Indexes Report

1. Index on student\_to\_course.course\_id

Feature Benefited: Classmates Filter

Queries:

SELECT student.id, student.name FROM student

JOIN student\_to\_course ON student.id = student\_to\_course.student\_id

WHERE student\_to\_course.course\_id = :course\_id

Justification: This index is crucial for quickly locating all student records associated with a particular course. The operation is read-intensive, especially in educational applications where filtering students by courses happens frequently. An index on course\_id in the student\_to\_course table accelerates this join operation by reducing the number of table scans required to match records.

2. Index on student.gender

Feature Benefited: Gender Distribution Report in Classmates Filter

Queries:

SELECT student.gender, COUNT(student.gender) AS gender\_count

FROM student

JOIN student\_to\_course ON student.id = student\_to\_course.student\_id

WHERE student\_to\_course.course\_id = :course\_id

GROUP BY student.gender

Justification: This index supports efficient aggregation and filtering of students by gender within a specified course. It's particularly useful for generating gender distribution reports, a common requirement in academic settings to analyze class composition. Indexing the gender column optimizes the group by operation and speeds up counting.

3. Primary Key Index on course.id and student.id

Feature Benefited: General Database Operations

Queries:

SELECT \* FROM course WHERE id = :course\_id

SELECT \* FROM student WHERE id = :student\_id

Justification: Primary keys are automatically indexed in most relational databases. These indexes are vital for all features involving direct lookups of courses or students by their IDs. They ensure quick access to specific records and are essential for maintaining the integrity and performance of the database.

4. Composite Index on student\_to\_course(student\_id, course\_id)

Feature Benefited: Common Courses Filter

Queries:

SELECT course.id, course.name FROM course

JOIN student\_to\_course ON course.id = student\_to\_course.course\_id

WHERE student\_to\_course.student\_id IN (:student\_ids)

GROUP BY course.id

Justification: A composite index on both student\_id and course\_id in the student\_to\_course table enhances performance for queries that need to filter or aggregate data based on multiple criteria across these columns. It's particularly useful for features that need to determine common courses among a group of students, reducing the I/O by allowing the database to quickly locate relevant rows.

Conclusion

Implementing the recommended indexes will enhance the performance of your application by reducing query response times and improving user experience during data-intensive operations. As your database grows in size, these indexes become even more critical in maintaining an efficient and responsive application.