Wittig School Student Management System Database Documentation

Author: Rodrigue Deguenon

Date: June 2025

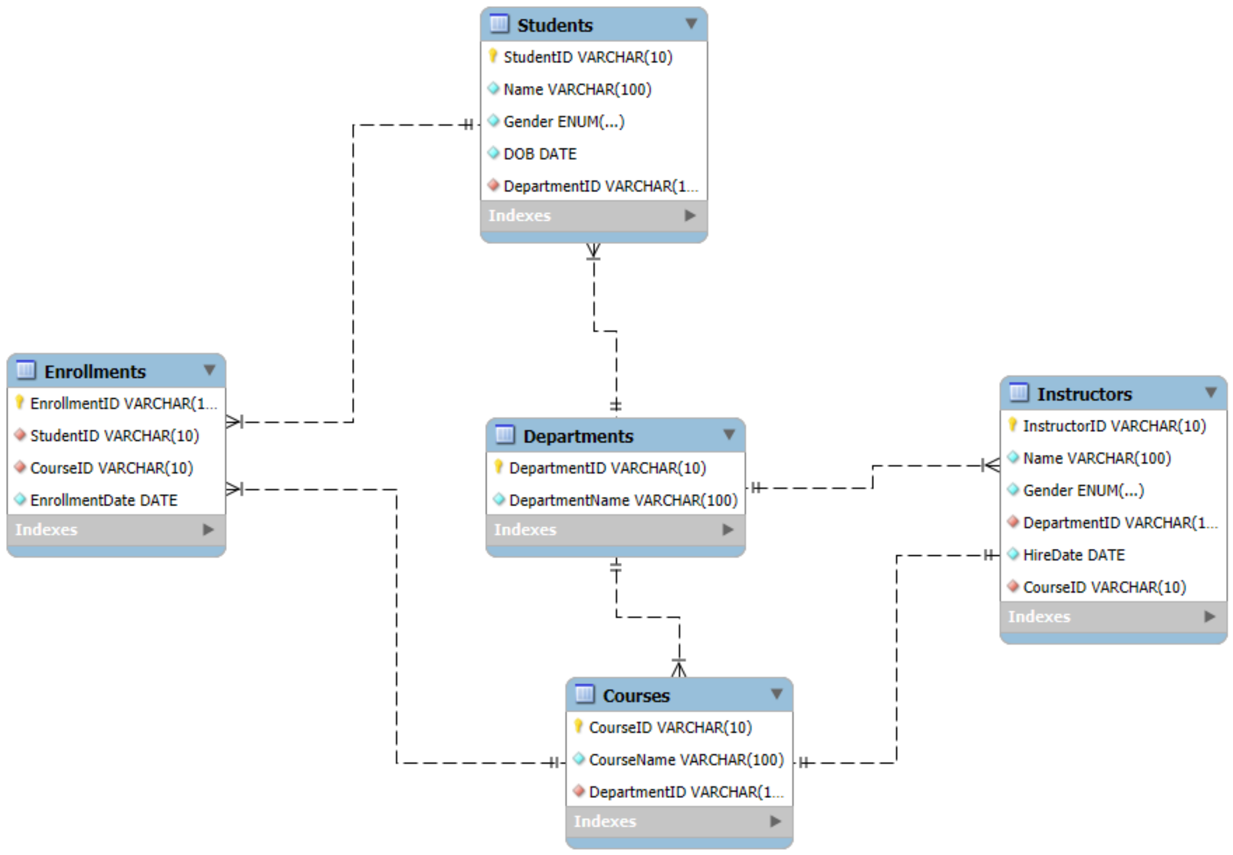
This document provides a comprehensive overview of the Wittig School Student Management System Database, including its structure, relationships, and query logic. The database is designed to manage student records, course enrollments, departmental affiliations, and instructor assignments efficiently.

Table of Contents

Database Overview

The Wittig School Student Management System Database tracks student academic journeys, including enrollment in courses, departmental affiliations, instructor assignments, and course offerings. The schema adheres to **3rd Normal Form (3NF)** and enforces referential integrity across all relationships.

Entity Relationship Diagram



The ER diagram visualizes the key entities (students, departments, instructors, courses, enrollments) and the relationships between them, primarily one-to-many and many-to-many relationships.

Sample data link: Google Sheets

Tables

departments

Column	Type	Constraints	Description
DepartmentID	VARCHAR(10)	PK, NOT NULL	Unique department identifier
DepartmentName	VARCHAR(100)	NOT NULL, UNIQUE	Official department name

instructors

Column	Type	Constraints	Description
InstructorID	VARCHAR(10)	PK, NOT NULL	Unique instructor ID
Name	VARCHAR(100)	NOT NULL	Instructor full name
Gender	ENUM('Male','Female','Other')	NOT NULL	Gender
DepartmentID	INT	FK to `departments`	Affiliated department
HireDate	DATE	NOT NULL	Date of hiring
CourseID	VARCHAR(10)	FK to `courses`, NULLABLE	Primary course taught

students

Column	Type	Constraints	Description
StudentID	VARCHAR(10)	PK, NOT NULL	Unique student ID
Name	VARCHAR(100)	NOT NULL	Full name
Gender	ENUM('Male','Female','Other')	NOT NULL	Gender
DOB	DATE	NOT NULL	Date of birth
DepartmentID	VARCHAR(10)	FK to `departments`	Student's major department

---

`courses`

Column	Type	Constraints	Description
CourseID	VARCHAR(10)	PK, NOT NULL	Unique course ID
CourseName	VARCHAR(100)	NOT NULL	Title of course
DepartmentID	VARCHAR(10)	FK to `departments`	Department offering course

---

`enrollments`

Column	Type	Constraints	Description
EnrollmentID	VARCHAR(10)	PK, NOT NULL	Unique enrollment ID
StudentID	VARCHAR(10)	FK to `students`, NOT NULL	Student being enrolled
CourseID	VARCHAR(10)	FK to `courses`, NOT NULL	Course student is enrolled in
EnrollmentDate	DATE	DEFAULT CURRENT_DATE, NOT NULL	Date of enrollment

---

Relationships

- **One-to-Many**:
  - A department can have many students, instructors, and courses.
- **Many-to-Many**:
  - Students can enroll in multiple courses (via `enrollments`)
  - Courses can have many students enrolled (via `enrollments`)
- **One-to-One**:
  - Each course is taught by one instructor.

Query Logic & Analysis

📊 Students & Enrollment Reports

1. ***Enrolled students per course***

This query shows the number of students enrolled in each course. It uses a `LEFT JOIN` to include all courses, even those with zero enrollments, and groups the results by course name.

```
SELECT c.CourseName, COUNT(e.StudentID) AS EnrolledStudent
FROM courses c
LEFT JOIN enrollments e ON c.CourseID = e.CourseID
GROUP BY c.CourseName;
```

2. ***Students enrolled in multiple courses and the list of courses***

This query identifies students enrolled in more than one course and lists all courses each such student is taking. It uses a common table expression (CTE) to filter students by course count, then aggregates course names for each student.

```
WITH multi_course_students AS (
  SELECT s.StudentID, s.Name, COUNT(e.CourseID) AS CourseCount
  FROM students s
  JOIN enrollments e ON s.StudentID = e.StudentID
  GROUP BY s.StudentID
  HAVING COUNT(e.CourseID) > 1
)
SELECT
    m.StudentID,
    m.Name,
    m.CourseCount,
    GROUP_CONCAT(c.CourseName SEPARATOR ', ') AS EnrolledCourses
FROM multi_course_students AS m
JOIN enrollments AS e ON m.StudentID = e.StudentID
JOIN courses c ON e.CourseID = c.CourseID
GROUP BY m.StudentID, m.Name, m.CourseCount
ORDER BY m.CourseCount DESC, m.Name;
```

*3. **Students per department***

This query counts the number of students in each department. It uses a `LEFT JOIN` to include departments with no students and groups by department.

```sql
SELECT
  d.DepartmentID,
  d.DepartmentName,
  COUNT(s.StudentID) AS StudentNumber
FROM departments d
LEFT JOIN students s ON d.DepartmentID = s.DepartmentID
GROUP BY d.DepartmentID, d.DepartmentName
ORDER BY StudentNumber DESC;
```

Course & Instructor Analysis

*1. **Top 5 courses by enrollment***

This query finds the five courses with the highest number of enrollments. It joins courses and enrollments, groups by course, and orders by enrollment count in descending order.

```sql
SELECT c.CourseID, c.CourseName, COUNT(e.EnrollmentID) AS EnrollmentCount
FROM courses c
JOIN enrollments e ON c.CourseID = e.CourseID
GROUP BY c.CourseID, c.CourseName
ORDER BY EnrollmentCount DESC
LIMIT 5;
```

*2. **Department with the fewest students***

This query identifies the department with the smallest student population. It joins departments and students, groups by department, and orders by student count ascending, returning only the top result.

```sql
SELECT d.DepartmentID, d.DepartmentName, COUNT(s.StudentID) AS StudentNumber
FROM departments d
JOIN students s ON d.DepartmentID = s.DepartmentID
GROUP BY d.DepartmentID, d.DepartmentName
ORDER BY StudentNumber ASC
LIMIT 1;
```

✓ Data Integrity & Operational Insights

*1. **Students not enrolled in any course***

This query lists students who are not enrolled in any course. It uses a `WHERE NOT EXISTS` clause to filter out students present in the enrollments table.

```
SELECT s.StudentID, s.Name
FROM students s
WHERE NOT EXISTS (
    SELECT 1
    FROM enrollments e
    WHERE e.StudentID = s.StudentID
);
```

*2. **Average number of courses per student***

This query calculates the average number of courses each student is enrolled in. It divides the total number of enrollments by the number of unique students.

```
SELECT
    COUNT(*) * 1.0 / COUNT(DISTINCT StudentID) AS AvgCoursesPerStudent
FROM enrollments;
```

*3. **Gender distribution across courses and instructors***

This query shows the gender breakdown of students in each course, along with the instructor for each course. It joins enrollments, students, courses, and instructors, and groups by course, instructor, and gender.

```
SELECT
    c.CourseID,
    c.CourseName,
    i.InstructorID,
    i.Name AS InstructorName,
    s.Gender,
    COUNT(*) AS StudentCount
FROM enrollments e
JOIN students s ON e.StudentID = s.StudentID
JOIN courses c ON e.CourseID = c.CourseID
JOIN instructors i ON i.InstructorID = c.InstructorID
GROUP BY c.CourseID, c.CourseName, i.InstructorID, i.Name, s.Gender
ORDER BY c.CourseID, s.Gender;
```

*4. **Course with highest male or female enrollment***

This query finds the course and gender combination with the highest number of enrolled students. It groups by course and gender, orders by student count descending, and returns the top result.

```sql
SELECT
    c.CourseID,
    c.CourseName,
    s.Gender,
    COUNT(*) AS StudentCount
FROM enrollments e
JOIN students s ON e.StudentID = s.StudentID
JOIN courses c ON e.CourseID = c.CourseID
GROUP BY c.CourseID, c.CourseName, s.Gender
ORDER BY StudentCount DESC
LIMIT 1;
```

Appendix

System Information

- **Database System**: MySQL 8.0+
- **Character Set**: `utf8mb4`
- **Collation**: `utf8mb4_unicode_ci`
- **Storage Engine**: InnoDB
- **Last Updated**: June 2025