===========================================================================

# 39. ユーザーとロール：アクセス権限管理

## はじめに

前章では、データベースの設計と構造管理について学習しました。この章からは「セキュリティと権限管理」について学習していきます。どんなに優れたデータベースを設計しても、適切なセキュリティ対策がなければ、データの漏洩や不正な操作が起こる可能性があります。

ユーザー管理とアクセス権限の制御は、データベースセキュリティの最も基本的で重要な要素です。特に学校のような教育機関では、学生の個人情報や成績情報などの機密データを扱うため、誰がどのデータにアクセスできるかを厳密に管理する必要があります。

ユーザーとロール管理が必要となる場面の例：

- 「学生は自分の成績のみ閲覧でき、他の学生の情報は見られないようにしたい」
- 「教師は担当クラスの成績入力はできるが、他のクラスは変更できないようにしたい」
- 「事務職員は学生の基本情報は管理できるが、成績は見られないようにしたい」
- 「システム管理者のみがユーザーの追加・削除を行えるようにしたい」
- 「開発者用のテスト環境では全権限を持つが、本番環境では制限をかけたい」
- 「外部業者にはバックアップ作業のみ許可し、データの閲覧は禁止したい」
- 「退職した職員のアカウントを無効化して、データアクセスを完全に遮断したい」

この章では、MySQLでのユーザー管理とロールベースのアクセス制御について、基本概念から実践的な運用まで詳しく学んでいきます。

## ユーザーとロールとは

### ユーザー（User）

**ユーザー**とは、データベースにアクセスする人やシステムのことです。各ユーザーには固有の名前とパスワードが設定され、それぞれに異なる権限を与えることができます。

### ロール（Role）

**ロール**とは、複数の権限をまとめたグループのことです。例えば「教師」というロールには「成績の閲覧・入力権限」をまとめて設定し、教師ユーザーにそのロールを割り当てることで、権限管理を効率化できます。

### アクセス制御の基本概念

> **用語解説**：
>
> - **ユーザー（User）**：データベースにログインして操作を行う主体のことです。
> - **ロール（Role）**：権限をまとめたグループで、複数のユーザーに同じ権限セットを効率的に付与できます。

> - **権限（Privilege）**：データベースで実行できる操作の種類です（SELECT、INSERT、UPDATE等）。
> - **認証（Authentication）**：ユーザーが本人であることを確認するプロセスです（ユーザー名とパスワードの確認等）。
> - **認可（Authorization）**：認証されたユーザーが特定の操作を実行する権限があるかを確認するプロセスです。
> - **最小権限の原則**：ユーザーには業務に必要な最小限の権限のみを与えるセキュリティの基本原則です。
> - **ホスト（Host）**：ユーザーがデータベースに接続する元のコンピューターやサーバーのことです。
> - **アカウント**：ユーザー名とホストの組み合わせで構成される、MySQLでの識別単位です。
> - **スキーマ**：データベース内のテーブルやビューなどのオブジェクトをまとめる名前空間です。

## MySQLのユーザー管理システム

### ユーザーアカウントの構成

MySQLでは、ユーザーアカウントは「ユーザー名@ホスト名」の形式で管理されます。

```
例：
- 'student_user'@'localhost'      # ローカルからのみアクセス可能
- 'teacher_user'@'192.168.1.%'    # 特定のネットワークからのみアクセス可能
- 'admin_user'@'%'                # どこからでもアクセス可能（非推奨）
```

### 権限の階層構造

MySQLの権限は以下のような階層構造になっています：

| 階層レベル | 説明 | 適用範囲 |
|---|---|---|
| **グローバル** | サーバー全体 | 全データベース・全テーブル |
| **データベース** | 特定のデータベース | 指定されたデータベース内の全テーブル |
| **テーブル** | 特定のテーブル | 指定されたテーブルのみ |
| **カラム** | 特定のカラム | 指定されたカラムのみ |
| **ルーチン** | ストアドプロシージャ・関数 | 指定されたプロシージャ・関数のみ |

## 基本的なユーザー管理操作

### 1. 現在のユーザー情報確認

```sql
-- 現在ログインしているユーザーを確認
SELECT USER(), CURRENT_USER();

-- システム内の全ユーザーを確認
SELECT User, Host FROM mysql.user;
```

```
-- 現在のユーザーの権限を確認
SHOW GRANTS;

-- 特定ユーザーの権限を確認
SHOW GRANTS FOR 'username'@'hostname';
```

## 2. ユーザーの作成

**基本的なユーザー作成**

```
-- 基本的なユーザー作成
CREATE USER 'new_user'@'localhost' IDENTIFIED BY 'secure_password';

-- 複数のユーザーを同時作成
CREATE USER
    'student1'@'localhost' IDENTIFIED BY 'student_pass1',
    'student2'@'localhost' IDENTIFIED BY 'student_pass2',
    'teacher1'@'localhost' IDENTIFIED BY 'teacher_pass1';

-- 作成されたユーザーの確認
SELECT User, Host, account_locked, password_expired
FROM mysql.user
WHERE User IN ('student1', 'student2', 'teacher1');
```

**パスワード要件の設定**

```
-- パスワードの有効期限を設定したユーザー作成
CREATE USER 'temp_user'@'localhost'
IDENTIFIED BY 'temporary_pass'
PASSWORD EXPIRE INTERVAL 90 DAY;

-- 初回ログイン時にパスワード変更を強制
CREATE USER 'new_employee'@'localhost'
IDENTIFIED BY 'initial_pass'
PASSWORD EXPIRE;

-- アカウントをロックした状態で作成
CREATE USER 'inactive_user'@'localhost'
IDENTIFIED BY 'locked_pass'
ACCOUNT LOCK;
```

## 3. ユーザーの変更

```
-- パスワードの変更
ALTER USER 'student1'@'localhost' IDENTIFIED BY 'new_student_pass';
```

```
-- 複数ユーザーのパスワードを同時変更
ALTER USER
    'student1'@'localhost' IDENTIFIED BY 'new_pass1',
    'student2'@'localhost' IDENTIFIED BY 'new_pass2';

-- アカウントのロック・アンロック
ALTER USER 'student1'@'localhost' ACCOUNT LOCK;
ALTER USER 'student1'@'localhost' ACCOUNT UNLOCK;

-- パスワード有効期限の設定
ALTER USER 'teacher1'@'localhost' PASSWORD EXPIRE INTERVAL 180 DAY;
ALTER USER 'teacher1'@'localhost' PASSWORD EXPIRE NEVER;
```

### 4. ユーザーの削除

```
-- 単一ユーザーの削除
DROP USER 'temp_user'@'localhost';

-- 複数ユーザーの同時削除
DROP USER
    'old_student'@'localhost',
    'graduated_student'@'localhost';

-- 存在しない場合でもエラーにしない安全な削除
DROP USER IF EXISTS 'might_not_exist'@'localhost';
```

## 学校システムでの実践的ユーザー管理

### 1. 学校システム用ユーザーの作成

```
-- 管理者ユーザー（システム全体の管理）
CREATE USER 'school_admin'@'localhost'
IDENTIFIED BY 'AdminSecure2025!'
PASSWORD EXPIRE INTERVAL 60 DAY;

-- 教師ユーザー（成績管理・出席管理）
CREATE USER 'teacher_tanaka'@'localhost'
IDENTIFIED BY 'TeacherPass2025!'
PASSWORD EXPIRE INTERVAL 90 DAY;

CREATE USER 'teacher_sato'@'localhost'
IDENTIFIED BY 'TeacherPass2025!'
PASSWORD EXPIRE INTERVAL 90 DAY;

-- 学生ユーザー（自分の情報閲覧のみ）
CREATE USER 'student_301'@'localhost'
IDENTIFIED BY 'StudentPass301!'
PASSWORD EXPIRE INTERVAL 180 DAY;
```

```sql
CREATE USER 'student_302'@'localhost'
IDENTIFIED BY 'StudentPass302!'
PASSWORD EXPIRE INTERVAL 180 DAY;

-- 事務職員ユーザー（学生情報管理）
CREATE USER 'office_staff'@'localhost'
IDENTIFIED BY 'OfficePass2025!'
PASSWORD EXPIRE INTERVAL 90 DAY;

-- 読み取り専用ユーザー（レポート作成用）
CREATE USER 'report_viewer'@'localhost'
IDENTIFIED BY 'ReportView2025!'
PASSWORD EXPIRE INTERVAL 365 DAY;

-- 作成されたユーザーの確認
SELECT
    User as username,
    Host as allowed_host,
    password_expired,
    account_locked,
    password_lifetime as password_expires_days
FROM mysql.user
WHERE User LIKE 'school_%' OR User LIKE 'teacher_%' OR User LIKE 'student_%' OR
User LIKE 'office_%' OR User LIKE 'report_%'
ORDER BY User;
```

## 2. 接続元制限の設定

```sql
-- 校内ネットワークからのみアクセス可能な教師ユーザー
CREATE USER 'teacher_remote'@'192.168.1.%'
IDENTIFIED BY 'RemoteTeacher2025!';

-- 特定のIPアドレスからのみアクセス可能な管理者
CREATE USER 'secure_admin'@'192.168.1.100'
IDENTIFIED BY 'SecureAdmin2025!';

-- 複数の接続元を許可する場合は、複数のアカウントを作成
CREATE USER 'flexible_teacher'@'localhost' IDENTIFIED BY 'FlexTeacher2025!';
CREATE USER 'flexible_teacher'@'192.168.1.%' IDENTIFIED BY 'FlexTeacher2025!';
CREATE USER 'flexible_teacher'@'10.0.0.%' IDENTIFIED BY 'FlexTeacher2025!';
```

## 3. ユーザー情報の管理

```sql
-- 現在アクティブなユーザーセッションの確認
SELECT
    User,
    Host,
    db as current_database,
    Command,
    Time as session_time_seconds,
    State,
    Info as current_query
FROM information_schema.processlist
WHERE User != 'system user'
ORDER BY Time DESC;

-- ユーザーの最終ログイン時間確認（MySQL 8.0以降）
SELECT
    USER,
    HOST,
    CURRENT_CONNECTIONS,
    TOTAL_CONNECTIONS
FROM performance_schema.accounts
WHERE USER NOT IN ('mysql.sys', 'mysql.session', 'mysql.infoschema')
ORDER BY TOTAL_CONNECTIONS DESC;

-- パスワード期限が近いユーザーの確認
SELECT
    User,
    Host,
    password_expired,
    password_lifetime,
    password_last_changed
FROM mysql.user
WHERE password_lifetime IS NOT NULL
AND password_last_changed IS NOT NULL
AND DATE_ADD(password_last_changed, INTERVAL password_lifetime DAY) <=
DATE_ADD(NOW(), INTERVAL 7 DAY)
ORDER BY password_last_changed;
```

## ロールベースアクセス制御（MySQL 8.0以降）

### 1. ロールの基本概念

MySQL 8.0以降では、ロール機能が本格的にサポートされています。

```
-- ロール機能が有効かどうか確認
SELECT @@activate_all_roles_on_login;

-- ロール機能を有効化（必要に応じて）
SET GLOBAL activate_all_roles_on_login = ON;
```

### 2. 学校システム用ロールの作成

```
-- 基本的なロールの作成
CREATE ROLE 'school_admin_role';
CREATE ROLE 'teacher_role';
CREATE ROLE 'student_role';
CREATE ROLE 'office_staff_role';
CREATE ROLE 'report_reader_role';

-- 特定業務用のロール
CREATE ROLE 'grade_manager';      -- 成績管理専用
CREATE ROLE 'attendance_manager'; -- 出席管理専用
CREATE ROLE 'course_coordinator'; -- 講座管理専用

-- 作成されたロールの確認
SELECT User as role_name, Host
FROM mysql.user
WHERE account_locked = 'Y' AND password_expired = 'Y'
ORDER BY User;
```

### 3. ロールへの権限付与

```
-- 学生ロール：自分の情報のみ閲覧可能
GRANT SELECT ON school_db.students TO 'student_role';
GRANT SELECT ON school_db.grades TO 'student_role';
GRANT SELECT ON school_db.attendance TO 'student_role';

-- 教師ロール：担当クラスの管理が可能
GRANT SELECT, INSERT, UPDATE ON school_db.grades TO 'teacher_role';
GRANT SELECT, INSERT, UPDATE ON school_db.attendance TO 'teacher_role';
GRANT SELECT ON school_db.students TO 'teacher_role';
GRANT SELECT ON school_db.courses TO 'teacher_role';

-- 事務職員ロール：学生情報の管理が可能
GRANT SELECT, INSERT, UPDATE ON school_db.students TO 'office_staff_role';
GRANT SELECT ON school_db.courses TO 'office_staff_role';
GRANT SELECT ON school_db.teachers TO 'office_staff_role';

-- 管理者ロール：すべての権限
GRANT ALL PRIVILEGES ON school_db.* TO 'school_admin_role';
GRANT CREATE USER, DROP USER, RELOAD ON *.* TO 'school_admin_role';

-- レポート閲覧ロール：読み取り専用
GRANT SELECT ON school_db.* TO 'report_reader_role';

-- 権限の確認
SHOW GRANTS FOR 'teacher_role';
SHOW GRANTS FOR 'student_role';
```

### 4. ユーザーにロールを割り当て

```
-- 既存ユーザーにロールを付与
GRANT 'school_admin_role' TO 'school_admin'@'localhost';
GRANT 'teacher_role' TO 'teacher_tanaka'@'localhost';
GRANT 'teacher_role' TO 'teacher_sato'@'localhost';
GRANT 'student_role' TO 'student_301'@'localhost';
GRANT 'student_role' TO 'student_302'@'localhost';
GRANT 'office_staff_role' TO 'office_staff'@'localhost';
GRANT 'report_reader_role' TO 'report_viewer'@'localhost';

-- 複数のロールを同時に付与
GRANT 'teacher_role', 'grade_manager' TO 'teacher_tanaka'@'localhost';

-- ロールの継承（ロールから別のロールへ）
GRANT 'grade_manager' TO 'teacher_role';
GRANT 'attendance_manager' TO 'teacher_role';

-- ユーザーの持つロールを確認
SHOW GRANTS FOR 'teacher_tanaka'@'localhost';

-- アクティブなロールを確認
SELECT CURRENT_ROLE();
```

### 5. ロールのアクティベーション

```
-- ログイン時に自動的にロールをアクティベート
ALTER USER 'teacher_tanaka'@'localhost' DEFAULT ROLE ALL;
ALTER USER 'student_301'@'localhost' DEFAULT ROLE ALL;

-- 特定のロールのみを自動アクティベート
ALTER USER 'teacher_sato'@'localhost' DEFAULT ROLE 'teacher_role';

-- セッション中にロールを手動でアクティベート
SET ROLE 'teacher_role';
SET ROLE ALL;
SET ROLE DEFAULT;

-- 現在アクティブなロールを確認
SELECT CURRENT_ROLE();
```

## 実践的なセキュリティ設定

### 1. パスワードポリシーの設定

```
-- パスワード検証プラグインの状況確認
SHOW VARIABLES LIKE 'validate_password%';

-- パスワードポリシーの設定例（グローバル設定）
-- 注意：これらは管理者権限が必要です
-- SET GLOBAL validate_password.length = 12;
-- SET GLOBAL validate_password.mixed_case_count = 1;
-- SET GLOBAL validate_password.number_count = 1;
-- SET GLOBAL validate_password.special_char_count = 1;
-- 強力なパスワードでユーザーを作成
CREATE USER 'secure_teacher'@'localhost'
IDENTIFIED BY 'TeacherSecure123!@#'
PASSWORD EXPIRE INTERVAL 60 DAY
FAILED_LOGIN_ATTEMPTS 3
PASSWORD_LOCK_TIME 1; -- 1日間ロック
```

### 2. 接続制限の設定

```
-- 同時接続数の制限付きユーザー作成
CREATE USER 'limited_user'@'localhost'
IDENTIFIED BY 'LimitedPass2025!'
WITH MAX_CONNECTIONS_PER_HOUR 100
     MAX_UPDATES_PER_HOUR 50
     MAX_QUERIES_PER_HOUR 1000
     MAX_USER_CONNECTIONS 5;

-- 既存ユーザーに制限を追加
ALTER USER 'student_301'@'localhost'
WITH MAX_CONNECTIONS_PER_HOUR 50
     MAX_QUERIES_PER_HOUR 500
     MAX_USER_CONNECTIONS 2;

-- 制限の確認
SELECT
    User, Host, max_connections, max_questions,
    max_updates, max_user_connections
FROM mysql.user
WHERE User = 'student_301';
```

### 3. 監査用のユーザー作成

```
-- 監査ログ専用ユーザー
CREATE USER 'audit_logger'@'localhost'
IDENTIFIED BY 'AuditLog2025!'
PASSWORD EXPIRE NEVER
ACCOUNT LOCK;

-- 監査用ロール
CREATE ROLE 'audit_role';
GRANT SELECT ON mysql.general_log TO 'audit_role';
GRANT SELECT ON mysql.slow_log TO 'audit_role';
GRANT SELECT ON information_schema.* TO 'audit_role';

-- バックアップ専用ユーザー
CREATE USER 'backup_user'@'localhost'
IDENTIFIED BY 'BackupSecure2025!'
PASSWORD EXPIRE INTERVAL 30 DAY;
```

```
CREATE ROLE 'backup_role';
GRANT SELECT, LOCK TABLES, SHOW VIEW, EVENT, TRIGGER ON school_db.* TO
'backup_role';
GRANT 'backup_role' TO 'backup_user'@'localhost';
ALTER USER 'backup_user'@'localhost' DEFAULT ROLE ALL;
```

## エラーと対処法

### 1. ユーザー作成時の一般的なエラー

```
-- エラー例1：既存ユーザーの重複作成
-- CREATE USER 'existing_user'@'localhost' IDENTIFIED BY 'password';
-- エラー: ERROR 1396 (HY000): Operation CREATE USER failed for
'existing_user'@'localhost'

-- 対処法：IF NOT EXISTSを使用
CREATE USER IF NOT EXISTS 'existing_user'@'localhost' IDENTIFIED BY 'password';

-- または、事前にユーザーの存在を確認
SELECT COUNT(*) as user_exists
FROM mysql.user
WHERE User = 'test_user' AND Host = 'localhost';

-- 存在する場合は削除してから作成
DROP USER IF EXISTS 'test_user'@'localhost';
CREATE USER 'test_user'@'localhost' IDENTIFIED BY 'new_password';
```

### 2. 権限関連のエラー

```
-- エラー例2：存在しないデータベースへの権限付与
-- GRANT SELECT ON non_existent_db.* TO 'user'@'localhost';
-- エラー: ERROR 1146 (42S02): Table 'non_existent_db.user' doesn't exist

-- 対処法：データベースの存在確認
SHOW DATABASES LIKE 'school_db';

-- データベースが存在しない場合は作成
CREATE DATABASE IF NOT EXISTS school_db;
GRANT SELECT ON school_db.* TO 'user'@'localhost';
```

### 3. ロール関連のエラー

```
-- エラー例3：MySQL 8.0未満でのロール使用
-- CREATE ROLE 'test_role';
-- エラー: ERROR 1064 (42000): You have an error in your SQL syntax
-- 対処法：MySQLバージョンの確認
SELECT VERSION();

-- MySQL 8.0未満の場合は、ユーザーベースの権限管理を使用
-- または、グループ用のユーザーを作成して権限管理を行う
CREATE USER 'group_teacher'@'localhost' IDENTIFIED BY 'unused_password' ACCOUNT
LOCK;
GRANT SELECT, INSERT, UPDATE ON school_db.grades TO 'group_teacher'@'localhost';
```

### 4. パスワードポリシー違反

```
-- エラー例4：弱いパスワードでのユーザー作成
-- CREATE USER 'weak_user'@'localhost' IDENTIFIED BY '123';
-- エラー: ERROR 1819 (HY000): Your password does not satisfy the current policy
requirements

-- 対処法：パスワードポリシーに準拠したパスワードの使用
CREATE USER 'strong_user'@'localhost' IDENTIFIED BY 'StrongPassword123!';

-- または、一時的にポリシーを確認
SHOW VARIABLES LIKE 'validate_password%';
```

## 運用時のベストプラクティス

### 1. ユーザー管理の標準手順

```
-- 新規ユーザー作成の標準手順
DELIMITER //

CREATE PROCEDURE create_school_user(
    IN p_username VARCHAR(50),
    IN p_host VARCHAR(60),
    IN p_password VARCHAR(255),
    IN p_user_type ENUM('admin', 'teacher', 'student', 'office', 'report'),
    IN p_expire_days INT
)
BEGIN
    DECLARE user_exists INT DEFAULT 0;
    DECLARE role_name VARCHAR(50);

    -- ユーザーの存在確認
    SELECT COUNT(*) INTO user_exists
    FROM mysql.user
    WHERE User = p_username AND Host = p_host;

    IF user_exists > 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'User already exists';
    END IF;
    -- ユーザータイプに応じたロール設定
    SET role_name = CASE p_user_type
        WHEN 'admin' THEN 'school_admin_role'
        WHEN 'teacher' THEN 'teacher_role'
        WHEN 'student' THEN 'student_role'
        WHEN 'office' THEN 'office_staff_role'
        WHEN 'report' THEN 'report_reader_role'
    END;

    -- ユーザー作成
    SET @sql = CONCAT('CREATE USER ''', p_username, '''@''', p_host, '''
IDENTIFIED BY ''', p_password, ''' PASSWORD EXPIRE INTERVAL ', p_expire_days, '
DAY');
    PREPARE stmt FROM @sql;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;

    -- ロール付与
    SET @sql = CONCAT('GRANT ''', role_name, ''' TO ''', p_username, '''@''',
p_host, '''');
    PREPARE stmt FROM @sql;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;

    -- デフォルトロール設定
    SET @sql = CONCAT('ALTER USER ''', p_username, '''@''', p_host, ''' DEFAULT
ROLE ALL');
    PREPARE stmt FROM @sql;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;

    -- ログ記録
    INSERT INTO user_management_log (action, username, host, user_type,
created_by, created_at)
    VALUES ('CREATE', p_username, p_host, p_user_type, USER(), NOW());

END //

DELIMITER ;

-- ログテーブルの作成
CREATE TABLE IF NOT EXISTS user_management_log (
    log_id INT AUTO_INCREMENT PRIMARY KEY,
    action ENUM('CREATE', 'MODIFY', 'DELETE', 'LOCK', 'UNLOCK') NOT NULL,
    username VARCHAR(50) NOT NULL,
    host VARCHAR(60) NOT NULL,
    user_type VARCHAR(20),
    created_by VARCHAR(50),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    notes TEXT,

    INDEX idx_username (username),
    INDEX idx_created_at (created_at)
);

-- プロシージャの使用例
-- CALL create_school_user('new_teacher', 'localhost', 'NewTeacher2025!',
'teacher', 90);
```

## 2. 定期的なユーザー監査

```
-- ユーザー監査レポート生成
CREATE VIEW v_user_audit_report AS
SELECT
    u.User as username,
    u.Host as host,
    u.account_locked,
    u.password_expired,
    u.password_last_changed,
    CASE
        WHEN u.password_lifetime IS NULL THEN 'Never expires'
        ELSE CONCAT(u.password_lifetime, ' days')
    END as password_policy,
    CASE
        WHEN u.password_last_changed IS NULL THEN 'Unknown'
        WHEN DATE_ADD(u.password_last_changed, INTERVAL
IFNULL(u.password_lifetime, 365) DAY) < NOW() THEN 'Expired'
        WHEN DATE_ADD(u.password_last_changed, INTERVAL
IFNULL(u.password_lifetime, 365) DAY) < DATE_ADD(NOW(), INTERVAL 7 DAY) THEN
'Expires Soon'
        ELSE 'Valid'
    END as password_status,
    u.max_connections,
    u.max_user_connections,
    -- 最終ログイン情報（可能な場合）
    COALESCE(acc.TOTAL_CONNECTIONS, 0) as total_login_count,
    COALESCE(acc.CURRENT_CONNECTIONS, 0) as current_connections
FROM mysql.user u
LEFT JOIN performance_schema.accounts acc ON u.User = acc.USER AND u.Host =
acc.HOST
WHERE u.User NOT IN ('mysql.sys', 'mysql.session', 'mysql.infoschema', 'root')
ORDER BY u.User, u.Host;

-- 監査レポートの実行
SELECT * FROM v_user_audit_report;

-- パスワード期限切れユーザーの確認
SELECT username, host, password_last_changed, password_status
FROM v_user_audit_report
WHERE password_status IN ('Expired', 'Expires Soon')
ORDER BY password_last_changed;
```

## 3. セキュリティ設定の確認

```sql
-- セキュリティ関連設定の確認
SELECT
    'Password Validation' as category,
    variable_name,
    variable_value
FROM performance_schema.global_variables
WHERE variable_name LIKE 'validate_password%'

UNION ALL

SELECT
    'SSL/TLS Configuration',
    variable_name,
    variable_value
FROM performance_schema.global_variables
WHERE variable_name IN ('have_ssl', 'ssl_ca', 'ssl_cert', 'ssl_key',
'require_secure_transport')

UNION ALL

SELECT
    'General Security',
    variable_name,
    variable_value
FROM performance_schema.global_variables
WHERE variable_name IN ('local_infile', 'secure_file_priv', 'sql_mode')

ORDER BY category, variable_name;

-- 危険な設定のチェック
SELECT
    CASE
        WHEN COUNT(*) > 0 THEN 'WARNING: Users with weak passwords found'
        ELSE 'OK: No users with empty passwords'
    END as password_check
FROM mysql.user
WHERE authentication_string = '' OR authentication_string IS NULL;

-- 管理者権限を持つユーザーの確認
SELECT DISTINCT
    grantee as privileged_user,
    'Has dangerous privileges' as warning
FROM information_schema.user_privileges
WHERE privilege_type IN ('SUPER', 'CREATE USER', 'GRANT OPTION', 'FILE',
'PROCESS', 'RELOAD', 'SHUTDOWN')
AND grantee NOT LIKE '%root%'
ORDER BY grantee;
```

### 4. 自動化されたユーザー管理

```
-- 期限切れユーザーの自動処理
DELIMITER //

CREATE PROCEDURE maintain_user_accounts()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE v_user VARCHAR(50);
    DECLARE v_host VARCHAR(60);
    DECLARE v_expire_date DATE;

    -- 期限切れユーザーを取得するカーソル
    DECLARE expired_cursor CURSOR FOR
        SELECT User, Host,
               DATE_ADD(password_last_changed, INTERVAL IFNULL(password_lifetime,
90) DAY) as expire_date
        FROM mysql.user
        WHERE password_last_changed IS NOT NULL
        AND DATE_ADD(password_last_changed, INTERVAL IFNULL(password_lifetime, 90)
DAY) < NOW()
        AND account_locked = 'N'
        AND User NOT IN ('root', 'mysql.sys', 'mysql.session',
'mysql.infoschema');

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN expired_cursor;

    read_loop: LOOP
        FETCH expired_cursor INTO v_user, v_host, v_expire_date;
        IF done THEN
            LEAVE read_loop;
        END IF;

        -- アカウントをロック
        SET @sql = CONCAT('ALTER USER ''', v_user, '''@''', v_host, ''' ACCOUNT
LOCK');
        PREPARE stmt FROM @sql;
        EXECUTE stmt;
        DEALLOCATE PREPARE stmt;

        -- ログに記録
        INSERT INTO user_management_log (action, username, host, notes,
created_by)
        VALUES ('LOCK', v_user, v_host,
               CONCAT('Auto-locked due to password expiry on ', v_expire_date),
               'SYSTEM');

    END LOOP;

    CLOSE expired_cursor;

    -- 結果をレポート
    SELECT
```

```
        CONCAT('Locked ', ROW_COUNT(), ' expired user accounts') as
maintenance_result,
        NOW() as maintenance_time;

END //

DELIMITER ;

-- 定期メンテナンスの実行（例：毎日実行）
-- このプロシージャは管理者が定期的に実行するか、
-- cron job等で自動実行するように設定します
-- CALL maintain_user_accounts();
```

## 練習問題

### 問題39-1：基本的なユーザー作成

学校システム用に以下のユーザーを作成してください：

1. `librarian`：図書館司書用ユーザー（ローカルからのみアクセス、パスワード期限90日）
2. `counselor`：学習相談員用ユーザー（学内ネットワーク192.168.1.%からアクセス、パスワード期限180日）
3. `parent_viewer`：保護者用閲覧ユーザー（どこからでもアクセス可、パスワード期限365日） 各ユーザーに適切なパスワードを設定し、作成後にユーザー一覧で確認してください。

### 問題39-2：ロールベースの権限管理

以下の要件でロールを作成し、適切なユーザーに割り当ててください：

1. `library_manager_role`：図書館管理ロール
    - `books`テーブルの全操作権限
    - `book_loans`テーブルの全操作権限
    - `students`テーブルの閲覧権限のみ
2. `guidance_counselor_role`：相談員ロール
    - `students`テーブルの閲覧・更新権限
    - `grades`テーブルの閲覧権限のみ
    - `attendance`テーブルの閲覧権限のみ
3. 問題39-1で作成したユーザーに適切なロールを割り当て、デフォルトロールに設定

### 問題39-3：セキュリティ強化設定

以下のセキュリティ要件を満たすユーザーを作成してください：

1. `security_admin`ユーザー：
    - 特定IPアドレス（192.168.1.10）からのみアクセス可能
    - パスワード期限30日
    - ログイン失敗3回でアカウントロック
    - 同時接続数最大2まで
2. `temp_contractor`ユーザー：
    - 初回ログイン時にパスワード変更を強制

  - 1時間あたりの接続回数を50回に制限
  - 作成時はアカウントロック状態

### 問題39-4：ユーザー監査機能

以下の監査機能を実装してください：

1. パスワード期限が7日以内に切れるユーザーをリストアップするクエリ
2. 過去30日間ログインしていないユーザーを特定するクエリ（可能な範囲で）
3. 管理者権限を持つユーザーをすべて表示するクエリ
4. 同時接続数制限が設定されているユーザーの一覧表示

### 問題39-5：動的ユーザー管理

以下の機能を持つストアドプロシージャを作成してください：

1. `manage_student_user(action, student_id, operation)`：
   - `action`が'CREATE'の場合：学生IDベースでユーザーを作成
   - `action`が'LOCK'の場合：該当ユーザーをロック
   - `action`が'UNLOCK'の場合：該当ユーザーのロックを解除
   - `action`が'DELETE'の場合：該当ユーザーを削除
   - すべての操作をログテーブルに記録
2. 5名の架空の学生ユーザーで動作テストを実行

### 問題39-6：包括的なユーザー管理システム

学校システムの完全なユーザー管理システムを設計・実装してください：

1. 以下の役職すべてに対応したロールとユーザーの作成：
   - 校長（全データアクセス可能）
   - 教務主任（成績・出席管理）
   - 学年主任（担当学年のみ管理）
   - 一般教師（担当クラスのみ管理）
   - 事務職員（学生情報管理）
   - 保健室職員（健康情報管理）
   - 図書館司書（図書管理）
   - システム管理者（ユーザー管理）
2. 各役職の責任範囲に応じた適切な権限設定
3. ユーザー作成・管理の自動化プロシージャ
4. セキュリティ監査機能
5. 運用ドキュメントの作成（コメント形式）

## 解答

### 解答39-1

```
-- 1. 図書館司書用ユーザー
CREATE USER 'librarian'@'localhost'
IDENTIFIED BY 'LibrarianSecure2025!'
```

```
PASSWORD EXPIRE INTERVAL 90 DAY;

-- 2. 学習相談員用ユーザー
CREATE USER 'counselor'@'192.168.1.%'
IDENTIFIED BY 'CounselorSecure2025!'
PASSWORD EXPIRE INTERVAL 180 DAY;

-- 3. 保護者用閲覧ユーザー
CREATE USER 'parent_viewer'@'%'
IDENTIFIED BY 'ParentView2025!'
PASSWORD EXPIRE INTERVAL 365 DAY;

-- 作成したユーザーの確認
SELECT
    User as username,
    Host as allowed_host,
    password_expired,
    password_lifetime as password_expires_days,
    account_locked
FROM mysql.user
WHERE User IN ('librarian', 'counselor', 'parent_viewer')
ORDER BY User;

-- より詳細な確認
SHOW GRANTS FOR 'librarian'@'localhost';
SHOW GRANTS FOR 'counselor'@'192.168.1.%';
SHOW GRANTS FOR 'parent_viewer'@'%';
```

解答39-2

```
-- 1. ロールの作成
CREATE ROLE 'library_manager_role';
CREATE ROLE 'guidance_counselor_role';

-- 2. library_manager_roleの権限設定
GRANT SELECT, INSERT, UPDATE, DELETE ON school_db.books TO 'library_manager_role';
GRANT SELECT, INSERT, UPDATE, DELETE ON school_db.book_loans TO
'library_manager_role';
GRANT SELECT ON school_db.students TO 'library_manager_role';

-- 3. guidance_counselor_roleの権限設定
GRANT SELECT, UPDATE ON school_db.students TO 'guidance_counselor_role';
GRANT SELECT ON school_db.grades TO 'guidance_counselor_role';
GRANT SELECT ON school_db.attendance TO 'guidance_counselor_role';

-- 4. ユーザーにロールを割り当て
GRANT 'library_manager_role' TO 'librarian'@'localhost';
GRANT 'guidance_counselor_role' TO 'counselor'@'192.168.1.%';

-- 保護者用は閲覧のみのロールを作成
CREATE ROLE 'parent_viewer_role';
```

```
GRANT SELECT ON school_db.students TO 'parent_viewer_role';
GRANT SELECT ON school_db.grades TO 'parent_viewer_role';
GRANT 'parent_viewer_role' TO 'parent_viewer'@'%';

-- 5. デフォルトロールの設定
ALTER USER 'librarian'@'localhost' DEFAULT ROLE ALL;
ALTER USER 'counselor'@'192.168.1.%' DEFAULT ROLE ALL;
ALTER USER 'parent_viewer'@'%' DEFAULT ROLE ALL;

-- 権限の確認
SHOW GRANTS FOR 'library_manager_role';
SHOW GRANTS FOR 'guidance_counselor_role';
SHOW GRANTS FOR 'parent_viewer_role';

-- ユーザーの権限確認
SHOW GRANTS FOR 'librarian'@'localhost';
SHOW GRANTS FOR 'counselor'@'192.168.1.%';
SHOW GRANTS FOR 'parent_viewer'@'%';
```

## 解答39-3

```
-- 1. security_adminユーザーの作成
CREATE USER 'security_admin'@'192.168.1.10'
IDENTIFIED BY 'SecurityAdmin2025!@#'
PASSWORD EXPIRE INTERVAL 30 DAY
FAILED_LOGIN_ATTEMPTS 3
PASSWORD_LOCK_TIME 1
WITH MAX_USER_CONNECTIONS 2;

-- 2. temp_contractorユーザーの作成
CREATE USER 'temp_contractor'@'localhost'
IDENTIFIED BY 'TempInitial2025!'
PASSWORD EXPIRE
WITH MAX_CONNECTIONS_PER_HOUR 50
ACCOUNT LOCK;

-- 作成結果の確認
SELECT
    User,
    Host,
    account_locked,
    password_expired,
    password_lifetime,
    Failed_login_attempts,
    Password_lock_time,
    max_connections,
    max_user_connections
FROM mysql.user
WHERE User IN ('security_admin', 'temp_contractor');

-- 接続制限の詳細確認
```

```
SELECT
    User,
    Host,
    max_questions as hourly_queries,
    max_updates as hourly_updates,
    max_connections as hourly_connections,
    max_user_connections as concurrent_connections
FROM mysql.user
WHERE User IN ('security_admin', 'temp_contractor');
```

## 解答39-4

```
-- 1. パスワード期限が7日以内に切れるユーザーのリストアップ
SELECT
    User as username,
    Host as host,
    password_last_changed,
    password_lifetime,
    DATE_ADD(password_last_changed, INTERVAL password_lifetime DAY) as expires_on,
    DATEDIFF(DATE_ADD(password_last_changed, INTERVAL password_lifetime DAY),
CURRENT_DATE) as days_until_expiry
FROM mysql.user
WHERE password_last_changed IS NOT NULL
    AND password_lifetime IS NOT NULL
    AND DATE_ADD(password_last_changed, INTERVAL password_lifetime DAY) <=
DATE_ADD(CURRENT_DATE, INTERVAL 7 DAY)
    AND DATE_ADD(password_last_changed, INTERVAL password_lifetime DAY) >=
CURRENT_DATE
    AND User NOT IN ('root', 'mysql.sys', 'mysql.session', 'mysql.infoschema')
ORDER BY expires_on;

-- 2. 過去30日間ログインしていないユーザー（performance_schemaを使用）
SELECT
    u.User as username,
    u.Host as host,
    COALESCE(acc.TOTAL_CONNECTIONS, 0) as total_login_count,
    COALESCE(acc.CURRENT_CONNECTIONS, 0) as current_connections,
    CASE
        WHEN acc.USER IS NULL THEN 'Never logged in'
        WHEN acc.TOTAL_CONNECTIONS = 0 THEN 'Never logged in'
        ELSE 'Login history exists (detailed tracking not available in this
version)'
    END as login_status
FROM mysql.user u
LEFT JOIN performance_schema.accounts acc ON u.User = acc.USER AND u.Host =
acc.HOST
WHERE u.User NOT IN ('root', 'mysql.sys', 'mysql.session', 'mysql.infoschema')
ORDER BY u.User;

-- 3. 管理者権限を持つユーザーの表示
SELECT DISTINCT
```

```
    up.grantee as privileged_user,
    GROUP_CONCAT(DISTINCT up.privilege_type ORDER BY up.privilege_type) as
dangerous_privileges
FROM information_schema.user_privileges up
WHERE up.privilege_type IN ('SUPER', 'CREATE USER', 'GRANT OPTION', 'FILE',
'PROCESS', 'RELOAD', 'SHUTDOWN', 'ALL PRIVILEGES')
    AND up.grantee NOT LIKE '%root%@%'
GROUP BY up.grantee
ORDER BY up.grantee;

-- グローバル権限の詳細確認
SELECT
    grantee,
    privilege_type,
    is_grantable
FROM information_schema.user_privileges
WHERE privilege_type IN ('SUPER', 'CREATE USER', 'GRANT OPTION', 'ALL PRIVILEGES')
ORDER BY grantee, privilege_type;

-- 4. 同時接続数制限が設定されているユーザーの一覧
SELECT
    User as username,
    Host as host,
    max_questions as hourly_queries_limit,
    max_updates as hourly_updates_limit,
    max_connections as hourly_connections_limit,
    max_user_connections as concurrent_connections_limit,
    CASE
        WHEN max_user_connections > 0 OR max_connections > 0 OR max_questions > 0
OR max_updates > 0
        THEN 'Has Limits'
        ELSE 'No Limits'
    END as limitation_status
FROM mysql.user
WHERE (max_user_connections > 0 OR max_connections > 0 OR max_questions > 0 OR
max_updates > 0)
    AND User NOT IN ('root', 'mysql.sys', 'mysql.session', 'mysql.infoschema')
ORDER BY User;
```

### 解答39-5

```
-- ログテーブルの作成（まだ存在しない場合）
CREATE TABLE IF NOT EXISTS user_management_log (
    log_id INT AUTO_INCREMENT PRIMARY KEY,
    action ENUM('CREATE', 'MODIFY', 'DELETE', 'LOCK', 'UNLOCK') NOT NULL,
    username VARCHAR(50) NOT NULL,
    host VARCHAR(60) NOT NULL,
    user_type VARCHAR(20),
    created_by VARCHAR(50),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    notes TEXT,
    INDEX idx_username (username),
    INDEX idx_created_at (created_at),
    INDEX idx_action (action)
);

-- 学生ユーザー管理プロシージャ
DELIMITER //

CREATE PROCEDURE manage_student_user(
    IN p_action ENUM('CREATE', 'LOCK', 'UNLOCK', 'DELETE'),
    IN p_student_id BIGINT,
    IN p_operation VARCHAR(100)
)
BEGIN
    DECLARE v_username VARCHAR(50);
    DECLARE v_host VARCHAR(60) DEFAULT 'localhost';
    DECLARE v_password VARCHAR(50);
    DECLARE user_exists INT DEFAULT 0;
    DECLARE operation_result VARCHAR(200);

    -- 学生IDからユーザー名を生成
    SET v_username = CONCAT('student_', p_student_id);
    SET v_password = CONCAT('Student', p_student_id, 'Pass2025!');

    -- ユーザーの存在確認
    SELECT COUNT(*) INTO user_exists
    FROM mysql.user
    WHERE User = v_username AND Host = v_host;

    CASE p_action
        WHEN 'CREATE' THEN
            IF user_exists > 0 THEN
                SET operation_result = 'ERROR: User already exists';
            ELSE
                -- ユーザー作成
                SET @sql = CONCAT('CREATE USER ''', v_username, '''@''', v_host,
''' IDENTIFIED BY ''', v_password, ''' PASSWORD EXPIRE INTERVAL 180 DAY');
                PREPARE stmt FROM @sql;
                EXECUTE stmt;
                DEALLOCATE PREPARE stmt;

                -- 学生ロールが存在する場合は付与
                IF EXISTS (SELECT 1 FROM mysql.user WHERE User = 'student_role'
AND Host = '') THEN
                    SET @sql = CONCAT('GRANT ''student_role'' TO ''', v_username,
'''@''', v_host, '''');
                    PREPARE stmt FROM @sql;
                    EXECUTE stmt;
                    DEALLOCATE PREPARE stmt;

                    SET @sql = CONCAT('ALTER USER ''', v_username, '''@''',
v_host, ''' DEFAULT ROLE ALL');
                    PREPARE stmt FROM @sql;
                    EXECUTE stmt;
                    DEALLOCATE PREPARE stmt;
                END IF;

                SET operation_result = 'SUCCESS: User created';
            END IF;

        WHEN 'LOCK' THEN
            IF user_exists = 0 THEN
                SET operation_result = 'ERROR: User does not exist';
            ELSE
                SET @sql = CONCAT('ALTER USER ''', v_username, '''@''', v_host,
''' ACCOUNT LOCK');
                PREPARE stmt FROM @sql;
                EXECUTE stmt;
                DEALLOCATE PREPARE stmt;
                SET operation_result = 'SUCCESS: User locked';
            END IF;

        WHEN 'UNLOCK' THEN
            IF user_exists = 0 THEN
                SET operation_result = 'ERROR: User does not exist';
            ELSE
                SET @sql = CONCAT('ALTER USER ''', v_username, '''@''', v_host,
''' ACCOUNT UNLOCK');
                PREPARE stmt FROM @sql;
                EXECUTE stmt;
                DEALLOCATE PREPARE stmt;
                SET operation_result = 'SUCCESS: User unlocked';
            END IF;

        WHEN 'DELETE' THEN
            IF user_exists = 0 THEN
                SET operation_result = 'ERROR: User does not exist';
            ELSE
                SET @sql = CONCAT('DROP USER ''', v_username, '''@''', v_host,
'''');
                PREPARE stmt FROM @sql;
                EXECUTE stmt;
                DEALLOCATE PREPARE stmt;
                SET operation_result = 'SUCCESS: User deleted';
            END IF;

        ELSE
            SET operation_result = 'ERROR: Invalid action';
    END CASE;

    -- ログに記録
    INSERT INTO user_management_log (action, username, host, user_type,
created_by, notes)
    VALUES (p_action, v_username, v_host, 'student', USER(),
            CONCAT(p_operation, ' - Result: ', operation_result));

    -- 結果を返す
    SELECT
        p_action as requested_action,
        v_username as target_user,
        operation_result as result,
        NOW() as operation_time;

END //

DELIMITER ;

-- テスト実行：5名の架空学生ユーザーで動作テスト

-- 1. ユーザー作成テスト
CALL manage_student_user('CREATE', 1001, 'Create test student 1001');
CALL manage_student_user('CREATE', 1002, 'Create test student 1002');
CALL manage_student_user('CREATE', 1003, 'Create test student 1003');
CALL manage_student_user('CREATE', 1004, 'Create test student 1004');
CALL manage_student_user('CREATE', 1005, 'Create test student 1005');

-- 2. 作成されたユーザーの確認
SELECT User, Host, account_locked, password_expired
FROM mysql.user
WHERE User LIKE 'student_100%'
ORDER BY User;

-- 3. ロック/アンロックテスト
CALL manage_student_user('LOCK', 1001, 'Lock test for student 1001');
CALL manage_student_user('UNLOCK', 1001, 'Unlock test for student 1001');

-- 4. 削除テスト
CALL manage_student_user('DELETE', 1005, 'Delete test for student 1005');

-- 5. エラーケーステスト
CALL manage_student_user('CREATE', 1001, 'Duplicate creation test'); -- 重複作成エラー
CALL manage_student_user('DELETE', 9999, 'Delete non-existent user test'); -- 存在しないユーザー削除

-- 6. ログの確認
SELECT
    action,
    username,
    notes,
    created_by,
    created_at
FROM user_management_log
WHERE username LIKE 'student_100%'
ORDER BY created_at DESC;

-- 7. 最終状態確認
SELECT
    User as username,
    Host as host,
    account_locked,
    password_expired,
    'Active' as status
FROM mysql.user
WHERE User LIKE 'student_100%'
ORDER BY User;
```

## 解答39-6

```
-- 包括的なユーザー管理システム

-- 1. 役職別ロールの作成
CREATE ROLE 'principal_role';          -- 校長
CREATE ROLE 'academic_director_role';  -- 教務主任
CREATE ROLE 'grade_supervisor_role';   -- 学年主任
CREATE ROLE 'teacher_role';            -- 一般教師
CREATE ROLE 'office_staff_role';       -- 事務職員
CREATE ROLE 'health_staff_role';       -- 保健室職員
CREATE ROLE 'librarian_role';          -- 図書館司書
CREATE ROLE 'system_admin_role';       -- システム管理者

-- 2. 権限設定

-- 校長ロール：全データアクセス可能
GRANT ALL PRIVILEGES ON school_db.* TO 'principal_role';
GRANT SELECT ON mysql.user TO 'principal_role';

-- 教務主任ロール：成績・出席管理
GRANT SELECT, INSERT, UPDATE, DELETE ON school_db.grades TO
'academic_director_role';
GRANT SELECT, INSERT, UPDATE, DELETE ON school_db.attendance TO
'academic_director_role';
GRANT SELECT, INSERT, UPDATE ON school_db.courses TO 'academic_director_role';
GRANT SELECT, INSERT, UPDATE ON school_db.course_schedule TO
'academic_director_role';
GRANT SELECT ON school_db.students TO 'academic_director_role';
GRANT SELECT ON school_db.teachers TO 'academic_director_role';

-- 学年主任ロール：担当学年のみ管理
GRANT SELECT, INSERT, UPDATE ON school_db.students TO 'grade_supervisor_role';
GRANT SELECT, INSERT, UPDATE ON school_db.grades TO 'grade_supervisor_role';
GRANT SELECT, INSERT, UPDATE ON school_db.attendance TO 'grade_supervisor_role';
GRANT SELECT ON school_db.courses TO 'grade_supervisor_role';
GRANT SELECT ON school_db.teachers TO 'grade_supervisor_role';

-- 一般教師ロール：担当クラスのみ管理
GRANT SELECT ON school_db.students TO 'teacher_role';
GRANT SELECT, INSERT, UPDATE ON school_db.grades TO 'teacher_role';
GRANT SELECT, INSERT, UPDATE ON school_db.attendance TO 'teacher_role';
GRANT SELECT ON school_db.courses TO 'teacher_role';

-- 事務職員ロール：学生情報管理
```

```sql
GRANT SELECT, INSERT, UPDATE ON school_db.students TO 'office_staff_role';
GRANT SELECT ON school_db.courses TO 'office_staff_role';
GRANT SELECT ON school_db.teachers TO 'office_staff_role';
GRANT SELECT, INSERT, UPDATE ON school_db.student_courses TO 'office_staff_role';

-- 保健室職員ロール：健康情報管理
GRANT SELECT ON school_db.students TO 'health_staff_role';
-- 健康情報テーブルがある場合
-- GRANT SELECT, INSERT, UPDATE ON school_db.health_records TO
'health_staff_role';

-- 図書館司書ロール：図書管理
-- 図書関連テーブルがある場合の権限設定例
-- GRANT SELECT, INSERT, UPDATE, DELETE ON school_db.books TO 'librarian_role';
-- GRANT SELECT, INSERT, UPDATE, DELETE ON school_db.book_loans TO
'librarian_role';
GRANT SELECT ON school_db.students TO 'librarian_role';

-- システム管理者ロール：ユーザー管理
GRANT CREATE USER, DROP USER, RELOAD ON *.* TO 'system_admin_role';
GRANT SELECT, INSERT, UPDATE ON mysql.user TO 'system_admin_role';
GRANT ALL PRIVILEGES ON school_db.user_management_log TO 'system_admin_role';

-- 3. 役職別ユーザーの作成プロシージャ
DELIMITER //

CREATE PROCEDURE create_school_staff_user(
    IN p_username VARCHAR(50),
    IN p_password VARCHAR(255),
    IN p_position ENUM('principal', 'academic_director', 'grade_supervisor',
'teacher', 'office_staff', 'health_staff', 'librarian', 'system_admin'),
    IN p_host VARCHAR(60) DEFAULT 'localhost',
    IN p_expire_days INT DEFAULT 90
)
BEGIN
    DECLARE v_role_name VARCHAR(50);
    DECLARE user_exists INT DEFAULT 0;

    -- ユーザーの存在確認
    SELECT COUNT(*) INTO user_exists
    FROM mysql.user
    WHERE User = p_username AND Host = p_host;

    IF user_exists > 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'User already exists';
    END IF;

    -- 役職に応じたロール名の設定
    SET v_role_name = CASE p_position
        WHEN 'principal' THEN 'principal_role'
        WHEN 'academic_director' THEN 'academic_director_role'
        WHEN 'grade_supervisor' THEN 'grade_supervisor_role'
        WHEN 'teacher' THEN 'teacher_role'
        WHEN 'office_staff' THEN 'office_staff_role'
        WHEN 'health_staff' THEN 'health_staff_role'
        WHEN 'librarian' THEN 'librarian_role'
        WHEN 'system_admin' THEN 'system_admin_role'
    END;

    -- ユーザー作成
    SET @sql = CONCAT('CREATE USER ''', p_username, '''@''', p_host, ''' IDENTIFIED BY ''', p_password, ''' PASSWORD EXPIRE INTERVAL ', p_expire_days, ' DAY');
    PREPARE stmt FROM @sql;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;

    -- ロール付与
    SET @sql = CONCAT('GRANT ''', v_role_name, ''' TO ''', p_username, '''@''', p_host, '''');
    PREPARE stmt FROM @sql;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;

    -- デフォルトロール設定
    SET @sql = CONCAT('ALTER USER ''', p_username, '''@''', p_host, ''' DEFAULT ROLE ALL');
    PREPARE stmt FROM @sql;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;

    -- 追加の制限設定（役職に応じて）
    IF p_position IN ('teacher', 'office_staff', 'health_staff', 'librarian') THEN
        SET @sql = CONCAT('ALTER USER ''', p_username, '''@''', p_host, ''' WITH MAX_USER_CONNECTIONS 3');
        PREPARE stmt FROM @sql;
        EXECUTE stmt;
        DEALLOCATE PREPARE stmt;
    END IF;

    -- ログ記録
    INSERT INTO user_management_log (action, username, host, user_type, created_by, notes)
    VALUES ('CREATE', p_username, p_host, p_position, USER(),
            CONCAT('Created ', p_position, ' user with role: ', v_role_name));

    -- 結果返却
    SELECT
        p_username as created_user,
        p_host as host,
        p_position as position,
        v_role_name as assigned_role,
        'SUCCESS' as result;

END //

DELIMITER ;
```

```sql
-- 4. セキュリティ監査機能
CREATE VIEW v_comprehensive_user_audit AS
SELECT
    u.User as username,
    u.Host as host,
    CASE
        WHEN u.User LIKE '%principal%' THEN 'Principal'
        WHEN u.User LIKE '%academic%' THEN 'Academic Director'
        WHEN u.User LIKE '%grade%' THEN 'Grade Supervisor'
        WHEN u.User LIKE '%teacher%' THEN 'Teacher'
        WHEN u.User LIKE '%office%' THEN 'Office Staff'
        WHEN u.User LIKE '%health%' THEN 'Health Staff'
        WHEN u.User LIKE '%librarian%' THEN 'Librarian'
        WHEN u.User LIKE '%admin%' THEN 'System Admin'
        WHEN u.User LIKE '%student%' THEN 'Student'
        ELSE 'Other'
    END as estimated_role,
    u.account_locked,
    u.password_expired,
    u.password_last_changed,
    u.password_lifetime,
    CASE
        WHEN u.password_last_changed IS NULL THEN 'Never changed'
        WHEN DATE_ADD(u.password_last_changed, INTERVAL
IFNULL(u.password_lifetime, 90) DAY) < NOW() THEN 'EXPIRED'
        WHEN DATE_ADD(u.password_last_changed, INTERVAL
IFNULL(u.password_lifetime, 90) DAY) < DATE_ADD(NOW(), INTERVAL 7 DAY) THEN
'EXPIRES SOON'
        ELSE 'Valid'
    END as password_status,
    u.max_user_connections,
    COALESCE(acc.TOTAL_CONNECTIONS, 0) as total_logins,
    COALESCE(acc.CURRENT_CONNECTIONS, 0) as current_sessions
FROM mysql.user u
LEFT JOIN performance_schema.accounts acc ON u.User = acc.USER AND u.Host =
acc.HOST
WHERE u.User NOT IN ('mysql.sys', 'mysql.session', 'mysql.infoschema', 'root')
ORDER BY
    CASE estimated_role
        WHEN 'Principal' THEN 1
        WHEN 'System Admin' THEN 2
        WHEN 'Academic Director' THEN 3
        WHEN 'Grade Supervisor' THEN 4
        WHEN 'Teacher' THEN 5
        WHEN 'Office Staff' THEN 6
        WHEN 'Health Staff' THEN 7
        WHEN 'Librarian' THEN 8
        WHEN 'Student' THEN 9
        ELSE 10
    END,
    u.User;

-- 5. サンプルユーザーの作成（コメントアウト状態）
/*
```

```
CALL create_school_staff_user('principal_yamada', 'PrincipalSecure2025!',
'principal', 'localhost', 60);
CALL create_school_staff_user('academic_suzuki', 'AcademicSecure2025!',
'academic_director', 'localhost', 90);
CALL create_school_staff_user('grade_head_tanaka', 'GradeSecure2025!',
'grade_supervisor', 'localhost', 90);
CALL create_school_staff_user('teacher_sato', 'TeacherSecure2025!', 'teacher',
'localhost', 120);
CALL create_school_staff_user('office_kimura', 'OfficeSecure2025!',
'office_staff', 'localhost', 90);
CALL create_school_staff_user('nurse_takahashi', 'HealthSecure2025!',
'health_staff', 'localhost', 120);
CALL create_school_staff_user('librarian_watanabe', 'LibrarySecure2025!',
'librarian', 'localhost', 120);
CALL create_school_staff_user('sysadmin_ito', 'SystemSecure2025!', 'system_admin',
'localhost', 30);
*/

-- 6. 定期メンテナンスプロシージャ
DELIMITER //

CREATE PROCEDURE comprehensive_user_maintenance()
BEGIN
    DECLARE maintenance_summary TEXT DEFAULT '';

    -- 期限切れアカウントのロック
    UPDATE mysql.user
    SET account_locked = 'Y'
    WHERE password_last_changed IS NOT NULL
    AND DATE_ADD(password_last_changed, INTERVAL IFNULL(password_lifetime, 90)
DAY) < NOW()
    AND account_locked = 'N'
    AND User NOT IN ('root', 'mysql.sys', 'mysql.session', 'mysql.infoschema');

    SET maintenance_summary = CONCAT('Locked ', ROW_COUNT(), ' expired accounts.
');

    -- 警告が必要なアカウントのカウント
    SELECT COUNT(*) INTO @warning_count
    FROM mysql.user
    WHERE password_last_changed IS NOT NULL
    AND DATE_ADD(password_last_changed, INTERVAL IFNULL(password_lifetime, 90)
DAY) <= DATE_ADD(NOW(), INTERVAL 7 DAY)
    AND DATE_ADD(password_last_changed, INTERVAL IFNULL(password_lifetime, 90)
DAY) > NOW()
    AND User NOT IN ('root', 'mysql.sys', 'mysql.session', 'mysql.infoschema');

    SET maintenance_summary = CONCAT(maintenance_summary, @warning_count, '
accounts expire within 7 days.');

    -- メンテナンスログ
    INSERT INTO user_management_log (action, username, host, created_by, notes)
    VALUES ('MAINTENANCE', 'SYSTEM', 'SYSTEM', 'AUTO_MAINTENANCE',
maintenance_summary);
```

```
        -- 結果レポート
        SELECT
            'User Maintenance Completed' as status,
            maintenance_summary as summary,
            NOW() as maintenance_time;

    END //

    DELIMITER ;

    -- 7. 運用ドキュメント（コメント形式）
    /*
    === 学校システム ユーザー管理運用ガイド ===

    【役職別権限一覧】
    1. 校長 (principal_role)
       - 全データベースへの完全アクセス権限
       - ユーザー情報の閲覧権限
       - 最高責任者として全権限を保有

    2. 教務主任 (academic_director_role)
       - 成績データの完全管理権限
       - 出席データの完全管理権限
       - 講座・スケジュールの管理権限
       - 学生・教師情報の閲覧権限

    3. 学年主任 (grade_supervisor_role)
       - 担当学年の学生情報管理権限
       - 成績・出席データの管理権限
       - 講座・教師情報の閲覧権限

    4. 一般教師 (teacher_role)
       - 担当クラスの成績入力・修正権限
       - 担当クラスの出席管理権限
       - 学生・講座情報の閲覧権限

    5. 事務職員 (office_staff_role)
       - 学生基本情報の管理権限
       - 受講情報の管理権限
       - 講座・教師情報の閲覧権限

    6. 保健室職員 (health_staff_role)
       - 学生基本情報の閲覧権限
       - 健康関連情報の管理権限（該当テーブル存在時）

    7. 図書館司書 (librarian_role)
       - 図書関連データの管理権限（該当テーブル存在時）
       - 学生基本情報の閲覧権限

    8. システム管理者 (system_admin_role)
       - ユーザー管理の完全権限
       - システム関連権限
       - 監査ログの管理権限
```

```
【セキュリティ設定】
- パスワード有効期限：校長・システム管理者30-60日、その他90-120日
- 同時接続制限：一般ユーザー3接続まで
- 接続元制限：必要に応じてIPアドレス制限を実装
- 自動ロック：パスワード期限切れで自動ロック

【定期メンテナンス】
1. 毎日：comprehensive_user_maintenance()の実行
2. 週次：v_comprehensive_user_audit視野でのユーザー状況確認
3. 月次：不要アカウントの棚卸し・削除
4. 四半期：権限設定の見直し

【緊急時対応】
1. アカウント侵害疑い：即座にACCOUNT LOCK実行
2. 大量ログイン失敗：該当アカウントの確認・一時ロック
3. 権限昇格攻撃：システム管理者アカウントの緊急確認

【問い合わせ対応】
1. パスワードリセット：管理者がALTER USERで対応
2. 権限追加要求：役職に応じた適切な権限確認後対応
3. アカウント無効化：退職者等のアカウント即座にロック・削除

【監査要求対応】
1. v_comprehensive_user_audit視野でのレポート生成
2. user_management_logテーブルでの操作履歴確認
3. 定期的なアクセスログの分析・報告
*/

-- 監査レポートの確認
SELECT * FROM v_comprehensive_user_audit;

-- 最新の管理ログ確認
SELECT * FROM user_management_log ORDER BY created_at DESC LIMIT 10;
```

## まとめ

この章では、MySQLにおけるユーザーとロールの管理について詳しく学びました：

1. **ユーザー管理の基本概念**：
    - ユーザーアカウントの構成（ユーザー名@ホスト名）
    - 認証と認可の違い
    - 権限の階層構造（グローバル→データベース→テーブル→カラム）
2. **基本的なユーザー操作**：
    - CREATE USER文によるユーザー作成
    - ALTER USER文によるユーザー設定変更
    - DROP USER文によるユーザー削除
    - パスワードポリシーと有効期限の設定

3. **ロールベースアクセス制御**：
    - MySQL 8.0以降のロール機能
    - ロールの作成と権限付与
    - ユーザーへのロール割り当て
    - ロールのアクティベーション管理
4. **学校システムでの実践応用**：
    - 役職別ロールの設計
    - 適切な権限分離
    - セキュリティポリシーの実装
    - 接続制限とアカウント管理
5. **セキュリティ強化策**：
    - 強力なパスワードポリシー
    - 接続元制限とアクセス制御
    - アカウントロック機能
    - 定期的な監査とメンテナンス
6. **運用管理手法**：
    - 自動化されたユーザー管理プロシージャ
    - 包括的な監査システム
    - エラー処理と例外対応
    - 運用ドキュメントの整備

ユーザー管理は、データベースセキュリティの最も基本的で重要な要素です。**最小権限の原則**を守り、各ユーザーには業務に必要な最小限の権限のみを付与することが、安全なデータベース運用の鍵となります。

次の章では、「GRANT/REVOKE：権限の付与と取り消し」について学び、より詳細な権限管理の手法を理解していきます。

---

# 40. GRANT/REVOKE：権限の付与と取り消し

## はじめに

前章では、ユーザーとロールの基本的な作成・管理方法について学習しました。この章では、作成したユーザーやロールに対して具体的な権限を付与する「GRANT文」と、権限を取り消す「REVOKE文」について詳しく学習します。

権限管理は、データベースセキュリティの中核となる重要な機能です。学校システムを例に取ると、「教師は担当クラスの成績のみ変更できる」「学生は自分の成績のみ閲覧できる」「事務職員は学生の基本情報を管理できるが成績は見られない」といった、きめ細かな権限制御が必要になります。

GRANT/REVOKEが必要となる場面の例：

- 「新任の教師に成績入力権限を付与したい」

- 「退職する職員からすべての権限を取り消したい」
- 「臨時職員には期間限定で特定のテーブルの閲覧権限のみ与えたい」
- 「システム管理者から一部の危険な権限を一時的に取り消したい」
- 「部門異動により、職員の担当データへのアクセス権限を変更したい」
- 「外部監査のために、監査法人に読み取り専用権限を付与したい」
- 「セキュリティインシデント発生時に、該当ユーザーの権限を緊急停止したい」

この章では、基本的なGRANT/REVOKE文から、複雑な権限管理まで、実践的な例を通じて学習していきます。

## 権限管理の基本概念

### 権限の種類

MySQLでは、以下のような種類の権限が定義されています：

> **用語解説**：
>
> - **GRANT文**：ユーザーやロールに権限を付与するSQL文です。
> - **REVOKE文**：ユーザーやロールから権限を取り消すSQL文です。
> - **権限（Privilege）**：データベースで実行できる特定の操作を表します（SELECT、INSERT等）。
> - **グランター（Grantor）**：権限を付与する側のユーザーです。
> - **グランティー（Grantee）**：権限を付与される側のユーザーまたはロールです。
> - **WITH GRANT OPTION**：付与された権限を他のユーザーにさらに付与する権限です。
> - **権限の継承**：ロールからユーザーへ、またはロールから別のロールへ権限が継承される仕組みです。
> - **権限の累積**：複数の権限付与により、ユーザーが持つ権限が積み重なることです。
> - **最小権限の原則**：ユーザーには業務に必要な最小限の権限のみを付与するセキュリティの基本原則です。

### データベース権限の分類

| 権限カテゴリ | 主な権限 | 説明 |
|---|---|---|
| **データ操作権限** | SELECT, INSERT, UPDATE, DELETE | テーブルのデータを操作する権限 |
| **構造管理権限** | CREATE, ALTER, DROP, INDEX | データベース構造を変更する権限 |
| **実行権限** | EXECUTE | ストアドプロシージャや関数を実行する権限 |
| **管理権限** | CREATE USER, GRANT OPTION, RELOAD | ユーザー管理やシステム管理の権限 |
| **特殊権限** | FILE, PROCESS, SUPER | ファイル操作やシステム制御の権限 |

### 権限の適用レベル

```
グローバルレベル：      GRANT SELECT ON *.* TO user;
データベースレベル：    GRANT SELECT ON database.* TO user;
テーブルレベル：        GRANT SELECT ON database.table TO user;
カラムレベル：          GRANT SELECT (column1, column2) ON database.table TO user;
```

## GRANT文の基本文法

### 1. 基本的なGRANT文

```
-- 基本文法
GRANT 権限 ON 対象 TO ユーザー [WITH GRANT OPTION];

-- 単一権限の付与
GRANT SELECT ON school_db.students TO 'teacher1'@'localhost';

-- 複数権限の同時付与
GRANT SELECT, INSERT, UPDATE ON school_db.grades TO 'teacher1'@'localhost';

-- 全権限の付与
GRANT ALL PRIVILEGES ON school_db.* TO 'admin_user'@'localhost';
```

### 2. 学校システムでの基本的な権限付与例

```
-- 教師ユーザーに成績管理権限を付与
GRANT SELECT, INSERT, UPDATE ON school_db.grades TO 'teacher_tanaka'@'localhost';
GRANT SELECT, INSERT, UPDATE ON school_db.attendance TO
'teacher_tanaka'@'localhost';
GRANT SELECT ON school_db.students TO 'teacher_tanaka'@'localhost';
GRANT SELECT ON school_db.courses TO 'teacher_tanaka'@'localhost';

-- 学生ユーザーに自分の情報閲覧権限を付与
GRANT SELECT ON school_db.students TO 'student_301'@'localhost';
GRANT SELECT ON school_db.grades TO 'student_301'@'localhost';
GRANT SELECT ON school_db.attendance TO 'student_301'@'localhost';

-- 事務職員に学生情報管理権限を付与
GRANT SELECT, INSERT, UPDATE ON school_db.students TO 'office_staff'@'localhost';
GRANT SELECT, INSERT, UPDATE ON school_db.student_courses TO
'office_staff'@'localhost';
GRANT SELECT ON school_db.courses TO 'office_staff'@'localhost';

-- 権限付与の確認
SHOW GRANTS FOR 'teacher_tanaka'@'localhost';
SHOW GRANTS FOR 'student_301'@'localhost';
SHOW GRANTS FOR 'office_staff'@'localhost';
```

### 3. カラムレベルの権限付与

```
-- 学生の個人情報の一部のみアクセス可能にする
GRANT SELECT (student_id, student_name) ON school_db.students TO
'limited_viewer'@'localhost';

-- 成績の特定項目のみ更新可能にする
GRANT UPDATE (score, submission_date) ON school_db.grades TO
'grade_assistant'@'localhost';

-- カラム権限と通常権限の組み合わせ
GRANT SELECT ON school_db.students TO 'counselor'@'localhost';
GRANT UPDATE (student_name, student_email) ON school_db.students TO
'counselor'@'localhost';

-- カラムレベル権限の確認
SHOW GRANTS FOR 'limited_viewer'@'localhost';
```

## REVOKE文の基本文法

### 1. 基本的なREVOKE文

```
-- 基本文法
REVOKE 権限 ON 対象 FROM ユーザー;

-- 単一権限の取り消し
REVOKE UPDATE ON school_db.grades FROM 'teacher1'@'localhost';

-- 複数権限の同時取り消し
REVOKE SELECT, INSERT ON school_db.students FROM 'temp_user'@'localhost';

-- 全権限の取り消し
REVOKE ALL PRIVILEGES ON school_db.* FROM 'old_admin'@'localhost';
```

### 2. 実践的な権限取り消し例

```
-- 退職した教師からすべての権限を取り消し
REVOKE ALL PRIVILEGES ON school_db.* FROM 'former_teacher'@'localhost';

-- 臨時職員から特定権限のみ取り消し
REVOKE INSERT, UPDATE, DELETE ON school_db.students FROM 'temp_staff'@'localhost';

-- 学生から一時的に閲覧権限を停止
REVOKE SELECT ON school_db.grades FROM 'suspended_student'@'localhost';

-- WITH GRANT OPTIONの取り消し
REVOKE GRANT OPTION ON school_db.* FROM 'supervisor'@'localhost';
```

## 実践的な権限管理シナリオ

### シナリオ1：新学期の教師権限設定

```
-- 新学期開始時の権限設定例

-- 1. 新任教師の基本権限設定
CREATE USER 'teacher_yamamoto'@'localhost' IDENTIFIED BY 'NewTeacher2025!';

-- 基本的な閲覧権限
GRANT SELECT ON school_db.students TO 'teacher_yamamoto'@'localhost';
GRANT SELECT ON school_db.courses TO 'teacher_yamamoto'@'localhost';
GRANT SELECT ON school_db.classrooms TO 'teacher_yamamoto'@'localhost';
GRANT SELECT ON school_db.class_periods TO 'teacher_yamamoto'@'localhost';

-- 担当講座の管理権限
GRANT SELECT, INSERT, UPDATE ON school_db.grades TO
'teacher_yamamoto'@'localhost';
GRANT SELECT, INSERT, UPDATE ON school_db.attendance TO
'teacher_yamamoto'@'localhost';
GRANT SELECT, INSERT, UPDATE ON school_db.course_schedule TO
'teacher_yamamoto'@'localhost';

-- 2. 昇進した教師（学年主任）への追加権限
-- 従来の権限に加えて、学年全体の管理権限を追加
GRANT UPDATE ON school_db.student_courses TO 'teacher_promoted'@'localhost';
GRANT SELECT ON school_db.teacher_unavailability TO
'teacher_promoted'@'localhost';

-- 3. 兼任講師（パートタイム）の制限付き権限
CREATE USER 'parttime_lecturer'@'localhost' IDENTIFIED BY 'PartTime2025!';

-- 基本閲覧権限（制限付き）
GRANT SELECT ON school_db.students TO 'parttime_lecturer'@'localhost';
GRANT SELECT ON school_db.courses TO 'parttime_lecturer'@'localhost';

-- 担当講座のみの成績管理（UPDATEは制限）
GRANT SELECT, INSERT ON school_db.grades TO 'parttime_lecturer'@'localhost';
GRANT SELECT, INSERT ON school_db.attendance TO 'parttime_lecturer'@'localhost';

-- 接続制限も設定
ALTER USER 'parttime_lecturer'@'localhost'
WITH MAX_USER_CONNECTIONS 2
     MAX_CONNECTIONS_PER_HOUR 50;

-- 設定した権限の確認
SHOW GRANTS FOR 'teacher_yamamoto'@'localhost';
SHOW GRANTS FOR 'teacher_promoted'@'localhost';
SHOW GRANTS FOR 'parttime_lecturer'@'localhost';
```

### シナリオ2：学期末の権限調整

```
-- 学期末の権限調整例

-- 1. 臨時職員の権限期限管理
-- 契約終了に伴う権限段階的削除

-- まず、データ変更権限を取り消し（閲覧は残す）
REVOKE INSERT, UPDATE, DELETE ON school_db.* FROM 'temp_assistant'@'localhost';

-- 一週間後に全権限を取り消し予定のため、制限を強化
ALTER USER 'temp_assistant'@'localhost'
WITH MAX_CONNECTIONS_PER_HOUR 10
     MAX_USER_CONNECTIONS 1;

-- 2. 休職中の教師の権限一時停止
-- アカウントを無効化せずに権限のみ停止
REVOKE ALL PRIVILEGES ON school_db.* FROM 'teacher_on_leave'@'localhost';

-- 復帰予定があるため、ユーザーアカウントは維持
-- ALTER USER 'teacher_on_leave'@'localhost' ACCOUNT LOCK; -- 必要に応じて

-- 3. 成績確定後の権限制限
-- 成績変更期間終了後、教師から成績UPDATE権限を一時的に取り消し
REVOKE UPDATE ON school_db.grades FROM 'teacher_sato'@'localhost';
REVOKE UPDATE ON school_db.grades FROM 'teacher_tanaka'@'localhost';

-- 緊急時の成績修正が必要な場合に備え、管理者権限は維持
-- 必要時に再付与可能な状態を保持

-- 権限変更ログの記録
INSERT INTO user_management_log (action, username, host, user_type, created_by,
notes)
VALUES
    ('MODIFY', 'temp_assistant', 'localhost', 'temp_staff', USER(), 'Contract
ending - reduced privileges'),
    ('MODIFY', 'teacher_on_leave', 'localhost', 'teacher', USER(), 'Medical leave
- suspended privileges'),
    ('MODIFY', 'teacher_sato', 'localhost', 'teacher', USER(), 'Grade finalization
- UPDATE revoked'),
    ('MODIFY', 'teacher_tanaka', 'localhost', 'teacher', USER(), 'Grade
finalization - UPDATE revoked');
```

### シナリオ3：監査・検査対応の権限管理

```
-- 外部監査対応の特別権限設定

-- 1. 監査法人用の読み取り専用アカウント
CREATE USER 'audit_firm'@'192.168.100.%'
IDENTIFIED BY 'AuditSecure2025!@#'
PASSWORD EXPIRE INTERVAL 30 DAY;
```

```
-- 必要なデータのみ閲覧権限を付与
GRANT SELECT ON school_db.students TO 'audit_firm'@'192.168.100.%';
GRANT SELECT ON school_db.grades TO 'audit_firm'@'192.168.100.%';
GRANT SELECT ON school_db.courses TO 'audit_firm'@'192.168.100.%';
GRANT SELECT ON school_db.teachers TO 'audit_firm'@'192.168.100.%';

-- システムログの閲覧権限（監査用）
GRANT SELECT ON mysql.general_log TO 'audit_firm'@'192.168.100.%';
GRANT SELECT ON information_schema.user_privileges TO
'audit_firm'@'192.168.100.%';

-- セッション数を制限
ALTER USER 'audit_firm'@'192.168.100.%'
WITH MAX_USER_CONNECTIONS 3
     MAX_CONNECTIONS_PER_HOUR 100;

-- 2. 内部監査用の拡張権限アカウント
CREATE USER 'internal_auditor'@'localhost'
IDENTIFIED BY 'InternalAudit2025!'
PASSWORD EXPIRE INTERVAL 60 DAY;

-- より広範囲なアクセス権限
GRANT SELECT ON school_db.* TO 'internal_auditor'@'localhost';
GRANT SELECT ON mysql.user TO 'internal_auditor'@'localhost';
GRANT SELECT ON performance_schema.accounts TO 'internal_auditor'@'localhost';

-- ユーザー管理ログの閲覧権限
GRANT SELECT ON school_db.user_management_log TO 'internal_auditor'@'localhost';

-- 3. 監査期間終了後の権限取り消し自動化
DELIMITER //

CREATE PROCEDURE revoke_audit_access()
BEGIN
    DECLARE audit_end_date DATE DEFAULT '2025-12-31';

    IF CURRENT_DATE > audit_end_date THEN
        -- 外部監査法人の権限を全て取り消し
        REVOKE ALL PRIVILEGES ON school_db.* FROM 'audit_firm'@'192.168.100.%';
        REVOKE ALL PRIVILEGES ON mysql.general_log FROM
'audit_firm'@'192.168.100.%';
        REVOKE ALL PRIVILEGES ON information_schema.user_privileges FROM
'audit_firm'@'192.168.100.%';

        -- アカウントをロック
        ALTER USER 'audit_firm'@'192.168.100.%' ACCOUNT LOCK;

        -- ログに記録
        INSERT INTO user_management_log (action, username, host, user_type,
created_by, notes)
        VALUES ('REVOKE', 'audit_firm', '192.168.100.%', 'auditor', 'SYSTEM',
'Audit period ended - all privileges revoked');

        SELECT 'Audit access revoked successfully' as result;
    ELSE
        SELECT CONCAT('Audit period active until ', audit_end_date) as result;
    END IF;
END //

DELIMITER ;

-- 権限確認
SHOW GRANTS FOR 'audit_firm'@'192.168.100.%';
SHOW GRANTS FOR 'internal_auditor'@'localhost';
```

## WITH GRANT OPTIONの活用

### 1. 権限委譲の仕組み

```
-- WITH GRANT OPTIONの基本使用法
GRANT SELECT, INSERT, UPDATE ON school_db.students TO
'department_head'@'localhost' WITH GRANT OPTION;

-- 部門長が他のスタッフに権限を委譲
-- (department_headとしてログインした状態で実行)
-- GRANT SELECT ON school_db.students TO 'staff_member'@'localhost';

-- 2. 階層的な権限管理
-- 校長 → 教務主任 → 学年主任 → 一般教師

-- 校長への全権限付与（権限委譲可能）
GRANT ALL PRIVILEGES ON school_db.* TO 'principal'@'localhost' WITH GRANT OPTION;

-- 教務主任への成績管理権限付与（部下への権限委譲可能）
GRANT SELECT, INSERT, UPDATE ON school_db.grades TO
'academic_director'@'localhost' WITH GRANT OPTION;
GRANT SELECT, INSERT, UPDATE ON school_db.attendance TO
'academic_director'@'localhost' WITH GRANT OPTION;

-- 学年主任への制限付き委譲権限
GRANT SELECT ON school_db.students TO 'grade_supervisor'@'localhost' WITH GRANT
OPTION;
GRANT SELECT, INSERT, UPDATE ON school_db.grades TO
'grade_supervisor'@'localhost';
-- 注意：grade_supervisorはgradesの権限委譲はできない

-- 3. 委譲権限の管理と制限
CREATE VIEW v_grant_delegation AS
SELECT
    grantor,
    grantee,
    table_schema,
    table_name,
    privilege_type,
    is_grantable
```

```
FROM information_schema.table_privileges
WHERE is_grantable = 'YES'
AND table_schema = 'school_db'
ORDER BY grantor, grantee;

-- 委譲状況の確認
SELECT * FROM v_grant_delegation;
```

## 2. GRANT OPTIONの取り消し

```
-- 特定の権限委譲能力のみ取り消し
REVOKE GRANT OPTION ON school_db.students FROM 'department_head'@'localhost';

-- 全ての権限委譲能力を取り消し
REVOKE ALL PRIVILEGES ON school_db.* FROM 'former_supervisor'@'localhost';

-- 委譲された権限の連鎖取り消し
-- 権限委譲者の権限を取り消すと、委譲された権限も自動的に取り消される場合がる
-- ただし、複数の経路で同じ権限が付与されている場合は残る

-- 安全な権限委譲取り消し手順
DELIMITER //

CREATE PROCEDURE safe_revoke_delegation(
    IN p_username VARCHAR(50),
    IN p_host VARCHAR(60)
)
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE v_grantee VARCHAR(100);

    -- 該当ユーザーが委譲した権限を受けているユーザーを取得
    DECLARE delegation_cursor CURSOR FOR
        SELECT DISTINCT grantee
        FROM information_schema.table_privileges
        WHERE grantor = CONCAT('''', p_username, '''@''', p_host, '''')
        AND is_grantable = 'YES'
        AND table_schema = 'school_db';

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    -- 委譲関係をログに記録
    INSERT INTO user_management_log (action, username, host, user_type,
created_by, notes)
    VALUES ('AUDIT', p_username, p_host, 'delegation', USER(),
            'Before revoking delegation - recording current state');

    OPEN delegation_cursor;

    read_loop: LOOP
        FETCH delegation_cursor INTO v_grantee;
        IF done THEN
            LEAVE read_loop;
        END IF;

        -- 委譲関係をログに記録
        INSERT INTO user_management_log (action, username, host, user_type,
created_by, notes)
        VALUES ('AUDIT', p_username, p_host, 'delegation', USER(),
                CONCAT('Had delegated privileges to: ', v_grantee));
    END LOOP;

    CLOSE delegation_cursor;

    -- 実際の権限取り消し
    SET @sql = CONCAT('REVOKE ALL PRIVILEGES ON school_db.* FROM ''', p_username,
'''@''', p_host, '''');
    PREPARE stmt FROM @sql;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;

    -- 最終ログ
    INSERT INTO user_management_log (action, username, host, user_type,
created_by, notes)
    VALUES ('REVOKE', p_username, p_host, 'delegation', USER(), 'All privileges
and delegations revoked');

END //

DELIMITER ;

-- 使用例（コメントアウト）
-- CALL safe_revoke_delegation('department_head', 'localhost');
```

## 権限の確認と監査

### 1. 権限状況の確認方法

```
-- 1. 基本的な権限確認
SHOW GRANTS FOR 'teacher_tanaka'@'localhost';
SHOW GRANTS FOR CURRENT_USER();

-- 2. システム全体の権限状況確認
SELECT
    grantee,
    table_schema,
    table_name,
    privilege_type,
    is_grantable
FROM information_schema.table_privileges
WHERE table_schema = 'school_db'
ORDER BY grantee, table_name, privilege_type;
```

```sql
-- 3. ユーザー別権限サマリー
SELECT
    grantee,
    table_schema,
    COUNT(DISTINCT table_name) as accessible_tables,
    GROUP_CONCAT(DISTINCT privilege_type ORDER BY privilege_type) as privileges
FROM information_schema.table_privileges
WHERE table_schema = 'school_db'
GROUP BY grantee, table_schema
ORDER BY grantee;

-- 4. 危険な権限を持つユーザーの特定
SELECT DISTINCT
    grantee,
    privilege_type
FROM information_schema.user_privileges
WHERE privilege_type IN ('SUPER', 'CREATE USER', 'GRANT OPTION', 'FILE',
'PROCESS', 'RELOAD')
ORDER BY grantee, privilege_type;
```

### 2. 権限監査用ビューの作成

```sql
-- 包括的な権限監査ビュー
CREATE VIEW v_comprehensive_privileges AS
SELECT
    'TABLE' as privilege_level,
    tp.grantee,
    tp.table_schema as db_name,
    tp.table_name as object_name,
    tp.privilege_type,
    tp.is_grantable,
    'N/A' as column_name
FROM information_schema.table_privileges tp
WHERE tp.table_schema NOT IN ('information_schema', 'mysql', 'performance_schema',
'sys')

UNION ALL

SELECT
    'COLUMN' as privilege_level,
    cp.grantee,
    cp.table_schema as db_name,
    cp.table_name as object_name,
    cp.privilege_type,
    cp.is_grantable,
    cp.column_name
FROM information_schema.column_privileges cp
WHERE cp.table_schema NOT IN ('information_schema', 'mysql', 'performance_schema',
'sys')
```

```sql
UNION ALL

SELECT
    'SCHEMA' as privilege_level,
    sp.grantee,
    sp.table_schema as db_name,
    'ALL_TABLES' as object_name,
    sp.privilege_type,
    sp.is_grantable,
    'N/A' as column_name
FROM information_schema.schema_privileges sp
WHERE sp.table_schema NOT IN ('information_schema', 'mysql', 'performance_schema', 
'sys')

UNION ALL

SELECT
    'GLOBAL' as privilege_level,
    up.grantee,
    'GLOBAL' as db_name,
    'GLOBAL' as object_name,
    up.privilege_type,
    up.is_grantable,
    'N/A' as column_name
FROM information_schema.user_privileges up

ORDER BY grantee, privilege_level, db_name, object_name;

-- 監査レポートの生成
SELECT * FROM v_comprehensive_privileges WHERE grantee LIKE '%teacher%';

-- 権限統計レポート
SELECT
    privilege_level,
    COUNT(DISTINCT grantee) as users_count,
    COUNT(*) as total_privileges,
    COUNT(CASE WHEN is_grantable = 'YES' THEN 1 END) as grantable_privileges
FROM v_comprehensive_privileges
WHERE db_name = 'school_db'
GROUP BY privilege_level
ORDER BY users_count DESC;
```

### 3. 権限変更履歴の管理

```sql
-- 権限変更履歴テーブル
CREATE TABLE privilege_change_log (
    log_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    change_type ENUM('GRANT', 'REVOKE') NOT NULL,
    grantee VARCHAR(100) NOT NULL,
    privilege_type VARCHAR(50) NOT NULL,
    object_type ENUM('GLOBAL', 'SCHEMA', 'TABLE', 'COLUMN') NOT NULL,
    object_name VARCHAR(200),
    column_name VARCHAR(64),
    is_grantable ENUM('YES', 'NO') DEFAULT 'NO',
    changed_by VARCHAR(100) NOT NULL,
    changed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    reason TEXT,

    INDEX idx_grantee (grantee),
    INDEX idx_change_date (changed_at),
    INDEX idx_change_type (change_type)
);

-- 権限変更記録プロシージャ
DELIMITER //

CREATE PROCEDURE log_privilege_change(
    IN p_change_type ENUM('GRANT', 'REVOKE'),
    IN p_grantee VARCHAR(100),
    IN p_privilege_type VARCHAR(50),
    IN p_object_type ENUM('GLOBAL', 'SCHEMA', 'TABLE', 'COLUMN'),
    IN p_object_name VARCHAR(200),
    IN p_column_name VARCHAR(64),
    IN p_is_grantable ENUM('YES', 'NO'),
    IN p_reason TEXT
)
BEGIN
    INSERT INTO privilege_change_log
        (change_type, grantee, privilege_type, object_type, object_name,
         column_name, is_grantable, changed_by, reason)
    VALUES
        (p_change_type, p_grantee, p_privilege_type, p_object_type, p_object_name,
         p_column_name, p_is_grantable, USER(), p_reason);
END //

DELIMITER ;

-- 使用例
-- CALL log_privilege_change('GRANT', 'teacher_yamamoto@localhost', 'SELECT',
'TABLE', 'school_db.students', NULL, 'NO', 'New teacher onboarding');
-- CALL log_privilege_change('REVOKE', 'former_teacher@localhost', 'UPDATE',
'TABLE', 'school_db.grades', NULL, 'NO', 'Staff departure');

-- 権限変更履歴レポート
SELECT
    change_type,
    grantee,
    privilege_type,
    object_name,
    changed_by,
    changed_at,
    reason
FROM privilege_change_log
WHERE changed_at >= DATE_SUB(NOW(), INTERVAL 30 DAY)
ORDER BY changed_at DESC;
```

## エラーと対処法

### 1. 権限不足エラー

```
-- エラー例1: 権限を付与する権限がない
-- GRANT SELECT ON school_db.students TO 'new_user'@'localhost';
-- エラー: ERROR 1044 (42000): Access denied for user 'regular_user'@'localhost' to
database 'school_db'

-- 対処法1: 適切な権限を持つユーザーで実行
-- 管理者としてログインして実行

-- 対処法2: WITH GRANT OPTIONを持つユーザーで実行
-- GRANT SELECT ON school_db.students TO 'department_head'@'localhost' WITH GRANT
OPTION;

-- 現在のユーザーの権限確認
SHOW GRANTS FOR CURRENT_USER();

-- 権限付与に必要な権限を確認
SELECT
    grantee,
    table_schema,
    privilege_type,
    is_grantable
FROM information_schema.table_privileges
WHERE grantee = CONCAT('''', USER(), '''')
AND table_schema = 'school_db'
AND is_grantable = 'YES';
```

### 2. 存在しないユーザーやテーブルへの権限付与

```
-- エラー例2: 存在しないユーザーへの権限付与
-- GRANT SELECT ON school_db.students TO 'nonexistent_user'@'localhost';
-- エラー: ERROR 1133 (42000): Can't find any matching row in the user table

-- 対処法: ユーザーの存在確認と作成
SELECT COUNT(*) as user_exists
FROM mysql.user
WHERE User = 'nonexistent_user' AND Host = 'localhost';

-- ユーザーが存在しない場合は作成
CREATE USER IF NOT EXISTS 'nonexistent_user'@'localhost' IDENTIFIED BY
'SecurePassword2025!';
GRANT SELECT ON school_db.students TO 'nonexistent_user'@'localhost';

-- エラー例3: 存在しないテーブルへの権限付与
-- GRANT SELECT ON school_db.nonexistent_table TO 'user'@'localhost';
```

```sql
-- エラー: ERROR 1146 (42S02): Table 'school_db.nonexistent_table' doesn't exist

-- 対処法: テーブルの存在確認
SELECT COUNT(*) as table_exists
FROM information_schema.tables
WHERE table_schema = 'school_db' AND table_name = 'nonexistent_table';

-- テーブル一覧の確認
SHOW TABLES FROM school_db;
```

3. 権限取り消し時のエラー

```sql
-- エラー例4: 存在しない権限の取り消し
-- REVOKE UPDATE ON school_db.students FROM 'user'@'localhost';
-- エラー: ERROR 1147 (42000): There is no such grant defined for user 'user' on
host 'localhost'

-- 対処法: 現在の権限を確認してから取り消し
SHOW GRANTS FOR 'user'@'localhost';

-- 安全な権限取り消しプロシージャ
DELIMITER //

CREATE PROCEDURE safe_revoke_privilege(
    IN p_privilege VARCHAR(50),
    IN p_object VARCHAR(200),
    IN p_grantee VARCHAR(100)
)
BEGIN
    DECLARE privilege_exists INT DEFAULT 0;

    -- 権限の存在確認（テーブルレベル権限の場合）
    SELECT COUNT(*) INTO privilege_exists
    FROM information_schema.table_privileges
    WHERE CONCAT(grantee) = CONCAT('''', p_grantee, '''')
    AND CONCAT(table_schema, '.', table_name) = p_object
    AND privilege_type = p_privilege;

    IF privilege_exists > 0 THEN
        SET @sql = CONCAT('REVOKE ', p_privilege, ' ON ', p_object, ' FROM ',
p_grantee);
        PREPARE stmt FROM @sql;
        EXECUTE stmt;
        DEALLOCATE PREPARE stmt;

        SELECT CONCAT('Successfully revoked ', p_privilege, ' on ', p_object, '
from ', p_grantee) as result;
    ELSE
        SELECT CONCAT('Privilege ', p_privilege, ' on ', p_object, ' does not
exist for ', p_grantee) as result;
    END IF;
END //

DELIMITER ;

-- 使用例
-- CALL safe_revoke_privilege('UPDATE', 'school_db.students', 'user@localhost');
```

4. 循環的な権限委譲

```
-- 権限委譲の循環参照を防ぐチェック関数
DELIMITER //

CREATE FUNCTION check_delegation_cycle(
    p_grantor VARCHAR(100),
    p_grantee VARCHAR(100)
) RETURNS BOOLEAN
READS SQL DATA
DETERMINISTIC
BEGIN
    DECLARE cycle_exists BOOLEAN DEFAULT FALSE;

    -- 簡単な循環チェック: 付与先が付与元に権限を委譲しているかチェック
    SELECT COUNT(*) > 0 INTO cycle_exists
    FROM information_schema.table_privileges
    WHERE grantee = p_grantor
    AND grantor = p_grantee
    AND is_grantable = 'YES';

    RETURN cycle_exists;
END //

DELIMITER ;

-- 安全な権限委譲プロシージャ
DELIMITER //

CREATE PROCEDURE safe_grant_with_option(
    IN p_privilege VARCHAR(50),
    IN p_object VARCHAR(200),
    IN p_grantee VARCHAR(100)
)
BEGIN
    DECLARE current_user_str VARCHAR(100);
    DECLARE cycle_risk BOOLEAN DEFAULT FALSE;

    SET current_user_str = CONCAT('''', SUBSTRING_INDEX(USER(), '@', 1), '''@''',
SUBSTRING_INDEX(USER(), '@', -1), '''');

    -- 循環参照チェック
    SELECT check_delegation_cycle(current_user_str, p_grantee) INTO cycle_risk;
    IF cycle_risk THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Delegation cycle detected - operation cancelled';
    ELSE
        SET @sql = CONCAT('GRANT ', p_privilege, ' ON ', p_object, ' TO ', p_grantee, ' WITH GRANT OPTION');
        PREPARE stmt FROM @sql;
        EXECUTE stmt;
        DEALLOCATE PREPARE stmt;

        SELECT CONCAT('Successfully granted ', p_privilege, ' with delegation rights') as result;
    END IF;
END //

DELIMITER ;
```

## 高度な権限管理テクニック

### 1. 条件付き権限管理

```
-- 時間制限付き権限管理
CREATE TABLE temporary_privileges (
    temp_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    grantee VARCHAR(100) NOT NULL,
    privilege_type VARCHAR(50) NOT NULL,
    object_name VARCHAR(200) NOT NULL,
    granted_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    expires_at TIMESTAMP NOT NULL,
    is_active BOOLEAN DEFAULT TRUE,
    granted_by VARCHAR(100) NOT NULL,

    INDEX idx_expiry (expires_at, is_active),
    INDEX idx_grantee (grantee)
);

-- 一時的権限付与プロシージャ
DELIMITER //

CREATE PROCEDURE grant_temporary_privilege(
    IN p_privilege VARCHAR(50),
    IN p_object VARCHAR(200),
    IN p_grantee VARCHAR(100),
    IN p_duration_hours INT
)
BEGIN
    DECLARE expiry_time TIMESTAMP;

    SET expiry_time = DATE_ADD(NOW(), INTERVAL p_duration_hours HOUR);

    -- 実際に権限を付与
    SET @sql = CONCAT('GRANT ', p_privilege, ' ON ', p_object, ' TO ', p_grantee);
    PREPARE stmt FROM @sql;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;

    -- 一時権限テーブルに記録
    INSERT INTO temporary_privileges (grantee, privilege_type, object_name,
expires_at, granted_by)
    VALUES (p_grantee, p_privilege, p_object, expiry_time, USER());

    SELECT CONCAT('Temporary privilege granted until ', expiry_time) as result;
END //

-- 期限切れ権限の自動取り消し
CREATE PROCEDURE revoke_expired_privileges()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE v_grantee VARCHAR(100);
    DECLARE v_privilege VARCHAR(50);
    DECLARE v_object VARCHAR(200);
    DECLARE v_temp_id BIGINT;

    DECLARE expired_cursor CURSOR FOR
        SELECT temp_id, grantee, privilege_type, object_name
        FROM temporary_privileges
        WHERE expires_at <= NOW()
        AND is_active = TRUE;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN expired_cursor;

    read_loop: LOOP
        FETCH expired_cursor INTO v_temp_id, v_grantee, v_privilege, v_object;
        IF done THEN
            LEAVE read_loop;
        END IF;

        -- 権限を取り消し
        SET @sql = CONCAT('REVOKE ', v_privilege, ' ON ', v_object, ' FROM ',
v_grantee);
        PREPARE stmt FROM @sql;
        EXECUTE stmt;
        DEALLOCATE PREPARE stmt;

        -- 記録を無効化
        UPDATE temporary_privileges
        SET is_active = FALSE
        WHERE temp_id = v_temp_id;

    END LOOP;

    CLOSE expired_cursor;
    SELECT CONCAT('Revoked ', ROW_COUNT(), ' expired privileges') as result;
END //

DELIMITER ;

-- 使用例（コメントアウト）
-- CALL grant_temporary_privilege('SELECT', 'school_db.grades',
'temp_auditor@localhost', 24); -- 24時間限定
-- CALL revoke_expired_privileges(); -- 期限切れ権限の一括取り消し
```

## 2. 動的権限管理

```
-- 役職ベース動的権限管理
CREATE TABLE role_privilege_mapping (
    mapping_id INT AUTO_INCREMENT PRIMARY KEY,
    role_name VARCHAR(50) NOT NULL,
    privilege_type VARCHAR(50) NOT NULL,
    object_pattern VARCHAR(200) NOT NULL,
    condition_sql TEXT,
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    UNIQUE KEY unique_role_privilege (role_name, privilege_type, object_pattern)
);

-- 基本的な役職-権限マッピング
INSERT INTO role_privilege_mapping (role_name, privilege_type, object_pattern,
condition_sql) VALUES
('teacher', 'SELECT', 'school_db.students', 'WHERE teacher_id =
@current_teacher_id'),
('teacher', 'UPDATE', 'school_db.grades', 'WHERE course_id IN (SELECT course_id
FROM courses WHERE teacher_id = @current_teacher_id)'),
('student', 'SELECT', 'school_db.grades', 'WHERE student_id =
@current_student_id'),
('office_staff', 'UPDATE', 'school_db.students', 'WHERE student_id NOT IN (SELECT
student_id FROM restricted_students)');

-- 動的権限適用プロシージャ
DELIMITER //

CREATE PROCEDURE apply_role_based_privileges(
    IN p_username VARCHAR(50),
    IN p_host VARCHAR(60),
    IN p_role VARCHAR(50)
)
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE v_privilege VARCHAR(50);
    DECLARE v_object VARCHAR(200);

    DECLARE privilege_cursor CURSOR FOR
```

```
        SELECT privilege_type, object_pattern
        FROM role_privilege_mapping
        WHERE role_name = p_role
        AND is_active = TRUE;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN privilege_cursor;

    read_loop: LOOP
        FETCH privilege_cursor INTO v_privilege, v_object;
        IF done THEN
            LEAVE read_loop;
        END IF;

        -- 権限を付与
        SET @sql = CONCAT('GRANT ', v_privilege, ' ON ', v_object, ' TO ''',
p_username, '''@''', p_host, '''');
        PREPARE stmt FROM @sql;
        EXECUTE stmt;
        DEALLOCATE PREPARE stmt;

    END LOOP;

    CLOSE privilege_cursor;

    -- 適用ログ
    INSERT INTO user_management_log (action, username, host, user_type,
created_by, notes)
    VALUES ('GRANT', p_username, p_host, p_role, USER(),
            CONCAT('Applied role-based privileges for role: ', p_role));

END //

DELIMITER ;
```

## 練習問題

### 問題40-1：基本的な権限付与

学校システムに以下の3つの新しいユーザーを作成し、それぞれに適切な権限を付与してください：

1. `lab_assistant`：実験助手（ローカルからのみアクセス）
    - `students`テーブルの閲覧権限
    - `courses`テーブルの閲覧権限
    - `grades`テーブルの実験科目のみ入力・更新権限
2. `guest_lecturer`：外部講師（学内ネットワーク192.168.1.%からアクセス）
    - `students`テーブルの名前と学籍番号のみ閲覧権限
    - `attendance`テーブルの入力権限
3. `data_analyst`：データ分析専門職（どこからでもアクセス可能）
    - 全テーブルの閲覧権限のみ

- ただし個人情報関連カラムは除外

各ユーザーの権限付与後、`SHOW GRANTS`で確認してください。

### 問題40-2：段階的権限管理

以下のシナリオに沿って権限の付与・取り消しを実行してください：

1. `temp_teacher`（臨時教師）を作成し、基本的な閲覧権限のみ付与
2. 正式採用決定により、成績入力権限を追加付与
3. 担当クラス変更により、出席管理権限も追加付与
4. 契約期間終了により、データ変更権限をすべて取り消し（閲覧は維持）
5. 最終退職により、すべての権限を取り消し

各段階で権限の状況を確認し、変更前後の権限の違いを記録してください。

### 問題40-3：WITH GRANT OPTIONの活用

以下の階層的権限管理システムを構築してください：

1. `department_head`（学科長）に以下の権限を`WITH GRANT OPTION`で付与：
   - `teachers`テーブルの管理権限
   - `courses`テーブルの管理権限
   - `grades`テーブルの閲覧・更新権限
2. `department_head`として、部下の`senior_teacher`に権限を委譲：
   - `grades`テーブルの閲覧・更新権限（委譲権限なし）
   - `courses`テーブルの閲覧権限
3. 権限委譲の確認と、`department_head`の権限取り消しが`senior_teacher`に与える影響を検証

### 問題40-4：カラムレベル権限制御

個人情報保護を考慮した細かい権限制御を実装してください：

1. `privacy_officer`ユーザーを作成
2. `students`テーブルの以下のカラムのみアクセス権限を付与：
   - 閲覧権限：`student_id`, `student_name`
   - 更新権限：`student_email`（メールアドレス変更対応）
3. `research_assistant`ユーザーを作成
4. `grades`テーブルの以下のカラムのみアクセス権限を付与：
   - 閲覧権限：`course_id`, `grade_type`, `score`（個人特定情報は除外）
5. 設定した権限で実際にデータアクセスを試し、制限が正しく働くことを確認

### 問題40-5：監査対応権限管理

外部監査に対応するための一時的権限管理システムを作成してください：

1. 監査期間（30日間）限定のアクセス権限設定：
   - `external_auditor`ユーザーの作成（特定IPからのみアクセス）
   - 全テーブルの閲覧権限（個人情報は制限）
   - システムログの閲覧権限
2. 内部監査人`internal_auditor`向けの権限設定：

    - より広範囲なデータアクセス権限
    - ユーザー管理ログの閲覧権限
    - 権限変更履歴の閲覧権限
3. 監査終了時の権限自動取り消し機能の実装
4. 監査期間中の権限使用状況の記録・レポート機能

### 問題40-6：総合的な権限管理システム

学校システム全体の包括的な権限管理システムを設計・実装してください：

1. 以下の役職に対応した権限体系の構築：
    - 校長：全権限
    - 教務主任：成績・授業管理権限
    - 学年主任：担当学年の管理権限
    - 一般教師：担当クラス管理権限
    - 事務職員：学生情報管理権限
    - 保健担当：健康関連情報管理権限
    - 図書館司書：図書館システム管理権限
2. 動的権限管理機能：
    - 役職変更時の自動権限更新
    - 期間限定権限の自動管理
    - 異常なアクセスパターンの検知
3. 監査機能：
    - 全権限変更の履歴管理
    - 定期的な権限レビューレポート
    - セキュリティリスクの自動検出
4. 運用管理機能：
    - 権限付与・取り消しの承認ワークフロー
    - 一括権限管理機能
    - 緊急時の権限停止機能

## 解答

### 解答40-1

```sql
-- 1. lab_assistant（実験助手）の作成と権限付与
CREATE USER 'lab_assistant'@'localhost'
IDENTIFIED BY 'LabAssistant2025!'
PASSWORD EXPIRE INTERVAL 120 DAY;

-- 基本閲覧権限
GRANT SELECT ON school_db.students TO 'lab_assistant'@'localhost';
GRANT SELECT ON school_db.courses TO 'lab_assistant'@'localhost';

-- 実験科目の成績管理権限（実際の運用では条件付きビューを使用することが多い）
GRANT SELECT, INSERT, UPDATE ON school_db.grades TO 'lab_assistant'@'localhost';

-- 2. guest_lecturer（外部講師）の作成と権限付与
CREATE USER 'guest_lecturer'@'192.168.1.%'
```

```sql
IDENTIFIED BY 'GuestLecturer2025!'
PASSWORD EXPIRE INTERVAL 90 DAY;

-- 学生の名前と学籍番号のみ閲覧権限（カラムレベル権限）
GRANT SELECT (student_id, student_name) ON school_db.students TO
'guest_lecturer'@'192.168.1.%';

-- 出席入力権限
GRANT SELECT, INSERT ON school_db.attendance TO 'guest_lecturer'@'192.168.1.%';

-- 3. data_analyst（データ分析専門職）の作成と権限付与
CREATE USER 'data_analyst'@'%'
IDENTIFIED BY 'DataAnalyst2025!'
PASSWORD EXPIRE INTERVAL 180 DAY;

-- 全テーブル閲覧権限（個人情報関連カラムは除外）
GRANT SELECT ON school_db.courses TO 'data_analyst'@'%';
GRANT SELECT ON school_db.classrooms TO 'data_analyst'@'%';
GRANT SELECT ON school_db.class_periods TO 'data_analyst'@'%';
GRANT SELECT ON school_db.terms TO 'data_analyst'@'%';
GRANT SELECT ON school_db.course_schedule TO 'data_analyst'@'%';

-- 学生テーブルは個人情報を除外したカラムのみ
GRANT SELECT (student_id) ON school_db.students TO 'data_analyst'@'%';

-- 成績テーブルは統計分析用カラムのみ
GRANT SELECT (course_id, grade_type, score, max_score, submission_date) ON
school_db.grades TO 'data_analyst'@'%';

-- 出席テーブルも個人特定情報を除外
GRANT SELECT (schedule_id, status) ON school_db.attendance TO 'data_analyst'@'%';

-- 接続制限の設定
ALTER USER 'data_analyst'@'%'
WITH MAX_USER_CONNECTIONS 3
     MAX_CONNECTIONS_PER_HOUR 100;

-- 権限確認
SHOW GRANTS FOR 'lab_assistant'@'localhost';
SHOW GRANTS FOR 'guest_lecturer'@'192.168.1.%';
SHOW GRANTS FOR 'data_analyst'@'%';

-- 作成したユーザーの一覧確認
SELECT
    User as username,
    Host as allowed_host,
    account_locked,
    password_expired,
    max_user_connections
FROM mysql.user
WHERE User IN ('lab_assistant', 'guest_lecturer', 'data_analyst')
ORDER BY User;
```

## 解答40-2

```sql
-- 段階的権限管理シナリオ

-- 1. temp_teacher（臨時教師）の作成と基本権限付与
CREATE USER 'temp_teacher'@'localhost'
IDENTIFIED BY 'TempTeacher2025!'
PASSWORD EXPIRE INTERVAL 90 DAY;

-- 基本閲覧権限のみ
GRANT SELECT ON school_db.students TO 'temp_teacher'@'localhost';
GRANT SELECT ON school_db.courses TO 'temp_teacher'@'localhost';
GRANT SELECT ON school_db.classrooms TO 'temp_teacher'@'localhost';

-- 初期権限の確認
SELECT 'Step 1: Basic privileges only' as stage;
SHOW GRANTS FOR 'temp_teacher'@'localhost';

-- 2. 正式採用決定により成績入力権限を追加
GRANT SELECT, INSERT, UPDATE ON school_db.grades TO 'temp_teacher'@'localhost';

-- 権限追加後の確認
SELECT 'Step 2: Added grade management privileges' as stage;
SHOW GRANTS FOR 'temp_teacher'@'localhost';

-- 3. 担当クラス変更により出席管理権限も追加
GRANT SELECT, INSERT, UPDATE ON school_db.attendance TO
'temp_teacher'@'localhost';
GRANT SELECT, INSERT, UPDATE ON school_db.course_schedule TO
'temp_teacher'@'localhost';

-- 権限追加後の確認
SELECT 'Step 3: Added attendance management privileges' as stage;
SHOW GRANTS FOR 'temp_teacher'@'localhost';

-- 4. 契約期間終了により、データ変更権限をすべて取り消し（閲覧は維持）
REVOKE INSERT, UPDATE ON school_db.grades FROM 'temp_teacher'@'localhost';
REVOKE INSERT, UPDATE ON school_db.attendance FROM 'temp_teacher'@'localhost';
REVOKE INSERT, UPDATE ON school_db.course_schedule FROM
'temp_teacher'@'localhost';

-- 権限取り消し後の確認
SELECT 'Step 4: Revoked modification privileges, kept read access' as stage;
SHOW GRANTS FOR 'temp_teacher'@'localhost';

-- 5. 最終退職により、すべての権限を取り消し
REVOKE ALL PRIVILEGES ON school_db.* FROM 'temp_teacher'@'localhost';

-- 最終状態の確認
SELECT 'Step 5: All privileges revoked' as stage;
SHOW GRANTS FOR 'temp_teacher'@'localhost';
```

```
-- 各段階の記録用テーブル作成とログ
CREATE TABLE IF NOT EXISTS privilege_change_demo_log (
    step_number INT,
    stage_description VARCHAR(200),
    privileges_granted TEXT,
    privileges_revoked TEXT,
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- 段階的変更の記録
INSERT INTO privilege_change_demo_log (step_number, stage_description,
privileges_granted) VALUES
(1, 'Basic privileges only', 'SELECT on students, courses, classrooms'),
(2, 'Added grade management', 'SELECT, INSERT, UPDATE on grades'),
(3, 'Added attendance management', 'SELECT, INSERT, UPDATE on attendance,
course_schedule'),
(4, 'Contract ending - revoked modifications', NULL),
(5, 'Final departure - all revoked', NULL);

UPDATE privilege_change_demo_log SET privileges_revoked = 'INSERT, UPDATE on
grades, attendance, course_schedule' WHERE step_number = 4;
UPDATE privilege_change_demo_log SET privileges_revoked = 'ALL PRIVILEGES' WHERE
step_number = 5;

-- ログの確認
SELECT * FROM privilege_change_demo_log ORDER BY step_number;
```

### 解答40-3

```
-- WITH GRANT OPTIONの活用例

-- 1. department_head（学科長）の作成とWITH GRANT OPTION権限付与
CREATE USER 'department_head'@'localhost'
IDENTIFIED BY 'DeptHead2025!'
PASSWORD EXPIRE INTERVAL 60 DAY;

-- 権限委譲可能な権限を付与
GRANT SELECT, INSERT, UPDATE, DELETE ON school_db.teachers TO
'department_head'@'localhost' WITH GRANT OPTION;
GRANT SELECT, INSERT, UPDATE, DELETE ON school_db.courses TO
'department_head'@'localhost' WITH GRANT OPTION;
GRANT SELECT, UPDATE ON school_db.grades TO 'department_head'@'localhost' WITH
GRANT OPTION;

-- 学科長の権限確認
SELECT 'Department Head privileges with GRANT OPTION:' as status;
SHOW GRANTS FOR 'department_head'@'localhost';

-- 2. senior_teacher（上級教師）の作成
CREATE USER 'senior_teacher'@'localhost'
IDENTIFIED BY 'SeniorTeacher2025!'
```

```
PASSWORD EXPIRE INTERVAL 90 DAY;

-- 3. department_headとして権限を委譲（実際の運用では学科長としてログインして実行）
-- ここでは管理者として代理実行し、委譲のシミュレーションを行う

-- senior_teacherに権限委譲（委譲権限なし）
GRANT SELECT, UPDATE ON school_db.grades TO 'senior_teacher'@'localhost';
GRANT SELECT ON school_db.courses TO 'senior_teacher'@'localhost';

-- 委譲記録用のログ
INSERT INTO user_management_log (action, username, host, user_type, created_by,
notes)
VALUES ('GRANT', 'senior_teacher', 'localhost', 'teacher',
'department_head@localhost',
        'Delegated by department head: grades SELECT/UPDATE, courses SELECT');

-- 委譲後の権限確認
SELECT 'Senior Teacher privileges (delegated):' as status;
SHOW GRANTS FOR 'senior_teacher'@'localhost';

-- 4. 権限委譲状況の確認
SELECT
    grantee,
    table_schema,
    table_name,
    privilege_type,
    is_grantable,
    'Current delegation status' as note
FROM information_schema.table_privileges
WHERE grantee IN ('''department_head''@''localhost''',
'''senior_teacher''@''localhost''')
AND table_schema = 'school_db'
ORDER BY grantee, table_name, privilege_type;

-- 5. department_headの権限取り消しによる影響検証

-- まず現在の委譲状況を記録
CREATE TEMPORARY TABLE delegation_before AS
SELECT
    grantee,
    table_schema,
    table_name,
    privilege_type,
    is_grantable
FROM information_schema.table_privileges
WHERE table_schema = 'school_db'
AND (grantee LIKE '%department_head%' OR grantee LIKE '%senior_teacher%');

SELECT 'Before revoking department_head privileges:' as status;
SELECT * FROM delegation_before;

-- department_headの権限を取り消し
REVOKE ALL PRIVILEGES ON school_db.* FROM 'department_head'@'localhost';
```

```
-- 取り消し後の状況確認
SELECT 'After revoking department_head privileges:' as status;
SELECT
    grantee,
    table_schema,
    table_name,
    privilege_type,
    is_grantable,
    'After revocation' as note
FROM information_schema.table_privileges
WHERE table_schema = 'school_db'
AND (grantee LIKE '%department_head%' OR grantee LIKE '%senior_teacher%')
ORDER BY grantee, table_name, privilege_type;

-- senior_teacherの権限が残っているか確認
SHOW GRANTS FOR 'senior_teacher'@'localhost';

-- 影響分析レポート
SELECT
    'GRANT OPTION Impact Analysis' as report_type,
    CASE
        WHEN EXISTS (
            SELECT 1 FROM information_schema.table_privileges
            WHERE grantee = '''senior_teacher''@''localhost'''
            AND table_schema = 'school_db'
        ) THEN 'senior_teacher privileges RETAINED (direct grant or other
grantor)'
        ELSE 'senior_teacher privileges REVOKED (dependent on department_head)'
    END as impact_result;
```

### 解答40-4

```
-- カラムレベル権限制御の実装

-- 1. privacy_officer（個人情報保護担当）ユーザーの作成
CREATE USER 'privacy_officer'@'localhost'
IDENTIFIED BY 'PrivacyOfficer2025!'
PASSWORD EXPIRE INTERVAL 90 DAY;

-- studentsテーブルの制限付きアクセス権限
GRANT SELECT (student_id, student_name) ON school_db.students TO
'privacy_officer'@'localhost';
GRANT UPDATE (student_email) ON school_db.students TO
'privacy_officer'@'localhost';

-- 2. research_assistant（研究助手）ユーザーの作成
CREATE USER 'research_assistant'@'localhost'
IDENTIFIED BY 'ResearchAssist2025!'
PASSWORD EXPIRE INTERVAL 120 DAY;

-- gradesテーブルの統計分析用アクセス権限（個人特定情報は除外）
```

```
GRANT SELECT (course_id, grade_type, score, max_score, submission_date) ON
school_db.grades TO 'research_assistant'@'localhost';

-- 統計分析に必要な関連テーブルの権限
GRANT SELECT ON school_db.courses TO 'research_assistant'@'localhost';
GRANT SELECT ON school_db.terms TO 'research_assistant'@'localhost';

-- 3. 設定した権限の確認
SELECT 'Privacy Officer column-level privileges:' as user_type;
SHOW GRANTS FOR 'privacy_officer'@'localhost';

SELECT 'Research Assistant column-level privileges:' as user_type;
SHOW GRANTS FOR 'research_assistant'@'localhost';

-- 4. カラムレベル権限の詳細確認
SELECT
    grantee,
    table_schema,
    table_name,
    column_name,
    privilege_type,
    is_grantable
FROM information_schema.column_privileges
WHERE table_schema = 'school_db'
AND grantee IN ('''privacy_officer''@''localhost''',
'''research_assistant''@''localhost''')
ORDER BY grantee, table_name, column_name;

-- 5. 権限制限の動作確認用テストクエリ（実際には各ユーザーでログインして実行）

-- privacy_officerでアクセス可能なクエリ例
SELECT 'Test queries for privacy_officer (would work):' as test_type;
-- SELECT student_id, student_name FROM school_db.students LIMIT 5;
-- UPDATE school_db.students SET student_email = 'new@example.com' WHERE
student_id = 301;

-- privacy_officerで制限されるクエリ例
SELECT 'Test queries for privacy_officer (would FAIL):' as test_type;
-- SELECT * FROM school_db.students; -- エラー: アクセス権限のないカラムを含む
-- UPDATE school_db.students SET student_name = 'New Name' WHERE student_id = 301;
-- エラー: 更新権限なし

-- research_assistantでアクセス可能なクエリ例
SELECT 'Test queries for research_assistant (would work):' as test_type;
-- SELECT course_id, AVG(score) as avg_score FROM school_db.grades GROUP BY
course_id;
-- SELECT grade_type, COUNT(*) as count FROM school_db.grades GROUP BY grade_type;

-- research_assistantで制限されるクエリ例
SELECT 'Test queries for research_assistant (would FAIL):' as test_type;
-- SELECT student_id, score FROM school_db.grades; -- エラー: student_idカラムへのアクセ
ス権限なし
-- INSERT INTO school_db.grades (...) VALUES (...); -- エラー: INSERT権限なし
-- 6. カラムレベル権限の監視用ビュー
CREATE VIEW v_column_level_privileges AS
SELECT
    cp.grantee,
    cp.table_schema,
    cp.table_name,
    cp.column_name,
    cp.privilege_type,
    cp.is_grantable,
    'COLUMN' as privilege_level
FROM information_schema.column_privileges cp
WHERE cp.table_schema = 'school_db'

UNION ALL

SELECT
    tp.grantee,
    tp.table_schema,
    tp.table_name,
    'ALL_COLUMNS' as column_name,
    tp.privilege_type,
    tp.is_grantable,
    'TABLE' as privilege_level
FROM information_schema.table_privileges tp
WHERE tp.table_schema = 'school_db'

ORDER BY grantee, table_name, privilege_level, column_name;

-- カラムレベル権限の現状確認
SELECT * FROM v_column_level_privileges
WHERE grantee IN ('''privacy_officer''@''localhost''',
'''research_assistant''@''localhost''');

-- 7. セキュリティ検証レポート
SELECT
    'Column-Level Security Verification' as report_title,
    COUNT(DISTINCT grantee) as users_with_column_restrictions,
    COUNT(*) as total_column_privileges,
    COUNT(CASE WHEN privilege_type = 'SELECT' THEN 1 END) as read_only_columns,
    COUNT(CASE WHEN privilege_type = 'UPDATE' THEN 1 END) as updateable_columns
FROM information_schema.column_privileges
WHERE table_schema = 'school_db';
```

解答40-5

```
-- 監査対応権限管理システム

-- 1. 外部監査用の一時的権限設定

-- 外部監査人ユーザーの作成（特定IPからのみアクセス）
CREATE USER 'external_auditor'@'192.168.100.%'
```

```
IDENTIFIED BY 'ExternalAudit2025!@#'
PASSWORD EXPIRE INTERVAL 30 DAY
WITH MAX_USER_CONNECTIONS 2
     MAX_CONNECTIONS_PER_HOUR 50;

-- 全テーブル閲覧権限（個人情報制限付き）
GRANT SELECT ON school_db.courses TO 'external_auditor'@'192.168.100.%';
GRANT SELECT ON school_db.teachers TO 'external_auditor'@'192.168.100.%';
GRANT SELECT ON school_db.classrooms TO 'external_auditor'@'192.168.100.%';
GRANT SELECT ON school_db.terms TO 'external_auditor'@'192.168.100.%';
GRANT SELECT ON school_db.course_schedule TO 'external_auditor'@'192.168.100.%';

-- 個人情報を除外したカラムのみアクセス
GRANT SELECT (student_id) ON school_db.students TO
'external_auditor'@'192.168.100.%';
GRANT SELECT (course_id, grade_type, score, max_score, submission_date) ON
school_db.grades TO 'external_auditor'@'192.168.100.%';
GRANT SELECT (schedule_id, status) ON school_db.attendance TO
'external_auditor'@'192.168.100.%';

-- システムログ閲覧権限
GRANT SELECT ON mysql.general_log TO 'external_auditor'@'192.168.100.%';
GRANT SELECT ON information_schema.table_privileges TO
'external_auditor'@'192.168.100.%';
GRANT SELECT ON information_schema.user_privileges TO
'external_auditor'@'192.168.100.%';

-- 2. 内部監査人の権限設定
CREATE USER 'internal_auditor'@'localhost'
IDENTIFIED BY 'InternalAudit2025!'
PASSWORD EXPIRE INTERVAL 60 DAY;

-- より広範囲なデータアクセス権限
GRANT SELECT ON school_db.* TO 'internal_auditor'@'localhost';

-- システム管理関連の権限
GRANT SELECT ON mysql.user TO 'internal_auditor'@'localhost';
GRANT SELECT ON mysql.db TO 'internal_auditor'@'localhost';
GRANT SELECT ON performance_schema.accounts TO 'internal_auditor'@'localhost';

-- ユーザー管理ログの閲覧権限
GRANT SELECT ON school_db.user_management_log TO 'internal_auditor'@'localhost';

-- 3. 監査期間管理テーブル
CREATE TABLE audit_periods (
    audit_id INT AUTO_INCREMENT PRIMARY KEY,
    audit_type ENUM('external', 'internal') NOT NULL,
    auditor_user VARCHAR(100) NOT NULL,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    is_active BOOLEAN DEFAULT TRUE,
    created_by VARCHAR(100) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    notes TEXT,
    INDEX idx_end_date (end_date, is_active),
    INDEX idx_auditor (auditor_user)
);

-- 監査期間の登録
INSERT INTO audit_periods (audit_type, auditor_user, start_date, end_date,
created_by, notes) VALUES
('external', 'external_auditor@192.168.100.%', CURRENT_DATE,
DATE_ADD(CURRENT_DATE, INTERVAL 30 DAY), USER(), 'Annual external audit'),
('internal', 'internal_auditor@localhost', CURRENT_DATE, DATE_ADD(CURRENT_DATE,
INTERVAL 90 DAY), USER(), 'Quarterly internal audit');

-- 4. 監査期間終了時の自動権限取り消し機能
DELIMITER //

CREATE PROCEDURE auto_revoke_audit_access()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE v_auditor_user VARCHAR(100);
    DECLARE v_audit_type ENUM('external', 'internal');
    DECLARE v_audit_id INT;

    DECLARE expired_audits CURSOR FOR
        SELECT audit_id, audit_type, auditor_user
        FROM audit_periods
        WHERE end_date <= CURRENT_DATE
        AND is_active = TRUE;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN expired_audits;

    audit_loop: LOOP
        FETCH expired_audits INTO v_audit_id, v_audit_type, v_auditor_user;
        IF done THEN
            LEAVE audit_loop;
        END IF;

        -- 外部監査人の権限取り消し
        IF v_audit_type = 'external' THEN
            -- 全権限を取り消し
            SET @sql = CONCAT('REVOKE ALL PRIVILEGES ON school_db.* FROM ',
v_auditor_user);
            PREPARE stmt FROM @sql;
            EXECUTE stmt;
            DEALLOCATE PREPARE stmt;

            SET @sql = CONCAT('REVOKE ALL PRIVILEGES ON mysql.general_log FROM ',
v_auditor_user);
            PREPARE stmt FROM @sql;
            EXECUTE stmt;
            DEALLOCATE PREPARE stmt;
            -- アカウントをロック
            SET @sql = CONCAT('ALTER USER ', v_auditor_user, ' ACCOUNT LOCK');
            PREPARE stmt FROM @sql;
            EXECUTE stmt;
            DEALLOCATE PREPARE stmt;
        END IF;

        -- 内部監査人は権限を減らすが完全には取り消さない
        IF v_audit_type = 'internal' THEN
            -- 管理者権限のみ取り消し
            SET @sql = CONCAT('REVOKE SELECT ON mysql.user FROM ',
v_auditor_user);
            PREPARE stmt FROM @sql;
            EXECUTE stmt;
            DEALLOCATE PREPARE stmt;
        END IF;

        -- 監査期間を無効化
        UPDATE audit_periods SET is_active = FALSE WHERE audit_id = v_audit_id;

        -- ログ記録
        INSERT INTO user_management_log (action, username, host, user_type,
created_by, notes)
        VALUES ('REVOKE', SUBSTRING_INDEX(v_auditor_user, '@', 1),
SUBSTRING_INDEX(v_auditor_user, '@', -1),
                v_audit_type, 'AUTO_SYSTEM', CONCAT('Audit period ended -
privileges auto-revoked'));

    END LOOP;

    CLOSE expired_audits;

    SELECT CONCAT('Processed ', ROW_COUNT(), ' expired audit periods') as result;
END //

DELIMITER ;

-- 5. 監査期間中の権限使用状況記録
CREATE TABLE audit_access_log (
    log_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    auditor_user VARCHAR(100) NOT NULL,
    accessed_table VARCHAR(100),
    query_type ENUM('SELECT', 'INSERT', 'UPDATE', 'DELETE', 'OTHER') NOT NULL,
    access_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    query_hash VARCHAR(64), -- クエリのハッシュ値（プライバシー保護）
    row_count INT,

    INDEX idx_auditor_time (auditor_user, access_time),
    INDEX idx_access_time (access_time)
);

-- 監査アクセス記録プロシージャ
DELIMITER //
```

```
CREATE PROCEDURE log_audit_access(
    IN p_auditor_user VARCHAR(100),
    IN p_table_name VARCHAR(100),
    IN p_query_type ENUM('SELECT', 'INSERT', 'UPDATE', 'DELETE', 'OTHER'),
    IN p_row_count INT
)
BEGIN
    INSERT INTO audit_access_log (auditor_user, accessed_table, query_type,
row_count)
    VALUES (p_auditor_user, p_table_name, p_query_type, p_row_count);
END //

DELIMITER ;

-- 6. 監査レポート生成
CREATE VIEW v_audit_activity_report AS
SELECT
    ap.audit_type,
    ap.auditor_user,
    ap.start_date,
    ap.end_date,
    ap.is_active,
    COALESCE(access_stats.total_accesses, 0) as total_accesses,
    COALESCE(access_stats.unique_tables, 0) as unique_tables_accessed,
    COALESCE(access_stats.total_rows_accessed, 0) as total_rows_accessed
FROM audit_periods ap
LEFT JOIN (
    SELECT
        auditor_user,
        COUNT(*) as total_accesses,
        COUNT(DISTINCT accessed_table) as unique_tables,
        SUM(row_count) as total_rows_accessed
    FROM audit_access_log
    GROUP BY auditor_user
) access_stats ON ap.auditor_user = access_stats.auditor_user
ORDER BY ap.start_date DESC;

-- 監査状況の確認
SELECT * FROM v_audit_activity_report;

-- 現在アクティブな監査の確認
SELECT
    audit_type,
    auditor_user,
    DATEDIFF(end_date, CURRENT_DATE) as days_remaining,
    CASE
        WHEN end_date <= CURRENT_DATE THEN 'EXPIRED'
        WHEN DATEDIFF(end_date, CURRENT_DATE) <= 7 THEN 'EXPIRES_SOON'
        ELSE 'ACTIVE'
    END as status
FROM audit_periods
WHERE is_active = TRUE
ORDER BY end_date;
```

```
-- 権限確認
SHOW GRANTS FOR 'external_auditor'@'192.168.100.%';
SHOW GRANTS FOR 'internal_auditor'@'localhost';

-- 定期実行用（管理者が定期的に実行）
-- CALL auto_revoke_audit_access();
```

解答40-6

```
-- 総合的な権限管理システム

-- 1. 役職体系に対応したロールの作成
CREATE ROLE 'principal_role';           -- 校長
CREATE ROLE 'academic_director_role';   -- 教務主任
CREATE ROLE 'grade_supervisor_role';    -- 学年主任
CREATE ROLE 'general_teacher_role';     -- 一般教師
CREATE ROLE 'office_staff_role';        -- 事務職員
CREATE ROLE 'health_coordinator_role';  -- 保健担当
CREATE ROLE 'librarian_role';           -- 図書館司書

-- 2. 各ロールの権限設定

-- 校長：全権限
GRANT ALL PRIVILEGES ON school_db.* TO 'principal_role';
GRANT CREATE USER, DROP USER ON *.* TO 'principal_role';

-- 教務主任：成績・授業管理権限
GRANT SELECT, INSERT, UPDATE, DELETE ON school_db.grades TO
'academic_director_role';
GRANT SELECT, INSERT, UPDATE, DELETE ON school_db.attendance TO
'academic_director_role';
GRANT SELECT, INSERT, UPDATE ON school_db.courses TO 'academic_director_role';
GRANT SELECT, INSERT, UPDATE ON school_db.course_schedule TO
'academic_director_role';
GRANT SELECT ON school_db.students TO 'academic_director_role';
GRANT SELECT ON school_db.teachers TO 'academic_director_role';

-- 学年主任：担当学年の管理権限
GRANT SELECT, UPDATE ON school_db.students TO 'grade_supervisor_role';
GRANT SELECT, INSERT, UPDATE ON school_db.grades TO 'grade_supervisor_role';
GRANT SELECT, INSERT, UPDATE ON school_db.attendance TO 'grade_supervisor_role';
GRANT SELECT ON school_db.courses TO 'grade_supervisor_role';

-- 一般教師：担当クラス管理権限
GRANT SELECT ON school_db.students TO 'general_teacher_role';
GRANT SELECT, INSERT, UPDATE ON school_db.grades TO 'general_teacher_role';
GRANT SELECT, INSERT, UPDATE ON school_db.attendance TO 'general_teacher_role';
GRANT SELECT ON school_db.courses TO 'general_teacher_role';

-- 事務職員：学生情報管理権限
GRANT SELECT, INSERT, UPDATE ON school_db.students TO 'office_staff_role';
GRANT SELECT, INSERT, UPDATE ON school_db.student_courses TO 'office_staff_role';
GRANT SELECT ON school_db.courses TO 'office_staff_role';
GRANT SELECT ON school_db.teachers TO 'office_staff_role';

-- 保健担当：健康関連情報管理権限
GRANT SELECT ON school_db.students TO 'health_coordinator_role';
-- 健康管理テーブルがある場合の例
-- GRANT SELECT, INSERT, UPDATE ON school_db.health_records TO
'health_coordinator_role';

-- 図書館司書：図書館システム管理権限
GRANT SELECT ON school_db.students TO 'librarian_role';
-- 図書館管理テーブルがある場合の例
-- GRANT SELECT, INSERT, UPDATE, DELETE ON school_db.books TO 'librarian_role';
-- GRANT SELECT, INSERT, UPDATE, DELETE ON school_db.book_loans TO
'librarian_role';

-- 3. 役職管理テーブル
CREATE TABLE staff_positions (
    position_id INT AUTO_INCREMENT PRIMARY KEY,
    user_account VARCHAR(100) NOT NULL,
    position_type ENUM('principal', 'academic_director', 'grade_supervisor',
'general_teacher',
                      'office_staff', 'health_coordinator', 'librarian') NOT NULL,
    role_name VARCHAR(50) NOT NULL,
    department VARCHAR(100),
    grade_level INT, -- 学年主任の場合の担当学年
    start_date DATE NOT NULL,
    end_date DATE,
    is_active BOOLEAN DEFAULT TRUE,
    updated_by VARCHAR(100),
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,

    UNIQUE KEY unique_active_position (user_account, position_type, is_active),
    INDEX idx_position_type (position_type),
    INDEX idx_active_positions (is_active, end_date)
);

-- 4. 動的権限管理機能
DELIMITER //

-- 役職変更時の自動権限更新
CREATE PROCEDURE update_staff_position(
    IN p_user_account VARCHAR(100),
    IN p_old_position ENUM('principal', 'academic_director', 'grade_supervisor',
'general_teacher',
                          'office_staff', 'health_coordinator', 'librarian'),
    IN p_new_position ENUM('principal', 'academic_director', 'grade_supervisor',
'general_teacher',
                          'office_staff', 'health_coordinator', 'librarian'),
    IN p_department VARCHAR(100),
    IN p_grade_level INT
)
BEGIN
```

```
    DECLARE v_old_role VARCHAR(50);
    DECLARE v_new_role VARCHAR(50);

    -- 役職に対応するロール名を設定
    SET v_old_role = CONCAT(p_old_position, '_role');
    SET v_new_role = CONCAT(p_new_position, '_role');

    -- 古い役職の記録を無効化
    UPDATE staff_positions
    SET is_active = FALSE, end_date = CURRENT_DATE, updated_by = USER()
    WHERE user_account = p_user_account AND position_type = p_old_position AND
is_active = TRUE;

    -- 新しい役職の記録を追加
    INSERT INTO staff_positions (user_account, position_type, role_name,
department, grade_level, start_date, updated_by)
    VALUES (p_user_account, p_new_position, v_new_role, p_department,
p_grade_level, CURRENT_DATE, USER());

    -- 古いロールを取り消し
    SET @sql = CONCAT('REVOKE ''', v_old_role, ''' FROM ', p_user_account);
    PREPARE stmt FROM @sql;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;

    -- 新しいロールを付与
    SET @sql = CONCAT('GRANT ''', v_new_role, ''' TO ', p_user_account);
    PREPARE stmt FROM @sql;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;

    -- デフォルトロールを更新
    SET @sql = CONCAT('ALTER USER ', p_user_account, ' DEFAULT ROLE ALL');
    PREPARE stmt FROM @sql;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;

    -- 変更ログを記録
    INSERT INTO user_management_log (action, username, host, user_type,
created_by, notes)
    VALUES ('MODIFY', SUBSTRING_INDEX(p_user_account, '@', 1),
SUBSTRING_INDEX(p_user_account, '@', -1),
            p_new_position, USER(),
            CONCAT('Position changed from ', p_old_position, ' to ',
p_new_position));

    SELECT CONCAT('Successfully updated position from ', p_old_position, ' to ',
p_new_position) as result;
END //

-- 期間限定権限の管理
CREATE PROCEDURE grant_temporary_role_privilege(
    IN p_user_account VARCHAR(100),
    IN p_additional_role VARCHAR(50),
    IN p_duration_days INT,
    IN p_reason TEXT
)
BEGIN
    DECLARE expiry_date DATE;

    SET expiry_date = DATE_ADD(CURRENT_DATE, INTERVAL p_duration_days DAY);

    -- 追加ロールを付与
    SET @sql = CONCAT('GRANT ''', p_additional_role, ''' TO ', p_user_account);
    PREPARE stmt FROM @sql;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;

    -- 一時権限記録
    INSERT INTO temporary_privileges (grantee, privilege_type, object_name,
expires_at, granted_by, notes)
    VALUES (p_user_account, 'ROLE', p_additional_role,
            TIMESTAMP(expiry_date), USER(), p_reason);

    SELECT CONCAT('Granted temporary role ', p_additional_role, ' until ',
expiry_date) as result;
END //

-- 異常なアクセスパターンの検知
CREATE PROCEDURE detect_unusual_access_patterns()
BEGIN
    DECLARE unusual_activity_found BOOLEAN DEFAULT FALSE;

    -- 深夜の大量アクセス検知
    SELECT COUNT(*) > 0 INTO unusual_activity_found
    FROM information_schema.processlist p
    WHERE TIME(NOW()) BETWEEN '22:00:00' AND '06:00:00'
    AND p.User NOT IN ('root', 'mysql.sys', 'mysql.session')
    AND p.Command != 'Sleep'
    GROUP BY p.User
    HAVING COUNT(*) > 10;

    IF unusual_activity_found THEN
        INSERT INTO user_management_log (action, username, host, user_type,
created_by, notes)
        VALUES ('ALERT', 'SYSTEM', 'SYSTEM', 'security', 'AUTO_MONITOR',
                'Unusual access pattern detected - high activity during off-
hours');
    END IF;

    -- 権限昇格の検知
    SELECT COUNT(*) > 0 INTO unusual_activity_found
    FROM mysql.general_log
    WHERE event_time >= DATE_SUB(NOW(), INTERVAL 1 HOUR)
    AND argument LIKE '%GRANT%'
    AND user_host NOT LIKE '%root%';

    IF unusual_activity_found THEN
```

```
        INSERT INTO user_management_log (action, username, host, user_type,
created_by, notes)
        VALUES ('ALERT', 'SYSTEM', 'SYSTEM', 'security', 'AUTO_MONITOR',
                'Privilege escalation activity detected');
    END IF;

END //

DELIMITER ;

-- 5. 監査機能

-- 包括的権限監査ビュー
CREATE VIEW v_comprehensive_privilege_audit AS
SELECT
    sp.user_account,
    sp.position_type,
    sp.role_name,
    sp.department,
    sp.grade_level,
    sp.start_date,
    sp.end_date,
    sp.is_active as position_active,
    tp.privilege_type as table_privilege,
    tp.table_name,
    tp.is_grantable,
    CASE
        WHEN tp.grantee IS NULL THEN 'NO_TABLE_PRIVILEGES'
        ELSE 'HAS_TABLE_PRIVILEGES'
    END as privilege_status
FROM staff_positions sp
LEFT JOIN information_schema.table_privileges tp ON sp.user_account = tp.grantee
WHERE sp.is_active = TRUE
ORDER BY sp.position_type, sp.user_account;

-- 定期権限レビューレポート
CREATE PROCEDURE generate_privilege_review_report()
BEGIN
    -- アクティブユーザーの権限サマリー
    SELECT
        'Active User Privilege Summary' as report_section,
        position_type,
        COUNT(DISTINCT user_account) as user_count,
        COUNT(DISTINCT role_name) as roles_assigned,
        AVG(DATEDIFF(CURRENT_DATE, start_date)) as avg_tenure_days
    FROM staff_positions
    WHERE is_active = TRUE
    GROUP BY position_type
    ORDER BY user_count DESC;

    -- 長期未使用アカウントの特定
    SELECT
        'Long-term Inactive Accounts' as report_section,
        u.User as username,
        u.Host,
        COALESCE(acc.TOTAL_CONNECTIONS, 0) as total_logins,
        sp.position_type,
        sp.start_date
    FROM mysql.user u
    LEFT JOIN performance_schema.accounts acc ON u.User = acc.USER AND u.Host =
acc.HOST
    LEFT JOIN staff_positions sp ON CONCAT('''', u.User, '''@''', u.Host, '''') =
sp.user_account
    WHERE u.User NOT IN ('root', 'mysql.sys', 'mysql.session', 'mysql.infoschema')
    AND (acc.TOTAL_CONNECTIONS IS NULL OR acc.TOTAL_CONNECTIONS = 0)
    AND sp.start_date < DATE_SUB(CURRENT_DATE, INTERVAL 90 DAY)
    ORDER BY sp.start_date;

    -- 危険な権限を持つユーザー
    SELECT
        'Users with High-Risk Privileges' as report_section,
        up.grantee,
        sp.position_type,
        GROUP_CONCAT(up.privilege_type) as dangerous_privileges
    FROM information_schema.user_privileges up
    LEFT JOIN staff_positions sp ON up.grantee = sp.user_account AND sp.is_active
= TRUE
    WHERE up.privilege_type IN ('SUPER', 'CREATE USER', 'GRANT OPTION', 'FILE',
'PROCESS')
    GROUP BY up.grantee, sp.position_type
    ORDER BY sp.position_type;
END //

-- セキュリティリスクの自動検出
CREATE PROCEDURE detect_security_risks()
BEGIN
    -- 共有アカウントの検出
    SELECT
        'Potential Shared Accounts' as risk_category,
        acc.USER,
        acc.HOST,
        acc.CURRENT_CONNECTIONS,
        'Multiple simultaneous connections from same account' as risk_description
    FROM performance_schema.accounts acc
    WHERE acc.CURRENT_CONNECTIONS > 3
    AND acc.USER NOT IN ('root', 'mysql.sys');

    -- 過度な権限を持つアカウント
    SELECT
        'Over-privileged Accounts' as risk_category,
        tp.grantee,
        COUNT(DISTINCT tp.table_name) as accessible_tables,
        'Access to excessive number of tables' as risk_description
    FROM information_schema.table_privileges tp
    WHERE tp.table_schema = 'school_db'
    GROUP BY tp.grantee
    HAVING COUNT(DISTINCT tp.table_name) > 8; -- 閾値は環境に応じて調整
```

```sql
    -- 期限切れパスワードで有効なアカウント
    SELECT
        'Expired Password Active Accounts' as risk_category,
        u.User,
        u.Host,
        u.password_expired,
        u.account_locked,
        'Account with expired password still active' as risk_description
    FROM mysql.user u
    WHERE u.password_expired = 'Y'
    AND u.account_locked = 'N'
    AND u.User NOT IN ('root', 'mysql.sys', 'mysql.session', 'mysql.infoschema');

END //

DELIMITER ;

-- 6. 運用管理機能

-- 権限付与・取り消しの承認ワークフロー
CREATE TABLE privilege_approval_requests (
    request_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    requested_by VARCHAR(100) NOT NULL,
    target_user VARCHAR(100) NOT NULL,
    request_type ENUM('GRANT', 'REVOKE') NOT NULL,
    privilege_details TEXT NOT NULL,
    business_justification TEXT NOT NULL,
    requested_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    approved_by VARCHAR(100),
    approved_at TIMESTAMP NULL,
    status ENUM('PENDING', 'APPROVED', 'REJECTED', 'EXECUTED') DEFAULT 'PENDING',
    execution_notes TEXT,

    INDEX idx_status (status),
    INDEX idx_requested_by (requested_by),
    INDEX idx_target_user (target_user)
);

-- 承認ワークフロープロシージャ
DELIMITER //

CREATE PROCEDURE submit_privilege_request(
    IN p_target_user VARCHAR(100),
    IN p_request_type ENUM('GRANT', 'REVOKE'),
    IN p_privilege_details TEXT,
    IN p_justification TEXT
)
BEGIN
    INSERT INTO privilege_approval_requests
        (requested_by, target_user, request_type, privilege_details,
business_justification)
    VALUES (USER(), p_target_user, p_request_type, p_privilege_details,
p_justification);
```

```
    SELECT CONCAT('Privilege request submitted with ID: ', LAST_INSERT_ID()) as
result;
END //

CREATE PROCEDURE approve_privilege_request(
    IN p_request_id BIGINT,
    IN p_execution_notes TEXT
)
BEGIN
    DECLARE v_target_user VARCHAR(100);
    DECLARE v_request_type ENUM('GRANT', 'REVOKE');
    DECLARE v_privilege_details TEXT;
    DECLARE v_status ENUM('PENDING', 'APPROVED', 'REJECTED', 'EXECUTED');

    -- リクエスト情報を取得
    SELECT target_user, request_type, privilege_details, status
    INTO v_target_user, v_request_type, v_privilege_details, v_status
    FROM privilege_approval_requests
    WHERE request_id = p_request_id;

    IF v_status != 'PENDING' THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Request is not in pending
status';
    END IF;

    -- 承認状態に更新
    UPDATE privilege_approval_requests
    SET approved_by = USER(),
        approved_at = NOW(),
        status = 'APPROVED',
        execution_notes = p_execution_notes
    WHERE request_id = p_request_id;

    -- 実際の権限変更を実行（簡単な例）
    IF v_request_type = 'GRANT' THEN
        -- 実際の環境では、privilege_detailsを解析して適切なGRANT文を構築
        SET @sql = v_privilege_details;
        PREPARE stmt FROM @sql;
        EXECUTE stmt;
        DEALLOCATE PREPARE stmt;
    ELSEIF v_request_type = 'REVOKE' THEN
        -- 実際の環境では、privilege_detailsを解析して適切なREVOKE文を構築
        SET @sql = v_privilege_details;
        PREPARE stmt FROM @sql;
        EXECUTE stmt;
        DEALLOCATE PREPARE stmt;
    END IF;

    -- 実行完了状態に更新
    UPDATE privilege_approval_requests
    SET status = 'EXECUTED'
    WHERE request_id = p_request_id;

    -- ログ記録
    INSERT INTO user_management_log (action, username, host, user_type,
created_by, notes)
    VALUES (v_request_type, SUBSTRING_INDEX(v_target_user, '@', 1),
SUBSTRING_INDEX(v_target_user, '@', -1),
            'workflow', USER(), CONCAT('Approved request ID: ', p_request_id));

    SELECT 'Privilege request approved and executed' as result;
END //

DELIMITER ;

-- 一括権限管理機能
DELIMITER //

CREATE PROCEDURE bulk_privilege_management(
    IN p_operation ENUM('GRANT', 'REVOKE'),
    IN p_role_name VARCHAR(50),
    IN p_user_list TEXT -- カンマ区切りのユーザーリスト
)
BEGIN
    DECLARE v_user VARCHAR(100);
    DECLARE v_pos INT DEFAULT 1;
    DECLARE v_next_pos INT;

    -- ユーザーリストを分割して処理
    WHILE v_pos <= LENGTH(p_user_list) DO
        SET v_next_pos = LOCATE(',', p_user_list, v_pos);

        IF v_next_pos = 0 THEN
            SET v_next_pos = LENGTH(p_user_list) + 1;
        END IF;

        SET v_user = TRIM(SUBSTRING(p_user_list, v_pos, v_next_pos - v_pos));

        IF LENGTH(v_user) > 0 THEN
            IF p_operation = 'GRANT' THEN
                SET @sql = CONCAT('GRANT ''', p_role_name, ''' TO ', v_user);
                PREPARE stmt FROM @sql;
                EXECUTE stmt;
                DEALLOCATE PREPARE stmt;

                SET @sql = CONCAT('ALTER USER ', v_user, ' DEFAULT ROLE ALL');
                PREPARE stmt FROM @sql;
                EXECUTE stmt;
                DEALLOCATE PREPARE stmt;
            ELSE
                SET @sql = CONCAT('REVOKE ''', p_role_name, ''' FROM ', v_user);
                PREPARE stmt FROM @sql;
                EXECUTE stmt;
                DEALLOCATE PREPARE stmt;
            END IF;

            -- ログ記録
            INSERT INTO user_management_log (action, username, host, user_type,
created_by, notes)
            VALUES (p_operation, SUBSTRING_INDEX(v_user, '@', 1),
SUBSTRING_INDEX(v_user, '@', -1),
                    'bulk', USER(), CONCAT('Bulk ', p_operation, ' of role: ',
p_role_name));
        END IF;

        SET v_pos = v_next_pos + 1;
    END WHILE;

    SELECT CONCAT('Bulk ', p_operation, ' completed for role: ', p_role_name) as
result;
END //

DELIMITER ;

-- 緊急時の権限停止機能
DELIMITER //

CREATE PROCEDURE emergency_privilege_suspension(
    IN p_user_account VARCHAR(100),
    IN p_reason TEXT
)
BEGIN
    -- アカウントを即座にロック
    SET @sql = CONCAT('ALTER USER ', p_user_account, ' ACCOUNT LOCK');
    PREPARE stmt FROM @sql;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;

    -- 現在のセッションを強制終了
    -- 注意: 実際の環境では慎重に実行する必要がある
    /*
    UPDATE information_schema.processlist
    SET COMMAND = 'Kill'
    WHERE USER = SUBSTRING_INDEX(p_user_account, '@', 1)
    AND HOST LIKE CONCAT('%', SUBSTRING_INDEX(p_user_account, '@', -1), '%');
    */

    -- 緊急停止ログを記録
    INSERT INTO user_management_log (action, username, host, user_type,
created_by, notes)
    VALUES ('EMERGENCY_LOCK', SUBSTRING_INDEX(p_user_account, '@', 1),
SUBSTRING_INDEX(p_user_account, '@', -1),
            'emergency', USER(), CONCAT('EMERGENCY SUSPENSION: ', p_reason));

    -- 管理者に通知（実際の環境では外部通知システムと連携）
    SELECT CONCAT('EMERGENCY: User ', p_user_account, ' has been suspended.
Reason: ', p_reason) as alert;
END //

DELIMITER ;

-- 7. システム使用例とテスト
-- サンプルユーザーの作成（コメントアウト状態）
/*
-- 各役職のサンプルユーザー作成
CREATE USER 'principal_yamada'@'localhost' IDENTIFIED BY 'Principal2025!';
CREATE USER 'academic_suzuki'@'localhost' IDENTIFIED BY 'Academic2025!';
CREATE USER 'grade_head_tanaka'@'localhost' IDENTIFIED BY 'GradeHead2025!';
CREATE USER 'teacher_sato'@'localhost' IDENTIFIED BY 'Teacher2025!';
CREATE USER 'office_kimura'@'localhost' IDENTIFIED BY 'Office2025!';
CREATE USER 'health_takahashi'@'localhost' IDENTIFIED BY 'Health2025!';
CREATE USER 'librarian_watanabe'@'localhost' IDENTIFIED BY 'Library2025!';

-- ロールの付与
GRANT 'principal_role' TO 'principal_yamada'@'localhost';
GRANT 'academic_director_role' TO 'academic_suzuki'@'localhost';
GRANT 'grade_supervisor_role' TO 'grade_head_tanaka'@'localhost';
GRANT 'general_teacher_role' TO 'teacher_sato'@'localhost';
GRANT 'office_staff_role' TO 'office_kimura'@'localhost';
GRANT 'health_coordinator_role' TO 'health_takahashi'@'localhost';
GRANT 'librarian_role' TO 'librarian_watanabe'@'localhost';

-- デフォルトロールの設定
ALTER USER 'principal_yamada'@'localhost' DEFAULT ROLE ALL;
ALTER USER 'academic_suzuki'@'localhost' DEFAULT ROLE ALL;
ALTER USER 'grade_head_tanaka'@'localhost' DEFAULT ROLE ALL;
ALTER USER 'teacher_sato'@'localhost' DEFAULT ROLE ALL;
ALTER USER 'office_kimura'@'localhost' DEFAULT ROLE ALL;
ALTER USER 'health_takahashi'@'localhost' DEFAULT ROLE ALL;
ALTER USER 'librarian_watanabe'@'localhost' DEFAULT ROLE ALL;

-- 役職情報の登録
INSERT INTO staff_positions (user_account, position_type, role_name, department,
start_date, updated_by) VALUES
('principal_yamada@localhost', 'principal', 'principal_role', '学校全体',
CURRENT_DATE, USER()),
('academic_suzuki@localhost', 'academic_director', 'academic_director_role', '教務
部', CURRENT_DATE, USER()),
('grade_head_tanaka@localhost', 'grade_supervisor', 'grade_supervisor_role', '普通
科', 2, CURRENT_DATE, USER()),
('teacher_sato@localhost', 'general_teacher', 'general_teacher_role', '普通科',
CURRENT_DATE, USER()),
('office_kimura@localhost', 'office_staff', 'office_staff_role', '事務部',
CURRENT_DATE, USER()),
('health_takahashi@localhost', 'health_coordinator', 'health_coordinator_role',
'保健室', CURRENT_DATE, USER()),
('librarian_watanabe@localhost', 'librarian', 'librarian_role', '図書館',
CURRENT_DATE, USER());
*/

-- 8. 運用レポートとモニタリング

-- 現在の権限状況レポート
SELECT 'Current System Privilege Status' as report_title;
```

```
SELECT
    'Active Staff Positions' as section,
    position_type,
    COUNT(*) as count,
    GROUP_CONCAT(DISTINCT role_name) as assigned_roles
FROM staff_positions
WHERE is_active = TRUE
GROUP BY position_type
ORDER BY count DESC;

-- 権限変更履歴サマリー
SELECT
    'Recent Privilege Changes (Last 30 Days)' as section,
    action,
    user_type,
    COUNT(*) as change_count,
    COUNT(DISTINCT username) as affected_users
FROM user_management_log
WHERE created_at >= DATE_SUB(NOW(), INTERVAL 30 DAY)
GROUP BY action, user_type
ORDER BY change_count DESC;

-- セキュリティアラート状況
SELECT
    'Security Alerts Summary' as section,
    DATE(created_at) as alert_date,
    COUNT(*) as alert_count,
    GROUP_CONCAT(DISTINCT notes) as alert_types
FROM user_management_log
WHERE action = 'ALERT'
AND created_at >= DATE_SUB(NOW(), INTERVAL 7 DAY)
GROUP BY DATE(created_at)
ORDER BY alert_date DESC;

-- 権限承認ワークフロー状況
SELECT
    'Privilege Request Workflow Status' as section,
    status,
    COUNT(*) as request_count,
    AVG(TIMESTAMPDIFF(HOUR, requested_at, IFNULL(approved_at, NOW()))) as
avg_processing_hours
FROM privilege_approval_requests
WHERE requested_at >= DATE_SUB(NOW(), INTERVAL 30 DAY)
GROUP BY status
ORDER BY request_count DESC;

-- 最終チェック: 全体的な権限監査
SELECT * FROM v_comprehensive_privilege_audit LIMIT 10;

-- 定期メンテナンス実行例（コメントアウト）
-- CALL generate_privilege_review_report();
-- CALL detect_security_risks();
-- CALL auto_revoke_audit_access();
```

## まとめ

この章では、MySQLにおけるGRANT/REVOKEを使った権限の付与と取り消しについて詳しく学びました：

1. **権限管理の基本概念**：
   - 権限の種類と分類（データ操作、構造管理、実行、管理、特殊権限）
   - 権限の適用レベル（グローバル、データベース、テーブル、カラム）
   - グランターとグランティーの関係
2. **GRANT文の活用**：
   - 基本的な権限付与の構文
   - 複数権限の同時付与
   - カラムレベルの細かい権限制御
   - WITH GRANT OPTIONによる権限委譲
3. **REVOKE文による権限取り消し**：
   - 基本的な権限取り消し構文
   - 段階的な権限削減
   - 安全な権限取り消し手順
   - 委譲権限の連鎖取り消し
4. **実践的な権限管理シナリオ**：
   - 新学期の教師権限設定
   - 学期末の権限調整
   - 監査・検査対応の特別権限設定
   - 役職変更に伴う権限更新
5. **高度な権限管理テクニック**：
   - 条件付き権限管理
   - 時間制限付き権限
   - 動的権限管理
   - 役職ベースの権限体系
6. **包括的な権限管理システム**：
   - 承認ワークフローの実装
   - 一括権限管理機能
   - 緊急時の権限停止機能
   - 監査とレポート機能
7. **エラー処理と安全対策**：
   - 権限不足エラーの対処
   - 存在しないオブジェクトへの権限付与エラー
   - 循環的な権限委譲の防止
   - セキュリティリスクの自動検出

GRANT/REVOKEは、データベースセキュリティの中核となる重要な機能です。**最小権限の原則**を基本とし、各ユーザーには業務に必要な最小限の権限のみを付与することが安全な運用の鍵となります。また、権限の変更は必ず記録し、定期的な監査を行うことで、セキュリティインシデントの防止と早期発見が可能になります。

次の章では、「行レベルセキュリティ：ビューとプロシージャによる細粒度制御」について学び、より詳細なアクセス制御手法を理解していきます。

---

# 41. 行レベルセキュリティ：ビューとプロシージャによる細粒度制御

## はじめに

前章では、GRANT/REVOKE文を使った基本的な権限管理について学習しました。この章では、より細かい制御が可能な「行レベルセキュリティ」について学習します。

従来のテーブルレベル・カラムレベルの権限制御では、「教師は成績テーブルにアクセスできる」といったレベルの制御しかできませんでした。しかし実際の学校システムでは、「教師は担当クラスの学生の成績のみアクセスできる」「学生は自分の成績のみ閲覧できる」といった、データの行（レコード）レベルでの細かい制御が必要です。

行レベルセキュリティが必要となる場面の例：

- 「教師は担当するクラスの学生情報のみ閲覧・変更できる」
- 「学生は自分の成績や出席状況のみ確認できる」
- 「学年主任は担当学年の学生のみ管理できる」
- 「保護者は自分の子供の情報のみアクセスできる」
- 「事務職員は在籍中の学生のみ管理でき、卒業生は閲覧のみ」
- 「カウンセラーは相談対象の学生のみアクセスできる」
- 「医務室スタッフは健康診断対象の学生のみ健康情報を管理できる」

この章では、MySQLでビューとストアドプロシージャを活用して、柔軟で安全な行レベルセキュリティを実装する方法を学習します。

## 行レベルセキュリティの基本概念

### 行レベルセキュリティとは

**行レベルセキュリティ**（Row-Level Security, RLS）とは、テーブル内の特定の行に対するアクセスを、ユーザーの属性や条件に基づいて制御するセキュリティ機能です。

### 実装方式の比較

> **用語解説**：
>
> - **行レベルセキュリティ（Row-Level Security, RLS）**：テーブル内の特定の行へのアクセスを制御するセキュリティ機能です。

> - **セキュリティビュー（Security View）**：行レベルの制御を行うために作成される、特定の条件でデータをフィルタリングするビューです。
> - **セキュリティコンテキスト**：現在のユーザーの属性や権限を表す情報です（ユーザーID、役職、部署等）。
> - **述語（Predicate）**：行レベルセキュリティで使用される、アクセス可能な行を決定する条件式です。
> - **セキュリティポリシー**：どのユーザーがどの行にアクセスできるかを定義するルールです。
> - **動的セキュリティ**：ユーザーの属性やコンテキストに基づいて動的に変化するセキュリティ制御です。
> - **VPD（Virtual Private Database）**：各ユーザーに対して仮想的に専用のデータベースを提供する技術です。
> - **アプリケーションレベルセキュリティ**：データベース機能ではなく、アプリケーションコードで実装するセキュリティです。
> - **セキュリティコンテナ**：セキュリティ制御を行う単位（クラス、学年、部署等）です。

| 実装方式 | メリット | デメリット | 適用場面 |
|---|---|---|---|
| **セキュリティビュー** | 実装が簡単、理解しやすい | ビューの管理が複雑化 | 基本的な行レベル制御 |
| **ストアドプロシージャ** | 柔軟な制御、複雑なロジック対応 | 実装が複雑、保守が困難 | 高度な業務ロジック |
| **アプリケーション制御** | 最大の柔軟性 | セキュリティホールのリスク | 複雑な業務要件 |
| **関数ベース制御** | 再利用性が高い | パフォーマンスに注意が必要 | 共通ロジックの実装 |

## ビューによる行レベル制御

### 1. 基本的なセキュリティビューの作成

```sql
-- 学校システムでの基本的なセキュリティビュー例

-- 1. 教師用の担当学生ビュー
-- 教師は自分が担当する講座を受講している学生のみ閲覧可能
CREATE VIEW v_teacher_students AS
SELECT DISTINCT
    s.student_id,
    s.student_name,
    s.student_email,
    sc.course_id,
    c.course_name
FROM students s
JOIN student_courses sc ON s.student_id = sc.student_id
JOIN courses c ON sc.course_id = c.course_id
JOIN teachers t ON c.teacher_id = t.teacher_id
WHERE t.teacher_name = USER() -- 現在のユーザー名と一致する教師
   OR USER() = 'root'; -- 管理者は全て閲覧可能
```

```
-- 2. 学生用の自分専用ビュー
-- 学生は自分の情報のみ閲覧可能
CREATE VIEW v_student_own_info AS
SELECT
    student_id,
    student_name,
    student_email
FROM students
WHERE CONCAT('student_', student_id) = SUBSTRING_INDEX(USER(), '@', 1) --
student_301 のようなユーザー名
   OR USER() = 'root';

-- 3. 学生の成績閲覧ビュー
CREATE VIEW v_student_own_grades AS
SELECT
    g.student_id,
    g.course_id,
    c.course_name,
    g.grade_type,
    g.score,
    g.max_score,
    g.submission_date,
    ROUND((g.score / g.max_score) * 100, 1) as percentage
FROM grades g
JOIN courses c ON g.course_id = c.course_id
WHERE CONCAT('student_', g.student_id) = SUBSTRING_INDEX(USER(), '@', 1)
   OR USER() = 'root';

-- ビューの使用例確認
SELECT * FROM v_teacher_students;
SELECT * FROM v_student_own_info;
SELECT * FROM v_student_own_grades;
```

## 2. 複雑な条件を持つセキュリティビュー

```
-- より複雑な業務ロジックを含むセキュリティビュー

-- 1. 学年主任用の担当学年学生ビュー
CREATE VIEW v_grade_supervisor_students AS
SELECT
    s.student_id,
    s.student_name,
    s.student_email,
    s.admission_date,
    YEAR(CURRENT_DATE) - YEAR(s.admission_date) + 1 as current_grade,
    e.course_id,
    c.course_name
FROM students s
LEFT JOIN student_courses e ON s.student_id = e.student_id
LEFT JOIN courses c ON e.course_id = c.course_id
```

```
WHERE (
    -- 学年主任は担当学年の学生のみ
    CASE
        WHEN USER() LIKE '%grade_1%' THEN YEAR(CURRENT_DATE) -
YEAR(s.admission_date) + 1 = 1
        WHEN USER() LIKE '%grade_2%' THEN YEAR(CURRENT_DATE) -
YEAR(s.admission_date) + 1 = 2
        WHEN USER() LIKE '%grade_3%' THEN YEAR(CURRENT_DATE) -
YEAR(s.admission_date) + 1 = 3
        ELSE FALSE
    END
    OR USER() = 'root'
);

-- 2. 事務職員用の在籍学生ビュー（卒業生は除外）
CREATE VIEW v_office_active_students AS
SELECT
    s.student_id,
    s.student_name,
    s.student_email,
    s.admission_date,
    -- 在籍状況の判定
    CASE
        WHEN YEAR(CURRENT_DATE) - YEAR(s.admission_date) >= 3 THEN '卒業予定'
        WHEN YEAR(CURRENT_DATE) - YEAR(s.admission_date) >= 2 THEN '3年生'
        WHEN YEAR(CURRENT_DATE) - YEAR(s.admission_date) >= 1 THEN '2年生'
        ELSE '1年生'
    END as student_status
FROM students s
WHERE (
    -- 事務職員は在籍中の学生のみ（3年未満）
    (USER() LIKE '%office%' AND YEAR(CURRENT_DATE) - YEAR(s.admission_date) < 3)
    OR USER() = 'root'
    OR USER() LIKE '%principal%' -- 校長は全て閲覧可能
);

-- 3. 動的な期間制限を含む成績ビュー
CREATE VIEW v_recent_grades_by_role AS
SELECT
    g.student_id,
    s.student_name,
    g.course_id,
    c.course_name,
    g.grade_type,
    g.score,
    g.submission_date,
    -- ユーザーの役職に応じて閲覧期間を制限
    CASE
        WHEN USER() LIKE '%teacher%' THEN DATEDIFF(CURRENT_DATE,
g.submission_date) <= 365 -- 教師は1年分
        WHEN USER() LIKE '%student%' THEN DATEDIFF(CURRENT_DATE,
g.submission_date) <= 180 -- 学生は半年分
        ELSE TRUE -- 管理者は全期間
    END as within_access_period
```

```
FROM grades g
JOIN courses c ON g.course_id = c.course_id
JOIN students s ON g.student_id = s.student_id
WHERE (
    -- 期間制限の適用
    CASE
        WHEN USER() LIKE '%teacher%' THEN DATEDIFF(CURRENT_DATE,
g.submission_date) <= 365
        WHEN USER() LIKE '%student%' THEN
            DATEDIFF(CURRENT_DATE, g.submission_date) <= 180
            AND CONCAT('student_', g.student_id) = SUBSTRING_INDEX(USER(), '@', 1)
        ELSE TRUE
    END
);

-- ビューのテスト
DESCRIBE v_grade_supervisor_students;
DESCRIBE v_office_active_students;
DESCRIBE v_recent_grades_by_role;
```

### 3. セキュリティビューの権限設定

```
-- セキュリティビューに対する適切な権限設定

-- 1. 教師向けビューの権限設定
-- 教師ロールに担当学生ビューの閲覧権限を付与
GRANT SELECT ON school_db.v_teacher_students TO 'teacher_role';

-- 担当学生の成績管理権限（ビューではなく条件付きテーブルアクセス）
-- 実際の成績入力用の制限付きビューを作成
CREATE VIEW v_teacher_grade_management AS
SELECT
    g.student_id,
    g.course_id,
    g.grade_type,
    g.score,
    g.max_score,
    g.submission_date
FROM grades g
JOIN courses c ON g.course_id = c.course_id
JOIN teachers t ON c.teacher_id = t.teacher_id
WHERE t.teacher_name = USER() OR USER() = 'root';

-- 教師に成績管理ビューの更新権限を付与
GRANT SELECT, INSERT, UPDATE ON school_db.v_teacher_grade_management TO
'teacher_role';

-- 2. 学生向けビューの権限設定
GRANT SELECT ON school_db.v_student_own_info TO 'student_role';
GRANT SELECT ON school_db.v_student_own_grades TO 'student_role';
```

```
-- 3. 管理職向けビューの権限設定
GRANT SELECT ON school_db.v_grade_supervisor_students TO 'grade_supervisor_role';
GRANT SELECT ON school_db.v_office_active_students TO 'office_staff_role';

-- 4. 制限付きアクセスの確認
-- 各ロールに適切にビューの権限が設定されているか確認
SELECT
    table_name,
    grantee,
    privilege_type
FROM information_schema.table_privileges
WHERE table_schema = 'school_db'
AND table_name LIKE 'v_%'
ORDER BY table_name, grantee;
```

## ストアドプロシージャによるセキュリティ実装

### 1. ユーザーコンテキスト管理

```
-- ユーザーコンテキスト管理システム

-- 1. ユーザーコンテキスト情報テーブル
CREATE TABLE user_security_context (
    context_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(100) NOT NULL,
    user_type ENUM('student', 'teacher', 'grade_supervisor', 'office_staff',
'principal', 'admin') NOT NULL,
    associated_id BIGINT, -- 学生ID、教師ID等
    department_id INT,
    grade_level INT, -- 担当学年（学年主任の場合）
    access_restrictions JSON, -- 追加の制限条件
    context_expires_at TIMESTAMP,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,

    UNIQUE KEY unique_username (username),
    INDEX idx_user_type (user_type),
    INDEX idx_associated_id (associated_id)
);

-- サンプルコンテキストデータの挿入
INSERT INTO user_security_context (username, user_type, associated_id,
grade_level) VALUES
('teacher_tanaka@localhost', 'teacher', 101, NULL),
('teacher_sato@localhost', 'teacher', 102, NULL),
('student_301@localhost', 'student', 301, NULL),
('student_302@localhost', 'student', 302, NULL),
('grade_head_1@localhost', 'grade_supervisor', NULL, 1),
('office_staff@localhost', 'office_staff', NULL, NULL);

-- 2. ユーザーコンテキスト取得プロシージャ
```

```
DELIMITER //

CREATE PROCEDURE get_user_security_context(
    OUT p_user_type VARCHAR(20),
    OUT p_associated_id BIGINT,
    OUT p_grade_level INT,
    OUT p_access_restrictions JSON
)
BEGIN
    DECLARE v_username VARCHAR(100);

    SET v_username = USER();

    SELECT user_type, associated_id, grade_level, access_restrictions
    INTO p_user_type, p_associated_id, p_grade_level, p_access_restrictions
    FROM user_security_context
    WHERE username = v_username;

    -- デフォルト値の設定（コンテキストが見つからない場合）
    IF p_user_type IS NULL THEN
        SET p_user_type = 'unknown';
        SET p_associated_id = NULL;
        SET p_grade_level = NULL;
        SET p_access_restrictions = '{}';
    END IF;
END //

DELIMITER ;

-- 3. セキュリティ検証プロシージャ
DELIMITER //

CREATE PROCEDURE verify_data_access(
    IN p_table_name VARCHAR(64),
    IN p_operation ENUM('SELECT', 'INSERT', 'UPDATE', 'DELETE'),
    IN p_student_id BIGINT,
    OUT p_access_granted BOOLEAN
)
BEGIN
    DECLARE v_user_type VARCHAR(20);
    DECLARE v_associated_id BIGINT;
    DECLARE v_grade_level INT;
    DECLARE v_restrictions JSON;
    DECLARE v_student_grade INT;

    SET p_access_granted = FALSE;

    -- ユーザーコンテキストを取得
    CALL get_user_security_context(v_user_type, v_associated_id, v_grade_level,
v_restrictions);

    -- 管理者は常にアクセス許可
    IF USER() = 'root' OR v_user_type = 'admin' THEN
        SET p_access_granted = TRUE;
    ELSEIF v_user_type = 'student' THEN
        -- 学生は自分のデータのみアクセス可能
        IF p_student_id = v_associated_id THEN
            SET p_access_granted = TRUE;
        END IF;
    ELSEIF v_user_type = 'teacher' THEN
        -- 教師は担当講座の学生のみアクセス可能
        SELECT COUNT(*) > 0 INTO p_access_granted
        FROM student_courses sc
        JOIN courses c ON sc.course_id = c.course_id
        WHERE sc.student_id = p_student_id
        AND c.teacher_id = v_associated_id;
    ELSEIF v_user_type = 'grade_supervisor' THEN
        -- 学年主任は担当学年の学生のみアクセス可能
        SELECT (YEAR(CURRENT_DATE) - YEAR(admission_date) + 1) INTO
v_student_grade
        FROM students
        WHERE student_id = p_student_id;

        IF v_student_grade = v_grade_level THEN
            SET p_access_granted = TRUE;
        END IF;
    ELSEIF v_user_type = 'office_staff' THEN
        -- 事務職員は在籍中の学生のみアクセス可能
        SELECT COUNT(*) > 0 INTO p_access_granted
        FROM students s
        WHERE s.student_id = p_student_id
        AND YEAR(CURRENT_DATE) - YEAR(s.admission_date) < 3; -- 3年未満
    END IF;

END //

DELIMITER ;
```

## 2. 安全なデータアクセスプロシージャ

```
-- 安全なデータアクセスのためのプロシージャ群

-- 1. 学生情報の安全な取得
DELIMITER //

CREATE PROCEDURE secure_get_student_info(
    IN p_student_id BIGINT
)
BEGIN
    DECLARE v_access_granted BOOLEAN DEFAULT FALSE;

    -- アクセス権限の検証
    CALL verify_data_access('students', 'SELECT', p_student_id, v_access_granted);

    IF v_access_granted THEN
```

```
        SELECT
            student_id,
            student_name,
            student_email,
            admission_date,
            YEAR(CURRENT_DATE) - YEAR(admission_date) + 1 as current_grade
        FROM students
        WHERE student_id = p_student_id;
    ELSE
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Access denied: Insufficient 
privileges for student data';
    END IF;
END //

-- 2. 成績情報の安全な取得
CREATE PROCEDURE secure_get_student_grades(
    IN p_student_id BIGINT,
    IN p_course_id VARCHAR(16)
)
BEGIN
    DECLARE v_access_granted BOOLEAN DEFAULT FALSE;
    DECLARE v_user_type VARCHAR(20);
    DECLARE v_associated_id BIGINT;
    DECLARE v_grade_level INT;
    DECLARE v_restrictions JSON;

    -- アクセス権限の検証
    CALL verify_data_access('grades', 'SELECT', p_student_id, v_access_granted);
    CALL get_user_security_context(v_user_type, v_associated_id, v_grade_level, 
v_restrictions);

    IF v_access_granted THEN
        SELECT
            g.student_id,
            s.student_name,
            g.course_id,
            c.course_name,
            g.grade_type,
            g.score,
            g.max_score,
            g.submission_date,
            ROUND((g.score / g.max_score) * 100, 1) as percentage
        FROM grades g
        JOIN students s ON g.student_id = s.student_id
        JOIN courses c ON g.course_id = c.course_id
        WHERE g.student_id = p_student_id
        AND (p_course_id IS NULL OR g.course_id = p_course_id)
        AND (
            v_user_type = 'admin'
            OR (v_user_type = 'student' AND g.student_id = v_associated_id)
            OR (v_user_type = 'teacher' AND c.teacher_id = v_associated_id)
            OR v_user_type IN ('grade_supervisor', 'office_staff')
        )
        ORDER BY g.submission_date DESC;
    ELSE
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Access denied: Insufficient
privileges for grade data';
    END IF;
END //

-- 3. 成績の安全な更新
CREATE PROCEDURE secure_update_student_grade(
    IN p_student_id BIGINT,
    IN p_course_id VARCHAR(16),
    IN p_grade_type VARCHAR(32),
    IN p_score DECIMAL(5,2),
    IN p_max_score DECIMAL(5,2)
)
BEGIN
    DECLARE v_access_granted BOOLEAN DEFAULT FALSE;
    DECLARE v_user_type VARCHAR(20);
    DECLARE v_associated_id BIGINT;
    DECLARE v_grade_level INT;
    DECLARE v_restrictions JSON;
    DECLARE v_teacher_authorized BOOLEAN DEFAULT FALSE;

    -- ユーザーコンテキストを取得
    CALL get_user_security_context(v_user_type, v_associated_id, v_grade_level,
v_restrictions);

    -- アクセス権限の検証
    CALL verify_data_access('grades', 'UPDATE', p_student_id, v_access_granted);

    -- 教師の場合、担当講座かどうかを追加確認
    IF v_user_type = 'teacher' THEN
        SELECT COUNT(*) > 0 INTO v_teacher_authorized
        FROM courses c
        WHERE c.course_id = p_course_id
        AND c.teacher_id = v_associated_id;

        IF NOT v_teacher_authorized THEN
            SET v_access_granted = FALSE;
        END IF;
    END IF;

    IF v_access_granted THEN
        -- 成績の更新または挿入
        INSERT INTO grades (student_id, course_id, grade_type, score, max_score,
submission_date)
        VALUES (p_student_id, p_course_id, p_grade_type, p_score, p_max_score,
CURRENT_DATE)
        ON DUPLICATE KEY UPDATE
            score = p_score,
            max_score = p_max_score,
            submission_date = CURRENT_DATE;

        -- 操作ログの記録
        INSERT INTO user_management_log (action, username, host, user_type,
created_by, notes)
        VALUES ('UPDATE', CONCAT('student_', p_student_id), 'system',
'grade_update', USER(),
                CONCAT('Updated grade for course ', p_course_id, ', type: ',
p_grade_type));

        SELECT 'Grade updated successfully' as result;
    ELSE
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Access denied: Insufficient
privileges to update grades';
    END IF;
END //

DELIMITER ;
```

3. 動的セキュリティプロシージャ

```
-- 動的な条件に基づくセキュリティ制御

-- 1. 時間制限付きアクセス制御
DELIMITER //

CREATE PROCEDURE secure_time_restricted_access(
    IN p_table_name VARCHAR(64),
    IN p_operation ENUM('SELECT', 'INSERT', 'UPDATE', 'DELETE'),
    IN p_filter_conditions TEXT,
    OUT p_access_result TEXT
)
BEGIN
    DECLARE v_user_type VARCHAR(20);
    DECLARE v_associated_id BIGINT;
    DECLARE v_grade_level INT;
    DECLARE v_restrictions JSON;
    DECLARE v_current_hour INT;
    DECLARE v_access_allowed BOOLEAN DEFAULT TRUE;

    SET v_current_hour = HOUR(NOW());

    -- ユーザーコンテキストを取得
    CALL get_user_security_context(v_user_type, v_associated_id, v_grade_level,
v_restrictions);

    -- 時間制限の確認
    IF v_user_type = 'student' THEN
        -- 学生は平日の8:00-18:00のみアクセス可能
        IF DAYOFWEEK(CURRENT_DATE) IN (1, 7) OR v_current_hour < 8 OR
v_current_hour > 18 THEN
            SET v_access_allowed = FALSE;
            SET p_access_result = 'Access denied: Outside allowed hours (weekdays
8:00-18:00)';
        END IF;
    ELSEIF v_user_type = 'teacher' THEN
        -- 教師は平日の7:00-20:00のみアクセス可能
        IF DAYOFWEEK(CURRENT_DATE) IN (1, 7) OR v_current_hour < 7 OR
v_current_hour > 20 THEN
            SET v_access_allowed = FALSE;
            SET p_access_result = 'Access denied: Outside allowed hours (weekdays
7:00-20:00)';
        END IF;
    END IF;

    -- 操作制限の確認
    IF v_access_allowed AND p_operation IN ('UPDATE', 'DELETE') THEN
        IF v_user_type = 'student' THEN
            SET v_access_allowed = FALSE;
            SET p_access_result = 'Access denied: Students cannot modify data';
        END IF;
    END IF;

    IF v_access_allowed THEN
        SET p_access_result = 'Access granted';
    END IF;

    -- アクセスログの記録
    INSERT INTO user_management_log (action, username, host, user_type,
created_by, notes)
    VALUES ('ACCESS_CHECK', USER(), 'system', v_user_type, 'SECURITY_SYSTEM',
            CONCAT('Time-restricted access check: ', p_access_result));

END //

-- 2. 条件付きデータフィルタリング
CREATE PROCEDURE get_filtered_student_data(
    IN p_search_criteria JSON
)
BEGIN
    DECLARE v_user_type VARCHAR(20);
    DECLARE v_associated_id BIGINT;
    DECLARE v_grade_level INT;
    DECLARE v_restrictions JSON;
    DECLARE v_where_clause TEXT DEFAULT '';

    -- ユーザーコンテキストを取得
    CALL get_user_security_context(v_user_type, v_associated_id, v_grade_level,
v_restrictions);

    -- ユーザータイプに応じた基本フィルターの設定
    CASE v_user_type
        WHEN 'student' THEN
            SET v_where_clause = CONCAT('s.student_id = ', v_associated_id);
        WHEN 'teacher' THEN
            SET v_where_clause = CONCAT(
                's.student_id IN (',
                'SELECT DISTINCT sc.student_id FROM student_courses sc ',
                'JOIN courses c ON sc.course_id = c.course_id ',
                'WHERE c.teacher_id = ', v_associated_id, ')'
            );
        WHEN 'grade_supervisor' THEN
            SET v_where_clause = CONCAT(
                'YEAR(CURRENT_DATE) - YEAR(s.admission_date) + 1 = ',
v_grade_level
            );
        WHEN 'office_staff' THEN
            SET v_where_clause = 'YEAR(CURRENT_DATE) - YEAR(s.admission_date) <
3';
        ELSE
            SET v_where_clause = '1=1'; -- 管理者等は制限なし
    END CASE;

    -- 動的クエリの構築と実行
    SET @sql = CONCAT(
        'SELECT s.student_id, s.student_name, s.student_email, ',
        'YEAR(CURRENT_DATE) - YEAR(s.admission_date) + 1 as current_grade ',
        'FROM students s WHERE ', v_where_clause
    );

    -- 追加の検索条件があれば適用
    IF JSON_UNQUOTE(JSON_EXTRACT(p_search_criteria, '$.name_pattern')) IS NOT NULL
THEN
        SET @sql = CONCAT(@sql, ' AND s.student_name LIKE ''%',
                         JSON_UNQUOTE(JSON_EXTRACT(p_search_criteria,
'$.name_pattern')), '%''');
    END IF;

    SET @sql = CONCAT(@sql, ' ORDER BY s.student_id');

    PREPARE stmt FROM @sql;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;

END //

DELIMITER ;
```

## 学校システムでの実践例

### 1. 担任教師システム

```
-- 担任教師に特化したセキュリティシステム

-- 1. 担任クラス管理テーブル
CREATE TABLE homeroom_assignments (
    assignment_id INT AUTO_INCREMENT PRIMARY KEY,
    teacher_id BIGINT NOT NULL,
    class_name VARCHAR(50) NOT NULL,
    grade_level INT NOT NULL,
    academic_year YEAR NOT NULL,
    student_list JSON, -- 担当学生のIDリスト
    is_active BOOLEAN DEFAULT TRUE,
    start_date DATE NOT NULL,
    end_date DATE,

    FOREIGN KEY (teacher_id) REFERENCES teachers(teacher_id),
    UNIQUE KEY unique_active_assignment (teacher_id, academic_year, is_active),
    INDEX idx_grade_year (grade_level, academic_year)
);

-- サンプルデータの挿入
INSERT INTO homeroom_assignments (teacher_id, class_name, grade_level,
academic_year, student_list, start_date) VALUES
(101, '1年A組', 1, 2025, JSON_ARRAY(301, 302, 303), '2025-04-01'),
(102, '2年B組', 2, 2025, JSON_ARRAY(304, 305), '2025-04-01');

-- 2. 担任教師専用の包括的ビュー
CREATE VIEW v_homeroom_teacher_comprehensive AS
SELECT
    s.student_id,
    s.student_name,
    s.student_email,
    s.admission_date,
    ha.class_name,
    ha.grade_level,
    -- 最新の成績情報
    recent_grades.latest_score,
    recent_grades.latest_submission,
    -- 出席状況
    attendance_stats.attendance_rate,
    attendance_stats.absent_days,
    -- 受講講座数
    course_stats.enrolled_courses
FROM students s
JOIN homeroom_assignments ha ON JSON_CONTAINS(ha.student_list, CAST(s.student_id
AS JSON))
LEFT JOIN (
    SELECT
        student_id,
        AVG(score/max_score*100) as latest_score,
        MAX(submission_date) as latest_submission
    FROM grades
    WHERE submission_date >= DATE_SUB(CURRENT_DATE, INTERVAL 30 DAY)
    GROUP BY student_id
) recent_grades ON s.student_id = recent_grades.student_id
LEFT JOIN (
    SELECT
        a.student_id,
        ROUND(COUNT(CASE WHEN a.status = 'present' THEN 1 END) * 100.0 / COUNT(*),
1) as attendance_rate,
        COUNT(CASE WHEN a.status = 'absent' THEN 1 END) as absent_days
    FROM attendance a
    JOIN course_schedule cs ON a.schedule_id = cs.schedule_id
    WHERE cs.schedule_date >= DATE_SUB(CURRENT_DATE, INTERVAL 30 DAY)
    GROUP BY a.student_id
) attendance_stats ON s.student_id = attendance_stats.student_id
LEFT JOIN (
    SELECT student_id, COUNT(DISTINCT course_id) as enrolled_courses
    FROM student_courses
    GROUP BY student_id
) course_stats ON s.student_id = course_stats.student_id
WHERE ha.is_active = TRUE
AND ha.teacher_id = (
    SELECT teacher_id FROM teachers WHERE teacher_name = USER()
    UNION SELECT NULL WHERE USER() = 'root'
);

-- 3. 担任教師用の操作プロシージャ
DELIMITER //

CREATE PROCEDURE homeroom_get_class_summary()
BEGIN
    DECLARE v_teacher_id BIGINT;

    -- 現在のユーザーが教師かどうかを確認
    SELECT teacher_id INTO v_teacher_id
    FROM teachers
    WHERE teacher_name = USER();

    IF v_teacher_id IS NULL AND USER() != 'root' THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Access denied: Not a
registered teacher';
    END IF;

    -- クラス全体のサマリーを表示
    SELECT
        ha.class_name,
        ha.grade_level,
        JSON_LENGTH(ha.student_list) as total_students,
        COUNT(s.student_id) as registered_students,
        ROUND(AVG(recent_grades.avg_score), 1) as class_average_grade,
        ROUND(AVG(attendance_stats.attendance_rate), 1) as class_attendance_rate
    FROM homeroom_assignments ha
    LEFT JOIN students s ON JSON_CONTAINS(ha.student_list, CAST(s.student_id AS
JSON))
    LEFT JOIN (
        SELECT
            student_id,
            AVG(score/max_score*100) as avg_score
        FROM grades
        WHERE submission_date >= DATE_SUB(CURRENT_DATE, INTERVAL 30 DAY)
        GROUP BY student_id
    ) recent_grades ON s.student_id = recent_grades.student_id
    LEFT JOIN (
        SELECT
            a.student_id,
            COUNT(CASE WHEN a.status = 'present' THEN 1 END) * 100.0 / COUNT(*) as
```

```
attendance_rate
        FROM attendance a
        JOIN course_schedule cs ON a.schedule_id = cs.schedule_id
        WHERE cs.schedule_date >= DATE_SUB(CURRENT_DATE, INTERVAL 30 DAY)
        GROUP BY a.student_id
    ) attendance_stats ON s.student_id = attendance_stats.student_id
    WHERE ha.teacher_id = v_teacher_id
    AND ha.is_active = TRUE
    GROUP BY ha.assignment_id, ha.class_name, ha.grade_level;

END //

CREATE PROCEDURE homeroom_get_student_detail(
    IN p_student_id BIGINT
)
BEGIN
    DECLARE v_teacher_id BIGINT;
    DECLARE v_student_authorized BOOLEAN DEFAULT FALSE;

    -- 現在のユーザーが教師かどうかを確認
    SELECT teacher_id INTO v_teacher_id
    FROM teachers
    WHERE teacher_name = USER();

    IF v_teacher_id IS NULL AND USER() != 'root' THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Access denied: Not a
registered teacher';
    END IF;

    -- 担任している学生かどうかを確認
    SELECT COUNT(*) > 0 INTO v_student_authorized
    FROM homeroom_assignments ha
    WHERE ha.teacher_id = v_teacher_id
    AND ha.is_active = TRUE
    AND JSON_CONTAINS(ha.student_list, CAST(p_student_id AS JSON));

    IF NOT v_student_authorized AND USER() != 'root' THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Access denied: Student not in
your homeroom class';
    END IF;

    -- 学生の詳細情報を取得
    SELECT * FROM v_homeroom_teacher_comprehensive
    WHERE student_id = p_student_id;

END //

DELIMITER ;
```

### 2. 保護者アクセスシステム

```
-- 保護者向けの限定的アクセスシステム

-- 1. 保護者-学生関係テーブル
CREATE TABLE parent_student_relationships (
    relationship_id INT AUTO_INCREMENT PRIMARY KEY,
    parent_username VARCHAR(100) NOT NULL,
    student_id BIGINT NOT NULL,
    relationship_type ENUM('father', 'mother', 'guardian', 'emergency_contact')
NOT NULL,
    is_primary_contact BOOLEAN DEFAULT FALSE,
    access_level ENUM('full', 'grades_only', 'attendance_only', 'emergency_only')
DEFAULT 'full',
    authorized_by VARCHAR(100), -- 承認者
    authorized_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    expires_at TIMESTAMP NULL,
    is_active BOOLEAN DEFAULT TRUE,

    FOREIGN KEY (student_id) REFERENCES students(student_id),
    UNIQUE KEY unique_primary_contact (student_id, relationship_type,
is_primary_contact),
    INDEX idx_parent_username (parent_username),
    INDEX idx_student_id (student_id)
);

-- サンプル保護者関係データ
INSERT INTO parent_student_relationships (parent_username, student_id,
relationship_type, is_primary_contact, authorized_by) VALUES
('parent_yamada@localhost', 301, 'father', TRUE, 'office_staff@localhost'),
('parent_tanaka@localhost', 302, 'mother', TRUE, 'office_staff@localhost'),
('parent_suzuki@localhost', 301, 'mother', FALSE, 'office_staff@localhost');

-- 2. 保護者専用セキュリティビュー
CREATE VIEW v_parent_student_info AS
SELECT
    s.student_id,
    s.student_name,
    s.admission_date,
    YEAR(CURRENT_DATE) - YEAR(s.admission_date) + 1 as current_grade,
    psr.relationship_type,
    psr.access_level,
    -- 最新の成績情報（アクセスレベルに応じて）
    CASE
        WHEN psr.access_level IN ('full', 'grades_only') THEN
recent_grades.latest_average
        ELSE NULL
    END as latest_grade_average,
    -- 出席状況（アクセスレベルに応じて）
    CASE
        WHEN psr.access_level IN ('full', 'attendance_only') THEN
attendance_stats.attendance_rate
        ELSE NULL
    END as attendance_rate,
    CASE
```

```
        WHEN psr.access_level IN ('full', 'attendance_only') THEN
attendance_stats.recent_absences
        ELSE NULL
    END as recent_absences
FROM students s
JOIN parent_student_relationships psr ON s.student_id = psr.student_id
LEFT JOIN (
    SELECT
        student_id,
        ROUND(AVG(score/max_score*100), 1) as latest_average
    FROM grades
    WHERE submission_date >= DATE_SUB(CURRENT_DATE, INTERVAL 30 DAY)
    GROUP BY student_id
) recent_grades ON s.student_id = recent_grades.student_id
LEFT JOIN (
    SELECT
        a.student_id,
        ROUND(COUNT(CASE WHEN a.status = 'present' THEN 1 END) * 100.0 / COUNT(*),
1) as attendance_rate,
        COUNT(CASE WHEN a.status = 'absent' THEN 1 END) as recent_absences
    FROM attendance a
    JOIN course_schedule cs ON a.schedule_id = cs.schedule_id
    WHERE cs.schedule_date >= DATE_SUB(CURRENT_DATE, INTERVAL 30 DAY)
    GROUP BY a.student_id
) attendance_stats ON s.student_id = attendance_stats.student_id
WHERE psr.parent_username = USER()
AND psr.is_active = TRUE
AND (psr.expires_at IS NULL OR psr.expires_at > NOW());

-- 3. 保護者用アクセスプロシージャ
DELIMITER //

CREATE PROCEDURE parent_get_child_grades(
    IN p_student_id BIGINT
)
BEGIN
    DECLARE v_access_authorized BOOLEAN DEFAULT FALSE;
    DECLARE v_access_level VARCHAR(20);

    -- 保護者の子供へのアクセス権限を確認
    SELECT
        COUNT(*) > 0,
        MAX(access_level)
    INTO v_access_authorized, v_access_level
    FROM parent_student_relationships psr
    WHERE psr.parent_username = USER()
    AND psr.student_id = p_student_id
    AND psr.is_active = TRUE
    AND (psr.expires_at IS NULL OR psr.expires_at > NOW())
    AND psr.access_level IN ('full', 'grades_only');

    IF NOT v_access_authorized THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Access denied: No
authorization to view grades';
```

```
    END IF;

    -- 成績情報の取得（過去3ヶ月分）
    SELECT
        g.course_id,
        c.course_name,
        g.grade_type,
        g.score,
        g.max_score,
        ROUND((g.score / g.max_score) * 100, 1) as percentage,
        g.submission_date
    FROM grades g
    JOIN courses c ON g.course_id = c.course_id
    WHERE g.student_id = p_student_id
    AND g.submission_date >= DATE_SUB(CURRENT_DATE, INTERVAL 90 DAY)
    ORDER BY g.submission_date DESC, c.course_name;

END //

CREATE PROCEDURE parent_get_attendance_summary(
    IN p_student_id BIGINT
)
BEGIN
    DECLARE v_access_authorized BOOLEAN DEFAULT FALSE;

    -- 保護者の子供へのアクセス権限を確認
    SELECT COUNT(*) > 0 INTO v_access_authorized
    FROM parent_student_relationships psr
    WHERE psr.parent_username = USER()
    AND psr.student_id = p_student_id
    AND psr.is_active = TRUE
    AND (psr.expires_at IS NULL OR psr.expires_at > NOW())
    AND psr.access_level IN ('full', 'attendance_only');

    IF NOT v_access_authorized THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Access denied: No
authorization to view attendance';
    END IF;

    -- 出席サマリーの取得（過去1ヶ月分）
    SELECT
        cs.schedule_date,
        c.course_name,
        cp.start_time,
        cp.end_time,
        a.status,
        CASE a.status
            WHEN 'present' THEN '出席'
            WHEN 'absent' THEN '欠席'
            WHEN 'late' THEN '遅刻'
            ELSE '不明'
        END as status_japanese
    FROM attendance a
    JOIN course_schedule cs ON a.schedule_id = cs.schedule_id
    JOIN courses c ON cs.course_id = c.course_id
    JOIN class_periods cp ON cs.period_id = cp.period_id
    WHERE a.student_id = p_student_id
    AND cs.schedule_date >= DATE_SUB(CURRENT_DATE, INTERVAL 30 DAY)
    ORDER BY cs.schedule_date DESC, cp.start_time;

END //

DELIMITER ;
```

## パフォーマンスとセキュリティの両立

### 1. インデックス戦略

```
-- セキュリティビューのパフォーマンス最適化

-- 1. セキュリティ制御用のインデックス作成
-- ユーザーコンテキストテーブル用
CREATE INDEX idx_user_context_username ON user_security_context (username);
CREATE INDEX idx_user_context_type_id ON user_security_context (user_type,
associated_id);

-- 学生-講座関係の高速検索用
CREATE INDEX idx_student_courses_student ON student_courses (student_id,
course_id);
CREATE INDEX idx_courses_teacher ON courses (teacher_id, course_id);

-- 成績テーブルの高速フィルタリング用
CREATE INDEX idx_grades_student_date ON grades (student_id, submission_date);
CREATE INDEX idx_grades_course_date ON grades (course_id, submission_date);

-- 出席テーブルの高速集計用
CREATE INDEX idx_attendance_student_status ON attendance (student_id, status);

-- 担任関係の高速検索用
CREATE INDEX idx_homeroom_teacher_active ON homeroom_assignments (teacher_id,
is_active);

-- 保護者関係の高速検索用
CREATE INDEX idx_parent_relationships_active ON parent_student_relationships
(parent_username, is_active);

-- 2. セキュリティビューのパフォーマンス分析
DELIMITER //

CREATE PROCEDURE analyze_security_view_performance()
BEGIN
    -- 各セキュリティビューの実行計画を分析
    SELECT 'Performance Analysis for Security Views' as analysis_title;

    -- 教師学生ビューの分析
    EXPLAIN FORMAT=JSON
    SELECT * FROM v_teacher_students LIMIT 1;

    -- 学生専用ビューの分析
    EXPLAIN FORMAT=JSON
    SELECT * FROM v_student_own_grades LIMIT 1;

    -- 担任ビューの分析
    EXPLAIN FORMAT=JSON
    SELECT * FROM v_homeroom_teacher_comprehensive LIMIT 1;

END //

DELIMITER ;

-- 3. キャッシュ機能付きセキュリティプロシージャ
CREATE TABLE security_cache (
    cache_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    cache_key VARCHAR(255) NOT NULL,
    cache_data JSON NOT NULL,
    created_by VARCHAR(100) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    expires_at TIMESTAMP NOT NULL,

    UNIQUE KEY unique_cache_key (cache_key),
    INDEX idx_expires_at (expires_at)
);

DELIMITER //

CREATE PROCEDURE cached_secure_get_student_grades(
    IN p_student_id BIGINT,
    IN p_cache_minutes INT DEFAULT 15
)
BEGIN
    DECLARE v_cache_key VARCHAR(255);
    DECLARE v_cached_data JSON;
    DECLARE v_cache_expires TIMESTAMP;

    SET v_cache_key = CONCAT('student_grades_', p_student_id, '_', USER());

    -- キャッシュの確認
    SELECT cache_data, expires_at INTO v_cached_data, v_cache_expires
    FROM security_cache
    WHERE cache_key = v_cache_key
    AND expires_at > NOW();

    IF v_cached_data IS NOT NULL THEN
        -- キャッシュからデータを返す
        SELECT
            JSON_UNQUOTE(JSON_EXTRACT(item, '$.course_name')) as course_name,
            JSON_UNQUOTE(JSON_EXTRACT(item, '$.grade_type')) as grade_type,
            CAST(JSON_UNQUOTE(JSON_EXTRACT(item, '$.score')) AS DECIMAL(5,2)) as
score,
            JSON_UNQUOTE(JSON_EXTRACT(item, '$.submission_date')) as
submission_date
        FROM JSON_TABLE(v_cached_data, '$[*]' COLUMNS (
            item JSON PATH '$'
        )) as cached_grades;
    ELSE
        -- データベースから取得してキャッシュに保存
        DROP TEMPORARY TABLE IF EXISTS temp_grades;
        CREATE TEMPORARY TABLE temp_grades AS
        SELECT
            c.course_name,
            g.grade_type,
            g.score,
            g.submission_date
        FROM grades g
        JOIN courses c ON g.course_id = c.course_id
        WHERE g.student_id = p_student_id
        AND (
            USER() = 'root'
            OR CONCAT('student_', g.student_id) = SUBSTRING_INDEX(USER(), '@', 1)
            OR EXISTS (
                SELECT 1 FROM courses c2
                JOIN teachers t ON c2.teacher_id = t.teacher_id
                WHERE c2.course_id = g.course_id
                AND t.teacher_name = USER()
            )
        );

        -- 結果をキャッシュに保存
        SET v_cached_data = (
            SELECT JSON_ARRAYAGG(
                JSON_OBJECT(
                    'course_name', course_name,
                    'grade_type', grade_type,
                    'score', score,
                    'submission_date', submission_date
                )
            )
            FROM temp_grades
        );

        INSERT INTO security_cache (cache_key, cache_data, created_by, expires_at)
        VALUES (v_cache_key, v_cached_data, USER(), DATE_ADD(NOW(), INTERVAL
p_cache_minutes MINUTE))
        ON DUPLICATE KEY UPDATE
            cache_data = v_cached_data,
            expires_at = DATE_ADD(NOW(), INTERVAL p_cache_minutes MINUTE);

        -- 結果を返す
        SELECT * FROM temp_grades;
    END IF;

END //
```

```
DELIMITER ;
```

## 2. セキュリティとパフォーマンスの監視

```
-- セキュリティアクセスの監視システム

-- 1. アクセスログテーブル
CREATE TABLE security_access_log (
    log_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(100) NOT NULL,
    access_type ENUM('VIEW', 'PROCEDURE', 'DIRECT') NOT NULL,
    target_object VARCHAR(100) NOT NULL,
    target_student_id BIGINT,
    access_granted BOOLEAN NOT NULL,
    execution_time_ms INT,
    rows_accessed INT,
    access_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    user_agent VARCHAR(500),
    ip_address VARCHAR(45),

    INDEX idx_username_timestamp (username, access_timestamp),
    INDEX idx_target_student (target_student_id, access_timestamp),
    INDEX idx_access_granted (access_granted, access_timestamp)
);

-- 2. アクセス監視プロシージャ
DELIMITER //

CREATE PROCEDURE log_security_access(
    IN p_access_type ENUM('VIEW', 'PROCEDURE', 'DIRECT'),
    IN p_target_object VARCHAR(100),
    IN p_target_student_id BIGINT,
    IN p_access_granted BOOLEAN,
    IN p_execution_time_ms INT,
    IN p_rows_accessed INT
)
BEGIN
    INSERT INTO security_access_log
        (username, access_type, target_object, target_student_id,
         access_granted, execution_time_ms, rows_accessed)
    VALUES
        (USER(), p_access_type, p_target_object, p_target_student_id,
         p_access_granted, p_execution_time_ms, p_rows_accessed);
END //

-- 3. セキュリティ異常検知
CREATE PROCEDURE detect_security_anomalies()
BEGIN
    -- 異常なアクセスパターンの検知
    -- 1. 短時間での大量アクセス
    SELECT 'High Frequency Access Detected' as alert_type,
           username,
           COUNT(*) as access_count,
           MIN(access_timestamp) as start_time,
           MAX(access_timestamp) as end_time
    FROM security_access_log
    WHERE access_timestamp >= DATE_SUB(NOW(), INTERVAL 5 MINUTE)
    GROUP BY username
    HAVING COUNT(*) > 100;

    -- 2. 通常と異なる時間帯のアクセス
    SELECT 'Off-Hours Access Detected' as alert_type,
           username,
           target_object,
           access_timestamp,
           HOUR(access_timestamp) as access_hour
    FROM security_access_log
    WHERE access_timestamp >= DATE_SUB(NOW(), INTERVAL 1 HOUR)
    AND (HOUR(access_timestamp) < 6 OR HOUR(access_timestamp) > 22)
    AND username NOT LIKE '%admin%'
    AND username != 'root';

    -- 3. アクセス拒否の集中
    SELECT 'Access Denial Spike Detected' as alert_type,
           username,
           COUNT(*) as denial_count,
           GROUP_CONCAT(DISTINCT target_object) as attempted_objects
    FROM security_access_log
    WHERE access_timestamp >= DATE_SUB(NOW(), INTERVAL 15 MINUTE)
    AND access_granted = FALSE
    GROUP BY username
    HAVING COUNT(*) > 10;

    -- 4. 学生IDの広範囲スキャン
    SELECT 'Student ID Scanning Detected' as alert_type,
           username,
           COUNT(DISTINCT target_student_id) as unique_students_accessed,
           COUNT(*) as total_attempts
    FROM security_access_log
    WHERE access_timestamp >= DATE_SUB(NOW(), INTERVAL 10 MINUTE)
    AND target_student_id IS NOT NULL
    GROUP BY username
    HAVING COUNT(DISTINCT target_student_id) > 20;

END //

DELIMITER ;

-- 4. パフォーマンス最適化レポート
CREATE PROCEDURE generate_security_performance_report()
BEGIN
    -- セキュリティ機能のパフォーマンス分析
    SELECT 'Security Performance Summary' as report_section;

    -- 平均実行時間
    SELECT
        target_object,
        access_type,
        COUNT(*) as access_count,
        AVG(execution_time_ms) as avg_execution_time,
        MAX(execution_time_ms) as max_execution_time,
        AVG(rows_accessed) as avg_rows_accessed
    FROM security_access_log
    WHERE access_timestamp >= DATE_SUB(NOW(), INTERVAL 24 HOUR)
    AND access_granted = TRUE
    GROUP BY target_object, access_type
    ORDER BY avg_execution_time DESC;

    -- キャッシュ効率
    SELECT
        'Cache Performance' as metric,
        COUNT(CASE WHEN cache_key IS NOT NULL THEN 1 END) as cache_hits,
        COUNT(*) as total_requests,
        ROUND(COUNT(CASE WHEN cache_key IS NOT NULL THEN 1 END) * 100.0 /
COUNT(*), 2) as cache_hit_rate
    FROM security_access_log sal
    LEFT JOIN security_cache sc ON CONCAT('student_grades_',
sal.target_student_id, '_', sal.username) = sc.cache_key
    WHERE sal.access_timestamp >= DATE_SUB(NOW(), INTERVAL 1 HOUR);

END //

DELIMITER ;
```

## 練習問題

### 問題41-1：基本的なセキュリティビューの作成

学校システムに以下の要件でセキュリティビューを作成してください：

1. `v_counselor_student_info`：カウンセラー用ビュー
    - カウンセラーは全学生の基本情報を閲覧可能
    - ただし、個人的な連絡先情報（メールアドレス）は除外
    - 現在の学年と入学年度を表示
2. `v_librarian_student_basic`：図書館司書用ビュー
    - 図書館司書は学生の基本情報のみ閲覧可能
    - 学籍番号、氏名、学年のみ表示
    - 休学・退学者は除外
3. 各ビューに適切な権限を設定し、動作確認を行ってください

### 問題41-2：動的セキュリティプロシージャの実装

以下の要件でセキュリティプロシージャを作成してください：

1. `secure_get_class_roster(class_identifier)`
   - クラス担任のみが担当クラスの名簿を取得可能
   - 学生の基本情報、最新の出席率、平均成績を含む
   - 非担任からのアクセスは拒否
2. `secure_update_attendance(student_id, schedule_id, status)`
   - 教師のみが担当講座の出席を更新可能
   - 担当外の講座への更新は拒否
   - 更新履歴をログに記録
3. エラーハンドリングとアクセスログ機能を含めてください

### 問題41-3：多階層セキュリティシステム

以下の階層的なアクセス制御システムを実装してください：

1. **レベル1（学生）**：自分の情報のみアクセス
2. **レベル2（教師）**：担当講座の学生情報にアクセス
3. **レベル3（学年主任）**：担当学年の全学生情報にアクセス
4. **レベル4（教務主任）**：全学年の学生情報にアクセス
5. **レベル5（校長）**：全ての情報にアクセス

各レベルで適切なビューとプロシージャを作成し、権限設定を行ってください。

### 問題41-4：時間制限付きアクセス制御

以下の時間制限機能を実装してください：

1. **平常時間制限**：
   - 学生：平日8:00-18:00のみアクセス可能
   - 教師：平日7:00-20:00のみアクセス可能
   - 管理職：時間制限なし
2. **定期試験期間制限**：
   - 学生の成績閲覧を一時的に制限
   - 教師の成績入力期間を限定
3. **緊急時アクセス**：
   - 緊急連絡先登録者は24時間アクセス可能
4. 時間制限の管理テーブルと制御プロシージャを作成してください

### 問題41-5：保護者アクセスシステムの拡張

以下の機能を持つ保護者システムを実装してください：

1. **段階的アクセス権限**：
   - フルアクセス（成績・出席・健康情報）
   - 制限アクセス（出席情報のみ）
   - 緊急時のみ（緊急連絡情報のみ）
2. **通知機能**：
   - 子供の出席状況に問題がある場合の自動通知
   - 成績変動の通知
3. **アクセス履歴管理**：

    - 保護者のアクセス履歴を記録
    - 異常なアクセスパターンの検知
4. **承認ワークフロー**：
    - 保護者アカウントの新規作成・変更に承認プロセス

### 問題41-6：総合セキュリティシステム

学校全体の包括的な行レベルセキュリティシステムを設計・実装してください：

1. **ユーザー管理**：
    - 動的なユーザーコンテキスト管理
    - 役職変更時の自動権限更新
    - 一時的な権限委譲機能
2. **データアクセス制御**：
    - 全テーブルに対する行レベル制御
    - 動的なフィルタリング条件
    - カスタムアクセスポリシー
3. **監査・ログ機能**：
    - 全アクセスの記録
    - 異常検知機能
    - コンプライアンスレポート生成
4. **パフォーマンス最適化**：
    - セキュリティキャッシュ機能
    - インデックス最適化
    - 実行計画の分析・改善

## 解答

### 解答41-1

```sql
-- 基本的なセキュリティビューの作成

-- 1. カウンセラー用ビューの作成
CREATE VIEW v_counselor_student_info AS
SELECT
    s.student_id,
    s.student_name,
    s.admission_date,
    YEAR(CURRENT_DATE) - YEAR(s.admission_date) + 1 as current_grade,
    YEAR(s.admission_date) as admission_year,
    -- メールアドレスは除外してセキュリティを確保
    'カウンセラー権限では非表示' as email_status,
    -- 在籍状況の判定
    CASE
        WHEN YEAR(CURRENT_DATE) - YEAR(s.admission_date) >= 3 THEN '卒業予定'
        WHEN YEAR(CURRENT_DATE) - YEAR(s.admission_date) >= 2 THEN '3年生'
        WHEN YEAR(CURRENT_DATE) - YEAR(s.admission_date) >= 1 THEN '2年生'
        ELSE '1年生'
    END as student_status
```

```sql
FROM students s
WHERE (
    USER() LIKE '%counselor%'
    OR USER() LIKE '%カウンセラー%'
    OR USER() = 'root'
    OR USER() LIKE '%principal%'
);

-- 2. 図書館司書用ビューの作成
CREATE VIEW v_librarian_student_basic AS
SELECT
    s.student_id,
    s.student_name,
    YEAR(CURRENT_DATE) - YEAR(s.admission_date) + 1 as current_grade
FROM students s
WHERE (
    -- 在籍中の学生のみ（休学・退学者は除外）
    YEAR(CURRENT_DATE) - YEAR(s.admission_date) < 3
    AND (
        USER() LIKE '%librarian%'
        OR USER() LIKE '%図書%'
        OR USER() = 'root'
        OR USER() LIKE '%principal%'
    )
);

-- 3. 権限設定

-- カウンセラーロールが存在しない場合は作成
CREATE ROLE IF NOT EXISTS 'counselor_role';
CREATE ROLE IF NOT EXISTS 'librarian_role';

-- カウンセラー用ビューの権限付与
GRANT SELECT ON school_db.v_counselor_student_info TO 'counselor_role';

-- 図書館司書用ビューの権限付与
GRANT SELECT ON school_db.v_librarian_student_basic TO 'librarian_role';

-- 4. テスト用ユーザーの作成（実際の運用では管理者が実行）
-- CREATE USER 'counselor_test'@'localhost' IDENTIFIED BY 'Counselor2025!';
-- CREATE USER 'librarian_test'@'localhost' IDENTIFIED BY 'Librarian2025!';
-- GRANT 'counselor_role' TO 'counselor_test'@'localhost';
-- GRANT 'librarian_role' TO 'librarian_test'@'localhost';
-- ALTER USER 'counselor_test'@'localhost' DEFAULT ROLE ALL;
-- ALTER USER 'librarian_test'@'localhost' DEFAULT ROLE ALL;

-- 5. 動作確認用クエリ

-- ビューの構造確認
DESCRIBE v_counselor_student_info;
DESCRIBE v_librarian_student_basic;

-- データ確認（管理者として）
SELECT 'Counselor View Sample:' as test_type;
SELECT * FROM v_counselor_student_info LIMIT 5;

SELECT 'Librarian View Sample:' as test_type;
SELECT * FROM v_librarian_student_basic LIMIT 5;

-- 権限確認
SELECT
    table_name,
    grantee,
    privilege_type
FROM information_schema.table_privileges
WHERE table_schema = 'school_db'
AND table_name IN ('v_counselor_student_info', 'v_librarian_student_basic')
ORDER BY table_name, grantee;

-- ビューが正しくフィルタリングしているかテスト
SELECT
    'View Security Test' as test_type,
    COUNT(*) as total_students_in_base_table
FROM students;

SELECT
    'Counselor View Count' as test_type,
    COUNT(*) as visible_students
FROM v_counselor_student_info;

SELECT
    'Librarian View Count' as test_type,
    COUNT(*) as visible_students
FROM v_librarian_student_basic;

-- カウンセラービューでメール情報が除外されているか確認
SELECT DISTINCT email_status FROM v_counselor_student_info;
```

## 解答41-2

```
-- 動的セキュリティプロシージャの実装

-- 1. 前提テーブルの確認・作成
-- クラス情報を管理するテーブル（存在しない場合）
CREATE TABLE IF NOT EXISTS class_assignments (
    assignment_id INT AUTO_INCREMENT PRIMARY KEY,
    teacher_id BIGINT NOT NULL,
    class_name VARCHAR(50) NOT NULL,
    grade_level INT NOT NULL,
    academic_year YEAR NOT NULL,
    is_active BOOLEAN DEFAULT TRUE,

    FOREIGN KEY (teacher_id) REFERENCES teachers(teacher_id),
    INDEX idx_teacher_active (teacher_id, is_active)
);
```

```
-- サンプルクラス割当データ
INSERT IGNORE INTO class_assignments (teacher_id, class_name, grade_level,
academic_year) VALUES
(101, '1年A組', 1, 2025),
(102, '2年B組', 2, 2025);

-- アクセスログテーブル（存在しない場合）
CREATE TABLE IF NOT EXISTS security_procedure_log (
    log_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    procedure_name VARCHAR(100) NOT NULL,
    username VARCHAR(100) NOT NULL,
    parameters JSON,
    access_granted BOOLEAN NOT NULL,
    execution_time_ms INT,
    error_message TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    INDEX idx_procedure_user (procedure_name, username),
    INDEX idx_created_at (created_at)
);

-- 2. クラス名簿取得プロシージャ
DELIMITER //

CREATE PROCEDURE secure_get_class_roster(
    IN p_class_identifier VARCHAR(50)
)
BEGIN
    DECLARE v_start_time TIMESTAMP DEFAULT NOW(6);
    DECLARE v_teacher_id BIGINT;
    DECLARE v_class_authorized BOOLEAN DEFAULT FALSE;
    DECLARE v_execution_time INT;
    DECLARE v_error_msg TEXT DEFAULT NULL;

    -- 現在のユーザーが教師かどうかを確認
    SELECT teacher_id INTO v_teacher_id
    FROM teachers
    WHERE teacher_name = USER()
    LIMIT 1;

    IF v_teacher_id IS NULL AND USER() != 'root' THEN
        SET v_error_msg = 'Access denied: User is not a registered teacher';
        SET v_execution_time = TIMESTAMPDIFF(MICROSECOND, v_start_time, NOW(6)) /
1000;

        INSERT INTO security_procedure_log
            (procedure_name, username, parameters, access_granted,
execution_time_ms, error_message)
        VALUES
            ('secure_get_class_roster', USER(), JSON_OBJECT('class_identifier',
p_class_identifier),
             FALSE, v_execution_time, v_error_msg);
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_error_msg;
    END IF;

    -- 担当クラスかどうかを確認
    SELECT COUNT(*) > 0 INTO v_class_authorized
    FROM class_assignments ca
    WHERE ca.teacher_id = v_teacher_id
    AND ca.class_name = p_class_identifier
    AND ca.is_active = TRUE;

    IF NOT v_class_authorized AND USER() != 'root' THEN
        SET v_error_msg = CONCAT('Access denied: Not authorized for class ',
p_class_identifier);
        SET v_execution_time = TIMESTAMPDIFF(MICROSECOND, v_start_time, NOW(6)) /
1000;

        INSERT INTO security_procedure_log
            (procedure_name, username, parameters, access_granted,
execution_time_ms, error_message)
        VALUES
            ('secure_get_class_roster', USER(), JSON_OBJECT('class_identifier',
p_class_identifier),
             FALSE, v_execution_time, v_error_msg);

        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_error_msg;
    END IF;

    -- クラスの名簿と統計情報を取得
    SELECT
        s.student_id,
        s.student_name,
        s.student_email,
        s.admission_date,
        ca.class_name,
        ca.grade_level,
        -- 最新の出席率（過去30日）
        COALESCE(attendance_stats.attendance_rate, 0) as attendance_rate_30days,
        COALESCE(attendance_stats.total_classes, 0) as total_classes_30days,
        -- 平均成績（過去30日）
        COALESCE(grade_stats.average_score, 0) as average_score_30days,
        COALESCE(grade_stats.grade_count, 0) as grade_count_30days
    FROM students s
    CROSS JOIN class_assignments ca
    LEFT JOIN (
        SELECT
            a.student_id,
            ROUND(COUNT(CASE WHEN a.status = 'present' THEN 1 END) * 100.0 /
COUNT(*), 1) as attendance_rate,
            COUNT(*) as total_classes
        FROM attendance a
        JOIN course_schedule cs ON a.schedule_id = cs.schedule_id
        WHERE cs.schedule_date >= DATE_SUB(CURRENT_DATE, INTERVAL 30 DAY)
        GROUP BY a.student_id
    ) attendance_stats ON s.student_id = attendance_stats.student_id
    LEFT JOIN (
        SELECT
            g.student_id,
            ROUND(AVG(g.score / g.max_score * 100), 1) as average_score,
            COUNT(*) as grade_count
        FROM grades g
        WHERE g.submission_date >= DATE_SUB(CURRENT_DATE, INTERVAL 30 DAY)
        GROUP BY g.student_id
    ) grade_stats ON s.student_id = grade_stats.student_id
    WHERE ca.teacher_id = v_teacher_id
    AND ca.class_name = p_class_identifier
    AND ca.is_active = TRUE
    AND YEAR(CURRENT_DATE) - YEAR(s.admission_date) + 1 = ca.grade_level
    ORDER BY s.student_name;

    -- 成功ログの記録
    SET v_execution_time = TIMESTAMPDIFF(MICROSECOND, v_start_time, NOW(6)) /
1000;
    INSERT INTO security_procedure_log
        (procedure_name, username, parameters, access_granted, execution_time_ms)
    VALUES
        ('secure_get_class_roster', USER(), JSON_OBJECT('class_identifier',
p_class_identifier),
         TRUE, v_execution_time);

END //

-- 3. 出席情報安全更新プロシージャ
CREATE PROCEDURE secure_update_attendance(
    IN p_student_id BIGINT,
    IN p_schedule_id BIGINT,
    IN p_status ENUM('present', 'absent', 'late')
)
BEGIN
    DECLARE v_start_time TIMESTAMP DEFAULT NOW(6);
    DECLARE v_teacher_id BIGINT;
    DECLARE v_course_authorized BOOLEAN DEFAULT FALSE;
    DECLARE v_execution_time INT;
    DECLARE v_error_msg TEXT DEFAULT NULL;
    DECLARE v_course_id VARCHAR(16);

    -- 現在のユーザーが教師かどうかを確認
    SELECT teacher_id INTO v_teacher_id
    FROM teachers
    WHERE teacher_name = USER()
    LIMIT 1;

    IF v_teacher_id IS NULL AND USER() != 'root' THEN
        SET v_error_msg = 'Access denied: User is not a registered teacher';
        SET v_execution_time = TIMESTAMPDIFF(MICROSECOND, v_start_time, NOW(6)) /
1000;

        INSERT INTO security_procedure_log
            (procedure_name, username, parameters, access_granted,
execution_time_ms, error_message)
        VALUES
            ('secure_update_attendance', USER(),
             JSON_OBJECT('student_id', p_student_id, 'schedule_id', p_schedule_id,
'status', p_status),
             FALSE, v_execution_time, v_error_msg);

        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_error_msg;
    END IF;

    -- スケジュールIDから講座IDを取得し、担当教師かどうかを確認
    SELECT cs.course_id INTO v_course_id
    FROM course_schedule cs
    WHERE cs.schedule_id = p_schedule_id;

    IF v_course_id IS NULL THEN
        SET v_error_msg = 'Invalid schedule ID';
        SET v_execution_time = TIMESTAMPDIFF(MICROSECOND, v_start_time, NOW(6)) /
1000;

        INSERT INTO security_procedure_log
            (procedure_name, username, parameters, access_granted,
execution_time_ms, error_message)
        VALUES
            ('secure_update_attendance', USER(),
             JSON_OBJECT('student_id', p_student_id, 'schedule_id', p_schedule_id,
'status', p_status),
             FALSE, v_execution_time, v_error_msg);

        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_error_msg;
    END IF;

    -- 担当講座かどうかを確認
    SELECT COUNT(*) > 0 INTO v_course_authorized
    FROM courses c
    WHERE c.course_id = v_course_id
    AND c.teacher_id = v_teacher_id;

    IF NOT v_course_authorized AND USER() != 'root' THEN
        SET v_error_msg = CONCAT('Access denied: Not authorized for course ',
v_course_id);
        SET v_execution_time = TIMESTAMPDIFF(MICROSECOND, v_start_time, NOW(6)) /
1000;

        INSERT INTO security_procedure_log
            (procedure_name, username, parameters, access_granted,
execution_time_ms, error_message)
        VALUES
            ('secure_update_attendance', USER(),
             JSON_OBJECT('student_id', p_student_id, 'schedule_id', p_schedule_id,
'status', p_status),
             FALSE, v_execution_time, v_error_msg);

        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_error_msg;
    END IF;

    -- 出席情報の更新
    INSERT INTO attendance (schedule_id, student_id, status)
    VALUES (p_schedule_id, p_student_id, p_status)
    ON DUPLICATE KEY UPDATE
        status = p_status;

    -- 成功ログの記録
    SET v_execution_time = TIMESTAMPDIFF(MICROSECOND, v_start_time, NOW(6)) /
1000;
    INSERT INTO security_procedure_log
        (procedure_name, username, parameters, access_granted, execution_time_ms)
    VALUES
        ('secure_update_attendance', USER(),
         JSON_OBJECT('student_id', p_student_id, 'schedule_id', p_schedule_id,
'status', p_status),
         TRUE, v_execution_time);

    -- 更新履歴を別途記録
    INSERT INTO user_management_log (action, username, host, user_type,
created_by, notes)
    VALUES ('UPDATE', CONCAT('student_', p_student_id), 'system',
'attendance_update', USER(),
            CONCAT('Updated attendance for schedule ', p_schedule_id, ' to: ',
p_status));

    SELECT CONCAT('Attendance updated successfully for student ', p_student_id) as
result;

END //

DELIMITER ;

-- 4. テスト実行（管理者として）

-- プロシージャの動作テスト
-- CALL secure_get_class_roster('1年A組');
-- CALL secure_update_attendance(301, 1, 'present');

-- ログの確認
SELECT
    procedure_name,
    username,
    access_granted,
    execution_time_ms,
    created_at,
    error_message
FROM security_procedure_log
ORDER BY created_at DESC
LIMIT 10;

-- エラーケースのテスト用クエリ（実際には適切なユーザーで実行）
-- 存在しないクラスでのテスト
```

```sql
-- CALL secure_get_class_roster('存在しないクラス');

-- 権限のないスケジュールでの出席更新テスト
-- CALL secure_update_attendance(999, 999, 'present');
```

## 解答41-3

```sql
-- 多階層セキュリティシステムの実装

-- 1. ユーザー階層定義テーブル
CREATE TABLE user_access_levels (
    level_id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(100) NOT NULL,
    access_level INT NOT NULL, -- 1=学生, 2=教師, 3=学年主任, 4=教務主任, 5=校長
    level_name VARCHAR(50) NOT NULL,
    associated_id BIGINT, -- 学生ID、教師ID等
    grade_restriction INT, -- 学年主任の場合の担当学年
    department_restriction VARCHAR(100), -- 部署制限
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    UNIQUE KEY unique_username (username),
    INDEX idx_access_level (access_level),
    INDEX idx_active (is_active)
);

-- サンプル階層データ
INSERT INTO user_access_levels (username, access_level, level_name, associated_id,
grade_restriction) VALUES
('student_301@localhost', 1, '学生', 301, NULL),
('student_302@localhost', 1, '学生', 302, NULL),
('teacher_tanaka@localhost', 2, '教師', 101, NULL),
('teacher_sato@localhost', 2, '教師', 102, NULL),
('grade_head_1@localhost', 3, '学年主任', 103, 1),
('grade_head_2@localhost', 3, '学年主任', 104, 2),
('academic_director@localhost', 4, '教務主任', 105, NULL),
('principal@localhost', 5, '校長', 106, NULL);

-- 2. 階層別セキュリティビュー

-- レベル1: 学生専用ビュー（自分の情報のみ）
CREATE VIEW v_level1_student_own AS
SELECT
    s.student_id,
    s.student_name,
    s.student_email,
    s.admission_date,
    YEAR(CURRENT_DATE) - YEAR(s.admission_date) + 1 as current_grade,
    'レベル1アクセス' as access_level_info
FROM students s
JOIN user_access_levels ual ON ual.associated_id = s.student_id
```

```sql
WHERE ual.username = USER()
AND ual.access_level = 1
AND ual.is_active = TRUE;

-- レベル2: 教師ビュー（担当講座の学生）
CREATE VIEW v_level2_teacher_students AS
SELECT
    s.student_id,
    s.student_name,
    s.student_email,
    s.admission_date,
    YEAR(CURRENT_DATE) - YEAR(s.admission_date) + 1 as current_grade,
    c.course_id,
    c.course_name,
    'レベル2アクセス' as access_level_info
FROM students s
JOIN student_courses sc ON s.student_id = sc.student_id
JOIN courses c ON sc.course_id = c.course_id
JOIN user_access_levels ual ON ual.associated_id = c.teacher_id
WHERE ual.username = USER()
AND ual.access_level = 2
AND ual.is_active = TRUE;

-- レベル3: 学年主任ビュー（担当学年の学生）
CREATE VIEW v_level3_grade_supervisor AS
SELECT
    s.student_id,
    s.student_name,
    s.student_email,
    s.admission_date,
    YEAR(CURRENT_DATE) - YEAR(s.admission_date) + 1 as current_grade,
    'レベル3アクセス' as access_level_info
FROM students s
JOIN user_access_levels ual ON ual.username = USER()
WHERE ual.access_level = 3
AND ual.is_active = TRUE
AND YEAR(CURRENT_DATE) - YEAR(s.admission_date) + 1 = ual.grade_restriction;

-- レベル4: 教務主任ビュー（全学年の学生）
CREATE VIEW v_level4_academic_director AS
SELECT
    s.student_id,
    s.student_name,
    s.student_email,
    s.admission_date,
    YEAR(CURRENT_DATE) - YEAR(s.admission_date) + 1 as current_grade,
    'レベル4アクセス' as access_level_info
FROM students s
WHERE EXISTS (
    SELECT 1 FROM user_access_levels ual
    WHERE ual.username = USER()
    AND ual.access_level = 4
    AND ual.is_active = TRUE
);
```

```
-- レベル5: 校長ビュー（全ての情報）
CREATE VIEW v_level5_principal AS
SELECT
    s.student_id,
    s.student_name,
    s.student_email,
    s.admission_date,
    YEAR(CURRENT_DATE) - YEAR(s.admission_date) + 1 as current_grade,
    'レベル5アクセス' as access_level_info,
    -- 追加情報
    COUNT(DISTINCT sc.course_id) as enrolled_courses,
    AVG(g.score/g.max_score*100) as overall_average
FROM students s
LEFT JOIN student_courses sc ON s.student_id = sc.student_id
LEFT JOIN grades g ON s.student_id = g.student_id
WHERE EXISTS (
    SELECT 1 FROM user_access_levels ual
    WHERE ual.username = USER()
    AND ual.access_level = 5
    AND ual.is_active = TRUE
)
GROUP BY s.student_id, s.student_name, s.student_email, s.admission_date;

-- 3. 統合アクセス制御プロシージャ
DELIMITER //

CREATE PROCEDURE get_accessible_student_info(
    IN p_student_id BIGINT DEFAULT NULL
)
BEGIN
    DECLARE v_access_level INT;
    DECLARE v_associated_id BIGINT;
    DECLARE v_grade_restriction INT;

    -- ユーザーのアクセスレベルを取得
    SELECT access_level, associated_id, grade_restriction
    INTO v_access_level, v_associated_id, v_grade_restriction
    FROM user_access_levels
    WHERE username = USER()
    AND is_active = TRUE;

    -- アクセスレベルが見つからない場合
    IF v_access_level IS NULL THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Access denied: User not
registered in access control system';
    END IF;

    -- アクセスレベルに応じた処理
    CASE v_access_level
        WHEN 1 THEN -- 学生
            IF p_student_id IS NOT NULL AND p_student_id != v_associated_id THEN
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Level 1 Access: Can
only view own information';
```

```
            END IF;
            SELECT * FROM v_level1_student_own
            WHERE (p_student_id IS NULL OR student_id = p_student_id);

        WHEN 2 THEN -- 教師
            SELECT * FROM v_level2_teacher_students
            WHERE (p_student_id IS NULL OR student_id = p_student_id);

        WHEN 3 THEN -- 学年主任
            SELECT * FROM v_level3_grade_supervisor
            WHERE (p_student_id IS NULL OR student_id = p_student_id);

        WHEN 4 THEN -- 教務主任
            SELECT * FROM v_level4_academic_director
            WHERE (p_student_id IS NULL OR student_id = p_student_id);

        WHEN 5 THEN -- 校長
            SELECT * FROM v_level5_principal
            WHERE (p_student_id IS NULL OR student_id = p_student_id);

        ELSE
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid access level';
    END CASE;

    -- アクセスログの記録
    INSERT INTO security_procedure_log
        (procedure_name, username, parameters, access_granted, execution_time_ms)
    VALUES
        ('get_accessible_student_info', USER(),
         JSON_OBJECT('student_id', p_student_id, 'access_level', v_access_level),
         TRUE, 0);

END //

-- 4. 階層権限チェック関数
CREATE FUNCTION check_hierarchical_access(
    p_target_student_id BIGINT,
    p_required_level INT
) RETURNS BOOLEAN
READS SQL DATA
DETERMINISTIC
BEGIN
    DECLARE v_user_level INT;
    DECLARE v_associated_id BIGINT;
    DECLARE v_grade_restriction INT;
    DECLARE v_student_grade INT;
    DECLARE v_has_access BOOLEAN DEFAULT FALSE;

    -- ユーザーのアクセスレベルを取得
    SELECT access_level, associated_id, grade_restriction
    INTO v_user_level, v_associated_id, v_grade_restriction
    FROM user_access_levels
    WHERE username = USER()
    AND is_active = TRUE;
    -- アクセスレベルが要求レベル以上かチェック
    IF v_user_level >= p_required_level THEN
        CASE v_user_level
            WHEN 1 THEN -- 学生：自分の情報のみ
                SET v_has_access = (p_target_student_id = v_associated_id);
            WHEN 2 THEN -- 教師：担当講座の学生のみ
                SELECT COUNT(*) > 0 INTO v_has_access
                FROM student_courses sc
                JOIN courses c ON sc.course_id = c.course_id
                WHERE sc.student_id = p_target_student_id
                AND c.teacher_id = v_associated_id;
            WHEN 3 THEN -- 学年主任：担当学年の学生のみ
                SELECT YEAR(CURRENT_DATE) - YEAR(admission_date) + 1 INTO
v_student_grade
                FROM students WHERE student_id = p_target_student_id;
                SET v_has_access = (v_student_grade = v_grade_restriction);
            WHEN 4, 5 THEN -- 教務主任、校長：全アクセス
                SET v_has_access = TRUE;
        END CASE;
    END IF;

    RETURN v_has_access;
END //

DELIMITER ;

-- 5. 権限設定
-- 各レベルのロールに対応するビューの権限を設定
CREATE ROLE IF NOT EXISTS 'level1_student_role';
CREATE ROLE IF NOT EXISTS 'level2_teacher_role';
CREATE ROLE IF NOT EXISTS 'level3_grade_supervisor_role';
CREATE ROLE IF NOT EXISTS 'level4_academic_director_role';
CREATE ROLE IF NOT EXISTS 'level5_principal_role';

-- 各レベルのビューに権限付与
GRANT SELECT ON school_db.v_level1_student_own TO 'level1_student_role';
GRANT SELECT ON school_db.v_level2_teacher_students TO 'level2_teacher_role';
GRANT SELECT ON school_db.v_level3_grade_supervisor TO
'level3_grade_supervisor_role';
GRANT SELECT ON school_db.v_level4_academic_director TO
'level4_academic_director_role';
GRANT SELECT ON school_db.v_level5_principal TO 'level5_principal_role';

-- プロシージャの実行権限
GRANT EXECUTE ON PROCEDURE school_db.get_accessible_student_info TO
'level1_student_role';
GRANT EXECUTE ON PROCEDURE school_db.get_accessible_student_info TO
'level2_teacher_role';
GRANT EXECUTE ON PROCEDURE school_db.get_accessible_student_info TO
'level3_grade_supervisor_role';
GRANT EXECUTE ON PROCEDURE school_db.get_accessible_student_info TO
'level4_academic_director_role';
GRANT EXECUTE ON PROCEDURE school_db.get_accessible_student_info TO
```

```
'level5_principal_role';

-- 6. テスト実行例

-- 各レベルでのアクセステスト
SELECT 'Level 1 Test (Student Own Info):' as test_type;
-- CALL get_accessible_student_info(301);

SELECT 'Level 2 Test (Teacher Students):' as test_type;
-- CALL get_accessible_student_info(NULL);

-- 階層権限チェック関数のテスト
SELECT
    'Hierarchical Access Check Results' as test_type,
    301 as student_id,
    check_hierarchical_access(301, 1) as level1_access,
    check_hierarchical_access(301, 2) as level2_access,
    check_hierarchical_access(301, 3) as level3_access;

-- アクセスレベル確認
SELECT
    username,
    access_level,
    level_name,
    associated_id,
    grade_restriction
FROM user_access_levels
WHERE is_active = TRUE
ORDER BY access_level, username;
```

## 解答41-4

```
-- 時間制限付きアクセス制御の実装

-- 1. 時間制限管理テーブル
CREATE TABLE time_access_policies (
    policy_id INT AUTO_INCREMENT PRIMARY KEY,
    policy_name VARCHAR(100) NOT NULL,
    user_type ENUM('student', 'teacher', 'admin', 'emergency') NOT NULL,
    allowed_days JSON, -- [1,2,3,4,5] (月-金)
    start_time TIME NOT NULL,
    end_time TIME NOT NULL,
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    INDEX idx_user_type (user_type),
    INDEX idx_active (is_active)
);

-- 基本的な時間制限ポリシー
INSERT INTO time_access_policies (policy_name, user_type, allowed_days,
start_time, end_time) VALUES
('学生平常時間', 'student', JSON_ARRAY(2,3,4,5,6), '08:00:00', '18:00:00'),
('教師平常時間', 'teacher', JSON_ARRAY(2,3,4,5,6), '07:00:00', '20:00:00'),
('管理者時間', 'admin', JSON_ARRAY(1,2,3,4,5,6,7), '00:00:00', '23:59:59'),
('緊急時アクセス', 'emergency', JSON_ARRAY(1,2,3,4,5,6,7), '00:00:00', '23:59:59');

-- 2. 特別期間管理テーブル
CREATE TABLE special_access_periods (
    period_id INT AUTO_INCREMENT PRIMARY KEY,
    period_name VARCHAR(100) NOT NULL,
    period_type ENUM('exam', 'maintenance', 'emergency', 'holiday') NOT NULL,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    access_modifications JSON, -- 特別期間中のアクセス変更ルール
    is_active BOOLEAN DEFAULT TRUE,
    created_by VARCHAR(100),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    INDEX idx_period_dates (start_date, end_date),
    INDEX idx_period_type (period_type)
);

-- 定期試験期間の設定例
INSERT INTO special_access_periods (period_name, period_type, start_date,
end_date, access_modifications, created_by) VALUES
('2025年度第1回定期試験', 'exam', '2025-07-01', '2025-07-15',
 JSON_OBJECT(
     'student_grade_access', false,
     'teacher_grade_input_only', true,
     'extended_teacher_hours', JSON_OBJECT('start_time', '06:00:00', 'end_time',
'22:00:00')
 ),
 'admin'),
('夏季休暇期間', 'holiday', '2025-08-01', '2025-08-31',
 JSON_OBJECT(
     'student_access', false,
     'teacher_limited', true,
     'admin_maintenance', true
 ),
 'admin');

-- 3. 緊急連絡先管理テーブル
CREATE TABLE emergency_contacts (
    contact_id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(100) NOT NULL,
    contact_type ENUM('parent', 'guardian', 'medical', 'admin') NOT NULL,
    student_id BIGINT,
    contact_reason VARCHAR(200),
    authorized_by VARCHAR(100),
    expires_at TIMESTAMP,
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    FOREIGN KEY (student_id) REFERENCES students(student_id),
    INDEX idx_username (username),
    INDEX idx_student_contact (student_id, contact_type)
);

-- 緊急連絡先の例
INSERT INTO emergency_contacts (username, contact_type, student_id,
contact_reason, authorized_by) VALUES
('emergency_parent_301@localhost', 'parent', 301, '緊急時連絡', 'admin'),
('emergency_medical@localhost', 'medical', NULL, '医療緊急時', 'admin');

-- 4. 時間制限チェックプロシージャ
DELIMITER //

CREATE PROCEDURE check_time_access_permission(
    OUT p_access_granted BOOLEAN,
    OUT p_restriction_reason TEXT
)
BEGIN
    DECLARE v_current_day INT;
    DECLARE v_current_time TIME;
    DECLARE v_current_date DATE;
    DECLARE v_user_type VARCHAR(20);
    DECLARE v_is_emergency_contact BOOLEAN DEFAULT FALSE;
    DECLARE v_special_period JSON DEFAULT NULL;
    DECLARE v_policy_found BOOLEAN DEFAULT FALSE;

    SET p_access_granted = FALSE;
    SET p_restriction_reason = '';
    SET v_current_day = DAYOFWEEK(CURRENT_DATE); -- 1=日曜日, 2=月曜日...
    SET v_current_time = CURRENT_TIME;
    SET v_current_date = CURRENT_DATE;

    -- ユーザータイプの判定
    SET v_user_type = CASE
        WHEN USER() LIKE '%student%' THEN 'student'
        WHEN USER() LIKE '%teacher%' THEN 'teacher'
        WHEN USER() LIKE '%admin%' OR USER() = 'root' THEN 'admin'
        ELSE 'unknown'
    END;

    -- 緊急連絡先かどうかチェック
    SELECT COUNT(*) > 0 INTO v_is_emergency_contact
    FROM emergency_contacts
    WHERE username = USER()
    AND is_active = TRUE
    AND (expires_at IS NULL OR expires_at > NOW());

    IF v_is_emergency_contact THEN
        SET v_user_type = 'emergency';
    END IF;

    -- 特別期間中かどうかチェック
    SELECT access_modifications INTO v_special_period
    FROM special_access_periods
```

```sql
    WHERE v_current_date BETWEEN start_date AND end_date
    AND is_active = TRUE
    ORDER BY period_id DESC
    LIMIT 1;

    -- 特別期間中の制限チェック
    IF v_special_period IS NOT NULL THEN
        CASE v_user_type
            WHEN 'student' THEN
                IF JSON_EXTRACT(v_special_period, '$.student_access') = false THEN
                    SET p_restriction_reason = 'Student access restricted during special period';
                    LEAVE check_proc;
                END IF;
                IF JSON_EXTRACT(v_special_period, '$.student_grade_access') = false THEN
                    SET p_restriction_reason = 'Student grade access restricted during exam period';
                    LEAVE check_proc;
                END IF;
            WHEN 'teacher' THEN
                IF JSON_EXTRACT(v_special_period, '$.teacher_limited') = true THEN
                    -- 特別期間中の教師時間制限適用
                    IF JSON_EXTRACT(v_special_period, '$.extended_teacher_hours') IS NOT NULL THEN
                        -- 拡張時間の適用
                        SET v_current_time = v_current_time; -- 拡張時間チェックは後で実装
                    END IF;
                END IF;
        END CASE;
    END IF;

    -- 通常の時間制限ポリシーチェック
    SELECT COUNT(*) > 0 INTO v_policy_found
    FROM time_access_policies
    WHERE user_type = v_user_type
    AND is_active = TRUE
    AND JSON_CONTAINS(allowed_days, CAST(v_current_day AS JSON))
    AND v_current_time BETWEEN start_time AND end_time;

    IF v_policy_found THEN
        SET p_access_granted = TRUE;
    ELSE
        SET p_restriction_reason = CONCAT('Access denied: Outside allowed hours for ', v_user_type);
    END IF;

    check_proc: BEGIN END; -- ラベル用

    -- アクセスログの記録
    INSERT INTO security_procedure_log
        (procedure_name, username, parameters, access_granted, execution_time_ms, error_message)
    VALUES
        ('check_time_access_permission', USER(),
         JSON_OBJECT('user_type', v_user_type, 'check_time', v_current_time),
         p_access_granted, 0, p_restriction_reason);

END //

-- 5. 時間制限付きデータアクセスプロシージャ
CREATE PROCEDURE time_restricted_get_student_info(
    IN p_student_id BIGINT
)
BEGIN
    DECLARE v_access_granted BOOLEAN;
    DECLARE v_restriction_reason TEXT;

    -- 時間制限チェック
    CALL check_time_access_permission(v_access_granted, v_restriction_reason);

    IF NOT v_access_granted THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_restriction_reason;
    END IF;

    -- 通常のセキュリティチェック後、データを取得
    CALL get_accessible_student_info(p_student_id);

END //

-- 6. 特別期間管理プロシージャ
CREATE PROCEDURE set_special_access_period(
    IN p_period_name VARCHAR(100),
    IN p_period_type ENUM('exam', 'maintenance', 'emergency', 'holiday'),
    IN p_start_date DATE,
    IN p_end_date DATE,
    IN p_access_modifications JSON
)
BEGIN
    -- 管理者権限チェック
    IF USER() NOT LIKE '%admin%' AND USER() != 'root' THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Access denied: Admin
privileges required';
    END IF;

    -- 特別期間の登録
    INSERT INTO special_access_periods
        (period_name, period_type, start_date, end_date, access_modifications,
created_by)
    VALUES
        (p_period_name, p_period_type, p_start_date, p_end_date,
p_access_modifications, USER());

    -- 通知ログの記録
    INSERT INTO user_management_log (action, username, host, user_type,
created_by, notes)
    VALUES ('SPECIAL_PERIOD', 'SYSTEM', 'SYSTEM', 'time_control', USER(),
            CONCAT('Special access period set: ', p_period_name, ' (', 
p_start_date, ' to ', p_end_date, ')'));

    SELECT CONCAT('Special access period "', p_period_name, '" has been set') as 
result;

END //

DELIMITER ;

-- 7. 時間制限監視システム
CREATE EVENT IF NOT EXISTS cleanup_expired_emergency_contacts
ON SCHEDULE EVERY 1 HOUR
DO
  UPDATE emergency_contacts
  SET is_active = FALSE
  WHERE expires_at <= NOW() AND is_active = TRUE;

-- テスト実行例
-- CALL check_time_access_permission(@granted, @reason);
-- SELECT @granted as access_granted, @reason as restriction_reason;

-- 特別期間設定例
-- CALL set_special_access_period('緊急メンテナンス', 'maintenance', '2025-06-01', 
'2025-06-01',
--      JSON_OBJECT('student_access', false, 'teacher_access', false, 
'admin_only', true));
```

解答41-5

```
-- 保護者アクセスシステムの拡張実装

-- 1. 拡張保護者関係テーブル
CREATE TABLE enhanced_parent_relationships (
    relationship_id INT AUTO_INCREMENT PRIMARY KEY,
    parent_username VARCHAR(100) NOT NULL,
    student_id BIGINT NOT NULL,
    relationship_type ENUM('father', 'mother', 'guardian', 'emergency_contact') 
NOT NULL,
    access_level ENUM('full', 'attendance_only', 'emergency_only') DEFAULT 'full',
    is_primary_contact BOOLEAN DEFAULT FALSE,
    notification_preferences JSON, -- 通知設定
    access_restrictions JSON, -- 追加制限
    approval_status ENUM('pending', 'approved', 'rejected', 'suspended') DEFAULT 
'pending',
    approved_by VARCHAR(100),
    approved_at TIMESTAMP NULL,
    expires_at TIMESTAMP NULL,
    is_active BOOLEAN DEFAULT FALSE, -- 承認後にTRUEに
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (student_id) REFERENCES students(student_id),
    INDEX idx_parent_student (parent_username, student_id),
    INDEX idx_approval_status (approval_status),
    INDEX idx_active_relationships (is_active, expires_at)
);

-- 保護者関係サンプルデータ
INSERT INTO enhanced_parent_relationships
    (parent_username, student_id, relationship_type, access_level,
notification_preferences, approval_status, approved_by, approved_at, is_active)
VALUES
('parent_yamada@localhost', 301, 'father', 'full',
 JSON_OBJECT('grade_alerts', true, 'attendance_alerts', true, 'emergency_alerts',
true),
 'approved', 'admin', NOW(), TRUE),
('parent_tanaka@localhost', 302, 'mother', 'attendance_only',
 JSON_OBJECT('grade_alerts', false, 'attendance_alerts', true, 'emergency_alerts',
true),
 'approved', 'admin', NOW(), TRUE);

-- 2. 通知管理テーブル
CREATE TABLE parent_notifications (
    notification_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    parent_username VARCHAR(100) NOT NULL,
    student_id BIGINT NOT NULL,
    notification_type ENUM('attendance_alert', 'grade_change', 'emergency',
'general') NOT NULL,
    severity ENUM('low', 'medium', 'high', 'critical') DEFAULT 'medium',
    title VARCHAR(200) NOT NULL,
    message TEXT NOT NULL,
    related_data JSON, -- 関連データ（成績、出席等）
    is_read BOOLEAN DEFAULT FALSE,
    read_at TIMESTAMP NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    expires_at TIMESTAMP,

    INDEX idx_parent_unread (parent_username, is_read),
    INDEX idx_student_notifications (student_id, created_at),
    INDEX idx_notification_type (notification_type, severity)
);

-- 3. 保護者アクセス履歴テーブル
CREATE TABLE parent_access_history (
    access_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    parent_username VARCHAR(100) NOT NULL,
    student_id BIGINT,
    access_type ENUM('login', 'view_grades', 'view_attendance', 'view_info',
'download_report') NOT NULL,
    access_details JSON,
    ip_address VARCHAR(45),
    user_agent VARCHAR(500),
    session_duration_seconds INT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    INDEX idx_parent_access (parent_username, created_at),
    INDEX idx_student_access (student_id, created_at),
    INDEX idx_access_type (access_type)
);

-- 4. 段階的アクセス権限管理プロシージャ
DELIMITER //

CREATE PROCEDURE parent_get_child_info_enhanced(
    IN p_student_id BIGINT,
    IN p_info_type ENUM('basic', 'grades', 'attendance', 'health', 'emergency')
)
BEGIN
    DECLARE v_access_level VARCHAR(20);
    DECLARE v_relationship_active BOOLEAN DEFAULT FALSE;
    DECLARE v_notification_prefs JSON;
    DECLARE v_access_granted BOOLEAN DEFAULT FALSE;

    -- 保護者の子供へのアクセス権限確認
    SELECT
        access_level,
        is_active,
        notification_preferences
    INTO v_access_level, v_relationship_active, v_notification_prefs
    FROM enhanced_parent_relationships
    WHERE parent_username = USER()
    AND student_id = p_student_id
    AND approval_status = 'approved'
    AND (expires_at IS NULL OR expires_at > NOW());

    IF NOT v_relationship_active THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Access denied: No active
relationship with student';
    END IF;

    -- アクセスレベルに応じた権限チェック
    CASE v_access_level
        WHEN 'full' THEN
            SET v_access_granted = TRUE;
        WHEN 'attendance_only' THEN
            SET v_access_granted = (p_info_type IN ('basic', 'attendance'));
        WHEN 'emergency_only' THEN
            SET v_access_granted = (p_info_type = 'emergency');
        ELSE
            SET v_access_granted = FALSE;
    END CASE;

    IF NOT v_access_granted THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =
            CONCAT('Access denied: ', v_access_level, ' access does not permit ',
p_info_type, ' information');
    END IF;

    -- 要求された情報タイプに応じてデータを返す
    CASE p_info_type
        WHEN 'basic' THEN
            SELECT
                s.student_id,
                s.student_name,
                YEAR(CURRENT_DATE) - YEAR(s.admission_date) + 1 as current_grade,
                s.admission_date,
                '基本情報' as info_type
            FROM students s
            WHERE s.student_id = p_student_id;

        WHEN 'grades' THEN
            SELECT
                g.course_id,
                c.course_name,
                g.grade_type,
                g.score,
                g.max_score,
                ROUND((g.score / g.max_score) * 100, 1) as percentage,
                g.submission_date
            FROM grades g
            JOIN courses c ON g.course_id = c.course_id
            WHERE g.student_id = p_student_id
            AND g.submission_date >= DATE_SUB(CURRENT_DATE, INTERVAL 90 DAY)
            ORDER BY g.submission_date DESC;

        WHEN 'attendance' THEN
            SELECT
                cs.schedule_date,
                c.course_name,
                cp.start_time,
                cp.end_time,
                a.status,
                CASE a.status
                    WHEN 'present' THEN '出席'
                    WHEN 'absent' THEN '欠席'
                    WHEN 'late' THEN '遅刻'
                END as status_japanese
            FROM attendance a
            JOIN course_schedule cs ON a.schedule_id = cs.schedule_id
            JOIN courses c ON cs.course_id = c.course_id
            JOIN class_periods cp ON cs.period_id = cp.period_id
            WHERE a.student_id = p_student_id
            AND cs.schedule_date >= DATE_SUB(CURRENT_DATE, INTERVAL 30 DAY)
            ORDER BY cs.schedule_date DESC;

        WHEN 'emergency' THEN
            SELECT
                s.student_name,
                '緊急連絡情報のみ表示' as emergency_info,
                ec.contact_type,
                ec.contact_reason
            FROM students s
            JOIN emergency_contacts ec ON s.student_id = ec.student_id
            WHERE s.student_id = p_student_id
            AND ec.is_active = TRUE;
    END CASE;

    -- アクセス履歴の記録
    INSERT INTO parent_access_history (parent_username, student_id, access_type,
access_details)
    VALUES (USER(), p_student_id, CONCAT('view_', p_info_type),
            JSON_OBJECT('access_level', v_access_level, 'timestamp', NOW()));

END //

-- 5. 自動通知システム
CREATE PROCEDURE generate_parent_notifications()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE v_parent_username VARCHAR(100);
    DECLARE v_student_id BIGINT;
    DECLARE v_notification_prefs JSON;

    DECLARE parent_cursor CURSOR FOR
        SELECT parent_username, student_id, notification_preferences
        FROM enhanced_parent_relationships
        WHERE is_active = TRUE
        AND approval_status = 'approved';

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN parent_cursor;

    notification_loop: LOOP
        FETCH parent_cursor INTO v_parent_username, v_student_id,
v_notification_prefs;
        IF done THEN
            LEAVE notification_loop;
        END IF;

        -- 出席アラートの生成
        IF JSON_EXTRACT(v_notification_prefs, '$.attendance_alerts') = true THEN
            INSERT INTO parent_notifications
                (parent_username, student_id, notification_type, severity, title,
message, related_data)
            SELECT
                v_parent_username,
                v_student_id,
                'attendance_alert',
                'medium',
                CONCAT(s.student_name, 'さんの出席状況について'),
                CONCAT('過去1週間で', absent_count, '回の欠席があります。'),
                JSON_OBJECT('absent_days', absent_count, 'period', 'last_7_days')
            FROM students s
            JOIN (
                SELECT
                    a.student_id,
                    COUNT(CASE WHEN a.status = 'absent' THEN 1 END) as
absent_count
                FROM attendance a
                JOIN course_schedule cs ON a.schedule_id = cs.schedule_id
                WHERE a.student_id = v_student_id
                AND cs.schedule_date >= DATE_SUB(CURRENT_DATE, INTERVAL 7 DAY)
                GROUP BY a.student_id
                HAVING absent_count >= 3
            ) attendance_summary ON s.student_id = attendance_summary.student_id
            WHERE s.student_id = v_student_id
            AND NOT EXISTS (
                SELECT 1 FROM parent_notifications pn
                WHERE pn.parent_username = v_parent_username
                AND pn.student_id = v_student_id
                AND pn.notification_type = 'attendance_alert'
                AND pn.created_at >= DATE_SUB(NOW(), INTERVAL 1 DAY)
            );
        END IF;

        -- 成績変動アラートの生成
        IF JSON_EXTRACT(v_notification_prefs, '$.grade_alerts') = true THEN
            INSERT INTO parent_notifications
                (parent_username, student_id, notification_type, severity, title,
message, related_data)
            SELECT
                v_parent_username,
                v_student_id,
                'grade_change',
                'low',
                CONCAT(s.student_name, 'さんの新しい成績が登録されました'),
                CONCAT(c.course_name, 'の', g.grade_type, ': ', g.score, '/',
g.max_score, '点'),
                JSON_OBJECT('course_id', g.course_id, 'grade_type', g.grade_type,
'score', g.score)
            FROM grades g
            JOIN students s ON g.student_id = s.student_id
            JOIN courses c ON g.course_id = c.course_id
            WHERE g.student_id = v_student_id
            AND g.submission_date >= DATE_SUB(NOW(), INTERVAL 1 DAY)
            AND NOT EXISTS (
                SELECT 1 FROM parent_notifications pn
                WHERE pn.parent_username = v_parent_username
                AND pn.student_id = v_student_id
                AND pn.notification_type = 'grade_change'
                AND JSON_EXTRACT(pn.related_data, '$.course_id') = g.course_id
                AND JSON_EXTRACT(pn.related_data, '$.grade_type') = g.grade_type
            );
        END IF;

    END LOOP;

    CLOSE parent_cursor;

    SELECT CONCAT('Generated notifications for active parent relationships') as
```

```
result;

END //

-- 6. 異常アクセス検知プロシージャ
CREATE PROCEDURE detect_parent_access_anomalies()
BEGIN
    -- 短時間での大量アクセス検知
    INSERT INTO parent_notifications
        (parent_username, student_id, notification_type, severity, title, message,
related_data)
    SELECT
        pah.parent_username,
        pah.student_id,
        'general',
        'high',
        'セキュリティアラート: 異常なアクセスパターン',
        CONCAT('過去1時間で', access_count, '回のアクセスがありました。'),
        JSON_OBJECT('access_count', access_count, 'time_period', '1_hour',
'alert_type', 'high_frequency')
    FROM (
        SELECT
            parent_username,
            student_id,
            COUNT(*) as access_count
        FROM parent_access_history
        WHERE created_at >= DATE_SUB(NOW(), INTERVAL 1 HOUR)
        GROUP BY parent_username, student_id
        HAVING access_count > 50
    ) pah
    WHERE NOT EXISTS (
        SELECT 1 FROM parent_notifications pn
        WHERE pn.parent_username = pah.parent_username
        AND pn.notification_type = 'general'
        AND JSON_EXTRACT(pn.related_data, '$.alert_type') = 'high_frequency'
        AND pn.created_at >= DATE_SUB(NOW(), INTERVAL 1 HOUR)
    );

    -- 深夜アクセスの検知
    INSERT INTO parent_notifications
        (parent_username, student_id, notification_type, severity, title, message,
related_data)
    SELECT DISTINCT
        pah.parent_username,
        pah.student_id,
        'general',
        'medium',
        'セキュリティ通知: 深夜時間帯のアクセス',
        '深夜時間帯（22:00-6:00）にアクセスがありました。',
        JSON_OBJECT('access_time', pah.created_at, 'alert_type', 'off_hours')
    FROM parent_access_history pah
    WHERE pah.created_at >= DATE_SUB(NOW(), INTERVAL 1 DAY)
    AND (HOUR(pah.created_at) >= 22 OR HOUR(pah.created_at) <= 6)
    AND NOT EXISTS (
            SELECT 1 FROM parent_notifications pn
            WHERE pn.parent_username = pah.parent_username
            AND pn.notification_type = 'general'
            AND JSON_EXTRACT(pn.related_data, '$.alert_type') = 'off_hours'
            AND DATE(pn.created_at) = DATE(pah.created_at)
        );

END //

-- 7. 承認ワークフロープロシージャ
CREATE PROCEDURE approve_parent_relationship(
    IN p_relationship_id INT,
    IN p_approval_decision ENUM('approved', 'rejected'),
    IN p_approval_notes TEXT
)
BEGIN
    DECLARE v_parent_username VARCHAR(100);
    DECLARE v_student_id BIGINT;

    -- 管理者権限チェック
    IF USER() NOT LIKE '%admin%' AND USER() NOT LIKE '%office%' AND USER() !=
'root' THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Access denied: Admin or office
staff privileges required';
    END IF;

    -- 関係情報を取得
    SELECT parent_username, student_id
    INTO v_parent_username, v_student_id
    FROM enhanced_parent_relationships
    WHERE relationship_id = p_relationship_id
    AND approval_status = 'pending';

    IF v_parent_username IS NULL THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Relationship not found or not
pending approval';
    END IF;

    -- 承認状態を更新
    UPDATE enhanced_parent_relationships
    SET
        approval_status = p_approval_decision,
        approved_by = USER(),
        approved_at = NOW(),
        is_active = (p_approval_decision = 'approved')
    WHERE relationship_id = p_relationship_id;

    -- 承認通知を生成
    INSERT INTO parent_notifications
        (parent_username, student_id, notification_type, severity, title, message,
related_data)
    VALUES
        (v_parent_username, v_student_id, 'general',
         CASE p_approval_decision WHEN 'approved' THEN 'low' ELSE 'medium' END,
         CASE p_approval_decision
             WHEN 'approved' THEN 'アカウント承認完了'
             ELSE 'アカウント承認拒否'
         END,
         CASE p_approval_decision
             WHEN 'approved' THEN 'お子様の情報へのアクセスが承認されました。'
             ELSE CONCAT('アカウント承認が拒否されました。理由: ',
IFNULL(p_approval_notes, '詳細は学校にお問い合わせください。'))
         END,
         JSON_OBJECT('approval_decision', p_approval_decision, 'approved_by',
USER()));

    -- ログ記録
    INSERT INTO user_management_log (action, username, host, user_type,
created_by, notes)
    VALUES ('APPROVAL', v_parent_username, 'parent_system', 'parent_approval',
USER(),
            CONCAT('Parent relationship ', p_approval_decision, ' for student ',
v_student_id));

    SELECT CONCAT('Parent relationship ', p_approval_decision, ' successfully') as
result;

END //

DELIMITER ;

-- 8. 定期実行イベント
CREATE EVENT IF NOT EXISTS parent_notification_generator
ON SCHEDULE EVERY 1 HOUR
DO
  CALL generate_parent_notifications();

CREATE EVENT IF NOT EXISTS parent_access_anomaly_detector
ON SCHEDULE EVERY 6 HOUR
DO
  CALL detect_parent_access_anomalies();

-- テスト実行例
-- CALL parent_get_child_info_enhanced(301, 'grades');
-- CALL approve_parent_relationship(1, 'approved', 'Initial setup');
-- CALL generate_parent_notifications();
```

### 解答41-6

```
-- 総合セキュリティシステムの実装

-- 1. 統合ユーザーコンテキスト管理
CREATE TABLE unified_security_context (
    context_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(100) NOT NULL,
    primary_role ENUM('student', 'teacher', 'grade_supervisor',
'academic_director', 'principal', 'office_staff', 'parent', 'admin') NOT NULL,
    role_hierarchy_level INT NOT NULL, -- 1-10の階層レベル
    associated_entities JSON, -- 関連するID群（学生ID、教師ID、クラスID等）
    access_policies JSON, -- カスタムアクセスポリシー
    temporary_permissions JSON, -- 一時的な権限
    security_clearance_level INT DEFAULT 1, -- セキュリティクリアランス
    context_metadata JSON, -- 追加のメタデータ
    is_active BOOLEAN DEFAULT TRUE,
    last_updated TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,

    UNIQUE KEY unique_username (username),
    INDEX idx_role_level (primary_role, role_hierarchy_level),
    INDEX idx_clearance (security_clearance_level),
    INDEX idx_active (is_active)
);

-- 統合コンテキストサンプルデータ
INSERT INTO unified_security_context
    (username, primary_role, role_hierarchy_level, associated_entities,
access_policies, security_clearance_level)
VALUES
('student_301@localhost', 'student', 1, JSON_OBJECT('student_id', 301),
 JSON_OBJECT('data_retention_days', 90, 'export_allowed', false), 1),
('teacher_tanaka@localhost', 'teacher', 3, JSON_OBJECT('teacher_id', 101,
'courses', JSON_ARRAY('1', '2')),
 JSON_OBJECT('grade_modification_window_hours', 72, 'bulk_operations', true), 2),
('principal@localhost', 'principal', 8, JSON_OBJECT('authority_scope', 'all'),
 JSON_OBJECT('emergency_override', true, 'audit_exempt', false), 4);

-- 2. 動的アクセスポリシーエンジン
CREATE TABLE access_policy_rules (
    rule_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    rule_name VARCHAR(100) NOT NULL,
    target_table VARCHAR(64) NOT NULL,
    operation_type ENUM('SELECT', 'INSERT', 'UPDATE', 'DELETE', 'ALL') NOT NULL,
    condition_template TEXT NOT NULL, -- 動的条件のテンプレート
    role_requirements JSON, -- 必要な役職・レベル
    time_restrictions JSON, -- 時間制限
    data_sensitivity_level INT DEFAULT 1, -- データ機密レベル
    is_active BOOLEAN DEFAULT TRUE,
    created_by VARCHAR(100),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    INDEX idx_target_operation (target_table, operation_type),
    INDEX idx_sensitivity (data_sensitivity_level),
    INDEX idx_active (is_active)
);

-- 動的ポリシールールの例
INSERT INTO access_policy_rules
    (rule_name, target_table, operation_type, condition_template,
role_requirements, time_restrictions, data_sensitivity_level, created_by)
VALUES
```

```sql
('学生自分情報アクセス', 'students', 'SELECT',
 'student_id = {{user.associated_entities.student_id}}',
 JSON_OBJECT('min_level', 1, 'allowed_roles', JSON_ARRAY('student')),
 JSON_OBJECT('weekdays_only', true, 'hours', JSON_OBJECT('start', 8, 'end', 18)),
2, 'system'),
('教師担当学生アクセス', 'students', 'SELECT',
 'student_id IN (SELECT sc.student_id FROM student_courses sc JOIN courses c ON
sc.course_id = c.course_id WHERE c.teacher_id =
{{user.associated_entities.teacher_id}})',
 JSON_OBJECT('min_level', 3, 'allowed_roles', JSON_ARRAY('teacher',
'grade_supervisor')),
 JSON_OBJECT('weekdays_only', true, 'hours', JSON_OBJECT('start', 7, 'end', 20)),
2, 'system');

-- 3. カスタムアクセスポリシー管理
DELIMITER //

CREATE PROCEDURE apply_dynamic_access_policy(
    IN p_username VARCHAR(100),
    IN p_table_name VARCHAR(64),
    IN p_operation ENUM('SELECT', 'INSERT', 'UPDATE', 'DELETE'),
    OUT p_access_condition TEXT,
    OUT p_access_granted BOOLEAN
)
BEGIN
    DECLARE v_user_context JSON;
    DECLARE v_role VARCHAR(50);
    DECLARE v_level INT;
    DECLARE v_clearance INT;
    DECLARE v_rule_condition TEXT;
    DECLARE v_time_allowed BOOLEAN DEFAULT TRUE;
    DECLARE v_current_hour INT;

    SET p_access_granted = FALSE;
    SET p_access_condition = '1=0'; -- デフォルトでアクセス拒否
    SET v_current_hour = HOUR(NOW());

    -- ユーザーコンテキストを取得
    SELECT
        JSON_OBJECT(
            'role', primary_role,
            'level', role_hierarchy_level,
            'clearance', security_clearance_level,
            'entities', associated_entities,
            'policies', access_policies
        ),
        primary_role,
        role_hierarchy_level,
        security_clearance_level
    INTO v_user_context, v_role, v_level, v_clearance
    FROM unified_security_context
    WHERE username = p_username
    AND is_active = TRUE;
    IF v_user_context IS NULL THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'User security context not
found';
    END IF;

    -- 適用可能なポリシールールを検索
    SELECT condition_template INTO v_rule_condition
    FROM access_policy_rules apr
    WHERE apr.target_table = p_table_name
    AND (apr.operation_type = p_operation OR apr.operation_type = 'ALL')
    AND JSON_CONTAINS(JSON_EXTRACT(apr.role_requirements, '$.allowed_roles'),
JSON_QUOTE(v_role))
    AND v_level >= JSON_EXTRACT(apr.role_requirements, '$.min_level')
    AND v_clearance >= apr.data_sensitivity_level
    AND apr.is_active = TRUE
    ORDER BY apr.data_sensitivity_level DESC, apr.rule_id
    LIMIT 1;

    IF v_rule_condition IS NOT NULL THEN
        -- 時間制限チェック
        SELECT COUNT(*) = 0 INTO v_time_allowed
        FROM access_policy_rules apr
        WHERE apr.target_table = p_table_name
        AND JSON_EXTRACT(apr.time_restrictions, '$.weekdays_only') = true
        AND DAYOFWEEK(NOW()) IN (1, 7); -- 土日

        IF v_time_allowed THEN
            SELECT COUNT(*) = 0 INTO v_time_allowed
            FROM access_policy_rules apr
            WHERE apr.target_table = p_table_name
            AND (v_current_hour < JSON_EXTRACT(apr.time_restrictions,
'$.hours.start')
                 OR v_current_hour > JSON_EXTRACT(apr.time_restrictions,
'$.hours.end'));
        END IF;

        IF v_time_allowed THEN
            -- テンプレート変数を実際の値に置換
            SET p_access_condition = REPLACE(v_rule_condition,
'{{user.associated_entities.student_id}}',

JSON_UNQUOTE(JSON_EXTRACT(v_user_context, '$.entities.student_id')));
            SET p_access_condition = REPLACE(p_access_condition,
'{{user.associated_entities.teacher_id}}',

JSON_UNQUOTE(JSON_EXTRACT(v_user_context, '$.entities.teacher_id')));
            SET p_access_granted = TRUE;
        END IF;
    END IF;

    -- アクセスログ記録
    INSERT INTO security_procedure_log
        (procedure_name, username, parameters, access_granted, execution_time_ms,
error_message)
```

```
    VALUES
        ('apply_dynamic_access_policy', p_username,
         JSON_OBJECT('table', p_table_name, 'operation', p_operation, 'condition',
p_access_condition),
         p_access_granted, 0, CASE WHEN NOT p_access_granted THEN 'Policy
evaluation failed' ELSE NULL END);

END //

-- 4. 一時的権限委譲システム
CREATE PROCEDURE delegate_temporary_permission(
    IN p_grantor_username VARCHAR(100),
    IN p_grantee_username VARCHAR(100),
    IN p_permission_scope JSON, -- 委譲する権限の範囲
    IN p_duration_hours INT,
    IN p_justification TEXT
)
BEGIN
    DECLARE v_grantor_level INT;
    DECLARE v_grantee_level INT;
    DECLARE v_expiry_time TIMESTAMP;

    SET v_expiry_time = DATE_ADD(NOW(), INTERVAL p_duration_hours HOUR);

    -- 委譲者の権限レベルチェック
    SELECT role_hierarchy_level INTO v_grantor_level
    FROM unified_security_context
    WHERE username = p_grantor_username AND is_active = TRUE;

    SELECT role_hierarchy_level INTO v_grantee_level
    FROM unified_security_context
    WHERE username = p_grantee_username AND is_active = TRUE;

    IF v_grantor_level IS NULL OR v_grantee_level IS NULL THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid grantor or grantee';
    END IF;

    IF v_grantor_level <= v_grantee_level THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cannot delegate to equal or
higher level user';
    END IF;

    -- 一時的権限を付与
    UPDATE unified_security_context
    SET temporary_permissions = JSON_MERGE_PATCH(
        IFNULL(temporary_permissions, '{}'),
        JSON_OBJECT(
            CONCAT('delegation_', UNIX_TIMESTAMP()),
            JSON_OBJECT(
                'grantor', p_grantor_username,
                'scope', p_permission_scope,
                'expires_at', v_expiry_time,
                'justification', p_justification
            )
            )
        )
    WHERE username = p_grantee_username;

    -- 委譲ログ記録
    INSERT INTO user_management_log (action, username, host, user_type,
created_by, notes)
    VALUES ('DELEGATE', p_grantee_username, 'delegation_system',
'temp_permission', p_grantor_username,
            CONCAT('Temporary permission delegated until ', v_expiry_time, '.
Scope: ', p_permission_scope));

    SELECT CONCAT('Permission delegated successfully until ', v_expiry_time) as
result;

END //

-- 5. 全テーブル行レベル制御システム
CREATE PROCEDURE setup_universal_row_level_security()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE v_table_name VARCHAR(64);

    -- 学校DBの全テーブルを取得
    DECLARE table_cursor CURSOR FOR
        SELECT table_name
        FROM information_schema.tables
        WHERE table_schema = 'school_db'
        AND table_type = 'BASE TABLE';

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN table_cursor;

    security_loop: LOOP
        FETCH table_cursor INTO v_table_name;
        IF done THEN
            LEAVE security_loop;
        END IF;

        -- 各テーブルに対してセキュリティビューを作成
        CASE v_table_name
            WHEN 'students' THEN
                -- 学生テーブル用セキュリティビュー（既存のロジックを使用）
                SET @sql = CONCAT('CREATE OR REPLACE VIEW v_secure_',
v_table_name, ' AS
                    SELECT * FROM ', v_table_name, ' WHERE 1=1'); -- 基本フレーム
            WHEN 'grades' THEN
                -- 成績テーブル用セキュリティビュー
                SET @sql = CONCAT('CREATE OR REPLACE VIEW v_secure_',
v_table_name, ' AS
                    SELECT * FROM ', v_table_name, ' WHERE 1=1'); -- 基本フレーム
            ELSE
                -- その他のテーブル用デフォルトビュー
                SET @sql = CONCAT('CREATE OR REPLACE VIEW v_secure_',
v_table_name, ' AS
                    SELECT * FROM ', v_table_name, ' WHERE 1=1'); -- 基本フレーム
        END CASE;

        PREPARE stmt FROM @sql;
        EXECUTE stmt;
        DEALLOCATE PREPARE stmt;

    END LOOP;

    CLOSE table_cursor;

    SELECT 'Universal row-level security views created' as result;

END //

-- 6. 包括的監査システム
CREATE TABLE comprehensive_audit_log (
    audit_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    event_timestamp TIMESTAMP(6) DEFAULT CURRENT_TIMESTAMP(6),
    username VARCHAR(100) NOT NULL,
    session_id VARCHAR(64),
    event_type ENUM('LOGIN', 'LOGOUT', 'DATA_ACCESS', 'DATA_MODIFY',
'PERMISSION_CHANGE', 'POLICY_VIOLATION', 'SECURITY_ALERT') NOT NULL,
    target_object VARCHAR(100),
    operation VARCHAR(20),
    affected_records JSON, -- 影響を受けたレコードのID
    query_hash VARCHAR(64), -- 実行されたクエリのハッシュ
    execution_time_ms INT,
    ip_address VARCHAR(45),
    user_agent VARCHAR(500),
    risk_score INT DEFAULT 0, -- 0-100のリスクスコア
    compliance_tags JSON, -- コンプライアンス関連タグ
    additional_metadata JSON,

    INDEX idx_timestamp (event_timestamp),
    INDEX idx_username_timestamp (username, event_timestamp),
    INDEX idx_event_type (event_type),
    INDEX idx_risk_score (risk_score),
    PARTITION BY RANGE (YEAR(event_timestamp)) (
        PARTITION p2024 VALUES LESS THAN (2025),
        PARTITION p2025 VALUES LESS THAN (2026),
        PARTITION p2026 VALUES LESS THAN (2027),
        PARTITION pmax VALUES LESS THAN MAXVALUE
    )
);

-- 監査ログ記録プロシージャ
CREATE PROCEDURE log_comprehensive_audit(
    IN p_event_type ENUM('LOGIN', 'LOGOUT', 'DATA_ACCESS', 'DATA_MODIFY',
'PERMISSION_CHANGE', 'POLICY_VIOLATION', 'SECURITY_ALERT'),
    IN p_target_object VARCHAR(100),
    IN p_operation VARCHAR(20),
    IN p_affected_records JSON,
    IN p_risk_score INT,
    IN p_additional_metadata JSON
)
BEGIN
    DECLARE v_session_id VARCHAR(64);
    DECLARE v_query_hash VARCHAR(64);

    -- セッションIDの生成（簡易版）
    SET v_session_id = CONCAT(CONNECTION_ID(), '_', UNIX_TIMESTAMP());

    -- クエリハッシュの生成（実際の環境では実行されたクエリをハッシュ化）
    SET v_query_hash = SHA2(CONCAT(p_target_object, p_operation, NOW()), 256);

    INSERT INTO comprehensive_audit_log
        (username, session_id, event_type, target_object, operation,
affected_records,
         query_hash, risk_score, additional_metadata)
    VALUES
        (USER(), v_session_id, p_event_type, p_target_object, p_operation,
p_affected_records,
         v_query_hash, p_risk_score, p_additional_metadata);

END //

-- 7. コンプライアンスレポート生成
CREATE PROCEDURE generate_compliance_report(
    IN p_report_type ENUM('GDPR', 'FERPA', 'SECURITY_SUMMARY', 'ACCESS_REVIEW'),
    IN p_start_date DATE,
    IN p_end_date DATE
)
BEGIN
    CASE p_report_type
        WHEN 'GDPR' THEN
            -- GDPR準拠レポート（個人データアクセス記録）
            SELECT
                'GDPR Compliance Report' as report_type,
                cal.username,
                cal.target_object,
                COUNT(*) as access_count,
                MIN(cal.event_timestamp) as first_access,
                MAX(cal.event_timestamp) as last_access,
                GROUP_CONCAT(DISTINCT cal.operation) as operations,
                JSON_ARRAYAGG(cal.affected_records) as accessed_records
            FROM comprehensive_audit_log cal
            WHERE cal.event_timestamp BETWEEN p_start_date AND
DATE_ADD(p_end_date, INTERVAL 1 DAY)
            AND cal.event_type IN ('DATA_ACCESS', 'DATA_MODIFY')
            AND cal.target_object IN ('students', 'grades', 'attendance')
            GROUP BY cal.username, cal.target_object
            ORDER BY access_count DESC;

        WHEN 'FERPA' THEN
            -- FERPA準拠レポート（教育記録アクセス）
            SELECT
                'FERPA Compliance Report' as report_type,
                cal.username,
                COUNT(*) as educational_record_accesses,
                AVG(cal.risk_score) as avg_risk_score,
                COUNT(CASE WHEN cal.risk_score > 50 THEN 1 END) as
high_risk_accesses
            FROM comprehensive_audit_log cal
            WHERE cal.event_timestamp BETWEEN p_start_date AND
DATE_ADD(p_end_date, INTERVAL 1 DAY)
            AND cal.target_object IN ('grades', 'attendance', 'students')
            GROUP BY cal.username
            ORDER BY educational_record_accesses DESC;

        WHEN 'SECURITY_SUMMARY' THEN
            -- セキュリティサマリーレポート
            SELECT
                'Security Summary Report' as report_type,
                cal.event_type,
                COUNT(*) as event_count,
                AVG(cal.risk_score) as avg_risk_score,
                COUNT(DISTINCT cal.username) as unique_users,
                MAX(cal.risk_score) as max_risk_score
            FROM comprehensive_audit_log cal
            WHERE cal.event_timestamp BETWEEN p_start_date AND
DATE_ADD(p_end_date, INTERVAL 1 DAY)
            GROUP BY cal.event_type
            ORDER BY avg_risk_score DESC;

        WHEN 'ACCESS_REVIEW' THEN
            -- アクセスレビューレポート
            SELECT
                'Access Review Report' as report_type,
                usc.username,
                usc.primary_role,
                usc.role_hierarchy_level,
                usc.security_clearance_level,
                access_summary.total_accesses,
                access_summary.high_risk_accesses,
                access_summary.last_access,
                CASE
                    WHEN access_summary.last_access < DATE_SUB(NOW(), INTERVAL 30
DAY) THEN 'INACTIVE'
                    WHEN access_summary.high_risk_accesses > 10 THEN 'HIGH_RISK'
                    ELSE 'NORMAL'
                END as access_status
            FROM unified_security_context usc
            LEFT JOIN (
                SELECT
                    cal.username,
                    COUNT(*) as total_accesses,
                    COUNT(CASE WHEN cal.risk_score > 70 THEN 1 END) as
high_risk_accesses,
                    MAX(cal.event_timestamp) as last_access
                FROM comprehensive_audit_log cal
                WHERE cal.event_timestamp BETWEEN p_start_date AND
DATE_ADD(p_end_date, INTERVAL 1 DAY)
                GROUP BY cal.username
            ) access_summary ON usc.username = access_summary.username
            WHERE usc.is_active = TRUE
            ORDER BY access_summary.high_risk_accesses DESC,
access_summary.total_accesses DESC;
    END CASE;

END //

DELIMITER ;

-- 8. セキュリティダッシュボード機能
CREATE VIEW v_security_dashboard AS
SELECT
    'Active Users' as metric_name,
    COUNT(*) as metric_value,
    'count' as metric_type
FROM unified_security_context
WHERE is_active = TRUE

UNION ALL

SELECT
    'High Risk Activities (Last 24h)',
    COUNT(*),
    'count'
FROM comprehensive_audit_log
WHERE event_timestamp >= DATE_SUB(NOW(), INTERVAL 24 HOUR)
AND risk_score > 70

UNION ALL

SELECT
    'Policy Violations (Last 7 days)',
    COUNT(*),
    'count'
FROM comprehensive_audit_log
WHERE event_timestamp >= DATE_SUB(NOW(), INTERVAL 7 DAY)
AND event_type = 'POLICY_VIOLATION'

UNION ALL

SELECT
    'Unique Data Accessors (Today)',
    COUNT(DISTINCT username),
    'count'
FROM comprehensive_audit_log
WHERE DATE(event_timestamp) = CURRENT_DATE
AND event_type = 'DATA_ACCESS';

-- 9. 自動化タスク
```

```
CREATE EVENT IF NOT EXISTS cleanup_expired_permissions
ON SCHEDULE EVERY 1 HOUR
DO
  UPDATE unified_security_context
  SET temporary_permissions = JSON_REMOVE(
      temporary_permissions,
      CONCAT('$."',
             (SELECT JSON_UNQUOTE(JSON_EXTRACT(JSON_KEYS(temporary_permissions),
'$[0]'))
              WHERE JSON_UNQUOTE(JSON_EXTRACT(temporary_permissions, CONCAT('$.',
JSON_UNQUOTE(JSON_EXTRACT(JSON_KEYS(temporary_permissions), '$[0]')),
'.expires_at'))) < NOW()
              LIMIT 1),
             '"')
  )
  WHERE temporary_permissions IS NOT NULL;

-- セキュリティダッシュボードの確認
SELECT * FROM v_security_dashboard;

-- テスト実行例
-- CALL setup_universal_row_level_security();
-- CALL generate_compliance_report('SECURITY_SUMMARY', '2025-05-01', '2025-05-
31');
-- CALL delegate_temporary_permission('principal@localhost',
'teacher_tanaka@localhost',
--      JSON_OBJECT('tables', JSON_ARRAY('students'), 'operations',
JSON_ARRAY('SELECT')), 24, 'Emergency access');
```

## まとめ

この章では、MySQLにおける行レベルセキュリティの実装について詳しく学びました：

1. **行レベルセキュリティの基本概念**：
   - テーブルレベルを超えた細粒度なアクセス制御
   - ビュー、プロシージャ、関数を活用した実装手法
   - セキュリティコンテキストとポリシーの管理
2. **ビューによる行レベル制御**：
   - 基本的なセキュリティビューの作成
   - 複雑な条件を持つ動的フィルタリング
   - ユーザーコンテキストに基づく制御
3. **ストアドプロシージャによる高度な制御**：
   - ユーザーコンテキスト管理システム
   - 安全なデータアクセスプロシージャ
   - 動的セキュリティ制御の実装
4. **実践的なセキュリティシステム**：

- 担任教師システム
- 保護者アクセスシステム
- 多階層権限管理

5. **パフォーマンスとセキュリティの両立**：

   - セキュリティビューの最適化
   - キャッシュ機能の実装
   - 監視システムの構築

6. **包括的なセキュリティフレームワーク**：

   - 統合セキュリティコンテキスト
   - 動的アクセスポリシー
   - 一時的権限委譲
   - 包括的な監査システム

行レベルセキュリティは、データベースセキュリティの最も細かいレベルでの制御を可能にします。**データの機密性を保ちつつ、業務効率を損なわない**バランスの取れた実装が重要です。また、セキュリティ制御が複雑になるほど、適切な監査とログ管理が不可欠になります。

次の章では、「データ暗号化：実用的なデータ保護手法」について学び、データそのものを保護する手法を理解していきます。

---