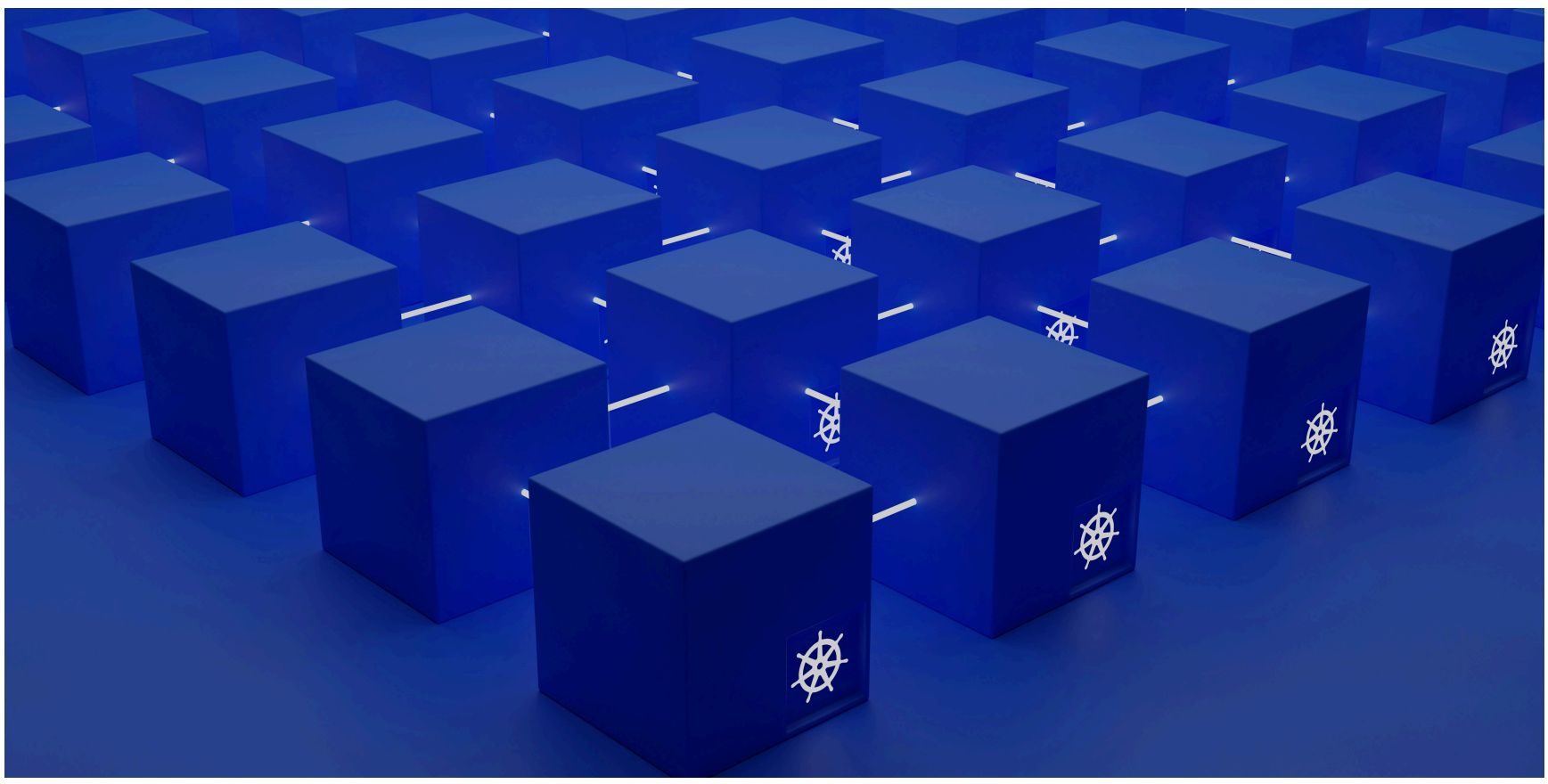


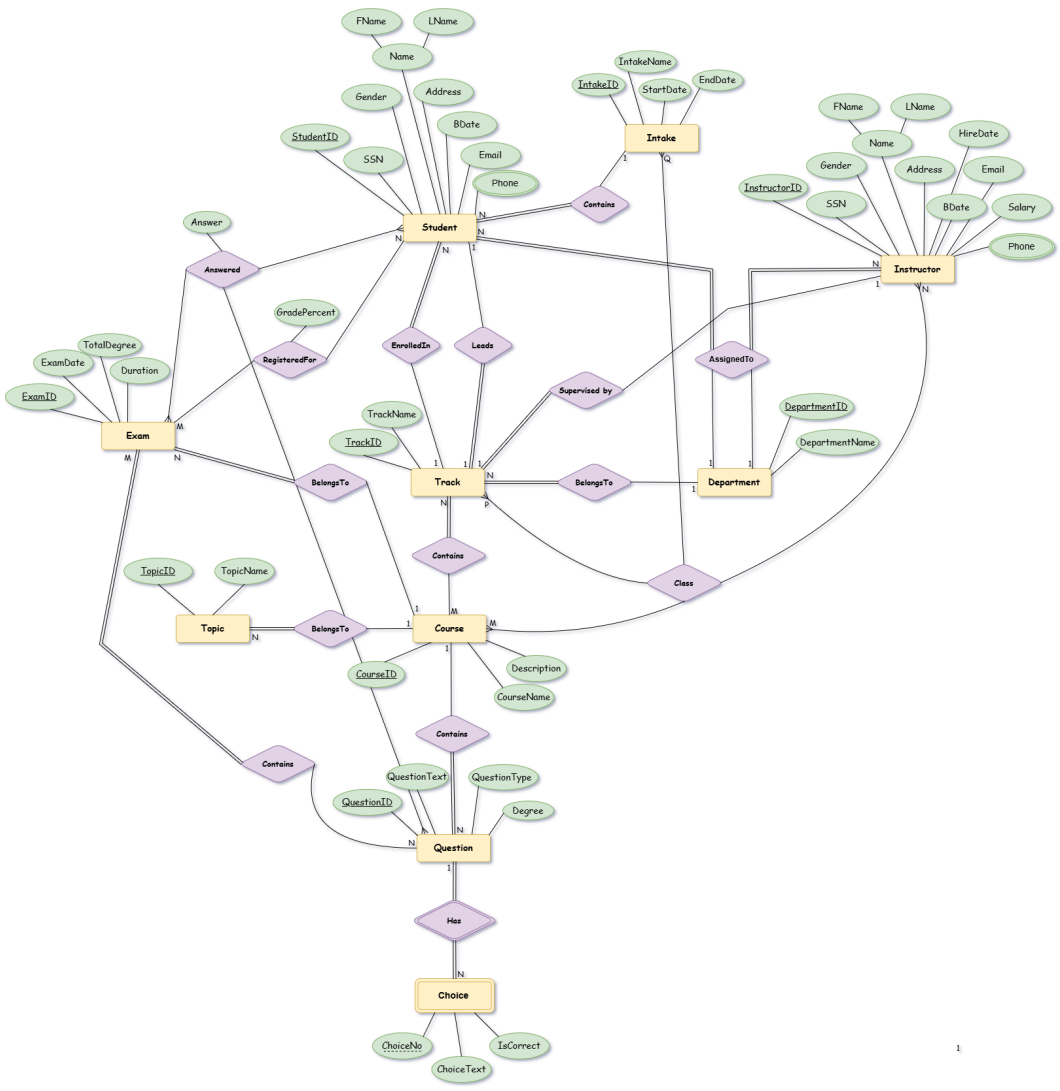
Documentation DB Project

Project Database Documentation

Overview

This database is designed to manage an educational system that includes departments, tracks, courses, instructors, students, exams, and evaluation processes. It supports academic structure, teaching assignments, exams, questions, student answers, and results.

ERD



ERD Description (Conceptual Explanation)

This Entity Relationship Diagram (ERD) models a complete academic and examination management system. The design clearly separates core academic entities, associative entities, and assessment-related entities to ensure clarity, scalability, and normalization.

Core Academic Entities

- **Department** is the top-level entity that represents academic departments. Each department can have multiple instructors, students, and tracks, but each instructor, student, and track belongs to exactly one department.
- **Track** represents a specialization within a department. Each track belongs to one department, may be supervised by one instructor, and may have one student acting as a track leader. A track can include multiple courses and students.
- **Course** represents an academic subject. Courses belong to multiple tracks and can be taught by different instructors across intakes. Each course consists of multiple topics and questions and can have multiple exams.
- **Intake** represents a time-bounded academic batch. Students enroll in exactly one intake, and instructors are assigned to teach courses within tracks for a specific intake.

Human Entities

- **Student** stores personal and academic enrollment data. A student belongs to one department, one track, and one intake. Students can register for exams, submit answers, and receive final grades.
- **Instructor** stores personal and employment data. Instructors belong to a department, may supervise tracks, and are assigned to teach courses for specific tracks and intakes.

Assessment Entities

- **Exam** represents an evaluation for a course. An exam contains multiple questions and can be taken by many students.
- **Question** represents an exam question related to a course. Questions may appear in multiple exams and contain multiple choices.
- **Choice** represents possible answers for multiple-choice questions, with one or more marked as correct.

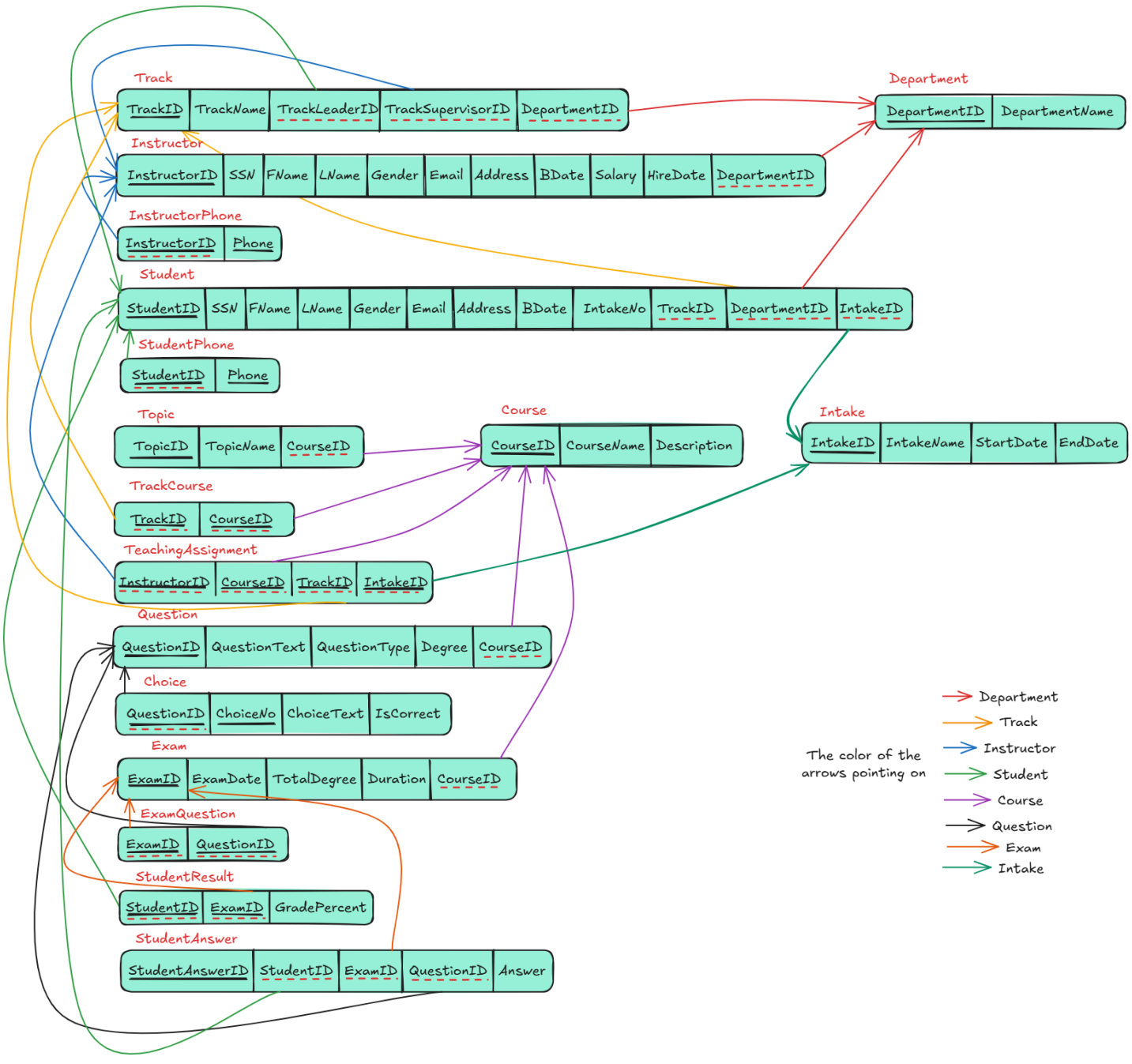
Associative (Bridge) Entities

- **TrackCourse** resolves the many-to-many relationship between tracks and courses.
- **Teaching** resolves the many-to-many relationship between instructors, courses, tracks, and intakes, defining who teaches what, where, and when.
- **ExamQuestion** resolves the many-to-many relationship between exams and questions.
- **StudentAnswer** records the answers submitted by students for each question in an exam.
- **StudentResult** stores the final calculated grade for each student per exam.

Design Rationale

- The ERD follows **Third Normal Form (3NF)** to eliminate redundancy.
- Associative entities are used to correctly model complex many-to-many relationships.
- Referential integrity is enforced using foreign keys.
- The design supports extensibility for future features such as multiple attempts, grading rules, or analytics.

Schema



Relational Schema Documentation (Database Schema)

This section describes the physical database schema, including tables, primary keys, foreign keys, and how relationships are implemented at the relational level.

---

1. Department

Stores academic departments.

- **DepartmentID** (PK)
- DepartmentName

Referenced by:

- Instructor.DepartmentID
- Student.DepartmentID
- Track.DepartmentID

---

2. Instructor

Stores instructor personal and employment data.

- **InstructorID** (PK)
- SSN (UNIQUE)
- FName
- LName
- Gender
- Email
- Address

- BDate
- Salary
- HireDate
- **DepartmentID** (FK → Department)

Relationships:

- One Instructor → Many InstructorPhone
- One Instructor → Many TeachingAssignment
- One Instructor → Many Tracks (as supervisor)

---

3. InstructorPhone

Stores instructor phone numbers.

- **InstructorID** (PK, FK → Instructor)
- **Phone** (PK)

Notes:

- Composite primary key
- Cascade delete on Instructor deletion

---

4. Student

Stores student personal and academic data.

- **StudentID** (PK)
- SSN (UNIQUE)
- FName
- LName
- Gender
- Email
- Address
- BDate
- IntakeNo
- **TrackID** (FK → Track)
- **DepartmentID** (FK → Department)
- **IntakeID** (FK → Intake)

Relationships:

- One Student → Many StudentPhone
- One Student → Many StudentAnswer
- One Student → Many StudentResult
- One Student → One Track (optional leader role)

---

5. StudentPhone

Stores student phone numbers.

- **StudentID** (PK, FK → Student)
- **Phone** (PK)

Notes:

- Composite primary key
- Cascade delete on Student deletion

---

6. Track

Represents academic tracks.

- **TrackID** (PK)
- TrackName
- **TrackLeaderID** (FK → Student)
- **TrackSupervisorID** (FK → Instructor)
- **DepartmentID** (FK → Department)

Relationships:

- One Track → Many Students
- One Track → Many TeachingAssignment
- Many Tracks ↔ Many Courses (via TrackCourse)

---

7. Course

Represents academic courses.

- **CourseID** (PK)
- CourseName
- Description

Relationships:

- One Course → Many Topics
- One Course → Many Questions
- One Course → Many Exams
- Many Courses ↔ Many Tracks (via TrackCourse)
- Many Courses ↔ Many Instructors (via TeachingAssignment)

---

8. Topic

Represents course topics.

- **TopicID** (PK)
- TopicName
- **CourseID** (FK → Course)

---

9. TrackCourse

Associative table between Track and Course.

- **TrackID** (PK, FK → Track)
- **CourseID** (PK, FK → Course)

Purpose:

- Implements many-to-many relationship between tracks and courses

---

10. Intake

Represents academic intakes.

- **IntakeID** (PK)
- IntakeName
- StartDate
- EndDate

Referenced by:

- Student.IntakeID
- TeachingAssignment.IntakeID

---

11. TeachingAssignment

Defines teaching assignments per intake.

- **InstructorID** (PK, FK → Instructor)
- **CourseID** (PK, FK → Course)
- **TrackID** (PK, FK → Track)
- **IntakeID** (PK, FK → Intake)

Purpose:

- Resolves many-to-many relationships between Instructor, Course, Track, and Intake

---

12. Question

Stores exam questions.

- **QuestionID** (PK)
- QuestionText
- QuestionType
- Degree
- **CourseID** (FK → Course)

Relationships:

- One Question → Many Choices
- One Question → Many StudentAnswers
- Many Questions ↔ Many Exams (via ExamQuestion)

---

13. Choice

Stores choices for MCQ questions.

- **QuestionID** (PK, FK → Question)
- **ChoiceNo** (PK)
- ChoiceText
- IsCorrect

---

14. Exam

Represents exams.

- **ExamID** (PK)
- ExamDate
- TotalDegree
- Duration
- **CourseID** (FK → Course)

Relationships:

- One Exam → Many ExamQuestions
- One Exam → Many StudentAnswers
- One Exam → Many StudentResults

---

15. ExamQuestion

Associative table between Exam and Question.

- **ExamID** (PK, FK → Exam)
- **QuestionID** (PK, FK → Question)

---

16. StudentAnswer

Stores student answers per question per exam.

- **StudentAnswerID** (PK)
- **StudentID** (FK → Student)
- **ExamID** (FK → Exam)
- **QuestionID** (FK → Question)
- Answer

---

17. StudentResult

Stores final exam results.

- **StudentID** (PK, FK → Student)
- **ExamID** (PK, FK → Exam)
- GradePercent

Schema Design Notes

- The schema follows **Third Normal Form (3NF)**.
- Composite primary keys are used in associative tables.
- Foreign keys enforce referential integrity.
- Junction tables correctly implement many-to-many relationships.
- The schema directly maps from the conceptual ERD to a physical relational model.

---

1. Department

Purpose: Stores academic departments.

Column	Type	Description
DepID	int (PK, Identity)	Unique identifier for department
DepName	nvarchar(100)	Department name

Relationships:

- One Department → Many Instructors
- One Department → Many Students
- One Department → Many Tracks

---

2. Course

Purpose: Stores courses offered by the institution.

Column	Type	Description
CourseID	int (PK, Identity)	Course identifier
CourseName	nvarchar(100)	Course name
Description	nvarchar(255)	Course description

Relationships:

- One Course → Many Topics
- One Course → Many Questions
- One Course → Many Exams

- Many Courses ↔ Many Tracks (via TrackCourse)

---

3. Instructor

Purpose: Stores instructor personal and employment data.

Column	Type	Description
InstructorID	int (PK, Identity)	Instructor identifier
SSN	char(11)	National ID (unique)
FName	nvarchar(50)	First name
LName	nvarchar(50)	Last name
Gender	varchar(1)	Gender
Email	varchar(50)	Email address
Address	varchar(100)	Address
BDate	date	Birth date
HireDate	date	Hiring date
Salary	money	Salary
DepID	int (FK)	Department

Relationships:

- One Instructor → Many Phones
- One Instructor → Many Teaching records
- Instructor may supervise a Track

---

4. InstructorPhone

Purpose: Stores multiple phone numbers per instructor.

Column	Type	Description
InstructorID	int (PK, FK)	Instructor
Phone	char(12) (PK)	Phone number

Notes: Cascade delete when instructor is removed.

---

5. Topic

Purpose: Represents course topics.

Column	Type	Description
TopicID	int (PK, Identity)	Topic identifier
TopicName	nvarchar(75)	Topic name
CourseID	int (FK)	Related course

---

6. Question

Purpose: Stores exam questions.

Column	Type	Description
QuestionID	int (PK, Identity)	Question identifier
QuestionText	nvarchar(MAX)	Question text
QuestionType	nvarchar(50)	MCQ / T-F
Degree	int	Question degree
CourseID	int (FK)	Related course

---

7. Choice

Purpose: Stores choices for MCQ questions.

Column	Type	Description
QuestionID	int (PK, FK)	Question
ChoiceNo	char(1) (PK)	Choice label (A,B,C)
ChoiceText	nvarchar(100)	Choice text
IsCorrect	bit	Correct answer flag

---

8. Exam

Purpose: Represents exams.

Column	Type	Description
ExamID	int (PK, Identity)	Exam identifier
ExamDate	date	Exam date
TotalDegree	int	Total marks
Duration	int	Duration (minutes)
CourseID	int (FK)	Related course

---

9. ExamQuestion

Purpose: Associates questions with exams.

Column	Type	Description
ExamID	int (PK, FK)	Exam
QuestionID	int (PK, FK)	Question

Relationship: Many-to-Many between Exam and Question

---

10. Track

Purpose: Represents academic tracks.

Column	Type	Description
TrackID	int (PK, Identity)	Track identifier
TrackName	nvarchar(100)	Track name
DepID	int (FK)	Department
TrackSuperID	int (FK)	Supervisor instructor
TrackLeaderID	int (FK)	Student leader

---

11. Student

Purpose: Stores student data.

Column	Type	Description
StudentID	int (PK, Identity)	Student identifier
FName	nvarchar(50)	First name
LName	nvarchar(50)	Last name
Gender	varchar(1)	Gender
Email	varchar(50)	Email
Address	varchar(100)	Address
BDate	date	Birth date
SSN	char(11)	National ID
IntakeID	int (FK)	Intake
DepID	int (FK)	Department
TrackID	int (FK)	Track

12. StudentPhone

Purpose: Stores student phone numbers.

Column	Type	Description
StudentID	int (PK, FK)	Student
Phone	char(12) (PK)	Phone number

13. StudentAnswer

Purpose: Stores student answers per question.

Column	Type	Description
StudentAnswerID	int (PK, Identity)	Answer ID
StudentID	int (FK)	Student
ExamID	int (FK)	Exam
QuestionID	int (FK)	Question
Answer	char(1)	Selected choice

14. StudentResult

Purpose: Stores final exam results.

Column	Type	Description
StudentID	int (PK, FK)	Student
ExamID	int (PK, FK)	Exam
GradePercent	decimal(5,2)	Final grade

15. TrackCourse

Purpose: Maps courses to tracks.

Column	Type	Description
TrackID	int (PK, FK)	Track
CourseID	int (PK, FK)	Course

16. Intake

Purpose: Represents academic intakes.

Column	Type	Description
IntakeID	int (PK, Identity)	Intake identifier
IntakeName	nvarchar(100)	Intake name
StartDate	date	Start date
EndDate	date	End date

17. Teaching

Purpose: Assigns instructors to courses per track and intake.

Column	Type	Description
InstructorID	int (PK, FK)	Instructor
CourseID	int (PK, FK)	Course
TrackID	int (PK, FK)	Track
IntakeID	int (PK, FK)	Intake

Relationship: Resolves many-to-many between Instructor, Course, Track, and Intake

Relationships (Logical View)

This section summarizes how tables are connected and the type of relationship between them.

Department Relationships

- **Department** → **Instructor** : One-to-Many
- **Department** → **Student** : One-to-Many
- **Department** → **Track** : One-to-Many

Course Relationships

- **Course** → **Topic** : One-to-Many
- **Course** → **Question** : One-to-Many
- **Course** → **Exam** : One-to-Many
- **Course** ↔ **Track** : Many-to-Many (via TrackCourse)
- **Course** ↔ **Instructor** : Many-to-Many (via Teaching)

Instructor Relationships

- **Instructor** → **InstructorPhone** : One-to-Many
- **Instructor** ↔ **Course** : Many-to-Many (via Teaching)
- **Instructor** → **Track (Supervisor)** : One-to-Many

Student Relationships

- **Student** → **StudentPhone** : One-to-Many
- **Student** ↔ **Exam** : Many-to-Many (via StudentAnswer & StudentResult)
- **Student** → **Department** : Many-to-One
- **Student** → **Track** : Many-to-One
- **Student** → **Intake** : Many-to-One
- **Student** → **Track (Leader)** : One-to-One (optional role)

Exam & Assessment Relationships

- **Exam** ↔ **Question** : Many-to-Many (via ExamQuestion)
- **Exam** → **StudentAnswer** : One-to-Many
- **Exam** → **StudentResult** : One-to-Many

Question & Answer Relationships

- **Question** → **Choice** : One-to-Many
- **Question** → **StudentAnswer** : One-to-Many

Track & Academic Structure

- **Track** ↔ **Course** : Many-to-Many (via TrackCourse)
- **Track** ↔ **Instructor** : Many-to-Many (via Teaching)
- **Track** → **Student** : One-to-Many

Intake Relationships

- **Intake** → **Student** : One-to-Many
- **Intake** ↔ **Instructor / Course / Track** : Many-to-Many (via Teaching)

Stored Procedures Documentation (CRUDs)

This section documents all stored procedures used in the system, describing their **purpose**, **parameters**, and **operations**.

Stored procedures are used to encapsulate business logic, ensure data integrity, and provide a secure and reusable interface to the database.

---

1. Student Stored Procedures

1.1 `sp_createStudent`

Purpose:

Creates a new student record in the `Student` table.

Parameters:

- `@FName` (nvarchar) – Student first name
- `@LName` (nvarchar) – Student last name
- `@Gender` (varchar) – Student gender
- `@Email` (varchar) – Student email
- `@Address` (varchar) – Student address
- `@BDate` (date) – Birth date
- `@IntakeID` (int) – Intake identifier
- `@DepID` (int) – Department identifier
- `@TrackID` (int) – Track identifier

Operation:

Inserts a new student record with the provided information.

1.2 `sp_readStudents`

Purpose:

Retrieves student data.

Parameters:

- `@StudentId` (int, optional)

Operation:

- If `@StudentId` is NULL → returns all students.
- Otherwise → returns the specified student.

1.3 `sp_updateStudent`

Purpose:

Updates student information.

Parameters:

- `@StudentId` (int)
- Optional fields: `@FName`, `@LName`, `@Gender`, `@Email`, `@Address`, `@BDate`, `@IntakeID`, `@DepID`, `@TrackID`

Operation:

- Validates student existence.
- Validates birth date.
- Updates only provided fields using `ISNULL`.

1.4 `sp_deleteStudent`

Purpose:

Deletes a student record.

Parameters:

- `@StudentId` (int)

Operation:

Deletes the student and cascades related phone records.

---

2. Exam Stored Procedures

2.1 `sp_createExam`

Purpose:

Creates a new exam.

Parameters:

- `@ExamDate` (date)
- `@duration` (int)
- `@CourseID` (int)

2.2 `sp_readExams`

Purpose:

Retrieves exam data.

Parameters:

- `@ExamID` (int, optional)

2.3 `sp_updateExam`

Purpose:

Updates exam details.

Parameters:

- `@ExamDate` (date)
- `@duration` (int)
- `@CourseID` (int)

2.4 `sp_deleteExam`

Purpose:

Validates existence of an exam before deletion.

Parameters:

- `@ExamID` (int)

---

3. Department Stored Procedures

3.1 `sp_createDepartment`

Purpose:

Creates a new department.

Parameters:

- `@DepName` (nvarchar)

3.2 `sp_readDepartments`

Purpose:

Retrieves department data.

Parameters:

- `@DepID` (int, optional)

3.3 `sp_updateDepartment`

Purpose:

Updates department name.

Parameters:

- `@DepID` (int)
- `@DepName` (nvarchar)

3.4 `sp_deleteDepartment`

Purpose:

Deletes a department.

Parameters:

- `@DepID` (int)

---

4. Course Stored Procedures

4.1 `sp_createCourse`

Purpose:

Creates a new course.

Parameters:

- `@CourseName` (nvarchar)
- `@Description` (nvarchar)

4.2 `sp_readCourses`

Purpose:

Retrieves courses.

Parameters:

- `@CourseID` (int, optional)

4.3 `sp_updateCourse`

Purpose:

Updates course details.

4.4 `sp_deleteCourse`

Purpose:

Deletes a course.

---

5. Question & Choice Stored Procedures

5.1 `sp_createQuestion`

Purpose:

Adds a new question to a course.

5.2 `sp_readQuestions`

Purpose:

Retrieves questions.

5.3 `sp_updateQuestion`

Purpose:

Updates question content and degree.

5.4 `sp_deleteQuestion`

Purpose:

Deletes a question.

5.5 `sp_createChoice`

Purpose:

Adds a choice to a question.

5.6 `sp_readChoices`

Purpose:

Retrieves all choices of a question.

5.7 `sp_updateChoice`

Purpose:

Updates choice text or correctness.

5.8 `sp_deleteChoice`

Purpose:

Deletes a choice.

---

6. Exam–Question Stored Procedures

6.1 `sp_addQuestionToExam`

Purpose:

Associates a question with an exam.

6.2 `sp_readExamQuestions`

Purpose:

Retrieves all questions assigned to an exam.

6.3 `sp_updateQuestionExam`

Purpose:

Replaces a question in an exam.

6.4 `sp_removeQuestionFromExam`

Purpose:

Removes a question from an exam.

---

7. Instructor Stored Procedures

7.1 `sp_createInstructor`

Purpose:

Creates a new instructor record.

7.2 `sp_readInstructor`

Purpose:

Retrieves instructor data.

7.3 `sp_updateInstructor`

Purpose:

Updates instructor details.

7.4 `sp_deleteInstructor`

Purpose:

Deletes an instructor.

---

8. Instructor Phone Stored Procedures

8.1 `sp_addInstructorPhone`

Purpose:

Adds a phone number to an instructor.

8.2 `sp_readInstructorPhones`

Purpose:

Retrieves instructor phone numbers.

8.3 `sp_updateInstructorPhone`

Purpose:

Updates an instructor phone number.

8.4 `sp_deleteInstructorPhone`

Purpose:

Deletes an instructor phone number.

---

9. Intake Stored Procedures

9.1 `sp_createIntake`

Purpose:

Creates a new intake.

9.2 `sp_readIntakes`

Purpose:

Retrieves intake data.

9.3 `sp_updateIntake`

Purpose:

Updates intake details.

9.4 `sp_deleteIntake`

Purpose:

Deletes an intake.

---

10. Track Stored Procedures

10.1 `sp_createTrack`

Purpose:

Creates a new academic track.

10.2 `sp_readTracks`

Purpose:

Retrieves track data.

10.3 `sp_updateTrack`

Purpose:

Updates track information.

10.4 `sp_deleteTrack`

Purpose:

Deletes a track.

---

11. Teaching Stored Procedures

11.1 `sp_createTeaching`

Purpose:

Assigns an instructor to teach a course in a track for a specific intake.

11.2 `sp_readTeaching`

Purpose:

Retrieves teaching assignments.

11.3 `sp_updateTeachingInstructor`

Purpose:

Changes the instructor for a teaching assignment.

11.4 `sp_deleteTeaching`

Purpose:

Removes a teaching assignment.

---

12. Track–Course Stored Procedures

12.1 `sp_addCourseToTrack`

Purpose:

Assigns a course to a track.

12.2 `sp_readTrackCourses`

Purpose:

Retrieves courses assigned to tracks.

12.3 `sp_updateTrackCourse`

Purpose:

Moves a course to another track.

12.4 `sp_deleteTrackCourse`

Purpose:

Removes a course from a track.

---

13. Reporting Procedure

`sp_getExamAnswer`

Purpose:

Retrieves correct answers for all questions in a specific exam.

Parameters:

- `@ExamID` (int)

Operation:

Returns question details along with correct choices for reporting and evaluation.

Main Stored Procedures

1️⃣ `sp_generateExam`

Purpose

Generate a new exam automatically for a specific **Track** and **Course** by randomly selecting questions from the question bank, then calculating and updating the total exam degree.

Parameters

Parameter	Type	Description
`@TrackID`	INT	Track for which the exam is generated
`@CourseID`	INT	Course related to the exam
`@NoOfTF`	INT	Number of True/False questions
`@NoOfMcq`	INT	Number of Multiple Choice questions
`@ExamDuration`	INT (Default = 90)	Exam duration in minutes

Procedure Logic

1. Starts a database transaction.
2. Validates that the course belongs to the specified track.
3. Ensures that no exam already exists for the same track and course.
4. Creates a new exam record in the `Exam` table.
5. Randomly selects:
    - `@NoOfTF` True/False questions.
    - `@NoOfMcq` MCQ questions.
6. Inserts the selected questions into the `ExamQuestion` table.
7. Calculates the total exam degree by summing question degrees.
8. Updates the `TotalDegree` field in the `Exam` table.
9. Commits the transaction if successful.
10. Rolls back the transaction if any error occurs.

Output / Effect

- New exam created.

- Exam questions linked automatically.
- Total exam degree calculated and stored.

2️⃣ `sp_StudentAnswer`

Purpose

Store or update a student's answer for a specific question during exam submission.

Parameters

Parameter	Type	Description
`@ExamID`	INT	Exam identifier
`@StudentID`	INT	Student submitting the answer
`@QuestionID`	INT	Question being answered
`@Answer`	CHAR(1)	Student selected answer (ChoiceNo)

Procedure Logic

1. Checks if the student has already answered the given question in the exam.
2. If no record exists:
   - Inserts a new row into `StudentAnswer`.
3. If a record exists:
   - Updates the existing answer.
4. Uses `SET NOCOUNT ON` to improve performance and avoid extra messages.

Output / Effect

- Student answers are safely stored.
- Allows modifying answers before final correction.

3️⃣ `sp_correctexam`

Purpose

Correct a submitted exam by calculating the student's score and updating their final grade percentage.

Parameters

Parameter	Type	Description
`@examid`	INT	Exam to be corrected
`@studentid`	INT	Student whose exam is corrected

Procedure Logic

1. Calculates the student's earned degree by:
   - Matching student answers with correct choices.
   - Summing the degrees of correctly answered questions.
2. Retrieves the total exam degree from the `Exam` table.
3. Calculates the grade percentage:

```
(StudentDegree / TotalDegree) *100
```

4. Updates the `StudentResult` table with the calculated percentage.

Report Stored Procedures

Schema: `Report`

1️⃣ `Report.StuInfo`

Purpose

Return information of all students in a specific **Department**.

Parameters

Parameter	Type	Description
`@DepID`	INT	Department ID to filter students

Procedure Logic

- Selects all columns from `student` table where `DepID = @DepID`.

Output

- Student list for the given department.

2️⃣ `Report.StuGrades`

Purpose

Return the grades (percentage) of a student in all courses they have taken.

Parameters

Parameter	Type	Description
`@StudentID`	INT	Student identifier

Procedure Logic

- Joins `StudentResult`, `Exam`, and `Course` tables.
- Retrieves course names and corresponding student grade percentages.

Output

Column	Description
`CourseName`	Name of the course
`GradePercent`	Student's grade in percentage

3️⃣ `Report.sp_InstructorCoursesWithStudents`

Purpose

Return courses taught by a specific instructor and the number of students enrolled in each course.

Parameters

Parameter	Type	Description
`@InstructorID`	INT	Instructor identifier

Procedure Logic

- Joins `TrackCourse`, `Course`, `Teaching`, and `Student`.
- Counts distinct students per course for the instructor.
- Groups results by course name.

Output

Column	Description
`CourseName`	Name of the course
`StudentNo`	Number of students enrolled

4️⃣ `Report.sp_CourseTopics`

Purpose

Return all topics associated with a specific course.

Parameters

Parameter	Type	Description
`@CourseID`	INT	Course identifier

Procedure Logic

- Selects `TopicName` from `Topic` table where `CourseID = @CourseID`.

Output

- List of topic names for the course.

5️⃣ `Report.GetExamQuestionsWithChoices`

Purpose

Return all questions and their choices for a given exam.

Parameters

Parameter	Type	Description
`@ExamID`	INT	Exam identifier

Procedure Logic

- Joins `Exam`, `ExamQuestion`, `Question`, and `Choice`.
- Retrieves question text, type, degree, and choices.
- Orders results by question ID and choice number.

Output

Column	Description
`ExamID`	Exam identifier
`QuestionID`	Question identifier
`QuestionText`	Text of the question
`QuestionType`	Type (MCQ, TF, etc.)
`Degree`	Points for the question
`ChoiceNo`	Choice identifier (A, B, C, D)

Column	Description
`ChoiceText`	Choice text
`IsCorrect`	Indicates correct choice

6️⃣ `Report.sp_QuestionsWithStudentAnswer`

Purpose

Return all questions in an exam with the **student's answers** and correct choice.

Parameters

Parameter	Type	Description
`@ExamID`	INT	Exam identifier
`@StudentID`	INT	Student identifier

Procedure Logic

- Joins `Question`, `StudentAnswer`, and `Choice` tables.
- Filters only the correct choice and the student's submitted answer.
- Returns question details and the student's answer.

Output

Column	Description
`QuestionText`	Text of the question
`Degree`	Question's degree
`CorrectChoice`	Correct choice number (A, B, ...)
`ChoiceText`	Text of the correct choice
`StudentAnswer`	Student's submitted answer

Tools & Technologies Used

- **Database & Querying**:
  - SQL Server (for database management and stored procedures)
  - T-SQL (for writing queries, stored procedures, and functions)
- **Reporting**:
  - SQL Server Reporting Services (SSRS) / Report Builder (for creating reports based on stored procedures)
- **Development Environment**:
  - Visual Studio (for C# and Windows Forms application development)
  - C# (programming language for business logic and UI)
  - Windows Forms (WinForms) for desktop application interface
- **Additional Tools / Technologies**:
  - Git (for version control, if used)
  - SSRS Report Designer (part of Visual Studio or Report Builder, for designing reports)
  - Draw.io

Links (REPO)

- https://github.com/mohamedfathidev/ExamSystem