School Strength

Description

Objective:

To work with One to Many Relationship between entities ,implement as Bidirectional and use of Entity Relationship Dependency.

Concept Explanation:

1. When one instance of an entity is associated with multiple instances of another entity, then we call that relationship **one-to-many.**
2. That relationship will be **bi-directional** if both the entities are aware of each other and can navigate from one entity to another.

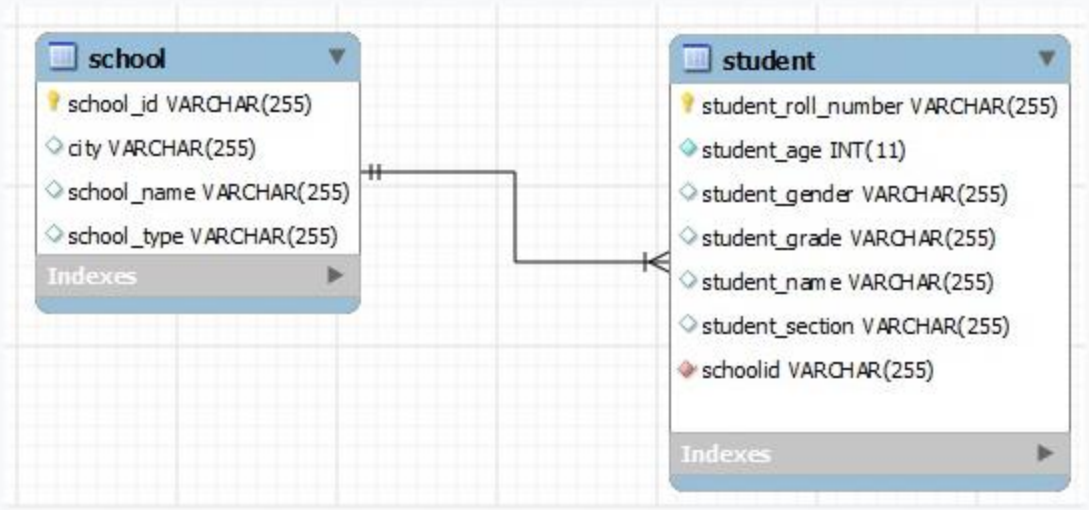
Concept Implementation:

1. We establish a one-to-many relationship between the School and Student entities by providing appropriate annotation above the studentList attribute in the School entity with the mappedBy attribute pointing to the relationship owner, School, thereby linking the two entities.
2. This ensures that changes made to one side of the relationship are reflected appropriately on the other side.
3. The Student entity holds a reference to the School entity, establishing a many-to-one relationship, where each student is associated with one school. Use of @JoinColumn annotation specifies the foreign key column in the Student table, linking each student to their respective school, enabling bidirectional retrieval of school and student details.

State education committee plans to conduct a survey. They want every student in the school to give feedback for improving the education system. As per the requirements for this report, they need to start the survey with the schools that have the highest number of student. Develop a Spring Data JPA application to store the details of Schools and Students and to manipulate them based on the city and the school strength.

Database Design:

Database Design:



Note:

- School and Student have one to many relationships.
- Establish the relationship between School and Student as Bi-directional.

Functionalities:

1. Add School

The state education committee will store the information of each school by providing the details like school id, school name, school type and city. This information will be stored in the School table.

Utility class:

Component Name	Method Name	Method Description	Arguments	Output
SchoolDAO	addSchool	This method stores the school details to the School table	School school	void

Model class

Component Name	Attributes	Methods
School	schoolId -String schoolName -String schoolType -String city -String For the student details - the attribute name should be "studentList"	getter and setter methods
Student	studentRollNumber - String studentName - String studentAge - int studentGender - String studentGrade - String studentSection - String For the school information- the attribute name should be "school".	getter and setter methods

2. Allocate a student to the school

To the already existing school names, the State education committee wants to register the students by providing the details of each student like student roll number, name, age, gender, grade and section.

Utility class:

Component Name	Method Name	Method Description	Arguments	Output
SchoolDAO	registerStudentToSchool	To the already existing school add the student. This method should add the student object to the Student table.	String schoolId, List<Student> student	void

3. Find School with maximum students

This functionality should list out the schools which have the highest students count in a particular city. (Hint: One or more state may have schools with an equally high number of students).

Utility class:

Component Name	Method Name	Method Description	Arguments	Output
SchoolDAO	schoolWithMaximumStudents	This method should return the schools which have the highest students count in a particular city.	String city	List<School>

Design Constraints

1. All the classes and methods should have public access Specifiers.
2. All the attributes should have private access Specifiers.
3. Use Annotation for all the mapping.
4. school_id and stud_roll_num should be marked as the primary key.

5. Use Spring data JPA API for persisting the object into the database.

6. The tables given below should get created by hibernate automatically with the proper parent and child relationship.

7. The table should be created with the correct table name and column name as specified in the database design diagram.

8. Do not alter the code skeleton and do not include any additional packages.

9. The column names and method name should be given as specified in the document.

10. Do not change the property name given in the application.properties files, you can change the value and you can include additional property if required.

11. **In the pom.xml we have included all the dependencies needed for developing the application.**