

基礎から固める SQL重要構文 完全攻略ノート

午前理論（DDL/DML/DCL・3値論理）を午後記述の手書き穴埋め力へと昇華させる必修リファレンス

【午前と午後の架け橋】 午前試験の「選択式（マーク）」で問われる文法知識は、午後試験では「手書き穴埋め（記述式）」で問われます。句の順序、カンマや括弧の有無、NULLの3値論理など、曖昧な記憶を排除して正確な記述力を身につけましょう。

言語分類	代表的な命令構文	午後問記述での役割と特徴
DDL（定義言語）	CREATE TABLE, ALTER TABLE, DROP TABLE, CREATE VIEW	スキーマ定義・表構造変更。主キー・外部キー制約の手書き記述が頻出。
DML（操作言語）	SELECT, INSERT, UPDATE, DELETE, カーソル（FETCH）	問合せ・行データ操作。午後問の主軸（結合・集約・副問合せ）。
DCL（制御言語）	GRANT（権限付与）, REVOKE（権限剥奪）	ユーザー・ロールへのセキュリティ設定。TO/FROMの方向性に注意。
TCL（トランザクション）	COMMIT（確定）, ROLLBACK（取消）	ACID特性の保持。ストアドや業務フロー記述の設問で出題。

SECTION 1: 表の定義（DDL）と整合性制約

社員表・部門表をベースとしたテーブル作成と参照整合性の定義

1. 表の定義と外部キー参照：CREATE TABLE / PRIMARY KEY / FOREIGN KEY

DDL基礎

午前の基礎 \$1to\$ 午後R4秋, R7秋

SQL 構文例（社員表の作成）

```
CREATE TABLE 社員表 (  
  社員ID   CHAR(4)      NOT NULL PRIMARY KEY,  -- 主キー: 一意性+NOT NULL  
  氏名     VARCHAR(20),  
  部門ID   CHAR(2),  
  給与     INT,  
  FOREIGN KEY (部門ID) REFERENCES 部門表(部門ID)  -- 外部キー制約  
  ON DELETE SET NULL                               -- 親削除時の連動動作  
);
```

【このSQL文は何をしているのか】

- 実表の作成：固定長文字列（CHAR）、可変長文字列（VARCHAR）、整数（INT）などを指定して社員表を定義。
- 主キー（PRIMARY KEY）：社員を一意に識別するキー。複合主キーの場合は文末に PRIMARY KEY（列1, 列2）と記述。
- 外部キー（FOREIGN KEY ... REFERENCES）：親表「部門表」に存在する部門IDしか入力できないよう参照整合性を保証。

午後記述の急所：複合主キーの括弧漏れや、外部キー定型句 FOREIGN KEY（列）REFERENCES 相手表（列）のスペルミス（特に FOREIGN の G の位置）に注意。

2. 射影（列抽出・計算式・別名）と選択（行絞り込み）：SELECT ... WHERE

DML基礎

午前の関係代数 \$1to\$ 午後必須

SQL 構文例（射影と選択の組み合わせ）

```
SELECT 氏名, 給与, 給与 * 1.05 AS 改訂給与  -- 射影: 計算式と別名付与  
FROM   社員表  
WHERE  部門ID IN ('01', '02')              -- 選択: 01 または 02 部門  
AND    性別 = 'F';                        -- 論理積: かつ女性
```

【このSQL文は何をしているのか】

- 選択（Selection）：全行の中から「部門IDが01または02」かつ「女性（性別=F）」である行を横方向に絞り込みます。
- 射影（Projection）：該当行から氏名と現給与を縦方向に抜き出し、給与を5%昇給させた計算値に AS 改訂給与 と命名します。

SECTION 2: 3値論理 (NULL) ・集約関数・グループ化と整列

午後問で最も減点されやすい「WHEREとHAVINGの境界」と「3値論理の判定」

3. NULLの判定と範囲・部分一致：IS NULL / BETWEEN / LIKE / DISTINCT

必須条件

午後出題率90%超

SQL 構文例

```
SELECT DISTINCT 都道府県
FROM 社員表
WHERE 生年月日 BETWEEN '1970-01-01' AND '1979-12-31' -- 境界値を含む (以上・以下)
      AND 退職日 IS NULL -- = NULL は誤り(3値論理)
      AND 氏名 LIKE '\%子'; -- 後方一致 (子で終わる)
```

【このSQL文は何をしているのか】

- 3値論理の罠：SQLのNULLは未定義値であり、= NULL は常にUNKNOWN (偽) と評価されます。必ず IS NULL (否定は IS NOT NULL) と手書きします。
- BETWEEN：`A以上B以下` (境界値を含む)。1970年生まれなら 1970-01-01 から 1979-12-31 を指定。
- DISTINCT：結果表から重複行を取り除きます (集計内で COUNT(DISTINCT 都道府県) と使う形も超頻出)。

午後記述の急所：重複件数を数える際、DISTINCT COUNT(列) は構文エラーです。必ず関数の内側に COUNT(DISTINCT 列) と書きます。

4. 集約とグループ後抽出：GROUP BY / HAVING / COUNT(*) vs COUNT(列)

合否の分岐点

午後問鉄板

SQL 構文例 (部門ごとの給与集計)

```
SELECT 部門ID, COUNT(*) AS 人数, AVG(給与) AS 平均給与
FROM 社員表
WHERE 性別 = 'F' -- ① 集計前の行絞り込み(女性のみ)
GROUP BY 部門ID -- ② 部門ごとにまとめる
HAVING COUNT(*) >= 3 -- ③ 集約後のグループ絞り込み
      AND AVG(給与) >= 300000; -- 集計値に対する条件はHAVING
```

【このSQL文は何をしているのか】

- COUNTの違い：COUNT(*) はNULL行も含めて全行数を数えます。COUNT(給与) や AVG(給与) はNULLを除外して計算します。
- WHEREとHAVINGの境界線：WHEREはグループ化「前」の行フィルタ。HAVINGはGROUP BYで集計された「後」のグループに対するフィルタ。WHERE句に集計関数を書くことは文法違反です。

5. 並び替え：ORDER BY (DESC / ASC / 列位置番号)

整列

ソート順の罠

SQL 構文例 (男性優先・若年順ソート)

```
SELECT *
FROM 社員表
ORDER BY 性別 DESC, -- 第1キー: 'M'(男) → 'F'(女) の順 (降順)
        生年月日 DESC; -- 第2キー: 日付が大きい(年齢が若い)順 (降順)
```

【このSQL文は何をしているのか】

- 男性が先：性別コード (M, F) を昇順 (ASC) にするとFが先頭になります。男性 (M) を先にするには DESC (降順) を指定します。
- 年齢の若い順：日付値が大きい (最新日付) ほど若いため、生年月日も DESC (降順) にします (ASCにすると高齢順になる)。

SECTION 3: 結合（内部・外部）・集合演算・副問合せ

複数テーブルの紐づけと「～の実績がないデータ」を浮き彫りにする定型構文

6. テーブル結合：INNER JOIN / LEFT OUTER JOIN (ON / USING)

最重要得点源

午後問主軸

SQL 構文例（外部結合と内部結合）

```
-- 1. LEFT OUTER JOIN: 所属社員が0人の部門も漏らさず残す
SELECT      D.部門ID, D.部名, E.氏名
FROM        部門表 D
LEFT OUTER JOIN 社員表 E ON D.部門ID = E.部門ID; -- 不一致行はNULL補完

-- 2. INNER JOIN + USING: 同名結合キーを簡略指定
SELECT 氏名, 部名 FROM 社員表 INNER JOIN 部門表 USING (部門ID);
```

【このSQL文は何をしているのか】

- 外部結合 (LEFT JOIN) : 左側「部門表」の全行を保持。新設部門で所属社員が0名であっても行は残り、社員氏名列にNULLが入ります。
- 内部結合 (INNER JOIN) : 両テーブルに共通して存在する行のみを残します（社員のいない部門は除外）。

午後記述の急所: 「一度も注文がない顧客」や「社員がいない部門」を抽出する設問は、LEFT JOIN ... WHERE 右表キー IS NULL で解くのが午後問の黄金パターンです。

7. 集合演算：UNION vs UNION ALL（処理性能と重複排除）

集合結合

R6秋, H31春

SQL 構文例

```
SELECT 社員ID, 氏名, 部門ID, 給与 * 1.05 AS 新給与 FROM 社員表 WHERE 部門ID = '00'
UNION ALL -- 重複排除のソートを行わず、そのまま縦に高速統合
SELECT 社員ID, 氏名, 部門ID, 給与 * 1.08 AS 新給与 FROM 社員表 WHERE 部門ID <> '00';
```

【このSQL文は何をしているのか】

- UNION : 2つの問合せ結果を縦に統合し、重複行を排除（内部でソートが発生するため負荷大）。
- UNION ALL : 重複行を排除せず全行をそのまま高速に統合（午後問で「高速に」「重複を排除せず」とあればこれ）。

8. 副問合せと存在検査：スカラー副問合せ / EXISTS / NOT EXISTS

合否の分かれ目

R2秋, H27春

SQL 構文例（受注実績のある社員の抽出）

```
-- 相関副問合せ: 主問合せの社員1行ごとに受注有無を判定
SELECT 社員ID, 氏名 FROM 社員表 X
WHERE EXISTS (
    SELECT 1 FROM 受注表
    WHERE 担当ID = X.社員ID -- 相関条件
);
```

【このSQL文は何をしているのか】

- EXISTSの動作 : 副問合せで条件に一致する行が1件でもあれば直ちに真 (TRUE) を返します。射影列は慣例として 1 や * を記述します。
- 最高値の抽出 (スカラー) : WHERE 給与 = (SELECT MAX(給与) FROM 社員表) のように単一の値を返す副問合せと比較します。

SECTION 4: 高度分析構文 (ウィンドウ関数・再帰CTE) & 制御言語 (DCL)

近年のDX・データ分析トレンド (R5秋・R6秋) を反映した最新難関構文と権限

9. ウィンドウ関数 (OLAP分析) : OVER (PARTITION BY ... ORDER BY)

高難度トレンド

R5秋, H30春

SQL 構文例 (各社員の明細を維持したまま部門平均・順位を算出)

```
SELECT 氏名, 部門ID, 給与,
       AVG(給与) OVER (PARTITION BY 部門ID) AS 部門平均給与,
       RANK() OVER (
         PARTITION BY 部門ID
         ORDER BY 給与 DESC
       ) AS 部門内順位
FROM 社員表;
```

【このSQL文は何をしているのか】

- 行を集約しない: GROUP BYと異なり、社員一人ひとりの行を潰さずに残したまま、部署ごとの集計値やランキングを並列して出力します。
- 構文要素: PARTITION BY でグループの仕切り枠を作り、ORDER BY でその枠内の整列順を定めます。

10. 再帰共通テーブル式 : WITH RECURSIVE (ツリー探索・部品表BOM)

最難関

R6秋

SQL 構文例 (組織階層ツリーの再帰的走査)

```
WITH RECURSIVE 組織ツリー (組織コード, 親組織コード, レベル) AS (
  /* 1. アンカーメンバ (最上位の組織をレベル1として特定) */
  SELECT 組織コード, 親組織コード, 1 FROM 組織 WHERE 親組織コード IS NULL
  UNION ALL
  /* 2. 再帰メンバ (子組織を順次結合して階層レベルを加算) */
  SELECT C.組織コード, C.親組織コード, P.レベル + 1
  FROM 組織 C JOIN 組織ツリー P ON C.親組織コード = P.組織コード
)
SELECT * FROM 組織ツリー;
```

【このSQL文は何をしているのか】

- ツリー探索: 最上位 (親がNULL) から末端の子組織までを芋づる式にループ結合して辿り切る強力な構文です。

11. データベース制御言語 (DCL) : GRANT / REVOKE

セキュリティ

H30秋

SQL 構文例

```
GRANT SELECT, INSERT ON 社員表 TO 営業部グループ; -- 権限付与 (TO)
REVOKE ALL ON 社員表 FROM 退職者グループ; -- 権限剥奪 (FROM)
```

【このSQL文は何をしているのか】

- 前置詞の方向性: 付与は GRANT ... TO ..., 剥奪は REVOKE ... FROM ...。穴埋めでの逆転に注意。

出題テーマ	午後記述での正解キーワード	試験本番での減点防止ポイント
未存在データの抽出	LEFT OUTER JOIN + IS NULL / NOT EXISTS	「～がないデータも含め」ならLEFT JOIN。「= NULL」は絶対不可。
集計値でのグループ絞込	HAVING COUNT(*) >= 3	WHERE句にAVGやCOUNTを書かない。HAVINGとWHEREを区別する。
参照整合性と連動削除	FOREIGN KEY ... REFERENCES / CASCADE	FOREIGN (Gの位置)、CASCADE (Eで終わる) のスペルミスを防止。
行を潰さない分析	OVER (PARTITION BY ... ORDER BY ...)	GROUP BYではなくPARTITION BY。OVERキーワードの手書き忘れに注意。