想定する。性能影響として最も適切なものはどれか。

ア：ロック管理コストを減らせる一方、ロック範囲が広がり他トランザクションとの競合が増える可能性がある。
イ：昇格すると必ず全競合がなくなる。
ウ：昇格すると必ずデータが削除される。
エ：昇格はSQLの列名を変更する機能である。

答え・解説 正答：ア

ア：ロック昇格は管理資源と同時実行性のトレードオフを伴う。
イ：範囲が広がることで競合が増えることがある。
ウ：ロック粒度変更でデータ削除は起こらない。
エ：排他制御の機能である。
総合解説：ロック昇格は管理資源と同時実行性のトレードオフを伴う。

対象：2026年度 / 基準日 2026-09-02

0090 3-3 トランザクション・排他制御 > ロック・デッドロック ｜ 難易度：応用

「残高が負の口座は存在しない」という範囲条件を確認後、別トランザクションのINSERTで条件該当行が追加されることまで防ぎたい。単一既存行のロックだけで不十分な理由として最も適切なものはどれか。

ア：存在しない行は必ず主キー0としてロックされるから。
イ：まだ存在しない将来行は既存行ロックでは捕捉できず、範囲/述語を保護する仕組みやSERIALIZABLE等が必要になるから。
ウ：INSERTはトランザクションの対象外だから。
エ：SELECTは常に全表排他ロックを取るから。

答え・解説 正答：イ

ア：そのような一般則はない。
イ：ファントムを防ぐには個別行だけでなく検索条件の範囲を保護する考え方が必要になる。
ウ：INSERTもトランザクション内の更新である。
エ：一般的ではなく、MVCC等では読取りが更新を直接ブロックしないことも多い。
総合解説：ファントムを防ぐには個別行だけでなく検索条件の範囲を保護する考え方が必要になる。

対象：2026年度 / 基準日 2026-09-02

0091 3-4 障害回復 > ログ・WAL ｜ 難易度：基礎

WAL（Write-Ahead Logging）の基本原則として最も適切なものはどれか。

ア：データページの変更を永続化する前に、その変更を表すログレコードを永続化する。
イ：データページを永続化した後にだけログを書き込む。
ウ：COMMIT後はログを残さず、更新済みページだけを保存する。
エ：ログは参照処理だけを記録し、更新処理は記録しない。

答え・解説 正答：ア

ア：WALでは変更内容を表すログを先に安定記憶へ書き、後からデータページを書けるようにする。
イ：順序が逆ではクラッシュ時に未記録の更新がデータだけに残り、回復根拠を失う。
ウ：COMMIT済み更新をREDOできるようログが必要である。
エ：障害回復に必要なのは主として更新に関するログである。
総合解説：WALでは変更内容を表すログを先に安定記憶へ書き、後からデータページを書けるようにする。 障害回復では、ログの永続化順序とトランザクションの確定状態を基に、REDOとUNDOの要否を判断する。

対象：2026年度 / 基準日 2026-09-02

0092 3-4 障害回復 > ログ・WAL ｜ 難易度：基礎

障害回復におけるREDOとUNDOの説明として最も適切なものはどれか。

ア：REDOは未コミット更新だけを取消し、UNDOはコミット済み更新だけを再適用する。
イ：REDOはコミット済み更新の再適用に、UNDOは未コミット更新の取消しに用いられる。
ウ：REDOとUNDOはいずれも索引の再作成だけに用いる。
エ：REDOはバックアップ取得、UNDOは統計情報更新を意味する。

答え・解説 正答：イ

ア：役割が逆である。
イ：一般的なログ回復ではREDOで確定済み変更を反映し、UNDOで未確定変更を取り消す。
ウ：索引も回復対象になり得るが、REDO/UNDOの定義は索引再作成に限定されない。
エ：バックアップや統計更新とは別概念である。
総合解説：一般的なログ回復ではREDOで確定済み変更を反映し、UNDOで未確定変更を取り消す。 障害回復では、ログの永続化順序とトランザクションの確定状態を基に、REDOとUNDOの要否を判断する。

対象：2026年度 / 基準日 2026-09-02

0093 3-4 障害回復 > ログ・WAL ｜ 難易度：基礎

更新トランザクションTがCOMMIT成功を利用者へ通知する条件として、WAL方式で特に重要なものはどれか。

ア：Tが更新した全データページが必ずデータファイルへ書き出されていること。
イ：Tが参照した全ページがバッファから破棄されていること。
ウ：Tのコミットを保証するのに必要なログが安定記憶へ書き出されていること。
エ：データベース全体のバックアップが完了していること。

答え・解説 正答：ウ

ア：no-force方式ではCOMMIT時に全変更ページを必ず書き出す必要はない。
イ：参照ページの破棄はコミット保証の条件ではない。
ウ：WALでは必要なログを先に永続化すれば、データページ自体は後で書き出してもREDOで回復できる。
エ：各トランザクションのコミットごとに全体バックアップは不要である。
総合解説：WALでは必要なログを先に永続化すれば、データページ自体は後で書き出してもREDOで回復できる。 障害回復では、ログの永続化順序とトランザクションの確定状態を基に、REDOとUNDOの要否を判断する。

対象：2026年度 / 基準日 2026-09-02

0094 3-4 障害回復 > ログ・WAL ｜ 難易度：標準

ある更新ログに「更新前の値」と「更新後の値」の両方が記録されている。この方式の利点として最も適切なものはどれか。

ア：更新前値だけで常にREDOとUNDOの両方を実行できる。
イ：更新後値だけで常にUNDOを実行できる。
ウ：両方の値を記録すると障害回復はできなくなる。
エ：更新前値をUNDOに、更新後値をREDOに利用できる。

答え・解説 正答：エ

ア：更新前値だけでは一般に更新後状態の再現に十分でない。
イ：更新後値だけでは一般に元の状態へ戻す情報がない。
ウ：両方を持つログはUNDO/REDO型回復の典型的な考え方である。
エ：before imageは取消し、after imageは再適用の根拠にできる。
総合解説：before imageは取消し、after imageは再適用の根拠にできる。 障害回復では、ログの永続化順序とトランザクションの確定状態を基に、REDOとUNDOの要否を判断する。

対象：2026年度 / 基準日 2026-09-02

0095 3-4 障害回復 > ログ・WAL ｜ 難易度：標準

バッファ管理がstealかつno-forceであるDBMSを考える。障害回復で必要になりやすい処理の組合せとして最も適切なものはどれか。

ア：未コミット更新がディスクに出る可能性があるのでUNDOが必要で、コミット済み更新が未書込みの可能性があるのでREDOも必要である。
イ：stealなのでREDOだけ必要で、UNDOは不要である。
ウ：no-forceなのでUNDOだけ必要で、REDOは不要である。
エ：stealとno-forceを併用すればログは不要になる。

答え・解説 正答：ア

ア：stealは未コミットページの書出しを許すためUNDO、no-forceはコミット時の全ページ書出しを強制しないためREDOが必要になる。
イ：stealにより未コミット更新が媒体へ出る可能性があるのでUNDOも要る。
ウ：no-forceでは確定済み変更がデータファイルに未反映のことがあるのでREDOも要る。
エ：むしろ両方の回復を支えるログが重要になる。
総合解説：stealは未コミットページの書出しを許すためUNDO、no-forceはコミット時の全ページ書出しを強制しないためREDOが必要になる。 障害回復では、ログの永続化順序とトランザクションの確定状態を基に、REDOとUNDOの要否を判断する。

対象：2026年度 / 基準日 2026-09-02

0096 3-4 障害回復 > ログ・WAL ｜ 難易度：標準

物理ログの説明として最も適切なものはどれか。

ア：「全社員の給与を5%増額」のような高水準操作だけを記録する。
イ：ページ内のどの位置をどの値へ変更したかなど、物理的な変更内容を記録する。
ウ：SQL文の文字列だけを保存し、ページ変更情報を一切持たない方式に限定される。
エ：バックアップ媒体の保管場所だけを記録する。

答え・解説 正答：イ

ア：これは論理ログに近い説明である。
イ：物理ログはページやレコード上の具体的な変更位置・内容を記録する考え方である。
ウ：物理ログをSQL文字列だけに限定するのは誤りである。
エ：媒体管理記録はトランザクション回復ログとは別である。
総合解説：物理ログはページやレコード上の具体的な変更位置・内容を記録する考え方である。 障害回復では、ログの永続化順序とトランザクションの確定状態を基に、REDOとUNDOの要否を判断する。

対象：2026年度 / 基準日 2026-09-02

0097 3-4 障害回復 > ログ・WAL ｜ 難易度：標準

ログレコードに単調増加するLSN（Log Sequence Number）を付与する主な目的として最も適切なものはどれか。

ア：各SQL文の実行時間を必ず秒単位で表す。
イ：ユーザーIDを暗号化する。
ウ：ログ中の位置や順序を識別し、どこまでのログが適用済みかなどを追跡しやすくする。
エ：テーブルの主キー値の代わりに必ず用いる。

答え・解説 正答：ウ

ア：実行時間そのものを表す値ではない。
イ：認証・暗号化のための識別子ではない。
ウ：LSNはログの順序・位置を識別するための値で、回復やレプリケーション進捗の基準にも使われる。
エ：業務データの主キーとは役割が異なる。
総合解説：LSNはログの順序・位置を識別するための値で、回復やレプリケーション進捗の基準にも使われる。 障害回復では、ログの永続化順序とトランザクションの確定状態を基に、REDOとUNDOの要否を判断する。

対象：2026年度 / 基準日 2026-09-02

0098 3-4 障害回復 > ログ・WAL ｜ 難易度：標準

ベースバックアップ取得後の更新をログアーカイブで保全している。途中のログ区間が失われた場合の説明として最も適切なものはどれか。

ア：後続ログがあれば欠損区間は常に自動推測できる。
イ：ベースバックアップがあれば、ログ欠損はポイントインタイム回復に全く影響しない。
ウ：ログはCOMMIT済みトランザクションの回復には使われないので影響しない。
エ：失われた区間より後のログが残っていても、通常はその区間をまたいだ連続的なロールフォワード回復はできない。

答え・解説 正答：エ

ア：更新内容を一般に推測で補完することはできない。
イ：ベースバックアップ後の時点へ進めるには対応するログが必要である。
ウ：COMMIT済み更新のREDOこそログの重要用途である。
エ：ログ連鎖に欠損があると、その先へ正しく状態遷移を再現できないのが通常である。
総合解説：ログ連鎖に欠損があると、その先へ正しく状態遷移を再現できないのが通常である。 障害回復では、ログの永続化順序とトランザクションの確定状態を基に、REDOとUNDOの要否を判断する。

対象：2026年度 / 基準日 2026-09-02

0099 3-4 障害回復 > ログ・WAL ｜ 難易度：応用

クラッシュ後、あるデータページには更新Xが反映されているが、安定記憶上のログには更新Xの記録が存在しなかった。WAL方式の観点から最も適切な評価はどれか。

ア：ログ先行書込みの条件が破られており、UNDO/REDOの根拠を欠く危険な状態である。
イ：WALではデータページを先に書くことが必須なので正常である。
ウ：COMMIT済みかどうかに関係なくログは不要なので問題ない。
エ：この状態は索引統計だけを再収集すれば必ず解消する。

答え・解説 正答：ア

ア：データ変更が永続化される前に対応ログが永続化されることがWALの核心である。
イ：順序が逆である。
ウ：ログなしでは障害回復の正当な根拠を失う。
エ：統計情報は回復ログの欠損を補えない。
総合解説：データ変更が永続化される前に対応ログが永続化されることがWALの核心である。 障害回復では、ログの永続化順序とトランザクションの確定状態を基に、REDOとUNDOの要否を判断する。

対象：2026年度 / 基準日 2026-09-02

0100 3-4 障害回復 > ログ・WAL ｜ 難易度：応用

障害発生時、T1はCOMMIT記録までログに残っているがデータページの一部が未反映、T2は更新ログがあるがCOMMIT記録がない。一般的なUNDO/REDO型回復で最も適切な方針はどれか。

ア：T1とT2をともにUNDOする。
イ：T1をREDOし、T2をUNDOする。
ウ：T1とT2をともにREDOする。
エ：T1をUNDOし、T2は何もしない。

答え・解説 正答：イ

ア：T1は確定済みなので取り消してはいけない。
イ：COMMIT済みT1は永続性を満たすためREDO、未コミットT2は原子性を満たすためUNDOの対象になる。
ウ：未コミットT2をREDOすると未確定変更が残る。
エ：T1は確定済みでありUNDO対象ではなく、T2は取消しが必要である。
総合解説：COMMIT済みT1は永続性を満たすためREDO、未コミットT2は原子性を満たすためUNDOの対象になる。 障害回復では、ログの永続化順序とトランザクションの確定状態を基に、REDOとUNDOの要否を判断する。

対象：2026年度 / 基準日 2026-09-02

0101 3-4 障害回復 > チェックポイント・ロールフォワード／ロールバック ｜ 難易度：基礎

チェックポイントを設ける主な目的として最も適切なものはどれか。

ア：すべてのトランザクションを永久に停止する。
イ：データベースの正規化を自動的に行う。
ウ：障害回復時に調査・再適用すべきログ範囲を小さくし、回復時間を抑える。
エ：ユーザー認証情報を暗号化する。

答え・解説 正答：ウ

ア：通常運用を永久停止する仕組みではない。
イ：論理設計の正規化とは別である。
ウ：チェックポイントは回復開始位置の目安を作り、クラッシュ後に最初から全ログを走査する負担を減らす。
エ：認証・暗号化機能ではない。
総合解説：チェックポイントは回復開始位置の目安を作り、クラッシュ後に最初から全ログを走査する負担を減らす。 チェックポイントは回復範囲を制御し、ロールフォワードはログ再適用、ロールバックは更新取消しという役割を持つ。

対象：2026年度 / 基準日 2026-09-02

0102 3-4 障害回復 > チェックポイント・ロールフォワード／ロールバック ｜ 難易度：基礎

バックアップを復元した後、そのバックアップ取得後の更新ログを順に再適用して障害直前へ近づける処理はどれか。

ア：ロールバック
イ：正規化
ウ：デフラグ
エ：ロールフォワード

答え・解説 正答：エ

ア：ロールバックは更新を取り消して前の状態へ戻す処理である。
イ：正規化はデータモデル設計手法である。
ウ：デフラグは記憶領域の断片化対策で、ログ回復の名称ではない。
エ：ロールフォワードはバックアップ後のログを再適用して状態を前へ進める回復処理である。
総合解説：ロールフォワードはバックアップ後のログを再適用して状態を前へ進める回復処理である。 チェックポイントは回復範囲を制御し、ロールフォワードはログ再適用、ロールバックは更新取消しという役割を持つ。

対象：2026年度 / 基準日 2026-09-02

0103 3-4 障害回復 > チェックポイント・ロールフォワード／ロールバック ｜ 難易度：基礎

未コミットトランザクションが行った更新を取り消し、開始前または指定した保存点の状態へ戻す処理はどれか。

ア：ロールバック
イ：ロールフォワード
ウ：チェックポイント
エ：バックアップ取得

答え・解説 正答：ア

ア：ロールバックはトランザクションの変更を取り消す操作である。
イ：ロールフォワードはログの再適用で状態を先へ進める。
ウ：チェックポイントは回復基準点の管理である。
エ：バックアップ取得は複製を保存する処理である。
総合解説：ロールバックはトランザクションの変更を取り消す操作である。 チェックポイントは回復範囲を制御し、ロールフォワードはログ再適用、ロールバックは更新取消しという役割を持つ。

対象：2026年度 / 基準日 2026-09-02

0104 3-4 障害回復 > チェックポイント・ロールフォワード／ロールバック ｜ 難易度：標準

チェックポイント間隔を極端に短くした場合に一般に考慮すべきトレードオフとして最も適切なものはどれか。

ア：障害回復時のREDO量も平常時I/Oも必ず無限に増える。
イ：障害回復時のREDO量は減りやすい一方、平常時の書込みI/O負荷が増える可能性がある。
ウ：チェックポイント間隔は性能にも回復時間にも影響しない。
エ：短くするほど索引の選択度が必ず高くなる。

答え・解説 正答：イ

ア：両方が必ず増えるという関係ではない。
イ：頻繁なチェックポイントは回復範囲を小さくできるが、ダーティページ書出しなどの負荷を増やし得る。
ウ：回復時間とI/O特性に影響する重要な運用パラメータである。
エ：索引選択度とは無関係である。
総合解説：頻繁なチェックポイントは回復範囲を小さくできるが、ダーティページ書出しなどの負荷を増やし得る。 チェックポイントは回復範囲を制御し、ロールフォワードはログ再適用、ロールバックは更新取消しという役割を持つ。

対象：2026年度 / 基準日 2026-09-02

0105 3-4 障害回復 > チェックポイント・ロールフォワード／ロールバック ｜ 難易度：標準

最新チェックポイントより前の変更はすべてデータファイルへ反映済みであることが保証される方式を考える。クラッシュ回復で主に調べるべき範囲として最も適切なものはどれか。
ア：システム稼働開始以来の全ログを必ず最初から再適用する。
イ：バックアップ媒体のファイル名だけを調べる。
ウ：最新チェックポイント以後のログを中心に調べる。
エ：SQLのDDL文だけを調べ、DMLログは無視する。

答え・解説 正答：ウ

ア：チェックポイントの目的の一つは全履歴走査を避けることである。
イ：媒体名だけではトランザクション状態を判定できない。
ウ：この前提ならチェックポイント以前の変更は反映済みなので、以後のログが主要な回復対象となる。
エ：更新回復にはDMLに対応するログも重要である。
総合解説：この前提ならチェックポイント以前の変更は反映済みなので、以後のログが主要な回復対象となる。 チェックポイントは回復範囲を制御し、ロールフォワードはログ再適用、ロールバックは更新取消しという役割を持つ。

対象：2026年度 / 基準日 2026-09-02

0106 3-4 障害回復 > チェックポイント・ロールフォワード／ロールバック ｜ 難易度：標準

SQLのSAVEPOINTとDBMSのチェックポイントの違いとして最も適切なものはどれか。
ア：両者は常に同一機能で名称だけが違う。
イ：SAVEPOINTはデータベース全体のバックアップを作成する。
ウ：チェックポイントはSQLの検索結果の並び順を保存する。
エ：SAVEPOINTは一つのトランザクション内で部分ロールバック位置を示し、チェックポイントは主にシステム障害回復の基準点を管理する。

答え・解説 正答：エ

ア：トランザクション内制御とシステム回復管理は別機能である。
イ：SAVEPOINTは全体バックアップではない。
ウ：並び順保存の仕組みではない。
エ：両者は「戻る位置」に見えてもスコープと目的が異なる。
総合解説：両者は「戻る位置」に見えてもスコープと目的が異なる。 チェックポイントは回復範囲を制御し、ロールフォワードはログ再適用、ロールバックは更新取消しという役割を持つ。

対象：2026年度 / 基準日 2026-09-02

0107 3-4 障害回復 > チェックポイント・ロールフォワード／ロールバック ｜ 難易度：標準

データファイルを格納したディスクが物理故障し、ファイル自体を失った。この場合の回復方針として最も適切なものはどれか。

ア：健全なバックアップを復元し、必要に応じてその後のログをロールフォワードする。
イ：メモリ上のバッファだけから必ず完全復旧する。
ウ：統計情報を更新するだけで失われたデータファイルが復元される。
エ：未コミットトランザクションをROLLBACKするだけで媒体の内容が戻る。

答え・解説 正答：ア

ア：媒体障害では元データファイルが失われるため、バックアップとログによる復旧が基本となる。
イ：物理媒体喪失では揮発性バッファだけに依存できない。
ウ：統計は実データの代替ではない。
エ：ROLLBACKだけでは失われた確定済みデータを再構成できない。
総合解説：媒体障害では元データファイルが失われるため、バックアップとログによる復旧が基本となる。 チェックポイントは回復範囲を制御し、ロールフォワードはログ再適用、ロールバックは更新取消しという役割を持つ。

対象：2026年度 / 基準日 2026-09-02

0108 3-4 障害回復 > チェックポイント・ロールフォワード／ロールバック ｜ 難易度：標準

高可用なDBMSで、チェックポイント中も更新処理を完全停止せずに進める方式を採用する理由として最も適切なものはどれか。

ア：更新を停止しない場合はログが一切不要になるためである。
イ：平常時の停止時間を抑えつつ、ログとページ状態を管理して回復可能な基準点を作るためである。
ウ：チェックポイント中の全トランザクションを自動的にCOMMITするためである。
エ：索引を必ず削除して再作成するためである。

答え・解説 正答：イ

ア：更新継続時こそ整合的なログ管理が重要である。
イ：ファジーなチェックポイントはオンライン性を保ちながら回復情報を整える考え方である。
ウ：未完了トランザクションを勝手に確定させる機能ではない。
エ：索引再作成が本質ではない。
総合解説：ファジーなチェックポイントはオンライン性を保ちながら回復情報を整える考え方である。チェックポイントは回復範囲を制御し、ロールフォワードはログ再適用、ロールバックは更新取消しという役割を持つ。

対象：2026年度 / 基準日 2026-09-02

0109 3-4 障害回復 > チェックポイント・ロールフォワード／ロールバック ｜ 難易度：応用

日曜0時にフルバックアップ、月〜水の更新ログを連続保管していた。水曜15時に媒体障害が発生した。水曜14時55分まで復旧したい。最も適切な手順はどれか。

ア：水曜のログだけを空のデータベースへ適用する。
イ：日曜のバックアップだけを復元し、ログは使わない。
ウ：日曜のバックアップを復元し、月曜以降のログを時系列順に14時55分まで適用する。
エ：月曜のログを逆順に適用してから日曜バックアップを復元する。

答え・解説 正答：ウ

ア：ログは通常、基礎となるデータ状態なしに完全なDBを構成するものではない。
イ：日曜以後の確定更新が失われる。
ウ：ベースとなるバックアップを先に復元し、その状態から後続ログを順に再適用する。
エ：復元後にログを順方向へ適用するのが基本である。
総合解説：ベースとなるバックアップを先に復元し、その状態から後続ログを順に再適用する。 チェックポイントは回復範囲を制御し、ロールフォワードはログ再適用、ロールバックは更新取消しという役割を持つ。

対象：2026年度 / 基準日 2026-09-02

0110 3-4 障害回復 > チェックポイント・ロールフォワード／ロールバック ｜ 難易度：応用

クラッシュ後の回復時間が長く、調査すると毎回大量のREDOログを再適用している。更新性能には余裕がある。改善策として最初に検討するものとして最も適切なのはどれか。

ア：ログをすべて無効化して回復処理自体をなくす。
イ：主キー制約を削除してREDOを減らす。
ウ：統計情報を削除してオプティマイザの処理を軽くする。
エ：チェックポイント間隔や関連設定を見直し、回復開始点以後のREDO量を減らせるか評価する。

答え・解説 正答：エ

ア：ログ無効化は耐障害性を損なう。
イ：主キー制約はこの回復時間問題の直接原因とは限らない。
ウ：実行計画統計とクラッシュREDO量は別問題である。
エ：更新時I/Oとのバランスを確認しつつチェックポイント頻度を調整するのが合理的である。
総合解説：更新時I/Oとのバランスを確認しつつチェックポイント頻度を調整するのが合理的である。 チェックポイントは回復範囲を制御し、ロールフォワードはログ再適用、ロールバックは更新取消しという役割を持つ。

対象：2026年度 / 基準日 2026-09-02

0111 3-4 障害回復 > バックアップ・リストア・リカバリ ｜ 難易度：基礎

フルバックアップの説明として最も適切なものはどれか。
ア：対象データを基準時点で一式保存するバックアップである。
イ：直前のバックアップ以後に変更されたデータだけを必ず保存する。
ウ：障害発生後にだけ取得できるバックアップである。
エ：ログファイルだけを保存し、データ本体は保存しない。

答え・解説 正答：ア

ア：フルバックアップは対象全体を基準として保存する。
イ：これは増分バックアップの代表的説明である。
ウ：平常時に計画的に取得する。
エ：ログだけの保存とは異なる。
総合解説：フルバックアップは対象全体を基準として保存する。 バックアップ設計では、復元手順が実行可能か、ログ連続性、RPO/RTO、世代管理まで含めて評価する。

対象：2026年度 / 基準日 2026-09-02

0112 3-4 障害回復 > バックアップ・リストア・リカバリ ｜ 難易度：基礎

一般的な「増分バックアップ」と「差分バックアップ」の違いとして最も適切なものはどれか。
ア：増分は毎回全データ、差分はログを一切保存しない。
イ：増分は直前のバックアップ以後の変更分、差分は直近フルバックアップ以後の変更分を保存する。
ウ：増分は論理バックアップ、差分は必ず物理バックアップを意味する。
エ：両者は必ず同じデータ量になる。

答え・解説 正答：イ

ア：増分は通常、全データを毎回保存しない。
イ：一般的な定義では基準点が異なるため、復元手順とバックアップ量の特性も異なる。
ウ：論理/物理という別分類とは直結しない。
エ：変更履歴によってデータ量は異なる。
総合解説：一般的な定義では基準点が異なるため、復元手順とバックアップ量の特性も異なる。 バックアップ設計では、復元手順が実行可能か、ログ連続性、RPO/RTO、世代管理まで含めて評価する。

対象：2026年度 / 基準日 2026-09-02

0113 3-4 障害回復 > バックアップ・リストア・リカバリ ｜ 難易度：基礎

バックアップ運用で、取得成功メッセージの確認だけでなく定期的なリストア試験を行う主な理由はどれか。

ア：バックアップを復元すると本番DBの正規化が自動改善するため。
イ：リストア試験を行えば以後のバックアップが不要になるため。
ウ：実際に復元可能であり、必要な手順・時間・媒体が有効かを確認するため。
エ：統計情報の選択度を高めるため。

答え・解説 正答：ウ

ア：論理設計改善とは別目的である。
イ：継続的なバックアップは必要である。
ウ：バックアップは「取れた」だけでなく「戻せる」ことを確認して初めて有効性が高まる。
エ：オプティマイザ統計とは別である。
総合解説：バックアップは「取れた」だけでなく「戻せる」ことを確認して初めて有効性が高まる。バックアップ設計では、復元手順が実行可能か、ログ連続性、RPO/RTO、世代管理まで含めて評価する。

対象：2026年度 / 基準日 2026-09-02

0114 3-4 障害回復 > バックアップ・リストア・リカバリ ｜ 難易度：標準

「障害時に失ってよいデータは最大5分分まで」という要求に最も直接対応する指標はどれか。

ア：RTO（Recovery Time Objective）
イ：MTBF（Mean Time Between Failures）
ウ：TPS（Transactions Per Second）
エ：RPO（Recovery Point Objective）

答え・解説 正答：エ

ア：RTOはサービス復旧までの許容時間を表す。
イ：MTBFは故障間隔に関する信頼性指標である。
ウ：TPSは処理性能の指標である。
エ：RPOはどの時点までデータを戻せる必要があるか、許容データ損失量を時間で表す目標である。
総合解説：RPOはどの時点までデータを戻せる必要があるか、許容データ損失量を時間で表す目標である。 バックアップ設計では、復元手順が実行可能か、ログ連続性、RPO/RTO、世代管理まで含めて評価する。

対象：2026年度 / 基準日 2026-09-02

0115 3-4 障害回復 > バックアップ・リストア・リカバリ ｜ 難易度：標準

任意の時点へポイントインタイムリカバリ（PITR）したい。一般に必要な構成として最も適切なものはどれか。

ア：基礎となるバックアップと、その後の対象時点まで連続した更新ログ。
イ：最新の統計情報だけ。
ウ：主キー定義とビュー定義だけ。
エ：レプリカ1台だけでログ保全は不要。

答え・解説 正答：ア

ア：バックアップの状態から対象時点までログを順次適用することでPITRを実現する。
イ：統計情報だけではデータを再構成できない。
ウ：スキーマ定義だけでは更新後データを再現できない。
エ：レプリカは誤更新も複製し得るため、PITRの代替とは限らない。
総合解説：バックアップの状態から対象時点までログを順次適用することでPITRを実現する。 バックアップ設計では、復元手順が実行可能か、ログ連続性、RPO/RTO、世代管理まで含めて評価する。

対象：2026年度 / 基準日 2026-09-02

0116 3-4 障害回復 > バックアップ・リストア・リカバリ ｜ 難易度：標準

DBを稼働させたまま物理バックアップを取得する場合に重要な考え方として最も適切なものはどれか。

ア：取得中は各ファイルが異なる時点でもよく、回復時の整合化手段は不要である。
イ：取得中に更新が進んでも、ログ等を用いて復元時に一貫した状態へ整合させられる仕組みが必要である。
ウ：バックアップ中は必ず全利用者をログアウトさせなければオンラインバックアップとは呼べない。
エ：オンラインバックアップでは更新ログを保存してはならない。

答え・解説 正答：イ

ア：時点ずれを放置すると不整合なコピーになり得る。
イ：稼働中バックアップは取得期間中の更新を考慮し、復元時に整合した状態を構成できることが重要である。
ウ：完全停止を必須とするのはコールドバックアップ寄りの考え方である。
エ：ログは整合化・回復に重要である。
総合解説：稼働中バックアップは取得期間中の更新を考慮し、復元時に整合した状態を構成できることが重要である。 バックアップ設計では、復元手順が実行可能か、ログ連続性、RPO/RTO、世代管理まで含めて評価する。

対象：2026年度 / 基準日 2026-09-02

0117 3-4 障害回復 > バックアップ・リストア・リカバリ ｜ 難易度：標準

日曜にフル、月曜・火曜・水曜に「直前バックアップ以後の変更だけ」を保存する増分バックアップを取得した。水曜終了時点へ戻すのに必要な復元対象はどれか。
ア：日曜フル＋水曜増分だけ
イ：月曜増分＋水曜増分だけ
ウ：日曜フル＋月曜増分＋火曜増分＋水曜増分
エ：水曜増分だけ

答え・解説 正答：ウ

ア：火曜までの変更分が欠ける。
イ：基礎となるフルがない。
ウ：増分が直前バックアップを基準にする方式では、フル以後の各増分を順に適用する必要がある。
エ：増分単独では基礎状態と途中変更が欠ける。
総合解説：増分が直前バックアップを基準にする方式では、フル以後の各増分を順に適用する必要がある。 バックアップ設計では、復元手順が実行可能か、ログ連続性、RPO/RTO、世代管理まで含めて評価する。

対象：2026年度 / 基準日 2026-09-02

0118 3-4 障害回復 > バックアップ・リストア・リカバリ ｜ 難易度：標準

同期レプリケーションを行っているのでバックアップは不要だ、という主張への評価として最も適切なものはどれか。
ア：同期レプリケーションがあれば過去時点への復元も必ず無制限に可能である。
イ：レプリケーションは媒体障害対策にもならないので全く無意味である。
ウ：バックアップは性能測定専用なので可用性とは関係しない。
エ：誤DELETEや論理破壊も複製され得るため、世代を保持する独立バックアップは別途必要である。

答え・解説 正答：エ

ア：複製だけでは任意の過去世代が残るとは限らない。
イ：冗長化として媒体・ノード障害対策に有効である。
ウ：バックアップは復旧・保全の基本手段である。
エ：レプリカは可用性向上に有効だが、誤操作・ランサムウェア・論理破壊などに対する履歴保全はバックアップの役割である。
総合解説：レプリカは可用性向上に有効だが、誤操作・ランサムウェア・論理破壊などに対する履歴保全はバックアップの役割である。 バックアップ設計では、復元手順が実行可能か、ログ連続性、RPO/RTO、世代管理まで含めて評価する。

対象：2026年度 / 基準日 2026-09-02

0119 3-4 障害回復 > バックアップ・リストア・リカバリ ｜ 難易度：応用

24時間稼働DBでRPO=10分、RTO=30分が求められる。1日1回のフルバックアップだけでは不足する可能性が高い主な理由はどれか。

ア：障害時に最大約24時間分の更新を失い得てRPOを満たせず、復元時間もRTOを超える可能性があるため。
イ：フルバックアップはRPOを必ず0秒にするため過剰である。
ウ：RTOはデータ損失量だけを示すので復元時間は無関係である。
エ：RPOとRTOはSQL構文の種類なので運用設計とは無関係である。

答え・解説 正答：ア

ア：短いRPOには高頻度のログ保全等が、短いRTOには迅速な復元・冗長化などが必要になる。
イ：1日1回では障害時刻によって大きなデータ損失が生じ得る。
ウ：RTOは復旧時間目標である。
エ：どちらも事業継続・回復設計の指標である。
総合解説：短いRPOには高頻度のログ保全等が、短いRTOには迅速な復元・冗長化などが必要になる。バックアップ設計では、復元手順が実行可能か、ログ連続性、RPO/RTO、世代管理まで含めて評価する。

対象：2026年度 / 基準日 2026-09-02

0120 3-4 障害回復 > バックアップ・リストア・リカバリ ｜ 難易度：応用

バックアップファイルの破損が取得から数週間後に判明するリスクへ備える施策として最も適切なものはどれか。

ア：常に最新1世代だけを上書きし、検証は行わない。
イ：複数世代を独立保管し、チェックサム等の整合性確認と定期的な復元試験を行う。
ウ：バックアップ取得後すぐ本番データも削除する。
エ：バックアップファイル名を長くすれば破損を防げる。

答え・解説 正答：イ

ア：最新1世代だけでは潜在破損発見時に代替がない。
イ：世代分散と検証により、潜在破損や誤上書き時にも利用可能な復元点を残しやすい。
ウ：本番データ削除は可用性を下げる。
エ：名称はデータ完全性を保証しない。
総合解説：世代分散と検証により、潜在破損や誤上書き時にも利用可能な復元点を残しやすい。 バックアップ設計では、復元手順が実行可能か、ログ連続性、RPO/RTO、世代管理まで含めて評価する。

対象：2026年度 / 基準日 2026-09-02

0121 3-5 SQL性能設計 > インデックス設計 ｜ 難易度：基礎

一般的なB-tree索引が特に適している検索条件はどれか。
ア：大規模表の全行を必ず読み出す検索だけ
イ：画像データの内容を自動理解する検索
ウ：キー値の等価検索や範囲検索
エ：外部バックアップ媒体からの復元

答え・解説 正答：ウ

ア：全件走査しか行わないなら索引の利点は小さい。
イ：画像の意味検索は通常のB-treeの目的ではない。
ウ：B-treeは順序性を持つため等価・大小比較・範囲検索に広く利用される。
エ：バックアップ復元とは別機能である。
総合解説：B-treeは順序性を持つため等価・大小比較・範囲検索に広く利用される。 索引は検索性能だけでなく、選択度、複合キー順序、更新負荷、容量、実行計画まで含めて設計する。

対象：2026年度 / 基準日 2026-09-02

0122 3-5 SQL性能設計 > インデックス設計 ｜ 難易度：基礎

索引の「選択度」が高い状態の説明として最も適切なものはどれか。
ア：条件に一致する行がほぼ全行である。
イ：列の値が全てNULLであることを必ず意味する。
ウ：索引ページが暗号化されていることを意味する。
エ：条件に一致する行が全体のごく一部に絞られ、索引から少数行へ到達しやすい。

答え・解説 正答：エ

ア：ほぼ全行一致では索引経由のランダムアクセスが不利になりやすい。
イ：NULL率とは別概念である。
ウ：暗号化の有無とは関係しない。
エ：高選択度の条件は候補行を大きく絞り込むため、索引の恩恵を受けやすい。
総合解説：高選択度の条件は候補行を大きく絞り込むため、索引の恩恵を受けやすい。 索引は検索性能だけでなく、選択度、複合キー順序、更新負荷、容量、実行計画まで含めて設計する。

対象：2026年度 / 基準日 2026-09-02

0123 3-5 SQL性能設計 > インデックス設計 ｜ 難易度：基礎

複合B-tree索引 (customer_id, order_date) がある。一般にこの索引を最も直接利用しやすい条件はどれか。

ア：customer_id = ? AND order_date BETWEEN ? AND ?
イ：order_date BETWEEN ? AND ? だけ
ウ：商品の説明文に特定単語を含むかの全文検索
エ：テーブルの全列を条件なしで取得

答え・解説 正答：ア

ア：先頭列customer_idの等価条件に続きorder_dateの範囲条件を使う形は複合B-treeと相性がよい。
イ：先頭列を使わない条件では索引利用性が低下することが多い。
ウ：全文検索には専用索引等が適する。
エ：無条件全件取得では索引の絞込み効果がない。
総合解説：先頭列customer_idの等価条件に続きorder_dateの範囲条件を使う形は複合B-treeと相性がよい。 索引は検索性能だけでなく、選択度、複合キー順序、更新負荷、容量、実行計画まで含めて設計する。

対象：2026年度 / 基準日 2026-09-02

0124 3-5 SQL性能設計 > インデックス設計 ｜ 難易度：基礎

1億行の表で status 列が「有効」「無効」の2値しかなく、有効が95%を占める。status="有効" の検索を高速化するため単独B-tree索引を作る案の評価として最も適切なものはどれか。

ア：2値列なら必ずB-tree索引だけで100倍高速になる。
イ：大量行が一致するため、単独索引が常に有利とは限らず、実行計画や他条件との複合設計を評価すべきである。
ウ：値の種類が少ないほど通常のB-tree索引は必ず最適になる。
エ：索引を作れば表の更新コストは必ず減る。

答え・解説 正答：イ

ア：索引効果は選択度やI/O特性に依存する。
イ：95%一致では表走査の方が効率的な場合もあり、実データ分布と条件全体で評価する必要がある。
ウ：低カーディナリティ単独列は一般に絞込み力が低い。
エ：索引は更新時の維持コストを追加する。
総合解説：95%一致では表走査の方が効率的な場合もあり、実データ分布と条件全体で評価する必要がある。 索引は検索性能だけでなく、選択度、複合キー順序、更新負荷、容量、実行計画まで含めて設計する。

対象：2026年度 / 基準日 2026-09-02

0125 3-5 SQL性能設計 > インデックス設計 ｜ 難易度：標準

検索条件とSELECTで必要な列を索引だけで満たせるよう設計する「カバリング索引」の主な狙いはどれか。
ア：すべての更新処理を不要にする。
イ：バックアップを不要にする。
ウ：条件によっては表本体へのアクセスを減らし、I/Oを削減する。
エ：外部キー制約を自動削除する。

答え・解説 正答：ウ

ア：索引があっても更新処理そのものは必要である。
イ：バックアップの代替ではない。
ウ：必要列が索引に含まれ、可視性などの条件も満たせば表本体アクセスを削減できる。
エ：制約削除とは無関係である。
総合解説：必要列が索引に含まれ、可視性などの条件も満たせば表本体アクセスを削減できる。 索引は検索性能だけでなく、選択度、複合キー順序、更新負荷、容量、実行計画まで含めて設計する。

対象：2026年度 / 基準日 2026-09-02

0126 3-5 SQL性能設計 > インデックス設計 ｜ 難易度：標準

参照性能を上げるため一つの表に多数の索引を追加したところ、INSERT/UPDATEが遅くなった。最も妥当な理由はどれか。
ア：索引が増えるとDBMSはCOMMITを禁止するため。
イ：索引を作ると必ず全テーブルが暗号化されるため。
ウ：索引数が増えるとSQL構文解析ができなくなるため。
エ：データ変更時に関連する各索引も更新・分割等の維持処理が必要になるため。

答え・解説 正答：エ

ア：COMMIT禁止になるわけではない。
イ：暗号化とは無関係である。
ウ：構文解析不能になることはない。
エ：索引は検索を速める一方、更新時に索引エントリの維持I/OとCPUを追加する。
総合解説：索引は検索を速める一方、更新時に索引エントリの維持I/OとCPUを追加する。 索引は検索性能だけでなく、選択度、複合キー順序、更新負荷、容量、実行計画まで含めて設計する。

対象：2026年度 / 基準日 2026-09-02

0127 3-5 SQL性能設計 > インデックス設計 ｜ 難易度：標準

order_date に通常の索引がある。日付範囲検索の索引利用性を高めやすい書き方として最も適切なものはどれか。

ア：order_date >= :start AND order_date < :next_day
イ：FUNCTION(order_date) = :day のように索引列を任意関数で包むことを常に強制する。
ウ：CAST(order_date AS TEXT) LIKE :prefix を必ず使う。
エ：WHERE句を削除して全件取得する。

答え・解説 正答：ア

ア：索引列そのものに対する範囲条件はB-treeで利用しやすい形である。
イ：列への関数適用は通常の索引を利用しにくくする場合がある。
ウ：文字列化は型変換を伴い、通常の日時索引利用を妨げ得る。
エ：全件取得では絞込みがない。
総合解説：索引列そのものに対する範囲条件はB-treeで利用しやすい形である。 索引は検索性能だけでなく、選択度、複合キー順序、更新負荷、容量、実行計画まで含めて設計する。

対象：2026年度 / 基準日 2026-09-02

0128 3-5 SQL性能設計 > インデックス設計 ｜ 難易度：標準

索引キー順に近い形でデータ行が物理的にもまとまって格納されている場合、特に期待しやすい効果はどれか。

ア：すべての等価検索が必ずO(1)になる。
イ：連続したキー範囲を読むときのランダムI/Oを減らしやすい。
ウ：更新ログが不要になる。
エ：トランザクション分離レベルが自動的にSERIALIZABLEになる。

答え・解説 正答：イ

ア：B-tree検索が定数時間になるわけではない。
イ：範囲内の行が近接配置されれば、複数ページを順次読むアクセスの局所性が高まる。
ウ：回復ログとは別概念である。
エ：物理配置と分離レベルは無関係である。
総合解説：範囲内の行が近接配置されれば、複数ページを順次読むアクセスの局所性が高まる。 索引は検索性能だけでなく、選択度、複合キー順序、更新負荷、容量、実行計画まで含めて設計する。

対象：2026年度 / 基準日 2026-09-02

0129 3-5 SQL性能設計 > インデックス設計 ｜ 難易度：標準

注文表の99%が完了済みで、業務画面は未処理注文だけを頻繁に検索する。DBMSが部分索引を提供している場合、有効な設計として最も適切なものはどれか。

ア：完了済み99%だけを対象にし、未処理検索にも必ず使えると考える。
イ：WHERE条件の有無に関係なく全列に巨大な索引を作る。
ウ：未処理行だけを対象とする部分索引を検討し、索引サイズと更新負荷を抑える。
エ：索引を使うために未処理行を別DBMSへ毎回手作業でコピーする。

答え・解説 正答：ウ

ア：検索対象を含まない部分索引は直接役立たない。
イ：無目的な巨大索引は更新・容量コストを増やす。
ウ：頻繁に対象となる少数行だけを索引化できれば、検索性と索引サイズの両面で有利になり得る。
エ：通常はデータ複製より索引設計で解決を検討する。
総合解説：頻繁に対象となる少数行だけを索引化できれば、検索性と索引サイズの両面で有利になり得る。 索引は検索性能だけでなく、選択度、複合キー順序、更新負荷、容量、実行計画まで含めて設計する。

対象：2026年度 / 基準日 2026-09-02

0130 3-5 SQL性能設計 > インデックス設計 ｜ 難易度：標準

検索の大半が tenant_id = 定数 AND created_at >= 日時 であり、tenant_id単位に多数行が存在する。複合B-tree索引の候補として最も自然なのはどれか。

ア：(created_at, tenant_id) だけが常に唯一の正解
イ：(本文LOB, tenant_id)
ウ：全列を無条件に含めた索引
エ：(tenant_id, created_at)

答え・解説 正答：エ

ア：created_at全体の範囲からtenantを絞る方が不利なことがあり、唯一の正解とはいえない。
イ：大きなLOBを先頭キーにするのは通常この検索要件に合わない。
ウ：全列索引は容量・更新コストが大きくなる。
エ：等価条件tenant_idを先頭、その中でcreated_at範囲をたどる構成は典型的に有効である。
総合解説：等価条件tenant_idを先頭、その中でcreated_at範囲をたどる構成は典型的に有効である。索引は検索性能だけでなく、選択度、複合キー順序、更新負荷、容量、実行計画まで含めて設計する。

対象：2026年度 / 基準日 2026-09-02

0131 3-5 SQL性能設計 > インデックス設計 ｜ 難易度：応用

新しい索引候補が性能改善に有効かを本番相当データで検証する方法として最も適切なものはどれか。

ア：代表SQLの実行計画と実測時間・I/Oを比較し、更新負荷や索引サイズも含めて評価する。
イ：索引名の長さだけで性能を判断する。
ウ：索引を作成したら実行計画を確認せず必ず採用する。
エ：SELECT性能だけ測り、更新性能や容量は無視する。

答え・解説 正答：ア

ア：索引は読取り・書込み・容量のトレードオフなので、代表負荷を実測して総合判断する。
イ：名称は性能根拠にならない。
ウ：オプティマイザが使わない索引もあり得る。
エ：更新コストと容量も重要な設計要件である。
総合解説：索引は読取り・書込み・容量のトレードオフなので、代表負荷を実測して総合判断する。索引は検索性能だけでなく、選択度、複合キー順序、更新負荷、容量、実行計画まで含めて設計する。

対象：2026年度 / 基準日 2026-09-02

0132 3-5 SQL性能設計 > インデックス設計 ｜ 難易度：応用

同じ表に索引I1=(A)とI2=(A,B)がある。I1を削除できるか判断する際に最も適切な考え方はどれか。

ア：I2が存在すればI1はどのDBMSでも必ず完全に無意味なので即削除する。
イ：I2がA条件を代替できる可能性はあるが、I1の方が小さいことによる性能差、制約用途、DBMS仕様、実行計画を確認して判断する。
ウ：I1とI2は列数が違うので必ず両方必要である。
エ：索引の冗長性は更新性能に影響しないため確認不要である。

答え・解説 正答：イ

ア：常に完全代替とは限らない。
イ：先頭列が同じ複合索引は単一列索引を代替し得るが、サイズや制約・実装差を含め実測判断が必要である。
ウ：逆に常に両方必須でもない。
エ：冗長索引は更新I/O・容量を増やし得る。
総合解説：先頭列が同じ複合索引は単一列索引を代替し得るが、サイズや制約・実装差を含め実測判断が必要である。 索引は検索性能だけでなく、選択度、複合キー順序、更新負荷、容量、実行計画まで含めて設計する。

対象：2026年度 / 基準日 2026-09-02

0133 3-5 SQL性能設計 > 実行計画・アクセス経路 ｜ 難易度：基礎

SQLの実行計画から主に確認できる情報はどれか。

ア：ユーザーのパスワード平文。
イ：バックアップ媒体の耐用年数。
ウ：表の走査方式、結合順序・結合方式、推定コストや行数など。
エ：業務要件書の承認履歴。

答え・解説 正答：ウ

ア：認証秘密情報を表示する目的ではない。
イ：媒体管理とは別である。
ウ：実行計画はオプティマイザが選んだアクセス経路と演算順序を理解するための情報である。
エ：要件承認履歴はSQL計画に含まれない。
総合解説：実行計画はオプティマイザが選んだアクセス経路と演算順序を理解するための情報である。実行計画では推定行数と実測行数、走査方式、結合方式、I/Oなどを対応付けてボトルネックを判断する。

対象：2026年度 / 基準日 2026-09-02

0134 3-5 SQL性能設計 > 実行計画・アクセス経路 ｜ 難易度：基礎

大規模表に索引があってもオプティマイザが全表走査を選ぶことが合理的なケースはどれか。

ア：1行だけを主キーで検索する場合に必ず全表走査が最適である。
イ：表に1件も行がない場合しか全表走査は選ばれない。
ウ：索引が存在すると全表走査は技術的に不可能になる。
エ：条件に一致する行が表の大部分を占め、索引経由の大量ランダムアクセスより連続走査が安いと推定される場合。

答え・解説 正答：エ

ア：主キー1行検索では一般に索引が有利である。
イ：全表走査は多様な条件で選択される。
ウ：索引があってもオプティマイザは他経路を選べる。
エ：高い一致率では索引をたどって多数ページを読むより全走査が効率的なことがある。
総合解説：高い一致率では索引をたどって多数ページを読むより全走査が効率的なことがある。 実行計画では推定行数と実測行数、走査方式、結合方式、I/Oなどを対応付けてボトルネックを判断する。

対象：2026年度 / 基準日 2026-09-02

0135 3-5 SQL性能設計 > 実行計画・アクセス経路 ｜ 難易度：基礎

外側入力がごく少数行で、内側表の結合キーに高選択度の索引がある場合に適しやすい結合方式はどれか。

ア：Nested Loop Join
イ：外側と内側を必ず全件ソートする方式だけ
ウ：Cartesian Productを無条件に作る方式
エ：バックアップから結合結果を復元する方式

答え・解説 正答：ア

ア：少数の外側行ごとに内側索引を効率よく探索できるためNested Loopが有利になりやすい。
イ：ソートが常に必要とは限らない。
ウ：無条件直積は不要な組合せを大量生成する。
エ：バックアップは実行時結合方式ではない。
総合解説：少数の外側行ごとに内側索引を効率よく探索できるためNested Loopが有利になりやすい。実行計画では推定行数と実測行数、走査方式、結合方式、I/Oなどを対応付けてボトルネックを判断する。

対象：2026年度 / 基準日 2026-09-02

0136 3-5 SQL性能設計 > 実行計画・アクセス経路 ｜ 難易度：標準

大きな2表を等価条件で結合し、適切な索引がなく、一方をメモリ上のハッシュ表へ収められる。適しやすい結合方式はどれか。

ア：範囲条件専用のMerge Joinしか使えない
イ：Hash Join
ウ：各行について必ず全内側表を再走査する素朴なNested Loopだけ
エ：索引再構築

答え・解説 正答：イ

ア：Merge Joinも等価結合に使えるが「しか使えない」は誤りである。
イ：等価結合では一方からハッシュ表を作り、他方をプローブするHash Joinが有効なことが多い。
ウ：索引なし大表同士で素朴な二重走査は高コストになりやすい。
エ：索引再構築は結合アルゴリズムではない。
総合解説：等価結合では一方からハッシュ表を作り、他方をプローブするHash Joinが有効なことが多い。 実行計画では推定行数と実測行数、走査方式、結合方式、I/Oなどを対応付けてボトルネックを判断する。

対象：2026年度 / 基準日 2026-09-02

0137 3-5 SQL性能設計 > 実行計画・アクセス経路 ｜ 難易度：標準

Merge Joinの特徴として最も適切なものはどれか。
ア：等価条件では絶対に利用できない。
イ：入力順序やソートとは無関係である。
ウ：両入力が結合キー順に整列していれば、順次比較しながら結合できる。
エ：必ず内側表に主キー索引が必要である。

答え・解説 正答：ウ

ア：等価結合は代表的利用場面である。
イ：入力の順序性が重要である。
ウ：整列済み入力を先頭から進めて照合するため、大量データの結合で有効な場合がある。
エ：主キー索引は必須条件ではなく、ソート済み入力などでも成立する。
総合解説：整列済み入力を先頭から進めて照合するため、大量データの結合で有効な場合がある。 実行計画では推定行数と実測行数、走査方式、結合方式、I/Oなどを対応付けてボトルネックを判断する。

対象：2026年度 / 基準日 2026-09-02

0138 3-5 SQL性能設計 > 実行計画・アクセス経路 ｜ 難易度：標準

実行計画では中間結果を10行と推定してNested Loopを選んだが、実際は100万行だった。性能悪化の主要因として最も妥当なのはどれか。
ア：SQL文の文字コードがUTF-8だからである。
イ：COMMITが存在するからである。
ウ：バックアップ世代数が多すぎるからである。
エ：カーディナリティ推定誤差により、実データ量に不適切な結合方式・順序が選ばれた。

答え・解説 正答：エ

ア：文字コードはこの行数推定問題の直接原因ではない。
イ：COMMITの有無は結合方式選択の主因ではない。
ウ：バックアップ世代はクエリ計画とは無関係である。
エ：行数推定はアクセス経路・結合順序選択の基礎で、大きな誤差は計画ミスを招く。
総合解説：行数推定はアクセス経路・結合順序選択の基礎で、大きな誤差は計画ミスを招く。 実行計画では推定行数と実測行数、走査方式、結合方式、I/Oなどを対応付けてボトルネックを判断する。

対象：2026年度 / 基準日 2026-09-02

0139 3-5 SQL性能設計 > 実行計画・アクセス経路 ｜ 難易度：標準

結合前に各表へ適用可能な絞込み条件を早い段階で適用する利点として最も適切なものはどれか。
ア：後続演算へ渡す行数を減らし、結合・ソート等の処理量を削減できる。
イ：必ず結果行数を増やす。
ウ：トランザクションの永続性を無効にする。
エ：バックアップ容量だけを削減する。

答え・解説 正答：ア

ア：選択演算を早めると中間結果が小さくなり、後続処理コストを下げやすい。
イ：通常は条件で行を減らす。
ウ：ACIDの永続性とは別概念である。
エ：クエリ実行最適化でありバックアップ目的ではない。
総合解説：選択演算を早めると中間結果が小さくなり、後続処理コストを下げやすい。 実行計画では推定行数と実測行数、走査方式、結合方式、I/Oなどを対応付けてボトルネックを判断する。

対象：2026年度 / 基準日 2026-09-02

0140 3-5 SQL性能設計 > 実行計画・アクセス経路 ｜ 難易度：標準

売上表を売上月で月次パーティション分割している。WHERE sale_date >= 2026-08-01 AND sale_date < 2026-09-01 の検索で期待できる最適化はどれか。
ア：すべてのパーティションを必ず同じ量だけ走査する。
イ：8月以外の不要パーティションを走査対象から除外する。
ウ：過去パーティションを自動削除する。
エ：条件式を無視して全件を返す。

答え・解説 正答：イ

ア：プルーニングの目的は不要走査の削減である。
イ：パーティションキー条件から必要区画を特定できれば、対象外区画を枝刈りできる。
ウ：検索がデータ削除を意味するわけではない。
エ：WHERE条件は結果集合を制限する。
総合解説：パーティションキー条件から必要区画を特定できれば、対象外区画を枝刈りできる。 実行計画では推定行数と実測行数、走査方式、結合方式、I/Oなどを対応付けてボトルネックを判断する。

対象：2026年度 / 基準日 2026-09-02

0141 3-5 SQL性能設計 > 実行計画・アクセス経路 ｜ 難易度：応用

あるSQLのEXPLAIN推定コストは低いが実際には遅い。次に確認する情報として最も有効なものはどれか。

ア：推定コストだけが絶対値なので実測は不要である。
イ：SQLのコメントを増やして性能が変わるかだけを見る。
ウ：実測実行時間、実行行数、I/O・バッファ使用量を取得し、推定行数との差や待機要因を調べる。
エ：バックアップを削除してから再実行する。

答え・解説 正答：ウ

ア：コストはオプティマイザ内部の比較値で、実時間保証ではない。
イ：コメント追加は通常の性能診断手段ではない。
ウ：推定と実績を比較することで統計誤差、I/O、メモリ不足などの原因を絞り込める。
エ：バックアップ削除は危険で無関係である。
総合解説：推定と実績を比較することで統計誤差、I/O、メモリ不足などの原因を絞り込める。 実行計画では推定行数と実測行数、走査方式、結合方式、I/Oなどを対応付けてボトルネックを判断する。

対象：2026年度 / 基準日 2026-09-02

0142 3-5 SQL性能設計 > 実行計画・アクセス経路 ｜ 難易度：応用

大規模ORDER BYの実行計画で、ソートがメモリに収まらず一時ディスクへ大量に書き出していることが分かった。最も妥当な評価はどれか。

ア：一時ディスク利用は必ずメモリ内ソートより高速なので対策不要である。
イ：ORDER BYを残したまま結果順序だけ無視すればよい。
ウ：WALを削除すればソートがメモリに収まる。
エ：ディスクI/Oがボトルネックになり得るので、対象行数削減・索引利用・ソート用メモリ設定などを総合検討する。

答え・解説 正答：エ

ア：通常、メモリ内処理よりI/O負荷が大きい。
イ：要求された順序を無視しては機能要件を満たさない。
ウ：WAL削除は無関係かつ危険である。
エ：外部ソートはディスクI/Oを伴うため、行数・索引・メモリの観点で改善余地を検討する。
総合解説：外部ソートはディスクI/Oを伴うため、行数・索引・メモリの観点で改善余地を検討する。 実行計画では推定行数と実測行数、走査方式、結合方式、I/Oなどを対応付けてボトルネックを判断する。

対象：2026年度 / 基準日 2026-09-02

0143 3-5 SQL性能設計 > SQLチューニング・統計情報 ｜ 難易度：基礎

オプティマイザが列値の分布や行数などの統計情報を利用する主な目的はどれか。

ア：条件の選択度や中間結果の行数を推定し、低コストな実行計画を選ぶため。
イ：バックアップファイルを暗号化するため。
ウ：トランザクションのCOMMITを取り消すため。
エ：テーブル名を自動翻訳するため。

答え・解説 正答：ア

ア：統計情報はカーディナリティ・選択度推定の基礎となる。
イ：暗号化とは別である。
ウ：COMMIT取消しは回復・トランザクション機能である。
エ：名称翻訳の機能ではない。
総合解説：統計情報はカーディナリティ・選択度推定の基礎となる。 SQLチューニングは統計情報と実行計画の推定精度、実測資源使用量、結果の正しさを合わせて評価する。

対象：2026年度 / 基準日 2026-09-02

0144 3-5 SQL性能設計 > SQLチューニング・統計情報 ｜ 難易度：基礎

大量データを一括ロードした直後、以前の統計情報のまま実行計画が不適切になった。まず行う対策として最も適切なものはどれか。

ア：主キーを削除する。
イ：統計情報を再収集し、データ量・分布の変化をオプティマイザへ反映する。
ウ：ログをすべて削除する。
エ：バックアップを停止する。

答え・解説 正答：イ

ア：整合性制約を削除する理由にはならない。
イ：大量ロード後は行数や値分布が大きく変わるため、統計更新が計画改善に直結しやすい。
ウ：ログ削除は回復可能性を損なう。
エ：バックアップ停止は性能計画の根本対策ではない。
総合解説：大量ロード後は行数や値分布が大きく変わるため、統計更新が計画改善に直結しやすい。 SQLチューニングは統計情報と実行計画の推定精度、実測資源使用量、結果の正しさを合わせて評価する。

対象：2026年度 / 基準日 2026-09-02

0145 3-5 SQL性能設計 > SQLチューニング・統計情報 ｜ 難易度：標準

indexed_col にB-tree索引がある。WHERE indexed_col + 1 = 100 を、通常の索引を利用しやすい形へ書き換える例として最も適切なものはどれか。

ア：WHERE indexed_col + 1 + 0 = 100
イ：WHERE CAST(indexed_col AS TEXT) = "99"
ウ：WHERE indexed_col = 99
エ：WHERE 1 = 1

答え・解説 正答：ウ

ア：列側の式が残っている。
イ：不要な型変換で通常索引を使いにくくし得る。
ウ：列側の演算を除き、同値な定数比較にすれば一般に索引条件へ変換しやすい。
エ：全行真になり絞込みを失う。
総合解説：列側の演算を除き、同値な定数比較にすれば一般に索引条件へ変換しやすい。 SQLチューニングは統計情報と実行計画の推定精度、実測資源使用量、結果の正しさを合わせて評価する。

対象：2026年度 / 基準日 2026-09-02

0146 3-5 SQL性能設計 > SQLチューニング・統計情報 ｜ 難易度：標準

大量行を返すAPIでSELECT *を使っているが、実際に必要なのは3列だけである。必要列だけを指定する主な性能上の利点として最も適切なものはどれか。

ア：WHERE句がなくても必ず1行だけ返るようになる。
イ：トランザクション分離レベルが自動的に上がる。
ウ：バックアップ世代数が減る。
エ：転送・バッファ・デコード量を減らせ、場合によってはカバリング索引も利用しやすくなる。

答え・解説 正答：エ

ア：取得列限定は行数を自動制限しない。
イ：分離レベルとは別である。
ウ：バックアップ管理とは無関係である。
エ：不要列を取得しないことでI/Oやネットワーク量を抑え、索引だけで完結する可能性も高められる。
総合解説：不要列を取得しないことでI/Oやネットワーク量を抑え、索引だけで完結する可能性も高められる。 SQLチューニングは統計情報と実行計画の推定精度、実測資源使用量、結果の正しさを合わせて評価する。

対象：2026年度 / 基準日 2026-09-02

0147 3-5 SQL性能設計 > SQLチューニング・統計情報 ｜ 難易度：標準

都道府県列と市区町村列のように強い相関がある2列を同時条件にすると、オプティマイザが独立と仮定して行数を大きく誤推定する。DBMSが多列統計・拡張統計を提供する場合の狙いとして最も適切なものはどれか。
ア：列間の依存関係を統計へ反映し、複合条件のカーディナリティ推定を改善する。
イ：SQLの構文エラーを自動修正する。
ウ：バックアップを圧縮する。
エ：外部キー違反を無視する。

答え・解説 正答：ア

ア：単一列統計だけでは列間相関を表せないため、多列統計で推定精度を上げることがある。
イ：構文修正機能ではない。
ウ：バックアップ圧縮とは無関係である。
エ：整合性制約を無効化するものではない。
総合解説：単一列統計だけでは列間相関を表せないため、多列統計で推定精度を上げることがある。SQLチューニングは統計情報と実行計画の推定精度、実測資源使用量、結果の正しさを合わせて評価する。

対象：2026年度 / 基準日 2026-09-02

0148 3-5 SQL性能設計 > SQLチューニング・統計情報 ｜ 難易度：標準

SQLが遅いという報告を受けた。チューニングの進め方として最も適切なものはどれか。
ア：根拠なく全SQLへ同じヒント句を追加する。
イ：実行時間、待機、I/O、実行計画、行数推定などを測定し、支配的なボトルネックを特定してから対策する。
ウ：すべての表へ可能な限り多くの索引を作る。
エ：統計情報を削除すれば常に最速になると仮定する。

答え・解説 正答：イ

ア：画一的ヒントは別環境で悪化させることもある。
イ：計測に基づき原因を特定し、SQL・索引・統計・設定のうち適切な層へ対策するのが基本である。
ウ：索引過多は更新性能・容量を悪化させる。
エ：統計削除は計画品質を下げ得る。
総合解説：計測に基づき原因を特定し、SQL・索引・統計・設定のうち適切な層へ対策するのが基本である。 SQLチューニングは統計情報と実行計画の推定精度、実測資源使用量、結果の正しさを合わせて評価する。

対象：2026年度 / 基準日 2026-09-02

0149 3-5 SQL性能設計 > SQLチューニング・統計情報 ｜ 難易度：応用

100万行の category 列で値Aが90%、その他999種類が残り10%に分布している。値分布を考慮しない一様分布の推定が問題になる理由として最も適切なものはどれか。

ア：値の偏りが大きいほど全検索の実行時間は必ず同じになるため。
イ：ヒストグラムがあるとSQLの結果値が変更されるため。
ウ：category="A" と希少値の検索で実際の一致行数が大きく異なるのに、同程度と見積もるとアクセス経路選択を誤り得るため。
エ：統計はバックアップ用途だけに使われるため。

答え・解説 正答：ウ

ア：偏りがあるからこそ条件ごとの最適経路が変わり得る。
イ：統計は結果の意味を変えず、計画選択に使われる。
ウ：偏りを表す統計がないと選択度推定を誤り、索引利用や結合方式の選択に影響する。
エ：統計の主用途はオプティマイザの推定である。
総合解説：偏りを表す統計がないと選択度推定を誤り、索引利用や結合方式の選択に影響する。 SQLチューニングは統計情報と実行計画の推定精度、実測資源使用量、結果の正しさを合わせて評価する。

対象：2026年度 / 基準日 2026-09-02

0150 3-5 SQL性能設計 > SQLチューニング・統計情報 ｜ 難易度：応用

SQLを書き換えて本番相当環境で平均時間が半分になったが、95パーセンタイル応答時間とCPU使用率が悪化した。採用判断として最も適切なものはどれか。

ア：平均時間が短ければ他指標は無条件で無視する。
イ：CPU使用率が上がればSQL結果が必ず誤りになる。
ウ：性能検証では同じ結果を返すか確認する必要はない。
エ：平均値だけで決めず、分位点・CPU・I/O・同時実行性能・結果同値性を含め、サービス目標に照らして再評価する。

答え・解説 正答：エ

ア：尾部遅延や資源競合が利用者影響を支配する場合がある。
イ：CPU増加だけで論理結果誤りとは限らない。
ウ：書換えでは結果同値性の検証が必須である。
エ：チューニングは単一平均値ではなく、負荷全体とSLO、正しさを含めて評価する。
総合解説：チューニングは単一平均値ではなく、負荷全体とSLO、正しさを含めて評価する。 SQLチューニングは統計情報と実行計画の推定精度、実測資源使用量、結果の正しさを合わせて評価する。

対象：2026年度 / 基準日 2026-09-02