概要

学生成績管理システムの SQL 定義とサンプル操作です。スキーマ作成、テーブル DDL、外部キー、トランザクション例、サンプルデータ、代表的な SELECT を含みます。コメントは日本語で要点のみ記載します。

1. データベースとスキーマ作成

```
-- データベースとスキーマの作成
CREATE DATABASE student_management_system;
-- スキーマを分離（PostgreSQL 例）
CREATE SCHEMA student_mgmt AUTHORIZATION CURRENT_USER;
SET search_path TO student_mgmt;
```

2. テーブル作成（DDL）

```
-- 学生テーブル
CREATE TABLE student_mgmt.students (
student_id SERIAL PRIMARY KEY,
student_number VARCHAR(20) UNIQUE NOT NULL,
last_name VARCHAR(50) NOT NULL,
first_name VARCHAR(50) NOT NULL,
date_of_birth DATE NOT NULL,
email VARCHAR(100) UNIQUE NOT NULL,
enrollment_date DATE NOT NULL,
grade_level SMALLINT NOT NULL
);
-- 教員テーブル
CREATE TABLE student_mgmt.teachers (
teacher_id SERIAL PRIMARY KEY,
teacher_number VARCHAR(20) UNIQUE NOT NULL,
last_name VARCHAR(50) NOT NULL,
first_name VARCHAR(50) NOT NULL,
email VARCHAR(100) UNIQUE NOT NULL,
department VARCHAR(50) NOT NULL
);
-- 科目テーブル
CREATE TABLE student_mgmt.courses (
course_id SERIAL PRIMARY KEY,
course_code VARCHAR(20) UNIQUE NOT NULL,
course_name VARCHAR(100) NOT NULL,
credits SMALLINT NOT NULL,
teacher_id INT NOT NULL,
CONSTRAINT fk_courses_teacher
FOREIGN KEY (teacher_id) REFERENCES student_mgmt.teachers(teacher_id)
ON UPDATE CASCADE ON DELETE RESTRICT
);
-- 履修テーブル
CREATE TABLE student_mgmt.enrollments (
enrollment_id SERIAL PRIMARY KEY,
student_id INT NOT NULL,
course_id INT NOT NULL,
enrollment_date DATE NOT NULL,
semester VARCHAR(20) NOT NULL,
CONSTRAINT uq_enrollment UNIQUE(student_id, course_id, semester),
CONSTRAINT fk_enroll_student
FOREIGN KEY (student_id) REFERENCES student_mgmt.students(student_id)
ON UPDATE CASCADE ON DELETE CASCADE,
CONSTRAINT fk_enroll_course
FOREIGN KEY (course_id) REFERENCES student_mgmt.courses(course_id)
ON UPDATE CASCADE ON DELETE RESTRICT
);
-- 成績テーブル
CREATE TABLE student_mgmt.grades (
grade_id SERIAL PRIMARY KEY,
enrollment_id INT NOT NULL,
exam_type VARCHAR(20) NOT NULL, -- midterm/final/quiz 等
score NUMERIC(5,2) NOT NULL,
grade_letter VARCHAR(2),
exam_date DATE NOT NULL,
comments TEXT,
CONSTRAINT fk_grades_enroll
FOREIGN KEY (enrollment_id) REFERENCES
student_mgmt.enrollments(enrollment_id)
ON UPDATE CASCADE ON DELETE CASCADE
);
-- 出欠テーブル
CREATE TABLE student_mgmt.attendances (
attendance_id SERIAL PRIMARY KEY,
enrollment_id INT NOT NULL,
attendance_date DATE NOT NULL,
status VARCHAR(10) NOT NULL, -- present/absent/late
CONSTRAINT uq_att UNIQUE(enrollment_id, attendance_date),
CONSTRAINT fk_att_enroll
FOREIGN KEY (enrollment_id) REFERENCES
student_mgmt.enrollments(enrollment_id)
ON UPDATE CASCADE ON DELETE CASCADE
);
```

3. 外部キー制約

-- 主要な FK は DDL 内で定義済み。全て ON UPDATE CASCADE。削除時は履修に依存するものは CASCADE、科目は RESTRICT。

4. トランザクション例

```
-- 学生をコースに履修登録
BEGIN;
INSERT INTO
student_mgmt.enrollments(student_id,course_id,enrollment_date,semester)
VALUES (1,1,CURRENT_DATE,'2025S1');
COMMIT;
-- 失敗時
BEGIN;
-- まとめて登録
```

```
INSERT INTO
student_mgmt.enrollments(student_id,course_id,enrollment_date,semester)
VALUES (2,1,CURRENT_DATE,’2025S1’),(3,1,CURRENT_DATE,’2025S1’);
ROLLBACK;
-- 複数学生の成績登録
BEGIN;
INSERT INTO
student_mgmt.grades(enrollment_id,exam_type,score,grade_letter,exam_date,comments)
VALUES (1,’midterm’,78,’C’,’2025-06-10’,’’),(2,’midterm’,92,’A’,’2025-06-10’,’良’);
COMMIT;
-- 出欠更新
BEGIN;
UPDATE student_mgmt.attendances SET status=’late’
WHERE enrollment_id=1 AND attendance_date=’2025-06-11’;
COMMIT;
```

5. サンプルデータ

```
-- 学生 5 名
INSERT INTO
student_mgmt.students(student_number,last_name,first_name,date_of_birth,email,enrollment_date,grade_level) VALUES
(’S2025001’,’佐藤’,’太郎’,’2007-04-12’,’taro.sato@example.jp’,’2025-04-01’,1),
(’S2025002’,’鈴木’,’花子’,’2006-11-03’,’hanako.suzuki@example.jp’,’2025-04-01’,2),
(’S2025003’,’高橋’,’健’,’2007-02-20’,’ken.takahashi@example.jp’,’2025-04-01’,1),
(’S2025004’,’田中’,’愛’,’2006-07-15’,’ai.tanaka@example.jp’,’2025-04-01’,3),
(’S2025005’,’伊藤’,’翔’,’2007-09-08’,’sho.ito@example.jp’,’2025-04-01’,1);
-- 教員 3 名
INSERT INTO
student_mgmt.teachers(teacher_number,last_name,first_name,email,department)
VALUES
(’T001’,’山本’,’明’,’akira.yamamoto@example.jp’,’数学’),
(’T002’,’中村’,’恵’,’megumi.nakamura@example.jp’,’英語’),
(’T003’,’小林’,’直樹’,’naoki.kobayashi@example.jp’,’理科’);
-- 科目 4 件
```

```
INSERT INTO student_mgmt.courses(course_code,course_name,credits,teacher_id)
VALUES
(’MATH101’,’数学基礎’,2,1),(’ENG101’,’英語表現’,2,2),
(’SCI101’,’科学入門’,2,3),(’HIS101’,’歴史概論’,2,3);
-- 履修
INSERT INTO
student_mgmt.enrollments(student_id,course_id,enrollment_date,semester) VALUES
(1,1,’2025-04-02’,’2025S1’),(1,2,’2025-04-02’,’2025S1’),
(2,1,’2025-04-02’,’2025S1’),(3,3,’2025-04-03’,’2025S1’),
(4,4,’2025-04-03’,’2025S1’),(5,1,’2025-04-04’,’2025S1’);
-- 成績
INSERT INTO
student_mgmt.grades(enrollment_id,exam_type,score,grade_letter,exam_date,comment
s) VALUES
(1,’midterm’,85,’B’,’2025-06-10’,’’),(1,’final’,90,’A’,’2025-07-20’,’’),
(2,’quiz’,78,’C’,’2025-05-15’,’小テスト’),
(3,’midterm’,92,’A’,’2025-06-10’,’よい’),
(4,’final’,66,’D’,’2025-07-20’,’要改善’),
(5,’midterm’,88,’B’,’2025-06-10’,’’);
-- 出欠
INSERT INTO student_mgmt.attendances(enrollment_id,attendance_date,status)
VALUES
(1,’2025-04-05’,’present’),(1,’2025-04-06’,’absent’),
(2,’2025-04-05’,’present’),(3,’2025-04-05’,’late’),
(4,’2025-04-05’,’present’),(5,’2025-04-05’,’present’);
```

6. 有用な SELECT 例

```
-- 特定学生の全成績
SELECT c.course_name,g.exam_type,g.score,g.grade_letter
FROM student_mgmt.grades g
JOIN student_mgmt.enrollments e ON g.enrollment_id=e.enrollment_id
JOIN student_mgmt.courses c ON e.course_id=c.course_id
WHERE e.student_id=1
ORDER BY c.course_name, g.exam_date;
-- コース別平均点
```

```
SELECT c.course_name, AVG(g.score) AS avg_score
FROM student_mgmt.courses c
JOIN student_mgmt.enrollments e ON e.course_id=c.course_id
JOIN student_mgmt.grades g ON g.enrollment_id=e.enrollment_id
GROUP BY c.course_name;
-- 出席率一覧
SELECT s.student_number, s.last_name, s.first_name,
ROUND(100.0SUM(CASE WHEN a.status='present' THEN 1 ELSE 0
END)/COUNT(),1) AS attendance_rate
FROM student_mgmt.students s
JOIN student_mgmt.enrollments e ON e.student_id=s.student_id
JOIN student_mgmt.attendances a ON a.enrollment_id=e.enrollment_id
GROUP BY s.student_number, s.last_name, s.first_name;
-- 成績上位者
SELECT s.student_number,s.last_name,s.first_name, AVG(g.score) AS avg_score
FROM student_mgmt.students s
JOIN student_mgmt.enrollments e ON e.student_id=s.student_id
JOIN student_mgmt.grades g ON g.enrollment_id=e.enrollment_id
GROUP BY s.student_number,s.last_name,s.first_name
HAVING COUNT(*)>=2
ORDER BY avg_score DESC
LIMIT 5;
-- コース履修統計
SELECT c.course_name, COUNT(DISTINCT e.student_id) AS students,
COUNT(g.grade_id) AS grade_records
FROM student_mgmt.courses c
LEFT JOIN student_mgmt.enrollments e ON e.course_id=c.course_id
LEFT JOIN student_mgmt.grades g ON g.enrollment_id=e.enrollment_id
GROUP BY c.course_name;
```